Fake News Detection

Hemalatha Subbiah

College of Science and Technology, Bellevue University

Professor. Amirfarrokh Iranitalab

June 23, 2024

Contents

Business Problem:

This project aims to classify news articles as real, or fake based on their content. Specifically, this project will use machine learning to build a model to predict whether a given news article is real, or fake based on its text.

Background/History:

Fake news is sometimes transmitted through the internet by some unauthorized sources, which creates issues for the targeted person, and it makes them panic and leads to even violence. To combat the spread of fake news, it's critical to determine the information's legitimacy, which this project can help with. This project is relevant to the media industry, news outlets, and social media platforms that are responsible for sharing news articles. Classifying news articles as real or fake can help these organizations improve their content moderation and reduce the spread of fake news.

Data Explanation (Data Prep/Data Dictionary/etc.):

A fake news dataset typically comprises a collection of articles, social media posts, or other textual content that is labeled as either true or false. These datasets are used to train and evaluate machine learning models for detecting misinformation. They often include various features such as the text of the content, metadata, and user interactions. Creating and maintaining such datasets involves significant challenges, such as ensuring the accuracy of labels, addressing the evolving nature of fake news, and addressing privacy concerns related to the data used. These datasets are crucial for advancing research and developing more effective fake news detection systems. Dataset separated in two files: 1. Fake.csv (23502 fake news article) 2. True.csv (21417 true news article)

Dataset columns:

1. Title: Title of news article

2. Text: Body text of news article

3. Subject: Subject of news article

4. Date: Publish date of news article

Checked for any missing values appropriately, no null values are found in dataset.

Data Exploration

```
24]: print(real_news_df.isnull().sum())
     print(fake_news_df.isnull().sum())
```

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

In the below figure (Figure:1), the distribution of article lengths in each dataset is plotted. The length of the articles is highly variable, with some articles being very short and others being quite long. We will need to take this into account when preprocessing the text.

Figure: 1

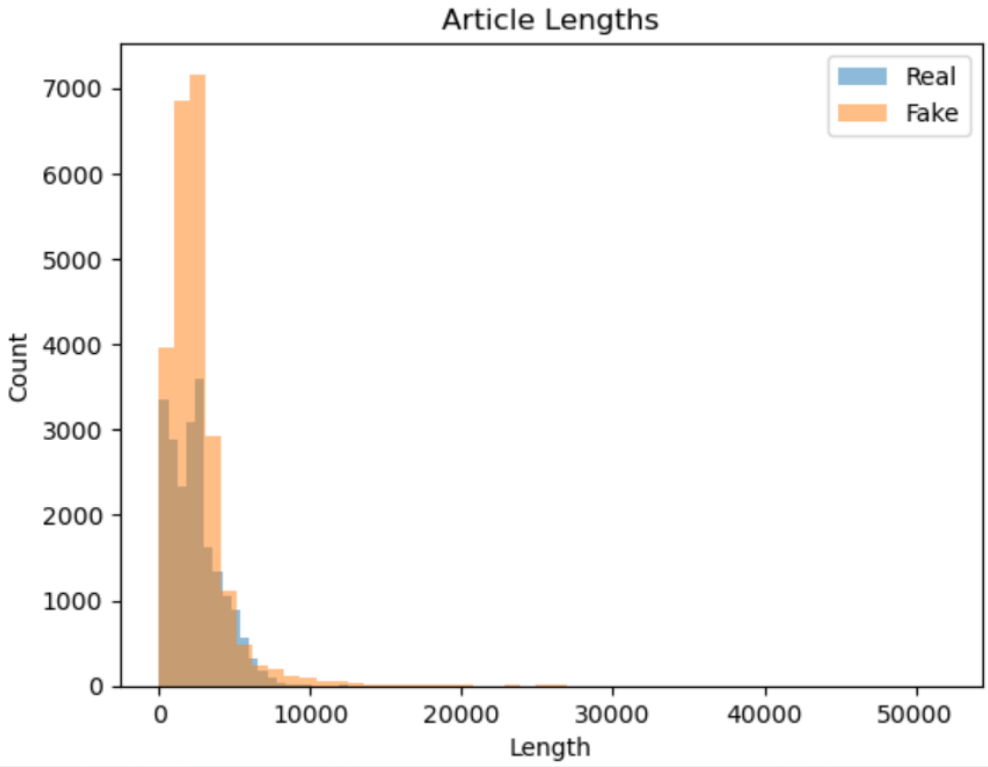


Figure 2 illustrates subject wise distribution of fake news

Figure 2:

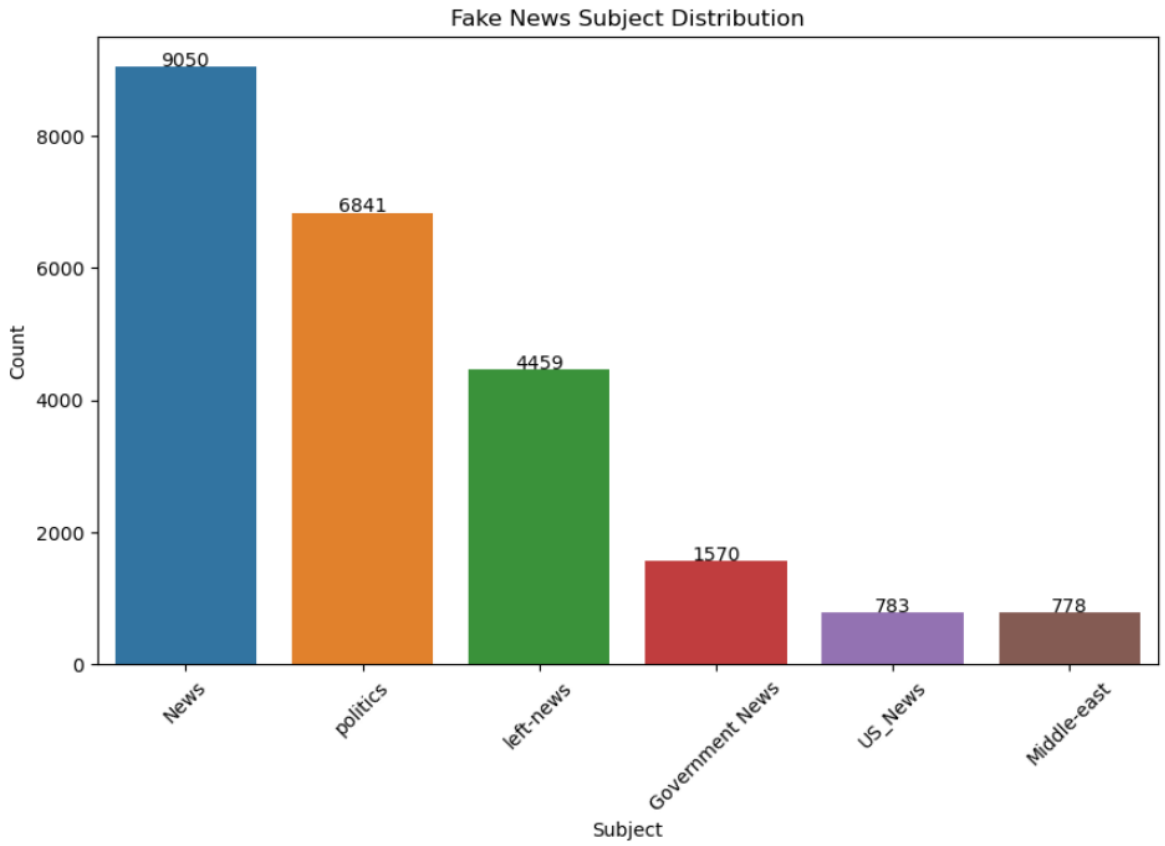
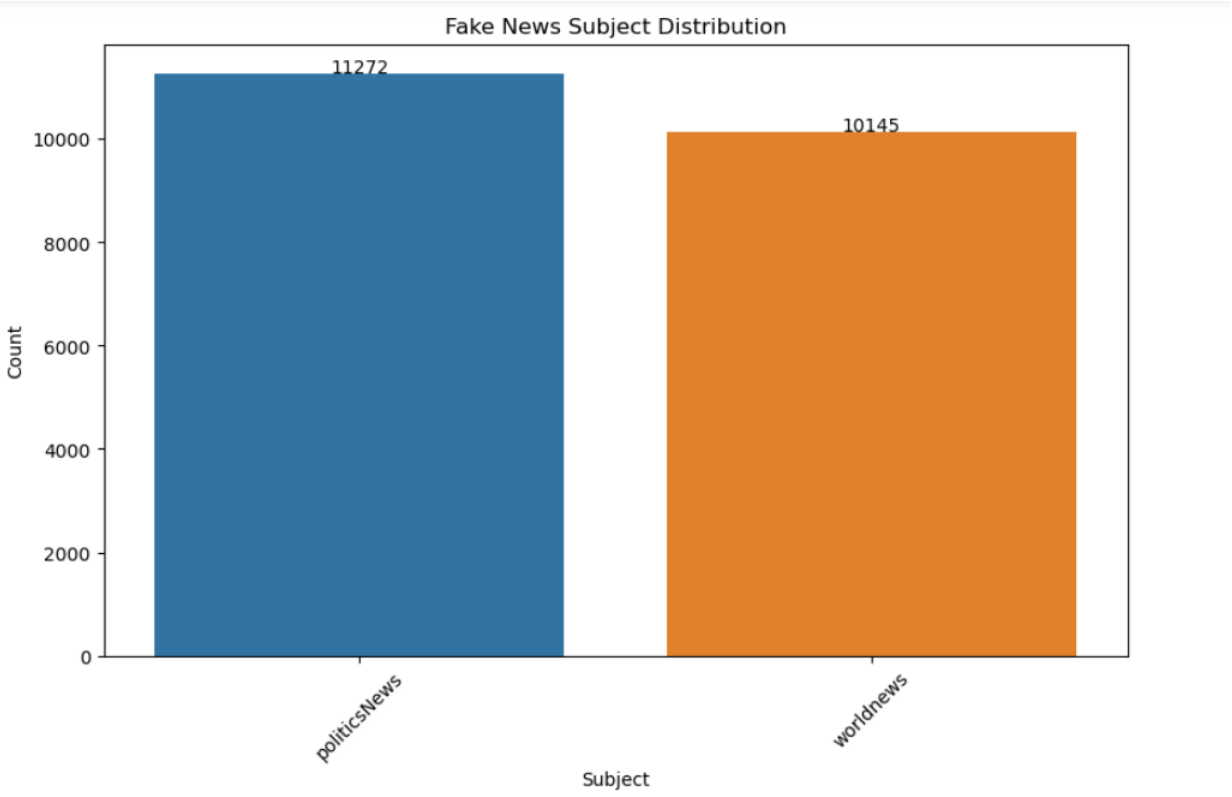


Figure 3

Figure 3 illustrates subject wise distribution of real news



Implementation Plan:

Once I downloaded the dataset, loaded it into a Pandas DataFrame. The 'real_news' DataFrame contains real news articles and their labels, and the 'fake_news' DataFrame contains fake news articles and their labels. Then preprocessed the text data lowercasing the text, removing punctuation and digits, removing stop words, stemming or lemmatizing the text.

CountVectorizer class from the sklearn library to convert the preprocessed text into feature vectors. CountVectorizer is a commonly used text preprocessing technique in natural language processing. Other methods for converting textual data into numerical features include TF-IDF (term frequency-inverse document frequency), Word2Vec, Doc2Vec, and GloVe (Global Vectors for Word Representation).

Using TfidfVectorizer for a fake news dataset involves transforming the text data into numerical features that can be used to train machine learning models. The TfidfVectorizer

converts a collection of raw documents into a matrix of TF-IDF (Term Frequency-Inverse Document Frequency) features, which reflect the importance of terms in each document relative to the entire dataset.

Following steps are followed:

1.Lowercasing the text

2.Removing punctuation and digits

3.Removing stop words

4.Stemming or lemmatizing the text

Then created a CountVectorizer object and fit it to the preprocessed text in the real news dataset and used the same vectorizer to transform the preprocessed text in the fake news dataset. Later stacked the feature matrices for both datasets vertically and create a corresponding label vector, y.

```
]: vectorizer = CountVectorizer()
   X_real = vectorizer.fit_transform(real_news_df['text'])
   X_fake = vectorizer.transform(fake_news_df['text'])

   X = sp.vstack([X_real, X_fake])
   y = np.concatenate([np.ones(X_real.shape[0]), np.zeros(X_fake.shape[0])])
```

Split the data into training and testing sets:

```
: from sklearn.model_selection import train_test_split

  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

I trained my model, evaluated its performance on the test set and calculated metrics for accuracy, precision, recall, and F1 score.

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

y_pred = clf.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print('Accuracy:', accuracy)
print('Precision:', precision)
print('Recall:', recall)
print('F1 Score:', f1)
```

I evaluated the accuracy of these algorithm

```
: from sklearn.naive_bayes import MultinomialNB
  from sklearn.linear_model import LogisticRegression
  from sklearn.tree import DecisionTreeClassifier
  from sklearn.ensemble import RandomForestClassifier
  from sklearn.svm import SVC


  models = [
      MultinomialNB(),
      LogisticRegression(),
      DecisionTreeClassifier(),
      RandomForestClassifier(),
      SVC()
  ]

  for model in models:
      model.fit(X_train, y_train)
      y_pred = model.predict(X_test)
      accuracy = accuracy_score(y_test, y_pred)
      print(f'{model.__class__.__name__}: {accuracy*100:.2f}')
      print("-"*30)
```

```
MultinomialNB: 94.22
------------------------------

LogisticRegression: 99.50
------------------------------
DecisionTreeClassifier: 99.52
------------------------------
RandomForestClassifier: 99.24
------------------------------
SVC: 99.19
------------------------------
```

Analysis:

I did several types of analysis on a fake news detection dataset to gain insights and improve model performance. I examined the distribution of real vs. fake news articles, analyzed the length of the articles to see if there are any noticeable differences between real and fake news and identified the most common words in real and fake news article. Finally did an analysis on the performance of different algorithms.

Assumptions:

When building a fake news detection model, key assumptions include:

1. **Data Assumptions**: The dataset's labels are accurate and balanced, and the data is representative and clean.
2. **Model Assumptions**: Features used are relevant, and models like Naive Bayes assume feature independence, while SVM and Logistic Regression assume linear separability.
3. **Training and Evaluation Assumptions**: Training and testing splits are representative, cross-validation is properly performed, and evaluation metrics like accuracy, precision, recall, and F1-score are appropriate.
4. **Algorithm Assumptions:** The selected algorithms match the problem complexity, and proper hyperparameter tuning is conducted.

Limitations:

Building a fake news detection model faces several limitations, including data Issues inaccurate labels, dataset bias, and data imbalance can mislead the model. Inadequate features, overfitting, and algorithmic biases can reduce effectiveness. Metrics like accuracy can be misleading, and improper cross-validation can skew performance estimates. Ethical misuse, the dynamic nature of fake news, interpretability issues, and scalability challenges impact the model's reliability and trustworthiness. Addressing these limitations requires careful data handling, robust model design, and ethical considerations.

Challenges/Issues:

Challenges in fake news detection include technical limitations such as algorithmic bias, false positives/negatives, and the rapid evolution of misinformation tactics. The complexity of detecting subtle misinformation and variations across languages and cultures adds to the difficulty. Ethical and legal issues involve balancing censorship with free speech, addressing privacy concerns, and ensuring accountability for automated decisions. Operational challenges include scalability, real-time detection, and resource constraints. Public trust is crucial, yet skepticism about detection systems can undermine their effectiveness. Additionally, efforts must consider the societal impact and the need for public education in media literacy.

Future Uses/Additional Applications:

Fake news detection models offer versatile applications beyond news verification, spanning social media monitoring, journalistic integrity, educational tools for media literacy, public health crisis management, corporate brand protection, governmental policy support, search engine content filtering, legal compliance, consumer protection, and cross-platform integration. These applications collectively enhance information integrity and trust across diverse sectors.

Ethical Assessment:

Ethical considerations in fake news detection include balancing freedom of speech with preventing misinformation, avoiding excessive censorship, and mitigating algorithmic biases to ensure fairness and transparency. Privacy concerns must be addressed, requiring informed user consent for data collection. Accuracy is crucial, minimizing false positives and negatives by using reliable sources. Clear accountability and robust governance are necessary, along with transparency to maintain public trust. Education and empowerment through media literacy and verification tools are essential. Global sensitivity to cultural differences and international collaboration, respecting diverse legal and ethical standards, is also important.

Conclusion:

Preprocessing is a crucial step in Natural Language Processing (NLP) as it involves preparing and cleaning text data to make it suitable for building models and extracting insights. Here are some common preprocessing steps in NLP - Text Cleaning, Tokenization, Stop Word Removal, Stemming and Lemmatization, Part-of-Speech (POS) Tagging.

Converts a collection of text documents to a matrix of token counts, which can then be used as input for machine learning algorithms. The choice of a machine learning algorithm can significantly impact the performance of a text classification task. In this project, we compared the performance of logistic regression, decision tree, random forest and support vector machines and found that logistic regression had the best performance.

References:

https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset

https://paperswithcode.com/dataset/liar