AI DEVELOPER TECHNICAL ASSESSMENT DOCUMENTATION

Develop a text classifier to distinguish Greeklish (Greek written in Latin characters) from standard English sentences using independently scraped data sources.

1. CHOICE OF DATA SOURCES

Greeklish Sources (3 distinct platforms):

1. **Reddit (r/greece)**: Posts and comments containing the term "greeklish" are scraped and filtered through a custom is_valid_greeklish() function to ensure the presence of common Greeklish words and the absence of Greek/Cyrillic script.
2. **Insomnia.gr Forum**: Forum content is scraped via BeautifulSoup targeting <p> tags. Posts are validated with the same function to filter only Greeklish content.
3. **YouTube Comments** (video ID: _akH1Bns2B8): Comments are scraped from a YouTube page, targeting <yt-formatted-string> tags. Filtered using the Greeklish validation function.

English Sources (2 distinct platforms):

1. **Reddit (r/AskReddit)**: Top posts and their comments are scraped using praw, then split into sentences using NLTK's sentence tokenizer.
2. **Wikipedia (NLP article)**: Scrapes paragraph content from the AI article on Wikipedia. Sentences are extracted from <p> tags and citations are stripped.

2. DATA SCRAPING AND PREPROCESSING

Scraping Methods:

- All sources use requests, BeautifulSoup, or praw for scraping.
- Comments, posts, and paragraphs are extracted and filtered using the is_valid_greeklish() function.
- Greeklish detection logic checks for:
    - Latin characters
    - Absence of Greek/Cyrillic script
    - Presence of ≥2 known Greeklish words (e.g., "kaneis", "sou", "thelw")

Preprocessing Steps:

1. **Sentence Splitting**: Long texts are split using ., !, ? into individual sentences.
2. **Lowercasing**: Converts all text to lowercase.
3. **Character Cleaning**: Removes special characters and digits using regex.
4. **Tokenization**: Uses NLTK to tokenize each sentence.

5. **Stopword Removal**: Removes common English stopwords.
6. **Reconstruction**: Remaining tokens are rejoined into cleaned sentences.

Output: A shuffled CSV file dataset.csv with two columns: sentence, label (either "Greeklish" or "English").

3. MODEL SELECTION AND TRAINING

Model Chosen*:* Logistic Regression

- **Why**: Lightweight, interpretable, effective for binary text classification with TF-IDF features.
- **Settings**: max_iter=1000, random_state=42 for reproducibility.

Feature Engineering*:*

- TF-IDF Vectorizer (max 5000 features)
- Trained on 80% of data, tested on 20% using stratified sampling to maintain class balance.

Evaluation Metrics*:*

- Accuracy
- Precision (weighted)
- Recall (weighted)
- F1-score (weighted)

Results*:*

Model Performance:

Accuracy: 0.9322

Precision: 0.9418

Recall: 0.9322

F1-Score: 0.9327

4. MODEL EXPORT AND PREDICTION

Model Persistence*:*

- Trained model and TF-IDF vectorizer are saved using joblib:
    - model/greeklish_classifier.pkl
    - model/tfidf_vectorizer.pkl

Prediction Function*:*

- predict_text(text) function loads the model and vectorizer, preprocesses input text, and returns a label prediction.
- Example usage:

predict_text("ti kaneis") → Greeklish

predict_text("Hello, how are you?") → English

5. CHALLENGES AND SOLUTIONS

1. ***Greeklish Detection****:*
    - Mixed text and inconsistent writing styles.
    - Solved using a regex + keyword matching filter.
2. ***Data Imbalance****:*
    - Some sources yielded fewer usable sentences.
    - Sentence splitting expanded samples.
3. ***YouTube Limitations****:*
    - Scraping comments via requests may return nothing due to JavaScript rendering.
    - Future work: use YouTube Data API or Selenium.
4. ***Reddit PRAW Warning****:*
    - Solved with check_for_async=False parameter to suppress async warnings in Colab or Jupyter.

6. CONCLUSION

The final solution uses real-world scraped data, an effective keyword and script-based filter for Greeklish, and a simple but powerful classifier. It performs with 93% accuracy, is reproducible, and can be extended with more data or alternate models.