Fake News Detection

Academic year : 2024/2025

Abdelilah El Fedg
Abdelilah.elfedg@etu.uae.ac.ma

Supervised by Pr. Khamjane Aziz
akhamjane@uae.ac.ma

Abstract

This project aims to develop an automated system for detecting fake news. The primary goal is to identify news articles containing false information using supervised learning models. Data was collected through web scraping from various news and fact-checking websites, creating a representative dataset. Three models were employed for training and evaluation: logistic regression, Naive Bayes classification, and decision trees. The proposed approach provides an efficient solution to address the growing issue of misinformation online.

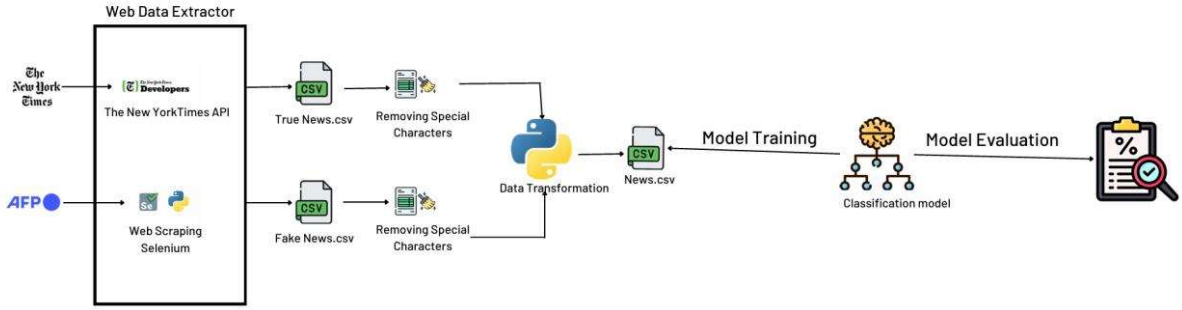
Introduction

In today's digital era, the rapid growth of social media platforms, online forums, and digital news outlets has led to an unprecedented flow of information. While this facilitates the easy dissemination of knowledge, it also brings with it the challenge of dealing with **fake news**—misleading or false information presented as credible news. The spread of fake news is a growing concern as it can be shared widely in just a few clicks, reaching millions of people within minutes, often without sufficient fact-checking or verification. This poses a serious threat to individuals' ability to distinguish between accurate and deceptive content, potentially leading to confusion, misinformation, and societal division.

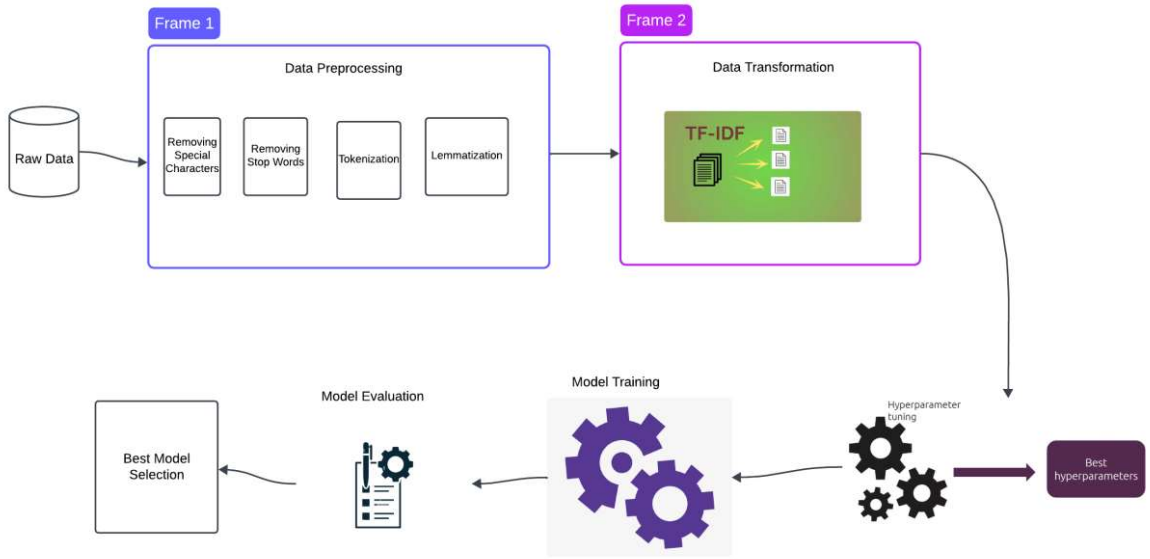
The consequences of fake news are far-reaching and can significantly impact individuals' beliefs, opinions, and actions. False information has been shown to influence public opinion, skew political discourse, and even cause harm in areas like public health. For example, the spread of fake news about medical treatments or political elections can cause panic, undermine trust in institutions, and result in misguided decisions. In the worst cases, fake news can contribute to social unrest or harmful behavior, such as people disregarding safety advice or making uninformed decisions based on false information.

This project addresses the urgent need for fake news detection using machine learning models. By leveraging techniques like **Logistic Regression**, **Naive Bayes**, and **Decision Trees**, we aim to build a system capable of distinguishing between true and false news articles.

Architecture of the project



Model Architecture



Data Preparation

1. Data collection :

data was gathered from two primary sources: The New York Times and AFP Fact Check. These sources were selected for their credibility and relevance in providing real and fake news articles, ensuring a balanced dataset for training the model.

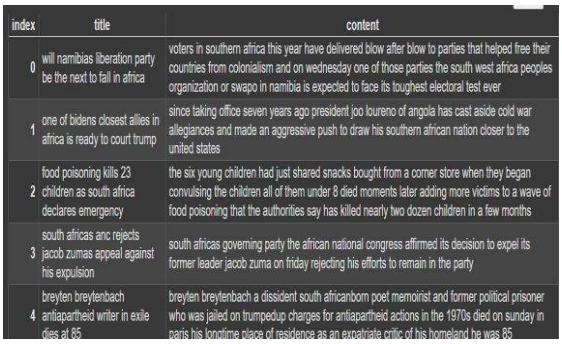
helping the model understand context across various cultural and regional

The New York Times (NYT):
To collect news articles from **The New York Times**, I utilized the official **NYT API**. This API allows access to a wide range of data, including article titles, content, and metadata. The collected data spans multiple regions worldwide, including Africa, Europe, Asia, and the Americas, providing a diverse set of articles. This geographical diversity enables the inclusion of different perspectives and topics,

2. Data preprocessing :

backgrounds. The API also allows filtering articles based on specific subjects or keywords, which made it easier to collect news articles on relevant topics, such as politics, health, and technology. This data serves as the ground truth for real news, offering a reliable reference to compare with fake news.

True News dataset :



index	title	content
0	will namibias liberation party be the next to fall in africa	voters in southern africa this year have delivered blow after blow to parties that helped free their countries from colonialism and on wednesday one of those parties the south west africa peoples organization or swapo in namibia is expected to face its toughest electoral test ever
1	one of bidens closest allies in africa is ready to court trump	since taking office seven years ago president joo loureno of angola has cast aside cold war allegiances and made an aggressive push to draw his southern african nation closer to the united states
2	food poisoning kills 23 children as south africa declares emergency	the six young children had just shared snacks bought from a corner store when they began convulsing the children all of them under 8 died moments later adding more victims to a wave of food poisoning that the authorities say has killed nearly two dozen children in a few months
3	south africas anc rejects jacob zumas appeal against his expulsion	south africas governing party the african national congress affirmed its decision to expel its former leader jacob zuma on friday rejecting his efforts to remain in the party
4	breyten breytenbach antiapartheid writer in exile dies at 85	breyten breytenbach a dissident south africanborn poet memoirist and former political prisoner who was jailed on trumpedup charges for antiapartheid actions in the 1970s died on sunday in paris his longtime place of residence as an expatriate critic of his homeland he was 85

AFP Fact Check:

The second source of data comes from **AFP Fact Check**, a platform dedicated to verifying the authenticity of news stories and claims. To collect articles from this site, **Selenium** was employed for web scraping. Selenium allows automated browsing and interaction with dynamic websites, making it ideal for scraping data from pages that require interaction (e.g., clicking buttons or loading additional content). Through this process, I was able to scrape titles and content of articles identified as **fake news**. The articles are categorized by AFP as false or misleading, making them valuable for training the fake news detection model. The automated scraping ensures that the data is up-to-date, with a continuous flow of newly flagged misinformation.

Fake News dataset :

	title	url	publication_date	content
0	Walz targeted by misleading Minnesota flag des...	https://factcheck.afp.com/doc.afp.com.36CD8J8	Published on August 9, 2024 at 22:00	["Walz changed his states (sic) flag to look ...
1	Tim Walz did not sign Minnesota law protecting...	https://factcheck.afp.com/doc.afp.com.36CQ6ZW	Published on August 9, 2024 at 19:59	["Last February, the state of Minnesota (& Go...
2	Jasper National Park wildfire reignites direct...	https://factcheck.afp.com/doc.afp.com.36C34NR	Published on August 9, 2024 at 14:53	["Was Jasper Devastation Caused by a DEW?" an...
3	Chinese posts falsely claim US Olympic swimmer...	https://factcheck.afp.com/doc.afp.com.36BL7X3	Published on August 9, 2024 at 08:05	["Behind American Walshi's 0 tests. God knows...
4	Image of Trump surrounded by Black	https://factcheck.afp.com/doc.afp.com.36AY6VD	Published on August 8, 2024 at	["They are afraid of black women

2.3. Tokenization :

To prepare the collected data for training the machine learning model, several preprocessing steps were applied. These steps are essential to clean the data and ensure its compatibility with the model. The preprocessing process is presented as follows:

2.1. Removing Special Characters :

The first step involved removing special characters and irrelevant elements from the text data. This included:

- **HTML tags**: Eliminating any residual HTML code embedded within the content.
- **Symbols**: Stripping away non-alphanumeric symbols such as @, #, $, %, etc.
- **URLs**: Removing links to external websites or references to ensure the focus remains on the textual content.
- **Extra whitespace**: Trimming excessive spaces to standardize the text structure.

2.2. Removing Stop Words :

To improve the quality of the dataset and focus on meaningful content, stop words (common words like "is," "the," and "and" that do not add significant semantic value) were removed from the text data. This step helps reduce noise and enhances the model's ability to differentiate between real and fake news. Using the **Natural Language Toolkit (NLTK)** library, a predefined set of English stop words was applied. Each article was processed by splitting its text into individual words and filtering out those listed as stop words.

Tokenization is a critical step in text preprocessing that involves breaking down text into individual units, such as words or phrases, known as tokens. This step allows for the transformation of unstructured text data into a structured format suitable for analysis. For this project, tokenization was performed using the **spaCy** library. The `nlp` model from spaCy was applied to each article in the dataset, automatically segmenting the text into tokens. Tokenization is essential for enabling further text processing steps, such as lemmatization, and for extracting meaningful features that can be used for machine learning.

2.4. Lemmatization :

Following tokenization, lemmatization was applied to standardize the text data by reducing words to their base or dictionary form (lemmas). For example, words like "running," "ran," and "runs" were converted to their root form, "run." This process reduces dimensionality and ensures that variations of a word are treated as a single entity, improving the model's ability to recognize patterns. The spaCy library's `en_core_web_sm` model was utilized to perform lemmatization efficiently. By processing each article with spaCy and extracting the `lemma_` attribute of each token, the text was transformed into its lemmatized form. This step ensured that the dataset was both normalized and semantically rich, making it more suitable for feature extraction and subsequent model training.

2.5. Text Vectorization using TF-IDF :

It predicts the probability of a data point belonging to one of two classes by applying

To convert the cleaned textual data into a numerical format suitable for machine learning models, the **Term Frequency-Inverse Document Frequency (TF-IDF)** vectorization method was applied. TF-IDF is a widely used technique that represents the importance of terms within a document relative to the entire dataset. It calculates the term frequency (**TF**), which measures how often a word appears in a document, and the inverse document frequency (**IDF**), which down-weights common terms that occur across many documents. This ensures that unique terms with higher discriminative power are given more weight.
The implementation of TF-IDF was performed using the **TfidfVectorizer** from the Scikit-learn library. The `fit_transform` method was applied to the training data to learn the vocabulary and compute the TF-IDF scores for each term, while the `transform` method was used to apply the same transformation to the test data. The resulting vectors effectively captured the importance of terms in distinguishing between real and fake news articles, enabling the model to process and classify textual data efficiently.

Model Development and Evaluation

To classify news articles as real or fake, three supervised learning models were implemented and evaluated: **Logistic Regression**, **Naive Bayes Classification**, and **Decision Tree Classification**.

1. Logistic Regression :

1.1 Introduction and Mathematical Representation :

Logistic Regression is a widely used linear model for binary classification tasks.

the sigmoid function to a linear combination of input features. The model outputs a probability score between 0 and 1, which is then thresholded to determine the predicted class.

The equation of Logistic Regression is as follows:

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

1.2. Hyperparameter Tuning :

To optimize the Logistic Regression model, a hyperparameter tuning process was performed using **GridSearchCV**, a systematic approach to searching for the best combination of hyperparameters. The tuning focused on maximizing the **F1-score**, which balances precision and recall, making it particularly suited for datasets with imbalanced class distributions.
The following hyperparameters were considered during the tuning process:

Regularization Strength (C): Values tested were `[0.01, 0.1, 1]`, where smaller values indicate stronger regularization.

Penalty: Regularization types included `'l1'` (Lasso) and `'l2'` (Ridge).

Lasso (Least Absolute Shrinkage and Selection Operator) : Lasso is a regularization method that adds a penalty based on the $L1$ norm of the model coefficients.

Ridge (Tikhonov Regularization) :

After training the logistic regression model with the optimal hyperparameters, the model's performance was evaluated on the test data. The evaluation results include the

Ridge is a regularization method that applies a penalty based on the L2 norm of the model coefficients.

Solver: Optimization algorithms tested were `'liblinear'` and `'saga'`.

`liblinear` is an optimization algorithm specifically designed for **linear classification** tasks, such as logistic regression and linear support vector machines (SVM).

`saga` is an optimization algorithm that is an extension of **stochastic gradient descent** (SGD) specifically designed to handle large-scale datasets.

Maximum Iterations (max_iter): Set to `1000` to ensure solver convergence during training.

```
params = {
    'C': [0.01, 0.1, 1],
    'penalty': ['l2', 'l1'],
    'solver': ['liblinear', 'saga']
}

model = LogisticRegression(max_iter = 1000)
grid = GridSearchCV(model, param_grid=params, scoring='f1', cv=5)
grid.fit(x_train, y_train)
```

The best hyperparameters for the logistic regression model were determined through tuning:

- **C = 1**
- Penalty = 'l2'
- **Solver = 'saga'**

1.3. Model Training and Evaluation :

The model was trained using the following code :

```
from sklearn.linear_model import LogisticRegression

LR = LogisticRegression(C=1, penalty='l2', solver='saga', max_iter=1000)

LR.fit(x1v_train, y1_train)
```

accuracy score, classification report, and confusion matrix.

Accuracy :

The accuracy of the model on the test set is **0.923**, indicating that the model correctly predicted 92% of the test cases.
On the train set the accuracy was 0.96 .

the model's high accuracy on both the training and test sets demonstrates that it has successfully learned from the data and can generalize well to new data, making it a reliable model for classifying news articles as true or fake.

Classification Report :

The classification report provides key metrics such as precision, recall, F1-score, and support for both classes (true and fake news):

	precision	recall	f1-score	support
0	0.94	0.89	0.91	1127
1	0.91	0.95	0.93	1377
accuracy			0.92	2504
macro avg	0.93	0.92	0.92	2504
weighted avg	0.92	0.92	0.92	2504

Precision :
measures how many of the predicted positive instances (fake news) are actually positive.

$$Precision = \frac{TP}{TP + FP}$$

Bayesian classification is based on **Bayes' Theorem**, which provides a probabilistic framework for predicting the class of a given instance (in this case, whether a news article is **true** or **fake**). The method applies

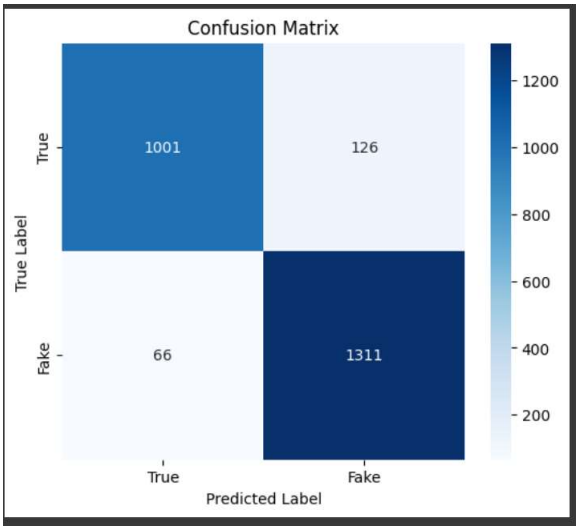
Recall :
measures how many of the actual positive instances (ake news) were correctly identified by the model.

$$Recall = \frac{TP}{TP + FN}$$

F1-score:
is the harmonic mean of precision and recall. It provides a single score that balances both precision and recall.

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

Confusion Matrix:
The confusion matrix shows the counts of true positive, true negative, false positive, and false negative predictions:



2. Bayesian Classification **:**

2.1. Introduction and Mathematical Representation :

the concept of conditional probability, calculating the probability of each class given the observed features (i.e., the words in the text). In the context of text classification, we assume that the presence of each feature (word) is independent of the others, which is known as the **Naive Bayes assumption**.

Bayes' Theorem is a fundamental concept in probability theory that describes the probability of an event occurring based on prior knowledge of related events. The formula for Bayes' Theorem is:

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

© Byjus.com

2.2. Hyperparameter Tuning :

For the **Naive Bayes** classifier, I used **hyperparameter tuning** to optimize the performance of the model. Specifically, I focused on tuning the **`alpha`** parameter of the **Multinomial Naive Bayes** model. The **`alpha`** parameter is a smoothing parameter that helps prevent zero probabilities for features that don't appear in the training data. By adjusting this parameter, the model can balance between underfitting and overfitting.
In the case of Naive Bayes, **`alpha`** is typically chosen from a set of values, and cross-validation is used to find the optimal value. **GridSearchCV** was employed to perform an exhaustive search over the specified parameter values and to identify the best combination of hyperparameters that maximized the **F1-score**, a key evaluation metric for imbalanced classification problems like fake news detection.

```
param_grid = {'alpha': [0.01, 0.5, 1, 2.0, 5.0]}
grid_search = GridSearchCV(MultinomialNB(), param_grid, scoring='f1', cv=10)
grid_search.fit(x_train, y_train)
```

The best hyperparameters for the Bayesian Classification model were determined through tuning:

- Alpha = 0.5

2.3. Model Training and Evaluation :

The model was trained using the following code :

```
NB = MultinomialNB(alpha=0.5)
NB.fit(x_train, y_train)
```

After training the Naive Bayes model with the best hyperparameters (α = 1.0), it is crucial to evaluate its performance using appropriate metrics.

Accuracy :

The accuracy of the model on the test set is **0.922**, indicating that the model correctly predicted 92% of the test cases.
On the train set the accuracy was 0.97 .

AUC-ROC (Area Under the Receiver Operating Characteristic Curve) :

AUC-ROC Score is 0.977, wish indicates that your model is performing very well in distinguishing between the two classes (fake news vs. real news).

measures the model's ability to distinguish between positive and negative classes. It represents the probability that the model ranks a randomly chosen positive instance higher than a randomly chosen negative instance.

3. Decision Tree :

3.1. Introduction and Mathematical Representation :

A **Decision Tree** is a supervised learning algorithm used for classification and regression tasks. It works by recursively splitting the data into subsets based on the feature values, forming a tree-like structure where each internal node represents a decision based on a feature, and each leaf node represents a predicted output class.

A Decision Tree algorithm recursively splits the data by selecting the feature that best separates the data points. The splitting is usually based on criteria such as **Gini Impurity**, **Entropy**, or **Variance** (for regression tasks).

For Classification (Gini Impurity):
At each node, the algorithm calculates the **Gini Impurity** to decide which feature to split on :

- **Gini Impurity** formula:

$$Gini = 1 - \sum_{i=1}^{n} (p_i)^2$$

3.2. Hyperparameter Tuning :

In order to improve the performance of the **Decision Tree** model, hyperparameter tuning was performed using **Randomized Search**. This technique explores a wide range of hyperparameters to find the the test set. However, the performance may be further improved through **hyperparameter tuning**.

I am currently optimizing the model using **Randomized Search** to find the best optimal combination that results in the best model performance. The hyperparameters tuned for the Decision Tree are as follows :

- **max_depth**: The maximum depth of the tree, which limits how deep the tree can grow. Limiting the depth prevents overfitting by ensuring the model doesn't become too complex.

- **min_samples_split**: The minimum number of samples required to split an internal node. Larger values prevent the model from creating nodes that represent only a small portion of the data.

- **min_samples_leaf**: The minimum number of samples required to be at a leaf node. This hyperparameter ensures that each leaf node has a sufficient number of samples to make reliable predictions.

- **max_leaf_nodes**: The maximum number of leaf nodes in the tree. By limiting the number of leaf nodes, the model is constrained in terms of its complexity.

- **min_impurity_decrease**: This parameter specifies the minimum decrease in impurity required to split a node. A higher value would prevent unnecessary splits and reduce overfitting.

3.3. Model Training and Evaluation :

hyperparameters. The process is time-consuming, but it is expected to improve both the model's performance and computational efficiency.

Conclusion

In this project, we aimed to build and evaluate models for **fake news detection** using a variety of machine learning techniques. The project followed a systematic approach, beginning with data collection from reliable news sources, followed by comprehensive preprocessing steps, including **removal of special characters**, **stop words removal**, **tokenization**, and **lemmatization**. These steps ensured that the dataset was cleaned and prepared for effective model training.

For the classification task, three models were used: **Logistic Regression**, **Naive Bayes**, and **Decision Tree**. Through the evaluation of each model, we found that **Logistic Regression** performed best with an accuracy of **0.92**, demonstrating its strength in handling binary classification problems. The **Naive Bayes** model also performed well, with an impressive **AUC-ROC** of **0.98**, indicating its robustness in distinguishing between fake and true news. The **Decision Tree** model, though offering a solid performance with an accuracy of **0.85**, still requires hyperparameter tuning to optimize its performance and reduce computation time.

In conclusion, this project successfully built effective models for detecting fake news, with **Logistic Regression** emerging as the best performing model. Future improvements in hyperparameter tuning

The current **Decision Tree** model achieved an accuracy score of **0.85** on and additional features could further enhance the system's ability to accurately classify news articles. The findings of this project have practical implications for building automated systems to identify and combat the spread of misinformation.