**Assignment: Sentiment Analysis Agent**

**Functional Requirements**  
The goal of this assignment is to design and implement a machine learning agent that performs sentiment analysis on a dataset of product reviews. The solution will classify reviews into positive, negative, or neutral sentiments, leveraging a robust ML model. This document provides the details of the tasks, input, output, and functional steps required to complete the assignment.

**Key Functional Requirements**

**1. Data Ingestion and Preprocessing**

**Input:**

* A dataset of product reviews containing:
  + **Fields:**
    - review\_text: The textual content of the product review.
    - sentiment (optional): The labeled sentiment (positive, negative, neutral) for supervised learning.
  + **Format:** CSV, Excel, or JSON file with at least 500 reviews.

**Processing Steps:**

1. Load the dataset into your application using libraries like pandas or json.
2. Clean the data:
   * Handle missing or null values in the review\_text field.
   * Remove special characters, extra spaces, URLs, and HTML tags from the reviews.
   * Convert text to lowercase for uniformity.
3. Tokenize and preprocess text:
   * Use libraries like NLTK or spaCy for:
     + Tokenization
     + Lemmatization
     + Stop word removal.
4. Vectorize the preprocessed text using one of the following:
   * Bag-of-Words (CountVectorizer)
   * TF-IDF Vectorizer
   * Word embeddings (e.g., Word2Vec or GloVe for advanced learners).

**Output:**

* A cleaned, tokenized, and vectorized dataset ready for model training.

**2. Model Development**

**Input:**

* Preprocessed dataset with review\_text (features) and sentiment (labels, if supervised).

**Processing Steps:**

1. Split the dataset:
   * Use an 80-20 or 70-30 split for training and testing.
2. Train a sentiment classification model using one or more algorithms:
   * Logistic Regression
   * Random Forest
   * Naïve Bayes
   * Basic Neural Network (optional for advanced learners).
3. Tune hyperparameters using GridSearchCV or RandomSearchCV for better performance.
4. Evaluate the model on the test set using:
   * Accuracy
   * Precision
   * Recall
   * F1-score

**Output:**

* A trained sentiment classification model with evaluation metrics and an optimized set of hyperparameters.

**3. Model Evaluation**

**Input:**

* Test dataset and trained model.

**Processing Steps:**

1. Predict sentiments for the test dataset using the trained model.
2. Generate evaluation metrics:
   * Accuracy, Precision, Recall, F1-score.
3. Create confusion matrix visualizations to analyze performance.
4. For advanced learners: Perform cross-validation and report the average metrics.

**Output:**

* Model evaluation results and visualizations (e.g., confusion matrix, precision-recall curve).

**4. Deployment as a REST API**

**Input:**

* Trained sentiment classification model.

**Processing Steps:**

1. Use Flask or FastAPI to create a REST API for the sentiment analysis agent.
2. Define endpoints:
   * /predict: Accepts a text input (product review) and returns the predicted sentiment.
   * /health: A health-check endpoint to confirm the API is running.
3. Example workflow for the /predict endpoint:
   * Input: JSON object with a review\_text field.
   * Output: JSON response with predicted sentiment (positive, negative, neutral) and confidence score.
4. Save the trained model using joblib or pickle and load it in the API backend.

**Output:**

* A REST API capable of classifying product review sentiments in real-time.

**5. Documentation**

**Input:**

* Entire implementation process, challenges faced, and results achieved.

**Processing Steps:**

1. Document the following:
   * Objective and approach of the project.
   * Data preprocessing steps and tools used.
   * Model training, hyperparameter tuning, and evaluation.
   * API design and deployment process.
   * Challenges faced and how they were resolved.
2. Include sample inputs and outputs for both the trained model and REST API.

**Output:**

* A comprehensive report explaining the steps, challenges, and results.

**Expected Outputs**

1. **Model:**
   * A trained sentiment classification model achieving at least 80% F1-score on the test dataset.
2. **REST API:**
   * A functional REST API with endpoints for sentiment prediction and health checks.
3. **Evaluation Metrics:**
   * Accuracy, Precision, Recall, F1-score, and a confusion matrix.
4. **Documentation:**
   * A detailed report of the project workflow, challenges, and solutions.

**Evaluation Criteria**

1. **Accuracy:**
   * The model should perform well on test data with at least 80% accuracy or F1-score.
2. **Completeness:**
   * All functional requirements (data preprocessing, model training, API deployment) should be implemented.
3. **Usability:**
   * The REST API should be user-friendly and handle edge cases like invalid or empty inputs gracefully.
4. **Code Quality:**
   * Clean, modular, and well-commented code.
5. **Presentation:**
   * Effective documentation with clear explanations and visualizations.

**Additional Notes**

* Use publicly available datasets, such as the Amazon Product Reviews dataset.
* Advanced learners may experiment with pre-trained models like BERT or GPT for sentiment analysis.
* Feel free to reach out if you need clarification or support.

Good luck, and happy coding! 😊