**Explanation of Emotion Classification Code with Time Complexity**

1. **Imports**

* pandas: Used for reading and manipulating the dataset.
* train\_test\_split: Splits the data into training and testing sets.
* TfidfVectorizer: Converts text into numerical vectors based on TF-IDF.
* LogisticRegression: The machine learning algorithm used.
* classification\_report: Provides precision, recall, F1-score, etc.
* joblib: Used for saving trained models and vectorizers.

1. **Load and Shuffle Dataset**

* Reads CSV containing 30,000 emotion-labeled sentences.
* sample(frac=1) shuffles the dataset.
* reset\_index(drop=True) resets index after shuffling.
* value\_counts() shows how many samples per label.
* Time Complexity:
* Reading CSV:O(n), n =no of rows
* Shuffling : O(n)

1. **Train – Test Split**

* Splits the dataset: 80% train, 20% test.
* stratify ensures class distribution is maintained
* Train: 24,000 samples
* Test: 6,000 samples
* **Time Complexity: O(n)**

1. **TF – IDF Vectorization**

* TF-IDF: Assigns weights to words based on frequency and importance across documents.
* max\_features=5000: Only keeps top 5,000 most relevant words.
* fit\_transform() learns vocab from train data and converts to sparse matrix.
* transform() only transforms test data using learned vocab.
* Time Complexity:
* fit\_transform: O(n × k), n = number of docs, k = avg tokens/doc
* transform: O(m × k) for m = test samples
* Data Shape:
* X\_train\_vec: (24000, 5000)
* X\_test\_vec: (6000, 5000)

1. **Model Training**

* Logistic Regression: Linear classifier using cross-entropy loss.
* max\_iter=1000: Max number of optimization steps.
* Time Complexity:
* Each iteration: O(n × d), n = training samples (24000), d = features (5000)
* Total: O(max\_iter × n × d) = O(1000 × 24000 × 5000) in worst case

1. **Model Evaluation**

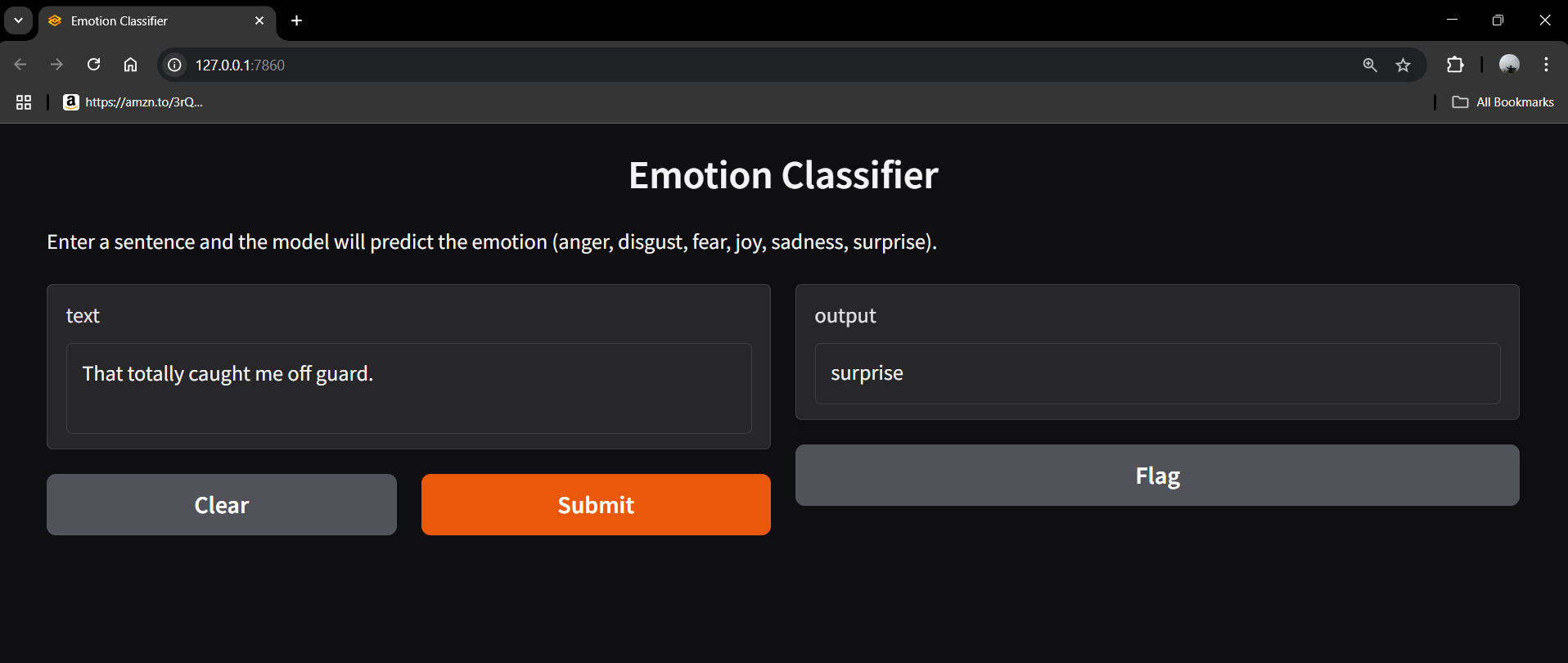
* Prediction: Calculates class probabilities and picks max.
* classification\_report: Outputs precision, recall, F1-score, and support for each emotion
* Time Complexity:
* Predict: O(m × d), m = test samples (6000), d = features (5000)
* classification\_report: O(m)

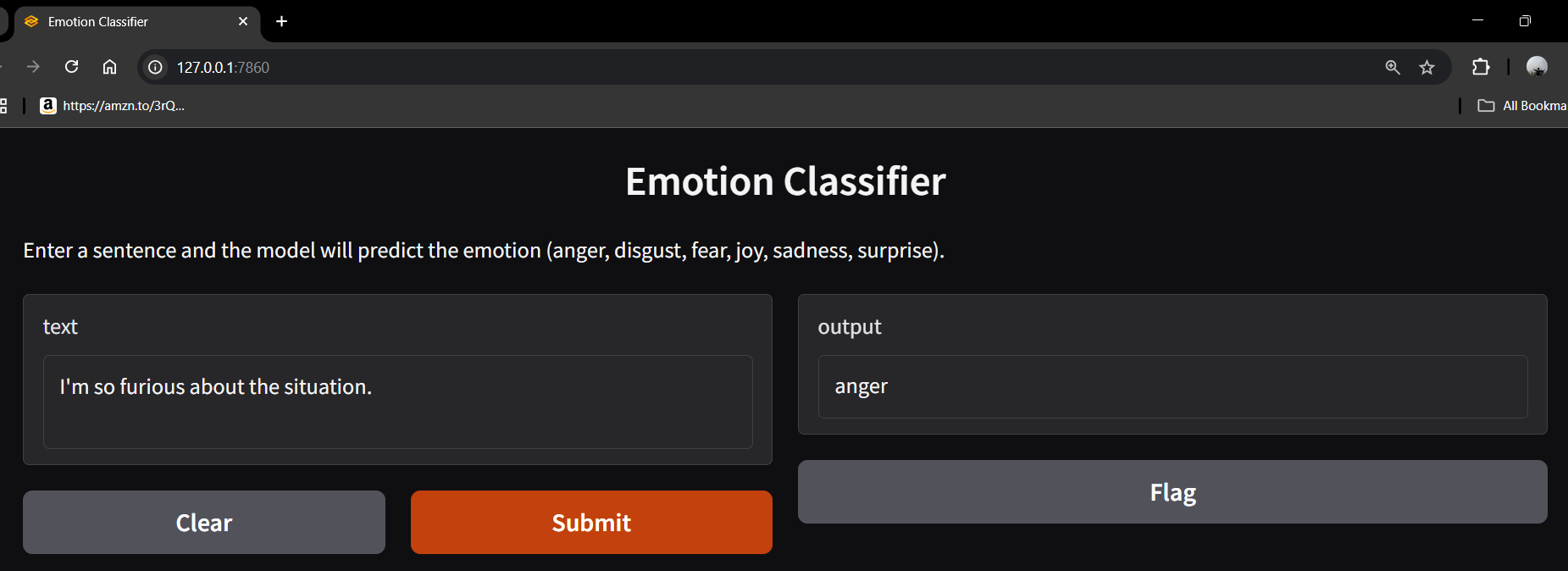
1. **Saving The Model**

* Saves trained model and vectorizer for later use
* Time Complexity:
* Saving object Size + model params: O(d)

1. **Gradio:**

|  |  |
| --- | --- |
| **Steps** | **Description** |
| Load Model | Load the trained emotion classifier and vectorizer |
| Input | User types a sentence |
| Transform | Convert text into TF-IDF features |
| Predict | Use Logistic Regression to classify emotion |
| Output | Display predicted emotion label in UI |

**9. Screenshots- **

****

