**AI Text Detection Model Development Process**

**Training Process:**

1. **Data Preparation**:
   * Loaded dataset (Training\_Essay\_Data.csv) with labeled text samples (AI-generated vs human-written)
   * Applied preprocessing: converting to lowercase, removing special characters, removing extra spaces
   * Split data (80% training, 20% testing) with stratified sampling
2. **Text Tokenization**:
   * Used Keras Tokenizer with vocabulary limited to 20,000 most frequent words
   * Converted texts to numerical sequences
   * Padded sequences to 500 tokens maximum length
3. **Model Architecture** - Hybrid Neural Network:
   * Embedding layer (128 dimensions)
   * CNN branch with 1D convolution for local patterns
   * Bidirectional LSTM branch for sequential patterns
   * Merged branches with dense layers and dropout
   * Final sigmoid activation for binary classification
4. **Training Configuration**:
   * Adam optimizer
   * Binary cross-entropy loss
   * 10 epochs with batch size of 64
   * Metrics: accuracy, precision, recall
5. **Model Evaluation**:
   * Test accuracy: 99.57%
   * Test precision: 99.57%
   * Test recall: 99.36%
   * Comprehensive classification report showing excellent performance on both classes

**API Creation Process:**

Based on your comment about transforming into Flask, you likely:

1. Created a Flask application
2. Loaded the saved model and tokenizer
3. Defined API endpoints to accept text input via POST requests
4. Implemented the prediction function to process input and return classification results
5. Added error handling and response formatting

**Plagiarism Detection System**

For plagiarism detection, a separate system was implemented that:

1. Takes an essay as input
2. Splits long text into ~500 word chunks
3. Performs Google searches using Google Custom Search API
4. Scrapes content from the search results
5. Calculates similarity using TF-IDF vectorization and cosine similarity
6. Generates a plagiarism report with scores and levels

**Tools & Libraries Used:**

**AI Text Detection:**

* pandas & numpy: Data manipulation
* tensorflow/keras: Neural network modeling
* scikit-learn: Data splitting, metrics
* nltk: Text processing
* matplotlib & seaborn: Visualization
* pickle: Saving tokenizer
* re: Regular expressions

**Plagiarism Detection:**

* requests: API calls and web scraping
* BeautifulSoup: HTML parsing
* scikit-learn: TF-IDF vectorization, cosine similarity
* Google Custom Search API

**Dataset:**

* The AI detection model used "Training\_Essay\_Data.csv" containing:
  + Human-written text (class 0): 3,502 samples
  + AI-generated text (class 1): 2,327 samples
  + Total: 5,829 samples

**Model Evaluation Results:**

* Test Loss: 0.0176
* Test Accuracy: 99.57%
* Test Precision: 99.57%
* Test Recall: 99.36%
* Classification Report shows nearly perfect scores (F1-score ~0.99 for both classes)

**Limitations:**

1. The AI detection model:
   * Only processes text in English/Latin alphabet
   * Removes all special characters including numbers and punctuation
   * Converts all text to lowercase
   * Has a maximum sequence length of 500 tokens
2. The plagiarism detection system:
   * Relies on Google API with rate limits
   * Limited by the quality and relevance of search results
   * May struggle with highly paraphrased content
   * Simple TF-IDF approach may miss semantic similarities

Then transform to a Flask API allows both systems to be accessed programmatically, making them suitable for integration into larger applications like essay evaluation platforms.

LINKS:

Plagia and scoring [**https://colab.research.google.com/drive/1dMYtqhl1hQ6PdPChP\_l4gOF0eTYI309K**](https://colab.research.google.com/drive/1dMYtqhl1hQ6PdPChP_l4gOF0eTYI309K)

Ai detection

[**https://colab.research.google.com/drive/16yn7lFf-y66-i36BSVnvukAgpw8yJFlh**](https://colab.research.google.com/drive/16yn7lFf-y66-i36BSVnvukAgpw8yJFlh)

Appendices: