Detecting Fake News with Machine Learning Techniques

This is groups project for CAPP 30254: Machine Learning for Public Policy

Done By: Big Brother Debunker
Team Member: Dingwei Liu & Qi Zhao

Introduction and research problem

Misinformation in the media, exemplified by events like the Arab Spring, can influence public opinion and incite social unrest. Traditional verification methods, reliant on human effort, struggle with the vast volume of online content.

This project aims to create a scalable tool that detects and classifies fake news, assisting users and tech companies in identifying reliable information and reducing the spread of misinformation. We proposes an automated solution to detect fake news by analysing textual features such as words, phrases, sources, and titles using machine learning. Utilizing a labelled dataset from Kaggle, we applied supervised machine learning algorithms and feature selection techniques to develop a predictive model. The model's accuracy is determined through testing on unseen data.

Methodology

We adhere to a structured approach consisting of data pre-processing, feature engineering, classification, and analysis. The detailed steps are as follows.

Data Pre-processing

- We begin by checking for missing values (NAs) across all columns and confirm that the label column contains no NAs. Since our goal is to extract additional attributes from other columns, there is no need to remove any columns at this initial stage.

  Shape

  - Rows: 20800
  - Columns: 5
    - id, title, author, text, label

Missing values

id	title	author	text	label
0	558	1957	39	0

Label distribution

label	0	1
count	10387	9855

- For the titles and text, we pre-process by removing non-English words, special characters, HTML tags, and stopwords. Additionally, we standardize text casing, perform lemmatization to reduce words to their base forms, strip extra spaces, and filter out common but uninformative words to enhance the clarity and effectiveness of our data for classification tasks.
- Split news source from the title. Source refers to the organization where the news is coming from. For example, The New York Times. However, only part of the title have source information. We believe containing source information or not is caused by crawling, which is irrelevant to whether one news is fake.

  For example, id=80:

  - Before:

    ```
    Louisiana, Simone Biles, U.S. Presidential Race: Your Tuesday
    Evening Briefing - The New York Times
    ```

  - After:

    ```
    Louisiana, Simone Biles, U.S. Presidential Race: Your Tuesday
    Evening Briefing
    ```

- Stemming: Stemming is performed to reduce words to their base form, thereby decreasing the number of word variations. This helps machine learning models better understand and process language data because different forms of the same root word (such as "run", "running", "runner") are semantically similar. With stemming, the model can treat these variations as the same word, reducing the complexity of the feature space and improving processing efficiency and model performance.
- Explore the word count:

Feature Engineering

Feature Extraction

Based on our literature review and initial data exploration—where we discovered a significant correlation between the news source and the incidence of fake news—we have expanded our feature extraction beyond the traditional TF-IDF. This will allow for more effective classification.

Initially, we construct our baseline model using **TF-IDF features**. We generate sparse vector representations of titles and article bodies by calculating the term frequency of each n-gram and normalizing these by their inverse document frequency. We begin by fitting a `TfidfVectorizer` to the concatenated texts of titles and bodies to establish the vocabulary. Subsequently, using this vocabulary, we separately fit and transform the n-grams of titles and bodies into sparse vectors. Additionally, we calculate the cosine similarity between the vectors of the titles and bodies.

Furthermore, we have extracted additional attributes such as:

- **check whether the news have title**
- **check whether the news have author**
- **check whether the news have source**
- **check whether the news' title starts with number**
- **check whether title contains ? and/or ! marks**
- **check whether all words are capital in title**
- Perform feature extraction by choosing lexical features, Such as **word count, average word length, length of article, number count, number of sections of speech (adjective).**
- Basic count for title and text tuple. We takes the **uni-grams, bi-grams and tri-grams** and creates various counts and ratios which could potentially signify how a body text is related to a title. Specifically, it counts how many times a gram appears in the title, how many unique grams there are in the title, and the ratio between the two. The same statistics are computed for the body text, too. It then calculates how many grams in the title also appear in the body text, and a normalized version of this overlapping count by the number of grams in the title. The results are saved in the pickle file which will be read back in by the classifier.
- **Sentiment features:** we uses the Sentiment Analyzer in the `NLTK` package to assign a sentiment polarity score to the headline and body separately. For example, negative score means the text shows a negative opinion of something. This score can be informative of whether the body is being positive about a subject while the headline is being negative. But it does not indicate whether it's

the same subject that appears in the body and headline; however, this piece of information should be preserved in other features.

- **Polarity features:** The polarity of the title and text is determined by counting the number of negative sentiment words. Here, the NRC Word-Emotion Association Lexicon is used to obtain a list of negative emotion associated English words. The identity of a negative word is determined by whether the word is found on the negative word list or not. We assign the polarity of a corpus based on whether it has odd or even number of negative words.

Feature Selection

We employ a two-stage method for selecting features for our model, aiming to optimize the predictive power for detecting fake news.

First Stage: We begin by evaluating new features using information gain and Pearson correlation, both with the label and among the features themselves, to assess their predictive relevance. If two features demonstrate a correlation exceeding 90%, we retain the feature with the higher information gain and discard the other. Additionally, features that exhibit zero information gain or no correlation with the label are removed.

Second Stage: In this phase, we integrate information gain, Pearson correlation, and Recursive Feature Elimination (RFE) to refine our feature selection. This approach combines the strengths of each method, enhancing the model’s ability to discriminate between genuine and fake news by focusing on the most informative and statistically significant features.

Based on this rigorous selection process, we ultimately identified 10 features that, alongside our baseline TF-IDF features, are used to train our model. This strategic combination allows us to leverage both textual and contextual information, significantly boosting the model’s accuracy and robustness.

Classification

We partition the dataset into training, testing and validation sets with ratios of 70%, 20%, and 10%, respectively, using Python's sklearn library. After applying the chosen algorithms, we document the processes and results in a Jupyter notebook (.ipynb file) that encapsulates the classification models.

- We use six supervised machine learning models for the detection:
- SVM
- Logistic Regression
- Decision Tree
- Random Forest
- Perceptron

- Gaussian Naive Bayes (GNB)

Analysis

We evaluate the precision of our model using the test portion of the dataset and generate a confusion matrix. We assess the model's performance by examining accuracy, precision, recall, F1-score, and AUC for both fake and real news categories.

Benchmark

- Accuracy:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

  - Using %accuracy is acceptable as the label is **well-balanced**
- Precision

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

- Recall

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

- F1-score:

$$F_1\ Score = 2 * \frac{precision * recall}{precision + recall} = \frac{TP}{TP + 1/2(FP + FN)}$$

- ROC AUC Score

$$\text{AUC} = \sum_{i=1}^{n-1} (\text{FPR}_{i+1} - \text{FPR}_i) \cdot \frac{\text{TPR}_{i+1} + \text{TPR}_i}{2}$$

Result

TF-IDF

	Accuracy	Precision	Recall	F1	AUC
SVC	94.96%	93.03%	96.39%	94.68%	95.05%
Logistic Regression	93.89%	92.14%	94.97%	93.53%	93.96%
Decision Tree	87.02%	85.93%	86.21%	86.07%	86.96%

	Accuracy	Precision	Recall	F1	AUC
Perceptron	93.84%	91.69%	95.40%	93.51%	93.94%
RandomForest	90.43%	93.11%	85.78%	89.29%	90.13%
GaussianNB	82.38%	81.07%	81.07%	81.07%	82.30%

Combined

	Accuracy	Precision	Recall	F1	AUC
SVC	96.02%	94.00%	97.70%	95.82%	96.14%
Logistic Regression	96.13%	94.39%	97.48%	95.91%	96.22%
Decision Tree	95.11%	94.95%	94.53%	94.74%	95.07%
Perceptron	95.11%	95.96%	93.44%	94.68%	95.00%
RandomForest	95.16%	97.01%	92.45%	94.68%	94.99%
GaussianNB	82.48%	81.53%	80.63%	81.08%	82.36%

Hyper-Parameter Modify

Decision Tree

TF-IDF

Combined

Conclusion