Journal of Management and Service Science, 2023,
Vol. 03, Iss. 01, S. No. 022, pp. 1-14
ISSN (Online): 2583-1798

Fake News Detection Using Naive Bayes Classifier: A Comparative Study

Abhinandan Yadav[1], Devaraju Venkata Rao[2]

[1]Amity School of Engineering and Technology, Amity University Uttar Pradesh, Lucknow Campus, India
[2]Department of Accounting and Management, Global Institute of Management, Mangalpally, Tehsil Ibrahimpatnam, Telangana, India
[1]abnanyad999@gmail.com, [2]venkatdevraj1968@gmail.com

How to cite this paper: A. Yadav and D. V. Rao, "Fake News Detection Using Naive Bayes Classifier: A Comparative Study," *Journal of Management and Service Science (JMSS),* Vol. 03, Iss. 01, S. No. 022, pp. 1-14, 2023.

https://doi.org/10.54060/jmss.2023.22

Received: 14/03/2023
Accepted: 15/04/2023
Published: 25/04/2023

Abstract

Machine learning is a subfield of artificial intelligence (AI) and computer science that utilizes data and algorithms to imitate how people learn, progressively improving its accuracy. Machine learning is an important component of the growing field of data science. Through the use of statistical methods, algorithms are trained to make classifications or predictions, uncovering key insights. Detecting fake news comes under a classification problem. Fake news is false or misleading information presented as news. The initial stage in classification is dataset collection, which is followed by pre-processing, feature selection, dataset training and testing, and finally executing the classifier. There is a large amount of written text in the news. This text is processed using NLP. NLP can perform an intelligent analysis of large amounts of plain written text and generate insights from it. It involves methods like data pre-processing and feature selection. Data pre-processing involves data cleaning, removing any incorrect, duplicate, or incomplete data within a dataset. Feature selection is done using the CountVectorizer and TF-IDF Vectorizer. Then comes dataset training and testing and the use of similar data for training and testing reduces the impact of data inconsistencies. After processing the model using the training set, the model is tested by making predictions against the test set. Then, to assess the performance of the classification model for the provided set of test data confusion matrix is used. The primary purpose is to use the Naive Bayes (NB) Classifier technique to generate two classification models one using CountVectorizer and other using TF-IDF Vectorizer and compare their accuracy.

Keywords

Classification Problem, Confusion Matrix, CountVectorizer, Fake news, Naïve Bayes Classifier, TF-IDF Vectorizer

1. Introduction

Nowadays, the terms "machine learning" and "data science" are both used often. Although these two expressions are commonly used together in sentences, they are not synonymous. Even though data science uses machine learning, it is a broad field with a variety of tools.

Fundamentally, data science is a discipline of study that seeks to utilize a scientific method to derive insights and meaning from data. A special set of skills and knowledge are required to practice data science. Data science has gained attention as a significant emerging field of study and as a paradigm that is influencing the development of research in fields like statistics, computing science, and intelligence science as well as practical changes in fields like science, engineering, the government sector, business, sociology, and lifestyle. The broader fields of A.I., data analytics, machine learning, pattern recognition, and natural language understanding are all included in this discipline. Additionally, it addresses related new scientific problems, such as data collection, creation, storage, retrieval, sharing, analysis, optimization, and visualization, as well as integrative analysis across heterogeneous and interdependent complex resources for better collaboration, decision-making, and, ultimately, value creation.

In contrast, a range of methods employed by data scientists under the umbrella of Machine Learning (ML) enable computers to learn from data. Without programming explicit rules, these methods give promising results. The overarching objective of ML is to discover patterns in data that guide how previously overlooked issues are addressed. In a very complicated system, such as a self-driving vehicle, for example, massive volumes of data from sensors must be converted into decisions on how to operate the car by a computer that has "learned" to detect the patterns of "danger."

ML powers chatbots and predictive text, language translation tools, Netflix recommendations, and the way our social media posts are displayed. It drives autonomous cars, and robots that can identify medical issues using photos. Machine learning begins with data, which can be numbers, images, or text, such as bank transactions, photographs of people or even confectioneries, repair records, time series data from sensors, or sales figures. The data is gathered and made ready for use as training data, or information on which the ML model will be trained. The more data there is, the better the program will be. Although machine learning is included under data science, it is a large area with many distinct techniques. ML and AI have dominated areas of data science in recent years, playing significant roles in data analytics and business intelligence. Machine learning helps in automating the process of data analysis and goes on to create predictions based on massive volumes of data on specific populations. To do this, models and algorithms are created.

2. Literature Review

2.1 Fake News

Many terms with similar concepts and definitions were used to characterize information creditability, including trustworthiness, believability, dependability, correctness, equality, objectivity, and others.

Fake news is information that falsely claims to be news and sometimes contains sensitive messages. When the communications are received, they are quickly sent to others. In today's digital environment, the spread of fake news has its effects on more than one group. The mixing of realistic and unreal content on social media has created a false sense of reality. The advent of false news, however, poses a significant threat to the safety of people's lives and property. In the spread of fake news, there is misinformation (the distributer believes it is genuine) and disinformation (the distributer knows it is not true but actively hoaxes) [1].

In contrast, other scholars see fake news as a result of unintentional concerns such as educational shock or unintentional activities such as the one in 2020 when there was widespread fake news about health, putting world health at danger. The

WHO issued a warning in early February 2020 that the COVID-19 epidemic has resulted in a large 'infodemic', or a burst of genuine and fake news, including a lot of misinformation.

Hence there is a need for a technique to identify fake news. There are a variety of ML algorithms used for a variety of applications, but for a classification problem such as the detection of fake news requires the use of ML classifiers [2].

2.2 Types of ML Algorithms

ML, at its most basic, employs programmed algorithms that gather and analyze input data in order to anticipate output values within an acceptable level. As new data is fed into these algorithms, they learn and improve their processes to increase performance, gradually acquiring 'intelligence'. ML algorithms are classified as follows: supervised, semi-supervised, unsupervised, and reinforcement.

2.2.1 Supervised Learning

In this, machines are trained by examples. The manipulator gives the ML algorithm a known dataset with needed inputs and outputs, and it is up to the system to determine a function that converts those inputs into outputs. While the manipulator is aware of the correct solutions, the algorithm finds patterns in data, learns from observations, and makes predictions. The algorithm creates predictions, which are then adjusted by the operator, and up till the algorithm performs to a high level, the process is repeated. Classification, regression, and forecasting, all fall under supervised learning. Fake news detection is a type of classification problem wherein we have to classify whether the news is Real or Fake [3].

2.2.2 Semi-supervised Learning

Semi-supervised learning is comparable to supervised learning just in this it employs both labelled and unlabeled data. Labeled data is information that includes meaningful labels so that the algorithm can comprehend it, but data without labels lacks that information. ML systems can learn to label unlabeled data using this technique.

2.2.3 Unsupervised Learning

The method of presuming underlying hidden patterns from previous data is known as unsupervised machine learning. A ML model in this technique attempts to detect any differences, patterns, structure, and similarities in data on its own. There is no requirement for prior human involvement. However, it may provide less accurate results [4].

2.2.4 Reinforcement Learning

It aims on a structured learning process that generates a collection of actions, parameters, and end values for ML algorithms. ML algorithms investigate several alternatives and possibilities by setting rules, monitoring, and evaluating each output, and attempting to discover which one is best for you. Reinforcement learning instructs machines through trial and error. It draws on prior experience to modify its approach to the circumstance in order to produce the best potential results.

2.3 Classification algorithm

Classification algorithms are used to identify new observations' categories based on training data. In classification, a program learns using the supplied dataset or observation and classifies the new observation into several classes or groups.

Various studies have indicated that Naive Bayes Classifier is fast and gives better accuracy than some of the other classification algorithms, therefore, we will be using Naive Bayes Classifier for the classification model [5].

2.3.1 Naive Bayes Classifier

It is a well-known classification approach based on Bayes theorem. Naive Bayes is a group of probabilistic algorithms that calculates the probability that every given data point will fall into one or the other or more of the categories (or not). Simply expressed, Naive Bayes supposes that a function in one category has no bearing on another. For example, a fruit is defined as an orange if it is orange in color, has spirals, and has a diameter of less than 3 inches. If these functions are based on one another or on distinct functions, and that even if they are dependent on both one another and other functions, Naive Bayes presumes that each of these functions do have their own proof of the oranges.

This algorithm computes the probability of all the tags (label or category) in a particular text and then the highest probability's output:

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)} \quad (1)$$

That is, A's probability if B is true is said to be equal to the probability of B if A is true multiplied by A's probability (true) divided by the probability of B (true). Moving from category to category, the probability of whether a data point belongs to a particular category is calculated: Yes/No.

In this paper, the type of Naive Bayes classifier to be used is Multinomial Naive Bayes. The rates by which some events were generated by a multinomial distribution are represented by feature vectors. This is the event model commonly used for document classification [6-7]. Despite its ease of use, the classifier performs well and is frequently used because it outperforms more advanced classification algorithms.

3. Methodology

This part illuminates the methodology for the classification. Using this method, a model for identifying fake articles is developed. The news is classified as real or fake using supervised ML. The dataset collecting phase is the initial stage in the classification, followed by preprocessing, feature selection, dataset training and testing, and lastly running the Naive Bayes classifier. Two methods for feature selection are used, one being CountVectorizer and the other TF-IDF Vectorizer. The classifier is run on both the vectorizers separately and their accuracy is compared to see which one performed better. The main aim is to compare the accuracy of two classification models generated by the Naive Bayes Classifier algorithm, one using CountVectorizer and the other using TF-IDF Vectorizer.

4. Results and Discussion

4.1. Making necessary imports

```
[1]: import numpy as np
     import pandas as pd
     from sklearn.model_selection import train_test_split
     from sklearn.feature_extraction.text import TfidfVectorizer
     from sklearn.naive_bayes import MultinomialNB
     from sklearn import metrics
     from sklearn.metrics import accuracy_score, confusion_matrix
```

Figure 1. Making necessary imports

numpy is a well-known Python library for large multi-dimensional array and matrix processing using a large collection of high-level mathematical functions and panda is built on top of numpy and it is applied when we need to manipulate or analyze data. Scikit-learn is the most productive and robust library in Python. It offers easy-going and effective tools for predictive data analysis. It is built on NumPy, SciPy, and matplotlib. Naive Bayes Classifier is implemented using this Scikit-learn Python library.

4.2 Loading the Dataset

```
[2]: df=pd.read_csv('news.csv')

[3]: df.shape

[3]: (6335, 4)

[4]: df.head()
```

[4]:

	Unnamed: 0	title	text	label
0	8476	You Can Smell Hillary's Fear	Daniel Greenfield, a Shillman Journalism Fello...	FAKE
1	10294	Watch The Exact Moment Paul Ryan Committed Pol...	Google Pinterest Digg Linkedin Reddit Stumbleu...	FAKE
2	3608	Kerry to go to Paris in gesture of sympathy	U.S. Secretary of State John F. Kerry said Mon...	REAL
3	10142	Bernie supporters on Twitter erupt in anger ag...	— Kaydee King (@KaydeeKing) November 9, 2016 T...	FAKE
4	875	The Battle of New York: Why This Primary Matters	It's primary day in New York and front-runners...	REAL

Figure 2. Loading the Dataset

This dataset was published for research purposes on Kaggle. We are using pandas to load the data (Figure 2). We will also use pandas to explore the dataset with informative statistics like using the head () function to get a peek at the dataset. The dataset that will be used in this Python project is named 'news.csv'. This dataset has a shape of 6335×4. The first column recognizes the news, the second and third contain the title and text, and the fourth column contains a label indicating whether the message is REAL or FAKE.

4.3 Natural Language Processing (NLP)

NLP is an AI field that enables machines to read, comprehend, and infer meaning from human languages. This is an area focused on the interaction of data science and human language. NLP can assist you with a broad range of tasks, and the domains of application appear to be expanding on a regular basis. Some examples can be prediction of diseases, classifying spam and genuine e-mails, voice driven interfaces, etc.

One of the key parts of Natural Language Processing is Data Pre-processing**.** Data pre-processing is the most difficult and time-consuming aspect in data science, but it is also one of the very critical. Failure to clean and prepare the data may adversely impact the model. In the ideal world, the dataset is perfectly fine. But, in the real-world there are always some issues in the data like missing data, typos, etc. that need to be addressed. Such issues are required to be adjusted so as to make the data more useful and understandable [8-10]. Thus, the data pre-processing step is where we clean and resolve the majority of the problems in the data.

4.3.1 Data Cleaning

This method identifies incomplete, incorrect, duplicated, irrelevant, or null values in data. One must either change or eliminate these issues after recognizing them (as shown in Figure 3). Let's look at some of the most typical problems:

Noisy Data

Noisy data in your dataset refers to useless data, inaccurate records, or redundant observations. When an observation doesn't make any sense, we can either set its value as null or delete it.

Missing Data

Another typical difficulty with real-world data is the lack of data points. Most ML models cannot manage missing values in data; thus we must intervene and alter the data before it can be used effectively inside the model.

Stop Words

They are a collection of the most usual words in a language, such as "a," "be," "should," "quite," and so on. They are frequently meaningless and offer nothing to the material. They are also most commonly seen in all texts. As a result, we assumed that removing stop words would have several benefits [11-12]. For starters, it reduces memory overhead because we reduced a large quantity of text (and hence narrows down a number of features to train our models on). Second, by deleting stop words, we can focus on more relevant contents (the greater distinguishing features between these two classes).

```
[5]: df.drop_duplicates(inplace= True)
     df.shape

[5]: (6335, 4)

[6]: df.isnull().sum()

[6]: Unnamed: 0    0
     title         0
     text          0
     label         0
     dtype: int64
```

Figure 3. Data Cleaning

4.4 Splitting the dataset

Let's divide this dataset into labels and features (as shown in Figure 4). We predict labels using these features.

```
[7]: y=df.label
     y.head()

[7]: 0    FAKE
     1    FAKE
     2    REAL
     3    FAKE
     4    REAL
     Name: label, dtype: object

[8]: x=df.text
     x.head()

[8]: 0    Daniel Greenfield, a Shillman Journalism Fello...
     1    Google Pinterest Digg Linkedin Reddit Stumbleu...
     2    U.S. Secretary of State John F. Kerry said Mon...
     3    — Kaydee King (@KaydeeKing) November 9, 2016 T...
     4    It's primary day in New York and front-runners...
     Name: text, dtype: object
```

Figure 4. Dividing the dataset into labels and features

Dividing the dataset into two parts (as shown in Figure 5): training set and testing set, using the train_test_split() function of the sklearn.model_selection package.

```
[9]: x_train, x_test, y_train, y_test=train_test_split(x, y, test_size=0.2, random_state=7)
```

Figure 5. Splitting the dataset

The line test_size=0.2 suggests that test data should contain 20% of the dataset, with the remaining being train data. train_test_split chooses train and test sizes at random based on the provided ratio. Every time we execute this function, train and test values will be chosen at random based on the train and test size ratio. This random pick results in " random_states" every time we run this. To prevent receiving different results for the train and test each time we select a random_state. The most commonly used values are 0 and 1. We are free to choose your own value. Suppose it's shuffling of cards, with the first shuffle being random state 1, the second shuffle being random state 2, and so on.

4.5 Feature Generation

Text data is utilized to create a variety of features such as word count, frequency of unique words, frequency of large words, n-grams, and so on. We can allow computers to read text and perform classification by generating a representation of words that captures their meanings, semantic connections, and the many sorts of context in which they are used [13].

4.5.1 Vectorizing Data

Vectorizing means encoding text as integers, or numeric values, to make feature vectors that ML algorithms can interpret. It is required as computers cannot directly understand the text data.

4.5.2 Count Vectorizer

One of the simplest and most effective methods is the count vectorizer. It determines a word's weight by counting how many times it appears in a document. The Count Vectorizer method tokenizes the text documents and creates a word vocabulary [14]. While TF-IDF Vectorizer gives words a score and so produces floats, Count Vectorizer counts words and thus returns integers (as shown in Figure 6).

Data = ['Newspapers', 'give', 'the', 'news', 'of', 'the', 'world']

	Newspapers	give	the	news	of	world
Data	1	1	2	1	1	1

Figure 6. CountVectorizer representation

4.5.3 TF-IDF Vectorizer

It computes the "relative frequency" of a term appearing in a document in comparison to its frequency in all documents. The TF-IDF weight shows a term's relative significance in the document and overall corpus.

TF (Term Frequency) represents the number of times a word appears in a given document. As a result, it is document specific. TF is calculated in the following way:

$$TF(t,d) = \frac{\text{No. of times term 't' occurs in a document}}{\text{Total No. of terms in a document}} \quad (2)$$

IDF (Inverse Document Frequency) measures how frequent or uncommon a word is over the whole corpus of documents. The important thing to remember is that it appears in all of the documents. If the term is common and appears in numerous documents, the idf value (normalised) will be close to 0; otherwise, it will be close to 1 if it's uncommon or rare. IDF is calculated in the following way:

$$IDF(t,d) = \log\left(\frac{\text{Total number of documents}}{\text{Number of documents in which the term 't' appears}}\right) \quad (3)$$

where, t = the term for which the idf value is being calculated

TF-IDF is applied to the main text, so in the document matrix, the relative count of each word in the sentence is stored.

$$TFIDF(t,d) = TF(t,d) * IDF(t) \quad (4)$$

4.5.4 Feature Extraction of text data using CountVectorizer

Initializing a CountVectorizer and then on the train set, fit and transform the vectorizer then on the test set, transform it (as shown in Figure 7).

```
In [29]: tf_vectorizer=CountVectorizer()

In [30]: x_train_tf=tf_vectorizer.fit_transform(x_train)
         x_test_tf=tf_vectorizer.transform(x_test)

In [31]: print("count_train=> n_samples: %d, n_features: %d" %x_train_tf.shape)
         print("count_test=> n_samples: %d, n_features: %d" %x_test_tf.shape)

         count_train=> n_samples: 5068, n_features: 61958
         count_test=> n_samples: 1267, n_features: 61958
```

Figure 7. Feature Extraction using CountVectorizer

4.5.5 Feature Extraction of text data using TfidfVectorizer

Initializing a TfidfVectorizer with stop words (English) and a maximum df (document frequency) of 0.7 (words with a greater rate of occurrence in the document will be ignored) [15].

```
[10]: tfidf_vectorizer=TfidfVectorizer(stop_words='english', max_df=0.7)
```

Figure 8. Feature Extraction using TfidfVectorizer

Next, on the train set, fit and transform the vectorizer then on the test set, transform it (as shown in Figure 9).

```
[11]: tfidf_train=tfidf_vectorizer.fit_transform(x_train)
      tfidf_test=tfidf_vectorizer.transform(x_test)

[12]: print("tfidf_train=> n_samples: %d, n_features: %d" %tfidf_train.shape)
      print("tfidf_test=> n_samples: %d, n_features: %d" %tfidf_test.shape)

      tfidf_train=> n_samples: 5068, n_features: 61651
      tfidf_test=> n_samples: 1267, n_features: 61651
```

Figure 9. Fitting and transforming the TfidfVectorizer

The values of x_train parameter are calculated by the fit() method. The transform function applies the values to the existing actual data and returns the normalised value. The fit_transform() function does both in one step.

4.6 Building and Evaluating the Model

Now, for CountVectorizer, we will initialize a Naive Bayes Classifier- MultinomialNB() and fit this model on x_train_tf, y_train [16-17].

```
In [32]: naive_bayes_classifier=MultinomialNB()
         naive_bayes_classifier.fit(x_train_tf, y_train)

Out[32]: MultinomialNB()
```

Figure 10. Initializing Naive Bayes Classifier for CountVectorizer

Now, we will do the same for TfidfVectorizer and fit the MultinomialNB() model on tfidf_train, y_train.

```
[13]: naive_bayes_classifier=MultinomialNB()
      naive_bayes_classifier.fit(tfidf_train, y_train)

[13]: MultinomialNB()
```

Figure 11. Initializing Naive Bayes Classifier for TfidfVectorizer

Following that, we will anticipate on the test set, ' x_test_tf', using the CountVectorizer, and compute the accuracy using accuracy score(), as shown in Figure 12.

```
In [33]: y_pred=naive_bayes_classifier.predict(x_test_tf)

In [35]: score=metrics.accuracy_score(y_test, y_pred)
         print(f'Accuracy: {round(score*100,2)}%')
         print(metrics.classification_report(y_test, y_pred, target_names=["Positive", "Negative"]))

         Accuracy: 88.32%
                       precision    recall  f1-score   support

             Positive       0.92      0.84      0.88       638
             Negative       0.85      0.93      0.89       629

             accuracy                           0.88      1267
            macro avg       0.89      0.88      0.88      1267
         weighted avg       0.89      0.88      0.88      1267
```

Figure 12. Calculating the accuracy score for CountVectorizer

Similarly, we will anticipate on the test set, 'tfidf_test', using the TfidfVectorizer, and compute the accuracy using accuracy score(), as shown in Figure 13.

```
[14]: y_pred=naive_bayes_classifier.predict(tfidf_test)

[15]: score=metrics.accuracy_score(y_test, y_pred)
      print(f'Accuracy: {round(score*100,2)}%')
      print(metrics.classification_report(y_test, y_pred, target_names=["Positive", "Negative"]))

Accuracy: 84.06%
              precision    recall  f1-score   support

    Positive       0.97      0.71      0.82       638
    Negative       0.77      0.98      0.86       629

    accuracy                           0.84      1267
   macro avg       0.87      0.84      0.84      1267
weighted avg       0.87      0.84      0.84      1267
```

Figure 13. Calculating the accuracy score for TfidfVectorizer

4.6.1 Confusion Matrix

The confusion matrix is a matrix that is used to assess the performance of classification models for a provided set of test data. It can be determined only if the true values of the test data are known. The matrix itself is simple to grasp, but the associated terminologies can be perplexing [18]. Because it displays the mistakes in the model performance as a matrix, it is also known as an error matrix.

True Positive (TP)

The actual result was positive, and the model predicted that it would be positive.

True Negative (TN)

The actual result was negative, as predicted by the model.

Type 1 error – False Positive (FP)

The model predicted a positive result, but the actual value was negative.

Type 2 error – False Negative (FN)

The actual result was negative, whereas the model projected a positive value.

The precision, recall, F-measure, and accuracy of the classifier are calculated by these formulas respectively

$$Precision = \frac{True\ Positive}{(\text{True Positive } + \text{ False Positive})} \tag{5}$$

$$Recall = \frac{True\ Positive}{(\text{True Positive } + \text{ False Negative})} \tag{6}$$

$$F - measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{7}$$

$$Accuracy = \frac{\text{True Positive} + \text{True Negative}}{(\text{True Positive} + \text{True Negative} \ + \ \text{False Positive} + \text{False Negative})} \tag{8}$$

Now, we will create a confusion matrix and print it to see how many false and true negatives and positives are there (as shown in Figures 14 and 15 and Figures 16 and 17) [19-20].

For CountVectorizer

```
In [36]: print("Confusion Matrix:")
         conf_matrix=metrics.confusion_matrix(y_test, y_pred)
         print(conf_matrix)

         Confusion Matrix:
         [[535 103]
          [ 45 584]]
```

Figure 14. Printing the Confusion Matrix for CountVectorizer

Pictorial representation of the confusion matrix

```
In [37]: from mlxtend.plotting import plot_confusion_matrix
         import matplotlib.pyplot as plt

         fig, ax = plot_confusion_matrix(conf_mat=conf_matrix, figsize=(6, 6), cmap=plt.cm.Greens)
         plt.xlabel('Predictions', fontsize=18)
         plt.ylabel('Actuals', fontsize=18)
         plt.title('Confusion Matrix', fontsize=18)
         plt.show()
```

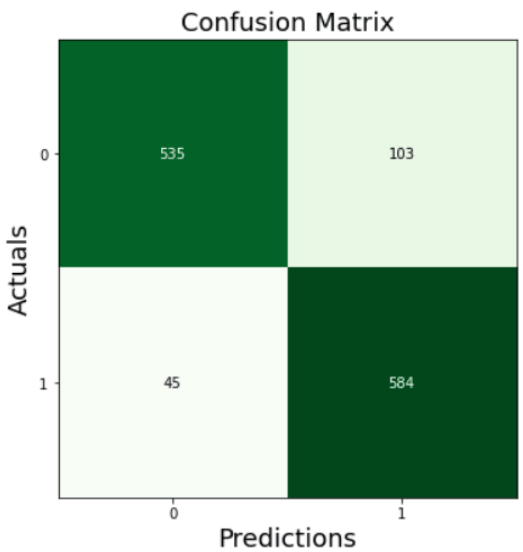

Figure 15. Confusion Matrix for CountVectorizer

We acquired an accuracy of 88.32% with the CountVectorizer NB model. And from the Confusion Matrix we have 584 true positives, 535 true negatives, 103 false positives, and 45 false negatives.

For TfidfVectorizer

```
[16]: print("Confusion Matrix:")
      conf_matrix=metrics.confusion_matrix(y_test, y_pred)
      print(conf_matrix)

      Confusion Matrix:
      [[450 188]
       [ 14 615]]
```

Figure 16. Printing the Confusion Matrix for TfidfVectorizer

Pictorial representation of the confusion matrix

```
[17]: from mlxtend.plotting import plot_confusion_matrix
      import matplotlib.pyplot as plt

      fig, ax = plot_confusion_matrix(conf_matrix, figsize=(4, 4), cmap=plt.cm.Greens)
      plt.xlabel('Predictions', fontsize=14)
      plt.ylabel('Actuals', fontsize=14)
      plt.title('Confusion Matrix', fontsize=14)
      plt.show()
```

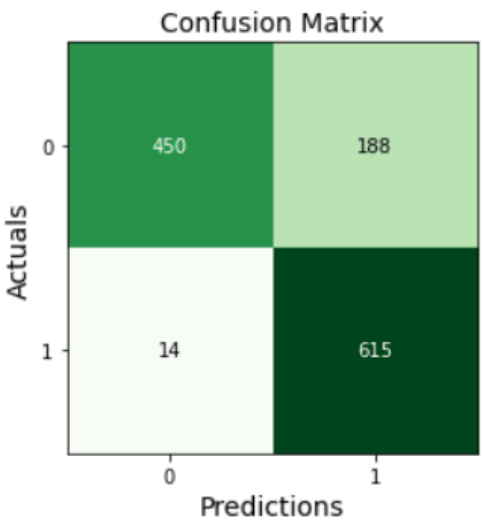

Figure 17. Confusion Matrix for TfidfVectorizer

We acquired an accuracy of 84.06% with the TfidfVectorizer NB model. And from the Confusion Matrix we have 615 true positives, 450 true negatives, 188 false positives, and 14 false negatives.

Table 1 compares the outcome of the two confusion matrices, one for CountVectorizer and the other for TfidfVectorizer.

Table 1. Outcome Comparison for the two confusion matrices.

Associated Terminologies	Feature Generation Using	
	CountVectorizer	TfidfVectorizer
True Positives	584	615
True Negatives	535	450
False Positives	103	188
False Negatives	45	14
Accuracy of the model	88.32%	84.06%

5. Conclusion

Fake news spreads quickly. There is no silver bullet. Tackling Fake News is a big problem but to every problem there is a solution. Efforts to combat false news should be coordinated with ongoing programmes (for example, critical thinking and media literacy) aimed at strengthening social resilience and national consensus. It should be strictly coordinated with the IT sector to keep on the research on fake news and how it can be detected using the ever-advancing ML approach. In addition to this, we implemented two simple ML models which are based on the Naive Bayes Classifier, one using CountVectorizer and other using TF-IDF Vectorizer and got an accuracy of 88.32% with CountVectorizer and 84.06% with TF-IDF Vectorizer. Because TF-IDF Vectorizer considers both the importance of the words and their frequency in the corpus, it is better than CountVectorizer. However, CountVectorizer provides more accurate results here than TF-IDF Vectorizer. This occurs for two reasons: first, the algorithm selection can affect which vectorizer performs best, and second, function words like pronouns are often used and would receive a lower weight in TF-IDF but would receive the same weight in the CountVectorizer as rare words. As a result, some important words in TF-IDF are given a lower weight whereas in CountVectorizer they are not down weighted, which yields higher accuracy. There are other state-of-the-art classifiers also which give promising accuracy scores, and which can be used to detect fake news efficiently in the real-world.

References

[1]. X. Zhang and A. A. Ghorbani, “An overview of online fake news: Characterization, detection, and discussion,” Inf. Process. Manag., vol. 57, no. 2, p. 102025, 2020.

[2]. S. Raza and C. Ding, “Fake news detection based on news content and social contexts: a transformer-based approach,” Int. J. Data Sci. Anal., vol. 13, no. 4, pp. 335–362, 2022.

[3]. J. C. S. Reis, A. Correia, F. Murai, A. Veloso, and F. Benevenuto, “Supervised learning for fake news detection,” IEEE Intell. Syst., vol. 34, no. 2, pp. 76–81, 2019.

[4]. “Unsupervised learning: Algorithms and examples,” AltexSoft, 14-Apr-2021.

[5]. A. Jain and A. Kasbe, “Fake News Detection,” in 2018 IEEE International Students’ Conference on Electrical, Electronics and Computer Science (SCEECS), 2018, pp. 1–5.

[6]. A. Singh, P. Singh, A. K. Tiwari, “A comprehensive survey on Machine Learning,” J. Manage. Serv. Sci., vol. 1, no. 1, pp. 1–17, 2021. DOI: https://doi.org/10.54060/JMSS/001.01.003

[7]. I. H. Sarker, “Machine learning: Algorithms, real-world applications and research directions,” SN Comput Sci, vol. 2, no. 3, p. 160, 2021.

[8]. N. Singh Kushwaha and P. Singh, “Fake News Detection using Machine Learning: A Comprehensive Analysis”,” J. Manage. Serv. Sci, vol. 2, no. 1, pp. 1–15, 2022. DOI: https://doi.org/10.54060/JMSS/002.01.001

[9]. I. Ahmad, M. Yousaf, S. Yousaf, and M. O. Ahmad, “Fake news detection using machine learning ensemble methods,” Complexity, vol. 2020, pp. 1–11, 2020.

[10]. U. Sharma, S. Saran, and M. Shankar, “Fake News Detection using Machine Learning Algorithms,” INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) NTASU, vol. 2020, no. 03, 2021.

[11]. S. Aphiwongsophon and P. Chongstitvatana, “Detecting fake news with machine learning method,” in 2018 15th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2018.

[12]. Z. Khanam, B. N. Alwasel, H. Sirafi, and M. Rashid, “Fake news detection using machine learning approaches,” IOP Conf. Ser. Mater. Sci. Eng., vol. 1099, no. 1, p. 012040, 2021.

[13]. Z. Tian and S. Baskiyar, “Fake news detection using machine learning with feature selection,” in 2021 6th International Conference on Computing, Communication and Security (ICCCS), 2021, pp. 1–6.

[14]. G. Carleo et al., “Machine learning and the physical sciences,” arXiv [physics.comp-ph], 2019.

[15]. Z. Guo, “Text classification based on naive Bayes with adjusted weights via frequency ratio of feature words,” in 2021 International Conference on Computer Technology and Media Convergence Design (CTMCD), 2021, pp. 263–267.

[16]. R. Srivastava, P. Singh, “Fake news Detection Using Naive Bayes Classifier,” J. Manage. Serv. Sci., vol. 2, no. 1, pp. 1–7, 2022. DOI: https://doi.org/10.54060/JMSS/002.01.005

[17]. M. Granik and V. Mesyura, “Fake news detection using naive Bayes classifier,” in 2017 IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON), 2017.

[18]. P. Jain, S. Sharma, Monica, and P. K. Aggarwal, “Classifying fake news detection using SVM, naive Bayes and LSTM,” in 2022 12th International Conference on Cloud Computing, Data Science & Engineering (Confluence), 2022, pp. 460–464.

[19]. S. Sarlis and I. Maglogiannis, “On the reusability of sentiment analysis datasets in applications with dissimilar contexts,” in IFIP Advances in Information and Communication Technology, Cham: Springer International Publishing, 2020, pp. 409–418.

[20]. L. Qadi, H. Rifai, S. Obaid, and A. Elnagar, “A scalable shallow learning approach for tagging Arabic news articles,” Jordanian j. comput. inf. technol., no. 0, p. 1, 2020.