

Sentiment Analysis on Restaurant Reviews

Dissertation submitted in fulfilment of the requirements for the Degree of

BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING

By
SAKTHI BALA
12104759

Supervisor
VED PRAKASH CHAUBEY



School of Computer Science and Engineering

Lovely Professional University
Phagwara, Punjab (India)

@ Copyright LOVELY PROFESSIONAL UNIVERSITY, Punjab (INDIA)

ALL RIGHTS RESERVED

DECLARATION STATEMENT

I hereby declare that the research work reported in the dissertation/dissertation proposal entitled “SENTIMENT ANALYSIS ON RESTAURANT REVIEWS” in partial fulfilment of the requirement for the award of Degree for Bachelor of Technology in Computer Science and Engineering at Lovely Professional University, Phagwara, Punjab is an authentic work carried out under supervision of my research supervisor Mr. VED PRAKASH CHAUBEY. I have not submitted this work elsewhere for any degree or diploma.

I understand that the work presented herewith is in direct compliance with Lovely Professional University’s Policy on plagiarism, intellectual property rights, and highest standards of moral and ethical conduct. Therefore, to the best of my knowledge, the content of this dissertation represents authentic and honest research effort conducted, in its entirety, by me. I am fully responsible for the contents of my dissertation work.

Signature of Candidate

SAKTHI BALA

12104759

SUPERVISOR'S CERTIFICATE

This is to certify that the work reported in the B. Tech Dissertation/dissertation proposal entitled “**SENTIMENT ANALYSIS ON RESTAURANT REVIEWS**”, submitted by **SAKTHI BALA** at **Lovely Professional University, Phagwara, India** is a bonafide record of his original work carried out under my supervision. This work has not been submitted elsewhere for any other degree.

Signature of Supervisor

VED PRAKASH CHAUBEY

Neutral Examiners:

External Examiner

Signature: _____

Name: _____

Affiliation: _____

Date: _____

Internal Examiner

Signature: _____

Name: _____

Date: _____

TABLE OF CONTENTS

S.NO	TITLE	PAGE. NO
1.	CHAPTER-1 (INTRODUCTION)	5
2.	CHAPTER- 2 DATA ACQUISITION AND EXPLORATION	9
3.	CHAPTER-3 DATA PREPROCESSING	12
4.	CHAPTER-4 MODEL SELECTION AND TRAINING	16
5.	CHAPTER-5 MODEL EVALUATION	19
6.	CHAPTER -6 HYPERPARAMTER TUNING	21
7.	CHAPTER-7 PREDICTION AND APPLICATION	24
8.	CHAPTER-8 CONCLUSION	27
9.	CHAPTER-9 REFERENCE	29
10.	CHAPTER-10 APPENDICES	

Chapter 1: Introduction

1. Introduction

In today's digitally interconnected world, where social media platforms and online review websites abound, understanding and leveraging customer sentiment has become essential for businesses to thrive. Sentiment analysis, a branch of natural language processing (NLP), offers a systematic approach to extracting, quantifying, and interpreting sentiments expressed in textual data. By analyzing customer feedback, businesses can gain valuable insights into consumer preferences, satisfaction levels, and areas for improvement.

1.1 Background

The emergence of sentiment analysis has revolutionized the way businesses perceive and respond to customer feedback. Traditionally, businesses relied on manual methods to gauge customer sentiment, such as surveys and focus groups. However, these methods were often time-consuming, subjective, and limited in scope. With the advent of sentiment analysis, businesses can now process vast amounts of textual data in real-time, allowing them to uncover actionable insights quickly and efficiently.

Within the restaurant industry, customer feedback plays a pivotal role in shaping business strategies and driving operational decisions. Whether through online reviews on platforms like Yelp and TripAdvisor or social media mentions on platforms like Twitter and Instagram, customers have a myriad of channels through which they can express their opinions about dining experiences. Understanding the sentiments conveyed in these reviews is crucial for restaurants to identify strengths, address weaknesses, and ultimately enhance the overall dining experience for patrons.

1.2 Introduction to the Restaurant Industry

The restaurant industry is renowned for its emphasis on customer satisfaction and culinary innovation. With a diverse array of dining options available to consumers, restaurants must continually strive to differentiate themselves and deliver exceptional experiences to diners. In this competitive landscape, customer feedback serves as a valuable source of information for restaurants to assess performance, refine offerings, and build lasting relationships with patrons.

Customer feedback in the restaurant industry encompasses various aspects of the dining experience, including food quality, service efficiency, ambiance, and overall value for money. Positive reviews can serve as powerful endorsements, attracting new customers and enhancing brand reputation, while negative reviews can highlight areas for improvement and prompt corrective action. By systematically analyzing and responding to customer feedback, restaurants can cultivate a loyal customer base, drive repeat business, and maintain a competitive edge in the market.

1.3 Objectives of the Project

The primary objectives of this project encompass analyzing sentiment in restaurant reviews, identifying key insights, and building predictive models. These objectives are intertwined with the overarching goal of enhancing customer satisfaction and driving business performance in the restaurant industry.

Analyzing Sentiment in Restaurant Reviews: One of the central objectives of this project is to employ sentiment analysis techniques to evaluate and interpret the sentiments expressed in restaurant reviews. By systematically analyzing the textual content of reviews, we aim to categorize them into positive, neutral, or negative sentiments. This analysis provides valuable insights into customer perceptions, preferences, and experiences with respect to various aspects of the dining experience, such as food quality, service, ambiance, and value for money.

Identifying Key Insights: Another key objective is to extract actionable insights from the analyzed restaurant reviews. By identifying recurring themes, trends, and sentiments expressed by customers, we aim to uncover valuable insights that can inform business decision-making. These insights may include identifying areas of strength that restaurants can capitalize on, pinpointing areas for improvement that require attention, and understanding customer preferences and expectations.

Building Predictive Models: In addition to analyzing sentiment and extracting insights from restaurant reviews, this project aims to develop predictive models capable of automatically categorizing reviews based on sentiment. By leveraging machine learning algorithms and labeled datasets of restaurant reviews, we seek to build classifiers that can accurately predict whether a review expresses positive or negative sentiment. These predictive models streamline the process of sentiment analysis, enabling businesses to efficiently assess customer feedback at scale and take proactive measures to address issues and capitalize on strengths.

Contribution to Improving Customer Satisfaction and Business Performance: Achieving these objectives directly contributes to improving customer satisfaction and driving business performance in the restaurant industry. By systematically analyzing sentiment in restaurant reviews, businesses gain a deeper understanding of customer perceptions and preferences, allowing them to tailor their offerings and services to meet customer expectations more effectively. Moreover, by identifying key insights and leveraging predictive models, restaurants can proactively address customer concerns, capitalize on strengths, and make data-driven decisions that enhance the overall dining experience and drive customer loyalty. Ultimately, by harnessing the power of sentiment analysis and predictive modeling, businesses can achieve higher levels of customer satisfaction, increase customer retention, and drive sustainable business growth in an increasingly competitive market landscape.

Chapter 2: Data Acquisition and Exploration

2.1 Dataset Description

2.1.1 Source of Data

The Restaurant Reviews dataset used in this project was obtained from [source]. This dataset comprises a collection of customer reviews for various restaurants, sourced from online review platforms such as Yelp or TripAdvisor.

2.1.2 Format and Size

The dataset is structured as a tabular dataset, typically in CSV (Comma-Separated Values) or TSV (Tab-Separated Values) format. It contains two main columns:

Review Text: This column contains the textual content of each restaurant review provided by customers.

Sentiment Label: This column indicates the sentiment associated with each review, typically binary (e.g., positive or negative), denoting whether the review is positive or negative.

The dataset size may vary depending on the number of reviews collected. A larger dataset may contain thousands or even millions of reviews, while a smaller dataset may contain a few hundred reviews.

2.1.3 Features

The primary features present in the dataset include:

Review Text: The textual content of each restaurant review, which serves as the primary input for sentiment analysis.

Sentiment Label: The sentiment label associated with each review, indicating whether it is positive or negative.

2.1.4 Biases and Limitations

It's essential to acknowledge potential biases or limitations in the dataset that may impact the analysis:

Selection Bias: The dataset may contain reviews primarily from certain demographics or geographic regions, leading to potential biases in the sentiments expressed.

Language Bias: The dataset may be limited to reviews written in a specific language, potentially excluding reviews from non-native speakers or multilingual customers.

Platform Bias: Reviews sourced from specific online review platforms may exhibit platform-specific biases or characteristics.

2.2 Data Exploration

2.2.1 Summary Statistics

Summary statistics were computed to gain insights into the dataset:

Review Lengths: Mean, median, and standard deviation of review lengths were calculated to understand the distribution of review lengths.

Sentiment Labels: Summary statistics, such as the count of positive and negative reviews, were calculated to assess the balance between positive and negative sentiments.

2.2.2 Class Distribution Visualization

A visualization of the class distribution, such as a bar plot or pie chart, was created to visually represent the balance between positive and negative reviews. This visualization helps understand the distribution of sentiments in the dataset.

2.2.3 Trends and Patterns

Exploratory data analysis techniques were employed to identify any trends or patterns in the data. This includes analyzing common words or phrases in positive and negative reviews, identifying frequent topics or themes, and detecting any temporal trends or seasonal patterns in the reviews.

In this chapter, we explored the Restaurant Reviews dataset, describing its source, format, and features, and discussed potential biases or limitations. We also conducted data exploration, including summary statistics, class distribution visualization, and identification of trends or patterns in the data. These insights lay the groundwork for subsequent analysis and model development in the following chapters.

Chapter 3: Data Preprocessing

3.1 Text Cleaning

3.1.1 Description of Text Cleaning Techniques

Text cleaning is a crucial preprocessing step that involves transforming raw text data into a format suitable for analysis. Key text cleaning techniques include:

Removal of Non-Alphabetic Characters: Non-alphabetic characters such as punctuation marks, digits, and special symbols are removed from the text. This helps to eliminate noise and irrelevant information from the data.

Conversion to Lowercase: All text is converted to lowercase to ensure consistency in text representation. This prevents the model from treating words with different cases (e.g., "Word" and "word") as distinct entities.

Stopwords Removal: Stopwords are common words that do not carry significant meaning in text analysis, such as "the", "is", "and". These words are removed from the text to reduce noise and focus on meaningful content.

Stemming: Stemming is the process of reducing words to their root or base form. The Porter stemming algorithm, a widely used stemming technique, is applied to transform words into their root form. For example, "running", "ran", and "runs" are stemmed to "run".

3.1.2 Importance of Preprocessing Steps

Each preprocessing step plays a crucial role in preparing the text data for analysis:

Removal of Non-Alphabetic Characters: Eliminates irrelevant characters that do not contribute to the semantic meaning of the text, reducing noise and improving the quality of the data.

Conversion to Lowercase: Ensures uniformity in text representation, preventing the model from treating words with different cases as distinct entities. This simplifies the analysis and improves the model's ability to generalize.

Stopwords Removal: Reduces the dimensionality of the text data by removing common words that do not carry significant meaning. This focuses the analysis on informative content and improves computational efficiency.

Stemming: Reduces words to their base form, which helps in standardizing variations of words and capturing their core meaning. This simplifies the vocabulary and improves the model's ability to generalize across different forms of words.

3.2 Feature Engineering

3.2.1 Overview of Feature Extraction Techniques

Feature engineering involves transforming raw data into a format suitable for machine learning models. In text analysis, common feature extraction techniques include:

Bag of Words (BoW) Representation: BoW represents text data as a collection of word vectors, where each vector represents the frequency of a word in the text corpus. This creates a numerical representation of text data that can be used as input for machine learning models.

Vectorization using CountVectorizer: CountVectorizer is a feature extraction technique that converts a collection of text documents into a matrix of token counts. Each row in the matrix represents a document, and each column represents a unique word in the corpus. The value in each cell indicates the frequency of the corresponding word in the document.

3.2.2 Rationale and Role of Feature Engineering

Feature engineering is essential in building machine learning models for sentiment analysis for the following reasons:

Normalization: Feature engineering techniques such as BoW representation and vectorization normalize the text data, transforming it into a structured format suitable for analysis.

Dimensionality Reduction: Feature engineering helps in reducing the dimensionality of the text data by representing it in a more compact and informative form. This simplifies the model and improves computational efficiency.

Model Interpretability: Feature engineering techniques provide a clear and interpretable representation of the text data, allowing the model to capture meaningful patterns and relationships between words and sentiments.

In summary, data preprocessing techniques such as text cleaning and feature engineering play a crucial role in preparing text data for sentiment analysis. These techniques help in standardizing, simplifying, and transforming the text data into a format suitable for machine learning models, enabling effective sentiment analysis and interpretation of results.

Chapter 4: Model Selection and Training

4.1 Overview of Classifiers

4.1.1 Introduction to Various Machine Learning Classifiers

In sentiment analysis, various machine learning classifiers can be employed to predict the sentiment of text data. Some commonly used classifiers include:

Naive Bayes: Naive Bayes classifiers are probabilistic models based on Bayes' theorem. They are simple yet effective for text classification tasks and work well with high-dimensional data such as text features.

Support Vector Machine (SVM): SVM classifiers aim to find the hyperplane that best separates different classes in the feature space. They are versatile and can handle linear as well as non-linear classification tasks using different kernel functions.

Decision Tree: Decision tree classifiers make decisions based on a series of if-else questions about the features of the data. They are interpretable and can capture complex relationships between features and target variables.

Random Forest: Random forest classifiers are an ensemble learning method that constructs multiple decision trees during training and outputs the mode of the classes (classification) or the mean prediction (regression) of the individual trees.

AdaBoost: AdaBoost classifiers combine multiple weak classifiers to create a strong classifier. They sequentially train classifiers on subsets of the data, with each subsequent model focusing on the instances misclassified by the previous models.

Gradient Boosting: Gradient boosting classifiers build decision trees sequentially, with each tree learning from the errors of its predecessors. They are powerful and often provide state-of-the-art results in many machine learning tasks.

4.1.2 Strengths and Weaknesses

Each classifier has its own strengths and weaknesses:

Naive Bayes: Simple and fast, works well with high-dimensional data, but assumes independence between features.

SVM: Effective in high-dimensional spaces, versatile with different kernel functions, but can be sensitive to the choice of parameters and computational resources.

Decision Tree: Easy to interpret, can capture complex relationships, but prone to overfitting and may not generalize well to unseen data.

Random Forest: Robust and accurate, handles high-dimensional data well, but may be slow to train and less interpretable than decision trees.

AdaBoost: Can improve performance with weak learners, less prone to overfitting, but sensitive to noisy data and outliers.

Gradient Boosting: Provides high accuracy, handles complex interactions well, but may require careful tuning of hyperparameters and can be computationally expensive.

4.2 Model Training

4.2.1 Description of Model Training Process

The model training process involves splitting the dataset into training and testing sets to evaluate the performance of the classifiers. The training set is used to train

the classifiers, while the testing set is used to assess their performance on unseen data.

4.2.2 Training Classifiers and Hyperparameters

Each classifier is trained using the training data, and their hyperparameters are tuned to optimize performance. Hyperparameters control the behavior of the classifiers and can significantly impact their effectiveness. Techniques such as grid search or random search may be used to find the optimal hyperparameters for each classifier.

In summary, model selection and training involve exploring various machine learning classifiers, understanding their strengths and weaknesses, and training them using the dataset. The performance of the classifiers is evaluated based on their ability to accurately predict the sentiment of text data.

Chapter 5: Model Evaluation

5.1 Evaluation Metrics

5.1.1 Explanation of Evaluation Metrics

Evaluation metrics are essential for assessing the performance of classifiers in sentiment analysis tasks. The following metrics are commonly used:

Accuracy: Measures the proportion of correctly classified instances out of the total instances. It provides an overall measure of the model's correctness.

Precision: Also known as positive predictive value, precision measures the proportion of true positive predictions out of all positive predictions. It indicates the model's ability to avoid false positives.

Recall: Also known as sensitivity or true positive rate, recall measures the proportion of true positive predictions out of all actual positive instances. It indicates the model's ability to capture all positive instances.

F1-score: The harmonic mean of precision and recall, F1-score provides a balance between precision and recall. It is particularly useful when the class distribution is imbalanced.

5.1.2 Discussion on Interpretation of Evaluation Metrics

Interpretation of evaluation metrics depends on the specific requirements of the sentiment analysis task and the trade-offs between precision and recall. For example:

High accuracy indicates overall model correctness but may not reveal performance imbalances between classes.

High precision suggests that the model makes few false positive predictions, which is crucial for tasks where false positives are costly.

High recall indicates that the model captures most positive instances, which is important for tasks where missing positive instances is undesirable.

5.2 Results

5.2.1 Presentation of Evaluation Results

Evaluation results for each classifier are presented, including accuracy, precision, recall, and confusion matrices. The confusion matrix provides a detailed breakdown of the classifier's predictions, showing true positives, true negatives, false positives, and false negatives.

5.2.2 Comparison of Classifier Performance

The performance of different classifiers is compared based on their evaluation metrics. The most effective model(s) are identified based on their ability to achieve high accuracy, precision, recall, and F1-score.

In summary, model evaluation involves assessing the performance of classifiers using various evaluation metrics and comparing their results to identify the most effective model(s) for sentiment analysis tasks.

Chapter 6: Hyperparameter Tuning

6.1 Grid Search Cross-Validation

6.1.1 Description of Hyperparameter Tuning

Hyperparameter tuning is a critical step in optimizing the performance of machine learning models. Grid Search Cross-Validation (GridSearchCV) is a technique used to systematically search for the best combination of hyperparameters for a given classifier.

6.1.2 Explanation of GridSearchCV

In GridSearchCV, a grid of hyperparameters is defined, and the model is trained and evaluated for each combination of hyperparameters using cross-validation. Cross-validation helps ensure that the model's performance is robust and not overly sensitive to the specific training and testing data splits.

6.1.3 Hyperparameters Tuned

For the selected classifier(s), hyperparameters are tuned to optimize model performance. The hyperparameters tuned may vary depending on the classifier, but common ones include regularization parameters, kernel types, and tree depths.

6.1.4 Impact on Model Performance

Tuning hyperparameters can have a significant impact on the model's performance. By finding the optimal combination of hyperparameters, the model can better capture the underlying patterns in the data, leading to improved accuracy, precision, recall, and overall effectiveness.

6.2 Tuned Model Evaluation

6.2.1 Presentation of Results

After hyperparameter tuning, the results are presented, including evaluation metrics such as accuracy, precision, recall, and confusion matrices. A comparison is made between the tuned models and the baseline models to assess the improvement in performance.

6.2.2 Discussion on Improvement

The improvement in model performance achieved through hyperparameter tuning is discussed. This may include insights into which hyperparameters had the most significant impact on performance and how the tuned models compare to the baseline models in terms of predictive accuracy and generalization capability.

In summary, hyperparameter tuning using techniques like GridSearchCV is a crucial step in optimizing the performance of machine learning models. By systematically searching for the best combination of hyperparameters, models can achieve higher accuracy and better generalization to unseen data.

Chapter 7: Prediction and Application

7.1 Predictive Analysis

7.1.1 Demonstration of Predictive Analysis

In predictive analysis, the trained sentiment classifier(s) are deployed to predict the sentiment of new, unseen reviews. This process involves several steps:

Preprocessing of New Reviews: Any new reviews obtained by the restaurant business or from external sources need to undergo the same preprocessing steps applied during model training. This includes removing non-alphabetic characters, converting text to lowercase, removing stopwords, and stemming.

Vectorization of Preprocessed Text: After preprocessing, the text data is transformed into numerical vectors using the same feature extraction techniques applied during training, such as the Bag of Words representation with CountVectorizer.

Prediction: The preprocessed and vectorized reviews are then input into the trained sentiment classifier(s) to predict their sentiment labels. The classifier

assigns a sentiment label (positive or negative) to each review based on the learned patterns in the training data.

7.1.2 Example Predictions

Example predictions are provided to demonstrate the effectiveness of the sentiment classifier(s) in accurately classifying new reviews. For instance:

Positive Review Prediction: A positive review such as "The food was amazing, and the service was excellent!" should be correctly classified as positive by the sentiment classifier.

Negative Review Prediction: A negative review like "The food was cold, and the service was slow and unfriendly." should be correctly classified as negative.

These example predictions showcase how the sentiment classifier(s) can effectively analyze and categorize the sentiment expressed in new reviews, enabling businesses to gain insights into customer opinions.

7.2 Business Application

7.2.1 Discussion on Implications

The implications of sentiment analysis findings for businesses in the restaurant industry are profound. Understanding customer sentiments can provide valuable insights into various aspects of the business, including:

Customer Satisfaction: Sentiment analysis helps gauge overall customer satisfaction levels by analyzing the sentiment expressed in reviews. Positive sentiments indicate satisfied customers, while negative sentiments signal areas for improvement.

Menu Optimization: Analyzing sentiment towards specific menu items can help identify popular dishes that resonate well with customers and unpopular items that may need improvement or removal from the menu.

Service Improvement: Sentiment analysis of reviews can uncover patterns related to service quality, helping businesses identify areas for improvement in customer service, staff behavior, responsiveness, and overall dining experience.

7.2.2 Suggestions for Utilizing Insights

To leverage sentiment analysis insights effectively, businesses can implement various strategies, such as:

Tailoring Marketing Strategies: Use positive sentiment from reviews to highlight strengths in marketing campaigns, while addressing negative sentiments through targeted efforts to improve customer experience.

Operational Adjustments: Act on feedback provided by sentiment analysis to make operational adjustments, such as refining menu offerings, training staff, or improving service processes.

Proactive Customer Engagement: Reach out to customers who express negative sentiments in reviews to address their concerns, offer apologies, and demonstrate a commitment to resolving issues, thereby improving customer loyalty.

Continuous Monitoring: Implement a system for continuous monitoring of sentiment trends over time to track the effectiveness of implemented changes and identify emerging issues or trends that require attention.

By incorporating sentiment analysis insights into their business operations, restaurants can better understand customer preferences, enhance service quality, and ultimately drive customer satisfaction and loyalty.

Chapter 8: Conclusion

8.1 Summary of Findings

8.1.1 Recap of Key Findings

In summary, this project aimed to analyze sentiment in restaurant reviews using machine learning techniques. Key findings from the project include:

Sentiment analysis of restaurant reviews provides valuable insights into customer opinions and satisfaction levels.

The trained sentiment classifier(s) demonstrated effectiveness in accurately classifying reviews as positive or negative based on the sentiment expressed.

Insights gained from sentiment analysis can inform various aspects of restaurant operations, including menu optimization, service improvement, and marketing strategies.

8.1.2 Overview of Strengths and Limitations

Strengths of the sentiment analysis approach used in the project include its ability to process large volumes of text data efficiently and provide actionable insights for businesses. However, limitations such as the reliance on predefined features and the potential for bias in the training data should be acknowledged.

8.2 Future Directions

8.2.1 Suggestions for Future Research

Exploration of Advanced Techniques: Future research could explore advanced sentiment analysis techniques, such as deep learning models, to further improve accuracy and robustness.

Integration of External Data Sources: Incorporating additional data sources, such as social media mentions or customer surveys, could enrich sentiment analysis findings and provide a more comprehensive understanding of customer sentiment.

Aspect-Based Sentiment Analysis: Analyzing sentiment at a more granular level by considering specific aspects of the dining experience, such as food quality, service, ambiance, etc., could provide deeper insights into areas for improvement.

8.2.2 Potential Extensions

Multi-Language Support: Extending sentiment analysis capabilities to support multiple languages could broaden the applicability of the approach to restaurants operating in diverse linguistic environments.

Real-Time Analysis: Developing real-time sentiment analysis systems that can continuously monitor and analyze incoming reviews could enable businesses to respond promptly to customer feedback and address issues in a timely manner.

Integration with Recommendation Systems: Integrating sentiment analysis with recommendation systems could enhance personalized dining experiences by suggesting menu items or promotions based on individual preferences and sentiments.

In conclusion, sentiment analysis of restaurant reviews offers valuable insights for businesses seeking to enhance customer satisfaction and drive operational improvements. By addressing the identified findings and exploring future research directions, businesses can leverage sentiment analysis to optimize their offerings and foster stronger customer relationships.

Chapter 9: References

1. Pang, Bo, and Lillian Lee. "Opinion mining and sentiment analysis." *Foundations and Trends® in Information Retrieval* 2, no. 1–2 (2008): 1-135.

2. Turney, Peter D. "Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews." In Proceedings of the 40th annual meeting on association for computational linguistics, pp. 417-424. Association for Computational Linguistics, 2002.
3. NLTK Documentation: Natural Language Toolkit. Available online: <https://www.nltk.org/>
4. Scikit-learn Documentation: Machine Learning in Python. Available online: <https://scikit-learn.org/stable/documentation.html>
5. Pandas Documentation: Data manipulation in Python. Available online: <https://pandas.pydata.org/docs/>
6. Matplotlib Documentation: Data visualization in Python. Available online: <https://matplotlib.org/stable/contents.html>
7. Seaborn Documentation: Statistical data visualization. Available online: <https://seaborn.pydata.org/>
8. Restaurant Reviews Dataset: [Source URL or Description of Dataset]

9. Porter, Martin F. "An algorithm for suffix stripping." Program: electronic library and information systems 14, no. 3 (1980): 130-137.
10. GridSearchCV Documentation: Hyperparameter tuning in scikit-learn. Available online: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
11. Multinomial Naive Bayes Classifier Documentation: Scikit-learn documentation. Available online: https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html
12. Support Vector Machine Classifier Documentation: Scikit-learn documentation. Available online: <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>
13. Decision Tree Classifier Documentation: Scikit-learn documentation. Available online: <https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>
14. Random Forest Classifier Documentation: Scikit-learn documentation. Available online: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>

15. AdaBoost Classifier Documentation: Scikit-learn documentation. Available online: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html>

16. Gradient Boosting Classifier Documentation: Scikit-learn documentation. Available online: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html>

Chapter 10: APPENDICES

10.1 TRAINING AND EVALUATION OF VARIOUS ML ALGORITHMS

```
In [10]: # Training the classifier
classifier = MultinomialNB()
classifier.fit(X_train, y_train)

Out[10]: MultinomialNB()

In [11]: # Evaluating the classifier
y_pred = classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)

In [12]: # Displaying results
print("\nMultinomial Naive Bayes Classifier:")
print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")

Multinomial Naive Bayes Classifier:
Accuracy: 0.77
Precision: 0.76
```

Figure 1 MULTINOMIAL NAIVE BAYES

In [14]: **# Support Vector Machine (SVM) Classifier - Model Training and Testing**

```
from sklearn.svm import SVC

# Training the classifier
classifier = SVC(kernel='linear', C=1.0, random_state=0)
classifier.fit(X_train, y_train)

# Evaluating the classifier
y_pred = classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)

# Displaying results
print("\nSupport Vector Machine (SVM) Classifier:")
print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print("Confusion Matrix:")
print(cm)
```

```
Support Vector Machine (SVM) Classifier:
Accuracy: 0.72
Precision: 0.75
Recall: 0.68
Confusion Matrix:
[[74 23]
 [33 70]]
```

In [15]: **# Decision Tree Classifier - Model Training and Testing**

```
from sklearn.tree import DecisionTreeClassifier

# Training the classifier
classifier = DecisionTreeClassifier(random_state=0)
classifier.fit(X_train, y_train)

# Evaluating the classifier
y_pred = classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)

# Displaying results
print("\nDecision Tree Classifier:")
print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print("Confusion Matrix:")
print(cm)
```

```
Decision Tree Classifier:
Accuracy: 0.65
Precision: 0.69
Recall: 0.57
Confusion Matrix:
[[71 26]
 [44 59]]
```

Figure 2 DECISION TREE CLASSIFIER

In [16]:  *#Random Forest Classifier - Model Training and Testing*

```
from sklearn.ensemble import RandomForestClassifier

# Training the classifier
classifier = RandomForestClassifier(random_state=0)
classifier.fit(X_train, y_train)

# Evaluating the classifier
y_pred = classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)

# Displaying results
print("\nRandom Forest Classifier:")
print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print("Confusion Matrix:")
print(cm)
```

```
Random Forest Classifier:
Accuracy: 0.70
Precision: 0.82
Recall: 0.53
Confusion Matrix:
[[85 12]
 [48 55]]
```

Figure 3 RANDOM FOREST CLASSIFIER

In [18]: **# Gradient Boosting Classifier - Model Training and Testing**

```
from sklearn.ensemble import GradientBoostingClassifier

# Training the classifier
classifier = GradientBoostingClassifier(random_state=0)
classifier.fit(X_train, y_train)

# Evaluating the classifier
y_pred = classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)

# Displaying results
print("\nGradient Boosting Classifier:")
print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print("Confusion Matrix:")
print(cm)
```

```
Gradient Boosting Classifier:
Accuracy: 0.74
Precision: 0.90
Recall: 0.55
Confusion Matrix:
[[91  6]
 [46 57]]
```

Figure 5 GRADIENT BOOSTING CLASSIFIER

In [19]: **# Logistic Regression Classifier - Model Training and Testing**

```
from sklearn.linear_model import LogisticRegression

# Training the classifier
classifier = LogisticRegression(max_iter=1000)
classifier.fit(X_train, y_train)

# Evaluating the classifier
y_pred = classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)

# Displaying results
print("\nLogistic Regression Classifier:")
print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print("Confusion Matrix:")
print(cm)
```

```
Logistic Regression Classifier:
Accuracy: 0.71
Precision: 0.76
Recall: 0.64
Confusion Matrix:
[[76 21]
 [37 66]]
```

Figure 4 REGRESSION CLASSIFIER

In [20]: **#K-Nearest Neighbors (KNN) Classifier - Model Training and Testing**

```
from sklearn.neighbors import KNeighborsClassifier

# Training the classifier
classifier = KNeighborsClassifier()
classifier.fit(X_train, y_train)

# Evaluating the classifier
y_pred = classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)

# Displaying results
print("\nK-Nearest Neighbors (KNN) Classifier:")
print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print("Confusion Matrix:")
print(cm)
```

```
K-Nearest Neighbors (KNN) Classifier:
Accuracy: 0.58
Precision: 0.64
Recall: 0.46
Confusion Matrix:
[[70 27]
 [56 47]]
```

Figure 6 K-NEAREST NEIGHBORS

In [17]: **#AdaBoost Classifier - Model Training and Testing**

```
from sklearn.ensemble import AdaBoostClassifier

# Training the classifier
classifier = AdaBoostClassifier(random_state=0)
classifier.fit(X_train, y_train)

# Evaluating the classifier
y_pred = classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)

# Displaying results
print("\nAdaBoost Classifier:")
print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print("Confusion Matrix:")
print(cm)
```

```
AdaBoost Classifier:
Accuracy: 0.71
Precision: 0.88
Recall: 0.50
Confusion Matrix:
[[90  7]
 [51 52]]
```

Figure 7 ADABOOST CLASSIFIER

In [21]:  #Gaussian Naive Bayes Classifier - Model Training and Testing

```
from sklearn.naive_bayes import GaussianNB


# Training the classifier
classifier = GaussianNB()
classifier.fit(X_train, y_train)

# Evaluating the classifier
y_pred = classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)

# Displaying results
print("\nGaussian Naive Bayes Classifier:")
print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print("Confusion Matrix:")
print(cm)
```

```
Gaussian Naive Bayes Classifier:
Accuracy: 0.73
Precision: 0.68
Recall: 0.88
Confusion Matrix:
[[55 42]
 [12 91]]
```

Figure 9 GAUSSIAN NAIVE BAYES

In [22]:  #Extra Trees Classifier - Model Training and Testing

```
from sklearn.ensemble import ExtraTreesClassifier

# Training the classifier
classifier = ExtraTreesClassifier(random_state=0)
classifier.fit(X_train, y_train)

# Evaluating the classifier
y_pred = classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)

# Displaying results
print("\nExtra Trees Classifier:")
print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print("Confusion Matrix:")
print(cm)
```

```
Extra Trees Classifier:
Accuracy: 0.70
Precision: 0.78
Recall: 0.59
Confusion Matrix:
[[80 17]
 [42 61]]
```

Figure 8 EXTRA TREE CLASSIFIER

10.2 RESULTS

```
In [27]: # Example review
review_example = "The food was delicious and the service was excellent!"

# Preprocess the review
review_example = re.sub('[^a-zA-Z]', ' ', review_example)
review_example = review_example.lower()
review_words = review_example.split()
review_words = [word for word in review_words if not word in set(stopwords.words('english'))]
review_example = [ps.stem(word) for word in review_words]
review_example = ' '.join(review_example)

# Vectorize the preprocessed review
review_example_vectorized = cv.transform([review_example]).toarray()

# Predict sentiment using the trained classifier
sentiment_prediction = classifier.predict(review_example_vectorized)

# Map sentiment label to human-readable format
sentiment_label = "Positive" if sentiment_prediction[0] == 1 else "Negative"

print("Review:", review_example)
print("Predicted Sentiment:", sentiment_label)

Review: food delici servic excel
Predicted Sentiment: Positive
```

Figure 10 RESULT OF POSITIVE REVIEW

```
In [28]: # Example negative review
review_example_negative = "The food was terrible, and the service was slow and rude."

# Preprocess the review
review_example_negative = re.sub('[^a-zA-Z]', ' ', review_example_negative)
review_example_negative = review_example_negative.lower()
review_words = review_example_negative.split()
review_words = [word for word in review_words if not word in set(stopwords.words('english'))]
review_example_negative = [ps.stem(word) for word in review_words]
review_example_negative = ' '.join(review_example_negative)

# Vectorize the preprocessed review
review_example_negative_vectorized = cv.transform([review_example_negative]).toarray()

# Predict sentiment using the trained classifier
sentiment_prediction_negative = classifier.predict(review_example_negative_vectorized)

# Map sentiment label to human-readable format
sentiment_label_negative = "Negative" if sentiment_prediction_negative[0] == 0 else "Positive"

print("Review:", review_example_negative)
print("Predicted Sentiment:", sentiment_label_negative)

Review: food terribl servic slow rude
```

Figure 11 RESULT OF NEGATIVE REVIEW