Phase 5 Capstone Project Proposal

Project Title: Fake News Detection

Group: 2B

Technical Mentor: Samuel Karu

Collaborators:

Abdirahman Yussuf

Daniel Mwaka

Eliana Kariuki

Gerald Wanjala

Marion Osong

1. Business Understanding

In today's information-driven society, the spread of misinformation and fake news through online news dissemination platforms and social media platforms has become a serious threat to public trust, democracy, health communication, and media integrity. The ability to automatically distinguish **fake** from **real** news is essential for:

- Online news dissemination platforms (e.g., dailymail, foxnews, etc..)
- Social media platforms (e.g., Meta, Twitter/X, Reddit).
- Fact-checking organizations.
- News aggregators (e.g., Google News, Apple News).
- The general public.

This project proposes building a **binary classification system** that flags fake news articles using the latest Natural Language Processing (NLP) techniques. The proposed project aims to support information legitimacy across news articles disseminated across web platforms. The findings, deduced recommendations, and suggested next steps from undertaking this project are anticipated to make substantial contributions in leveraging transformer models for NLP via transfer learning frameworks.

2. Data Understanding

2.1 Metadata

We use 4 labelled datasets (gossipcop_fake.csv, gossipcop_real.csv, politifact_fake.csv, and politifact_real.csv) cloned from https://github.com/KaiDMML/FakeNewsNet/tree/master/dataset. Each of the four datasets contain the following metadata:

- **id:** Unique identifier for each news article.
- **news_url:** The URL link for a news article.
- **title:** A news article's title.
- **tweet _id:** twitter profile id that shared the link to the social media platform.

2.2 Web Scrapping

We formulate a Python script ***(scraper.py)*** to scrape text data from the URL link of each entity across the 4 datasets. The script comprises:

- The **fetch_article_text(url)** function: Sends a GET request to each URL using a browser-mimicking user-agent to avoid request blocks. If successful, it parses the HTML to extract paragraph (<p>) tags and combines their text content into a single article body. In case of request failures or timeouts, appropriate exceptions are handled gracefully.
- The **scrape_dataset** function: Reads the CSV file into a DataFrame and applies fetch_article_text to each URL using ***tqdm*** for progress tracking. The extracted article text is stored in a new column called extracted_article_text. After processing, the enriched DataFrame is saved to a new CSV file.
- The **main() function:** Loops through all four datasets, checks if they exist, and applies the scraping routine sequentially with a short delay to avoid overwhelming servers. The modular approach ensures maintainability, robustness, and extensibility.

2.3 Compiled Data

Feature Engineering is performed on each of the four datasets to create a ***news_type*** feature (*gossip* or *political)*. The target variable (***class***) is label encoded:

0: Fake (Misinformation)

1: Real (Credible)

The four datasets are concatenated to yield the working data (16044, 5).

- **Features:** news_url, title, extracted_article_text, news_type
- **Target:** class

3. Data Preparation

3.1 Preprocessing

- Lowercasing, punctuation, regex, and stopword removal
- Lemmatization
- Tokenization (WordPiece for transformers)
- Label encoding

3.2 Text Representation (Vectorization)

- For baseline models: TF-IDF, Word Embeddings (Word2Vec).
- For deep learning: Tokenizer and pad_sequences for LSTM.
- For transfer learning: RoBERTa Tokenizer and Contexual Embeddings.

4. Modeling

Baseline Models:

- Logistic Regression
- XGBoost

Deep Learning Models:

- LSTM (Long Short-Term Memory)

Transfer Learning:

- RoBERTa

5. Evaluation

- Accuracy
- Precision
- Recall
- F1-score
- Confusion Matrix

Special emphasis on ***Precision*** and ***Recall t***o minimize false positives (real news marked fake) and false negatives (fake news missed).

6. Deployment

- **FastAPI:** Input validation, preprocessing incoming text, calling model serving framework, post-processing results.
- **Docker:** Containerize API code, tuned RoBERTa model, and python dependencies into a self-contained unit.
- **Streamlit:** Deploy an interactive web application that presents the results in the form of a dashboard. The app must allow users to: input a news URL, receive prediction: “Fake” or “Real”. see confidence score and explanation (e.g., using SHAP or attention weights).

7. Tools/ Methodologies

Programming Language:

- Python
- Jupyter Notebooks

Data Manipulation:

- Numpy
- Pandas

Data Processing:

- NLTK
- PyPI
- spaCy
- Regular Expressions
- String module

Visualization:

- MatPlotLib
- Seaborn

Machine Learning Utilities:

- **Scikit-learn:** Data splitting, performance metrics , confusion matrix, baseline models.
- **XGBoost:** Ensemble baseline models

Deep Learning Utilities:

- **Deep Learning Framework:** Tensorflow 2.x
- **Transformers:** Loading RoBERTa models and tokenizers.

Model Explainability & Interpretation:

- **LIME:** Explaining individual predictions.
- **SHAP:** Computing feature attributions and providing deeper model interpretability.

Deployment:

- **FastAPI:** Input validation, preprocessing incoming text, calling model serving framework, post-processing results.
- **Docker:** Containerize API code, tuned RoBERTa model, and python dependencies into a self-contained unit.
- **Streamlit:** Deploy an interactive web application.

Development Environment & Version Control:

- **Jupyter Notebooks / Google Colab:** Initial exploration, prototyping, and model training.
- **Git & GitHub**: Version control and collaborative development.
- **Conda/ pip:** Managing project dependencies and ensuring a reproducible environment.