**Test: Sentiment Analysis on a Custom Text Dataset**

**Objective:** The goal of this test is to assess the candidate's ability to perform sentiment analysis on a given dataset of text reviews using Python and NLP techniques. The candidate should build a model to classify the sentiment of the reviews as positive, negative, or neutral.

**Instructions:**

1. **Environment Setup:**
   * Create a new environment in Anaconda with the necessary libraries (e.g., numpy, pandas, scikit-learn, nltk, spacy, tensorflow or pytorch for deep learning models, etc.).
   * Use Jupyter Notebook for all coding and documentation.
2. **Dataset:**
   * A CSV file (reviews.csv) will be provided, containing two columns: review\_id and text.
   * The text column contains the text of the review, and the task is to predict the sentiment.
3. **Tasks:**
   * **Data Exploration:**
     + Load the dataset and provide an overview of the data, including basic statistics, data types, and any initial observations.
     + Identify and handle missing data, if any.
     + Visualize the distribution of word counts, sentiment classes, and other relevant features.
   * **Text Preprocessing:**
     + Implement text preprocessing steps such as tokenization, removing stopwords, stemming/lemmatization, and handling special characters.
     + Explain the rationale behind each preprocessing step in the notebook comments.
   * **Feature Engineering:**
     + Transform the text data into features suitable for machine learning models. This could include techniques like Bag of Words, TF-IDF, or word embeddings (Word2Vec, GloVe, etc.).
     + Justify the choice of feature engineering techniques.
   * **Model Building:**
     + Split the data into training and test sets.
     + Develop at least two different models (e.g., Logistic Regression, Naive Bayes, SVM, or a simple neural network) to classify the sentiment.
     + Use cross-validation to tune hyperparameters and improve model performance.
     + Evaluate the models using appropriate metrics (accuracy, precision, recall, F1-score, etc.).
   * **Model Interpretation:**
     + Discuss the performance of the models. Which model performed best and why?
     + Provide an interpretation of the model’s predictions using techniques such as feature importance or SHAP values.
   * **Conclusion and Recommendations:**
     + Summarize your findings, discussing the challenges faced and how they were addressed.
     + Provide recommendations for further improvement or potential next steps.
4. **Deliverables:**
   * A well-documented Jupyter Notebook file containing:
     + The code implementation with comments explaining each step.
     + Visualizations and results from each stage of the process.
     + A final section summarizing the approach and findings.
   * The Anaconda environment file (environment.yml) for reproducing the setup.
5. **Evaluation Criteria:**
   * **Technical Proficiency:** Ability to correctly implement NLP techniques and use Python effectively.
   * **Code Quality:** Clear, well-organized code with comments explaining the logic and methodology.
   * **Problem-Solving Approach:** Ability to address the problem in a systematic and thoughtful manner.
   * **Creativity:** Novel approaches to feature engineering, model selection, or interpretation.
   * **Communication:** Clarity in presenting results and discussing findings.

**Notes:**

* The candidate is encouraged to use both traditional machine learning models and more advanced deep learning models if appropriate.
* Creativity in feature engineering and model interpretation will be valued.
* The candidate should be prepared to discuss their approach in a follow-up interview.

This test will help you evaluate the candidate's technical skills in Python and NLP, as well as their problem-solving abilities, attention to detail, and communication skills.