Fake and real news dataset
Kaggle

Link of the project:

[*https://github.com/lmolinario/ML-Project-Fake-Real-News/tree/main*](https://github.com/lmolinario/ML-Project-Fake-Real-News/tree/main)

Michela Lecca [70/90/00343] Marco M
[70/90/00327] Lello Molinario [70/90/00

Introduction

Objective:

Distinguishing fake news from real news

Dataset:

fake-and-real-news-dataset (kaggle.com)

Tools:

Python, Pandas, scikit-learn, nltk …

Util modules:

nlp, classifiers, import_dataset, text_representation …

Michela Lecca [70/90/00343]
Marco Mulas [70/90/00327] Lello Molinario [70/90/00369]

Dataset description

Two files:

Fake.csv (23502 fake news article)

True.csv (21417 true news article)

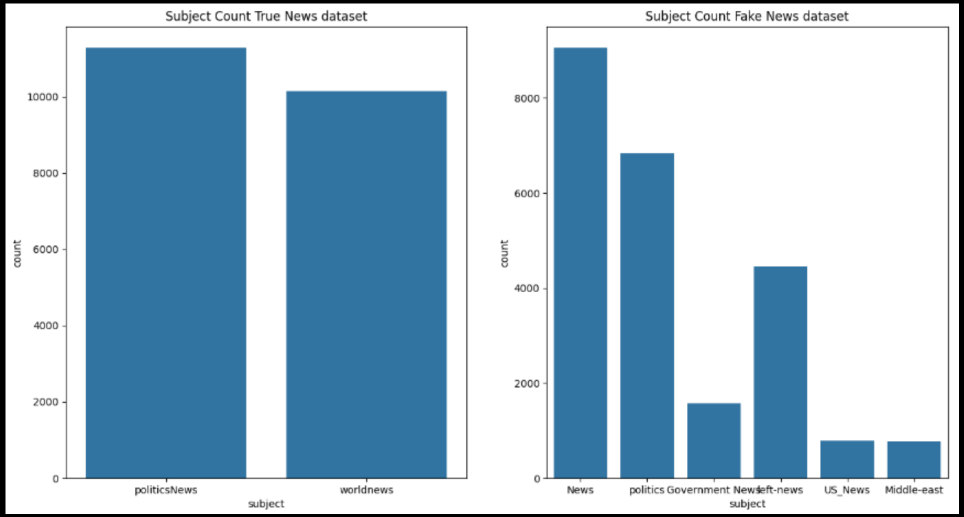
Features:

Title: title of news article

Text: body text of news article

Subject: subject of news article

Date: publish date of news article



Michela Lecca [70/90/00343]
Marco Mulas [70/90/00327] Lello Molinario [70/90/00369]

Dataset manipulation

- True.csv and Fake.csv loaded as Pandas Dataframe
- Add new 'label' field
  - 0 = true news
  - 1 = fake news
- Creation of a unique News Dataframe from the true and fake news
- Removed unused date column

Pre processing

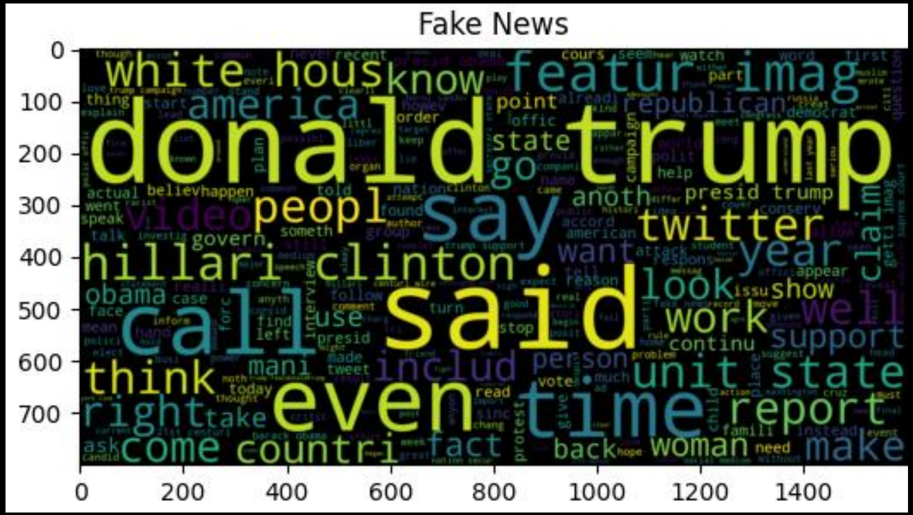
- Extract the most meaningful information.
  - Append 'title' and 'text' into a **new field** 'title_text'
  - Convert 'title_text' to **lowercase**
  - **Replace the non-alphanumeric** characters with spaces
  - **Remove words** shorter than 3 characters
  - **Lemmatization** and **stemming**
  - Isolation of all **unique words** (‘filtered_unique’) from ‘filtered string’

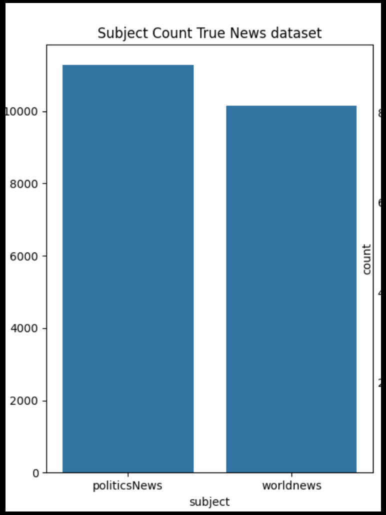
```
['recent', 'puerto', 'govern', 'hard', 'debbi'...
['thousand', 'govern', 'hard', 'time', 'babb',...
['counsel', 'recent', 'matter', 'time', 'want'...
['counsel', 'recent', 'govern', 'chicago', 'ma...
['flat', 'giant', 'govern', 'past', 'want', 'c...
```

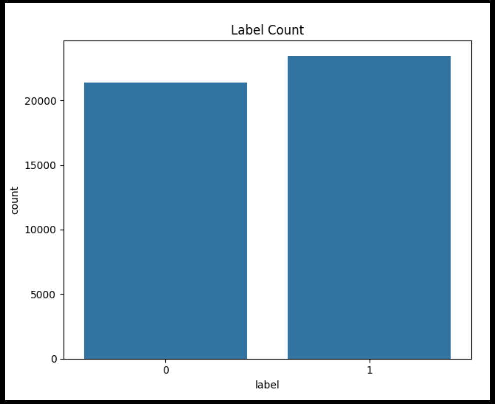
Michela Lecca [70/90/00343]
Marco Mulas [70/90/00327] Lello Molinario [70/90/00369]

Statistical analysis

- Identification of :
  - Total number of unique words: *41*
  - Document with max words: *id 43720 (46132 words):*
  - Document with min words: *id 33199 (28 words)*
  - Document with max unique words: *id 43923 (18196 words)*
  - Document with min unique words: *id 33199 (28 words)*
  - Word cloud of the most common words,
  - Count plots of subjects in each class,
  - Count plot of the proportion of fake and true news in the mixed dataset.







Michela Lecca [70/90/00343]
Marco Mulas [70/90/00327] Lello Molinario [70/90/00369]

Strategy Design Pattern:
modular text classification pipeline

Key Features:

- **Abstract Class**: Defines a common interface for data representation strategies.
- **K-Fold Cross-Validation**: Splits the data into training and testing sets.

Data Representation Methods:

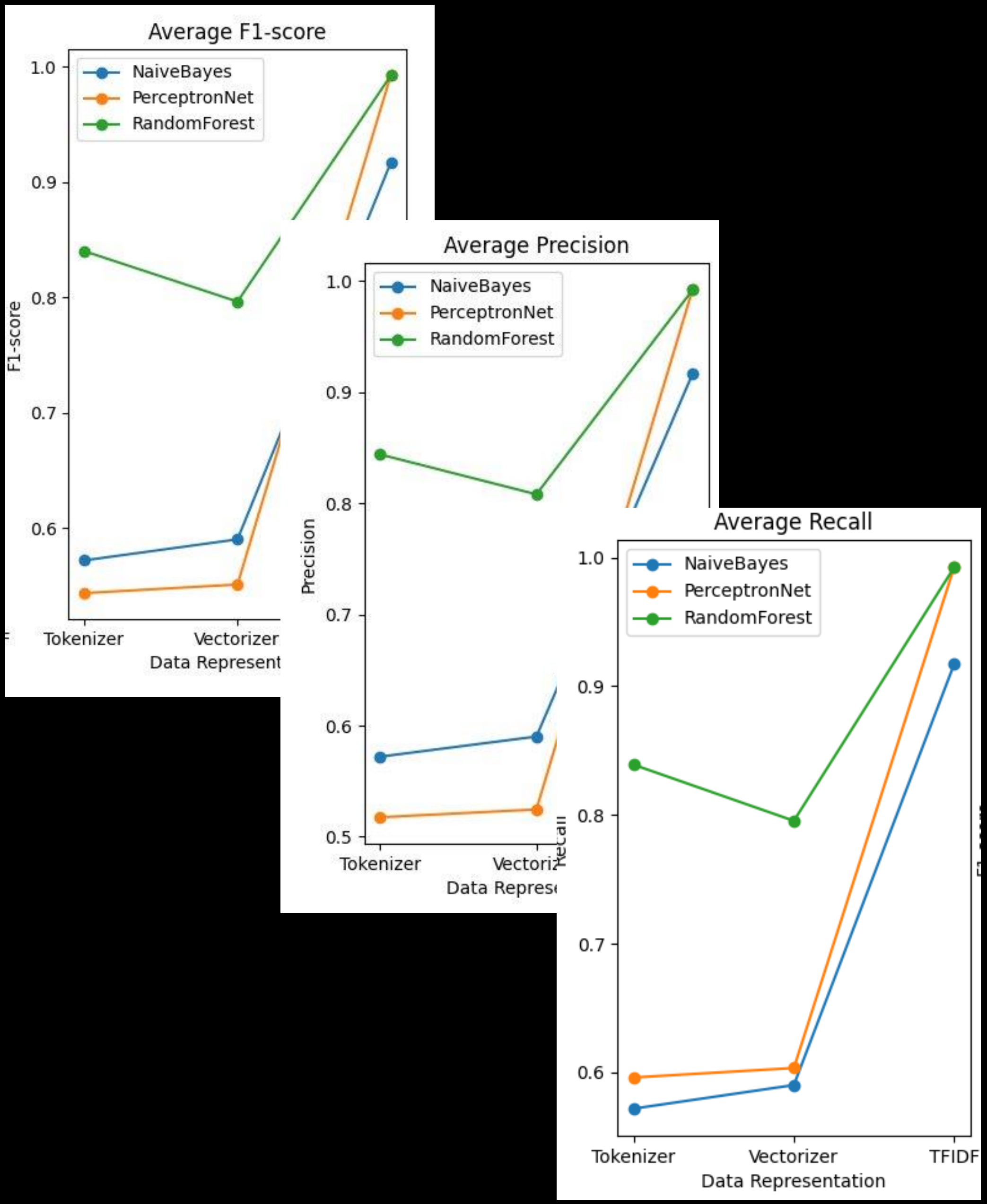
TokenizerRepresentation:
- Method: Keras Tokenizer
- Process: Converts text to integers and pads sequences to a fixed length
- Usage: suitable for embedding layers in neural networks

TextVectorizationRepresentation:
- Method: TensorFlow's TextVectorization
- Process: Converts text to integers, pads sequences to a fixed length
- Usage: provides additional preprocessing capabilities

TFIDFRepresentation:
- Method: Scikit-learn's TfidfVectorizer
- Process: Converts text into TF-IDF numeric features
- Usage: Emphatizing the importance of words within the corpus

Classification Strategies:

- **RandomForest**: multiple decision trees
- **NaiveBayes**: binary classification
- **MultiLayerPerceptronNet**: Neural network-based using multi-layer perceptron for classification



Performance Evaluation

Average Accuracy	Average recall	Average F1 Score
Precision is the ratio of actual positive predictions to the total predicted positives.	Recall is the ratio of true positive predictions to total actual positives.	The F1 score is the harmonic mean of precision and recall.

Evaluation Analysis

- **NaiveBayes**
  - **Best Performance Metric**: TF-IDF
  - **Why**: Metrics are consistently high for TF-IDF, indicating that NaiveBayes works exceptionally well with this data representation. It shows significant improvements in precision, recall, and F1-score, highlighting its ability to leverage the importance-weighted word representation effectively.
- **PerceptronNet**
  - **Best Performance Metric**: TF-IDF
  - **Why**: PerceptronNet exhibits the most dramatic improvement with TF-IDF, achieving near-perfect scores in all metrics. This suggests that PerceptronNet can capture and learn from the detailed features provided by TF-IDF, making it highly effective for this task.
- **RandomForest**
  - **Best Performance Metric**: TF-IDF
  - **Why**: RandomForest maintains high performance across all data representations, with the best metrics for TF-IDF. The high F1-score indicates a good balance between precision and recall, suggesting that it effectively handles both false positives and false negatives. Its robust performance across different representations makes it a versatile choice.

```
Across the following data representations:
['Tokenizer', 'Vectorizer', 'TFIDF']
```

```
Average Precision values:
NaiveBayes: [0.5720542895892652, 0.5901571236861897, 0.9166550895271154]
PerceptronNet: [0.5174980026241675, 0.5245509339100118, 0.9925334849678796]
RandomForest: [0.8439982447841204, 0.8081358682813198, 0.9920351698830965]

Average Recall values:
NaiveBayes: [0.5718871781676648, 0.5902159235921273, 0.917258173970109]
PerceptronNet: [0.5960983325542836, 0.6034691675444028, 0.9925577766874021]
RandomForest: [0.8390318235615268, 0.7954277605238836, 0.9923320036700861]

Average F1-score values:
NaiveBayes: [0.571905327391753, 0.5901578289124286, 0.9168567773351598]
PerceptronNet: [0.5435357078070865, 0.5510214146955907, 0.9925451110501333]
RandomForest: [0.8400051801608613, 0.7962118669954023, 0.9921684616295635]
```

Conclusion

- **Model with the Best Performance Overall**

Random Forest consistently has high metrics across all representations, indicating robust performance regardless of the data representation used.

- **Why Random Forest Works Better?**

Ensemble Learning: Uses multiple decision trees, reducing overfitting and improving generalization.

Robustness: Handles variance and bias balancing well by averaging multiple trees.

Feature Importance: Effectively handles high-dimensional data and identifies important features, improving performance across different data representations.