AirPods Review Sentiment Analysis Project Documentation

Problem Statement: The goal of this project is to analyse and classify customer reviews of Apple AirPods Pro into predefined sentiment categories using Natural Language Processing (NLP) and deep learning techniques. Automating sentiment classification helps businesses understand customer feedback, detect dissatisfaction early, and enhance user experience.

Methods

1. **Data Loading & Cleaning**
    - Loaded the AirPods reviews dataset (AirPodsPro_Reviews.csv) using **pandas**.
    - Removed duplicates and missing entries in the Review Text column.
    - Reset indices after cleaning for consistency.

- **Sentiment Labelling**
    - Implemented a **rule-based sentiment labelling function**:
        - Positive sentiment identified by words such as *good, great, amazing, excellent, love, awesome, best*.
        - Negative sentiment identified by words like *bad, poor, terrible, hate, worst, disappointing*.
        - Remaining reviews labelled as **neutral**.
- **Exploratory Data Analysis (EDA)**
    - **Class Distribution:** Plotted sentiment class counts to visualize imbalance.
    - **Review Length Distribution:** Histogram of review lengths (word counts).
    - **Word Clouds:** Generated word clouds for positive and negative reviews.
- **Text Preprocessing**
    - Converted reviews to lowercase, removed punctuation and HTML tags, tokenized text, and filtered out irrelevant characters.
    - Built a vocabulary of words appearing at least twice.
    - Encoded each review as a sequence of integers, with <PAD> and <UNK> tokens.
- **Label Encoding**
    - Encoded Sentiment labels into numeric values using **Label Encoder**.
- **Train-Test Split**
    - Divided dataset into **80% training** and **20% testing** using train_test_split.
    - Maintained class balance via stratify=y.

- **Modelling**
    - **Simple Feedforward Neural Network (Baseline):**
        - Used an embedding layer followed by a fully connected layer.
        - Represented each review as the mean of its embeddings.
    - **LSTM Classifier:**
        - Used a **bidirectional LSTM** for sequential modelling.
        - Captured context from both forward and backward directions.
    - Implemented models in **Torch** with **Adam optimizer** and class-weighted **Cross Entropy Loss**.
- **Evaluation**
    - Metrics: **Accuracy, Precision, Recall, F1-Score**.
    - Generated **classification reports** and **confusion matrices** for visual evaluation.

Insights

- The dataset is **imbalanced**, with more neutral reviews than positive reviews, affecting performance.
- The **Simple Neural Network** achieved **80% accuracy**, slightly outperforming the LSTM (75%).
- Word Clouds highlighted distinct positive and negative keywords, aiding interpretability.
- Sequential modelling (LSTM) captured context but suffered due to class imbalance and limited data size.
- Using embeddings instead of bag-of-words improved generalization and model performance.

Model Results

Simple Neural Network

Accuracy: 85

Precision: 0.4250

Recall: 0.5000

F1 Score: 0.4595

LSTM Classifier

Accuracy: 70

Precision: 0.5333

Recall: 0.5490

F1 Score: 0.5312

Challenges

- **Class Imbalance:** Few positive reviews caused lower precision and recall for minority classes.
- **Small Dataset:** Limited review samples restricted the potential of deep learning models.
- **No Negative Samples in Some Runs:** Some Word Clouds couldn't be generated due to missing negative reviews.
- **Hyperparameter Sensitivity:** Learning rate, hidden dimension, and sequence length affected stability.
- **Generalization:** Further data augmentation or pretrained embeddings like GloVe could improve performance.