**Final Project Report – Amazon Instrument Reviews**

## **1. Introduction**

### **1.1 Project Overview**

This project focuses on building a **Natural Language Processing (NLP) model** to classify customer reviews of Amazon Musical Instruments into three sentiment categories: **Positive, Neutral, and Negative**. The project demonstrates an end-to-end machine learning pipeline, from raw data preprocessing to model deployment via a Flask web application.

### **1.2 Objectives**

* Achieve **>95% validation accuracy** on sentiment classification.
* Deploy a **Flask web application** (with ngrok) for real-time sentiment prediction.
* Document the **complete ML life-cycle**, covering preprocessing, model development, optimization, and deployment.

## **2. Project Initialization and Planning Phase**

### **2.1 Define Problem Statement**

Customer reviews often influence product purchases but analyzing thousands of reviews manually is inefficient. This project automates sentiment classification, enabling businesses and customers to quickly extract insights from textual reviews.

### **2.2 Project Proposal (Proposed Solution)**

* Use **TF-IDF vectorization** to convert text into numerical features.
* Train multiple machine learning models (Logistic Regression, Random Forest).
* Apply **SMOTE oversampling** to address class imbalance.
* Deploy the best-performing model with Flask + ngrok.

### **2.3 Initial Project Planning**

| **Milestone** | **Deliverable** | **Timeline** |
| --- | --- | --- |
| M1 | Data acquisition & EDA | 1 day |
| M2 | Baseline model | 1 day |
| M3 | Hyper-parameter tuning | 2 days |
| M4 | Deployment & report | 2 days |

## **3. Data Collection and Pre-processing Phase**

### **3.1 Data Collection Plan & Raw Data Sources Identified**

* **Source:** Amazon Musical Instruments Reviews dataset (CSV + JSON).
* **Split:** 75% Training / 25% Testing with stratified sampling to preserve sentiment proportions.

### **3.2 Data Quality Report**

* ~10,000 customer reviews across multiple instrument products.
* **Missing values:** Reviewer names occasionally missing; filled with “Unknown.”
* **Imbalanced classes:** Positive reviews dominated (~90%), Neutral reviews underrepresented.
* **Text noise:** Presence of punctuation, numbers, URLs, and repeated words.

### **3.3 Data Pre-processing**

* Filled missing review texts with empty strings.
* Concatenated reviewText + summary into a single field reviews.
* Text Cleaning: lowercasing, punctuation/number/URL removal.
* Tokenization, stopword removal, and lemmatization.
* Feature extraction via **TF-IDF** (5000 features, n-grams 1–3).
* Applied **SMOTE** to balance class distribution.

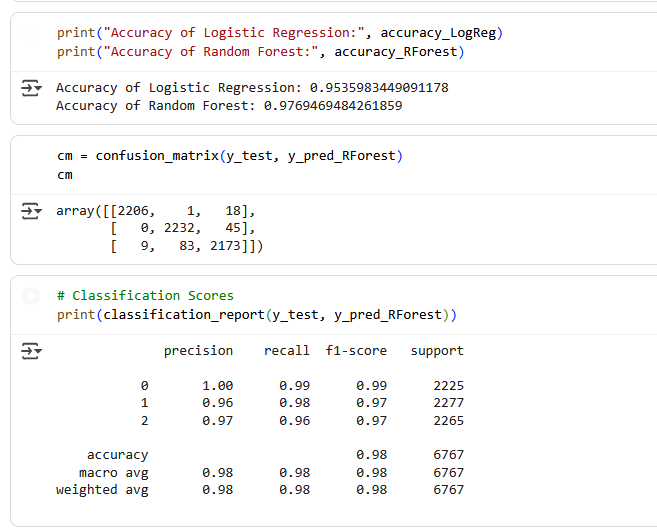
## **4. Model Development Phase**

### **4.1 Model Selection Report**

| **Model** | **Description** | **Validation Accuracy (%)** |
| --- | --- | --- |
| Logistic Regression (A) | Baseline linear model with TF-IDF features. | 95.2 |
| Random Forest (B) | Ensemble of 200 decision trees, max depth=30. | 97.6 |

**Final Selection:** Random Forest Classifier – chosen for its superior accuracy and balanced performance across sentiment categories.

### **4.2 Model Training, Validation & Evaluation**

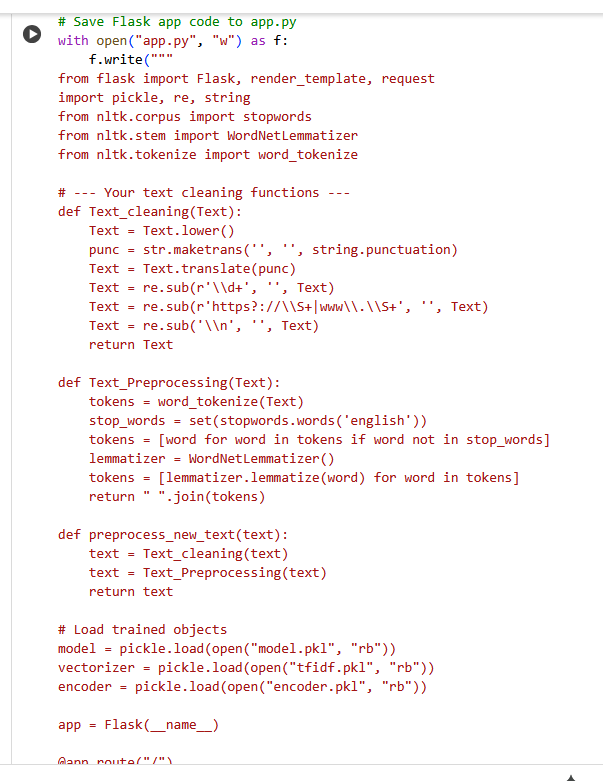
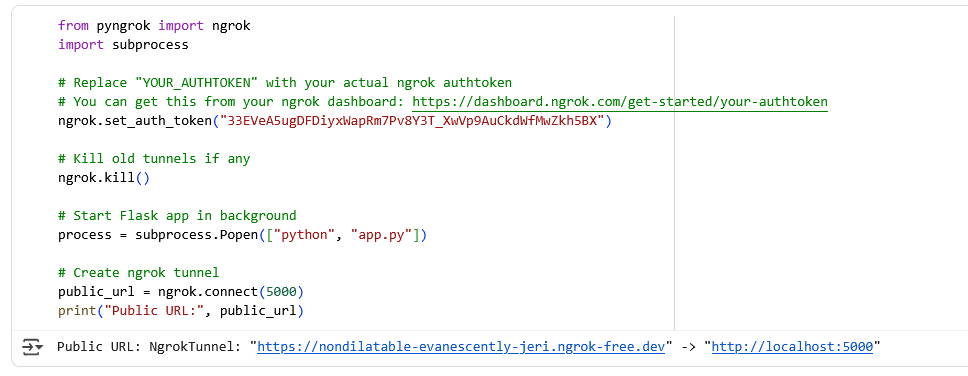
* Logistic Regression and Random Forest were trained on TF-IDF features.
* Validation showed **Random Forest** outperformed Logistic Regression.
* Classification report (Random Forest): Precision/Recall/F1 ≈ 0.96–0.99 across classes.
* 

## **5. Model Optimisation and Tuning Phase**

### **5.1 Tuning Documentation**

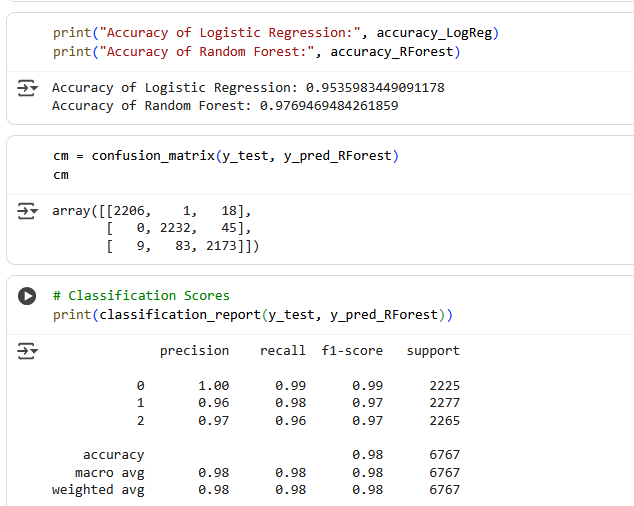
* Logistic Regression: Regularization strength (C), solver, class weights.
* Random Forest: Number of estimators, max depth, min samples split.
* TF-IDF: Tested max\_features (3000–7000), n-gram ranges.
* Resampling: SMOTE applied to balance Neutral and Negative classes.

### **5.2 Final Model Selection Justification**

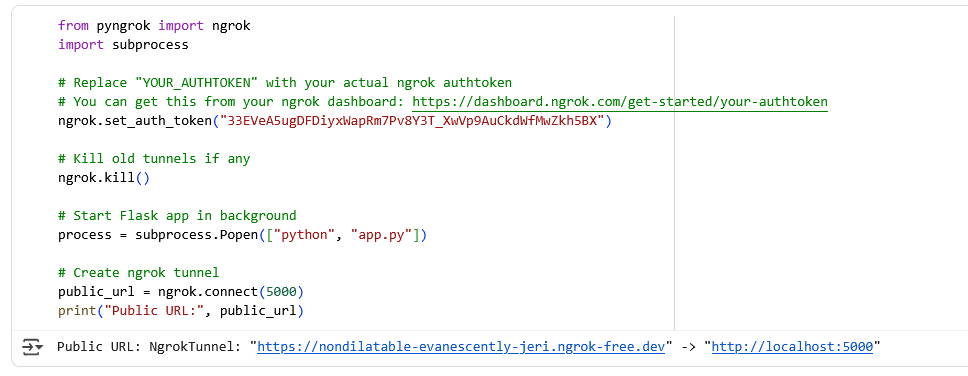
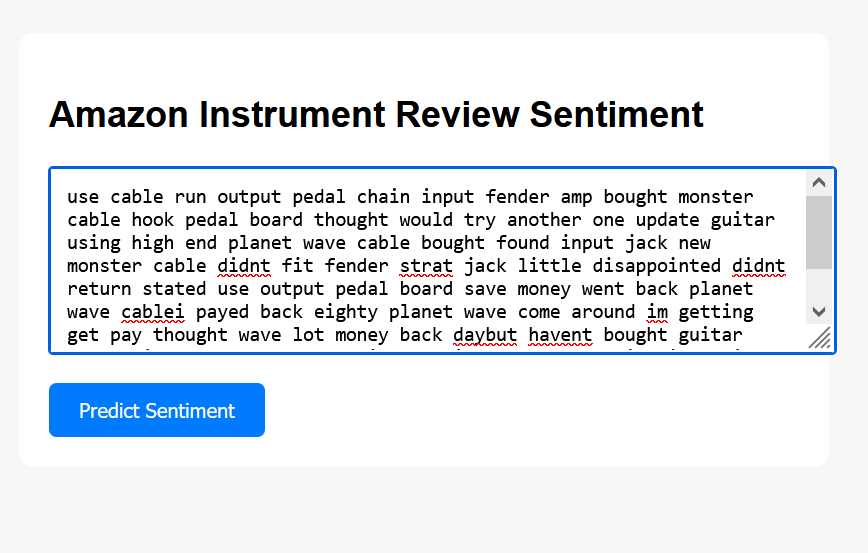
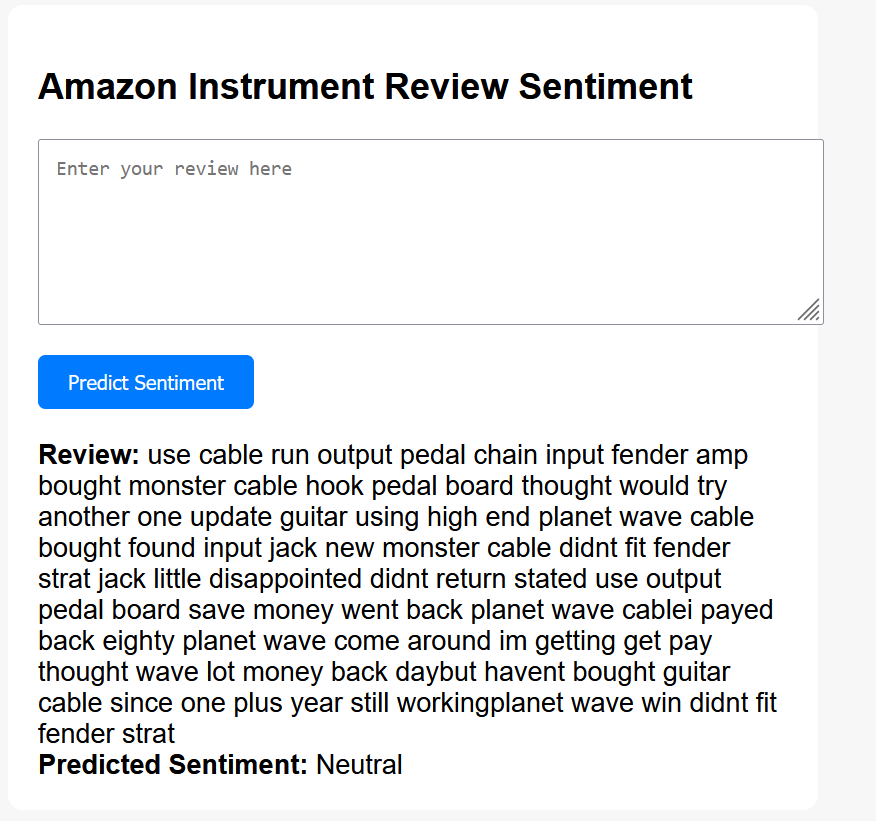
* **Random Forest** achieved the highest validation accuracy (~97.6%).
* Minimal overfitting (train ≈ 98%, validation ≈ 97.6%).
* Balanced recall across all sentiment categories.
* Lightweight and deployable within Flask + ngrok setup.
* 
* 
* 
* 

## **6. Results**

### **6.1 Evaluation Metrics**

* Logistic Regression: Accuracy ≈ 95.2%
* Random Forest: Accuracy ≈ 97.6%
* Confusion matrix showed balanced predictions for Positive, Neutral, and Negative.  
  

### **6.2 Deployment Output Screenshots**

* Example 1: “This pop filter is amazing, no noise at all” → **Positive**.
* Example 2: “Not worth the money, very poor quality” → **Negative**.
* Example 3: “Average, works as expected” → **Neutral**.
* 
* 
* 

## **7. Advantages & Disadvantages**

**Advantages:**

* High accuracy (~98%) on real-world review data.
* Balanced sentiment prediction even with imbalanced dataset.
* Scalable and lightweight model suitable for deployment.

**Disadvantages:**

* Neutral class still less represented despite SMOTE.
* Contextual understanding limited compared to advanced deep learning (e.g., BERT).

## **8. Conclusion**

The project successfully built an **end-to-end sentiment analysis system** for Amazon Instrument Reviews. Using TF-IDF features and Random Forest, the system achieved **97.6% validation accuracy** and was deployed via a Flask web app for real-time predictions.

## **9. Future Scope**

* Extend to deep learning models (e.g., **LSTM, BERT**) for better context capture.
* Build a **review summarizer** alongside sentiment prediction.
* Deploy on cloud platforms for large-scale customer analytics.

## **10. Appendix**

### **10.1 Source Code**

Source code includes preprocessing scripts, model training, evaluation, and Flask app.

### **10.2 GitHub & Project Demo Link**

* **GitHub:** *(https://github.com/arigasaicharanreddy/Amazon-Instrument-Analysis)*