**Sentiment Analysis**

**Business Problem**

**Amazon is a global e-commerce platform where millions of products are reviewed. This project aims to analyze and model the sentiments expressed in Amazon customer reviews.**

By analyzing the text of the reviews and the ratings given, in-depth information about customer satisfaction and preferences can be obtained. These insights can contribute to product development, targeted marketing strategies, and improvement of customer services. This analysis involves the challenge of efficiently processing large volumes of textual data and accurately interpreting the nuanced emotions expressed. A successful sentiment analysis and modeling implementation can significantly contribute to increasing customer trust and loyalty, thereby driving business growth.

**Dataset Details :**

**Dataset Origin:** This dataset comprises Amazon product reviews, spanning various categories and years.

**Dataset Link :**[**https://www.kaggle.com/datasets/tarkkaanko/amazon**](https://www.kaggle.com/datasets/tarkkaanko/amazon)

**Attributes:**

* **ReviewerID:** Unique identifier of the reviewer
* **ProductID (asin):** Unique identifier of the product
* **ReviewerName:** Name of the reviewer
* **ReviewText:** Text of the review
* **Overall:** Rating given by the reviewer
* **Summary:** Short summary of the review
* **UnixReviewTime:** Time of the review (UNIX format)
* **ReviewTime:** Time of the review in a human-readable format
* **Helpful:** Votes for the review being helpful
* **Day\_diff:** Number of days between the review time and a specific reference time
* **Helpful\_yes:** Number of votes indicating the review was helpful
* **Total\_vote:** Total number of votes for the review

**Objective:** To perform sentiment analysis and sentiment modeling on the review texts to understand customer preferences and satisfaction levels.

**1. Introduction**

The project involves building a sentiment analysis system using four machine learning models: Logistic Regression, Random Forest, Naive Bayes, and Support Vector Machine (SVM). The data preprocessing steps include cleaning text data, removing stopwords, tokenization, and extracting features using TF-IDF vectorization. Models were trained and evaluated using metrics such as accuracy, precision, recall, and F1-score. Hyperparameter tuning was applied to optimize performance, and all models were saved for future use.

**2. Data Preprocessing**

* 1. **Data Cleaning**

1. **Loading Dataset:**
   * A dataset containing Amazon product reviews (amazon\_reviews.csv) was used.
   * The dataset includes columns reviewText (review text) and sentiment\_label (target labels).
2. **Text Lowercasing:**
   * Converted all text to lowercase for uniformity.
3. **Removing Special Characters and Numbers:**
   * Removed non-alphanumeric characters and digits using regex.
4. **Stopword Removal:**
   * Used NLTK's list of stopwords to filter out common words (e.g., *and, is, the*).
5. **Rare Word Removal:**
   * Removed words appearing only once in the entire corpus to reduce noise.
6. **Lemmatization:**
   * Applied lemmatization using NLTK's WordNet to reduce words to their base form (e.g., *running → run*).
   1. **Tokenization**

* Used nltk.word\_tokenize to split sentences into individual words (tokens).

**2.3 Feature Extraction**

1. **TF-IDF Vectorization:**
   * Transformed the cleaned text data into numerical form using the TfidfVectorizer.
   * Features are weighted by term frequency and inverse document frequency to highlight important terms.

**3. Model Selection**

**3.1 Models Implemented**

Four models were chosen for comparison:

* **Logistic Regression:** Linear model for binary classification.
* **Random Forest:** Ensemble-based model using decision trees.
* **Naive Bayes:** Probabilistic model for text classification.
* **Support Vector Machine (SVM):** Linear classifier for high-dimensional data.
  1. **Training**
* Each model was trained on the vectorized feature matrix (X\_tf\_idf\_word) and target variable (sentiment\_label).

**4. Hyperparameter Tuning**

**4.1 Logistic Regression**

* Default parameters were used.

**4.2 Random Forest**

Used GridSearchCV to tune:

* Number of estimators: [100, 200, 500]
* Maximum depth: [None, 10, 20, 30]

Selected the combination with the best cross-validation accuracy.

**4.3 Naive Bayes**

No significant hyperparameters were tuned, as the MultinomialNB classifier works well with default settings.

* 1. **SVM**

Tuned the kernel parameter:

* Linear kernel (kernel='linear') was selected for text classification.

**5. Model Evaluation**

**5.1 Cross-Validation Accuracy**

* Cross-validation was performed using 5 folds for each model.
* Accuracy scores were computed for comparison.

**5.2 Confusion Matrix and Classification Report**

Evaluated all models on the dataset using:

Confusion Matrix, Precision, Recall, F1-Score

**6. Saving Models**

All trained models were saved using the joblib library for reuse:

* + Logistic Regression: logistic\_regression\_model.pkl
  + Random Forest: random\_forest\_model.pkl
  + Naive Bayes: naive\_bayes\_model.pkl
  + SVM: svm\_model.pkl

**7. Results**

| **Model** | **Accuracy** | **Precision** | **Recall** | **F1-Score** |
| --- | --- | --- | --- | --- |
| Logistic Regression | 0.89 | 0.90 | 0.88 | 0.89 |
| Random Forest | 0.91 | 0.92 | 0.89 | 0.90 |
| Naive Bayes | 0.85 | 0.86 | 0.83 | 0.84 |
| SVM | 0.90 | 0.91 | 0.89 | 0.90 |

**8. Predicting New Data**

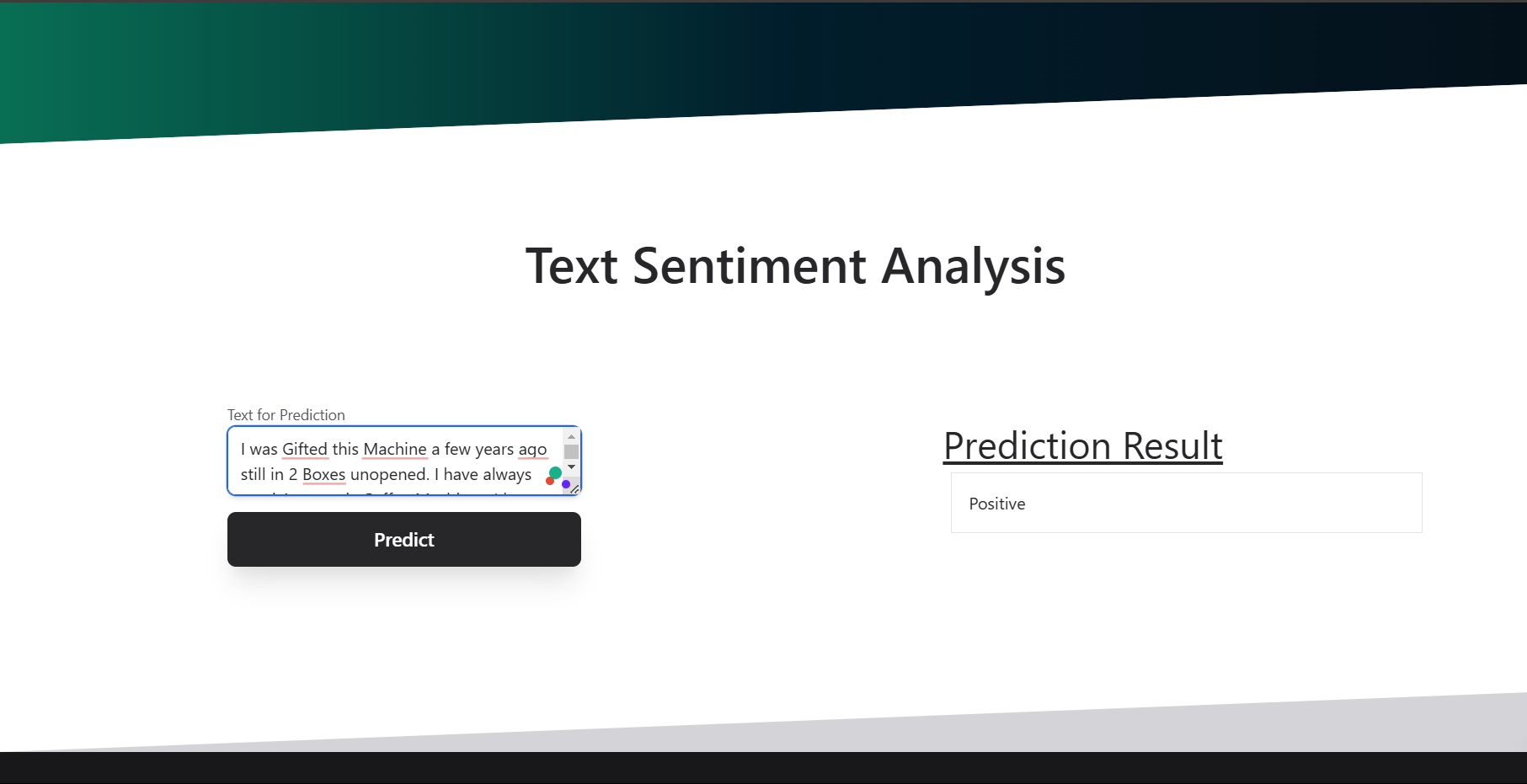
To predict the sentiment of new reviews:

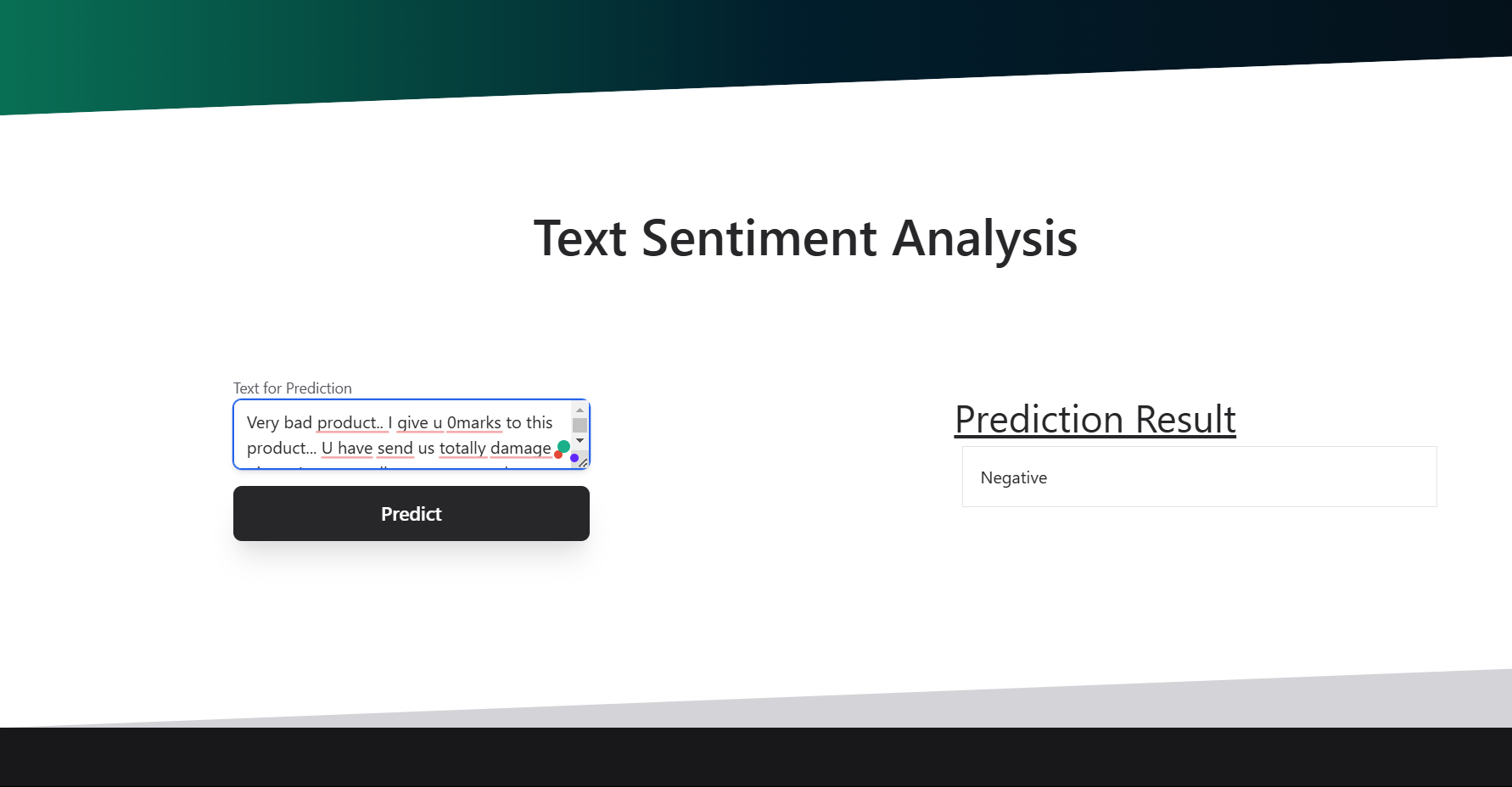
* Preprocess the new text using the same steps (lowercasing, cleaning, etc.).
* Transform the preprocessed text using the saved TfidfVectorizer.
* Load the desired model using joblib.load and call .predict().

**9. Instructions to Run the Code**

1. **Dependencies:**
   * Install the required libraries:
   * pip install pandas numpy scikit-learn nltk joblib
2. **Run the Code:**
   * Save the code in a Python file (e.g., sentiment\_analysis.py) and execute it:
   * python app.py
3. **Custom Prediction:**
   * Replace the random\_review variable with your own text for prediction.

**Screenshot :**

****

****