Project Report: Sentiment Analysis for Product Reviews

1. Project Problem Description

Problem Statement:

Online marketplaces like Amazon, Flipkart, or Daraz host millions of customer reviews every day.
These reviews provide valuable insights into customer satisfaction, product quality, and potential issues.However, manually analyzing such reviews is impractical due to scale.
The objective of this project is to build an end-to-end sentiment analysis system where users can
input a product review(or any text) and instantly receive its predicted sentiment (Positive/Negative), along with confidence Scores.

Why It Matters:

- For customers: Helps them quickly judge product reputation.
- For businesses: Allows automated monitoring of customer satisfaction and product feedback.
- For learners (like me): Demonstrates end-to-end application of machine learning, from data collection and preprocessing to model training and deployment with a frontend interface.

Dataset Details:

- Dataset Used: Amazon Reviews Polarity Dataset (from Hugging Face datasets library).
- Size: 3.6 million reviews (balanced between positive and negative).
- Classes: 0 = Negative, 1 = Positive.
- Sample: For faster training, a random subset of 100,000 reviews was used.
- Relevance: Directly matches our problem domain (product reviews), making the trained model
- more realistic.

2. Solution Overview

The project was implemented step by step as follows:

Step 1: Data Preprocessing

- Downloaded dataset using datasets library.

- Cleaned text by lowercasing, removing punctuation, and removing English stopwords (NLTK).
- Labeled data as 0 (Negative) and 1 (Positive).

Step 2: Feature Extraction

- Used TF-IDF Vectorizer to convert text into numerical features.
- Configured with max_features=10,000 to balance accuracy and training efficiency.

Step 3: Model Training

- Model: Logistic Regression (scikit-learn).
- Training set: 80% of dataset (80,000 reviews).
- Testing set: 20% (20,000 reviews).
- Achieved high accuracy (~90%) on test set.
- Saved trained model and vectorizer using joblib.

Step 4: Backend Integration

- Loaded saved model and vectorizer in a Python script.
- Added preprocessing function (must match training step).
- Added prediction function that outputs both label and confidence scores.

Step 5: Frontend with Gradio

Built a Gradio dashboard where users can input reviews and see predictions instantly.

3. Challenges and Learnings

Challenges:

1. Short Reviews Misclassification: Short texts sometimes misclassified due to limited context.
2. Preprocessing Consistency: Training and inference pipelines must match exactly.
3. Computational Constraints: Full dataset too large; had to use subset of 100k reviews.
4. Neutral Sentiment Handling: Dataset only had Positive/Negative; real-world reviews often include
   neutral opinions.

Learnings:

1. Simple models (Logistic Regression + TF-IDF) perform well with good preprocessing.
2. Lexicon-based methods like VADER complement ML models for short texts.
3. End-to-end ML requires both backend ML and frontend integration skills.
4. Explainability is important (token contributions helped debug misclassifications).

4. References

- Datasets: Amazon Reviews Polarity Dataset (Hugging Face)
- Libraries: scikit-learn, NLTK, Hugging Face Datasets, Gradio, Joblib

5. Project Source Code

https://github.com/arif123/ml-course-final-project