Sentiment Analysis Of Restaurant Reviews

by

Harshini Akshaya A S - 220701088

Guide
Dr. V.Auxilia Osvin Nancy.,M.Tech.,Ph.D.,
Assistant Professor
Department of Computer Science and Engineering,
Rajalakshmi Engineering College,
Chennai - 602 105.

Introduction

This project leverages machine learning to classify restaurant reviews as positive or negative, helping businesses understand customer sentiment. Using algorithms like **Naive Bayes, Logistic Regression** and **Random Forest** the model analyzes textual data to identify patterns in feedback related to service, food, ambiance, and pricing. Developed in Python, the system transforms unstructured reviews into actionable insights, aiding real-time decision-making and enhancing customer satisfaction.

Literature Survey

- **Dr. Ratna Patil et al. -** Applied ML algorithms (KNN, LR, SVM, NB) on restaurant reviews; SVM achieved 78% accuracy.
- **Ameen Aqlan et al. -** Reviewed sentiment classification techniques; highlighted Big Data (Hadoop) for scalable analysis.
- **Monali Bordoloi & Saroj Biswas -** Surveyed end-to-end SA pipeline; identified limitations and future interdisciplinary research directions.
- **Manika Lamba & Madhusudhan Margam -** Explored emotion extraction (joy, anger, sadness); showcased SA applications in libraries.

Objectives

- To analyze customer reviews from restaurants using sentiment analysis techniques.
- To build a machine learning model that classifies sentiments as positive or negative.
- To compare performance of algorithms like Naive Bayes and Logistic Regression.
- To preprocess and augment data for better model accuracy and generalization.
- To provide actionable insights for restaurants to improve customer experience.
- To develop a scalable solution for real-time sentiment interpretation.

Methodology - Data Preparation & Feature Engineering

Data Collection

- 1,000 labeled reviews from Kaggle (positive/negative).
- Expanded to 10,000 reviews via synonym replacement using WordNet (NLTK).
- Output saved as Augmented_Reviews.tsv.

Preprocessing Steps

- Lowercasing for uniformity.
- Noise removal (punctuation, special characters) using regex.
- Tokenization and stopword removal (NLTK).
- Lemmatization to reduce words to their root forms.

Feature Engineering

- Bag of Words (BoW): Word frequency matrix.
- TF-IDF: Highlights important, unique words.
- Custom Features:
- Review length
- Word count

Methodology - Model Training

Algorithms Used:

1. ***Naive Bayes***
   - Simple and effective for text classification
   - Assumes feature independence (works well with BoW)
2. ***Logistic Regression***
   - Probabilistic linear model for binary classification
   - Captures nuanced relationships between word frequencies and sentiment
3. ***Random Forest Classifier***
   - Ensemble of decision trees
   - Handles feature interactions and reduces overfitting

Training/Test Split:

- 80% data used for training
- 20% data used for testing

Methodology - Model Selection & Evaluation

- ***Naive Bayes***
  - Accuracy: 88.65%
  - Confusion Matrix: [[753, 221], [6, 1020]]
  - Fast and simple
- ***Logistic Regression***
  - Accuracy: 96.7%
  - Confusion Matrix: [[946, 28], [38, 988]]
  - Balanced precision and recall; better generalization
- ***Random Forest (Best Model)***
  - Accuracy: 98.25%
  - Confusion Matrix: [[959, 15], [20, 1006]]
  - Captured complex patterns; lowest misclassification
  - Chosen for final deployment

Evaluation Metrics Used:

- Accuracy: Overall correctness
- Confusion Matrix: Detailed error analysis
- Precision, Recall, F1-Score: Performance on both classes
- Cross-validation: Ensured model consistency and robustness

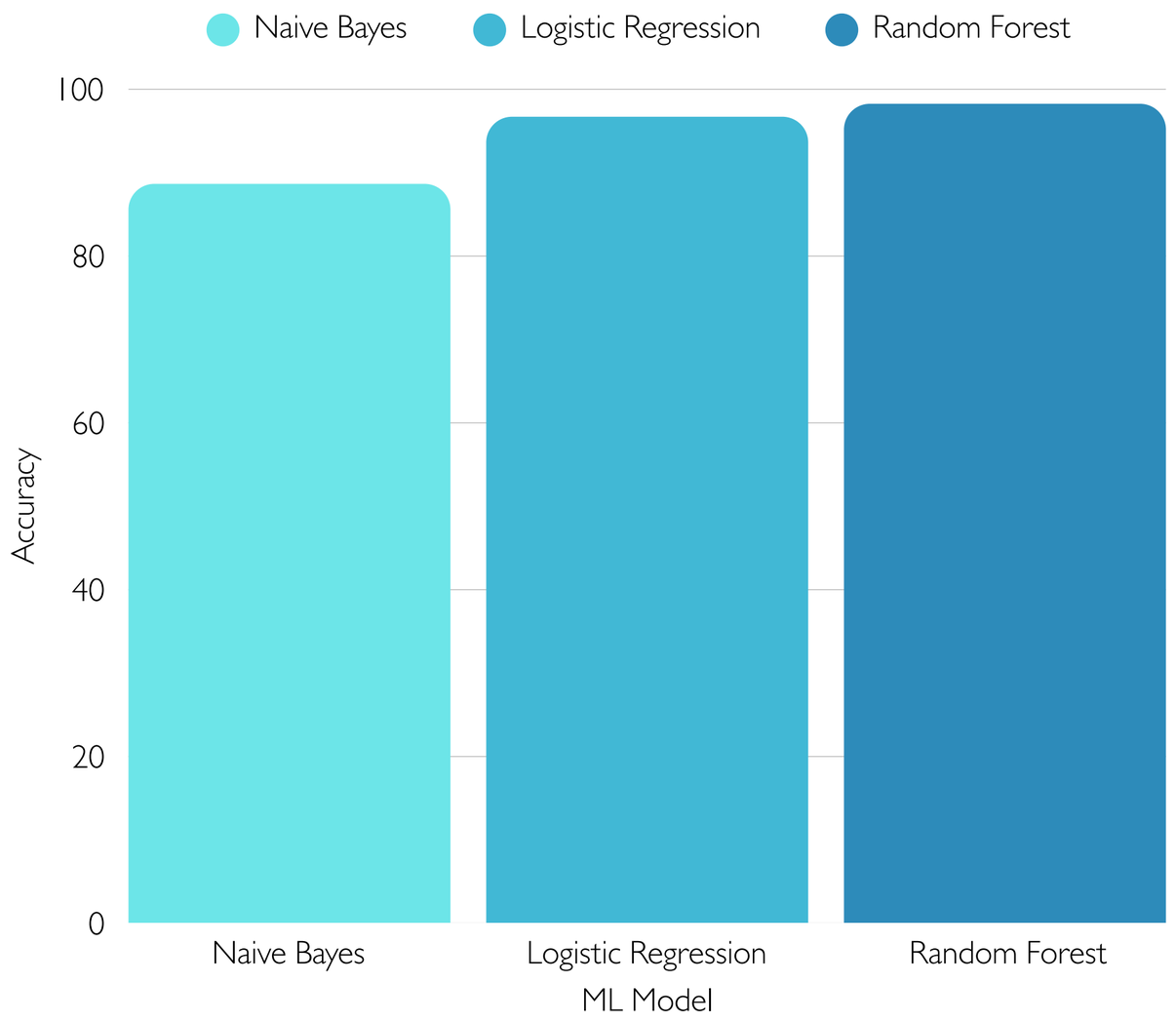
Methodology - Deployment

- The best-performing model (**Random Forest**) was selected for deployment.
- The model and vectorizer were serialized using Python's pickle module:
- Review_model.pkl – trained Random Forest model
- CountVectorizer.pkl – fitted vectorizer
- A Flask web application was developed to serve predictions.

Workflow:

- User inputs a review on the web interface.
- The review is preprocessed using the same pipeline as training.
- Transformed using the saved CountVectorizer.
- The model predicts sentiment in real-time.
- Enables real-time, on-demand sentiment analysis of restaurant reviews.

Results - Comparative Analysis



Conclusion

- Developed a sentiment analysis system to classify restaurant reviews as positive or negative.
- Implemented and compared three models: Naive Bayes, Logistic Regression, and Random Forest.
- Random Forest achieved the best performance with 98.25% accuracy, followed by Logistic Regression (96.7%) and Naive Bayes (88.65%).
- Deployed the best model using a Flask web app for real-time sentiment prediction.
- Demonstrated practical applications for platforms like food delivery apps and review aggregators.
- Successfully met the project's objective of building an accurate and user-friendly sentiment classifier.

Conclusion

- Developed a sentiment analysis system to classify restaurant reviews as positive or negative.
- Implemented and compared three models: Naive Bayes, Logistic Regression, and Random Forest.
- Random Forest achieved the best performance with 98.25% accuracy, followed by Logistic Regression (96.7%) and Naive Bayes (88.65%).
- Deployed the best model using a Flask web app for real-time sentiment prediction.
- Demonstrated practical applications for platforms like food delivery apps and review aggregators.
- Successfully met the project's objective of building an accurate and user-friendly sentiment classifier.

Future Enhancements

- Use Deep Learning Models - Improve context understanding with models like LSTM or BERT.
- Better Word Representation - Replace traditional methods (BoW/TF-IDF) with word embeddings like Word2Vec.
- Support for More Sentiment Labels - Move from binary to multi-class (e.g., positive, neutral, negative).
- Expand to More Languages - Add support for reviews in different languages.
- Improve Text Cleaning - Handle slang, emojis, and spelling mistakes better.
- Real-Time Updates - Retrain the model with user feedback to improve over time.
- Make the Model Explainable - Use tools to explain why the model gave a certain sentiment.

THANK YOU