# Restaurant Review Sentiment Classifier

This project implements a text classification model using Python and popular machine learning libraries to automatically determine the sentiment (Positive or Negative) of restaurant reviews. The classification is based on data extracted from a CSV file (scraped\_df.csv) which includes a pre-labeled column (hugging\_face\_label).

The initial analysis highlights a significant challenge: **severe class imbalance**, which heavily influences the model's performance.

## 🚀 Project Overview

The core of this project is a simple pipeline:

1. **Data Loading & Preprocessing:** Load the reviews and clean up the input data.
2. **Feature Extraction:** Convert raw text data into numerical features using **TF-IDF (Term Frequency-Inverse Document Frequency)**.
3. **Model Training:** Train a **Logistic Regression** model for binary classification.
4. **Evaluation:** Assess model performance using a Classification Report and Confusion Matrix.

## 🛠️ Requirements and Setup

To run this project, you need Python and the following libraries:

### Prerequisites

* Python 3.8+
* The data file scraped\_df.csv must be accessible (or placed in the project directory).

### Installation

Install the necessary dependencies using pip:

pip install pandas numpy scikit-learn matplotlib seaborn

### Data Structure

The classification relies on two key columns in scraped\_df.csv:

| **Column Name** | **Type** | **Description** |
| --- | --- | --- |
| Reviews | object (string) | The raw text of the restaurant review. |
| hugging\_face\_label | int (0 or 1) | The target variable: **0 for Negative** sentiment, **1 for Positive** sentiment. |

## 💻 Code Implementation (res-rev-class.py)

The main classification script is provided below. Note the use of **raw string** literal (r'...') for the file path on Windows to avoid OSError: [Errno 22] Invalid argument issues.

import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
from sklearn.model\_selection import train\_test\_split  
from sklearn.feature\_extraction.text import TfidfVectorizer  
from sklearn.linear\_model import LogisticRegression  
from sklearn.metrics import confusion\_matrix, classification\_report  
  
# 1. Load the data (Using a raw string path for compatibility)  
# dataset = pd.read\_csv(r'F:\My\_Projects\resturant-reviews-ml-project\scraped\_df.csv')   
# If running in the same directory, use:  
df = pd.read\_csv('scraped\_df.csv')  
  
# --- Data Preprocessing ---  
df.dropna(subset=['Reviews'], inplace=True)  
X = df['Reviews']  
y = df['hugging\_face\_label'].astype(int)  
  
# 3. Perform Train-Test Split (25% for testing, stratified)  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(  
 X, y, test\_size=0.25, random\_state=42, stratify=y  
)  
  
# 4. Vectorize the text data (TF-IDF)  
tfidf = TfidfVectorizer(stop\_words='english', max\_features=5000)  
X\_train\_vec = tfidf.fit\_transform(X\_train)  
X\_test\_vec = tfidf.transform(X\_test)  
  
# 5. Train a Classification Model (Logistic Regression)  
model = LogisticRegression(solver='liblinear', random\_state=42)  
model.fit(X\_train\_vec, y\_train)  
  
# 6. Make predictions  
y\_pred = model.predict(X\_test\_vec)  
  
# 7. Evaluate and Visualize Results  
  
# Classification Report  
print("\nClassification Report:")  
print(classification\_report(y\_test, y\_pred))  
  
# Confusion Matrix Visualization  
cm = confusion\_matrix(y\_test, y\_pred)  
plt.figure(figsize=(8, 6))  
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',  
 xticklabels=['Negative (0)', 'Positive (1)'],  
 yticklabels=['Negative (0)', 'Positive (1)'])  
plt.title('Confusion Matrix for Restaurant Review Classification')  
plt.xlabel('Predicted Label')  
plt.ylabel('True Label')  
plt.show()  
# plt.savefig('confusion\_matrix.png')

## 📈 Initial Results and Challenges

The initial model performed very poorly on the Negative class (0). This is a direct consequence of the data imbalance:

| **Class** | **Count** | **Percentage** |
| --- | --- | --- |
| **Positive (1)** | 587 | ~86% |
| **Negative (0)** | 92 | ~14% |

### Classification Report Summary (Example Result)

| **Metric** | **Negative (0)** | **Positive (1)** |
| --- | --- | --- |
| **Precision** | 0.00 | $\sim$0.86 |
| **Recall** | 0.00 | 1.00 |
| **F1-Score** | 0.00 | $\sim$0.93 |
| **Support** | 23 | 147 |

* **Interpretation:** A Recall of 0.00 for the Negative class means the model failed to correctly identify **any** of the actual negative reviews (it predicted them all as Positive). The model is biased toward predicting the majority class (Positive).

## 💡 Next Steps for Improvement

To create a useful classifier, the most crucial next step is to address the class imbalance.

1. **Class Balancing:** Implement techniques like **SMOTE (Synthetic Minority Oversampling Technique)** to create synthetic data points for the minority (Negative) class, or use **class weights** directly in the Logistic Regression model.
2. **Hyperparameter Tuning:** Use GridSearchCV or RandomizedSearchCV to find the optimal parameters for the LogisticRegression model.
3. **Alternative Models:** Test other classification algorithms that are often robust to high dimensionality and sparsity, such as **Support Vector Machines (SVM)** or a **Naive Bayes** classifier.