Restaurant Review Classifier

Project Description:

Restaurant Review Classifier is a machine learning project designed to automatically analyze and classify customer feedback for restaurants. The system reads text reviews from platforms like Yelp, Google Reviews or Zomato and predicts whether each review expresses a positive or negative experience.

By applying Natural Language Processing (NLP) techniques and supervised learning algorithms, the system converts unstructured text into meaningful features using TF-IDF and then classifies sentiment using a Bernoulli Naive Bayes model and other baseline models.

In addition to sentiment (Positive/Negative), the application also highlights key aspects mentioned in the review, such as:

- Food Quality
- Service
- Hygiene
- Ambience

The final solution is deployed as a Flask web application with a professional interface, allowing restaurant managers to paste any review and instantly see the sentiment, confidence score, and important aspects.

Project Scenarios:

Scenario 1: Brand Reputation Monitoring

A restaurant chain receives hundreds of reviews every week on Yelp and Google. Manually reading each review is time-consuming. The Restaurant Review Classifier automatically processes all reviews, classifies them as positive or negative, and highlights issues related to food, service, hygiene, or ambience. Management can quickly monitor brand reputation and identify branches that are underperforming.

Scenario 2: Branch Performance & Quality Control

Head office wants to compare customer satisfaction across multiple branches. Using the classifier, they can aggregate sentiment scores for each location and see which branches receive more negative reviews about service delays, cold food, or cleanliness. This helps them prioritize training, staffing, or quality improvements.

Scenario 3: Hospital Emergency Department Triage

A customer care team receives emails and feedback forms daily. The Restaurant Review Classifier can automatically scan the text of each complaint, determine whether it is highly negative, and tag the main issue (e.g., “Hygiene” or “Service”). Critical complaints are forwarded immediately to the manager, helping the team respond faster to serious problems.

Technical Diagram:

Prerequisites:

To complete this project, the following software and packages are required:

Software:

- Python 3.x
- Anaconda Navigator / VS Code or any IDE
- Google Colab or Jupyter Notebook (for model training)
- Web browser (for running and testing the Flask application)

Python Packages:

Install these packages in a virtual environment:

- numpy – numerical computations
- pandas – data loading and manipulation
- scikit-learn – ML models, TF-IDF, train-test split, metrics
- nltk – text preprocessing (stopwords, stemming)
- matplotlib / seaborn – visualizations and EDA plots
- wordcloud – word cloud visualizations (optional)
- Flask – web framework for deployment

Prior Knowledge:

To effectively understand and implement this project, you should know:

1. **Machine Learning Concepts**
   - Supervised learning
   - Train-test split, overfitting, evaluation
2. **Text Mining / NLP Basics**
   - Tokenization
   - Stopword removal
   - Stemming / Lemmatization
   - Bag-of-Words and TF-IDF
3. **Classification Algorithms**
   - Naive Bayes (BernoulliNB)
   - Logistic Regression

- Support Vector Machines (SVM / LinearSVC)
- Random Forest
- K-Nearest Neighbors (KNN)

4. **Evaluation Metrics**
    - Accuracy, Precision, Recall, F1-Score
    - Confusion Matrix
5. **Flask Basics**
    - Routes (GET/POST)
    - Jinja2 templates
    - Serving static files (CSS, JS)

Project Flow:

- User enters / pastes a restaurant review through the web interface.
- System preprocesses the input text using the NLP pipeline (cleaning, stopwords removal, stemming).
- Integrated machine learning model converts the cleaned text into TF-IDF features and analyzes them.
- Prediction result (Positive / Negative sentiment) is displayed along with a confidence score (if available).
- System identifies key aspects mentioned in the review such as Food Quality, Service, Hygiene and Ambience, providing quick insights for restaurant management.

Project Activities:

1 Data Collection & Preparation

- Collect the restaurant review dataset (Restaurant_Reviews.tsv).
- Data preparation and text cleaning (lowercasing, removing punctuation, removing stopwords, stemming and building the corpus).

2 Exploratory Data Analysis

- Descriptive statistics (total reviews, class distribution, review length).
- Visual analysis (bar chart of positive vs negative, histogram of review length, word clouds).
- Understanding common words and patterns in positive and negative reviews.

3 Model Building

- Converting cleaned text into numerical features using TF-IDF vectorizer.
- Training the model with multiple algorithms:
    - Bernoulli Naive Bayes
    - Logistic Regression
    - Support Vector Machine (SVM)
    - Random Forest
    - K-Nearest Neighbors

4 Performance Testing & Model Selection

- Testing models with multiple evaluation metrics (accuracy, precision, recall, F1-score, confusion matrix).
- Comparing model accuracy across different algorithms.
- Selecting the best performing model (Bernoulli Naive Bayes with 77% accuracy) for deployment.

5 Model Deployment

- Save the best model (sentiment_model.pkl) and TF-IDF vectorizer (tfidf_vectorizer.pkl).
- Integrate the model with a Flask web framework and design a professional UI for analysing new reviews.

Project Structure:

Create the Project folder which contains files as shown below

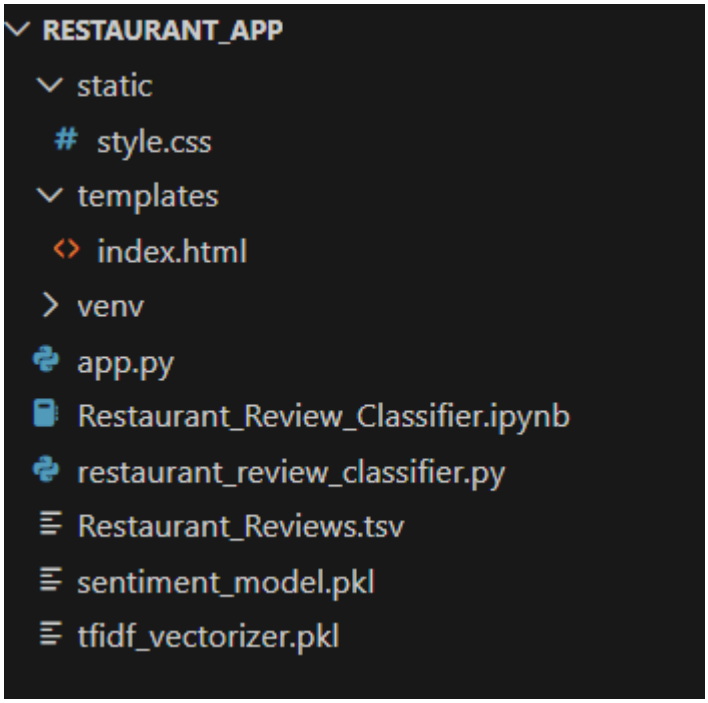


Figure 5.1: Project directory structure of Restaurant Review Classifier application showing dataset, trained model, web interface files, and Jupyter notebook.

Project Structure Explanation:

- **static/**
  Contains static UI resources used by the web application.
  - style.css – Defines the complete styling of the app including layout, theme colors (navy + grey), fonts, button styles, card design, and responsive UI.

- **templates/**
  Contains HTML templates used by the Flask backend for rendering web pages.

    - index.html – Main user interface where the user enters a review and views the output sentiment.

- **venv/**
  Python virtual environment that stores all installed libraries (Flask, nltk, sklearn, etc.).
  Ensures that the project runs with isolated dependencies, preventing version conflicts.

- **app.py**
  Primary Flask backend application.
  Handles:
    - Loading of model & vectorizer
    - Preprocessing input text
    - Predicting sentiment
    - Extracting aspects
    - Returning results to HTML template

This is the core operational engine of your (AI) app.

- **Restaurant_Review_Classifer.ipynb**
  Jupyter/Colab notebook containing model development steps:
    - Loading dataset
    - Cleaning text
    - TF-IDF vectorization
    - Model training
    - Model comparison
    - Saving final model & vectorizer

This notebook demonstrates the complete ML pipeline.

- **restaurant_review_classifier.py**
  Stand-alone Python script for loading the trained model and testing predictions without UI.
  Useful for debugging or running classification in a console.

- **Restaurant_Reviews.tsv**
  Dataset file containing restaurant reviews and sentiment labels.
    - Column 1: Review (text)
    - Column 2: Liked (0 or 1 sentiment)

- **sentiment_model.pkl**
  Stored (pickled) trained ML model — Bernoulli Naive Bayes.
  Used during prediction when the user submits a review.

- **tfidf_vectorizer.pkl**
  Stored TF-IDF vectorizer fitted to training corpus.
  Used to transform raw review input into numerical vectors before prediction.

Milestone 1: Data Collection & Preparation

This milestone focuses on collecting the restaurant review dataset and preparing the text data for NLP-based machine learning.

Activity 1.1: Collect the Dataset

For this project, we use a text dataset of restaurant reviews stored in Restaurant_Reviews.tsv. Each row contains:

- **Review** – text feedback from a customer
- **Liked** – sentiment label (1 = positive, 0 = negative)

The file is loaded using pandas.read_csv() with tab (\t) as the delimiter.

Data Set Link : Click here

Activity 1.2: Importing Required Libraries

We import the necessary Python libraries for text preprocessing, vectorization, machine learning model training, and evaluation. These include:

- pandas, numpy for data handling
- nltk for text preprocessing
- scikit-learn for vectorization & ML algorithms
- matplotlib / seaborn for visualizations

```
# Basic libraries
import numpy as np
import pandas as pd
import re
import pickle

# NLP libraries
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer

# ML libraries
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import BernoulliNB
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

Figure 1.2: Importing required Python libraries for model training and analysis.

Activity 1.3: Reading and Inspecting the Dataset

The dataset is read using:

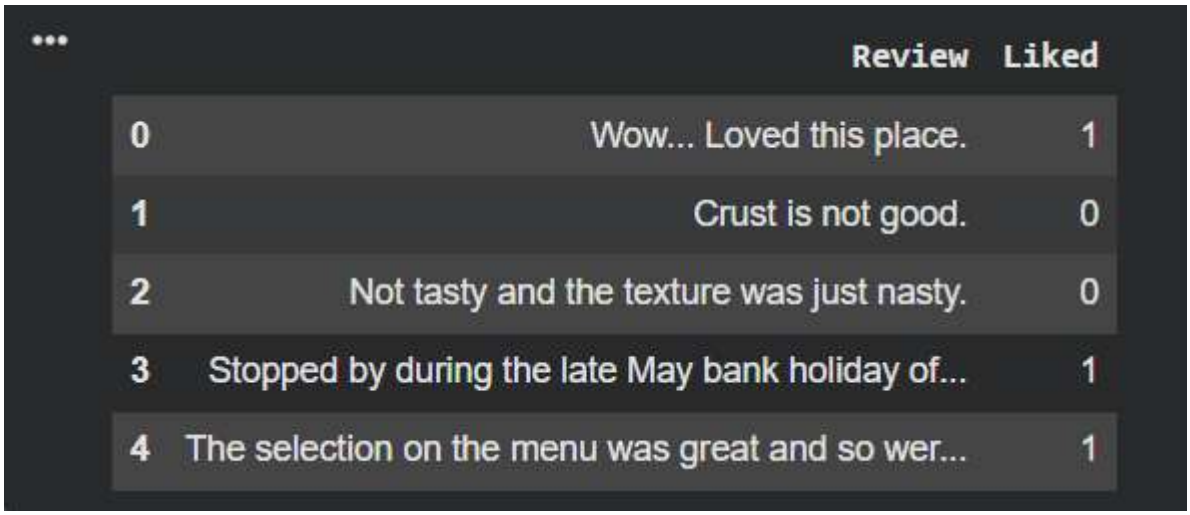
data = pd.read_csv('Restaurant_Reviews.tsv', sep='\t')

We then inspect its structure using:

- data.head()
- data.info()
- data['Liked'].value_counts()

This helps us understand:

- number of reviews
- sentiment distribution
- presence of imbalanced data



	Review	Liked
0	Wow... Loved this place.	1
1	Crust is not good.	0
2	Not tasty and the texture was just nasty.	0
3	Stopped by during the late May bank holiday of...	1
4	The selection on the menu was great and so wer...	1

Figure 1.3: Initial dataset inspection showing sample reviews and sentiment label distribution.

Activity 1.4: Text Cleaning & Corpus Creation

We apply the following preprocessing steps:

- Cleaning non-alphabetic characters
  Converting text to lowercase
  Removing stopwords
  Applying stemming (PorterStemmer)
  Constructing the cleaned corpus of reviews

This ensures that the model learns from meaningful words without noise.

```python
ps = PorterStemmer()
stop_words = set(stopwords.words('english'))

corpus = []

for review in data['Review']:
    # 1. Keep only letters
    review = re.sub('[^a-zA-Z]', ' ', review)
    # 2. Lowercase
    review = review.lower()
    # 3. Split into words
    words = review.split()
    # 4. Remove stopwords & apply stemming
    words = [ps.stem(word) for word in words if word not in stop_words]
    # 5. Join back into one string
    cleaned_review = ' '.join(words)
    corpus.append(cleaned_review)

# See few cleaned reviews
corpus[:5]
```

Figure 1.4: Text preprocessing applied to transform raw text into cleaned and stemmed tokens.

Activity 1.5: Final Prepared Corpus

After cleaning, each review is stored as a processed string in the corpus, ready for feature extraction using TF-IDF.

Example:

food amaz servic slow

staff friendli food great

tabl dirti servic terribl

This results in normalized and comparable textual data.

```
['wow love place',
 'crust good',
 'tasti textur nasti',
 'stop late may bank holiday rick steve recommend love',
 'select menu great price']
```

Figure 1.5: Example of finalized cleaned text data used for ML training.

Milestone 2: Exploratory Data Analysis (EDA)

This milestone focuses on exploring the sentiment distribution and key patterns in the dataset through visualizations and statistical analysis.

Activity 2.1: Sentiment Statistics & Distribution

We first analyze sentiment proportions in the dataset using:

data['Liked'].value_counts()

This shows the number of positive (1) vs negative (0) reviews.

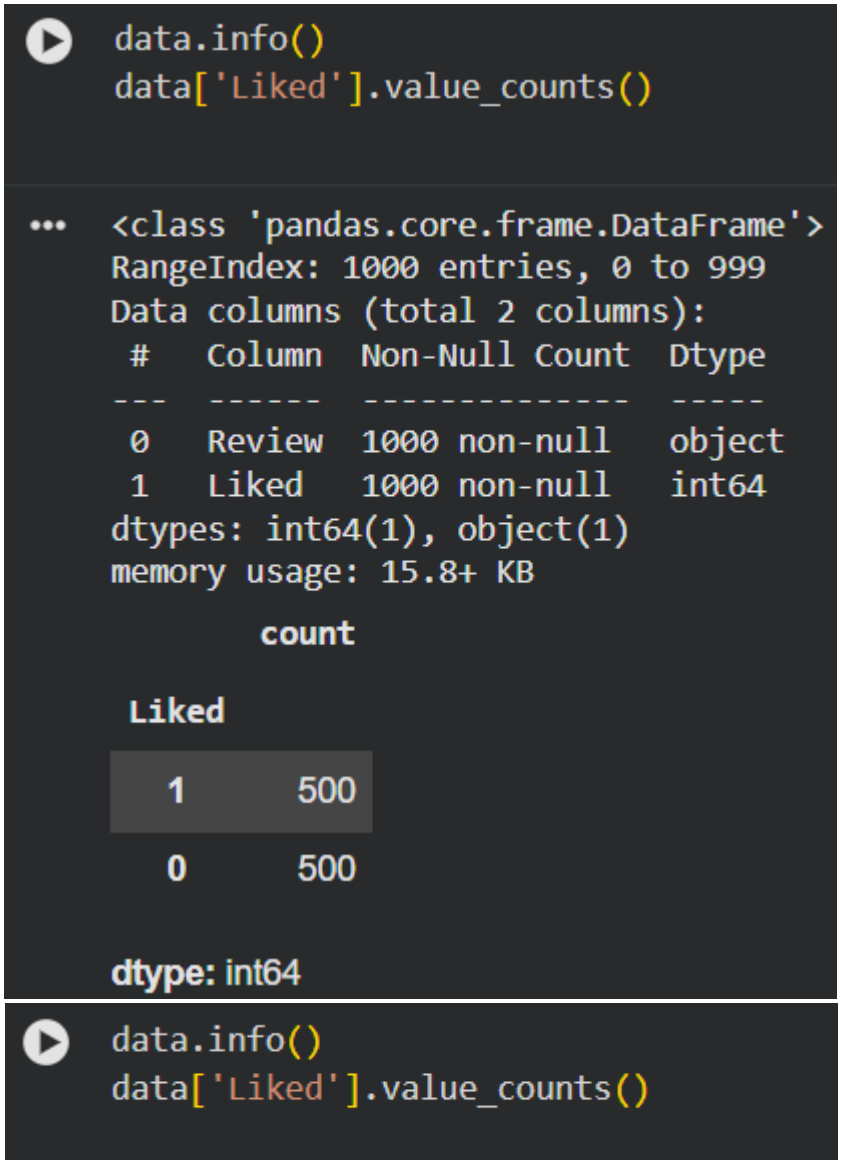


Figure 2.1: Statistical distribution of sentiment labels in the dataset (Positive vs Negative).

Activity 2.2: Visual Analysis

In this step, we visualize the number of positive and negative reviews to observe sentiment balance in the dataset. By plotting a bar chart of the sentiment labels, we can visually confirm that the data is balanced with equal representation of each class.

```
data['Liked'].value_counts().plot(kind='bar')
```

```
<Axes: xlabel='Liked'>
```

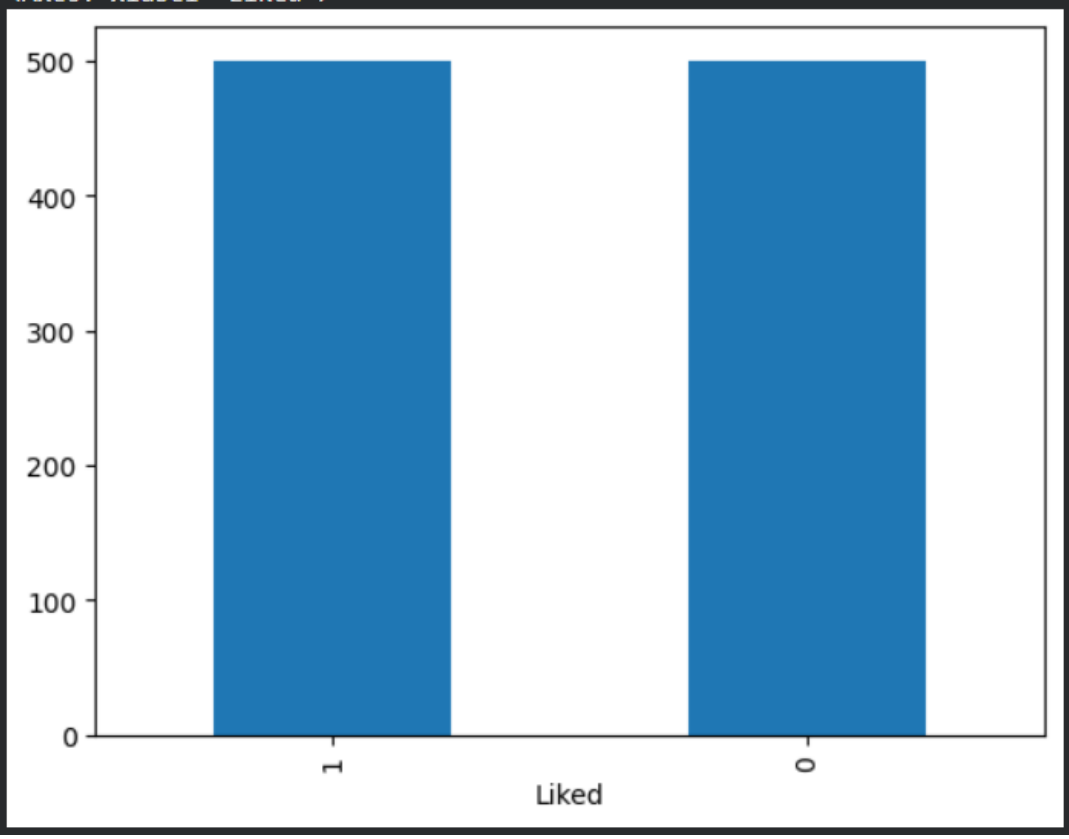


Figure 2.2: Bar chart showing distribution of positive (1) vs negative (0) restaurant reviews.

Milestone 3: Model Building

The objective of this milestone is to transform textual data into machine-understandable features and train multiple machine learning models.

Activity 3.1: Feature Extraction using TF-IDF

We convert the cleaned text corpus into numerical feature vectors using TF-IDF (Term Frequency-Inverse Document Frequency).
This helps capture the importance of words relative to the dataset.

```
# Convert text to numbers using TF-IDF
tfidf = TfidfVectorizer(max_features=1500)
X = tfidf.fit_transform(corpus).toarray()

# Target labels (0 = not liked, 1 = liked)
y = data['Liked'].values

X.shape, y.shape
```

```
((1000, 1500), (1000,))
```

Figure 3.1: Converting cleaned text corpus into TF-IDF numerical features.

Activity 3.2: Splitting the Dataset

We divide the dataset into training and testing subsets.

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,   # 20% test
    random_state=0
)

X_train.shape, X_test.shape
```

```
((800, 1500), (200, 1500))
```

Figure 3.2: Splitting TF-IDF features into training and testing datasets.

Activity 3.3: Training Multiple Models

Train the following models on the training set:

- Bernoulli Naive Bayes (BernoulliNB)
- Logistic Regression
- Support Vector Machine (LinearSVC)
- Random Forest Classifier
- K-Nearest Neighbors (KNN)

Each model is evaluated on the test set using accuracy and basic classification metrics.

```
BernoulliNB Accuracy: 0.7700
Logistic Regression Accuracy: 0.7500
SVM Accuracy: 0.7300
Random Forest Accuracy: 0.7250
KNN Accuracy: 0.6850
```

Figure 3.3: Accuracy comparison of multiple machine learning classifiers.

4. Performance Testing & Model Comparison

Activity 4.1: Model Comparison

Accuracy scores observed (example from your run):

Model	**Accuracy**
Bernoulli Naive Bayes	**0.77**
Logistic Regression	0.75
SVM (LinearSVC)	0.73
Random Forest	0.73
K-Nearest Neighbors	0.685

A bar chart is plotted to visually compare the performance of these models.

Conclusion:
Bernoulli Naive Bayes achieves the highest accuracy (**77%**) and is therefore selected as the final model for deployment.

Activity 4.2: Evaluation Metrics

Beyond accuracy, the following metrics are considered:

- **Precision** – of all predicted positive reviews, how many are truly positive
- **Recall** – of all actual positive reviews, how many were correctly identified
- **F1-Score** – harmonic mean of precision and recall
- **Confusion Matrix** – counts of True Positives, True Negatives, False Positives, and False Negatives

The classifier shows balanced performance for both classes, indicating that the model is reasonably reliable for real-world sentiment analysis.

---

5. Model Deployment

Activity 5.1: Save the Best Model

The best performing model and the TF-IDF vectorizer are saved using pickle:

- sentiment_model.pkl – trained BernoulliNB model
- tfidf_vectorizer.pkl – fitted TF-IDF vectorizer

These files are later loaded in the Flask application to make predictions on new reviews.

Activity 5.2: Flask Web Application Development

Project Structure

The project folder is organized as:

- **static/**
    - style.css – application styling (navy + light grey professional theme)
- **templates/**
    - index.html – main page for review input and result display
- **app.py**
    - Flask backend application
- **sentiment_model.pkl**
    - Trained sentiment classifier
- **tfidf_vectorizer.pkl**
    - Trained TF-IDF vectorizer
- **restaurant_reviews.ipynb**

    - Jupyter/Colab notebook containing data analysis and model training code
- **dataset/**
    - Restaurant_Reviews.tsv – source dataset of labeled reviews

Backend Implementation (app.py)

The Flask backend performs the core ML operations:

app

- **Model Loading:**
    - Loads sentiment_model.pkl and tfidf_vectorizer.pkl at startup.
- **Text Preprocessing:**
    - Applies the same cleaning pipeline (regex, lowercase, stopwords, stemming) to user input as used during training.
- **Feature Transformation:**
    - Uses the loaded TF-IDF vectorizer to convert the cleaned review into feature vectors.
- **Prediction & Confidence:**
    - Predicts sentiment (0 = Negative, 1 = Positive).
    - Attempts to compute prediction probability using predict_proba for a confidence score.
- **Aspect Extraction:**
    - Uses a simple keyword-based function to detect aspects such as Service, Food Quality, Hygiene, and Ambience from the raw review text.
- **Rendering Output:**
    - Passes sentiment label, confidence score, aspects, and original review to index.html for display.

Frontend Implementation (index.html + style.css)

The web interface is a professional dashboard-style page:

- **Header:**
    - Branding as “FoodSense AI – Restaurant Review Classifier”.
- **Input Section:**
    - Large textarea for pasting a customer review.
    - “Analyse Review” button submits the review to the backend.
- **Result Section:**

    - Color-accented card showing:
        - "Positive Experience" or "Negative Experience"
        - Confidence score (if available)
        - Aspect chips (Service, Food Quality, Hygiene, Ambience, or General Experience)
        - Original review text for reference
- **Styling:**
    - Navy (#2E4053) and light grey (#BFC9CA) theme with rounded cards and subtle shadows.
    - Responsive layout that works on desktop and mobile screens.

Application Screenshots and Workflow

You should include screenshots in your report similar to the Anemia Detection document:

1. **Home Page Interface**
    - Shows the brand header "FoodSense AI", the hero text, and the review input card.
2. **Prediction Result View**
    - After submitting a review, show the result card with:
        - Positive/Negative label
        - Confidence score
        - Aspect tags
        - Original review text

These screenshots demonstrate the complete flow from user input to AI-generated insight.

Future Implementations

Future enhancements for the Restaurant Review Classifier may include:

- **Multi-Class Sentiment:**
    - Extending from binary (Positive/Negative) to multi-level sentiment (Very Positive, Neutral, Very Negative).
- **Advanced Aspect-Based Sentiment Analysis:**
    - Using more sophisticated NLP models (e.g., BERT) to give separate sentiment scores for Food, Service, Hygiene, and Ambience.
- **Live Integration with Review Platforms:**
    - Connecting to Yelp/Google APIs to automatically fetch new reviews and update dashboards in real time.
- **Branch-Level Analytics Dashboard:**
    - Building an admin dashboard to show sentiment trends per branch, top issues over time, and overall brand health.

- **Multilingual Support:**
    - Supporting reviews written in local languages like Telugu, Hindi, or Tamil by using multilingual embeddings or translation.

Conclusion

The Restaurant Review Classifier project successfully demonstrates how machine learning and NLP can be applied to the food industry for intelligent feedback analysis. Using TF-IDF features and a Bernoulli Naive Bayes classifier, the system achieves an accuracy of approximately **77%** on unseen test data, outperforming baseline models such as Logistic Regression, SVM, Random Forest, and KNN.

The final product is a full-stack Flask web application that:

- Accepts free-form customer reviews
- Automatically predicts sentiment (Positive/Negative)
- Highlights key aspects such as Food Quality, Service, Hygiene, and Ambience
- Provides a clean and professional interface for non-technical users

This solution helps restaurant managers and decision-makers quickly understand customer opinions, identify problem areas, and improve overall service quality based on real customer feedback.