Real OR Fake News Analysis

**This dataset consists of about 40000 articles consisting of fake as well as real news. Our aim is train our model so that it can correctly predict whether a given piece of news is real or fake. The fake and real news data is given in two separate datasets with each dataset consisting around 20000 articles each.**

Business Objective:

1. Need to classify the fake and real news accurately.

Architecture level analysis:

1. Data transformation/Text processing using R/Python
2. Need to get sentiments Analysis and n-gram analysis with some charts like histogram, Density plot, Barplot, pie-plot etc.
3. Deployment through R Shiny or Flask/ Streamlit

**Milestones: Phase 1**

|  |  |  |
| --- | --- | --- |
| **Milestone** | **Duration** | **Task start - End Date** |
| Kick off and Business Objective discussion | 1 day |  |
| Data set Details | 1 Week – 1 ½ week |  |
| EDA | 1 Weeks – 1 ½ week |  |
| Model Building | 1 Week – 1 ½ week |  |
| Model Evaluation | 1 week |  |
| Feedback |  |
| Deployment | 1 Week |  |
| Final presentation | 1 day |  |

Protocols:

1. All participants should add here to agreed timelines and timelines will not be extended
2. All the documentation – Final presentation and R/python code to be submitted before the final presentation day
3. All the participants must attend review meetings
4. Preprocess the data: Clean and preprocess the text data by removing any irrelevant information, such as HTML tags, punctuation, and stopwords. You may also want to consider stemming or lemmatizing the words to reduce their variations.
5. Combine the datasets: Merge the two datasets containing real and fake news articles into a single dataset. Make sure to maintain the labels indicating whether each article is real or fake.
6. Split the data: Split the combined dataset into training, validation, and testing sets. The typical split is around 70% for training, 15% for validation, and 15% for testing. This will allow you to train the model, tune hyperparameters, and evaluate its performance.
7. Vectorize the text: Convert the text data into numerical representations that can be understood by machine learning algorithms. Common approaches include using bag-of-words models, TF-IDF (Term Frequency-Inverse Document Frequency), or word embeddings like Word2Vec or GloVe.
8. Train the model: Select a suitable machine learning algorithm, such as logistic regression, support vector machines, or neural networks, and train it on the training set. You can experiment with different models and hyperparameters to find the best performance.
9. Evaluate the model: Use the validation set to evaluate the performance of your trained model. Common evaluation metrics for binary classification tasks include accuracy, precision, recall, and F1 score. Adjust your model or experiment with different techniques if necessary.
10. Test the model: Once you are satisfied with the model's performance on the validation set, evaluate it on the testing set to get a final assessment of its accuracy and generalization ability.
11. Fine-tune and iterate: If the model's performance is not satisfactory, you can try fine-tuning the hyperparameters, experimenting with different architectures, or using more advanced techniques like ensemble learning or transfer learning.
12. [Import Libraries](https://chat.openai.com/#1)

* This section likely involves importing necessary libraries or modules required for the data analysis and modeling process.

1. [Load and Check Data](https://chat.openai.com/#2)

* This section involves loading the dataset and performing some initial checks to ensure data integrity and consistency.

1. [Visualization](https://chat.openai.com/#3)

* In this section, data may be visualized using various plotting techniques to gain insights and understand patterns.

1. [Data Cleaning](https://chat.openai.com/#4)

* This section deals with preparing the data for analysis by performing various cleaning tasks on the text data.

1. [Removal of HTML Contents](https://chat.openai.com/#5)
   * HTML tags, if present, are removed from the text data.
2. [Removal of Punctuation Marks and Special Characters](https://chat.openai.com/#6)
   * Punctuation marks and special characters are removed from the text data.
3. [Removal of Stopwords](https://chat.openai.com/#7)
   * Commonly occurring words with little or no significance (stopwords) are removed from the text data.
4. [Lemmatization](https://chat.openai.com/#8)
   * Words are reduced to their base or root form using lemmatization techniques.
5. [Perform it for all the examples](https://chat.openai.com/#9)
   * The cleaning tasks are applied to all examples in the dataset.
6. [N-Gram Analysis](https://chat.openai.com/#10)

* This section involves analyzing the text data by breaking it into N-grams, which are contiguous sequences of N items.

1. [Unigram Analysis](https://chat.openai.com/#11)
   * Analysis is done on single words (individual tokens) in the text data.
2. [Bigram Analysis](https://chat.openai.com/#12)
   * Analysis is done on pairs of consecutive words in the text data.
3. [Trigram Analysis](https://chat.openai.com/#13)
   * Analysis is done on triplets of consecutive words in the text data.
4. [Modeling](https://chat.openai.com/#14)

* This section focuses on building and training a machine learning model using the processed text data.

1. [Train - Test Split](https://chat.openai.com/#15)
   * The dataset is divided into training and testing sets to evaluate the model's performance.
2. [Tokenizing](https://chat.openai.com/#16)
   * The text data is converted into numerical tokens suitable for input to the model.
3. [Training LSTM Model](https://chat.openai.com/#17)
   * An LSTM (Long Short-Term Memory) model is trained using the tokenized data.
4. [Analysis After Training](https://chat.openai.com/#18)
   * Evaluation and analysis are performed on the trained model's performance and results.