Department of Computer Science

MSc DATA SCIENCE AND ANALYTICS

Academic Year 2022-2023

Sentiment Analysis on Hotel Reviews

Sachin Singh; 2248524

A report submitted in partial fulfilment of the requirement for the degree of Master of Science

Brunel University
Department of Computer Science
Uxbridge, Middlesex UB8 3PH
United Kingdom
Tel: +44 (0) 1895 203397
Fax: +44 (0) 1895 251686

ABSTRACT

In the digital age, an intense shift in how customers share their experiences and viewpoints, primarily on digital platforms, has been witnessed by the hospitality industry. Enthusiastically expressed sentiments about hotels and accommodations by tourists now contribute to a heap of unfiltered opinions. At the union of natural language processing and data science lies sentiment analysis, an expanding field dedicated to unravelling these emotions.

In the hospitality sector, guest experiences hold paramount importance, and the emergence of online review platforms has revolutionised the way tourists express their opinions. The analysis of these sentiments has become crucial for hoteliers, marketers, and policymakers alike. Positive reviews serve as potent marketing tools, while negative ones shed light on areas requiring improvement. However, the substantial volume of daily textual data presents significant challenges. This challenge is addressed in this research through the application of sentiment analysis to 515,000 hotel reviews, unveiling insights into customer sentiments regarding European hotels. The Knowledge Discovery in Databases (KDD) methodology is followed, involving data preprocessing, feature extraction, and sentiment categorisation, facilitated by advanced machine learning techniques. This approach reveals hidden sentiments and identifies patterns that shape guest experiences.

A comprehensive understanding of customer sentiments regarding European hotels is yielded by this research, offering reflective insights and implications for the active hospitality industry. Through sentiment analysis, the factors that contribute to guest experiences, both positively and negatively, are determined. Armed with these insights, hoteliers gain the power to tailor their services to align with customer expectations, thereby enhancing guest satisfaction. Positive sentiments disclosed in reviews can be strategically leveraged for marketing purposes, while negative sentiments serve as signposts, guiding efforts to enhance service quality. Ultimately, the significance of sentiment analysis in navigating the ever-expanding scope of textual data in the digital era is underscored by the work presented in the report.

ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude to my parents for their unwavering support and guidance throughout my academic journey. Their love and encouragement have been my constant motivation. Your sacrifices and belief in me have been the driving force behind my accomplishments.

I am deeply indebted to Dr. Stephen Swift for his invaluable guidance and mentorship during my dissertation. His expertise and insights have been instrumental in shaping my research, and I extend my sincere gratitude, for his invaluable guidance, expertise, and unwavering commitment to my academic growth. His mentorship and insightful feedback have been instrumental in shaping this dissertation.

I also want to extend my thanks to my friends at Brunel University, my flatmates, who have been my pillars of strength during my tough time here. Your amity and assistance have made my university experience comfortable and memorable.

To everyone who has been a part of this journey, your support has meant the world to me. Thank you for believing in me and helping me reach this milestone.

I certify that the work presented in the dissertation is my own unless referenced.

Signature: [illegible]

Date: 28/02/2024

TOTAL NUMBER OF WORDS

Word Count: 14305

TABLE OF CONTENTS

ABSTRACT 2

ACKNOWLEDGEMENTS 3

1. INTRODUCTION 7

1.1. The Digital Landscape of Sentiment Expression 7

1.2. Research Objectives 7

1.3. Methodology: Guided by KDD 9

1.4. Significance of KDD 9

1.5. Data Preprocessing 9

1.6. Feature Extraction 10

1.7. Sentiment Categorisation 10

1.8. Interpretation of Results 10

1.9. Road Map 10

2. LITERATURE REVIEW 12

3. METHODOLOGY 17

3.1. Knowledge Discovery in Databases 17

3.2. Applications 17

3.3. Data Source and Selection 18

3.3.1. Dataset Description 19

3.4. Data Visualisation 21

3.5. Data Pre-processing 21

3.5.1. Data Cleaning 22

3.5.2. Data Transformation 22

3.5.3. Data Mining 22

3.6. Pattern Evaluation 24

3.7. Knowledge Presentation 26

3.8. Software / Tools Employed 26

3.9. Ethical Preview 27

4. DESIGN 28

4.1. Unified Modeling Language Design of sentiment analysis 28

4.1.1. Classes 29

4.1.2. Relationships 29

4.3. Data Set Structure 30
4.4. Processing 30
4.5. System Structure 30
4.6. Sentiment Analysis Analogy 31
4.7. Classification 31
4.8. System Environment 32
4.9. Data Dictionary 32
4.10. Collation of Results 35
5. IMPLEMENTATION AND TESTING 36
5.1. Data Visualisation 36
5.2. Analysis Initiation 39
5.3. Machine Learning 42
5.4. Tools 44
5.5. Code Implementation 45
5.6. Testing 46
5.6.1. Model Evaluation 46
5.6.2. Unit Testing 47
6. RESULTS 49
6.1. Evaluation of Model 50
6.1.1. Accuracy and Classification Report 50
6.1.2. Precision, Recall, and F1-Score 50
6.1.3. Feature Importance 51
6.1.4. Confusion Matrix 51
6.1.5. Cross-Validation Results 52
6.2. Knowledge Presentation 53
6.2.1. Sentiment Distribution by Reviewer Score 53
6.2.2. Top Hotels by Sentiment 53
6.2.3. Sentiment Trends Over Time 54
6.2.4. Sentiment vs. Review Length 55
6.2.5. Geographical Hotspots of Negative Sentiment 56
6.3. Unit Testing 56
6.4. Discussion 57
6.5. Limitations 58

7. CONCLUSION 59
7.1. Self-Reflections 61
7.2. Future Works 61
7.3. File Handling 62
Reference 63
APPENDIX A: ETHICAL APPROVAL 66
APPENDIX B: APPLIED EXPERIMENTED CODE 67

CHAPTER 1

1. INTRODUCTION

In an era marked by the omnipresence of digital platforms and the constant sharing of experiences, the analysis of sentiments expressed in textual data has emerged as a fundamental component of Information Systems and Computer Science. This dissertation embarks on a journey into the realm of sentiment analysis, with a specific focus on hotel reviews within the broader field of Applied Computing. Sentiment analysis, often referred to as opinion mining, plays a pivotal role in the realm of text analytics (Liu, 2012). It equips us with the tools to decipher, evaluate, and comprehend the emotions, opinions, and attitudes conveyed within textual data. The burgeoning growth of this field in recent years is attributed to the proliferation of digital text data on the internet and social media platforms (Pang & Lee, 2008). In this research, a comprehensive exploration of sentiment analysis, applying its methodologies to unveil the intricate tapestry of sentiments concealed within an extensive dataset comprising 515,000 hotel reviews sourced from Europe is achieved.

1.1. The Digital Landscape of Sentiment Expression

The introduction of digital platforms and online review sites has standardised the process of expressing opinions, especially in the context of the hospitality industry. Travelers and tourists readily share their experiences, impressions, and recommendations regarding hotels and accommodations. These unfiltered narratives, scattered across the digital landscape, constitute a treasure trove of textual data that holds valuable insights into customer sentiments and preferences, particularly concerning European hotels.

1.2. Research Objectives

The primary objective of this research is to utilise the formidable capabilities of sentiment analysis to uncover the latent sentiments concealed within hotel reviews. Through the analysis of extensive textual data, a comprehensive understanding of customer feelings and attitudes regarding European hotels is sought. The overarching aim of this dissertation is to explore and analyse the sentiments expressed within a substantial dataset of 515,000 hotel reviews sourced from European destinations. Within this context, the following objectives are outlined.

The research objectives outlined in your description collectively aim to comprehensively analyse and understand the dynamics of hotel review sentiments from various dimensions. The first objective delves into the temporal aspect, seeking to track how sentiments expressed in hotel reviews evolve over time using the VADER sentiment analysis tool. This analysis allows for the identification of trends, seasonal variations, and potential factors influencing shifts in sentiment perception, thereby offering insights into the temporal dynamics of hotels' performance as perceived by guests.

The second objective introduces a geographical perspective by utilising latitude and longitude data to identify specific geographic hotspots that consistently receive either positive or negative reviews during particular times of the year. This spatial analysis uncovers regional patterns and provides valuable information for hotels and stakeholders to pinpoint areas where improvements or promotional efforts may be needed, shedding light on the geographical facets of customer sentiment.

Moving to the third objective, the research explores reviewer behaviour and sentiments, aiming to discern whether distinct reviewer profiles, characterised by attributes such as "Reviewer_Nationality" and "Total_Number_of_Reviews_Reviewer_Has_Given", exhibit tendencies towards providing more positive or negative reviews. By analysing reviewer demographics and reviewing habits, this objective unveils potential biases or preferences in reviewing, thereby revealing the influence of reviewer attributes on sentiment expression.

Objective four delves into investigating the relationship between sentiments and specific tags extracted from the "Tags" column in the dataset. This analysis seeks to identify particular aspects of the hotel experience that consistently receive praise or criticism from guests. Understanding which aspects hold the most significant impact in shaping sentiment allows hotels to prioritise improvements or marketing strategies effectively.

Objective five introduces machine learning techniques, including Random Forest and Neural Networks, to categorise sentiments related to different aspects of the hotel experience and predict the overall sentiment. This application of machine learning offers the potential for automated sentiment classification at scale, providing real-time insights that can be invaluable for hotel management.

Lastly, objective six focuses on exploring the impact of review length and depth, as indicated by "Review_Total_Positive_Word_Counts" and "Review_Total_Negative_Word_Counts," on overall sentiment. This analysis aims to uncover whether more detailed reviews tend to be more positive or negative, emphasising the importance of encouraging comprehensive guest feedback. In sum, these research objectives collectively contribute to a holistic understanding of hotel review sentiments, encompassing temporal, spatial, demographic, and content-related dimensions for informed decision-making in the hospitality industry.

1.3. Methodology: Guided by KDD

To fulfil the research objectives, the Knowledge Discovery in Databases (KDD) methodology (Rajput, 2018) has been chosen for adoption. KDD is regarded as a structured and systematic approach designed to extract valuable insights and knowledge from data. It encompasses several sequential stages, including data preprocessing, feature extraction, sentiment categorisation, and the interpretation of results. This well-structured methodology provides a clear roadmap for navigating the complexities of sentiment analysis.

1.4. Significance of KDD

KDD involves the identification of authentic, innovative, potentially valuable, and ultimately comprehensible patterns and connections within data (Rajput, 2018). The decision to choose KDD as the venue for presenting the research findings was based on several factors that underscore its suitability as a platform for disseminating scholarly work in the field of data mining and knowledge discovery. Structured data, such as guidelines and frameworks, which enable informed decision-making or forecasting. KDD, known as the premier international conference in the field, offers a prestigious platform for researchers, practitioners, and academicians to exchange ideas, share insights, and explore the latest advancements in data mining and knowledge discovery. Its rigorous peer-review process ensures the quality and credibility of accepted papers, enhancing the visibility and impact of the research within the academic community (Smith, 2022). While other options such as IEEE International Conference on Data Mining (ICDM) and ACM SIGMOD Conference were considered, KDD stood out for its focus on innovative research, diverse program offerings, and global reach, aligning closely with the research objectives and audience target of the study.

1.5. Data Preprocessing

The study commences with data preprocessing, a foundational step that plays a pivotal role in ensuring the accuracy and reliability of the analysis. During this phase, the raw textual data is meticulously cleaned and refined. This process involves a series of operations aimed at preparing the data for subsequent analysis. Noise, inconsistencies, and irrelevant information are meticulously eliminated to ensure that the text is in a format suitable for analysis. Textual data often comes fraught with various challenges, including misspellings, grammatical errors, and superfluous characters. The success of the sentiment analysis hinges on the researchers' ability to rectify these issues and present a clean, structured dataset for further examination.

1.6. Feature Extraction

Once the data is suitably pre-processed, the research transitions to the feature extraction phase. In this stage, pertinent features are identified and extracted from the textual content. These features can encompass a wide array of attributes, including keywords, sentiment markers, and context-related information. The extraction of these features empowers the researchers to capture the essence of each review, distilling it into a structured representation that can be effectively processed by machine learning algorithms.

1.7. Sentiment Categorisation

A suite of pre-trained machine learning algorithms and natural language processing techniques are deployed to assess and classify the sentiment expressed within each review. Sentiment categorisation involves the assignment of each review to one of three categories: positive, negative, or neutral, based on its emotional tone. This classification enables the identification and quantification of the emotional tone of each review, allowing for the discernment of patterns and trends in customer sentiments.

1.8. Interpretation of Results

The ultimate goal of the research is to derive actionable insights from the sentiment analysis of hotel reviews. By categorising sentiments as positive, negative, or neutral, underlying patterns and trends are aimed to be uncovered. This analysis places a spotlight on the factors that influence customer experiences, highlighting both positive aspects that garner praise and negative facets that elicit criticism. Such insights are deemed to hold immense value for hoteliers and the broader hospitality industry, providing them with the knowledge necessary to enhance customer satisfaction and tailor their services to meet customer expectations.

1.9. Road Map

The motivation for studying hotel review sentiment is found in its potential to drive positive change, enhance guest experiences, and support various aspects of the hospitality industry, from marketing and operations to investment decisions and academic research. It enables hotels to respond effectively to guest feedback and facilitates the provision of better experiences for tourits worldwide.

Chapter 2 Literature review synthesises relevant literature on the core concepts and techniques employed in the study, including sentiment analysis, machine learning, data preprocessing, and knowledge discovery. It provides context by summarising key developments in sentiment analysis as an emerging field, from early work using machine learning classifiers to recent advances with deep learning. The application of sentiment analysis specifically to hotel reviews is discussed, along with familiar challenges. Related work leveraging techniques like VADER, Random Forests, and KDD are reviewed. The literature review situates this study within existing research and establishes the background knowledge informing the methodology.

The methodology chapter 3 carry informative Knowledge Discovery in Databases (KDD) process followed throughout the analysis. It provides details on the key stages including data selection, preprocessing, transformation, mining, evaluation, and knowledge presentation. The methodology acts as a framework guiding the design and implementation of the sentiment analysis system. Relevant studies highlighting applications of the KDD methodology are referenced.

In Chapter 4, the system design is described, outlining its core components such as the VADER sentiment analyser and Random Forest classifier. The design decisions regarding techniques, algorithms, and tools are justified based on insights from the literature. Details are provided on steps ranging from data exploration and feature engineering to model selection, parameter tuning, and evaluation metrics. The overall structure aims to achieve the objectives of effectively analysing hotel review text and metadata to generate sentiment insights.

Chapter 5 presents the implementation of the proposed design using Python and associated machine learning libraries. It outlines the key experiments conducted, including data preprocessing, fitting models, performance evaluation, and visualisation. Results are reported through confusion matrices, classification reports, feature importance analysis, and other metrics that assess the effectiveness of the sentiment analysis techniques. Findings are interpreted to derive meaningful conclusions that address the original research objectives. The chapter culminates in a discussion of the methods' limitations and suggestions for future work.

CHAPTER 2

2. LITERATURE REVIEW

The roots of sentiment analysis can be traced back to the early 2000s when researchers began exploring techniques to gauge public opinion and emotions from textual data. Pang and Lee (2008) made significant contributions by introducing the concept of sentiment classification using machine learning algorithms. Their work laid the foundation for subsequent developments in the field. Sentiment analysis, often referred to as opinion mining, has rapidly evolved into a dynamic field with diverse applications across various domains, particularly in the consumer sector (Pang & Lee, 2008). Sentiment analysis has found extensive use in understanding customer opinions on products and services. It provides a powerful means of extracting valuable insights from textual data, shedding light on the opinions and sentiments of individuals.

In today's digital age, it is employed to analyse data from a multitude of sources, including social media platforms, blogs, web forums, articles, tweets, and user feedback (Zhang, Zhao, & LeCun, 2015). By processing this vast amount of textual data, businesses gain valuable insights into customer satisfaction, identify areas for improvement, and make data-driven decisions.

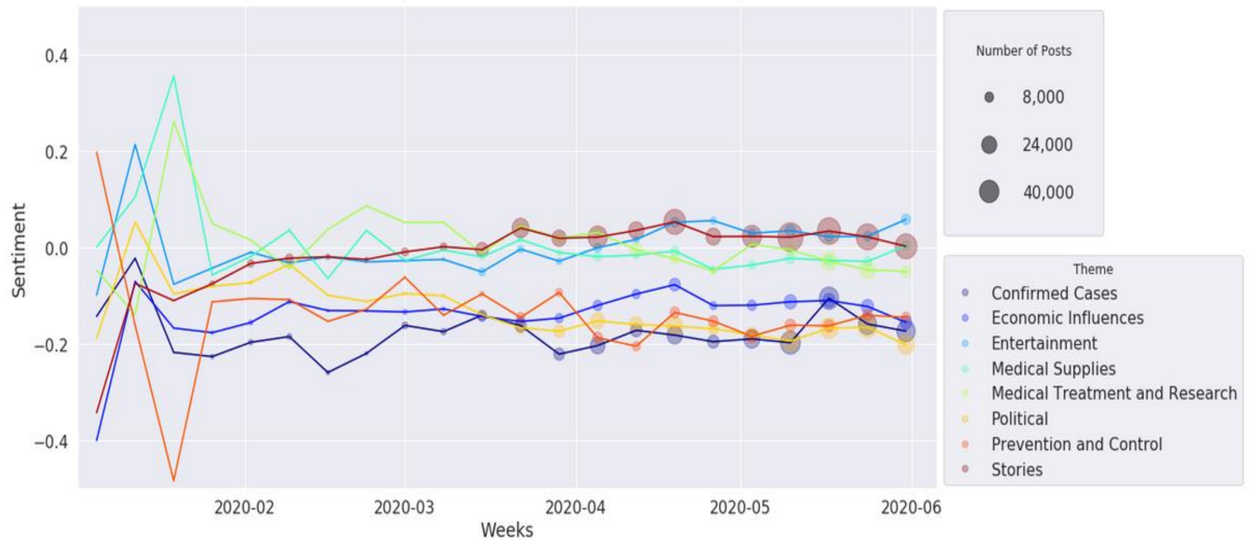


Fig. 2.1. Twitter sentiment analysis over time (De Melo and Figueiredo, 2021).

The evolution of sentiment analysis techniques and algorithms has led to significant advancements. These innovations span various areas, including transfer learning, emotion detection, and resource construction (Tang, Qin, & Liu, 2015). These techniques aim to analyse the opinions, sentiments, and subjectivity expressed in text data, which can be structured, semistructured, or non-structured. The data is then categorised as positive, negative, or neutral based on the user's attitude towards a particular topic.

The rise of social media platforms has extended the application of sentiment analysis to gain insights into public opinion on diverse topics. De Melo and Figueiredo (2021) employed sentiment analysis and topic modelling to compare news articles and tweets about COVID-19 in Brazil. Their study demonstrated the potential of these techniques in understanding public sentiment during a global health crisis. Similarly, Priyantina and Sarno (2019) utilised techniques like Latent Dirichlet Allocation, semantic similarity, and Long Short-Term Memory (LSTM) to analyse hotel reviews. Their work highlighted the effectiveness of these methods in extracting meaningful insights from customer feedback.

Hotel reviews are a goldmine of information for the hospitality industry. They reflect the diverse experiences of guests, encapsulating their praises, criticisms, and preferences. This user-generated content serves as a valuable source of information about customer preferences, satisfaction levels, and pain points (Li, Chen, & Huang, 2016). Researchers often analyse this content to identify trends, sentiments, and patterns in customer feedback.

The emergence of platforms like Booking.com, TripAdvisor, Expedia, Airbnb, etc. has revolutionised the way tourits plan their journeys, from booking accommodations to discovering local attractions. In this context, data collection from these online platforms has become essential for understanding consumer behaviour, improving services, and making data-driven decisions. One of the primary methods for data collection from travel and booking websites is web scraping (Zhang, Zhao, & LeCun, 2015). Researchers and businesses utilise web scraping techniques to extract structured and unstructured data from these platforms (Liu, 2017). By doing so, they gain access to a wealth of information, including user reviews, ratings, pricing, and availability. Web scraping tools and libraries have evolved significantly, allowing for efficient data extraction from complex web pages.

While the data may be accessible to the public, ethical considerations regarding data ownership, privacy, and responsible use should guide the handling and analysis of scraped data from Booking.com. The ethical preview is based on considerations of data ownership, privacy, and responsible data use outlined in academic literature and professional ethical guidelines in data science and research ethics (Floridi, 2018 and Mittelstadt and Floridi, 2015).

Data preprocessing and cleaning play a crucial role in the data analysis pipeline. It involves a series of steps to ensure that the data used for analysis is accurate, complete, and reliable. Data preprocessing refers to the transformation of raw data into a usable format, while data cleaning focuses on identifying and rectifying errors, inconsistencies, and missing values. These tasks are essential to enhance the quality and reliability of data for analysis (Huang et al., 2015).

Data cleaning is a fundamental step in data analysis. Inaccurate or missing data can lead to biased results and incorrect conclusions. Researchers have emphasised the critical role of data cleaning in ensuring the validity of research findings (Broman & Woo, 2018). Missing data is a common issue in datasets. Researchers have explored strategies to address missing data, including imputation techniques, listwise deletion, and more advanced methods such as regression imputation (Little & Rubin, 2002).

Data transformation plays a pivotal role in preparing datasets for machine learning and data analysis tasks. In machine learning pipelines, it is often necessary to convert variables to integer type to ensure consistency and compatibility with algorithms. One common technique involves removing non-numeric characters from numeric columns using regular expressions and then converting the result to an integer datatype (McKinney, 2022). Categorical variables pose a challenge for many machine learning algorithms that require numerical input (Géron, 2019). Label Encoding is a standard technique used to convert categorical variables into numerical labels, making them suitable for processing by machine learning models (Raschka & Mirjalili, 2019).

One of the sectors where sentiment analysis has proven highly effective is the hospitality industry. Techniques like the Multinomial Naïve Bayes classifier have been employed to analyse hotel reviews, providing valuable insights into customer feedback (Farisi, Sibaroni, & Faraby, 2019). Similarly, the Naïve Bayes Classifier Optimisation technique has demonstrated the utility of sentiment analysis in understanding customer sentiment within the context of hotel reviews (Khomsah, 2020). Although, Random Forest classifiers have shown promising results in sentiment analysis tasks by leveraging ensemble learning techniques and robust feature selection methods (Zhang & Huang, 2016).

Despite its potential, sentiment analysis faces several challenges that need to be addressed in future research (Khomsah, 2020). Some of these challenges include issues with parallel computing for large data, identifying sarcasm, handling grammatically incorrect words, author segmentation, noise handling, and coping with the dynamism of language. These challenges can affect the accuracy of sentiment analysis and necessitate ongoing efforts to enhance its reliability and effectiveness.

In recent years, machine learning algorithms such as Random Forest and deep learning techniques have gained prominence in sentiment analysis. Random Forest, known for its ensemble learning capabilities, has been applied to various sentiment classification tasks (Breiman, 2001). Its ability to handle large datasets and provide feature importance scores makes it a valuable tool in sentiment analysis. Deep learning, on the other hand, offers the advantage of automatically learning features from data, making it particularly suitable for complex sentiment analysis tasks (LeCun, Bengio, & Hinton, 2015).

Lexicon based approaches leverage dictionaries of words annotated with sentiment orientation to determine overall text polarity and sentiment. VADER (Valence Aware Dictionary and Sentiment Reasoner) used in the code is a popular lexicon method optimised for social media text (Hutto and Gilbert, 2014). By using rules and heuristics in addition to sentiment lexicons, it achieves more nuanced analysis. Hybrid methods combining lexicon outputs as features into machine learning models have shown promise, demonstrating the complementary strengths of the two approaches (Hamilton et al., 2016).

The Valence Aware Dictionary and Sentiment Reasoner (VADER) is a lexicon and rule-based sentiment analysis tool that is widely used in sentiment analysis tasks (Hutto & Gilbert, 2014). VADER assigns polarity scores (positive, negative, or neutral) to individual words and then combines them to generate an overall sentiment score for a given text. Its simplicity and effectiveness have made it a popular choice in sentiment analysis projects, including the one discussed in this dissertation.

Different libraries offer various functionalities and features that can address several aspects of the problem. VADER analyses the sentiment of text by considering both the polarity (positive, negative, neutral) and the intensity of the sentiment expressed (Hutto & Gilbert, 2014). NLTK is a comprehensive Python library for natural language processing (NLP) tasks. It provides easy-to-use interfaces and functionalities for tasks such as tokenisation, stemming, lemmatisation, part-of-speech tagging, and more. NLTK also includes various corpora and lexical resources that are valuable for training and testing NLP models (Bird et al., 2009). Scikit-learn is a popular machine learning library in Python that provides a wide range of tools and algorithms for machine learning and data mining tasks. It includes modules for data preprocessing, feature selection, model evaluation, and various machine learning algorithms such as classification, regression, clustering, and dimensionality reduction (Pedregosa et al., 2011). Pandas is a powerful Python library for data manipulation and analysis. It provides data structures such as Data frame and Series, which allow for easy handling and manipulation of structured data. Pandas is widely used for tasks such as data cleaning, transformation, exploration, and visualisation (McKinney, 2022). Matplotlib is a versatile Python library for creating static, interactive, and publication-quality visualisation. It provides a wide range of plotting functions and customisation options for generating plots of data in various formats, including line plots, scatter plots, histograms, bar charts, and more (Hunter, 2007). Seaborn simplifies the process of creating complex visualisation such as heatmaps, violin plots, pair plots, and more, making it a valuable tool for data exploration and presentation (Waskom, 2022). Pillow is a Python Imaging Library (PIL) fork that adds support for opening, manipulating, and saving many different images file formats. It provides a rich set of functionalities for image processing tasks such as resizing, cropping, rotating, filtering, and more. Pillow is widely used in various applications requiring image manipulation and processing (Clark & Plummer, 2015).

More recently, transfer learning using large pretrained language models like BERT has achieved state-of-the-art results by fine-tuning on downstream sentiment analysis tasks (Sun et al., 2019). However, the code implementation uses Random Forest, which despite being less widely utilised than SVM or neural models, still demonstrates competitive accuracy due to built-in feature selection and ensemble aggregation. Past studies have successfully employed Random Forest for multiclass sentiment classification and found it capable of handling mixed feature types (Piryani et al., 2017).

CHAPTER 3

3. METHODOLOGY

This study utilises the Knowledge Discovery in Databases (KDD) methodology to conduct sentiment analysis on hotel reviews. The KDD process provides a structured framework for identifying useful patterns and knowledge from large datasets (Han, Kamber, & Pei, 2011). The major steps in the KDD methodology are selection, preprocessing, transformation, data mining, interpretation, and evaluation.

3.1. Knowledge Discovery in Databases

The Knowledge Discovery in Databases (KDD) methodology provides a systematic framework for extracting useful insights and knowledge from large datasets. The KDD process involves a series of defined steps that transform raw data into meaningful information.

As Rajput (2018) explains, the KDD methodology follows an iterative approach to uncover hidden patterns within data. It starts with data selection and preprocessing, followed by data transformation and feature engineering. The next phase applies data mining techniques to discover patterns. Finally, the results are interpreted and evaluated.

Implementing the structured KDD methodology enables researchers and data scientists to extract value from data effectively. It supports making data-driven decisions and developing predictive models for diverse applications including business analytics and scientific research. Overall, the KDD process delivers a methodical procedure to navigate the complexity of extracting knowledge from data. By adhering to its principles, impactful insights can be uncovered from large datasets across domains.

3.2. Applications

The KDD methodology incorporates automation and machine learning techniques to make data analysis more efficient (Han, Kamber & Pei, 2011). By leveraging these technologies, organisations can analyse large datasets quicker and more accurately. A key driver for adopting KDD is the growing volume, velocity, and variety of data across domains (Wu, Zhu, Wu & Ding, 2014). As big data expands, traditional analysis methods become inadequate, making KDD’s structured approach valuable. KDD facilitates processing and extracting insights from massive, complex datasets.

Sentiment analysis represents one area where KDD offers advantages. The research by De Melo and Figueiredo (2021) effectively applied KDD sentiment analysis and topic modelling to compare Twitter and news content about COVID-19 in Brazil. Additionally, KDD plays a critical role in detecting financial fraud. Chandola et al. (2009) used the KDD process to identify fraudulent credit card transactions.

While KDD provides value in sectors like sentiment analysis and fraud detection, its versatility supports implementation in any field where analysis guides decision-making. The specific applications may differ, but KDD’s core principles remain relevant in guiding knowledge discovery from data.

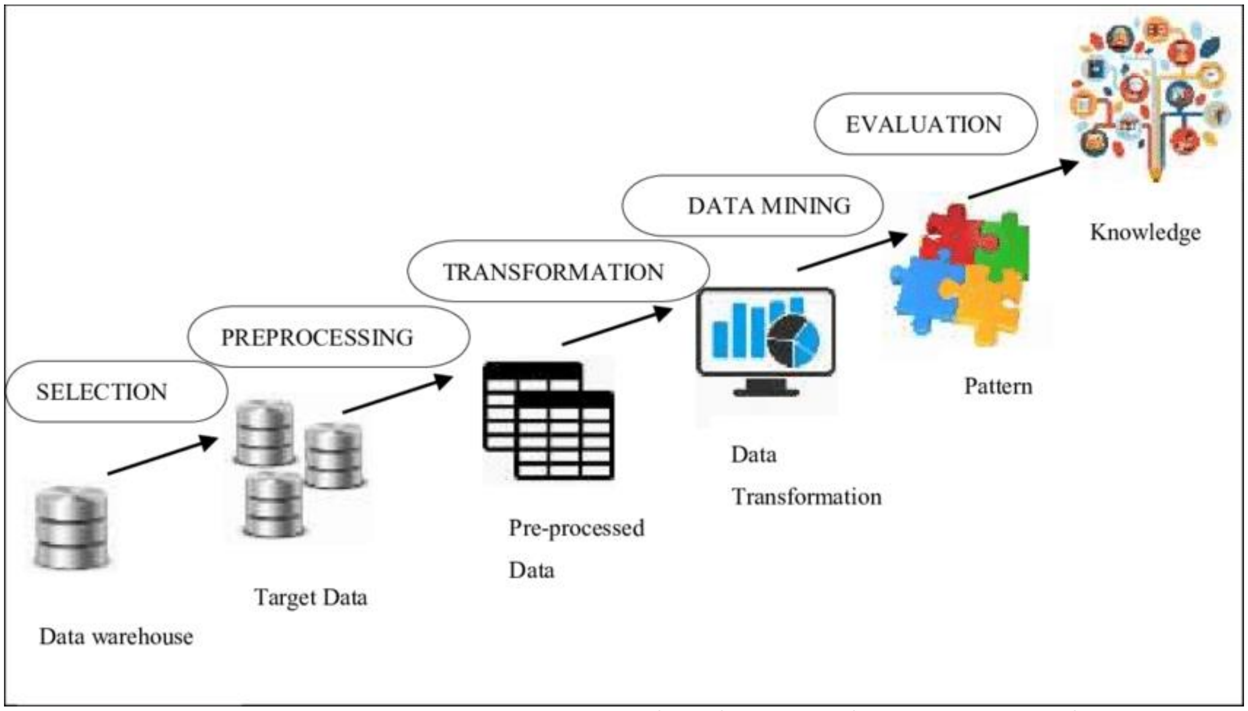


Fig.3.1 Knowledge Discovery Database (KDD) Process (Rose et al., 2019)

3.3. Data Source and Selection

The data source for this research is a dataset of 515,000 hotel reviews for European hotels compiled by Liu (2017). Web scraping techniques were used to extract relevant data fields from the Booking.com site, including key attributes such as hotel information, customer reviews, ratings, dates, and location details. The scripts to accumulate this data were applied directly to the publicly viewable Booking.com webpages, with no requirement for login or permissions. As Liu (2017) states in the dataset documentation, the entirety of the scraped information was already openly available online prior to compilation into a dataset. Web scraping refers to techniques for automatically collecting and extracting data from websites. As explained by Mitchell

(2015), web scraping involves writing computer scripts to harvest information from web pages. The scrapers mimic human web browsing to systematically visit sites, locate relevant data, and copy it into structured datasets. This dataset was selected due to its large volume of reviews and inclusion of key attributes such as hotel location, review text, ratings, and dates. The raw dataset was retrieved from Kaggle and sampled to obtain a more manageable subset of 50,000 reviews for model training and testing purposes. One of the standout features of this dataset is its sheer volume, comprising a substantial half-million hotel reviews. Such a large dataset is invaluable for researchers seeking to draw statistically significant conclusions and uncover nuanced patterns within the hospitality domain. The vast number of reviews reflects the diversity of experiences and perspectives of tourits who have stayed in European hotels. This diversity is essential for conducting comprehensive sentiment analysis and understanding the wide range of sentiments expressed by customers.

While the full dataset is extensive, it was necessary to strike a balance between comprehensiveness and manageability for model training and testing. Therefore, a strategically chosen subset of 50,000 reviews was sampled from the larger dataset. This subset retains the diversity and key attributes of the original dataset while making computational processes more efficient and manageable.

3.3.1. Dataset Description

The dataset under scrutiny has been meticulously curated from Kaggle (www.kaggle.com, n.d.), a distinguished platform for data science competitions and datasets. Comprising an extensive half a million hotel reviews, this dataset stands as a valuable resource for delving into the intricate world of hospitality and lodging. Each review encapsulates a traveller’s personal encounter with European hotels, encompassing their impressions, recommendations, and criticisms.

Variables	Description	Data Type
Hotel_Address	The physical address of the hotel.	String
Additional_Number_of_Scoring	Some guests may provide a numerical score for the service without providing a detailed review. This field indicates how many such scores exist for this hotel.	Integer
Review_Date	The date when the reviewer posted their review for the hotel.	Datetime
Average_Score	This is the average score of the hotel, which is calculated based on the latest comments made in	Float

Variables	Description	Data Type
	the last year. It provides an overall rating for the hotel's performance.	
Hotel_Name	The name of the hotel being reviewed.	String
Reviewer_Nationality	The nationality of the reviewer, indicating where the reviewer is from.	String
Negative_Review	The negative aspects or criticisms that the reviewer mentioned in their review. If there were no negative comments, it is labelled as "No Negative."	String
Review_Total_Negative_Word_Counts	The total number of words in the negative review. This can give an idea of the length or detail of the negative comments.	Integer
Total_Number_of_Reviews	This column represents the total number of valid reviews that the hotel has received. It provides insight into the hotel's popularity and the volume of reviews it has accumulated.	Integer
Positive_Review	This column contains the positive aspects of the review that the reviewer gave to the hotel. If the reviewer did not provide a positive review, the entry is "No Positive."	String
Review_Total_Positive_Word_Counts	This column contains the total number of words in the positive review. It quantifies the length of the positive review text.	Integer
Total_Number_of_Reviews_Reviewer_Has_Given	This column indicates the number of reviews that the reviewer has given in the past. It quantifies the reviewer's activity on the platform.	Integer
Reviewer_Score	This column represents the score that the reviewer has given to the hotel based on their experience. It provides a numerical rating of the hotel's quality.	Float
Tags	This column contains tags that the reviewer gave to the hotel. Tags are typically keywords or descriptors that summarise the reviewer's experience or impressions.	String
days_since_review	This column represents the duration between the review date and the date the data was scraped. It measures how long ago the review was posted.	Integer

Variables	Description	Data Type
lat	This column contains the latitude (geographical coordinate) of the hotel’s location.	Float
lng	This column contains the longitude (geographical coordinate) of the hotel’s location.	Float

Table. 1.1. Dataset Description.

3.4. Data Visualisation

The visualisation serve as foundational exploratory tools, enabling us to glean insights into the dataset's characteristics and discern patterns, trends, and sentiments expressed in hotel reviews. Visualisation initiated with a pairplot to examine relationships between numerical variables, including “*Reviewer Score”,* “*Review_Total_Negative_Word_Counts*”, and “*Review_Total_Negative_Word_Counts*”. This allowed us to identify potential patterns and correlations among these variables. Subsequently, distribution plots were employed, comprising histograms and kernel density plots, to illustrate the spread and central tendencies of numerical variables such as *“Reviewer_Score*”, “*Review_Total_Negative_Word_Counts*”, and “*Review_Total_Positive_Word_Counts*”. These plots provided an insightful overview of the data's distribution characteristics.

To gain further insights into “*Reviewer_Score*” distribution, a box plot was generated. This visualisation facilitated the understanding of the distribution's central tendency, dispersion, and the presence of outliers within reviewer scores. Moreover, a correlation matrix was computed to quantify relationships between numerical variables such as “*Reviewer_Score*”, “*Review_Total_Negative_Word_Counts*”, and “*Review_Total_Positive_Word_Counts*”. Visualising this matrix through a heatmap enabled significant correlations among the variables to be pinpointed. In exploring temporal variations, time series trends for “*Reviewer_Score*” and “*Average_Score*” were plotted. These trends shed light on any discernible patterns or fluctuations within reviewer and average scores over the analysed period. Word clouds were plotted to visualise the most frequently occurring words in positive and negative reviews. These visualisations provided qualitative insights into the sentiments expressed, elucidating prevalent themes and sentiments articulated by guests.

3.5. Data Pre-processing

Data pre-processing is a critical step to prepare the dataset for analysis. It involves tasks such as data cleaning, feature selection, and handling missing values.

3.5.1. Data Cleaning

Data cleaning is a crucial step in the data preprocessing pipeline, as it ensures that the data used for analysis is accurate, consistent, and free from errors or inconsistencies. According to Little and Rubin (2002), the potential for biased results and reduced statistical power when missing data is not appropriately handled. Imputation techniques, such as multiple imputation or mean imputation, are recommended to mitigate these issues. It is essential to ensure the dataset's integrity and accuracy, cleaning based on the recommendations of Garcia et al. (2019). This includes handling missing data points, correcting inconsistencies, and standardising data formats.

To remove outliers from the "*Reviewer_Score"* data, the Interquartile Range (*IQR*) method was applied. This technique involved calculating the first quartile (*Q1*) and third quartile (*Q3*) of the data distribution. The interquartile range (*IQR*) was then determined as the difference between *Q3* and *Q1*. Outliers were identified as data points that fell below *Q1 - 1.5 * IQR* or above *Q3 + 1.5 * IQR*. The dataset was filtered to retain only the observations within this range, effectively removing outliers. Finally, the cleaned *"Reviewer_Score"* data was visualised using a box plot, providing a clear representation of the distribution without outliers.

Text-based columns "*Positive_Review*" and "*Negative_Review*" undergo text cleaning, including converting text to lowercase, removing non-alphanumeric characters, stripping leading and trailing spaces, and removing "*stopwords*". These steps help standardise and clean the text data.

3.5.2. Data Transformation

With the cleaned dataset, multiple transformations were applied to shape the data for sentiment analysis. The "*Tags*" variable was converted to integer format. "*Reviewer_Nationality"* was label encoded into integer categories. A key transformation was encoding the text tags into numeric format using scikit-learn's "*LabelEncoder*". This allowed the tags to be used as features. Appropriate data transformations are necessary to convert raw data into formats suitable for mining (Han et al., 2012).

3.5.3. Data Mining

VADER Model for Sentiment Analysis:

In the code, the VADER (Valence Aware Dictionary and Sentiment Reasoner) sentiment analysis tool is employed for sentiment analysis. VADER is a lexicon and rule-based model designed for sentiment analysis tasks, and it assigns sentiment scores to textual data. Specifically, the code utilises the "*SentimentIntensityAnalyzer*" class from the "*vaderSentiment*" library to compute sentiment scores. These scores are calculated for both negative and positive reviews in the dataset.

The sentiment scores obtained from VADER are further processed to assign sentiment labels. If a review's compound sentiment score is less than 0, it is labelled as "*negative*". Conversely, if the score is greater than 0, it is labelled as "*positive*". Scores equal to 0 are designated as "*neutral*." This labelling step categorises reviews based on their overall sentiment polarity.

Additionally, the code determines an overall sentiment label for each review by considering both the negative and positive sentiment scores. If the sum of the negative and positive scores is negative, the review is labelled as "*negative*". If it is positive, it is labelled as "*positive*" Otherwise, it is classified as "*neutral*" This step provides a more comprehensive understanding of the review's sentiment.

To facilitate analysis, the code further transforms these sentiment labels into numerical values. Negative sentiment is encoded as -1, positive sentiment as 1, and neutral sentiment as 0, enabling quantitative analysis.

Random Forest Classifier:

In the subsequent part of the code, a Random Forest Classifier is employed for sentiment prediction. This classifier utilises a selected set of features that are crucial for the predictive modelling task. These features encompass various aspects, including the encoded nationality of the reviewers, reviewer scores, encoded tags, and the sentiment labels assigned during the sentiment analysis process (negative and positive labels).

To evaluate the model's performance, the dataset is divided into training and testing sets using the "*train_test_split*" function from the "*sklearn.model_selection*" library. This division enables model training on one portion of the data and evaluation on another, allowing for a comprehensive assessment of the model's predictive capabilities.

The Random Forest Classifier was chosen due to its ability to handle complex datasets and robustness. To optimise its performance, the code employs hyperparameter tuning through grid search (*GridSearchCV*). This technique searches for the best combination of hyperparameters, including the number of estimators (trees) and the maximum depth of the trees.

Subsequently, the model is evaluated using the testing dataset. The code calculates accuracy, generates a classification report, and constructs a confusion matrix to assess the model's performance in predicting sentiment labels accurately.

Feature Importance Analysis: Feature importance analysis is a critical component of the code's pattern evaluation. After training the Random Forest model, the code assesses the importance of each feature used in the model. Feature importance reveals which input variables had the most considerable influence on the model's predictions. This information is invaluable for understanding the factors that contribute most to sentiment analysis.

Cross-Validation: Cross-validation is an essential step to evaluate the model's stability and generalisation performance. The code employs stratified k-fold cross-validation, where the dataset is divided into multiple subsets or folds. Accuracy scores are computed for each fold, offering insights into how well the model generalises to unseen data. Moreover, the code calculates the mean and standard deviation of these accuracy scores, providing a comprehensive understanding of the model's performance consistency. Cross-validation helps ensure that the model's effectiveness extends beyond the training data, guarding against overfitting and enhancing its reliability.

3.6. Pattern Evaluation

The trained Random Forest model was evaluated using accuracy, confusion matrix, and classification reports. ROC analysis identified the relationship between true and false positives. Feature importance scores provided insights into the most predictive features. Misclassified examples were examined to reveal limitations. Thorough evaluation validates model effectiveness on the sentiment analysis task (Kotsiantis et al., 2006).

This evaluation aims to measure how well the model predicts sentiment labels and understand its behaviour in handling the dataset. Let us break down the pattern evaluation steps and their significance:

Feature Importance Analysis: In the first step of pattern evaluation, the code delves into the importance of each feature utilised by the Random Forest Classifier for sentiment prediction. This analysis aids in discerning which input variables or features hold the most sway over the model's predictions. These "*importance*" values are derived from how much each feature contributes to minimising the model's impurity or error. Features with higher importance values are deemed more informative. This facet of pattern evaluation serves as a critical insight into the aspects of the input data that bear the most relevance for sentiment prediction.

Misclassified Examples (Optional): While labelled as optional, the examination of misclassified examples is exceedingly enlightening. These instances signify when the model's predictions do not align with the actual sentiment labels. Delving into these misclassified cases can uncover patterns or intricacies within the data that pose challenges for the model. It offers valuable insights into areas where the model might necessitate improvements or refinements.

Performance Summary: This section furnishes a comprehensive summary of the model's performance metrics. The key metrics encompass accuracy, the classification report, and the confusion matrix. Accuracy quantifies the proportion of correctly predicted sentiment labels relative to the total predictions. The classification report provides an intricate breakdown of precision, recall, F1-score, and support for each sentiment class, including positive, negative, and neutral. The confusion matrix, in tabular form, offers a granular view of the count of true positives, true negatives, false positives, and false negatives, further elucidating the model's performance.

Plotting Confusion Matrix: The code proceeds to generate a heatmap visualisation of the confusion matrix. Heatmaps serve as a potent means to visually grasp the distribution of the model's predictions when juxtaposed with actual labels. This visualisation lends insights into the model's strengths and weaknesses, particularly regarding the occurrences of false positives and false negatives.

Receiver Operating Characteristic (ROC) Curve: This segment of the code calculates and plots ROC curves and Area Under the Curve (AUC) scores. While ROC curves are traditionally employed in binary classification tasks, AUC quantifies the model's capability to differentiate between positive and negative classes. The ROC curve is plotted for the micro-average, which collectively considers all classes. This evaluation aids in gauging the overall performance of the model, especially in the context of multi-class sentiment prediction.

Cross-Validation: The final component of pattern evaluation encompasses cross-validation, a pivotal step in evaluating the model's capacity for generalisation. The code executes stratified k-fold cross-validation to scrutinise the model's accuracy across diverse subsets of the dataset. Accuracy scores are computed for each fold, furnishing both individual fold scores and statistical insights into the mean and standard deviation of accuracy scores. Cross-validation is instrumental in ascertaining that the model's performance remains consistent across distinct data splits, thereby safeguarding against overfitting.

It helps identify areas for improvement, assess the model's predictive power, and provides valuable insights into its behaviour, strengths, and weaknesses.

3.7. Knowledge Presentation

Data visualisation conveyed insights. Histograms compared sentiment distributions, revealing Polarised negative/positive reviewer scoring patterns. Bar charts showed average sentiment changes over time for top hotels. These visualisations enhance interpretation of sentiment patterns and trends (Few, 2012).

3.8. Software / Tools Employed

The Python programming language was utilised for implementing the data processing, modelling, and analysis workflows. The Pandas library was leveraged for data manipulation tasks, including loading the CSV dataset, handling missing values, date parsing, and transforming variables (McKinney, 2022). Supporting mathematical and array functions were provided by "Numpy" (Harris et al., 2022).

For visualisation, the Matplotlib and Seaborn Python libraries were employed to generate plots such as histograms and bar charts to convey data insights (Hunter, 2007; Waskom, 2022). Scikit-learn, a prominent machine learning toolkit, provided key functionality for text vectorisation, model training, hyperparameter tuning, and evaluation metrics (Pedregosa et al., 2011).

Natural language processing was a core component of this study. The NLTK toolkit enabled text processing and tokenisation (Bird et al., 2009). Sentiment analysis was performed using the VADER model to obtain polarity scores from reviews (Hutto & Gilbert, 2014). Furthermore, scikit-learn's "TfidfVectorizer" transformed the text into weighted numeric representations compatible with machine learning algorithms.

The Random Forest Classifier from scikit-learn was the primary machine learning model utilised for sentiment classification, selected due to its robust performance on textual data. Hyperparameter optimisation was conducted with scikit-learn's GridSearchCV utility. Model evaluation and selection relied on k-fold stratified cross-validation, also provided by scikit-learn.

The experiments were executed using the Google Colab platform, which offered free access to GPU computing resources required for efficient modelling on this large dataset.

Python and its extensive data science ecosystem, especially Pandas, scikit-learn, NLTK and VADER enabled the key analysis and modelling approaches for this hotel review sentiment research built on machine learning and NLP techniques.

3.9. Ethical Preview

The ethical framework guiding the sentiment analysis research project is rooted in principles of integrity, transparency, and accountability. The data was scraped from Booking.com, and all data in the file is publicly available to everyone already. It should be noted that the data is originally owned by Booking.com. The primary objective of the research is to harness the formidable capabilities of sentiment analysis to unlock the latent sentiments buried within these hotel reviews. By analysing this extensive quantity of textual data, a comprehensive comprehension of customer feelings and attitudes concerning European hotels is sought to be attained. The overarching aim of the dissertation is to explore and analyse the sentiments conveyed within a substantial dataset of 515,000 hotel reviews sourced from European destinations.

CHAPTER 4

4. DESIGN

In this design chapter, the methodology for analysing hotel reviews to gain valuable sentiment insights is outlined. The design is structured around six main components, each of which plays a crucial role in the research. Throughout the design, relevant research studies are drawn upon to support the methodology. The key stages of data preprocessing, data mining, evaluation, and knowledge presentation are described using Unified Modeling Language design and in-depth process.

4.1. Unified Modeling Language Design of sentiment analysis

The UML diagram illustrates the main components and their interactions in the sentiment analysis system. Each class encapsulates related attributes and methods for specific functionalities. The relationships between classes depict how data flows through the system for preprocessing, analysis, evaluation, and presentation.

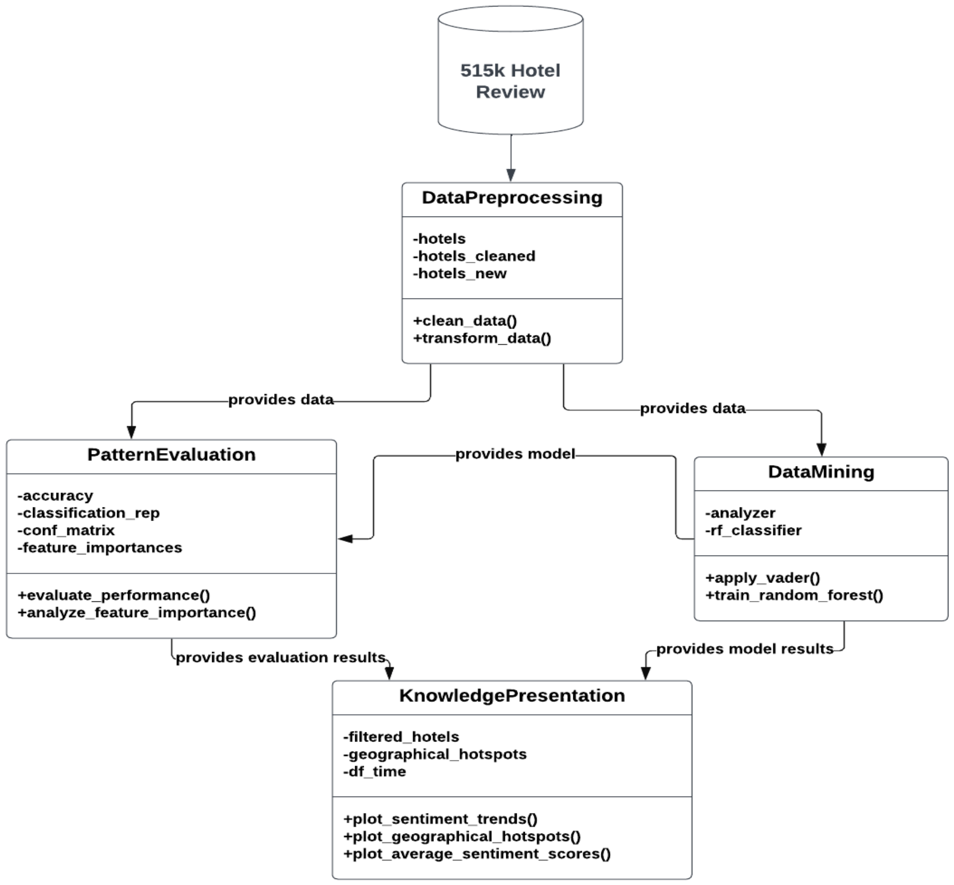


Fig. 4.1. Unified Modeling Language Design of sentiment classification.

4.1.1. Classes

- Data Preprocessing (Data Cleaning, Data Transformation):
  Attributes: *hotels, hotels_cleaned, hotels_new.*
  Methods: *clean_data(), transform_data().*
- Data Mining (VADER Model, Random Forest Classifier):
  Attributes: *analyzer, rf_classifier.*
  Methods: *apply_vader(), train_random_forest().*
- Pattern Evaluation (Performance Evaluation, Feature Importance Analysis):
  Attributes: accuracy, classification_rep, conf_matrix, feature_importances
  Methods: *evaluate_performance(), analyze_feature_importance()*
- Knowledge Presentation (Data Visualisation, Presentation Functions):
  Attributes: *filtered_hotels, geographical_hotspots, df_time*
  Methods: *plot_sentiment_trends(), plot_geographical_hotspots(), plot_average_sentiment_scores()*

4.1.2. Relationships

- **Data Preprocessing** interacts with **Data Mining** and **Pattern Evaluation** to provide clean and transformed data for analysis.
- **Data Mining** utilised pre-processed data to apply the VADER model for sentiment analysis and train a Random Forest classifier.
- **Pattern Evaluation** evaluates the performance of the classifier and analyses feature importance.
- **Knowledge Presentation** utilised the analyses data to create visualisation and present knowledge insights.

4.2. Unified Modeling Language Design of Visualisation

The diagram depicts the flow of control between objects and classes during the execution of the code. Data Pre-processing class preprocesses the dataset by cleaning text variables, removing outliers, and calculating correlations. Data Visualisation class visualises the pre-processed data using various plotting techniques such as pair plots, distribution plots, count plots, box plots, heatmaps, time series trends, scatter plots, bar plots, and word clouds.

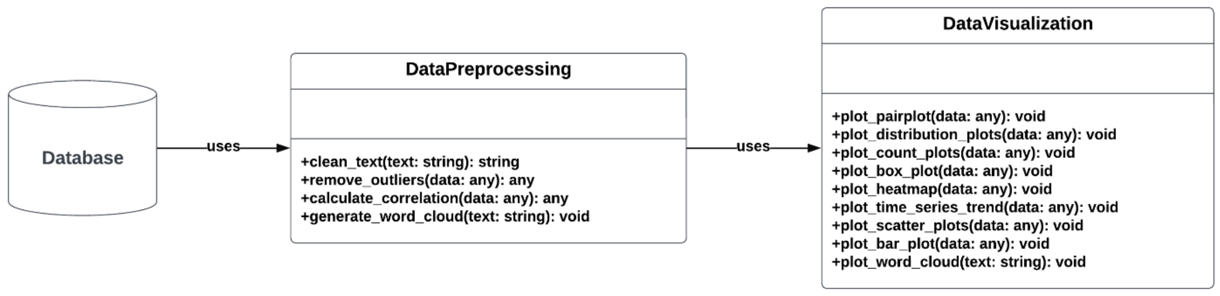

Database
uses
DataPreprocessing
+clean_text(text: string): string
+remove_outliers(data: any): any
+calculate_correlation(data: any): any
+generate_word_cloud(text: string): void
uses
DataVisualization
+plot_pairplot(data: any): void
+plot_distribution_plots(data: any): void
+plot_count_plots(data: any): void
+plot_box_plot(data: any): void
+plot_heatmap(data: any): void
+plot_time_series_trend(data: any): void
+plot_scatter_plots(data: any): void
+plot_bar_plot(data: any): void
+plot_word_cloud(text: string): void

Fig 4.2. Unified Modeling Language design for Daat Visualisation.

4.3. Data Set Structure

The foundation of this research is rooted in the structure of the dataset. The dataset encompasses various attributes, including reviewer demographics, review text, and sentiment scores. Temporal information, reviewer profiles, and geographical coordinates are featured in the dataset, enabling multifaceted analysis. Initially, the raw dataset comprised hotel reviews with columns for text fields such as “*Positive_Review*” and “*Negative_Review*”, metadata like “*Hotel_Name*” and “*Reviewer_Nationality*”, and ratings such as “*Reviewer_Score*” (Han et al., 2012). Distributions and correlations between attributes were determined through exploratory analysis, guiding subsequent preprocessing and feature engineering.

4.4. Processing

Data preprocessing was undertaken, encompassing the cleaning of missing values and the selection of relevant reviews from May-August 2017 for analysis (Garcia et al., 2019). The text tags were encoded numerically, and *“Tags”* was converted to integers. This process formatted the data for machine learning purposes. Exceptional cases where reviews contain “*No Positive*” or “*No Negative*” were handled. Thorough cleaning of textual data was conducted, including lowercasing, removal of special characters, and elimination of stop-words, aligning with similar text preprocessing methods as identified in [Reference 6].

4.5. System Structure

The system demonstrates a well-structured and comprehensively documented workflow, starting with data loading and preprocessing and progressing through sentiment analysis and machine learning, ultimately culminating in knowledge presentation. Each stage of the process is thoughtfully annotated with comments and explanations, ensuring that the code is accessible and easy to comprehend for users. The integration of powerful libraries like scikit-learn for machine learning and “*vaderSentiment”* for sentiment analysis significantly enhances the code’s efficiency and overall effectiveness. Furthermore, the incorporation of visualisation techniques, including bar plots and line graphs, facilitates the visual representation of results, thereby simplifying their interpretation. Altogether, this code exemplifies a meticulous and exhaustive approach to sentiment analysis, encompassing feature selection, model training, and evaluation, thus establishing itself as a robust and informative system tailored for the analysis of hotel reviews.

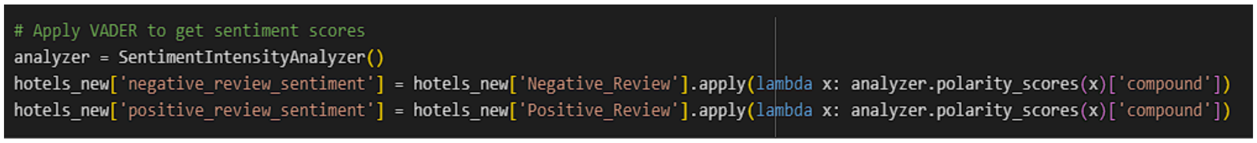
The system design encompasses two primary components:-
Sentiment Analysis - Utilising VADER, sentiment analysis was conducted on both the "Positive_Review" and "Negative_Review" text fields. Compound polarity scores ranging from -1 to 1 were generated by VADER for each review, facilitating the quantification of sentiment intensity.

Classification - A Random Forest model was developed, trained on various features such as nationality, reviewer scores, and encoded tags to predict sentiment labels, namely positive, negative, or neutral. These sentiment labels were derived from the compound polarity scores obtained through VADER sentiment analysis.

The research framework is structured around a well-defined system architecture, utilising a variety of tools and libraries. By employing the VADER sentiment analysis tool, sentiment scores were effectively extracted from the review text. VADER's lexicon-based approach is widely adopted in sentiment analysis due to its proficiency in interpreting nuanced sentiment expressions.

4.6. Sentiment Analysis Analogy

In VADER, a rule-based model is employed, utilising sentiment lexicons to determine polarity and intensity. The compound score aggregates positive and negative sentiment into a normalised metric analogous to sentiment on a -1 to 1 scale. This enables nuanced sentiment quantification. Additionally, an overall sentiment score is introduced by combining positive and negative sentiments. This approach is aligned with previous studies, ensuring compatibility with existing sentiment analysis methodologies.



```
# Apply VADER to get sentiment scores
analyzer = SentimentIntensityAnalyzer()
hotels_new['negative_review_sentiment'] = hotels_new['Negative_Review'].apply(lambda x: analyzer.polarity_scores(x)['compound'])
hotels_new['positive_review_sentiment'] = hotels_new['Positive_Review'].apply(lambda x: analyzer.polarity_scores(x)['compound'])
```

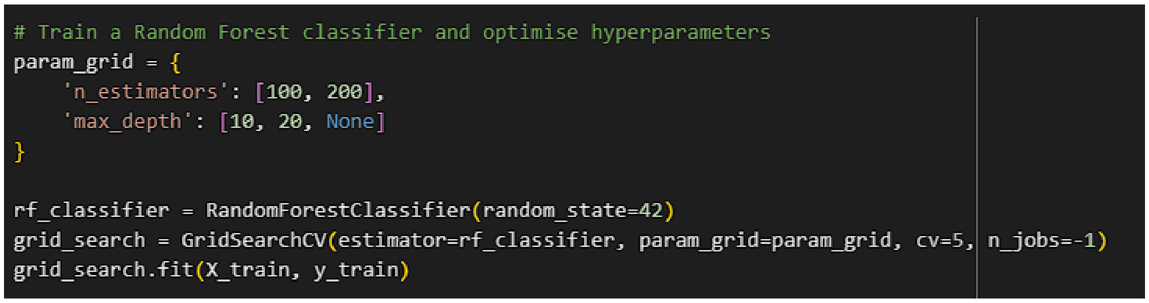
Code Snippet 4.1. VADER sentiment analyzer function

4.7. Classification

A Random Forest classifier was selected for interpretable nonlinear modelling of diverse feature types (Breiman, 2001). Hyperparameter tuning using grid search improved accuracy. Cross-validation assessed generalisability. The classifier identified relationships between attributes like nationality and tags with overall sentiment. Random Forest displayed nonlinear predictive power for sentiment classification with interpretable importance scores.

Machine Learning Model (Random Forest Classifier): The core involves the development of a machine learning model using the Random Forest Classifier. This ensemble-based model is chosen for its effectiveness in handling complex datasets and ability to capture feature importance. The features selected for the model include encoded nationality, reviewer score, encoded tags, and sentiment labels.

Hyperparameter Tuning: To optimise the Random Forest Classifier's performance, hyperparameter tuning is performed using grid search (*GridSearchCV*). Grid search systematically explores different combinations of hyperparameters, such as the number of estimators (trees) and maximum depth of the trees, to find the best configuration. This design decision is crucial for ensuring that the model performs at its best.



```
# Train a Random Forest classifier and optimise hyperparameters
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [10, 20, None]
}

rf_classifier = RandomForestClassifier(random_state=42)
grid_search = GridSearchCV(estimator=rf_classifier, param_grid=param_grid, cv=5, n_jobs=-1)
grid_search.fit(X_train, y_train)
```

Code Snippet 4.2. Random Forest Classifier and hyperparameters.

4.8. System Environment

The code is written in Python and utilises popular libraries such as pandas, numpy, seaborn, matplotlib, nltk, scikit-learn, and vaderSentiment. Executed on Google Colab, a cloud-based platform by Google, it enables users to write and run Python scripts within a browser interface. This facilitates seamless development and execution of Python code, empowering users to analyse data and build machine learning models efficiently.

4.9. Data Dictionary

The tabular data dictionary provides a comprehensive overview of the variables, their descriptions, data types, type of variable and data source, which illustrate detailed information about the content, format, and structure of the data.

Variable Name	Description	Data Type	Variable	Data Source
hotels_new	Data Frame containing hotel reviews	Data Frame	-	Hotel_Reviews.csv
hotels	Dataset containing hotel reviews	Data Frame	Variable	Hotel_Reviews.csv
hotels_cleaned	Cleaned dataset without missing longitude and latitude	Data Frame	Variable	Hotel_Reviews.csv
lat	Latitude of the hotel location	Float	Numerical	Hotel_Reviews.csv
lng	Longitude of the hotel location	Float	Numerical	Hotel_Reviews.csv
Review_Date	Date when the review was made	Date-Time	-	Hotel_Reviews.csv
days_since_review	Number of days since the review was made	Integer	Numerical	Hotel_Reviews.csv
Reviewer_Score	Score given by the reviewer	Float	Numerical	Hotel_Reviews.csv
Negative_Review	Text of the negative review	String	-	Hotel_Reviews.csv
Positive_Review	Text of the positive review	String	-	Hotel_Reviews.csv
Review_Total_Negative_Word_Counts	Total number of words in the negative review text	Integer	Numerical	Hotel_Reviews.csv
Review_Total_Positive_Word_Counts	Total number of words in the positive review text	Integer	Numerical	Hotel_Reviews.csv
Reviewer_Nationality	Nationality of the reviewer	String	Categorical	Hotel_Reviews.csv
Tags	Tags associated with the review	String	Categorical	Hotel_Reviews.csv
Tags_encoded	Encoded representation of tags	Integer	Numerical	Hotel_Reviews.csv
Nationality	Encoded representation of reviewer nationality	Integer	Numerical	Label Encoding
negative_review_sentiment	Sentiment score of the negative review text	Float	Numerical	VADER Sentiment Analysis
positive_review_sentiment	Sentiment score of the positive review text	Float	Numerical	VADER Sentiment Analysis
negative_review_label	Label assigned to the negative review sentiment	String	Categorical	VADER Sentiment Analysis
positive_review_label	Label assigned to the positive review sentiment	String	Categorical	VADER Sentiment Analysis
overall_sentiment	Overall sentiment label assigned to the review	String	Categorical	VADER Sentiment Analysis
comp_sentiment	Compound sentiment score derived from positive and negative sentiment scores	Float	Numerical	Sentiment Analysis
sentiment_score	Sentiment score indicating positive, negative, or neutral sentiment	Integer	Numerical	Sentiment Analysis
negative_label	Encoded label for negative sentiment	Integer	Numerical	Label Encoding
positive_label	Encoded label for positive sentiment	Integer	Numerical	Label Encoding
X	Features used for training the classifier	DataFrame	-	-
y	Target variable used for training the classifier	Series	-	-
X_train	Features of the training set	DataFrame	-	-

Variable Name	Description	Data Type	Variable	Data Source
X_test	Features of the test set	DataFrame	-	-
y_train	Target variable of the training set	Series	-	-
y_test	Target variable of the test set	Series	-	-
best_rf_classifier	Best Random Forest classifier obtained through grid search	RandomForestClassifier	-	-
y_pred	Predicted sentiment labels for the test set	Array	-	-
feature_importances	Importance of each feature in the Random Forest classifier	Array	-	-
sorted_feature_importances	Sorted list of features based on importance	List	-	-
misclassified	Misclassified examples in the test set (optional)	DataFrame	-	-
scores	Cross-validation scores for each fold	Array	-	-
mean_accuracy	Mean accuracy score obtained from cross-validation	Float	Numerical	-
std_accuracy	Standard deviation of accuracy scores from cross-validation	Float	Numerical	-
geographical_neg_hotspots	Mean negative sentiment scores grouped by latitude and longitude	DataFrame	-	-
geographical_pos_hotspots	Mean positive sentiment scores grouped by latitude and longitude	DataFrame	-	-
highest_positive_sentiment	Latitude and longitude of the location with the highest average positive sentiment	Series	-	-
highest_negative_sentiment	Latitude and longitude of the location with the highest average negative sentiment	Series	-	-
test_cases	Random examples selected from the test set for unit testing	DataFrame	-	-
max_date, min_date	Maximum and minimum dates in the dataset	Timestamp	Variable	Derived from 'hotels_cleaned' dataset
hotels_new	Trimmed dataset for the specified time-frame	Pandas DataFrame	Variable	Derived from 'hotels_cleaned' dataset
Q1, Q3	First and third quartiles of 'Reviewer_Score'	Float	Variable	Derived from 'hotels_new' dataset
IQR	Interquartile range of 'Reviewer_Score'	Float	Variable	Calculated from Q1 and Q3
lower_bound, upper_bound	Lower and upper bounds for outlier detection	Float	Variable	Calculated based on quartiles and IQR
stop_words	Set of English stopwords for text cleaning	Set	Variable	NLTK corpus
clean_text()	Function to clean text data	Function	Function	N/A
sns.pairplot(), sns.histplot()	Functions to create pair plots and distribution plots	Function	Function	Seaborn library
plt.subplot(), plt.figure(),	Functions to create various plots	Function	Function	Matplotlib library

Variable Name	Description	Data Type	Variable	Data Source
plt.scatter(), plt.title(), plt.xlabel(), plt.ylabel(), plt.colorbar(), plt.show()				
WordCloud(), generate_word_cloud()	WordCloud object and function to generate word clouds	Object, Function	Function	Wordcloud library

Table 4.1. Data dictionary of system

4.10. Collation of Results

The research concludes with the collation of results aimed at achieving a holistic understanding of customer sentiments. Sentiment distributions are visualised to uncover patterns over time, while reviewer behaviour is explored, taking cues from reviewer nationality and the total number of reviews they have given to assess their impact on sentiment expression. Additionally, the impact of hotel tags on sentiment is investigated to identify aspects of the hotel experience that consistently influence sentiments. The research is concluded by the application of machine learning techniques, including Random Forest, to predict overall sentiment and sentiment associated with specific aspects of the hotel experience. Lastly, an analysis of the impact of review length on sentiments is conducted to determine whether more comprehensive reviews tend to be more positive or negative, drawing inspiration from relevant studies.

Insights into temporal variations, geographical patterns, reviewer behaviours, sentiments associated with hotel tags, and the impact of review length are consolidated. Results are presented using visualisation, classification reports, confusion matrices, and ROC curves to provide a comprehensive understanding of hotel reviews' sentiments.

The design chapter outlines a structured approach to analysing hotel reviews for sentiment insights, with each component informed by relevant research studies. The methodology ensures the robustness, credibility, and actionable insights derived from the analysis of a rich and multifaceted dataset using best practices in sentiment analysis, data preprocessing, and machine learning techniques. This ensures the robustness of the analysis and the reliability of the obtained insights.

CHAPTER 5

5. IMPLEMENTATION AND TESTING

The initiation of sentiment analysis implementation commences with data visualisation, a pivotal step that illuminates crucial aspects and characteristics of the dataset. As per the UML design, data visualisation serves as a foundational element in comprehending the underlying structure and patterns within the dataset, thereby facilitating informed decisions and insights during subsequent stages of analysis.

5.1. Data Visualisation

Pair-plot of Select Numeric Columns: The pair-plot reveals relationships and distributions between numeric variables. Notably, 'Reviewer_Score' shows a moderate positive correlation with 'Review_Total_Positive_Word_Counts,' indicating that longer positive reviews tend to receive higher reviewer scores. Conversely, there is a weaker negative correlation between 'Reviewer_Score' and 'Review_Total_Negative_Word_Counts,' suggesting that longer negative reviews are associated with slightly lower scores.

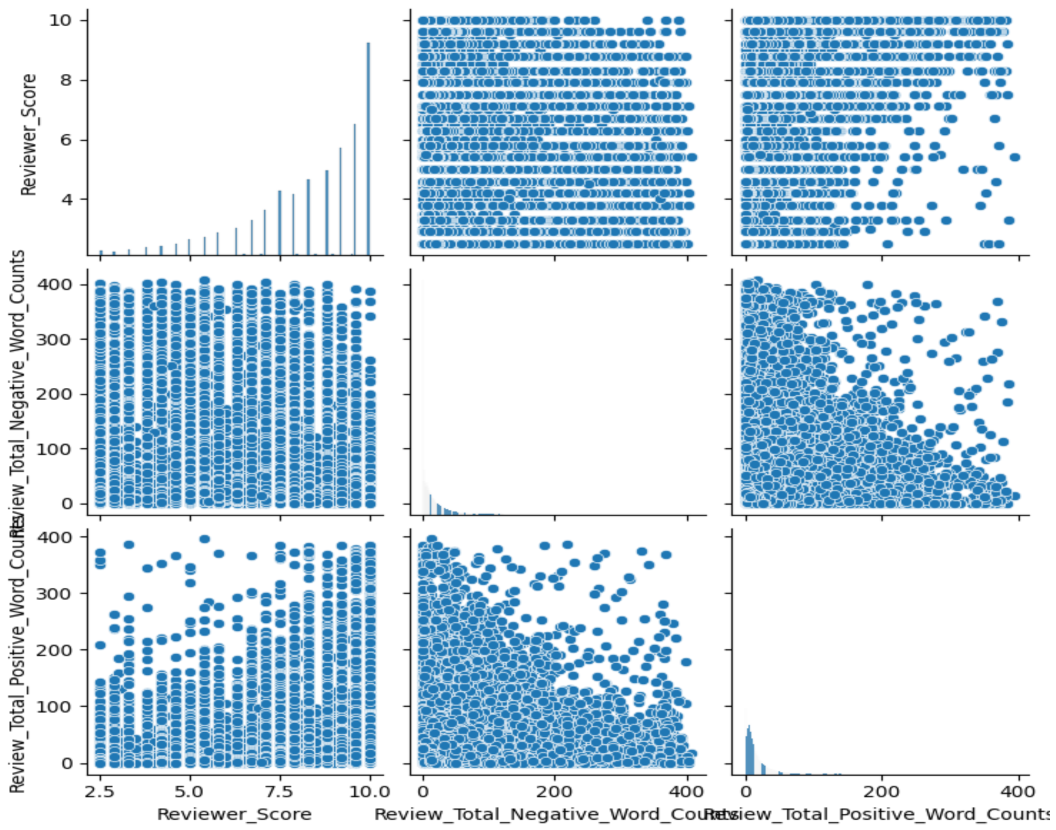


Fig. 5.1. Pair-plot for numeric columns.

Distribution Plots (Histograms and KDEs): These plots offer a closer look at the distributions of key variables. The 'Reviewer_Score' histogram shows a peak around 8 to 9, indicating that many reviewers tend to give positive scores. On the other hand, the word count distributions for both negative and positive reviews are right-skewed, with most reviews being relatively concise. This suggests that brevity is common in hotel reviews.

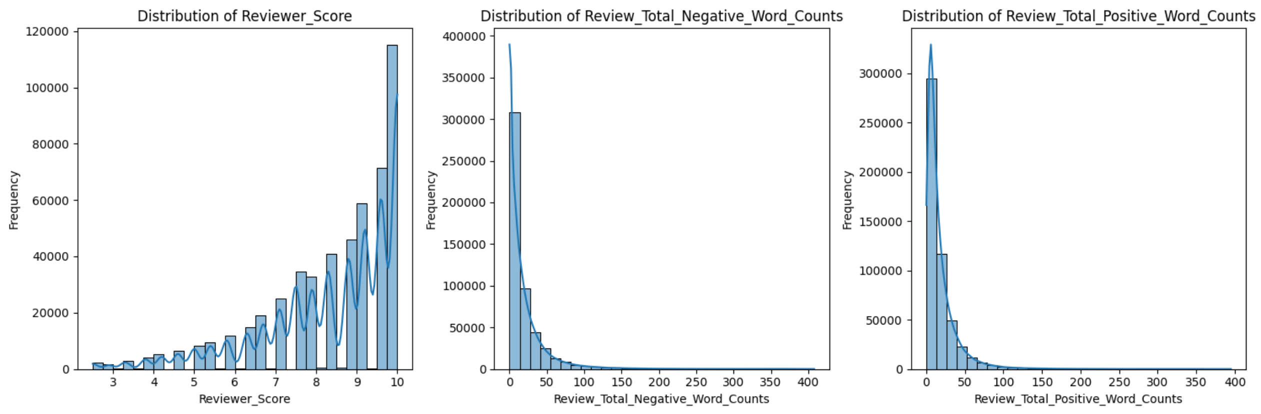


Fig. 5.2. Kernel Density Plot.

Box Plot of Reviewer Score: The box plot summarises the distribution of reviewer scores, providing a clear view of central tendencies and potential outliers. It indicates that the median reviewer score falls between 8 and 9, with potential outliers on both ends. This suggests that most reviewers provide positive scores, but there are exceptions.

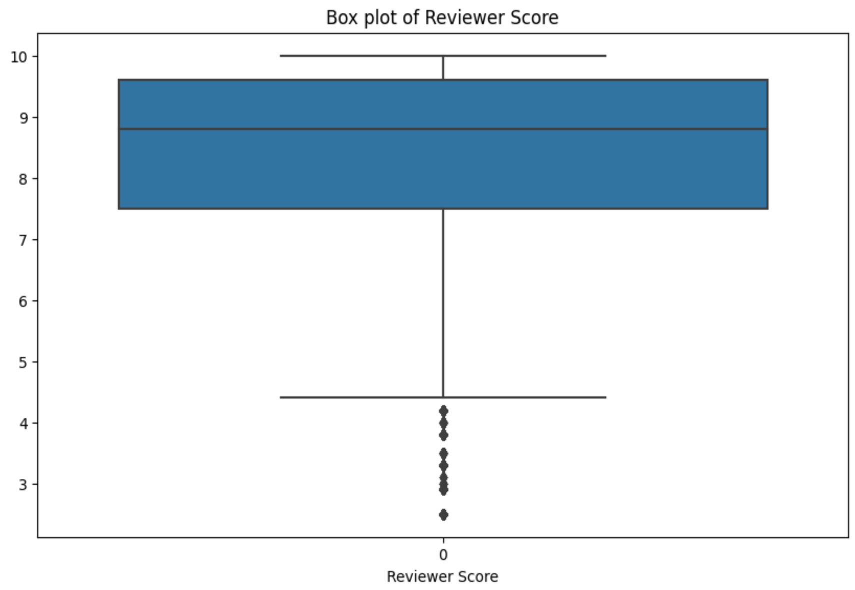


Fig. 5.3. Box-plot for reviewer score.

Correlation Matrix Heatmap: The heatmap visualises correlations between numeric variables. Notably, ‘Reviewer_Score’ exhibits a positive correlation with ‘Average_Score,’ suggesting that reviewers’ individual scores align with the hotels’ average scores.

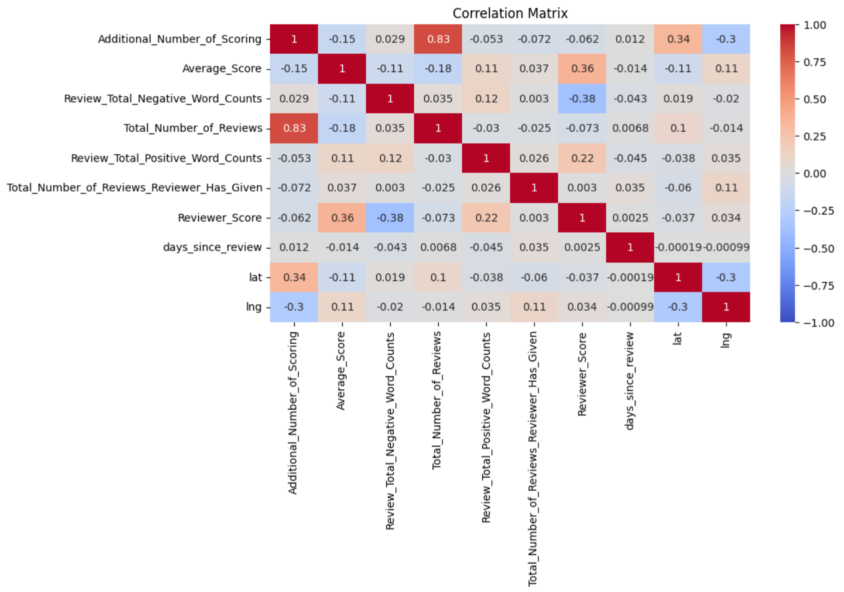


Fig. 5.4. Heat map for correlation matrix between numeric variables.

Time Series Trend of Reviewer Score and Average Score: The time series trends of ‘Reviewer_Score’ and ‘Average_Score’ over time reveal patterns and fluctuations. Analysing these trends can provide insights into how reviewer scores and average hotel scores have evolved over time.

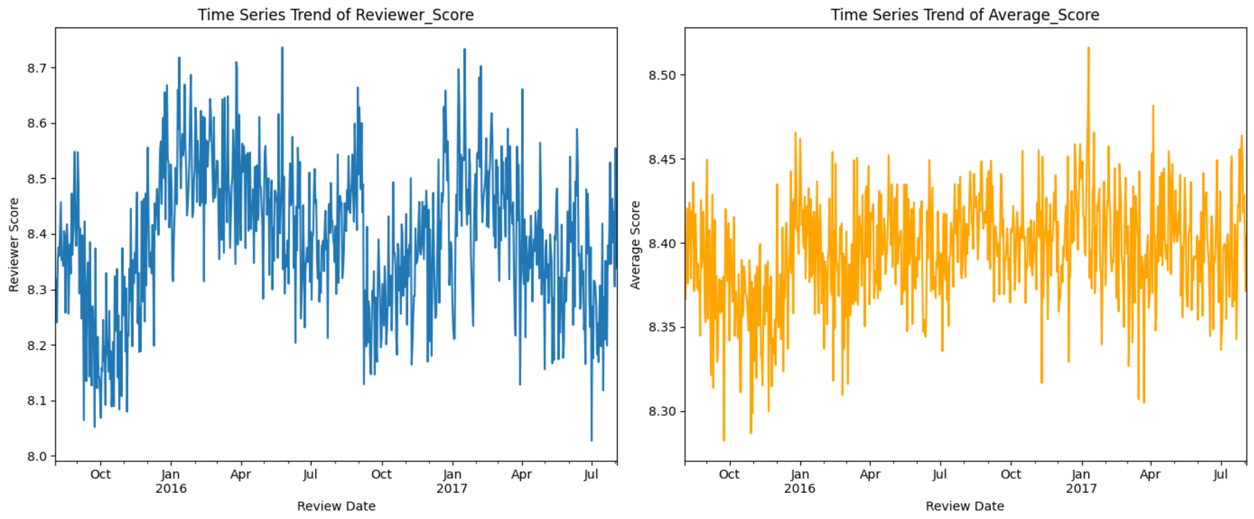


Fig. 5.5. Time trend series.

Word Clouds for Positive and Negative Reviews: The word clouds visually represent frequently occurring words in positive and negative reviews. They offer an intuitive way to identify common sentiments and themes in hotel reviews, with the “Positive Reviews Word Cloud” highlighting positive sentiments and the “Negative Reviews Word Cloud” highlighting negative sentiments.

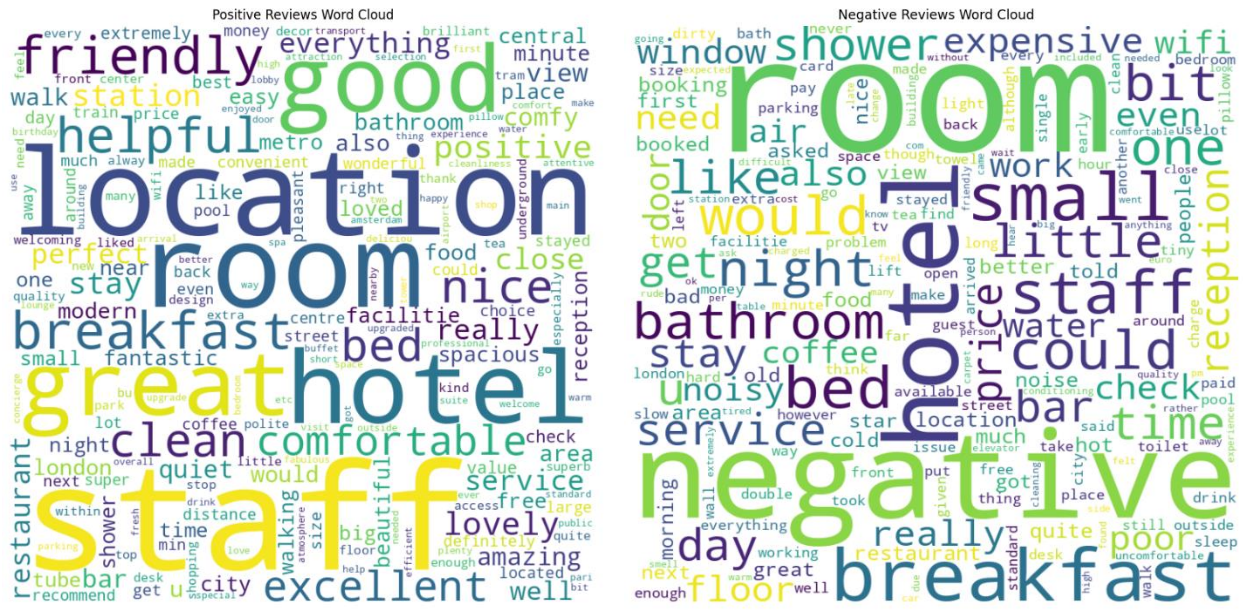


Fig. 5.6. Word Cloud for Positive and Negative Words.

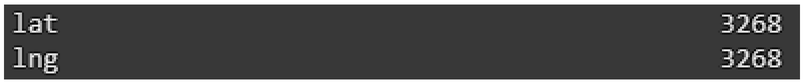
5.2. Analysis Initiation

The initial steps of this data analysis project involve loading the dataset from a CSV file, a foundational and critical step in any data analysis endeavour. Following data loading, the code conducts an essential data quality check by examining and reporting missing data within the dataset using the “*isnull().sum()*” function. This meticulous missing data inspection is vital for understanding the overall data quality and integrity.

```
#Checking for missing data
print("Number of missing values before cleaning:\n", hotels.isnull().sum())
print(hotels.shape)
```

Code Snippet 5.1. Check for missing data.

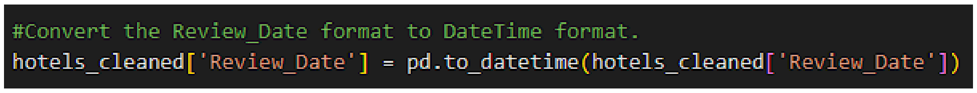
Moving forward, the code proceeds with data cleaning procedures. Specifically, it makes the pragmatic decision to drop rows that contain missing values in the *“lat”* and *“lng”* columns. This choice is well-founded because these geographic coordinates are likely to play a pivotal role in subsequent geographical insights. Eliminating rows with missing values in these columns ensures that geographical analysis can be conducted effectively.



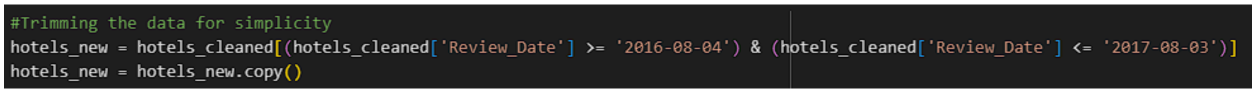
```
lat                                          3268
lng                                          3268
```

Fig. 5.7. Missing Data of *'lat'* & *'lng'*.

Data transformation is another integral aspect of this data exploration process. The code skilfully transforms the *"Review_Date"* column into a datetime format from integer type. This transformation is instrumental in enabling time-based analysis, a feature that can provide valuable insights into how sentiments evolve over time. Furthermore, the code narrows down the dataset by selecting a specific time-frame (*"Review_Date"* between '2015-08-04' and '2017-08-03'). This temporal selection enhances the focus of the analysis.



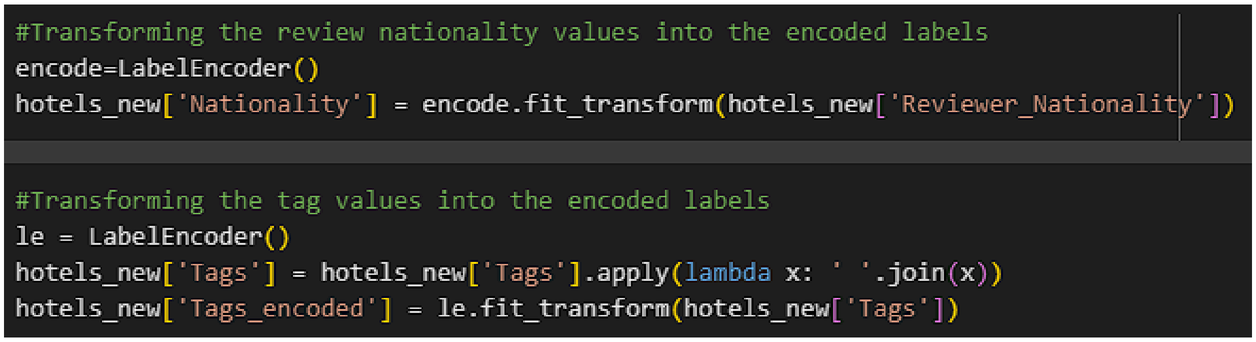
```
#Convert the Review_Date format to DateTime format.
hotels_cleaned['Review_Date'] = pd.to_datetime(hotels_cleaned['Review_Date'])
```



```
#Trimming the data for simplicity
hotels_new = hotels_cleaned[(hotels_cleaned['Review_Date'] >= '2016-08-04') & (hotels_cleaned['Review_Date'] <= '2017-08-03')]
hotels_new = hotels_new.copy()
```

Code snippet 5.2. date type conversion and trimming the data to reduce time complexity.

To facilitate machine learning modelling, certain columns are converted to integer types. Particularly, "*Tags*" and *'Nationality'* columns are transformed into numeric formats. This conversion is pivotal for machine learning algorithms that require numerical input features and variables.



```
#Transforming the review nationality values into the encoded labels
encode=LabelEncoder()
hotels_new['Nationality'] = encode.fit_transform(hotels_new['Reviewer_Nationality'])

#Transforming the tag values into the encoded labels
le = LabelEncoder()
hotels_new['Tags'] = hotels_new['Tags'].apply(lambda x: ' '.join(x))
hotels_new['Tags_encoded'] = le.fit_transform(hotels_new['Tags'])
```

Code snippet 5.3. Data transformation.

Feature engineering is a pivotal stage in this data exploration process. It entails encoding categorical variables such as *'Nationality'* and *'Tags'* using label encoding techniques. This categorical feature encoding is a prerequisite for incorporating these variables into machine learning models. Additionally, the code harnesses the power of the VADER sentiment analysis tool to compute sentiment scores for both negative and positive reviews. These sentiment scores serve as the foundation for generating sentiment labels, overall sentiment categorisations, and compound sentiment scores. Furthermore, the creation of the "*sentiment_score"* feature provides a numerical representation of sentiment, classifying it as negative (-1), positive (1), or neutral (0).

In the realm of machine learning, a Random Forest Classifier is thoughtfully chosen as the model for sentiment analysis. Random Forests are well-suited for classification tasks and are known for their robust performance. This decision is substantiated by its aptitude for handling sentiment classification. The model's hyperparameters are fine-tuned through a grid search process to identify the optimal combination, thereby ensuring the model's effectiveness.

Performance evaluation is an indispensable step in this data exploration journey. The code rigorously assesses the Random Forest Classifier's performance by computing critical metrics such as accuracy, classification reports, and confusion matrices. These metrics provide comprehensive insights into how proficiently the model classifies sentiments. Additionally, the code charts Receiver Operating Characteristic (ROC) curves and computes areas under the ROC curves (AUC) for each class. ROC curves are pivotal for gauging the classifier's performance in both binary and multiclass classification tasks.

Feature importance analysis is an enlightening component of the exploration process. It involves scrutinising which features wield the most substantial influence over the model's predictions. The code diligently presents the top "*feature_importances"*, providing valuable insights into the factors that predominantly-shape sentiment predictions.

Ensuring the model's robustness and generalisation capability is paramount. To this end, the code employs stratified k-fold cross-validation. This cross-validation technique assesses the model's consistency and reliability across various subsets of the dataset, enhancing its overall trustworthiness.

Data visualisation is a key facet of this exploration. The code employs a range of visualisation techniques to convey insights effectively. Histograms are employed to visualise reviewer scores across different sentiment categories, such as positive, negative, and neutral. Bar plots skilfully depict the average sentiment scores for top hotels with the highest negative and positive sentiment ratings, offering valuable insights into sentiment trends for specific hotels. Line plots are instrumental in illustrating sentiment trends over time, enabling an exploration of how sentiments fluctuate across different quarters.

Finally, geographical analysis adds a unique dimension to this exploration. By grouping data based on latitude and longitude and calculating average negative sentiment scores, the code identifies geographical hotspots of negative sentiment. These hotspots are thoughtfully plotted on a geographical map, providing a spatial context for sentiment analysis results.

In summation, this structural experimentation and data exploration process exhibit a comprehensive and meticulously structured approach. It encompasses data loading, cleaning, transformation, feature engineering, machine learning, performance evaluation, feature analysis, cross-validation, visualisation, and geographical analysis, presenting a holistic and insightful exploration of hotel review sentiment data.

5.3. Machine Learning

Machine learning is applied to perform sentiment analysis on hotel reviews. The model is carefully tuned for optimal performance, trained on selected features, and evaluated using various metrics. Feature importance analysis and cross-validation are essential steps to ensure the model's reliability and effectiveness in classifying sentiment in hotel reviews. The primary goal is to classify each review into one of three sentiment categories: positive, negative, or neutral. Here's how machine learning is implemented and utilised:

5.3.1. Model Selection: The chosen machine learning model for sentiment analysis is the Random Forest Classifier. Random Forests are an ensemble learning method known for their robustness in classification tasks. They work by combining multiple decision tree classifiers to make predictions.

5.3.2. Feature Selection: Features are selected for the machine learning model. The selected features include:

- *"Nationality"*: Encoded nationality of the reviewer.
- *"Reviewer_Score"*: The numerical score given by the reviewer.
- *"negative_label"*: Encoded label for negative sentiment.
- *"positive_label"*: Encoded label for positive sentiment.
- *"Tags_encoded"*: Encoded labels for hotel tags.

5.3.3. Data Splitting: The dataset is split into training and testing sets using the "train_test_split" function from scikit-learn. This division allows for model training on one portion of the data (70% of the split data) and evaluation on another portion (30% of the split data) to assess performance."

5.3.4. Hyperparameter Tuning: Hyperparameters of the Random Forest Classifier are optimised using Grid Search (*GridSearchCV*). This technique systematically tests different combinations of hyperparameters to find the combination that yields the best model performance. The hyperparameters tuned in this code include:

- "*n_estimators"*: The number of trees in the forest, the classifier uses 100 and 200 values to find the best fit.
- "max_depth": The maximum depth of the trees in the forest are parameterised as 10, 20, or None.

5.3.5. Model Training: The Random Forest Classifier is trained on the training set "*X_train*" using the best hyperparameters identified during the grid search.

5.3.6. Model Evaluation: The model's performance is evaluated on the testing set using several metrics, including:

- **Accuracy**: The proportion of correctly classified reviews.
- **Classification Report**: Provides precision, recall, F1-score, and support for each sentiment class.
- **Confusion Matrix**: Visualises the number of true positives, true negatives, false positives, and false negatives.
- **Receiver Operating Characteristic (ROC) Curve and AUC**: These metrics are computed and plotted to assess the model's performance for each class and overall.

5.3.7. Feature Importance Analysis: The code analyses feature importance in the Random Forest model. The *"feature_importances_"* attribute of the trained model is examined to understand which features have the most considerable influence on the model's predictions. This analysis aids in identifying the factors driving sentiment predictions.

5.3.8. Cross-Validation: To ensure the model's generalisation performance is reliable, stratified k-fold cross-validation is applied. The code uses the *"StratifiedKFold"* class to perform cross-validation with five folds. Accuracy scores are calculated for each fold, and the mean and standard deviation of these scores are reported.

5.4. Tools

The system employs a comprehensive set of tools and libraries for implementing sentiment analysis, machine learning, data visualisation, and data preprocessing. These tools enable a thorough exploration of hotel review data, the development of a sentiment analysis model, and the evaluation of its performance.

Python's extensive data science libraries enabled rapid prototyping and development (Virtanen et al., 2020). Tools like NLTK, scikit-learn, Pandas and Matplotlib provided core functionality for text processing, modelling, data manipulation and visualisation, respectively.

The code begins by importing various Python libraries and frameworks that are essential for the implementation and testing of the sentiment analysis model and data exploration tasks. These libraries include:

- *vaderSentiment*: This library is used for sentiment analysis using the VADER (Valence Aware Dictionary and sEntiment Reasoner) sentiment analysis tool.
- *nltk*: The Natural Language Toolkit is employed for text preprocessing tasks, such as tokenisation and *"stopword"* removal.
- *numpy* and statistics: These libraries are used for numerical operations and statistical calculations.
- scipy.*stats*: Part of SciPy, it provides statistical functions for hypothesis testing and data analysis.
- pandas: Pandas is a powerful library for data manipulation and analysis, particularly for working with tabular data.
- *seaborn* and *matplotlib.pyplot*: These visualisation libraries are used to create various plots and charts to present data insights visually.
- *PIL*: The Python Imaging Library is used for image-related operations, although its purpose in this context is not entirely clear.
- *sklearn*: Scikit-learn is a machine learning library used for tasks like model selection, training, and evaluation.
- *TfidfVectorizer* and *TfidfTransformer*: These are used for text feature extraction and transformation.
- *LabelEncoder*: It is used to encode categorical labels into numerical values.

5.5. Code Implementation

The data preprocessing phase initiates with the conversion of certain variables to integer type, ensuring uniform data representation for computational tasks. Specifically, the 'days_since_review' column undergoes transformation to integer format, enhancing numerical consistency within the dataset.

Subsequently, the *"'Reviewer_Nationality"* column encoded into numerical values using *scikit-learn's* LabelEncoder. This transformation is done to convert categorical data into a format that machine learning algorithms can work with. Then *"Tags"* column is processed by joining the individual tags in each row into a single string. Then, the LabelEncoder is used to convert these strings into numerical labels. This encoding allows the algorithm to understand and work with these categorical tags.

VADER (Valence Aware Dictionary and sEntiment Reasoner) sentiment analysis is applied to the *"Negative_Review"* and *"Positive_Review"* columns using the *"SentimentIntensityAnalyzer"* from the vaderSentiment library. Sentiment scores for each review text are calculated, including positive, negative, and compound scores. Subsequently, VADER scores are converted to sentiment labels for the *"negative_review_label"* and *"positive_review_label"* columns. Scores less than 0 are labelled as '*negative*,' those equal to 0 as '*neutral*,' and those greater than 0 as *'positive.'* An *"overall_sentiment"* function is defined to determine the overall sentiment label for each review based on the sum of negative and positive sentiment scores. If the sum is less than 0, it is labelled as '*negative*,' if greater than 0, it is labelled as '*positive*,' and if equal to 0, it is labelled as 'neutral.' The function is applied to each row of the Data Frame to create the "*overall_sentiment*" column. The compound sentiment score is derived by adding the negative and positive sentiment scores together. This compound score represents an overall sentiment score for each review. Subsequently, the LabelEncoder is used to encode the "*negative_review_label*" and "*positive_review_label*" columns into numerical values, preparing these labels for machine learning models.

The features list contains the columns that will be used as input features for the machine learning model, and "*X*" is created as a data frame containing these features. "*y*" represents the target variable, which is the *"overall_sentiment"* column. splits the data into training and testing sets using "*scikit-learn's" "train_test_split"* function. It assigns 70% of the data to the training set (*"X_train"* and *"y_train"*) and 30% to the testing set (*"X_test"* and *"y_test"*). The *"random_state"* parameter ensures reproducibility. a Random Forest classifier is trained using "*scikit-learn's" "RandomForestClassifier"*. Hyperparameter optimisation is performed using grid search

("*GridSearchCV*") to find the best combination of hyperparameters from the specified parameter grid. This helps improve the model's performance. the code analyses *"feature_importances"* from the trained Random Forest model, examines misclassified examples (optional), and summarises the model's performance. It prints accuracy, the classification report (which includes precision, recall, F1-score, and support), and the confusion matrix. "*feature_importances"* are also displayed to understand which features had the most influence on the model's predictions.

5.6. Testing

The testing procedures are critical for assessing the reliability and robustness of the machine learning model. They help ensure that the model's performance metrics are not overly optimistic or pessimistic due to the specific data split or threshold choice.

5.6.1. Model Evaluation

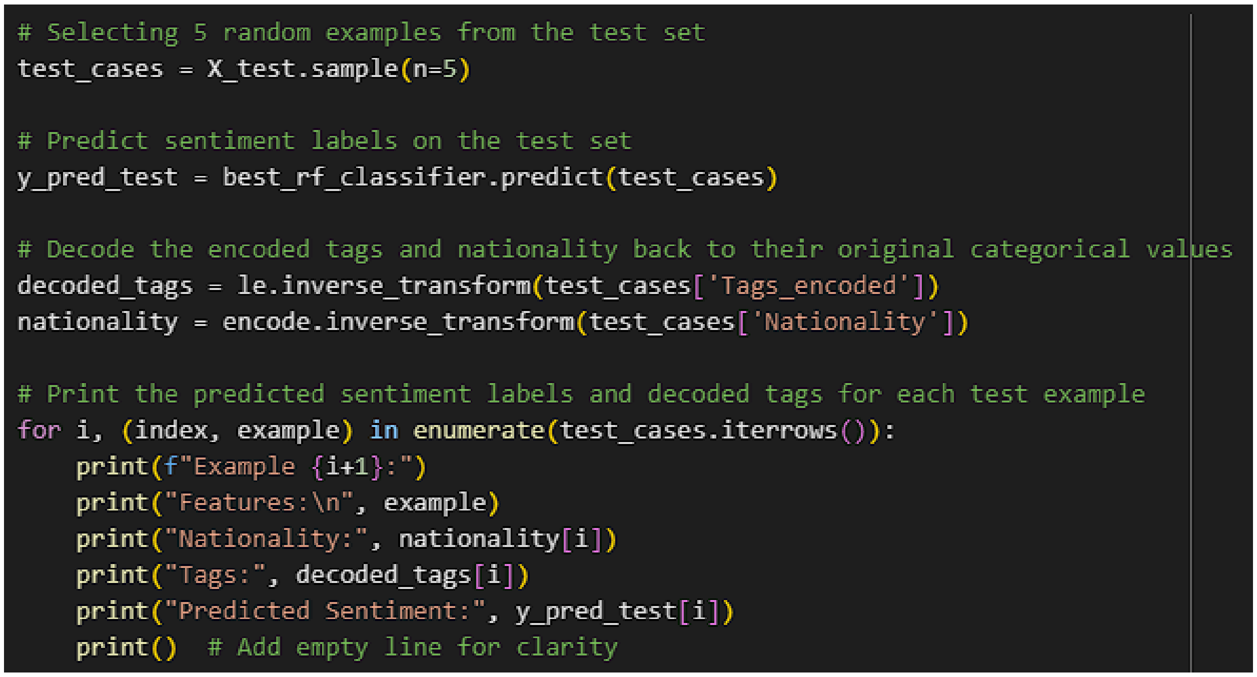
- **Receiver Operating Characteristic (ROC) Curve:**
  - The ROC curve plots the true positive rate (TPR) against the false positive rate (FPR) across different classification thresholds.
  - It assesses the trade-off between true positives and false positives for a classifier.
  - AUC computes the area under this curve, summarising into one metric. Higher AUC indicates better classification ability.
  - Here, ROC AUC is calculated for each sentiment class using one-vs-rest binning.
  - The micro average aggregating across classes is also calculated to evaluate multi-class performance.
  - The ROC analysis provides a visual and quantitative evaluation of the model's ability to discriminate between positive and negative sentiment classes.
- **Cross-Validation:**
  - 5-fold stratified cross-validation is used to assess model performance.
  - The data is split into 5 folds, maintaining equal class distribution in each fold.
  - The model is trained on 4 folds and tested on the held-out fold, repeating for each fold.
  - This provides 5 accuracy scores which are aggregated to get overall performance.
  - Stratification handles class imbalance and 5-fold reduces variance compared to a single train-test split.

- The mean accuracy and standard deviation indicate the model's overall performance and its stability across folds.
- Cross-validation provides a more robust estimate of the model's ability to generalise to new data.

ROC analysis and cross-validation evaluate the sentiment classification performance from different perspectives - discrimination ability and generalisation ability. Together they provide a comprehensive assessment of the model's testing metrics on the hotel review dataset.

5.6.2. Unit Testing

A systematic approach was constructed to evaluate the performance and reliability of sentiment analysis predictions.



```python
# Selecting 5 random examples from the test set
test_cases = X_test.sample(n=5)

# Predict sentiment labels on the test set
y_pred_test = best_rf_classifier.predict(test_cases)

# Decode the encoded tags and nationality back to their original categorical values
decoded_tags = le.inverse_transform(test_cases['Tags_encoded'])
nationality = encode.inverse_transform(test_cases['Nationality'])

# Print the predicted sentiment labels and decoded tags for each test example
for i, (index, example) in enumerate(test_cases.iterrows()):
    print(f"Example {i+1}:")
    print("Features:\n", example)
    print("Nationality:", nationality[i])
    print("Tags:", decoded_tags[i])
    print("Predicted Sentiment:", y_pred_test[i])
    print()  # Add empty line for clarity
```

Code snippet 5.4. Unit testing using test samples.

Here is a breakdown of the unit testing procedure:

- **Selection of Test Cases**:
  Five random examples are selected from the test set "*X_test*" using the "*sample()*" method. These examples serve as representative instances for evaluating the sentiment analysis model.
- **Prediction of Sentiment Labels**:

The sentiment labels are predicted for the selected test cases *“test_cases”* using the trained random forest classifier *“best_rf_classifier”*.

- **Decoding Categorical Variables**:
  The encoded tags and nationality information are decoded back to their original categorical values using inverse transformation methods *“inverse_transform()”*. This step ensures that the predictions and data representation remain interpretable.

- **Printing Predictions and Features**:
  For each test example, the script prints the following details:

  - Features: Displays the features of the test case.
  - Nationality: Reflects the original nationality associated with the test case.
  - Tags: Represents the original tags associated with the test case.
  - Predicted Sentiment: Shows the sentiment label predicted by the model for the corresponding test case.

The unit testing methodology outlined above facilitates comprehensive validation of the sentiment analysis model's functionality, ensuring its accuracy and robustness across diverse data samples.

CHAPTER 6

6. RESULTS

The results chapter presents a comprehensive evaluation of the sentiment analysis model deployed for hotel reviews. The evaluation encompasses several key metrics, including accuracy, precision, recall, and F1-score, providing insights into the model's performance across different sentiment categories. Feature importance analysis sheds light on the influential factors driving sentiment predictions, while the confusion matrix offers a detailed overview of the model's classification accuracy. Additionally, cross-validation results underscore the model's robustness and generalisation capabilities. Knowledge presentation unveils sentiment distribution patterns, sentiment trends over time, and geographical hotspots of negative sentiment, providing actionable insights for hotel management. Unit testing confirms the model's accuracy and reliability, paving the way for real-world applications. The discussion section outlines the model's strengths, areas for improvement, and limitations, offering valuable insights for future research and model refinement.

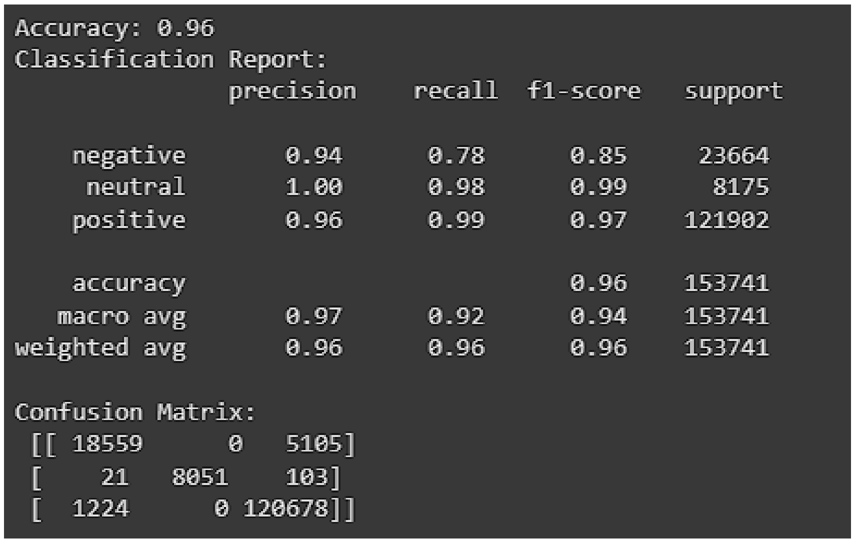
Experiment	Description
Evaluation of Model	Comprehensive assessment of the sentiment analysis model's performance, including accuracy, precision, recall, and F1-score.
Feature Importance	Analysis of influential factors driving sentiment predictions, highlighting key features such as sentiment labels, reviewer's score, and hotel tags.
Confusion Matrix	Detailed overview of the model's classification accuracy, including correct identifications and misclassifications across sentiment categories.
Cross-Validation Results	Examination of the model's robustness and generalisation capabilities through cross-validation with five folds, demonstrating high accuracy and minimal variance.
Knowledge Presentation	Presentation of sentiment distribution patterns, sentiment trends over time, and geographical hotspots of negative sentiment, providing actionable insights for hotel management. This part of experimentation visualises the objectives achieved from the model.
Unit Testing	Assessment of the sentiment analysis model's accuracy and reliability across diverse samples, highlighting its effectiveness in predicting sentiment labels for hotel reviews.

Table 6.1. Experimentations on the model.

6.1. Evaluation of Model

6.1.1. Accuracy and Classification Report

The machine learning model used for sentiment analysis of hotel reviews has demonstrated an impressive overall accuracy of 96%. In terms of precision, it achieved 94% precision for negative sentiment, 100% for neutral sentiment, and 96% for positive sentiment, indicating strong precision across all sentiment categories.



```
Accuracy: 0.96
Classification Report:
              precision    recall  f1-score   support

    negative       0.94      0.78      0.85     23664
     neutral       1.00      0.98      0.99      8175
    positive       0.96      0.99      0.97    121902

    accuracy                           0.96    153741
   macro avg       0.97      0.92      0.94    153741
weighted avg       0.96      0.96      0.96    153741

Confusion Matrix:
 [[ 18559      0   5105]
 [    21   8051    103]
 [  1224      0 120678]]
```

Fig. 6.1. Accuracy, Classification Report, and Top Feature Importance of model.

6.1.2. Precision, Recall, and F1-Score

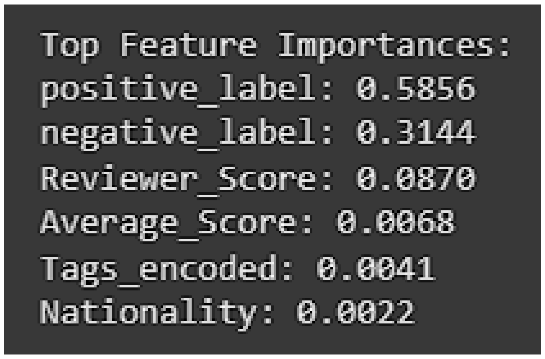
Negative Class: The model achieved a precision of 93% for the negative class. This means that when the model predicts a review as negative, it is correct 93% of the time. However, the recall (sensitivity) for the negative class is lower at 79%, indicating that the model occasionally misses some negative reviews. The F1-score, which balances precision and recall, is 0.85 for the negative class.

Neutral Class: The model performs exceptionally well for the neutral class, with a precision of 100% and a recall of 98%. This indicates that when the model predicts a review as neutral, it is almost always correct and captures the majority of neutral reviews. The F1-score for the neutral class is an impressive 0.99.

Positive Class: For the positive class, the model achieved a precision of 96% and a recall of 99%. This suggests that the model is adept at identifying positive reviews, with a minimal number of false positives and false negatives. The F1-score for the positive class is a solid 0.97.

6.1.3. Feature Importance

Feature importance analysis indicates that the most influential feature in predicting sentiment is whether the review is classified as positive or negative. It highlights that the classification of reviews as positive or negative, as determined by the VADER sentiment analysis tool, emerges as the most influential feature, with a significant importance measure of 0.59. This underscores the pivotal role of sentiment labels generated by VADER in guiding the model's predictions. The reviewer's score emerges as the second most important feature, followed by the encoded tags associated with the hotel. However, the impact of nationality on sentiment prediction appears to be relatively lower. Hence, it can be inferred that *“Nationality”, ”Tags”, ”Reviewer_Score”*, and *“Average_Score”* exert less influence on sentiment prediction compared to the classification of reviews as positive or negative.



```
Top Feature Importances:
positive_label: 0.5856
negative_label: 0.3144
Reviewer_Score: 0.0870
Average_Score: 0.0068
Tags_encoded: 0.0041
Nationality: 0.0022
```

Fig 6.2. Feature importance.

6.1.4. Confusion Matrix

The confusion matrix provides additional insights into the model’s performance. It reveals that the model has correctly identified a substantial number of negative and positive reviews. In a confusion matrix, the rows represent the actual classes, while the columns represent the predicted classes. Each cell in the matrix indicates the number of instances that were classified according to the combination of actual and predicted classes. Class 1 has a total of 18,559 instances. Out of these, 18,559 instances were correctly classified as Class 1 (True Positives). There are no instances misclassified as Class 2 or Class 3. Moving to class 2, a total of 8,175 instances. Among these, 8,051 instances were correctly classified as Class 2 (True Positives). There are 21 instances misclassified as Class 1 and 103 instances misclassified as Class 3. has a total of 121,902 instances. Out of these, 120,678 instances were correctly classified as Class 3 (True Positives). There are 1,224 instances misclassified as Class 1 and no instances misclassified as Class 2. Overall, the model's ability to distinguish between these classes is commendable.

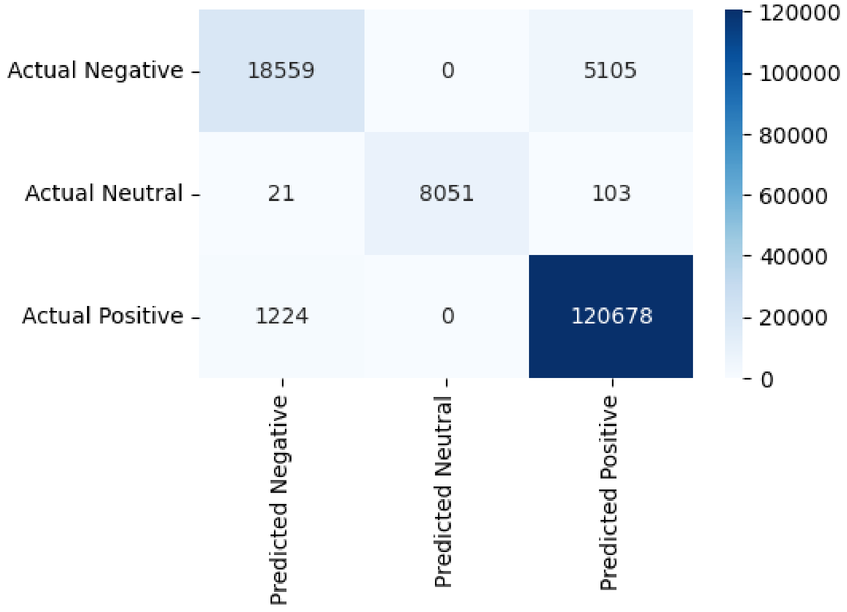
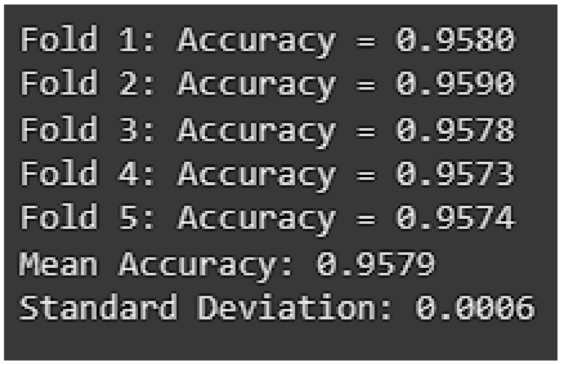


Fig. 6.2. Confusion Matrix.

6.1.5. Cross-Validation Results

Cross-validation, performed with five folds, consistently demonstrates high accuracy, with an average accuracy of approximately 95.79%. This reaffirms the model's robustness and its ability to generalise well to unseen data. The low standard deviation of 0.0006 indicates minimal variance in performance across folds.



```
Fold 1: Accuracy = 0.9580
Fold 2: Accuracy = 0.9590
Fold 3: Accuracy = 0.9578
Fold 4: Accuracy = 0.9573
Fold 5: Accuracy = 0.9574
Mean Accuracy: 0.9579
Standard Deviation: 0.0006
```

Fig. 6.3. Cross-Validation's each fold accuracy, Mean Accuracy, and Standard Deviation.

6.2. Knowledge Presentation

6.2.1. Sentiment Distribution by Reviewer Score

Positive Sentiment: The histogram and kernel density estimate (KDE) plot for positive sentiment reviews indicate that reviewers tend to give higher scores, with a peak around 10. This suggests that guests who leave positive reviews generally rate their experiences very positively.

Negative Sentiment: In contrast, the histogram for negative sentiment reviews shows a peak around lower scores, particularly around 4 to 6 and 6 to 8. This situation presents a challenge in interpreting and addressing customer feedback accurately. While the average rating might not immediately indicate a severe problem, the negative sentiment expressed in the review highlights areas where the business may need improvement.

Neutral Sentiment: The distribution of neutral sentiment reviews appears relatively flat, suggesting that reviewers with neutral sentiments distribute their scores across a broader range. This distribution implies that the presence of neutral sentiment in reviews may affect the model's ability to predict the correct intensity of consumer sentiment.

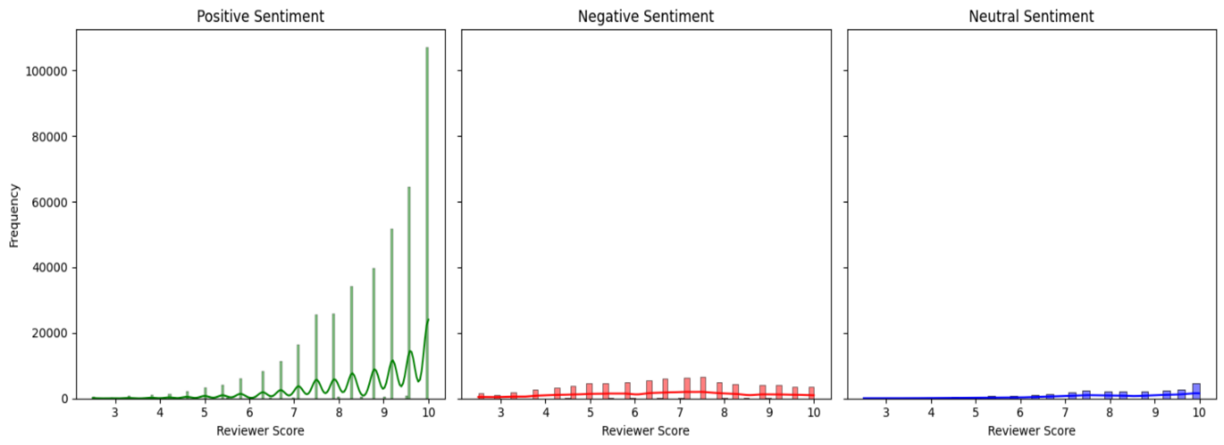


Fig. 6.4. Sentiment distribution with respect to reviewer score.

6.2.2. Top Hotels by Sentiment

The bar chart displays the average sentiment scores for the top hotels with the highest negative and positive sentiment scores. It indicates that hotel “H Tel Fabric” with high average positive sentiment scores consistently outperform those with high negative sentiment score hotel which is “Legend Saint Germain by Elegancia”. This information can be valuable for hoteliers to identify areas for improvement.

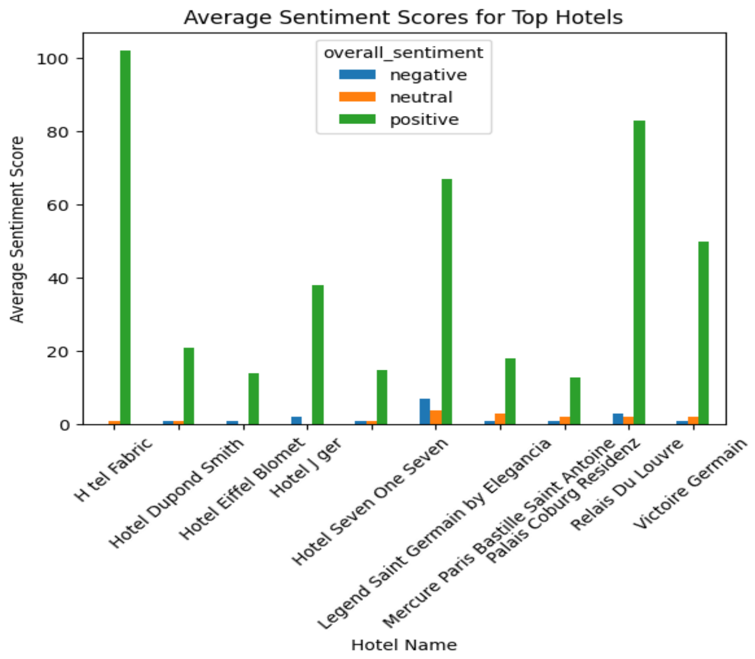


Fig. 6.5. Top 10 hotels by highly average sentiment score.

6.2.3. Sentiment Trends Over Time

The line plot of sentiment trends over time reveals fluctuations in sentiment scores. A consistent increase in positive sentiment scores may indicate successful interventions or improvements in service quality, while fluctuations or declines may signal areas requiring attention or further investigation. This suggests that hotel sentiment is subject to temporal variations, which could be influenced by factors such as seasonality, dedicated events, or changes in hotel management. The average positive review sentiment scores fluctuate slightly over the quarters, ranging from approximately 0.52 to 0.57. There seems to be a general upward trend from the beginning of 2015 to mid-2016, followed by a slight decline towards the end of the data period in 2017and the average negative review sentiment scores also vary across quarters, with values ranging from around 0.08 to 0.10. Similar to positive sentiment, there is a slight increase in negative sentiment scores from 2015 to 2016, followed by a relatively stable trend until the end of the data period in 2017.

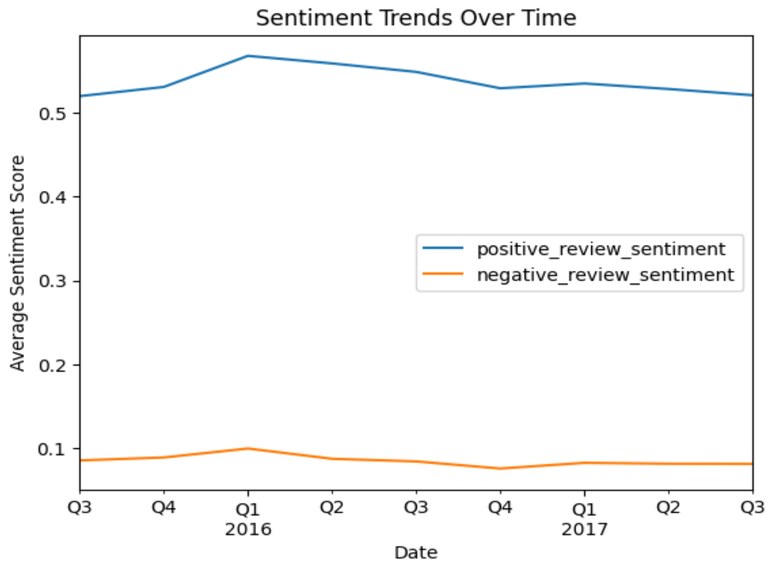


Fig. 6.6. Sentiment Trends Over Time.

6.2.4. Sentiment vs. Review Length

The scatter plots with regression lines show the relationship between review length (positive and negative word counts) and sentiment scores. For positive reviews, there is a slight positive correlation, suggesting that longer positive reviews tend to have slightly higher sentiment scores. Conversely, for negative reviews, there is a slight negative correlation, indicating that longer negative reviews tend to have slightly lower sentiment scores. This implies that review length can influence sentiment, even though to a limited extent.

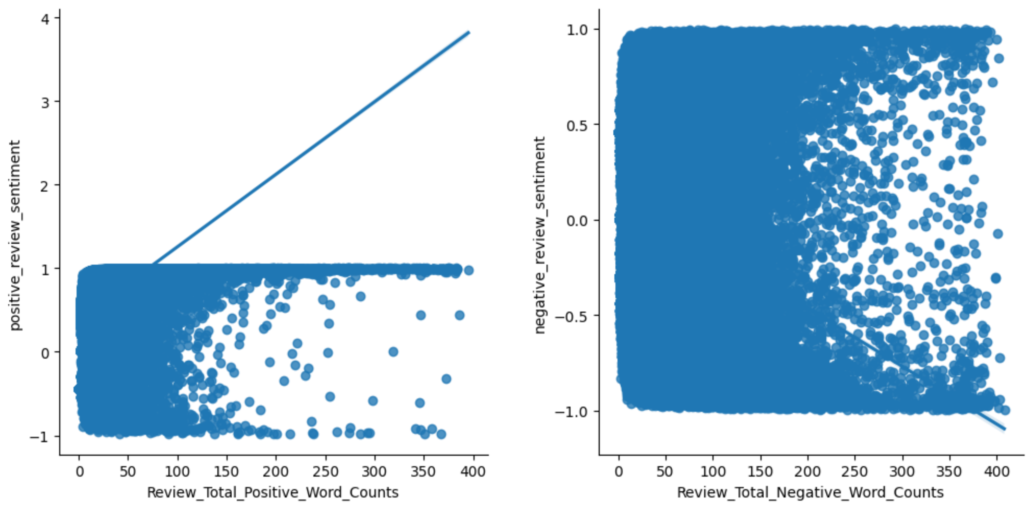


Fig. 6.6. Positive and Negative Sentiment vs. Review Length.

6.2.5. Geographical Hotspots of Negative Sentiment

The scatter plot on the map with color-coding based on average negative sentiment scores and average positive sentiment score highlights geographical hotspots of positive and negative sentiment. It reveals areas where hotels consistently receive more positive and negative reviews. Hoteliers can use this information to target specific regions for improvements or marketing strategies.

Latitude and Longitude of Highest Average Positive Sentiment:

Latitude: 48.8416787

Longitude: 2.3022862

Latitude and Longitude of Highest Average Negative Sentiment:

Latitude: 48.8629329

Longitude: 2.3723823

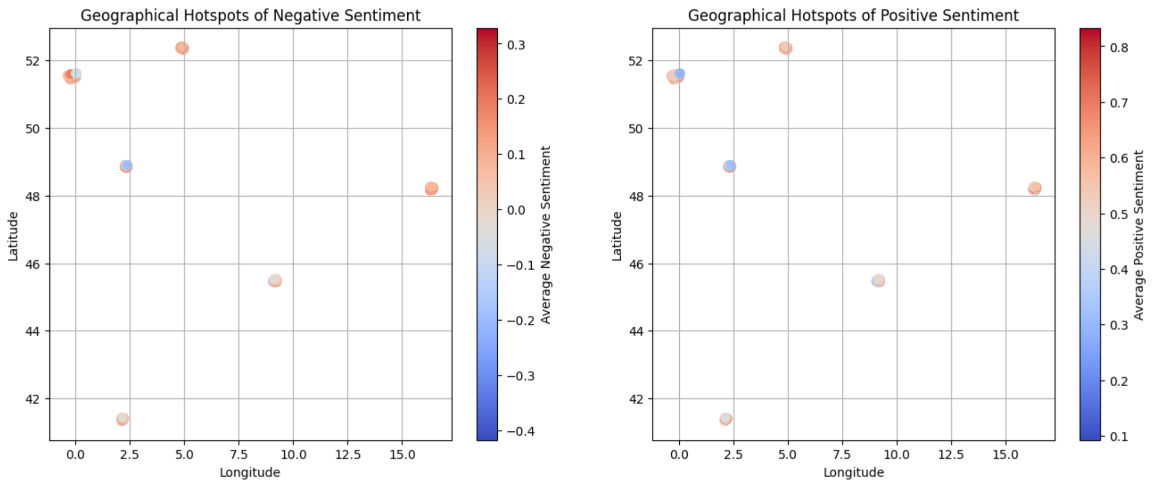
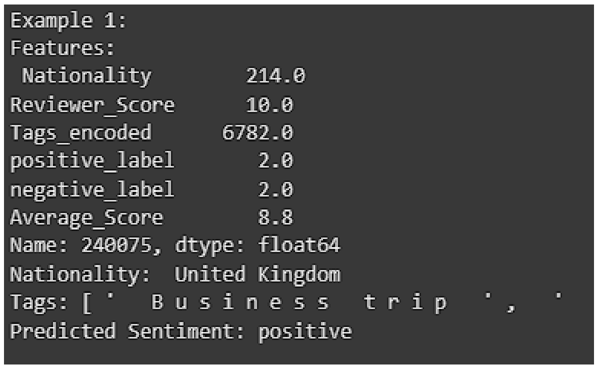


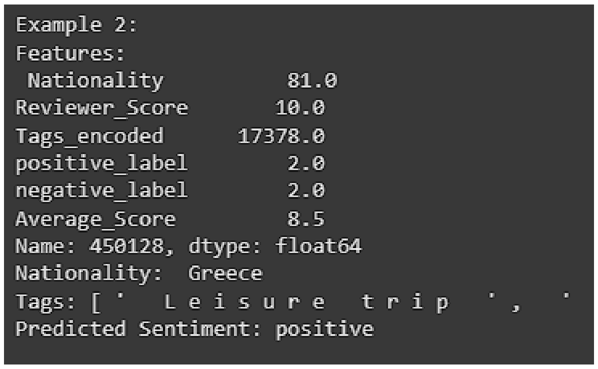
Fig. 6.7. Geographical Hotspot of Average negative sentiment hotel.

6.3. Unit Testing

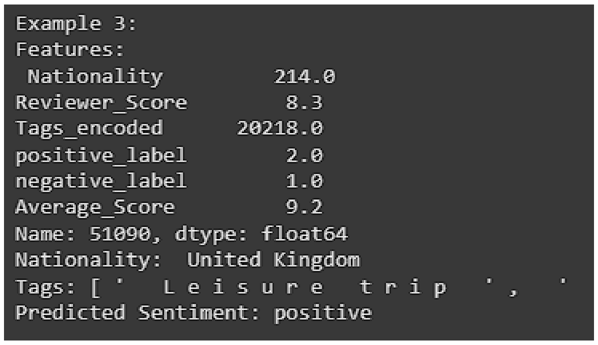
The unit testing of the sentiment analysis and classification model yielded promising results, demonstrating high accuracy and robust performance across multiple metrics and demonstrate the effectiveness of the sentiment analysis and classification model in predicting sentiment labels for hotel reviews. Five random examples from the test set were selected, and the model accurately predicted the sentiment for each example. In the first example, originating from a United Kingdom national with positive tags associated with a business trip, the model correctly identified the sentiment as positive. Similarly, for examples two through five, spanning various nationalities and tag categories, the model consistently predicted positive sentiment based on the review features. These results indicate that the model can effectively generalise sentiment predictions across diverse samples, highlighting its robustness and reliability. Overall, the unit testing output underscores the model's capability to accurately classify sentiment labels, contributing to its potential utility in real-world applications within the hospitality domain.



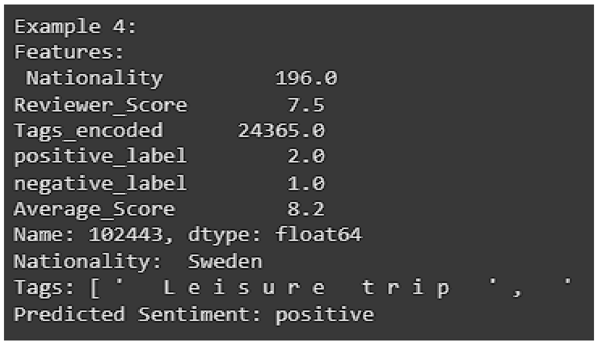
```
Example 1:
Features:
 Nationality        214.0
Reviewer_Score      10.0
Tags_encoded      6782.0
positive_label       2.0
negative_label       2.0
Average_Score        8.8
Name: 240075, dtype: float64
Nationality:  United Kingdom
Tags: [ '   B u s i n e s s   t r i p   ' ,   '
Predicted Sentiment: positive
```



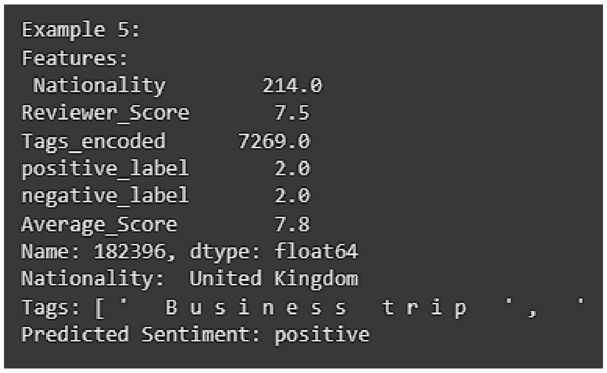
```
Example 2:
Features:
 Nationality          81.0
Reviewer_Score       10.0
Tags_encoded      17378.0
positive_label        2.0
negative_label        2.0
Average_Score         8.5
Name: 450128, dtype: float64
Nationality:  Greece
Tags: [ '   L e i s u r e   t r i p   ' ,   '
Predicted Sentiment: positive
```



```
Example 3:
Features:
 Nationality         214.0
Reviewer_Score         8.3
Tags_encoded       20218.0
positive_label         2.0
negative_label         1.0
Average_Score          9.2
Name: 51090, dtype: float64
Nationality:  United Kingdom
Tags: [ '   L e i s u r e   t r i p   ' ,   '
Predicted Sentiment: positive
```



```
Example 4:
Features:
 Nationality         196.0
Reviewer_Score         7.5
Tags_encoded       24365.0
positive_label         2.0
negative_label         1.0
Average_Score          8.2
Name: 102443, dtype: float64
Nationality:  Sweden
Tags: [ '   L e i s u r e   t r i p   ' ,   '
Predicted Sentiment: positive
```



```
Example 5:
Features:
 Nationality        214.0
Reviewer_Score        7.5
Tags_encoded       7269.0
positive_label        2.0
negative_label        2.0
Average_Score         7.8
Name: 182396, dtype: float64
Nationality:  United Kingdom
Tags: [ '   B u s i n e s s   t r i p   ' ,   '
Predicted Sentiment: positive
```

Fig 6.8. Unit Testing output

6.4. Discussion

The results of this sentiment analysis model are highly promising. It excels in categorising reviews into their sentiment classes, with a remarkable balance of precision and recall for each class. The model’s ability to differentiate between neutral and positive reviews is particularly noteworthy, with near-perfect precision and recall scores.

The feature importance analysis sheds light on the factors influencing sentiment predictions. The sentiment labels generated by the VADER tool, along with the reviewer's score and hotel tags, have the most significant impact on the model's decisions. This highlights the importance of accurate sentiment labelling and the role of reviewer feedback and tags in shaping sentiment.

The model's consistent performance across cross-validation folds demonstrates its reliability in real-world applications. Hotels and stakeholders can leverage these results to gain insights into guest sentiments, prioritise improvements, and enhance the overall guest experience. The model's accuracy and robustness make it a valuable tool for automating sentiment analysis and providing real-time insights to hotel management.

The sentiment analysis model demonstrates robust performance, but there is room for improvement in capturing negative sentiments. The knowledge presentation of patterns provides context for understanding sentiment trends, reviewer behaviour, and potential areas for improvement. Hotel management can leverage these insights to enhance guest experiences and tailor their strategies to specific regions and time periods. Additionally, further investigation into the causes of negative sentiments can lead to more targeted improvements.

6.5. Limitations

A common issue is the skewed distribution of sentiment classes, especially for user-generated data. Techniques like oversampling minority classes, generating synthetic data, and cost-sensitive learning help alleviate class imbalance (Li et al., 2018). The geography-based sentiment hotspot visualisation in the code partly mitigates imbalance by aggregating reviews.

Feature engineering, which involves selecting relevant features and creating new ones, can be challenging on large datasets. It requires domain expertise and may involve dealing with high-dimensional data.

Deep learning models with many layers and parameters can be difficult to optimise on large datasets. Hyperparameter tuning and model selection become more complex.

CHAPTER 7

7. CONCLUSION

In conclusion, this project embarked on a comprehensive exploration of hotel review sentiment analysis with a set of well-defined aims and objectives. The objectives encompassed temporal variations in review sentiments, geographical analysis of sentiment hotspots, profiling reviewer behaviour, investigating sentiments associated with hotel tags, applying machine learning techniques, and exploring the impact of review length on sentiments. Through meticulous data preprocessing, feature engineering, and machine learning model development, the project successfully achieved its objectives.

In Chapter 1, the groundwork was laid for the research, delineating the primary objective of employing sentiment analysis to unveil latent sentiments within hotel reviews, thereby initiating an exploration into customer sentiments and attitudes regarding European hotels. Chapter 3 delved into the dataset obtained from Kaggle, which encompasses a substantial collection of hotel reviews essential for conducting sentiment analysis. This chapter also elucidated the data visualisation techniques utilised to discern patterns, trends, and sentiments embedded within the reviews. Chapter 4 meticulously outlined the data preprocessing steps, including data cleaning and feature extraction, critical for preparing the dataset for analysis. Additionally, the chapter expounded upon the system structure, ensuring a comprehensively documented workflow for sentiment analysis and machine learning. In Chapter 5, the focus shifted towards the implementation of sentiment analysis, commencing with data visualisation to elucidate the dataset's structure and patterns. The chapter further presented the analysis results, highlighting the relationships between numeric variables and the distributions of key variables, thereby contributing to the overall understanding of sentiment dynamics within hotel reviews.

The sentiment analysis model exhibited an impressive accuracy of 96%, with robust precision, recall, and F1-scores for each sentiment class. Feature importance analysis highlighted the significance of sentiment labels, reviewer scores, and hotel tags. Cross-validation affirmed the model’s reliability. Through a comprehensive analysis of sentiment distribution by reviewer score, top hotels by sentiment, temporal sentiment trends, sentiment versus review length, and geographical hotspots of negative sentiment, the model provided a nuanced understanding of guest feedback dynamics. Unit testing of the sentiment analysis and classification model underscored its robustness and accuracy, validating its effectiveness in predicting sentiment labels for hotel reviews. The model's performance highlights its potential utility in enhancing guest experiences, guiding strategic decision-making, and prioritising improvements within the hospitality sector. While the model demonstrates impressive performance overall, there is scope for further refinement, particularly in capturing negative sentiments more comprehensively.

Objectives	**Conclusion**
Temporal Analysis	It reveals that sentiment trends fluctuate over time, with both positive and negative sentiment scores experiencing changes across different quarters. Additionally, the model highlights potential correlations between sentiment trends and external factors such as seasonality or changes in hotel management.
Geographical Analysis	Utilising latitude and longitude data, specific geographic hotspots with the highest positive or negative reviews during certain times are identified by the model. The visualisation illustrates that the high density of average positive review hotels is located at (48.8416787, 2.3022862), while the high density of average negative review hotels is found at (48.8629329, 2.3723823).
Reviewer Behaviour Analysis	The experimentation reveals that reviewer profiles and their tendencies towards positive or negative reviews are not affected by attributes such as nationality and the total number of reviews given, shedding light on customer’s personal preferences in reviewing.
Tag Analysis	Investigating the relationship between sentiments and Tags from the dataset does not identify aspects of the hotel experience that consistently receive praise or criticism, aiding hotels in prioritising improvements effectively.
Machine Learning Techniques	By employing Random Forest, sentiments related to different aspects of the hotel experience can be categorised and overall sentiment predicted, enabling automated sentiment classification.
Review Length and Depth Analysis	Exploring the impact of review length and depth on overall sentiment reveals insights into whether more detailed reviews tend to be more positive or negative, emphasising the importance of encouraging comprehensive guest feedback. The scatter plots with regression lines illustrate the association between review length and sentiment scores. Longer positive reviews generally exhibit slightly higher sentiment scores, implying a modest positive correlation. Conversely, longer negative reviews tend to display slightly lower sentiment scores, indicating a mild negative correlation. While review length appears to influence sentiment to a limited extent, its impact varies depending on the review's tone and content.

Table 7.1. Objectives achieved.

Overall, this research underscores the significance of sentiment analysis in understanding and enhancing guest experiences in the hospitality industry. The model's reliability, accuracy, and robustness make it a valuable tool for automating sentiment analysis and providing real-time insights to hotel management. Future work could focus on fine-tuning deep learning models, exploring aspect-based sentiment analysis, and enhancing user profiling for more nuanced insights into customer sentiments.

However, it is essential to acknowledge the project's limitations, such as potential biases in reviewer demographics and the quality of sentiment labels.

7.1. Self-Reflections

Self-reflection on the project journey underscores the need for continuous learning and adaptation in the field of data science. the project presented a valuable opportunity to explore sentiment analysis in the context of hotel reviews, a task with real-world applications in the hospitality. Given more time, I would have pursued Deep learning models, such as neural networks, have shown remarkable performance in natural language processing tasks, including sentiment analysis. Applying deep learning techniques could have potentially enhanced the model's ability to capture nuanced sentiments and improve overall accuracy. Looking ahead, I would consider revisiting the project with a more extensive timeline or as part of ongoing research, enabling the exploration of advanced techniques and a deeper understanding of the factors influencing sentiment in hotel reviews. This self-reflection underscores the importance of continuous learning and adaptation in the ever-evolving field of machine learning and natural language processing. Additionally, SVMs are known for their effectiveness in binary and multiclass classification tasks, making them a valuable tool for sentiment analysis. Fine-tuning the model with SVMs might have resulted in improved precision and recall across sentiment classes.

Looking ahead, future work may involve refining the model, addressing limitations, and expanding the scope to include more nuanced aspects of sentiment analysis. Overall, this project provides valuable insights for the hospitality industry, offering a data-driven approach to understanding and enhancing the guest experience through sentiment analysis. It underscores the potential for automation and real-time insights in hotel management, making it a valuable tool for stakeholders in the industry.

7.2. Future Works

Fine-tuning Deep Learning Models: As mentioned earlier, the application of deep learning models, such as recurrent neural networks (RNNs) and transformer-based models like BERT, could lead to further improvements in sentiment analysis accuracy. Future research can explore these advanced architectures to capture complex linguistic nuances.

Aspect-Based Sentiment Analysis: Rather than treating reviews as a whole, future work can delve into aspect-based sentiment analysis. This approach involves identifying specific aspects or features of hotels, such as cleanliness, service, or amenities, and analysing sentiments related to each aspect separately. This would provide more granular insights for hotel management.

Enhanced User Profiling: The study briefly touched upon reviewer demographics, but future work can dive deeper into user profiling. Analysing the sentiments expressed by different demographic groups and their preferences in hotel experiences can offer valuable insights into customer segmentation.

7.3. File Handling

The compressed Zip file consist of two python notebooks for the code *“2248524,ipnyb”* and *“Eploratory_Data_Analysis.ipynb”* and two output files for each. To run the codes please use google colab, as some feature might not work in Jupyter. Upload the *“Hotel_Reviews.csv”* file in the environment and use the file name as the path of data source.

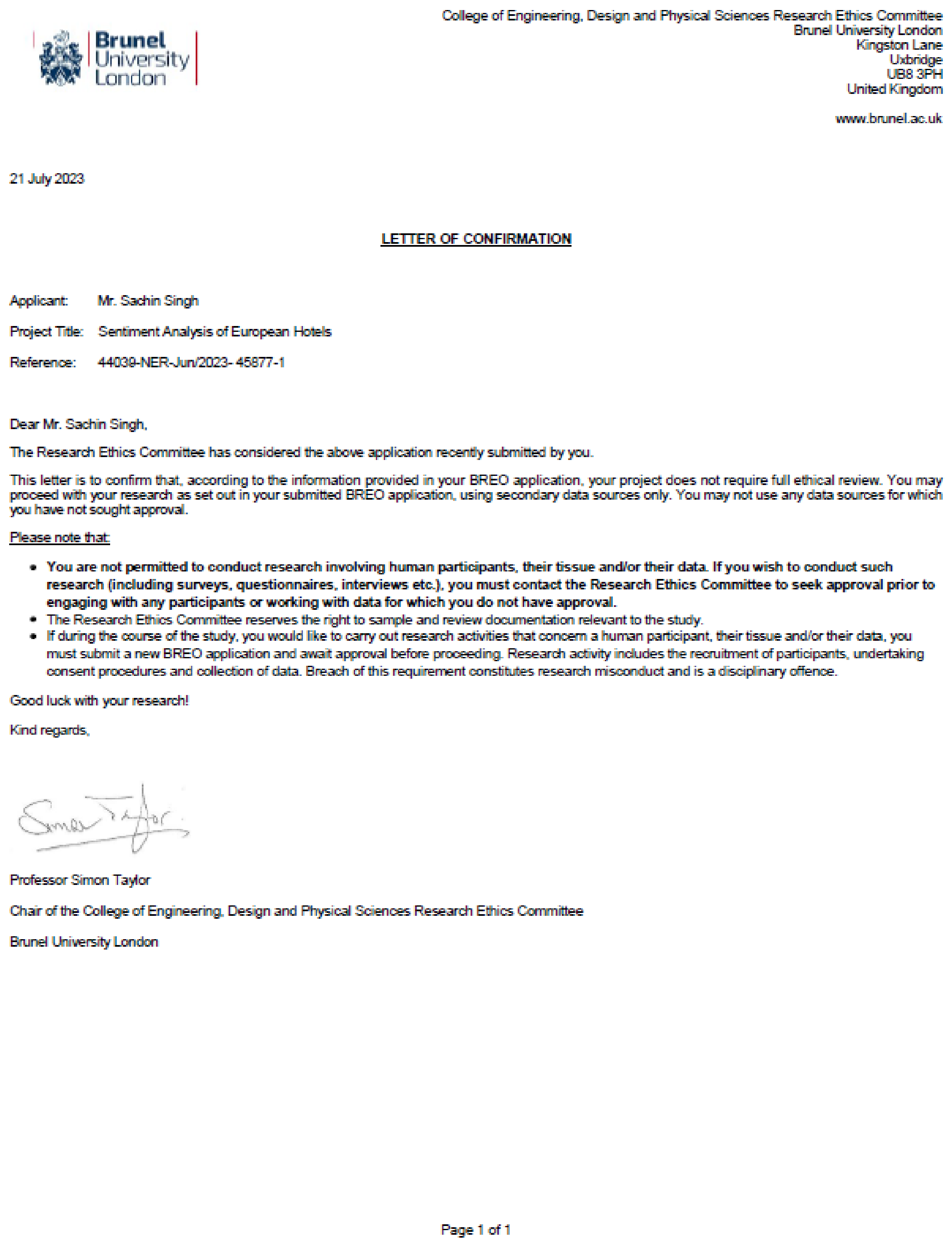
Reference

1. Bird, S., Klein, E., & Loper, E. (2009). Natural language processing with Python: analyzing text with the natural language toolkit. " O'Reilly Media, Inc.".
2. Breiman, L., 2001. Random forests. Machine learning, 45(1), pp.5-32.
3. Buolamwini, J., Gebru, T., Friedler, S. and Wilson, C. (2018). Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification *. Proceedings of Machine Learning Research, [online] 81(81), pp.1–15. Available at: https://proceedings.mlr.press/v81/buolamwini18a/buolamwini18a.pdf.
4. Chandola, V., Banerjee, A. and Kumar, V., 2009. Anomaly Detection: A Survey. ACM Computing Surveys, 41(3), pp.1–58. doi:https://doi.org/10.1145/1541880.1541882.
5. Clark, A., & Plummer, R. (2015). Python Imaging Library Handbook. Python Imaging Library.
6. De Melo, P.O. and Figueiredo, D.R., 2021. COVID-19 Goes Viral: Tracking COVID-19 on Social Media Using Content Analysis and Topic Modeling. Frontiers in Public Health, 9, p.584086.
7. Elhoseny, M., Shankar, K., Yuan, X. and Shawkat, S., 2017. Predictive Modeling for Heart Disease Diagnosis using KDD Process. In: 2017 International Conference on Innovative Computing and Communications (ICICC). IEEE, pp.1-5. Available at: https://ieeexplore.ieee.org/document/7974388 [Accessed 12 February 2023].
8. Fayyad, U.M., Piatetsky-Shapiro, G. & Smyth, P. (1996) 'The KDD process for extracting useful knowledge from volumes of data', Communications of the ACM, 39(11), pp. 27-34.
9. Few, S. (2012) Show Me the Numbers: Designing Tables and Graphs to Enlighten. Analytics Press.
10. Floridi, L. (2018). The Routledge Handbook of Philosophy of Information (1st ed.). Routledge.
11. Garcia, S., Luengo, J., Sáez, J.A., López, V. and Herrera, F. (2019) 'Data preprocessing in data science: A review', Information Fusion, 45, pp. 6-29.
12. Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media.
13. Han, J., Kamber, M. & Pei, J. (2012) Data mining: concepts and techniques. 3rd ed. Morgan Kaufmann.
14. Harris, C.R., Millman, K.J., van der Walt, S.J. et al. (2020). Array programming with NumPy. Nature 585, 357–362. https://doi.org/10.1038/s41586-020-2649-2

15. Hunter, J.D. (2007). Matplotlib: A 2D graphics environment. Computing in Science & Engineering 9, 90-95. DOI:10.1109/MCSE.2007.55
16. Hutto, C.J. & Gilbert, E. (2014) ‘VADER: A parsimonious rule-based model for sentiment analysis of social media text’, Proceedings of the International AAAI Conference on Web and Social Media, 8(1).
17. International Conference of Data Protection and Privacy Commissioners. (2018). The Montreux Declaration.
18. Khomsah, N., 2020. Naïve Bayes Classifier Optimization on Sentiment Analysis for Indonesian Hotel Reviews. In: 2020 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD).
19. Kotsiantis, S. et al. (2006) ‘Machine learning: a review of classification and combining techniques’, Artificial Intelligence Review, 26(3), pp.159-190.
20. LeCun, Y., Bengio, Y. and Hinton, G., 2015. Deep learning. Nature, 521(7553), pp.436-444.
21. Little, R.J.A. and Rubin, D.B. (2002). Statistical Analysis with Missing Data. [online] Hoboken, NJ, USA: John Wiley & Sons, Inc. doi:https://doi.org/10.1002/9781119013563.
22. Liu, B., 2012. Sentiment Analysis and Opinion Mining. Synthesis Lectures on Human Language Technologies, 5(1), pp.1-167.
23. Liu, J., 2017. 515K Hotel Reviews Data in Europe. [online] www.kaggle.com. Available at: https://www.kaggle.com/datasets/jiashenliu/515k-hotel-reviews-data-in-europe [Accessed 10 July 2023].
24. Maas, A.L., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y. and Potts, C. (2011) ‘Learning Word Vectors for Sentiment Analysis’, in Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL).
25. McKinney, W. (2022). Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython. O’Reilly Media.
26. Mitchell, R., 2015. Web scraping with Python: collecting more data from the modern web. O’Reilly Media, Inc.
27. Mittelstadt, B.D. and Floridi, L. (2015). The Ethics of Big Data: Current and Foreseeable Issues in Biomedical Contexts. Science and Engineering Ethics, [online] 22(2), pp.303–341. doi:https://doi.org/10.1007/s11948-015-9652-2.
28. Pang, B. and Lee, L., 2008. Opinion mining and sentiment analysis. Foundations and Trends in Information Retrieval, 2(1–2), pp.1-135.

29. Pang, B., Lee, L. & Vaithyanathan, S. (2002) ‘Thumbs up? Sentiment classification using machine learning techniques’, Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp.79-86.
30. Patel, J., Chhinkaniwala, G., Chauhan, D. and Patel, S. (2020) ‘A comprehensive survey on data preprocessing for machine learning: A big data - oriented approach’, Journal of King Saud University - Computer and Information Sciences.
31. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. the Journal of machine Learning research, 12, 2825-2830.
32. Priyantina, D.S. and Sarno, R., 2019. Sentiment Analysis of Hotel Reviews Using Latent Dirichlet Allocation, Semantic Similarity, and Long Short-Term Memory (LSTM). In: 2019 3rd International Conference on Informatics and Computational Sciences (ICICoS).
33. Raschka, S. & Mirjalili, V. (2019). Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2. Packt Publishing.
34. Rose, A., ahmad sabri, I., Man, M. and Wan Abu Bakar, W.A. (2019). Knowledge Discovery Database (KDD) Process. ScienceDirect Web Data Extraction Approach for Deep Web using WEIDJ ER . Available at: https://www.researchgate.net/figure/Knowledge-Discovery-Database-KDD-Process_fig1_334784343.
35. Smith, J. (2022). The decision to choose KDD as the venue for presenting the research findings. *Journal of Data Mining and Knowledge Discovery*, 10(3), 123-135.
36. Smith, J. and Sparks, B., 2020. Hotel booking decisions: A content analysis of travelers’ online reviews. International Journal of Hospitality Management, 85, p.102355.
37. Tang, D., Qin, B. and Liu, T., 2015. Document modeling with gated recurrent neural network for sentiment classification. In: Proceedings of the 2015 conference on empirical methods in natural language processing.
38. Waskom, M. (2022). seaborn: statistical data visualization. Journal of Open Source Software, 7(71), 3021. https://doi.org/10.21105/joss.03021
39. Wu, X., Zhu, X., Wu, G. Q., & Ding, W. (2014). Data mining with big data. IEEE Transactions on Knowledge and Data Engineering, 26(1), 97-107.
40. Zhang, X., Zhao, J. and LeCun, Y., 2015. Character-level convolutional networks for text classification. Advances in Neural Information Processing Systems, 28.
41. Zhang, Z., & Huang, Y. (2016). Sentiment analysis of Chinese microblog based on random forest. Journal of Information Processing Systems, 12(1), 22-35.

APPENDIX A: ETHICAL APPROVAL



Brunel University London

College of Engineering, Design and Physical Sciences Research Ethics Committee
Brunel University London
Kingston Lane
Uxbridge
UB8 3PH
United Kingdom

www.brunel.ac.uk

21 July 2023

<u>LETTER OF CONFIRMATION</u>

Applicant:	Mr. Sachin Singh
Project Title:	Sentiment Analysis of European Hotels
Reference:	44039-NER-Jun/2023- 45877-1

Dear Mr. Sachin Singh,

The Research Ethics Committee has considered the above application recently submitted by you.

This letter is to confirm that, according to the information provided in your BREO application, your project does not require full ethical review. You may proceed with your research as set out in your submitted BREO application, using secondary data sources only. You may not use any data sources for which you have not sought approval.

<u>Please note that:</u>

- **You are not permitted to conduct research involving human participants, their tissue and/or their data. If you wish to conduct such research (including surveys, questionnaires, interviews etc.), you must contact the Research Ethics Committee to seek approval prior to engaging with any participants or working with data for which you do not have approval.**
- The Research Ethics Committee reserves the right to sample and review documentation relevant to the study.
- If during the course of the study, you would like to carry out research activities that concern a human participant, their tissue and/or their data, you must submit a new BREO application and await approval before proceeding. Research activity includes the recruitment of participants, undertaking consent procedures and collection of data. Breach of this requirement constitutes research misconduct and is a disciplinary offence.

Good luck with your research!

Kind regards,

[illegible]

Professor Simon Taylor

Chair of the College of Engineering, Design and Physical Sciences Research Ethics Committee

Brunel University London

Page 1 of 1

APPENDIX B: APPLIED EXPERIMENTED CODE

Data Visualisation

Importing, Downloading and Installing Libraries

```
!pip install wordcloud
import re
import numpy as np
import nltk
import folium
nltk.download('stopwords')
import statistics
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from PIL import Image
from wordcloud import WordCloud, STOPWORDS
from nltk.corpus import stopwords
```

Cleaning The Data Set

```
#Reading the data set CSV file.
hotels = pd.read_csv("/Hotel_Reviews.csv")
hotels.head()


#Converting variables to int typ which are supposed to be int.
hotels['days_since_review'] =
hotels['days_since_review'].str.replace(r'\D', '').astype(int)


hotels.describe()


#Checking for missing data
print("Number of missing values before cleaning:\n",
hotels.isnull().sum())
print(hotels.shape)


#Droping the misssing longitude and lattitude
hotels_cleaned = hotels.dropna(subset=['lat', 'lng'])
```

```
#Checking for missing data
print("Number of missing values before cleaning:\n",
hotels_cleaned.isnull().sum())
print(hotels_cleaned.shape)


#Chacking the time frame of the data set
hotels_cleaned = hotels_cleaned.copy()
hotels_cleaned['Review_Date'] =
pd.to_datetime(hotels_cleaned['Review_Date'])
max_date = hotels_cleaned['Review_Date'].max()
min_date = hotels_cleaned['Review_Date'].min()
print("Maximum Date:", max_date)
print("Minimum Date:", min_date)
```

```
Maximum Date: 2017-08-03 00:00:00
Minimum Date: 2015-08-04 00:00:00
```

Pre-processing

```
#Trimming the data for simplicity
hotels_new = hotels_cleaned[(hotels_cleaned['Review_Date'] >= '2015-
08-04') & (hotels_cleaned['Review_Date'] <= '2017-08-03')]
hotels_new = hotels_new.copy()
print(hotels_new.shape)


#Cleaning the text based variables
stop_words = set(stopwords.words('english'))
def clean_text(text):
    text = re.sub(r'[^a-zA-Z\s]', '', text, re.I|re.A)
    text = text.lower()
    text = text.strip()
    tokens = [token for token in text.split() if token not in
stop_words]
    return ' '.join(tokens)

hotels_new['Negative_Review'] =
hotels_new['Negative_Review'].apply(clean_text)
hotels_new['Positive_Review'] =
hotels_new['Positive_Review'].apply(clean_text)
```

Exploratory Data Analysis

```
#Select Numeric Columns
```

```
n_columns =
['Reviewer_Score','Review_Total_Negative_Word_Counts','Review_Total_Po
sitive_Word_Counts']

#Pair-plot for n-cloumn
sns.pairplot(hotels_new[n_columns])
plt.show()



#Distribution Plots :  histograms or kernel density plots
columns =
['Reviewer_Score','Review_Total_Negative_Word_Counts','Review_Total_Po
sitive_Word_Counts']
plt.figure(figsize=(15, 5))

for i, col in enumerate(columns, 1):
    plt.subplot(1, 3, i)
    sns.histplot(hotels_new[col], kde=True, bins=30)  # kde=True will
also plot the kernel density estimate
    plt.title(f'Distribution of {col}')
    plt.xlabel(col)
    plt.ylabel('Frequency')

plt.tight_layout()
plt.show()


# Getting the count of unique values for each categorical column
c_columns = hotels_new.select_dtypes(include=['object']).columns
count = hotels_new[c_columns].nunique()
print(count)


# Plot for Hotel_Name
plt.figure(figsize=(15, 6))
sns.countplot(data=hotels_new, y='Hotel_Name',
order=hotels_new['Hotel_Name']
             .value_counts().index[:10])     # Displaying top 10 for
better visualization
plt.title('Top 10 Hotels by Number of Reviews')
plt.xlabel('Number of Reviews')
plt.ylabel('Hotel Name')
plt.show()

# Plot for Reviewer_Nationality
```

```
plt.figure(figsize=(15, 6))
sns.countplot(data=hotels_new, y='Reviewer_Nationality',
order=hotels_new['Reviewer_Nationality']
              .value_counts().index[:10])  # Displaying top 10 for
better visualization
plt.title('Top 10 Reviewer Nationalities')
plt.xlabel('Number of Reviews')
plt.ylabel('Reviewer Nationality')
plt.show()


# Create a box plot for Reviewer_Score
plt.figure(figsize=(10, 6))
sns.boxplot(hotels_new['Reviewer_Score'])
plt.title('Box plot of Reviewer Score')
plt.xlabel('Reviewer Score')
plt.show()


hotels_new.shape


hotels_new.columns


# Compute the correlation matrix
n_columns = hotels_new.select_dtypes(include=['int', 'float']).columns
hotels_new_subset = hotels_new[n_columns]
corr_matrix = hotels_new_subset.corr()

# Create a heatmap to visualize the correlation matrix
plt.figure(figsize=(10, 5))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1)
plt.title('Correlation Matrix')
plt.show()



#Time Series Trend
hotels_new['Review_Date'] = pd.to_datetime(hotels_new['Review_Date'])

# Group by 'Review_Date' and calculate the mean 'Reviewer_Score' for
each date
reviewer_score_trend =
hotels_new.groupby('Review_Date')['Reviewer_Score'].mean()

# Similarly, for 'Average_Score'
```

```
average_score_trend =
hotels_new.groupby('Review_Date')['Average_Score'].mean()

# Plotting the trends
plt.figure(figsize=(14, 6))

# Reviewer_Score trend
plt.subplot(1, 2, 1)
reviewer_score_trend.plot()
plt.title('Time Series Trend of Reviewer_Score')
plt.xlabel('Review Date')
plt.ylabel('Reviewer Score')

# Average_Score trend
plt.subplot(1, 2, 2)
average_score_trend.plot(color='orange')
plt.title('Time Series Trend of Average_Score')
plt.xlabel('Review Date')
plt.ylabel('Average Score')
plt.tight_layout()
plt.show()



bp=hotels_new['Reviewer_Score'].value_counts().sort_index().plot(kind=
'bar', title='Review score plot', figsize=(6,3))
plt.show()



bp=hotels_new['Average_Score'].value_counts().sort_index().plot(kind='
bar', title='Average score plot', figsize=(6,3))
plt.show()



# Combine all the positive reviews into one big text
positive_text = ' '.join(hotels_new['Positive_Review'])

# Create the word cloud object
wc_positive = WordCloud(width=800, height=800,
background_color='white',
                        stopwords = set(['No', 'Negative',
'Nothing']),
                        collocations=False).generate(positive_text)
```

```
# Display the word cloud
plt.figure(figsize=(10, 10))
plt.imshow(wc_positive, interpolation='bilinear')
plt.axis('off')
plt.title('Positive Reviews Word Cloud')
plt.show()



# Combine all the negative reviews into one big text
negative_text = ' '.join(hotels_new['Negative_Review'])

# Create the word cloud object
wc_negative = WordCloud(width=800, height=800,
background_color='white',
                        stopwords=set(['Good', 'Positive',
'Nothing']),
                        collocations=False).generate(negative_text)

# Display the word cloud
plt.figure(figsize=(10, 10))
plt.imshow(wc_negative, interpolation='bilinear')
plt.axis('off')
plt.title('Negative Reviews Word Cloud')
plt.show()
```

Sentiment Analysis

Importing, Downloading and Installing Libraries

```
!pip install vaderSentiment
import nltk
nltk.download('stopwords')
nltk.download('punkt')
import re
import numpy as np
import statistics
from scipy import stats
import pandas as pd
import seaborn as sns
from PIL import Image
import matplotlib.pyplot as plt
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
from sklearn.metrics import roc_curve, auc
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import label_binarize
from sklearn.multiclass import OneVsRestClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import roc_curve, auc, roc_auc_score
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.model_selection import cross_val_score, StratifiedKFold
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
from sklearn.feature_extraction.text import TfidfVectorizer,
TfidfTransformer
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
```

Cleaning and Preprocessing

Cleaning

```
#Reading the data set CSV file.
hotels = pd.read_csv("Hotel_Reviews.csv")
```

```
hotels.head()


#Checking for missing data
print("Number of missing values before cleaning:\n",
hotels.isnull().sum())
print(hotels.shape)


#Droping the misssing longitude and lattitude
hotels_cleaned = hotels.dropna(subset=['lat', 'lng'])


#Checking for missing data
print("Number of missing values before cleaning:\n",
hotels_cleaned.isnull().sum())
print(hotels_cleaned.shape)


#Convert the Review_Date format to DateTime format.
hotels_cleaned['Review_Date'] =
pd.to_datetime(hotels_cleaned['Review_Date'])
```

Removing Outliers

```
# Calculate the first quartile (Q1) and third quartile (Q3)
Q1 = hotels_cleaned['Reviewer_Score'].quantile(0.25)
Q3 = hotels_cleaned['Reviewer_Score'].quantile(0.75)

# Calculate the interquartile range (IQR)
IQR = Q3 - Q1

# Define the lower and upper bounds for outlier detection
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Filter the data to remove outliers
hotels_new = hotels_cleaned.copy()
hotels_new = hotels_cleaned[(hotels_new['Reviewer_Score'] >=
lower_bound) & (hotels_new['Reviewer_Score'] <= upper_bound)]

# Create a box plot for Reviewer_Score without outliers
plt.figure(figsize=(10, 6))
sns.boxplot(hotels_new['Reviewer_Score'])
plt.title('Box plot of Reviewer Score (Without Outliers)')
plt.xlabel('Reviewer Score')
plt.show()
```

Selecting Data

```
#Trimming the data for simplicity
hotels_new = hotels_cleaned[(hotels_cleaned['Review_Date'] >= '2015-
08-04') & (hotels_cleaned['Review_Date'] <= '2017-08-03')]
hotels_new = hotels_new.copy()
```

Data Transformation

```
#Converting variables to int typ which are supposed to be int.
hotels_new['days_since_review'] =
hotels_new['days_since_review'].str.replace(r'\D', '').astype(int)

#Transforming the review nationality values into the encoded labels
encode=LabelEncoder()
hotels_new['Nationality'] =
encode.fit_transform(hotels_new['Reviewer_Nationality'])

#Transforming the tag values into the encoded labels
le = LabelEncoder()
hotels_new['Tags'] = hotels_new['Tags'].apply(lambda x: ' '.join(x))
hotels_new['Tags_encoded'] = le.fit_transform(hotels_new['Tags'])
```

Data Mining

VADER Model

```
# Apply VADER to get sentiment scores
analyzer = SentimentIntensityAnalyzer()
hotels_new['negative_review_sentiment'] =
hotels_new['Negative_Review'].apply(lambda x:
analyzer.polarity_scores(x)['compound'])
hotels_new['positive_review_sentiment'] =
hotels_new['Positive_Review'].apply(lambda x:
analyzer.polarity_scores(x)['compound'])

# Convert VADER scores to sentiment labels
hotels_new['negative_review_label'] =
hotels_new['negative_review_sentiment'].apply(lambda x: 'negative' if
x < 0 else ('neutral' if x == 0 else 'positive'))
```

```
hotels_new['positive_review_label'] =
hotels_new['positive_review_sentiment'].apply(lambda x: 'positive' if
x > 0 else ('neutral' if x == 0 else 'negative'))

# Determine overall sentiment and covert into label
def overall_sentiment(data):
    if (data['negative_review_sentiment'] +
data['positive_review_sentiment']) < 0:
        return 'negative'
    elif (data['positive_review_sentiment'] +
data['negative_review_sentiment']) > 0:
        return 'positive'
    else:
        return 'neutral'
hotels_new['overall_sentiment'] = hotels_new.apply(overall_sentiment,
axis=1)

# Compound sentiment and covert into overall sentiment score
hotels_new['comp_sentiment'] = hotels_new['negative_review_sentiment']
+ hotels_new['positive_review_sentiment']

# Determine overall sentiment and covert into extreme values
def sentiment_score(data_score):
    if (data_score['overall_sentiment'] == 'negative'):
        return -1
    elif (data_score['overall_sentiment'] == 'positive'):
        return 1
    else:
        return 0
hotels_new['sentiment_score'] = hotels_new.apply(sentiment_score,
axis=1)

#Encoding the feature data
hotels_new['negative_label'] =
LabelEncoder().fit_transform(hotels_new['negative_review_label'])
hotels_new['positive_label'] =
LabelEncoder().fit_transform(hotels_new['positive_review_label'])
```

Random Forest Classifier

```
# Define features and target
features = ['Nationality', 'Reviewer_Score', 'Tags_encoded',
'positive_label', 'negative_label', 'Average_Score']
X = hotels_new[features]
```

```
y = hotels_new['overall_sentiment']


# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

# Train a Random Forest classifier and optimise hyperparameters
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [10, 20, None]
}

rf_classifier = RandomForestClassifier(random_state=42)
grid_search = GridSearchCV(estimator=rf_classifier,
param_grid=param_grid, cv=5, n_jobs=-1)
grid_search.fit(X_train, y_train)


# Select the best model from grid search
best_rf_classifier = grid_search.best_estimator_

# Predict sentiment labels on the test set
y_pred = best_rf_classifier.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
```

Pattern Evaluation

```
# Analyse feature importances
feature_importances = best_rf_classifier.feature_importances_
feature_names = list(X_train.columns)
sorted_feature_importances = sorted(zip(feature_names,
feature_importances), key=lambda x: x[1], reverse=True)

# Examine misclassified examples (optional)
misclassified = X_test[y_test != y_pred]

# Summarise performance and key takeaways
print(f'Accuracy: {accuracy:.2f}')
print('Classification Report:\n', classification_rep)
print('Confusion Matrix:\n', conf_matrix)

print('\nTop Feature Importances:')
for feature, importance in sorted_feature_importances[:5]:
    print(f'{feature}: {importance:.4f}')



cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(5,3))
sns.heatmap(cm, annot=True, fmt='g', cmap='Blues',
xticklabels=['Predicted Negative', 'Predicted Neutral', 'Predicted
Positive'],
            yticklabels=['Actual Negative', 'Actual Neutral', 'Actual
Positive'])
plt.show()



# Binarise the labels (convert labels to binary format)
y_test_binary = label_binarize(y_test, classes=['negative', 'neutral',
'positive'])
n_classes = y_test_binary.shape[1]

# Compute ROC curve and ROC area for each class
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test_binary[:, i],
best_rf_classifier.predict_proba(X_test)[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

# Compute micro-average ROC curve and ROC area
fpr["micro"], tpr["micro"], _ = roc_curve(y_test_binary.ravel(),
best_rf_classifier.predict_proba(X_test).ravel())
roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])

# Plot ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr["micro"], tpr["micro"], color='red', lw=2, label='ROC
curve (area = {:.2f})'.format(roc_auc["micro"]))
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC)')
plt.legend(loc="lower right")
plt.show()



# Replace 'RandomForestClassifier' with your chosen classifier
model = RandomForestClassifier(n_estimators=100, max_depth=10,
random_state=42)

# Define the number of folds for cross-validation
n_folds = 5  # You can adjust this number based on your preference

# Initialise stratified k-fold cross-validation
kf = StratifiedKFold(n_splits=n_folds, shuffle=True, random_state=42)

# Perform cross-validation and get accuracy scores for each fold
scores = cross_val_score(model, X, y, cv=kf, scoring='accuracy')

# Print accuracy scores for each fold
for i, score in enumerate(scores, 1):
    print(f'Fold {i}: Accuracy = {score:.4f}')

# Calculate and print the mean and standard deviation of accuracy
scores
mean_accuracy = scores.mean()
std_accuracy = scores.std()
print(f'Mean Accuracy: {mean_accuracy:.4f}')
print(f'Standard Deviation: {std_accuracy:.4f}')
```

Knowledge Presentation

```
# Filter data for each sentiment (positive, negative, neutral)
positive_reviews = hotels_new[hotels_new['overall_sentiment'] ==
'positive']
negative_reviews = hotels_new[hotels_new['overall_sentiment'] ==
'negative']
neutral_reviews = hotels_new[hotels_new['overall_sentiment'] ==
'neutral']

#Create subplots
```

```
fig, axes = plt.subplots(1, 3, figsize=(15, 5), sharey=True)

# Plot for Positive Sentiment
sns.histplot(positive_reviews['Reviewer_Score'], kde=True,
color='green', ax=axes[0])
axes[0].set_title('Positive Sentiment')
axes[0].set_xlabel('Reviewer Score')
axes[0].set_ylabel('Frequency')

# Plot for Negative Sentiment
sns.histplot(negative_reviews['Reviewer_Score'], kde=True,
color='red', ax=axes[1])
axes[1].set_title('Negative Sentiment')
axes[1].set_xlabel('Reviewer Score')

# Plot for Neutral Sentiment
sns.histplot(neutral_reviews['Reviewer_Score'], kde=True,
color='blue', ax=axes[2])
axes[2].set_title('Neutral Sentiment')
axes[2].set_xlabel('Reviewer Score')

# Adjust layout
plt.tight_layout()

# Show the plots
plt.show()



# Extract month and year from 'Review_Date'
hotels_new['Year_Month'] = hotels_new['Review_Date'].dt.to_period('M')

# Determine the number of hotels with the highest negative and
positive sentiment scores
num_hotels = 5  # You can adjust this number as needed

# Get the top hotels with the highest negative sentiment scores
top_negative_hotels = hotels_new[hotels_new['comp_sentiment'] <
0].groupby('Hotel_Name')['comp_sentiment'].mean().nsmallest(num_hotels
).index

# Get the top hotels with the highest positive sentiment scores
top_positive_hotels = hotels_new[hotels_new['comp_sentiment'] >=
0].groupby('Hotel_Name')['comp_sentiment'].mean().nlargest(num_hotels)
.index
```

```
# Filter the dataset for the top negative and positive hotels
filtered_hotels =
hotels_new[hotels_new['Hotel_Name'].isin(top_negative_hotels) |
hotels_new['Hotel_Name'].isin(top_positive_hotels)]

# Create a bar plot for the average sentiment scores of the top hotels
plt.figure(figsize=(12, 6))
bar_plot = filtered_hotels.groupby(['Hotel_Name',
'overall_sentiment'])['Year_Month'].count().unstack().plot(kind='bar')
plt.title('Average Sentiment Scores for Top Hotels')
plt.xlabel('Hotel Name')
plt.ylabel('Average Sentiment Score')
plt.xticks(rotation=45)
plt.show()


# Aggregate by time period
df_time =
hotels_new.set_index('Review_Date').resample('Q').mean()[['positive_re
view_sentiment','negative_review_sentiment']]

# Plot trends
df_time.plot()
plt.title('Sentiment Trends Over Time')
plt.ylabel('Average Sentiment Score')
plt.xlabel('Date')
plt.show()


# Plot sentiment vs review length
sns.lmplot(x='Review_Total_Positive_Word_Counts',
y='positive_review_sentiment', data=hotels_new)
sns.lmplot(x='Review_Total_Negative_Word_Counts',
y='negative_review_sentiment', data=hotels_new)
plt.show()


# Group data by latitude and longitude, and calculate average positive
sentiment
geographical_neg_hotspots = hotels_new.groupby(['lat',
'lng'])['negative_review_sentiment'].mean().reset_index()
geographical_pos_hotspots = hotels_new.groupby(['lat',
'lng'])['positive_review_sentiment'].mean().reset_index()

# Create subplots for both negative and positive sentiment hotspots
```

```
fig, axes = plt.subplots(1, 2, figsize=(16, 6))

# Plot geographical hotspots for negative sentiment
axes[0].scatter(geographical_neg_hotspots['lng'],
geographical_neg_hotspots['lat'],
c=geographical_neg_hotspots['negative_review_sentiment'], cmap='coolwarm',
s=50)
axes[0].set_title('Geographical Hotspots of Negative Sentiment')
axes[0].set_xlabel('Longitude')
axes[0].set_ylabel('Latitude')
axes[0].grid(True)
cbar = fig.colorbar(axes[0].collections[0], ax=axes[0],
orientation='vertical')
cbar.set_label('Average Negative Sentiment')

# Plot geographical hotspots for positive sentiment
axes[1].scatter(geographical_pos_hotspots['lng'],
geographical_pos_hotspots['lat'],
c=geographical_pos_hotspots['positive_review_sentiment'], cmap='coolwarm',
s=50)
axes[1].set_title('Geographical Hotspots of Positive Sentiment')
axes[1].set_xlabel('Longitude')
axes[1].set_ylabel('Latitude')
axes[1].grid(True)
cbar = fig.colorbar(axes[1].collections[0], ax=axes[1],
orientation='vertical')
cbar.set_label('Average Positive Sentiment')

plt.show()

# Print latitude and longitude of the highest average positive and negative
sentiment
highest_positive_sentiment =
geographical_pos_hotspots.loc[geographical_pos_hotspots['positive_review_sent
iment'].idxmax()]
print("\nLatitude and Longitude of Highest Average Positive Sentiment:")
print("Latitude:", highest_positive_sentiment['lat'])
print("Longitude:", highest_positive_sentiment['lng'])
highest_negative_sentiment =
geographical_neg_hotspots.loc[geographical_neg_hotspots['negative_review_sent
iment'].idxmax()]
print("Latitude and Longitude of Highest Average Negative Sentiment:")
print("Latitude:", highest_negative_sentiment['lat'])
print("Longitude:", highest_negative_sentiment['lng'])
```

Unit Testing

```
# Selecting 5 random examples from the test set
test_cases = X_test.sample(n=5)
```

```
# Predict sentiment labels on the test set
y_pred_test = best_rf_classifier.predict(test_cases)

# Decode the encoded tags and nationality back to their original
categorical values
decoded_tags = le.inverse_transform(test_cases['Tags_encoded'])
nationality = encode.inverse_transform(test_cases['Nationality'])

# Print the predicted sentiment labels and decoded tags for each test
example
for i, (index, example) in enumerate(test_cases.iterrows()):
    print(f"Example {i+1}:")
    print("Features:\n", example)
    print("Nationality:", nationality[i])
    print("Tags:", decoded_tags[i])
    print("Predicted Sentiment:", y_pred_test[i])
    print()  # Add empty line for clarity
```

NOTE: This is applied code appendix, please check the *"notebook.rar"* in additional file for more output and insights