Short Project Report — Sentiment Analysis Using DistilBERT

1. Dataset Description

For this project, I used a sentiment analysis dataset containing text reviews labeled as positive, negative, or neutral. The dataset included user-generated content such as product reviews and comments. Before training, I applied a preprocessing pipeline involving tokenization, stopword removal, lemmatization, and text normalization to clean the data.

2. Model Used

I used DistilBERT (distilbert-base-uncased) for fine-tuning. DistilBERT is a lighter and faster version of BERT while still retaining around 95% of BERT's performance, making it suitable for training on Google Colab. I used the DistilBertForSequenceClassification architecture with a classification head designed for sentiment prediction.

3. Training Setup

Platform: Google Colab

Library: HuggingFace Transformers (v4.57.1)

Training Arguments:

- Epochs: 3–5

- Learning rate: 5e-5

- Batch size: 16

- Weight decay: 0.01

- Warmup ratio: 0.1

Optimizer: AdamW

Loss Function: Cross-entropy

Tokenizer: DistilBERT tokenizer (uncased)

I performed training using the Trainer API, which handled batching, gradient updates, checkpoint saving, and evaluation automatically.

4. Evaluation Metrics

To evaluate my model, I used:

- Accuracy – overall correctness

- Precision – correctness of predicted positives

- Recall – percentage of actual positives detected

- F1-Score – balance between precision and recall

- Loss curves – for monitoring training and validation loss

My fine-tuned model achieved strong accuracy and a good F1-score, showing it generalizes well on unseen data.

5. Challenges and Observations

A major challenge was limited GPU availability in Colab, which sometimes slowed training. DistilBERT initially showed warnings because the classifier layer was randomly initialized, requiring full fine-tuning. I faced issues with W&B; logging, which required disabling or reconfiguring it. Ngrok produced authentication errors during deployment attempts. Despite these issues, the model trained well and consistently improved across epochs. I observed that the model performs best with balanced datasets and sometimes struggles with very long or ambiguous text.