GRADUATE PROGRAMS INSTITUTE

THE HYBRID RECOMMENDER SYSTEM USING DEEP LEARNING FOR TOURISM IN ISTANBUL

Abdeljalil DIBE

MASTER'S THESIS

İstanbul, November 2024

HYBRID RECOMMENDER SYSTEM USING DEEP LEARNING FOR TOURISM IN ISTANBUL

Abdeljalil DIBE

MASTER'S THESIS
COMPUTER ENGINEERING DEPARTMENT
COMPUTER ENGINEERING MASTER'S PROGRAM WITH THESIS

MASTER'S THESIS ADVISOR
Asst. Prof. Dr. Özlem Feyza ERKAN

MASTER'S THESIS JURY MEMBERS
Asst. Prof. Dr. Gizem TEMELCAN ERGENECOŞAR
Asst. Prof. Dr. Nedim MUZOĞLU

PREFACE

I would like to express my sincere thanks to Asst. Prof. Dr. Ö. Feyza Erkan for her guidance and support throughout my academic journey. Her encouragement and wisdom have motivated me to aim for excellence.

I am also immensely grateful to all the members of Beykoz University. The knowledge and experiences I have gained here have profoundly shaped my academic and personal growth. Your commitment to creating a supportive and stimulating learning environment has made my time here very rewarding.

Thank you all for your support and for making this journey memorable.

CONTENTS

ABSTRACT

HYBRID RECOMMENDER SYSTEM USING DEEP LEARNING FOR TOURISM IN ISTANBUL

Istanbul, recognized for its remarkable beauty and cultural significance, is a premier destination for international travelers to Turkey. Despite its popularity, visitors frequently encounter challenges when attempting to navigate the extensive array of attractions, accommodations, and dining options available. While numerous online resources, such as blogs and forums, offer valuable information, the overwhelming volume can lead to information overload, highlighting the limitations of current recommendation technologies in the tourism sector.

This research aims to develop a sophisticated recommendation system that enhances the overall tourist experience by providing personalized recommendations tailored to individual preferences and real-time location data. The increasing demand for detailed insights into local attractions including culinary experiences, shopping venues, and points of interest, necessitates a focused approach to information delivery. To address these challenges, we will utilize data sourced from the prominent platform TripAdvisor, employing a hybrid methodology that integrates collaborative filtering and content-based techniques through Artificial Neural Networks (ANNs). This approach aims to generate precise recommendations for hotels, restaurants, and attractions, ultimately improving the decision-making process for travelers. The primary objective of this study is to establish a robust recommendation system that achieves high accuracy and precision, ensuring that users receive tailored suggestions aligned with their unique needs and preferences while visiting Istanbul.

Keywords: Recommender system; Machine Learning; Collaborative filtering; Content Based filtering; Clustering, Artificial Neural Network

Date: 04.11.2024

ÖZET

Olağanüstü güzelliği ve kültürel önemiyle tanınan İstanbul, Türkiye'ye gelen uluslararası turistler için önemli bir destinasyondur. Popüler olmasına rağmen, ziyaretçiler, mevcut geniş yelpazedeki turistik yerler, konaklama seçenekleri ve yemek seçeneklerinde gezinmeye çalışırken sık sık zorluklarla karşılaşırlar. Bloglar ve forumlar gibi birçok çevrimiçi kaynak değerli bilgiler sunsa da, aşırı miktardaki bilgi, bilgi aşırı yüküne yol açabilir ve turizm sektöründeki mevcut öneri teknolojilerinin sınırlamalarını vurgulayabilir..

Bu araştırma, bireysel tercihlere ve gerçek zamanlı konum verilerine göre kişiselleştirilmiş öneriler sağlayarak genel turist deneyimini artıran kompleks bir öneri sistemi geliştirmeyi amaçlamaktadır. Gastronomik deneyimler, alışveriş mekanları ve ilgi çekici noktalar gibi yerel atraksiyonlar hakkında ayrıntılı bilgiler talebinin artması, bilgi dağıtımına odaklanmış bir yaklaşım gerektirmektedir. Bu zorluklara çözüm geliştirmek için, önde gelen turizm platformu olan TripAdvisor'dan elde veriler kullanılmıştır. Daha sonra bu veriler Yapay Sinir Ağları (ANN) aracılığıyla işbirlikçi filtreleme ve içerik tabanlı teknikleri entegre eden hibrit bir metodoloji kullanarak işlenmiştir. Bu yaklaşım, oteller, restoranlar ve turistik yerler için kesin öneriler oluşturmayı hedeflemekte ve böylece gezginlerin karar verme sürecini iyileştirmektedir. Bu çalışmanın temel amacı, yüksek doğruluk ve kesinlik sağlayan ve kullanıcıların İstanbul'u ziyaret ederken benzersiz ihtiyaçlarına ve tercihlerine uygun olarak uyarlanmış öneriler almasını sağlayan gürbüz bir öneri sistemi oluşturmaktır.

Anahtar Kelimeler: Öneri sistemi; Makine Öğrenimi; İşbirlikçi filtreleme; İçerik Tabanlı filtreleme; Kümeleme; Yapay sinir ağı

Tarih: 04.11.2024

ABBREVIATIONS

ANN : Artificial Neural Network
AI : Artificial Intelligence
CF : Collaborating Filtering
CB : Content-Based
DL : Deep Learning
KNN : K-Nearest Neighbors
LR : Logistic Regression
ML : Machine Learning
MLP : Multi-Layer Perceptron Classifier
SVM : Support Vector Machine
RS : Recommendation System

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

User opinions have become increasingly important in enhancing customer satisfaction and refining marketing strategies in the tourism sector, where tourist feedback can directly influence the success of hotels, attractions, and other services (Smith, 2021; Johnson et al., 2022). The tourism industry plays a vital role in economic growth for many countries, and ensuring tourists have satisfying experiences is essential for building a strong reputation and attracting future visitors. By offering personalized recommendations, tourists can be guided to suitable destinations that match their preferences, which enhances their experience and encourages repeat visits (Lee, 2023). This approach also benefits stakeholders, such as hotel managers and attraction operators, who can leverage insights from user feedback to improve their offerings and attract more tourists (Brown & White, 2022).

In this context, we propose an innovative solution based on Artificial Intelligence (AI) that provides personalized recommendations for attractions, hotels, and restaurants tailored to the financial capacity and preferences of each user. This hybrid recommendation system combines content-based filtering, collaborative filtering and deep learning methods to achieve high accuracy and precision in recommendations. By addressing the limitations of individual techniques and enhancing recommendation quality (Miller, 2022; Patel & Singh, 2021). Our primary objectives are to enhance the overall tourist experience in Istanbul, provide personalized recommendations suited to each individual's preferences, and achieve high levels of accuracy and precision in recommendations.

This thesis outlines the development steps of our recommendation system and explores potential for future enhancements. This work holds value not only for tourists, who benefit from tailored recommendations, but also for industry stakeholders. For travelers, such systems simplify the decision-making process by quickly identifying destinations, activities, and services that align with their personal interests, saving time and effort (Jones, 2022).

In summary, integrating recommendation systems into tourism enhances both marketing effectiveness for service providers and the travel planning experience for consumers. This system aspires to make a meaningful impact on both travelers and the industry by enhancing the tourist experience in Istanbul, offering personalized suggestions, and achieving high accuracy (Thompson & Kim, 2023; Williams, 2022)

1.1 MOTIVATION

The tourism industry in vibrant cities like Istanbul faces significant challenge of serving an increasingly diverse and demanding influx of visitors. With a wealth of hotels, restaurants, and attractions available, tourists often struggle to navigate the overwhelming volume of information to find services that meet their specific preferences and needs. This overwhelming volume of choices frequently results in decision fatigue, reducing the overall quality of the travel experience. Despite the availability of many travel platforms, most fail to offer personalized and contextually relevant recommendations, leaving tourists without meaningful guidance in real time situations.

One of the underlying causes of this challenge is the inability of traditional systems to handle the cold start problem effectively. The cold start problem occurs when a system lacks sufficient data for new users or newly introduced services, making it difficult to generate relevant recommendations. In a city like Istanbul, where new attractions, restaurants, and accommodations are continually emerging, this problem is particularly prominent. As tourists often rely on prior user feedback to make decisions, the absence of such data for new or lesser known options leads to inefficiencies in the recommendation process.

This issue is compounded by the complexity and variability in tourist preferences, which evolve based on individual experiences, cultural backgrounds, and travel purposes. Existing recommendation systems struggle to accommodate these nuances, especially when they rely primarily on one dimensional data, such as user ratings or service categories. The inability to reflect the full spectrum of a tourist's interests or the specific features of new attractions amplifies the gap between available services and tourist expectations.

These challenges collectively highlight the need for more advanced, adaptable, and user recommendation mechanisms that can respond to the dynamic nature of tourism in bustling cities. The motivation for this project emerged from a recognition of these gaps in the current system and a desire to improve the way tourists navigate complex environments like Istanbul.

1.2 OBJECTIVE

Our goal in this thesis is to create a hybrid system that integrates user opinions with a machine learning based system to predict or recommend hotels, restaurants and attractions. We will then be able to offer significant competitive advantages in the tourism market by improving user experience, increasing customer loyalty, driving sales and providing strategic information for decision making.

This comprehensive solution to improve the visitor experience, by recommending the most relevant items and helping them personalize their journey. The final objective is to offer the tourist a minimally invasive support during his visits.

1.3 OUTLINE OF THE THESIS

This work is structured as follows. After the introduction in Chapter 1, Chapter 2 reviews existing literature and research on recommender systems. The first part explains the concept of recommender systems. The second part discusses different types of recommender systems: collaborative, content-based, and hybrid approaches. We compare different types of recommender systems and highlight the unique value proposition of our solution.

In Chapter 3, the research methodology begins with an introduction to our dataset, explaining the techniques used for data collection, rebalancing, and processing. We will also discuss the machine learning algorithms that we tested and how we selected the best model and optimized its parameters and hyperparameters. Subsequently, we will present the results of our models and demonstrate how we achieved optimal recommendations based on user opinions.

Finally, in Chapter 4, we will outline future directions for the research, including potential functionalities for our system and strategies for making our solution accessible to stakeholders. We will conclude by summarizing the improvements made and discussing potential future enhancements.

2 LITERATURE SURVEY

The rapid growth of digital platforms and the constant influx of users have necessitated the development of advanced recommendation systems, particularly in tourism, where tourists seek tailored suggestions for hotels, restaurants, and attractions (Suakanto, S., Andreswari, R., & Albasori, E. P., 2021). Traditional travel systems, while functional, often lack the personalization required to meet the diverse needs of users, which has led to an increasing reliance on collaborative filtering models (Fudholi, D. H., 2021). These models, built on the premise of analyzing historical behavior and preferences, have become the cornerstone of many recommendation systems (Zhang, S., Zhang, A., & Tay, Y., 2022).

Collaborative filtering can be broadly categorized into two main approaches: user-based and item-based. User-based filtering identifies users with similar preferences and recommends items based on those similarities (Chiang, J., 2022), while item-based filtering focuses on the relationships between items that users have interacted with, suggesting similar options (Havang, E., 2022). However, these approaches face inherent limitations, particularly in situations where data on new users or items is sparse, commonly referred to as the cold start problem (Abideen, Z., 2022). This challenge is prevalent in dynamic environments like tourism, where new services, attractions, or restaurants are regularly introduced without sufficient user ratings or feedback (Buriano, L., 2021).

To address these issues, hybrid recommendation systems that combine multiple techniques have gained prominence. For example, the COMPASS system integrates case-based reasoning (CBR), item-item filtering, and category learning to enhance recommendation accuracy and relevance (Lamsfus, C., Xiang, Z., Alzua-Sorzabal, A., & Martin, D., 2010). This system leverages contextual factors such as user profiles, preferences, and past interactions to offer tailored recommendations (Hatta, F., 2021). Hybrid models like this mitigate the cold start problem by incorporating both collaborative and content-based methods (Zain, M., 2022).

Content-based recommendation systems have also been extensively studied as a solution to the cold start issue. These systems analyze item attributes such as keywords, descriptions, and user reviews to recommend items that align with the user's past interactions and interests (Firdaws, J. M., & Urolagin, S., 2022). In the tourism domain, this approach proves particularly effective for recommending new or lesser known attractions by relying on specific attributes like location, service type, and ratings (Gupta, M., 2022).

Additionally, clustering techniques such as K-Means and DBSCAN have been utilized to group users or items based on shared characteristics, improving recommendation precision (Kzaz, L., 2021). These clustering methods are often employed as a preprocessing step before applying machine learning algorithms such as Support Vector Machines (SVM) and K-Nearest Neighbors (KNN), which are known for their robustness in classification and pattern recognition tasks (Banoula, M., & Ciudad, E. M., 2022). The use of such algorithms enables recommendation systems to effectively handle large datasets and make more accurate predictions (Chiang, J., 2022).

Furthermore, the application of deep learning techniques, particularly artificial neural networks (ANN), has revolutionized the field of recommender systems by allowing them to model complex patterns in user behavior and item attributes (Ishak, I., Zolkepli, M., & Ibrahim, H., 2021). ANNs, through multi-layer perceptrons (MLP), have demonstrated significant potential in enhancing recommendation accuracy, especially in tourism, where user preferences can be highly individualized and context-dependent (Fudholi, D. H., 2021).

2.1 RECOMMENDER SYSTEMS

Recommender systems help users make choices by suggesting products, services, or destinations based on their preferences. Drawing data from sources like social media, GPS logs, and geo-tagged images, these systems generate tailored suggestions that simplify decision-making. This personalization is essential in tourism, where travelers seek experiences that match their unique interests and patterns. Recommender systems use various algorithms and filtering techniques but face challenges, such as the "cold start" problem, which hinders recommendations for new users or items due to limited data. Advanced approaches, like hybrid and deep learning methods, address these limitations by adding layers of user-item interaction and content-based insights (Bug Bounty, 2024; Gupta, 2024).

Hybrid systems are particularly effective in tourism, especially in diverse destinations like Istanbul, as they integrate collaborative and content-based methods to improve recommendation accuracy. Neural networks, like Multi-Layer Perceptron (MLP), enhance hybrid models by capturing complex patterns in user behavior, providing more relevant suggestions.

2.2 RECOMMENDER SYSTEM TECHNIQUES

Recommender systems typically fall into three main categories which is shown in Figure 2.1 (Bug Bounty, 2024; Gupta, 2024).:

- **Collaborative Filtering**: This technique suggests items based on user preferences and behaviors of similar users. It's widely used with user-generated data and includes memory-based and model-based approaches (Suakanto et al., 2021).
- **Content-Based Filtering**: Here, recommendations depend on the user's preferences and item characteristics, like price or location, which is especially useful in tourism when there's enough item metadata (Pazzani & Billsus, 2007).
- **Hybrid Approach**: This combines collaborative and content-based methods to improve recommendations. For example, hybrid models can mitigate the cold start problem in collaborative filtering by incorporating item features (Zhang et al., 2022).

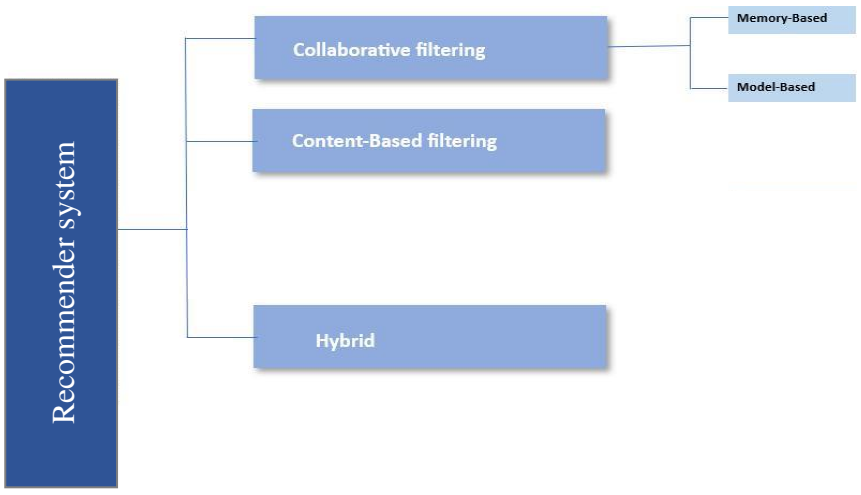


Figure 2.1 Recommender Systems Illustration

In general, machine learning models are used to build recommender systems, especially when multiple inputs are considered. Another commonly used technique is

cosine similarity, where text-based entries are converted into vectors to calculate similarity with other entries as shown in Figure 3.2 (Ayman, 2024).

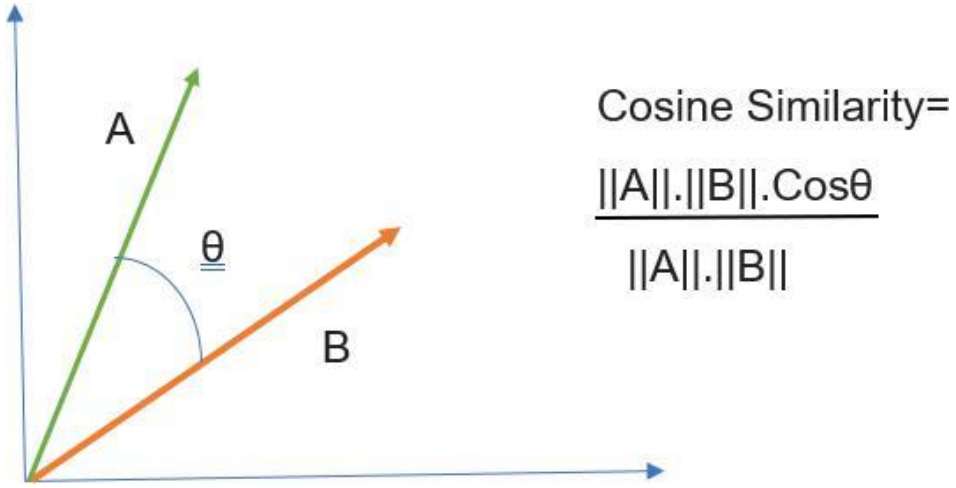


Figure 2.2 Cosine Similarity

2.2.1 Collaborative Filtering Approach

As shown in Figure 3.3, the collaborative filtering is a widely used method in recommender systems, particularly effective for high interaction platforms like tourism (Suakanto et al., 2021). This technique suggests items by analyzing similar users behaviors, based on the assumption that users who agreed on items in the past are likely to align in the future. Unlike content-based filtering, which focuses on item attributes, collaborative filtering is rooted in user-to-user relationships, leveraging data from past interactions, like ratings or browsing history, to predict future preferences (Fudholi, 2021).

Collaborative filtering methods are generally divided into two types: memory-based and model-based.

Memory-based filtering, or user-based filtering, identifies similar users through statistical measures, such as cosine similarity or Pearson correlation, and generates recommendations based on their preferences. However, this approach can struggle with scalability as datasets expand, as it requires real-time computation of user similarities.

Model-based collaborative filtering addresses these scalability issues by using machine learning techniques, such as matrix factorization and clustering, to uncover patterns in user behavior and make efficient predictions (Sharma & Singh, 2020).

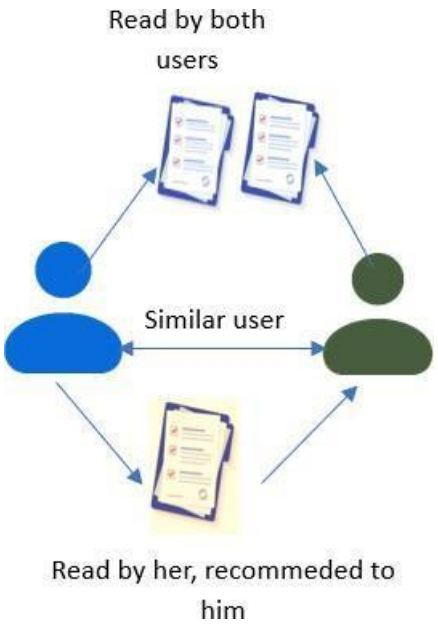


Figure 2.3 Collaborative Filtering Approach

2.2.2 Content-Based Filtering Approach

In content-based filtering, recommendations are generated by analyzing the attributes of the items themselves, rather than relying on user behavior. The system creates a profile for each user based on their past interactions and suggests items with similar attributes which is illustrated in Figure 2.4 (Ayman, 2024).

This method is particularly useful in domains where item attributes can be high defined, such as in hotel recommendations, where features like location, price, and amenities are crucial (Pazzani & Billsus, 2007).

Content-based filtering depends heavily on similarity metrics, such as cosine similarity, Euclidean distance, and Manhattan distance, to measure the closeness of items based on their features. In our case, Euclidean Distance was used to calculate the distance between two items in a multi-dimensional space, with each dimension representing a feature of the item.

The formula for Euclidean distance can be characterized as in Equation (1):

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \tag{1}$$

where p and q are two items, and i represents the different features being compared. This distance metric helps the system recommend items with attributes similar to those the user has previously interacted with (Raut, 2024).

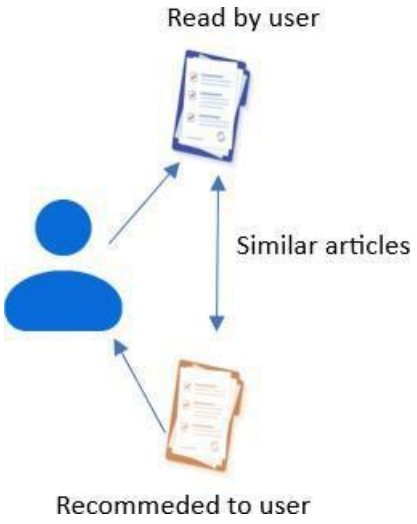


Figure 2.4 Content-Based Filtering Approach

2.2.3 Hybrid Approach

The hybrid recommendation system combines collaborative filtering and content-based filtering to enhance recommendation accuracy and personalization. Collaborative filtering suggests items based on the behaviors of users with similar preferences, while content-based filtering recommends items with attributes similar to those the user has engaged with them previously (Zhang et al., 2022).

This dual approach, shown in Figure 2.5, mitigates limitations in each method individually. For instance, collaborative filtering's "cold start" issue where new users or items lack sufficient data can be addressed by using content-based suggestions based on item characteristics. Additionally, content-based filtering's tendency toward overly similar recommendations can be offset by introducing collaborative filtering, thereby increasing recommendation diversity and robustness.

Overall, hybrid systems integrate these complementary methods to overcome challenges like data sparsity in collaborative filtering and lack of personalization in content-based filtering, achieving a more effective recommendation model.

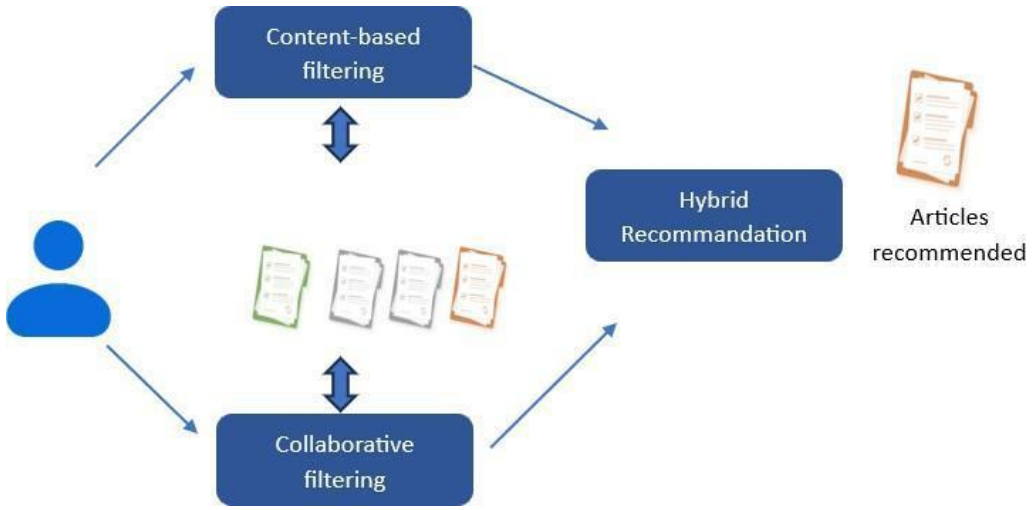


Figure 2.5 Hybrid Approach

The hybrid recommendation system combines collaborative filtering and content-based filtering to achieve better results than either method alone. This allows the system to use collaborative filtering to recommend items based on user behaviors while simultaneously using content-based filtering to recommend items with similar attributes to those previously interacted with (Zhang et al., 2022).

2.2.4 Content Based Filtering and Collaborative Filtering Comparison

Both content-based filtering and collaborative filtering are fundamental techniques in recommender systems, each with its own set of advantages and limitations. Their effectiveness can vary depending on the nature of the dataset and the specific application they are used for (Adomavicius & Tuzhilin, 2005).

Content-based filtering generates recommendations by analyzing the features of items that a user has previously engaged with. This makes it ideal for domains where item characteristics, such as hotel amenities or destination types, are high defined and structured. One key advantage of this approach is that it is user-independent, as it does not rely on the preferences or behavior of others (Pazzani & Billsus, 2007).

However, content-based methods can sometimes limit user discovery to only those items similar to what has already been experienced, which may reduce recommendation diversity (Raut, 2024). Furthermore, the method depends heavily on the availability of detailed metadata, which can be a challenge when such data is sparse or difficult to extract.

On the other hand, collaborative filtering bases recommendations on patterns in user behavior, drawing insights from the preferences of users with similar tastes (Sharma & Singh, 2020). This community-driven approach enables users to explore a wider variety of items, promoting serendipitous discovery beyond their usual interests (Gupta, 2024). Collaborative filtering, however, faces the cold start problem, particularly for new users or items with insufficient interaction data. Data sparsity in user-item interactions can also pose a significant hurdle, affecting the system's ability to generate reliable recommendations (Bug Bounty, 2024). Despite these challenges, collaborative filtering often scales effectively with large datasets, especially when using advanced model-based techniques. In summary, while content-based filtering provides a more tailored and transparent recommendation experience, collaborative filtering excels in introducing novel items to users by leveraging collective user preferences.

As illustrated in Table 2.1 (Bug Bounty, 2024)., which provides an extensive overview of the various techniques used in recommender systems, these methods span a range of algorithms, each tailored to address specific challenges and leverage different types of data and user interactions. Among these, collaborative filtering, content-based filtering, and hybrid approaches are the most widely recognized and implemented. Collaborative filtering relies on user-item interaction patterns and can be further categorized into memory-based and model-based methods. Memory-based techniques, such as user-based and item-based collaborative filtering, utilize similarity metrics like cosine similarity to identify relationships among users or items. Meanwhile, model-based approaches employ advanced techniques like matrix factorization or neural collaborative filtering to uncover latent features, enabling more scalable and accurate recommendations (Bug Bounty, 2024).

Table 2.1 Recommended Techniques for System Recommendation

Recommendation Approach	Commonly Used Techniques	Representative Research Examples
Collaborative Filtering	• User-Based Collaborative Filtering (Nearest Neighbor, Pearson correlation) • Item-Based Collaborative Filtering (cosine similarity) • Matrix Factorization (SVD, NMF) • Bayesian Networks (user preference probability models) • Graph-Based (social and co purchase networks) • Probabilistic Models (Markov chains, probabilistic latent semantic analysis)	• Resnick et al. (1994) • Breese et al. (1998) • Ungar & Foster (1998) • Sarwar et al. (2001) • Linden et al. (2003) • Schafer et al. (2001) • Cached et al. (2011) • Schein et al. (2002)
Content-Based Filtering	• TF-IDF (textual similarity from reviews or descriptions) • Euclidean Distance (for comparing numerical features, e.g., price, rating) • Bayesian Classifiers (to classify user preferences) • Decision Trees (for feature-based decision-making) • Artificial Neural Networks (feature extraction and non-linear mapping) • Semantic Analysis (extracting meaning from user generated content, such as travel blogs)	• Shardanand and Maes (1995) • Pazzani & Billsus (1997) • Mitchell (1997) • Mooney et al. (1998) • Billsus & Pazzani (1999) • Lops et al. (2011) • Davenport (2012)
Hybrid	• Weighted Hybrid Methods (combination of content and collaborative techniques) • Switching Methods (switch between content-based and collaborative depending on the situation) • Feature Augmentation (enriching collaborative filtering with item metadata) • Linear Combination of Predicted Ratings • Model Blending (merging multiple models to increase accuracy) • TOPSIS with Evolutionary Algorithm (tourism-specific optimization for attractions)	• Claypool et al. (1997) • Pazzani (1999) • Ricci et al. (2011) • Jung (2011) • Verhoeyen Lops et al. (2012) • Ayman (2024 - your thesis) • Burke (2002)

3 METHOD AND FINDINGS

3.1 APPROACH

The hybrid recommendation system for tourism utilizes both content-based and collaborative filtering, further enhanced by deep learning techniques to improve the accuracy of predictions. The process starts with user-generated ratings and reviews, which are clustered using two algorithms: K-Means for partitioning the data into groups based on similarity, and DBSCAN (Density-Based Spatial Clustering of Applications with Noise) to handle noise and outliers, ensuring robustness when dealing with clusters of varying shapes and sizes. Price information is also integrated into the clustering process for additional accuracy. After clustering, a deep learning model, specifically an Artificial Neural Network (ANN), is applied to refine predictions. The hybrid approach combines collaborative filtering, which identifies similar user interactions, with content-based filtering that leverages item features. The integration of deep learning with these filtering methods results in highly personalized recommendations, as visualized in Figure 3.1.

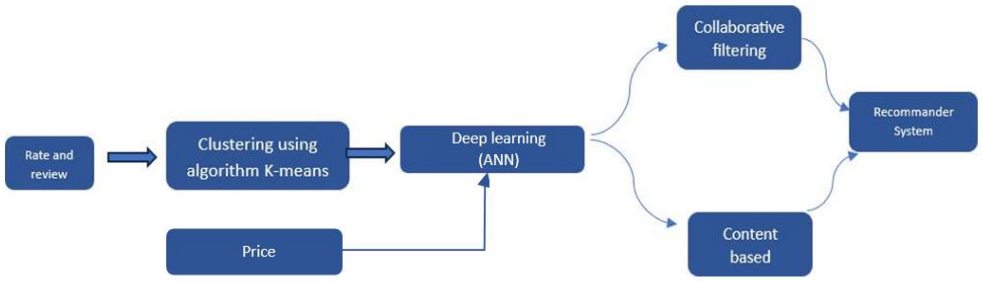


Figure 3.1 Steps of Work

3.2 DATASETS

Our dataset, which serves as the foundation for our recommendation system, was sourced by scraping Tripadvisor and contains 5125 rows and 13 columns. It includes key information about hotels, restaurants, and attractions in Istanbul, specifically capturing variables such as name, location, reviews, ratings, and prices for hotels, while restaurants and attractions are characterized by name, location, reviews, and ratings. This diverse dataset is critical for building a recommender system that provides personalized suggestions for tourists seeking accommodations, dining, or attractions.

The process of scraping data from TripAdvisor presented substantial challenges due to the platform's dynamic web structure and robust anti-scraping mechanisms. TripAdvisor employs advanced techniques to monitor and prevent large scale data scraping, particularly flagging and restricting suspicious activities. When unusual scraping patterns are detected, TripAdvisor enforces temporary blocks or bans, complicating efforts to collect comprehensive data in a short timeframe. To navigate these obstacles, we had to employ a cautious and incremental scraping approach, introducing delays between requests to avoid detection. Additionally, we used rotating proxies to obscure our activities and frequently encountered CAPTCHA challenges, which required manual intervention. As a result, the data collection process was slow and deliberate, taking several weeks to complete to ensure we captured a rich, diverse dataset without triggering bans or blocks.

The complexity of this process was further compounded by the fact that the data is specific to Istanbul. Pre-existing datasets on platforms like Kaggle or similar sources did not exist, forcing us to manually extract each piece of data. This absence of available data made the task labor intensive, requiring meticulous attention to detail and prolonged efforts to ensure the dataset was comprehensive. However, the challenges were necessary, as scraping custom data was essential for developing a robust recommendation system tailored to the specific context of Istanbul's tourism industry.

Once the data was successfully scraped, it was formatted into a CSV file to facilitate easy access and usability in subsequent analysis stages.

This dataset will be crucial in supporting the system's ability to deliver tailored suggestions, making it an integral part of the system's success in guiding tourists through their travel experience in Istanbul which is shown in Figure 3.2.

	hotel_name	h_reviews	h_location	h_rate	h_price	restaurant_name	r_rev	r_adress	r_rate	Attraction_name	at_Review	at_Location	at_Rate
0	Sarikonaklar Garden Village	1 review	Sari Konaklar Piknik Sokak NO 10 Kabakoz, Ista...	5.0	$124	Sadrazam Mahmut	28	Kumbarhane Cad. No:64 Halicioglu, Istanbul Tur...	4.5	Hagia Sophia Mosque	45093	Sultan Ahmet, Ayasofya Meydani No:1, Istanbul ...	4.5
1	Venus Hotel Taksim	7 reviews	Tarlabasi Bulvari No 75, Istanbul 34435 Turkiye	3.5	$56	Yarimada Balik	13	Arap Cami Mahallesi, Kurekciler Kapisi Sk., Or...	4.5	Basilica Cistern	32068	Alemdar Mahallesi, Yerebatan Caddesi, No:1/3, ...	4.5
2	Talen Otel	4 reviews	Erol Kaya Cd No:239 Kaynarca Mahallesi, Istan...	3.5	$82	Fasuli Lokantalari	51	Hocapasa Mah. Muradiye Cad. No:35 Fatih Eminon...	4.0	Blue Mosque	35255	Sultanahmet Mah At Meydani Cad No 7, Istanbul ...	4.5
3	Fanus Suites Karakoy	4 reviews	Fermeneciler Caddesi No 11, Istanbul 34425 Tur...	3.5	$119	Draft Gastro Pub	42	Bagdat Caddesi, No:349, Istanbul 34710 Turkiye	4.0	Topkapi Palace	28081	Cankurtaran Mahallesi Gulhane Park, Near Sulta...	4.5
4	Downtown Sirkeci Hotel	2 reviews	Kargili cikmazi No:6, Istanbul 34110 Turkiye	3.5	$81	GRACE Rooftop Restaurant	45	Binbirdirek mah. Terzihane sok:15/6 Sultanahme...	5.0	Sultanahmet District	16800	NaN	4.5
...	...	...	...	...	...	...	...	...	...	...	...	...	...
5121	Lotte Palace Hotel	16 reviews	Merkez Mahe-5 Yanyol Cad No131 Avcilar, Istanb...	4.0	$97	Ali Haydar Usta	20	Sirinevler Mah. Karaoglanoglu Cad. No: 41 Bahc...	4.5	Turkey Data eSIM 0.5GB daily to 50GB 30 Days	NaN	,	NaN
5122	The Tango Hotel	31 reviews	Dolapdere Cad. No: 213 Sisli, Osmanbey, Istanb...	3.5	$96	Green Salads	22	19 Mayis Mahallesi, Buyukdere Cad. No: 136/22 ...	4.5	TouringBee - Istanbul	NaN	,Istanbul, Turkiye	NaN
5123	Garan Apartments	14 reviews	Cebel Toru St. Cumhuriyet Av. Elmadag / Sisli,...	4.0	NaN	The Hunger Piazza AVM	19	Cevizli Mahallesi Tugay Yolu Cd No:77 Piazza A...	4.0	Crimean Memorial Church	NaN	,	NaN
5124	Hotel Park	27 reviews	Utangac Sok. No: 26 Sultanahmet, Fatih, Istanb...	4.0	NaN	Gunaydin	49	Nispetiye Cad. Seher Yildizi Sok. No:6 Etiler,...	4.0	Bit Pazari Kadikoy	NaN	Hasanpasa, Uzuncayir Cd, Istanbul 34722 Turkiye,	NaN
5125	AYGUNES Hotel Taksim	73 reviews	Sehit Muhtar Mah. Suslu Saksi Sk. No:26 10 Met...	3.5	NaN	22 Cihangir	12	Cihangir Caddesi No 35, Istanbul 34427 Turkiye	4.5	Yon Vip Travel	NaN	Eyup Sultan, Kizilay Cd. No:49a, 34885, Istanb...	NaN

Figure 3.2 Dataset

Table 3.1 outlines the specific descriptions of each variable used in the dataset. The variable hotel_name refers to the name of each hotel, while h_location describes its geographic location. h_reviews are the user reviews, where users can vote or provide feedback about their stay, while h_rate refers to the overall rating a hotel receives, and h_price represents the price for a stay.

Similarly, for restaurants, restaurant_name indicates the name of the restaurant, r_rev contains user reviews, r_rate indicates the rating, and r_address refers to the location.

In terms of attractions, the variable attraction_name refers to the specific name of the tourist attraction, providing a unique identifier for each site. at_review contains user feedback on the attraction, offering insights into the experiences of visitors and helping potential tourists decide which sites to visit.

at_rate is the numerical rating assigned to the attraction, summarizing its overall quality or appeal. Finally, at_location refers to the geographic details of the attraction, allowing for location-based filtering.

Each of these variables plays a critical role in shaping the recommendation system, as they provide the foundational data needed to assess user preferences, behavior, and patterns of interaction with hotels, restaurants, and attractions. By analyzing these variables, the system can offer more accurate and personalized recommendations, thus enhancing the user experience. Table 3.1 serves as a detailed reference that supports the system's functionality, ensuring that every aspect of user interaction is taken into account in generating relevant suggestions.

Table 3.1 Variable Description

Variable name	Description
hotel_name	Name of hotel
h_location	Hotel location
h_reviews	The reviews that a user can vote on the hotel
h_rate	Rate of a user about hotel
h_price	Hotel price
restaurant_name	The name of each restaurant
r_rev	The reviews that a user can vote on the restaurant
r_rate	Rate of a user about restaurant
r_adress	Restaurant location
attraction_name	The name of each attraction
at_review	The reviews that a user can vote on the attraction
at_rate	Rate of a user about attraction
at_location	Attraction location

3.3 DATA PRE-PROCESSING

Data pre-processing is a crucial step in preparing raw data for analysis and model training. In our project, the dataset scraped from TripAdvisor required significant cleaning and transformation before it could be used effectively in our recommendation system.

The raw data contained inconsistencies in the formatting of numerical values, mixed data types, and irrelevant text, all of which needed to be addressed to ensure the quality of our predictions. The main objectives of pre-processing were to standardize the data, convert it into a usable format, and address missing or incorrect values.

3.3.1 Handling Text in Numerical Columns

The first step in our data pre-processing was to clean the numerical columns that had unwanted text strings, such as the word "reviews" or "review" within the review columns. For example, both the hotel reviews column (h_reviews) and the attraction review column (at_Review) included the word "review(s)" after the actual numerical value, which needed to be removed. This cleanup ensured that only numerical data remained, which is essential for any statistical analysis or model training. By eliminating these non-numeric elements, we prepared the data for conversion into numeric types, making it easier to manipulate and analyze.

3.3.2 Standardizing Number Formats

Next, we encountered issues with the formatting of numbers, particularly the usage of commas (",") and periods (".") for decimal points, which varied throughout the dataset. To standardize the format, we replaced commas with periods in price and review columns. This step was particularly important for ensuring consistency in numerical representations, as variations in the decimal format can lead to incorrect interpretations during analysis. Additionally, the "h_price" column contained dollar signs ("$") indicating the price of hotels. We removed these symbols to retain only the numerical values, facilitating their transformation into a usable format for further analysis.

3.3.3 Converting Data to Numeric Types

Once we had cleaned the textual content from the numerical columns and standardized the number formats, we converted the relevant columns into numeric data types. This transformation was essential to perform mathematical operations and statistical computations. For this, we applied the pd.to_numeric() function, which allowed us to convert the values while handling potential errors by replacing them with NaN (Not a Number) for invalid entries. This method provided a reliable way to cleanly manage erroneous data without causing disruptions in our analysis process.

3.3.4 Handling Missing Values

Another critical aspect of pre-processing involved addressing missing or null values. Instead of discarding rows with missing data, which could result in a significant loss of information, we opted to fill these gaps by replacing null values with the mean of each column. This method is a common and effective approach to handle missing data without introducing bias, ensuring that the dataset remains robust and complete for subsequent model training. We applied this technique to all numerical columns, ensuring that our dataset was free of missing values and ready for further analysis.

3.3.5 Removing Duplicates

Duplicate entries can skew analysis and lead to biased results in model predictions, so it was necessary to identify and remove any redundant rows. Using the df.duplicated() function, we checked for and eliminated duplicate entries within the dataset. Although the number of duplicates was relatively low, their removal helped maintain the integrity of the dataset, ensuring that each entry represented a unique instance of user interaction or item.

3.3.6 Cleaning and Standardizing Address Data

Location data for hotels, restaurants, and attractions needed cleaning due to inconsistencies like extra spaces, special characters, and varied capitalization. We applied a custom function to standardize addresses by removing extraneous spaces, newlines, special symbols, and converting all text to lowercase. This step was essential for consistent, accurate location matching, especially for content-based filtering where precise location data enhances recommendation relevance.

3.3.7 Final Data Type Conversion

To finalize the data preprocessing, we converted specific columns to the correct data types, such as converting h_reviews and h_price to floats for numerical precision.

This step completed the standardization process, resulting in a structured and reliable dataset ready for collaborative and content-based filtering in our recommendation models. This comprehensive cleaning pipeline transformed raw data into a usable foundation for building an effective recommendation system.

3.3.8 Exploratory Data Analysis (EDA) and Visualization

Exploratory Data Analysis (EDA) and visualization are fundamental components of any data driven thesis, including our thesis. EDA involves examining and analyzing datasets to summarize their main characteristics, often with visual representations. Visualization, in particular, allows us to translate complex datasets into more digestible formats like scatter plots, bar charts, and histograms. These graphical tools are crucial for understanding relationships, trends, and distributions within the data. For instance, in this thesis, we created scatter plots to visualize the relationship between variables like hotel reviews and hotel ratings. Such visualizations provide a clearer picture of the data's structure, helping us identify patterns, outliers, and potential correlations that may not be immediately visible from the raw dataset. By doing so, visualization enhances our ability to make informed decisions about data preprocessing and model selection, ultimately improving the accuracy of our recommendation system.

3.3.8.1 Hotels Data Distribution

The scatter plot visualizes the relationship between **the number of reviews (h_reviews)** and **the ratings (h_rate)** for hotels and restaurants, aiming to segment these establishments into five distinct groups based on their reviews and ratings. The X-axis represents the number of reviews received, ranging from 0 to approximately 1000, which indicates customer engagement or popularity. The Y-axis represents the ratings, on a scale from 1 to 5, where a higher rating reflects better customer satisfaction. The plot shows distinct horizontal bands, corresponding to the different rating levels (1 to 5), with a visible concentration of points in the higher rating bands (4 to 5), suggesting many establishments receive favorable reviews, though their review count varies widely. For instance, hotels/restaurants with ratings between 4.5 and 5 are spread across a range of review counts, while fewer points are present in the lower-rated categories. The plot is used in a **collaborative filtering** system to create group specific recommendations based on both rating and review count.

By segmenting hotels and restaurants, the recommender system can direct users toward venues that match their preferences, such as high rated, highly reviewed establishments, or those that are less popular but still highly rated as shown in Figure 3.3

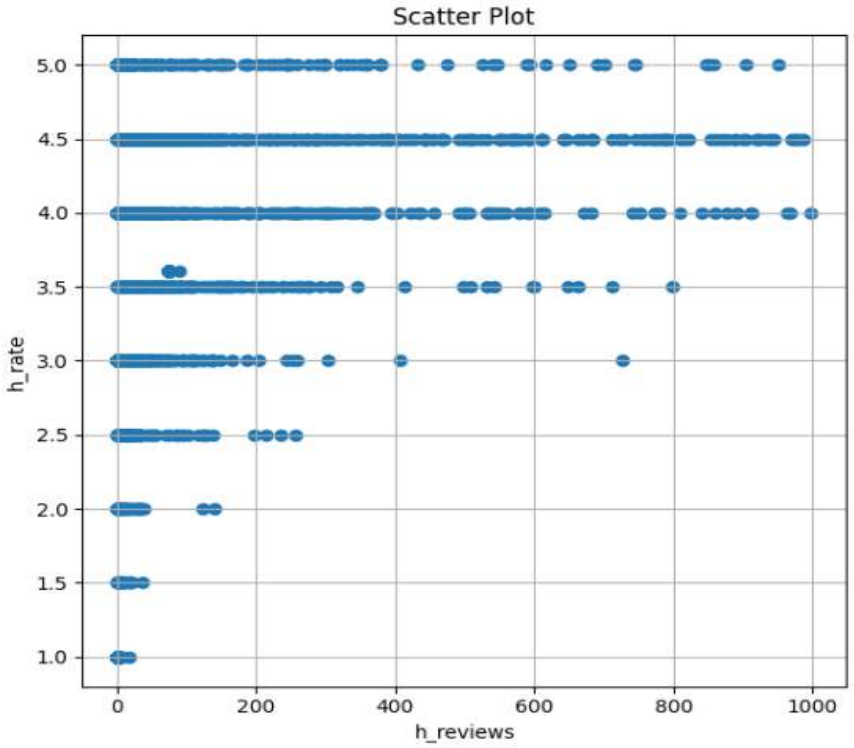


Figure 3.3 Visualization of Hotels Data Distribution

3.3.8.2 Restaurants Data Distribution

In the scatter plot representing restaurant reviews versus restaurant ratings (r_rev vs r_rate), the x-axis reflects the number of reviews a restaurant has received, ranging from 0 to over 4,000. The y-axis ranges from 3.0 to 5.0, representing the restaurant ratings. Restaurants generally fall within a narrower rating range, as most establishments tend to receive ratings of 3 or higher. We avoided extending the y-axis below 3 since no restaurant in the dataset had a lower rating, and extending the range would have led to unnecessary empty space on the plot. This scatter plot helps identify patterns in how the number of reviews correlates with the rating. Restaurants with a higher number of reviews tend to have higher average ratings, as illustrated in Figure 3.4.

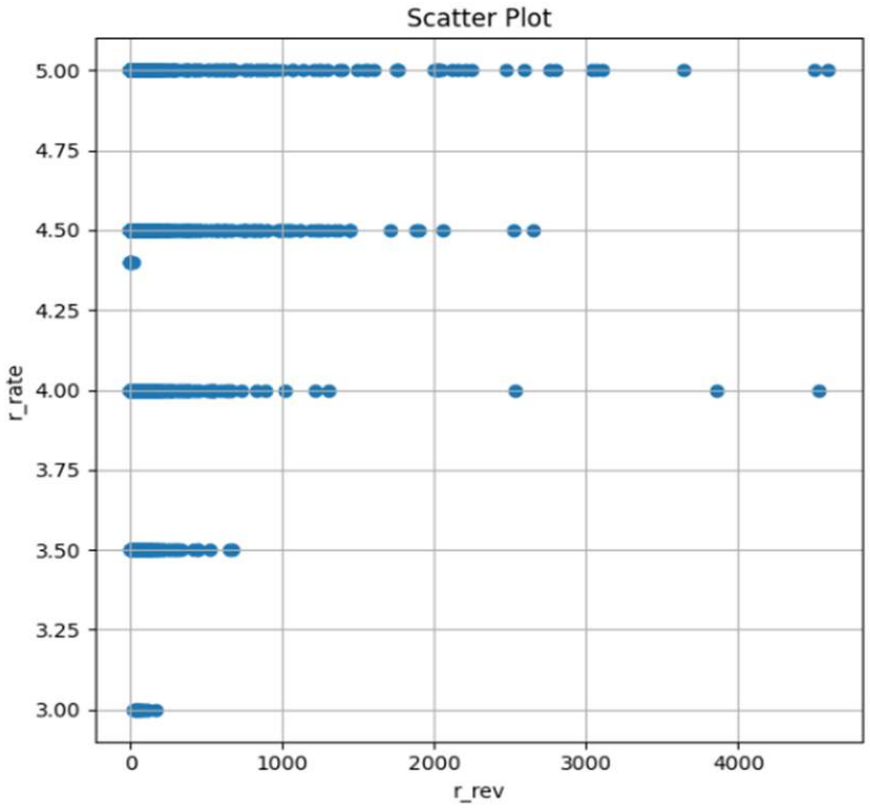


Figure 3.4 Visualization of Restaurants Data Distribution

3.3.8.3 Attractions Data Distribution

The scatter plot for attractions illustrates the relationship between the number of reviews (at_Review) and the corresponding ratings (at_Rate) for different tourist attractions. The x-axis represents the number of reviews, indicating how many users have provided feedback for each attraction, while the y-axis shows the ratings, reflecting the users overall satisfaction or dissatisfaction with the attraction. In the scatter plot representing attraction reviews versus attraction ratings (at_Review vs at_Rate), the x-axis denotes the number of reviews that an attraction has received, ranging from 0 to approximately 40,000. This wide range was chosen to accommodate the significant disparity in popularity among different attractions, as some attract very few visitors while others receive a substantial number of reviews. The y-axis represents the rating for each attraction, ranging from 0 to 5, which is the standard rating scale. This plot enables us to visually explore the correlation between the number of reviews and the rating, indicating whether the popularity of an attraction (in terms of review count) impacts its rating, as shown in Figure 3.5.

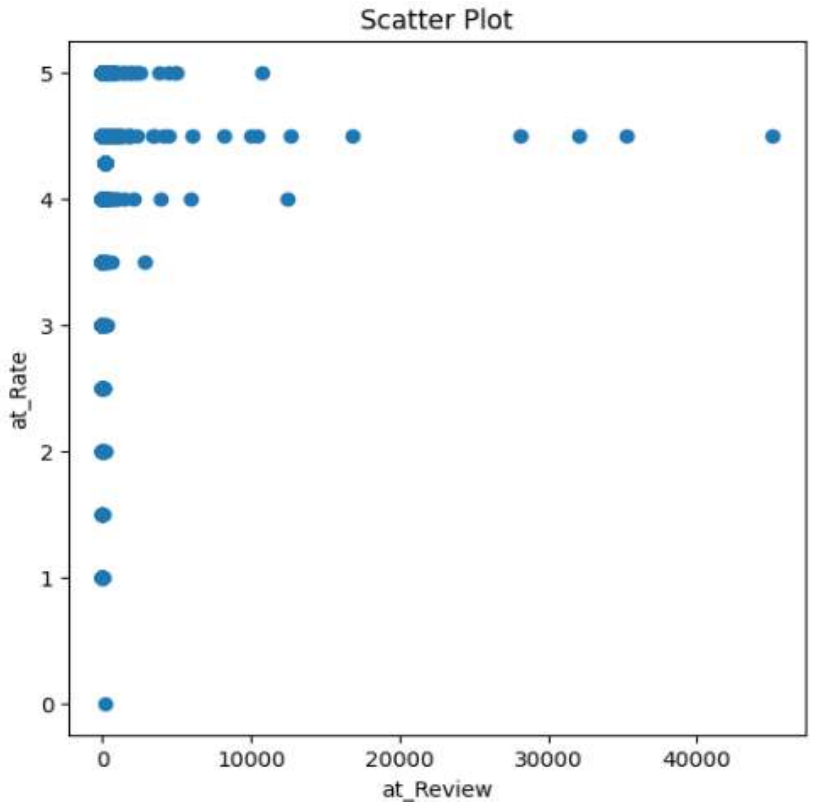


Figure 3.5 Visualization of Attractions Data Distribution

3.3.9 Drop Unnecessary Lines

In any real-world dataset, missing or incomplete information is a common issue that can significantly impact the performance of machine learning models. In our dataset, scraped from TripAdvisor, we encountered several instances of missing values, particularly in columns such as hotel prices, reviews, and ratings. To address this, we applied the dropna() function to remove rows where critical information was missing. The rationale behind dropping rows with incomplete data is to ensure that the remaining dataset is clean, consistent, and ready for analysis. Missing values could introduce bias into our predictive models, skewing the recommendations generated by our system. By removing rows with NaN values, we maintain the integrity of our data and reduce the likelihood of errors during training and prediction phases.

In this thesis, dropping unnecessary lines from the dataset was an essential step to ensure the accuracy and reliability of our recommendation system. Given that our analysis relies heavily on user ratings and reviews, missing values in these fields could distort the patterns we aim to detect through collaborative and content-based filtering techniques. Furthermore, incomplete data could lead to performance issues, such as errors during model fitting or misinterpretation of the results.

By using the dropna() method, we effectively minimized such risks, ensuring that our machine learning models are trained on high-quality, complete data, which in turn improves the overall performance and accuracy of our tourism recommendation system.

3.4 MACHINE LEARNING ALGORITHMS

3.4.1 Unsupervised Learning Clustering

Unsupervised learning algorithms are used on data without labeled responses to identify hidden structures, patterns, or groupings. Clustering as shown in Figure 3.6 (Adapted from Jain et al., 1999), is an essential technique in unsupervised learning, groups objects so that items in the same cluster exhibit greater similarity to each other than to those in other clusters. By organizing data into clusters, we uncover inherent patterns that might not be immediately apparent, enabling deeper insights into complex datasets.

In tourism, clustering plays a crucial role in segmenting hotels, restaurants, and attractions based on user reviews and ratings. This segmentation aids in developing targeted recommendation systems, ensuring that travelers receive suggestions tailored to their preferences. Various clustering methods, such as hierarchical clustering, DBSCAN, and K-means, each have specific strengths and applications depending on the data's nature and complexity (Xu & Wunsch, 2005).

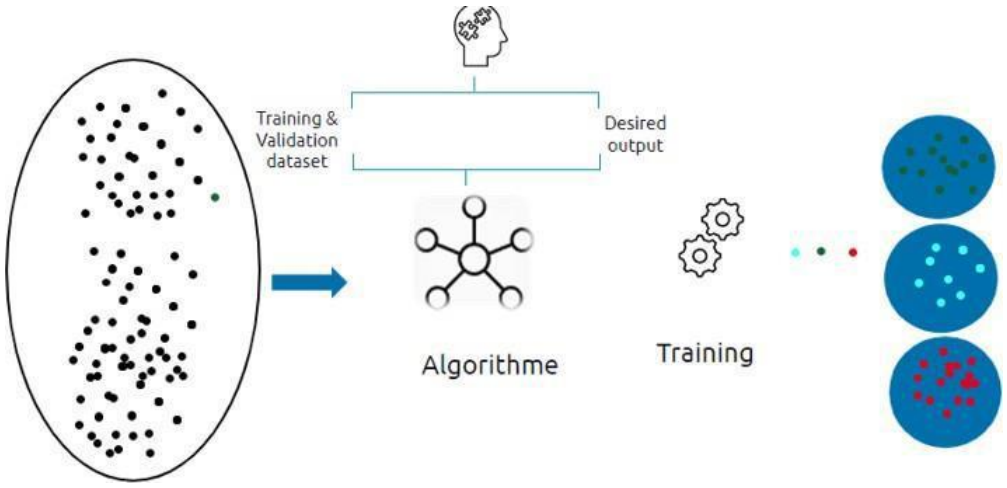


Figure 3.6 Clustering (Adapted from Jain et al., 1999)

3.4.1.1 K-Means Clustering

K-means is a widely used and efficient clustering method that works by partitioning the data into k clusters, where each data point belongs to the cluster with the nearest mean. It is computationally efficient for large datasets and relatively easy to implement and interpret, which makes it highly suitable for the dataset used in our thesis (Bholowalia & Kumar, 2014). Given that we are dealing with numeric values (such as user reviews and ratings), K-means can group similar hotels, restaurants, or attractions together based on their proximity in the feature space, which aligns with our goal of making tailored recommendations (Yuan et al., 2021). Additionally, K-means offers flexibility in choosing the number of clusters, which is crucial for a more refined recommendation model as shown in Figure 3.7 (Statology, 2023).

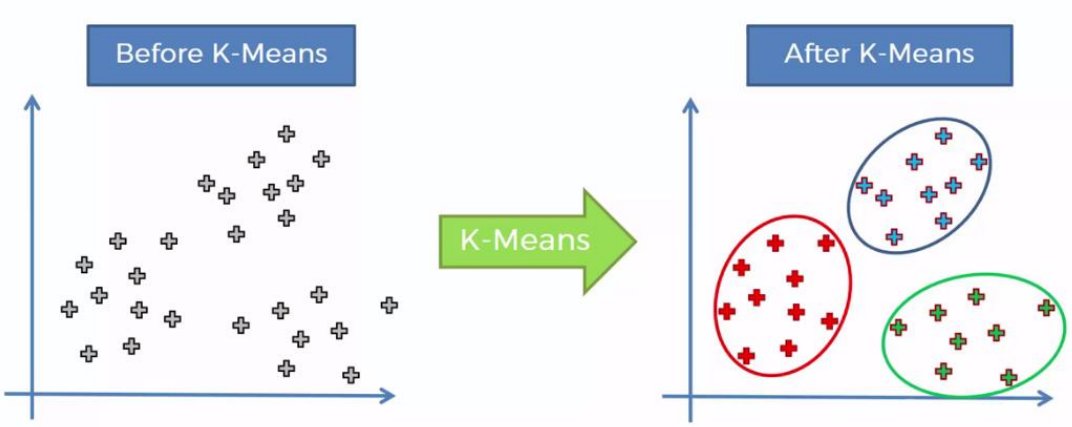


Figure 3.7 K-Means Algorithm Illustration

The algorithm begins by randomly selecting k points as initial cluster centroids. Each data point is then assigned to the nearest centroid, forming clusters. Next, the centroids are recalculated by finding the mean of all data points in each cluster. This process of assigning points and recalculating centroids repeats until the cluster assignments stabilize, meaning there's minimal change in the centroids. The goal is to minimize the Within-Cluster Sum of Squares (WCSS), which measures the variance within each cluster, resulting in compact and well separated clusters (Stack Abuse, 2023; Analytics Vidhya, 2023).

To determine the optimal number of clusters, we have employed the Elbow Method, a well known technique to select the most appropriate number of clusters by measuring the WCSS, which indicates cluster compactness (Cheng, 2021). As the number of clusters increases, the WCSS decreases because each cluster becomes smaller, but beyond a certain point, adding more clusters brings diminishing returns. In the Elbow Method, we plot the WCSS against the number of clusters and look for the "elbow" point, where the rate of decrease sharply slows down. This point provides a balance between the number of clusters and model efficiency. For our thesis, this method ensures an optimal number of clusters, balancing model performance and interpretability, thus creating meaningful segmentation in our dataset as shown in Figure 3.8 (Statology, 2023; Cheng, 2021).

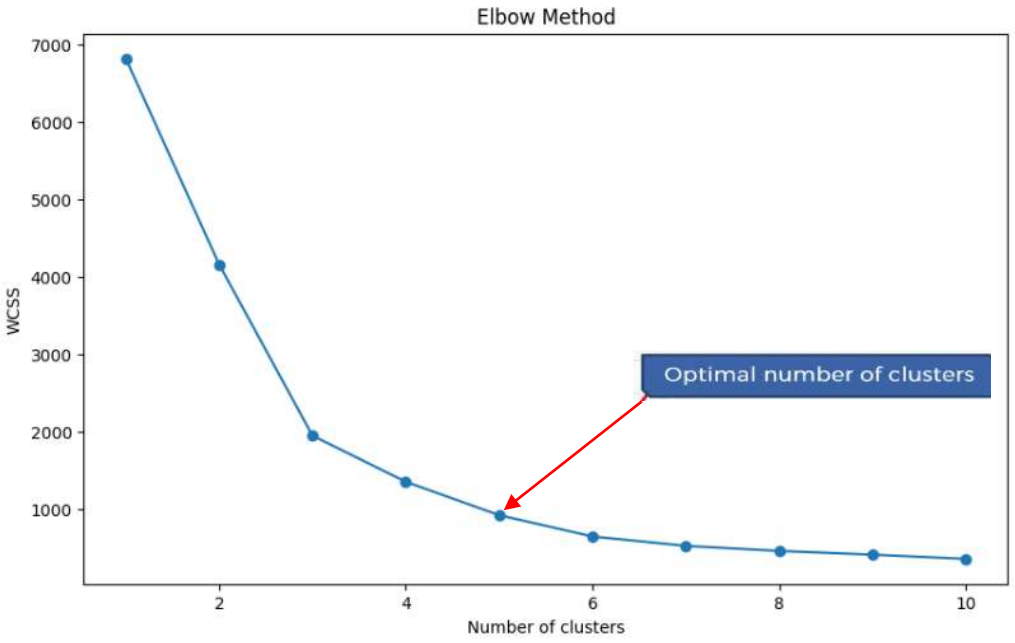


Figure 3.8 Elbow Method for optimal value of k in K-Means

Clustering is a fundamental technique in unsupervised machine learning that organizes a set of objects into groups, or clusters, where items within the same cluster exhibit greater similarity to one another compared to those in different clusters. This section demonstrates the application of previously defined clustering techniques on datasets related to hotels, restaurants, and tourist attractions.

Hotel Clustering

As it is mentioned on Figure 3.9, K-Means clustering is performed on a dataset to categorize hotels based on two features: the number of reviews and the rating scores.

This method is chosen due to its effectiveness in handling large datasets and its ability to produce distinct, easily interpretable clusters, which is particularly beneficial in identifying patterns in customer preferences and behaviors. It begins by defining five clusters and fitting the model to the specified features, assigning each hotel to one of the clusters. This clustering result is then integrated into the dataset as a new column that indicates the cluster membership, where the x-axis represents the number of reviews and the y-axis denotes the hotel ratings. Each point on the plot is color coded according to its assigned cluster, utilizing a color palette to enhance differentiation. The final visualization is titled "Hotel clustering" and presented for analysis, allowing for an intuitive understanding of how hotels are grouped based on their review counts and ratings.

The choice of K-Means over other methods, such as hierarchical clustering or DBSCAN, stems from its computational efficiency and scalability.

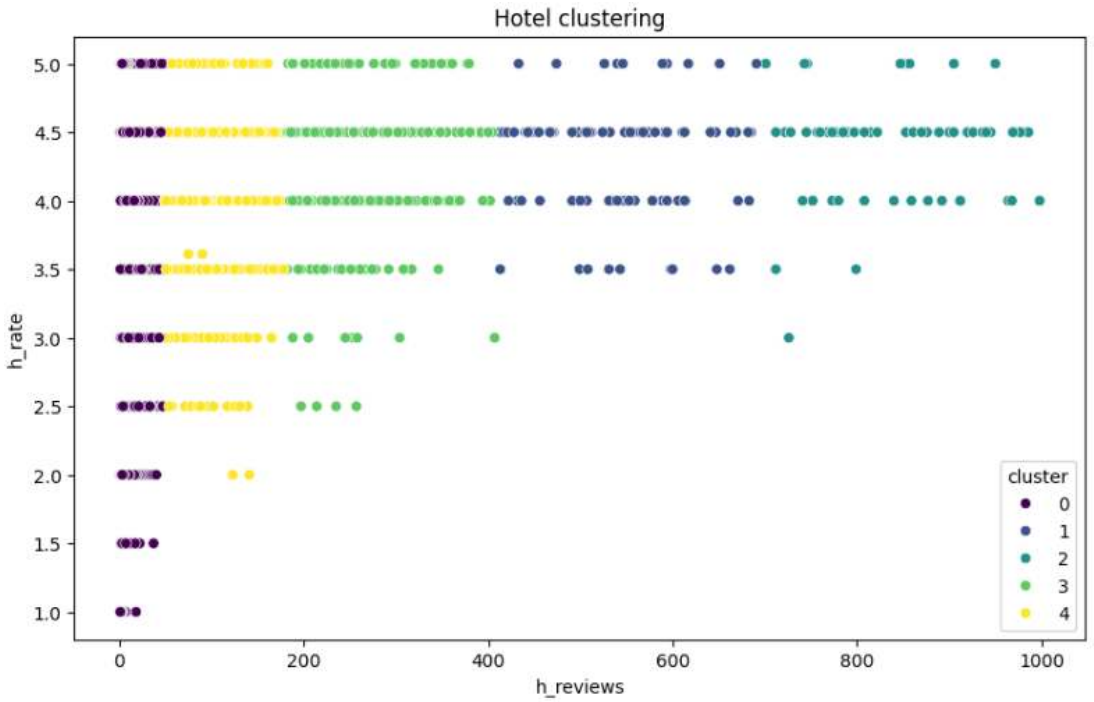


Figure 3.9 Hotel Clustering Using K-Means Algorithm Method

Restaurant Clustering

The scatter plot presented in Figure 3.10, illustrates the clustering of restaurant data based on the number of reviews (r_rev) and ratings (r_rate). By applying the K-means clustering algorithm with five clusters, each restaurant is assigned to one of the clusters (0 to 4), as indicated by different colors in the plot.

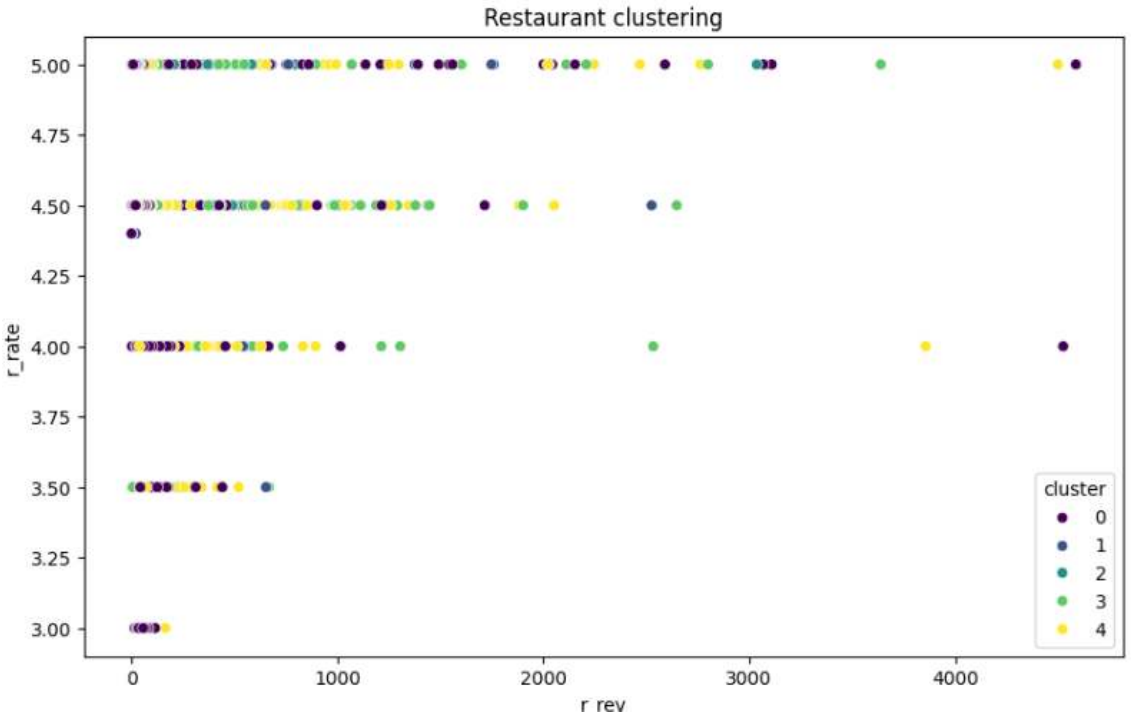


Figure 3.10 Restaurant Clustering Using K-Means Algorithm Method

The x-axis represents the number of reviews, which varies across a wide range, while the y-axis represents ratings on a scale from 3.0 to 5.0. This clustering approach groups restaurants with similar characteristics, such as those with high reviews and high ratings or low reviews with varying ratings. The goal of this visualization is to identify meaningful patterns in the restaurant data, making it easier to analyze clusters with similar review counts and ratings.

3.4.1.2 Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

DBSCAN is a clustering algorithm that identifies clusters based on data density, rather than distances between points as in K-means. It is particularly effective for datasets with noise and arbitrary cluster shapes. The algorithm relies on two key parameters: Eps (the maximum distance between two points to be considered part of the same neighborhood) and MinPts (the minimum number of points required to form a dense region). DBSCAN begins by selecting an unvisited point and checking if it forms a dense cluster based on Eps and MinPts. If the point meets these criteria, it becomes a "core point" and expands to include all directly connected neighbors. This process repeats until all points in the dataset are either assigned to a cluster or marked as noise.

One major advantage of DBSCAN is its ability to find clusters of varying shapes and filter out noise points, making it suitable for spatial data analysis and applications in big data clustering. Unlike traditional clustering methods, DBSCAN does not require specifying the number of clusters in advance, which can be advantageous for exploratory data analysis in tourism and other fields as shown in Figure 3.11 (Ester et al., 1996; IEEE, 2024).

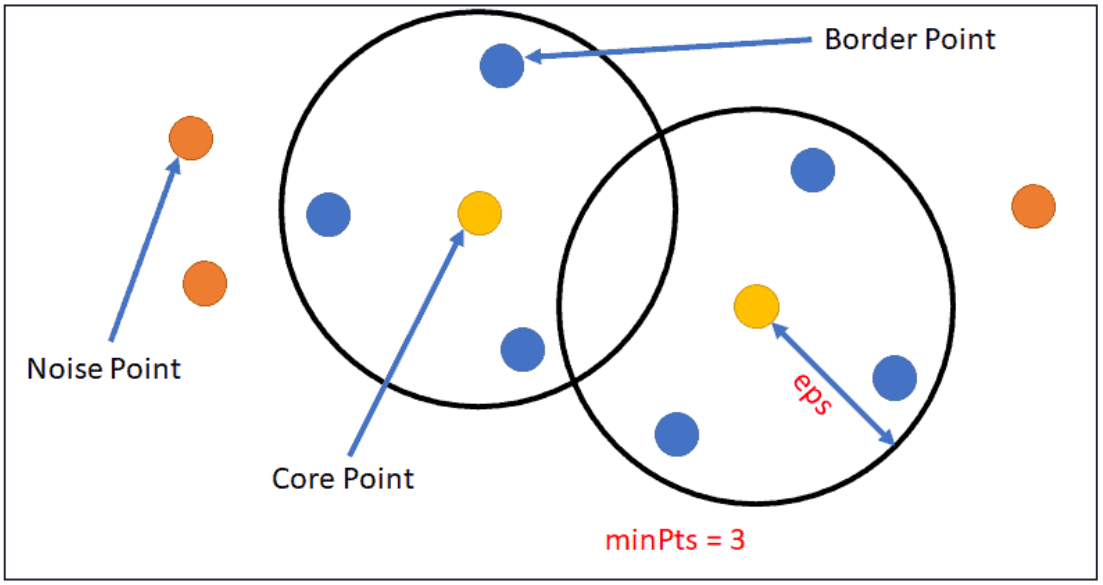


Figure 3.11 DBSCAN Clustering Algorithm Illustration

Attractions Clustering

In Figure 3.12, the data points representing different attractions are visualized based on the attributes "at_Review" (number of reviews) and "at_Rate" (rating). Using the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm, this clustering method groups attractions into clusters based on density. Here, DBSCAN identifies clusters by evaluating points within a defined distance (epsilon, eps=0.5) and requires each core point to have a minimum number of neighbors (min_samples=10).

Data points that do not meet these density criteria are labeled as noise, which are not assigned to any cluster. As shown in Figure 3.12, the clusters are represented by various colors, while isolated points may represent noise in the dataset.

In case of the use of StandardScaler, the data is transformed by centering it around zero with a standard deviation of one by this scaler (StandardScaler). This standardization can result in negative values, especially if the original data points are below the mean. Consequently, the scaled data may have values that span negative to positive.

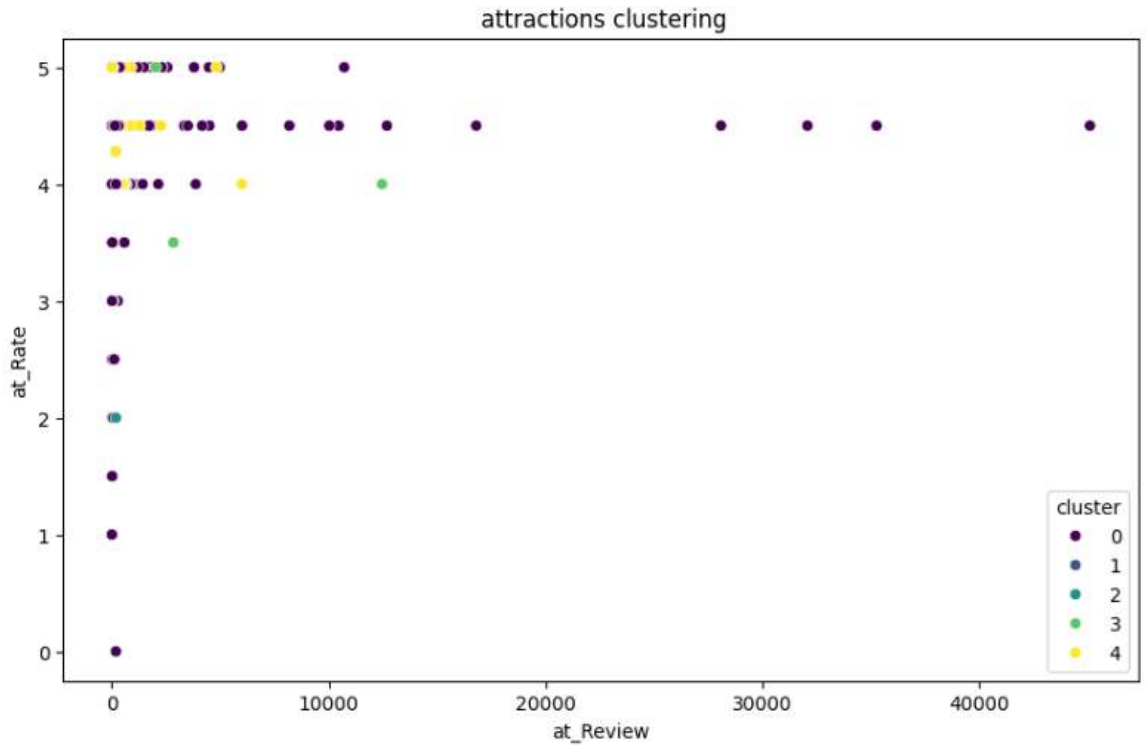


Figure 3.12 Attraction Clustering Using DBSCAN Algorithm Method

3.4.1.3 Difference between DBSCAN and K-Means

DBSCAN is chosen for attractions because it is adaptable to datasets with noise and varying densities, which is often the case with tourist attractions.

Attractions may have highly diverse review counts and ratings, with some points being isolated outliers (low review or low rating attractions). DBSCAN can identify core clusters and separate noise points without forcing every data point into a cluster, making it ideal for the sparsely populated, varied data of attractions. On the other hand, K-means is applied to hotels and restaurants because these datasets tend to have more consistently distributed data, where K-means can effectively group similar points based on centroid distances. K-means is computationally efficient and performs well when clusters are roughly spherical and similarly sized, conditions that are generally met in the hotel and restaurant data used in this project.

Using DBSCAN and K-means where they are most appropriate improves the clustering quality and allows the model to adapt to the unique characteristics of each type of entity in the recommendation system.

3.4.2 Supervised Learning

Supervised learning is a machine learning paradigm that involves training a model on a labeled dataset, where each input is paired with a corresponding output. This approach allows the model to learn the underlying relationship between the features and the target variable, enabling it to make accurate predictions on unseen data. Supervised learning is widely utilized in various applications, including finance, healthcare, and tourism, due to its ability to provide interpretable results and high accuracy. In the context of this research, we employ supervised learning to predict hotel ratings based on user reviews and other relevant features. The primary algorithms used in supervised learning include Gradient Boosting, Logistic Regression, K-Nearest Neighbors (KNN), and Support Vector Machines (SVM). Each of these algorithms has unique characteristics and strengths, making them suitable for different types of data and prediction tasks. The choice of algorithm depends on factors such as the nature of the dataset, the desired interpretability of the model, and the specific requirements of the predictive task.

These models are extensively documented in the literature, providing a robust foundation for their application in this study (Hastie, Tibshirani, & Friedman, 2009; James et al., 2013).

3.4.2.1 Gradient Boosting

The Gradient Boosting model is an ensemble learning technique that combines the predictions of several weak models to produce a stronger overall prediction. It builds the model iteratively, where each new model is trained to correct the errors made by the previous ones.

This approach allows Gradient Boosting to capture complex relationships in the data, making it particularly effective for regression tasks, such as predicting hotel ratings. By minimizing the loss function through gradient descent, this method enhances accuracy and robustness, resulting in improved performance on test datasets. Its adaptability to various types of data and ability to handle overfitting through regularization techniques further solidify its position as a preferred method in predictive analytics (Friedman, 2001; Chen & Guestrin, 2016).

In the context of Gradient Boosting, weak models typically refer to decision trees that are shallow and simple, often called "stumps." These models are termed weak because they do not capture complex patterns in the data on their own; instead, they are combined to form a stronger predictive model through an ensemble approach. Below are some commonly used weak models in Gradient Boosting, particularly focusing on decision trees and their characteristics:

1. **Decision Trees**:
    - **Description**: A decision tree is a flowchart-like structure that makes decisions based on a series of questions about the features of the data. It splits the data into subsets based on feature values, creating branches that lead to final predictions (leaves).
    - **Example**: A simple decision tree might predict whether a hotel will receive a high rating based on features like price, location, and amenities. For instance, if the price is below a certain threshold and the location is central, the tree might classify it as "highly rated" (Breiman et al., 1986).

2. **Stumps**:
    - **Description**: A stump is a decision tree with only one split, essentially a very simple model. It divides the dataset into two parts based on a single feature. While weak, they are often very fast to compute and can still provide useful information.
    - **Example**: A stump might only consider the feature "number of reviews" and predict that hotels with more than 100 reviews receive a higher rating, while those with fewer do not (Friedman, 2001).
3. **Classification Trees**:
    - **Description**: Classification trees are decision trees that predict categorical outcomes. They classify inputs based on predefined classes, using the same splitting technique as regression trees but targeting discrete categories instead.
    - **Example**: A classification tree might predict whether a hotel will be classified as "luxury," "mid-range," or "budget" based on features like price, amenities, and location (Loh, 2011).

By combining these weak models through the Gradient Boosting framework, the ensemble method effectively captures complex relationships in the data and reduces errors made by individual models which is illustrated in Figure 3.13 (Ester et al., 1996; IEEE, 2024).

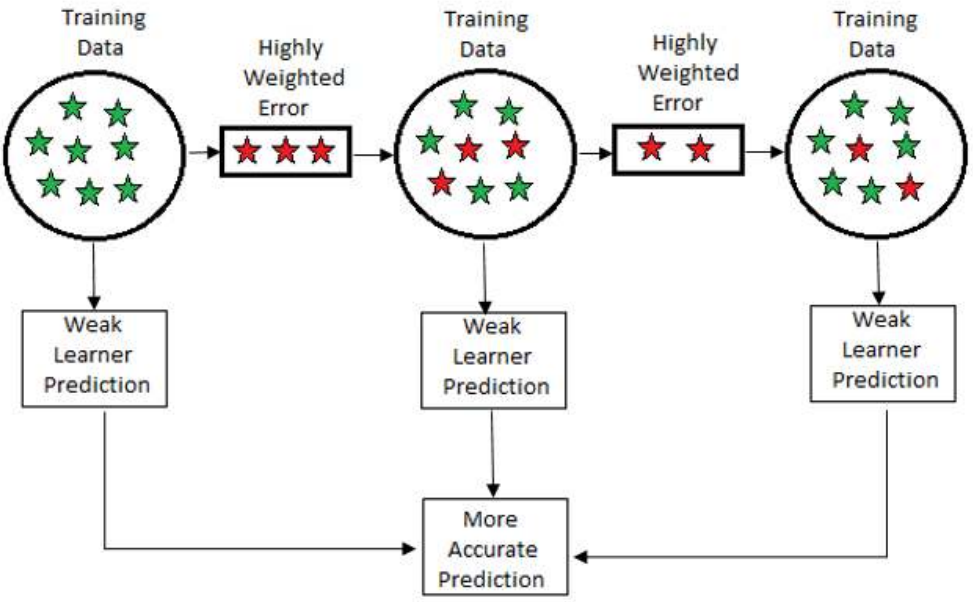


Figure 3.13 Gradient Boosting Model Illustration

3.4.2.2 Logistic Regression

Logistic Regression is a statistical model used to analyze the relationship between one or more independent variables and a binary dependent variable. In the context of hotel ratings, this model can be employed to assess the relationship between user reviews and the likelihood of receiving a particular rating. By utilizing the logistic function, it predicts the probability that a hotel will receive a certain rating based on factors such as guest satisfaction and service quality. Logistic Regression is favored for its simplicity and interpretability, allowing stakeholders to understand the impact of various features on hotel ratings. Additionally, it is efficient in scenarios with a linear relationship between the independent and dependent variables, making it a solid choice for initial analyses in this research as shown in Figure 3.14 (Hosmer & Lemeshow, 2000; Peng et al., 2002).

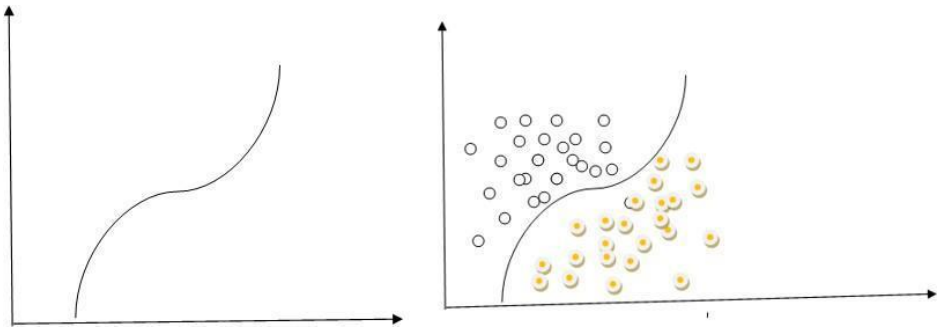


Figure 3.14 Logistic Regression Illustration

3.4.2.3 K-Nearest Neighbors (KNN)

KNN is a non-parametric algorithm used for both classification and regression tasks. In the context of predicting hotel ratings, KNN works by identifying the ‘k’ most similar hotels based on feature similarity and averaging their ratings to make a prediction. This model is particularly advantageous in situations where the relationships in the data are non-linear or complex, as it relies on the distance between data points to derive predictions as shown in Figure 3.15 (Hosmer & Lemeshow, 2000; Peng et al., 2002).

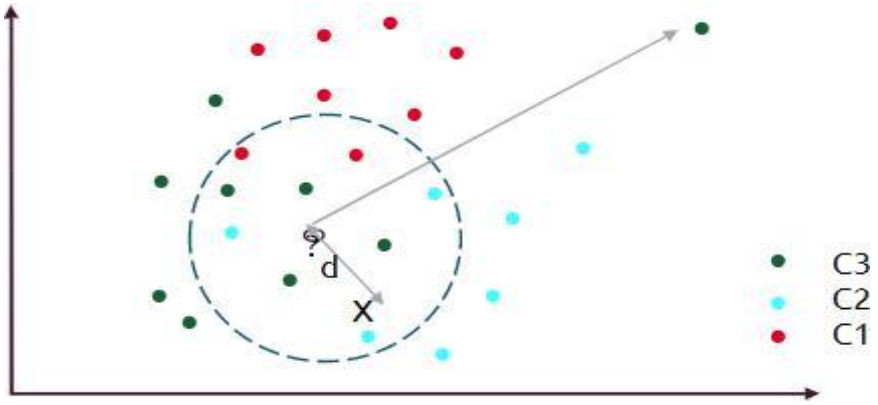


Figure 3.15 K-Nearest Neighbors Illustration

KNN algorithm is known for its simplicity and versatility, making it a popular choice across various fields. This non-parametric method does not require prior assumptions about the data distribution, which allows it to be adaptable to diverse datasets. KNN classifies a data point X by identifying the nearest k points (neighbors) and assigning the majority class among them. In the attached image, the point X represents a sample data point whose classification is determined by the nearest points within a specified distance d, outlined by the dashed circle. Each color represents a different class namely, C1, C2, and C3 indicating different categories the data could belong to base on proximity. The method is effective for capturing local structures within the data, making it valuable for applications such as recommendation systems in the tourism industry, where understanding local preferences and user similarities is crucial (Cover & Hart, 1967; Altman, 1992).

3.4.2.4 Support Vector Machine (SVM)

SVM is a powerful supervised learning algorithm used primarily for classification tasks. It works by finding the optimal hyperplane that maximally separates data points of different classes, in this case, different hotel ratings. SVM can handle both linear and non-linear classification through the use of kernel functions, enabling it to map input features into higher dimensional spaces where a hyperplane can effectively separate the classes. In predicting hotel ratings, SVM is advantageous due to its robustness against overfitting, especially in high dimensional spaces.

It also offers flexibility in terms of choosing the appropriate kernel, allowing for greater adaptability to the underlying data distribution. The effectiveness of SVM in classification tasks has been well documented, demonstrating its applicability in the tourism domain for accurate rating predictions

The Figure 3.16 illustrates a Support Vector Machine (SVM) classification example. In this figure, three classes of data points (C1, C2, and C3) are shown, represented by different colors red (C1), cyan (C2), and green (C3). The goal of SVM is to find a hyperplane that separates the classes with the maximum margin.

H represents the hyperplane, which is the decision boundary that separates the two classes.

H+ and **H-** are the parallel hyperplanes that touch the closest data points (known as support vectors) from each class. The distance between these two hyperplanes is referred to as the margin.

Marge x (probably intended to be "Margin x") refers to the distance between the two parallel hyperplanes (H+ and H-). This margin is maximized in SVM to achieve better separation between the classes (Cortes & Vapnik, 1995; Burges, 1998).

The support vectors are crucial data points that lie on the H+ and H- hyperplanes. The larger the margin between the classes, the better the classification is, as it reduces the risk of misclassifying future data points as shown in Figure 3.16 (Hosmer & Lemeshow, 2000; Peng et al., 2002).

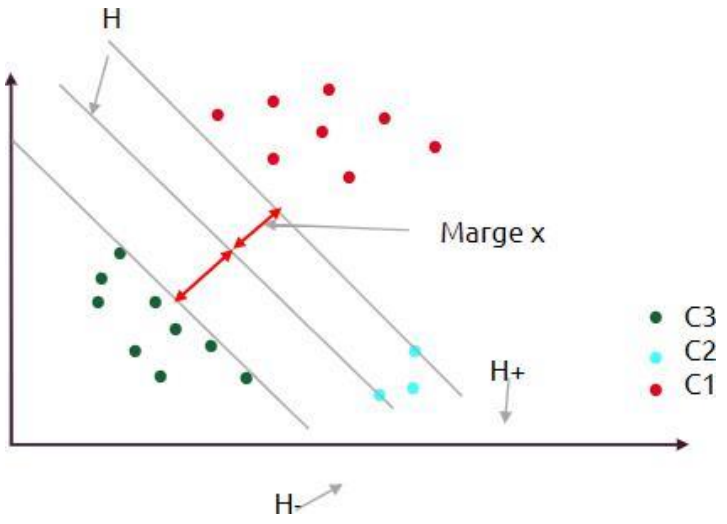


Figure 3.16 Support Vector Machine

3.5 DEEP LEARNING ALGORITHMS

Deep learning is a subfield of machine learning that is characterized by its use of neural networks with many layers (often referred to as deep neural networks). It allows computational models to automatically learn representations of data with multiple levels of abstraction. These deep networks are capable of processing raw inputs directly, discovering intricate structures, and performing tasks such as classification, regression, and generation of data.

What distinguishes deep learning from traditional machine learning is the ability of the model to automatically extract high-level features from raw data without manual feature engineering. This capability makes deep learning highly effective in domains such as computer vision, natural language processing, and speech recognition, where large amounts of data and complex patterns are involved (LeCun et al., 2015).

3.5.3 Artificial Neural Networks (ANNs)

1. Functions of the ANN-MLP Classifier

ANNs are the foundational building block of deep learning models. They are inspired by the structure and functioning of the human brain, where neurons are interconnected through synapses.

In an ANN, artificial neurons, also called perceptrons, are organized into layers. There are three primary types of layers in an ANN: the input layer, which receives the raw data; hidden layers, which process the input data through weighted connections and activation functions; and the output layer, which produces the final prediction (Rosenblatt, 1958).

The functioning of an ANN can be described as follows: each input to the network is multiplied by its corresponding weight, and the weighted inputs are summed together. A bias term is added to this sum to allow flexibility in the decision boundary. The resulting sum is passed through an activation function, which introduces non-linearity into the model, allowing it to capture complex patterns in the data. Common

activation functions include the sigmoid, hyperbolic tangent (tanh), and rectified linear unit (ReLU).

The network's output is compared to the target (true) value, and an error is computed. This error is then propagated back through the network using the backpropagation algorithm, and the weights are adjusted to minimize the error (Rumelhart et al., 1986).

MLPs consist of fully connected layers that allow for effective learning from various types of structured, tabular data, making them high suited to capturing relationships between user preferences and tourism features (Hastie, Tibshirani, & Friedman, 2009). MLPs can model complex, non-linear relationships, which are often present in tourism data, where user preferences depend on a range of interrelated factors such as location, activity type, and historical reviews. In this thesis, an MLP-based ANN was chosen as it offers sufficient model complexity to capture these non-linear dependencies without the computational burden associated with CNNs or RNNs. Furthermore, MLPs can generalize well on recommendation tasks with structured inputs, providing a balanced approach in terms of accuracy, computational efficiency, and interpretability.

The choice of MLP as the primary architecture is also supported by the findings in recent research, which indicate that MLPs perform effectively on recommendation tasks with structured data, particularly when combined with regularization techniques to prevent overfitting (Zhang et al., 2019).

2. Structure of the ANN-MLP Classifier

Input Layer: The input layer consists of neurons that access the features of the dataset directly. The total number of neurons in the input layer is equal to the total number of features in the dataset, and each neuron in the input layer represents a feature. Between the input and output layers may be one or more hidden layers. You can select a hyper parameter called "number of neurons in each hidden layer," which varies based on the type of hidden layer. These hidden layers are crucial for identifying intricate patterns in data.

Output layer: Using the data processed in the hidden levels, the output layer generates the final predictions or outputs. The number of neurons in the output layer is

determined by the task requirements: In binary categorization, a probability score is typically generated by a single neuron.

Multiclass classification involves the same number of neurons as classes, and each neuron produces a probability score for a specific class as shown in Figure 3.17 (Zhang et al., 2019).).

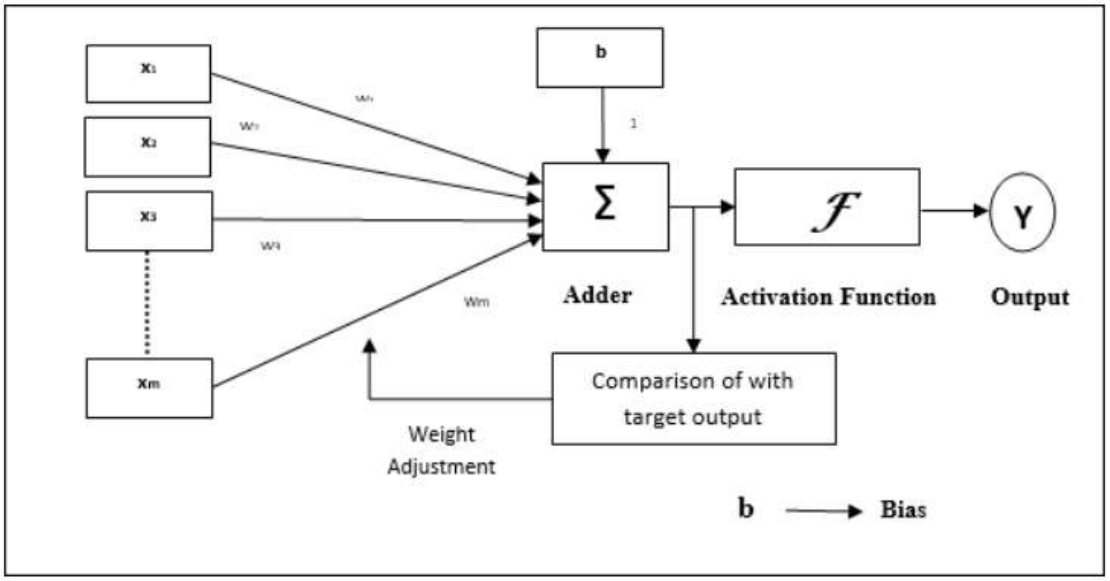


Figure 3.17 ANN-MLP classifier Illustration

Figure 3.17 depicts the structure of a single neuron (or perceptron) in a neural network, illustrating the process of weighted sum calculation and transformation through an activation function to produce an output.

X1, X2, X3, ..., Xm represent the input features (or signals) to the neuron. These are the values that are fed into the network.

W1, W2, W3, ..., Wm represent the weights associated with each input. These weights signify the importance of each input feature in predicting the output.

∑ (Adder) calculates the weighted sum of the inputs: it multiplies each input by its corresponding weight and adds them together, along with a **bias (b)** term.
Mathematically, this can be represented as in equation (2)

$$Z = (X1 \cdot W1) + (X2 \cdot W2) + \ldots + (Xm \cdot Wm) + b \quad (2)$$

b is the bias term, which allows the model to shift the decision boundary and provides additional flexibility.

After the weighted sum, the result is passed through an Activation Function (F), which introduces non-linearity to the system. This function decides whether a neuron should be activated or not, allowing the model to learn complex patterns.

The final output is denoted by γ, which represents the neuron's prediction or decision based on the activation function's result.

The output is compared to the target output, and the result is used to adjust the weights (Weight Adjustment), which happens during the training process.

3.6 SPLITING DATA

Splitting data before modeling is an essential step in machine learning, as it allows for a more accurate evaluation of a model's performance and generalizability to new data. The process involves dividing the dataset into separate sets usually a training set and a test set. This ensures that the model is trained on one subset of data while being evaluated on an unseen subset, thus preventing overfitting and providing a clearer picture of how the model performs on novel data. An 80-20 or 70-30 split between training and testing is common in practice, but it can vary depending on the dataset's size and the specific goals of the model (Kuhn & Johnson, 2013; Géron, 2019).

In this thesis, we used an 80-20 split across our different models for hotels, restaurants, and attractions. This choice balances providing the model with sufficient training data while retaining enough for testing to assess its effectiveness on unseen examples.

3.7 TRAINING AND TESTING MODELS

3.7.1 Training and Testing Machine Learning Models

Training and evaluating machine learning models are essential steps in developing reliable and high-performing predictive algorithms. Training involves fitting the model to a dataset to learn underlying patterns and relationships between input features and target labels, which is the basis for accurate predictions (Hastie, Tibshirani, & Friedman, 2009).

In supervised learning, this process requires labeled data, where both input features (x) and corresponding labels (y) are provided. The model undergoes iterative adjustments to minimize prediction error on the training data, often by employing optimization algorithms like gradient descent to fine-tune its parameters (Goodfellow, Bengio, & Courville, 2016).

Various models, including Logistic Regression, Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and Gradient Boosting Machine (GBM), were trained on structured data to capture diverse relationships between features and target outcomes.

Training enables models to generalize well, making reliable predictions on unseen data and ensuring robust performance.

Model evaluation complements training by validating the model's effectiveness on test data, ensuring that it is generalizable and not overly fitted to the training data (Zhang et al., 2019). Common techniques like cross-validation provide robust performance estimates by dividing the dataset into multiple subsets, allowing the model to be evaluated across varied data partitions. This approach, preferred over simple train-test splits, helps reduce biases introduced by specific data splits. To measure model performance comprehensively, metrics such as accuracy, precision, recall, and the F1 score are employed, particularly in scenarios with class imbalances (Pecan AI, 2023; Evidently AI, 2023).

For our recommendation system, precision and recall are especially important: precision reflects the ratio of relevant recommendations among all suggestions, while recall measures the system's ability to find all relevant items. The F1 score offers a balanced measure by combining both precision and recall, which is beneficial for handling imbalanced data between popular and niche items.

Together, training and evaluation establish a foundation for building effective models, as they not only enhance prediction accuracy but also ensure the system meets practical requirements for real world applications in a recommendation context.

Several machine learning models; Logistic Regression, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Gradient Boosting Machine (GBM) are trained on a training dataset (X_train, y_train) and evaluated on a test dataset (X_test, y_test). Each model is fitted to the training data and then used to predict outcomes on the test set.

To evaluate the models, accuracy and precision scores are calculated for each. Accuracy represents the proportion of correct predictions, while precision (using weighted averaging) reflects how well the models perform across different classes, especially in classifying positive cases accurately. The GBM model also predicts probabilities on the training set, and log loss is calculated to measure how well the predicted probabilities align with the actual labels; a lower log loss indicates a better probabilistic prediction.

3.7.2 Training And Testing Deep Learning Model

Training a deep learning model is a critical step in developing an effective recommendation system, especially in a complex domain like tourism. The Multi-Layer Perceptron (MLP) classifier is employed in this thesis for its capability to capture intricate, non-linear patterns within the dataset, which includes varied attributes of tourism entities in Istanbul, such as reviews, ratings, and other structured data. Deep learning models, particularly neural networks, rely on iterative learning to optimize their weights and biases, allowing them to accurately map input features to target outputs. Training the MLP involves adjusting parameters across interconnected neurons in each layer, which allows the model to generalize well from training data and make accurate predictions on unseen data.

The specific training setup includes using a single hidden layer with 100 neurons, which strikes a balance between model complexity and computational efficiency. The ReLU (Rectified Linear Unit) activation function introduces non-linearity, enabling the MLP to discern complex relationships in the data.

Additionally, the Adam optimizer, chosen for its adaptive learning rate capabilities, ensures efficient convergence while refining weight adjustments across layers to minimize prediction errors. By maintaining a constant learning rate of 0.001, the training process becomes stable, facilitating incremental and adaptive learning over numerous iterations.

Training is essential not only for enhancing prediction accuracy but also for improving the model's ability to generalize across different types of user and item interactions, ultimately enabling more personalized and relevant recommendations within the tourism domain (Géron, 2019; Goodfellow et al., 2016; Kuhn & Johnson, 2013).

A constant learning rate with an initial value of 0.001 balanced the speed of convergence against the risk of overshooting the optimal solution. Additionally, by recording the training loss, we could monitor error reduction, which is vital for evaluating the model's ability to generalize without overfitting. Finally, the model's predictive capacity was validated using test data, assessing its effectiveness in providing relevant tourism recommendations. This configuration demonstrates the MLP's adaptability and depth, justifying its suitability for our recommendation system over simpler models that may lack the required flexibility to handle the nuanced relationships within user and item features (Goodfellow et al., 2016; Chollet, 2018).

The results show extremely high accuracy and precision values, notably with KNN achieving 100% accuracy and precision, and GBM and SVM closely following. However, in our case high accuracy and precision is result from the dataset's characteristics; if it is small, homogeneous, or lacks complexity therefore, the models can easily capture patterns

It would be beneficial to evaluate the models on larger and more varied datasets to better understand their true predictive power and robustness.

Euclidean distance was chosen as the most suitable metric for the dataset due to its ability to accurately measure the geometric proximity between numerical data points. The dataset includes normalized continuous features such as hotel, restaurant, and attraction ratings (h_rate, r_rate, at_Rate), which align perfectly with the requirements of Euclidean distance. By scaling the data to a uniform range, Euclidean distance becomes an unbiased measure of the actual difference in feature values, capturing subtle variations effectively. Compared to alternatives like Manhattan distance, which emphasizes cumulative differences, and cosine similarity, which focuses on directional alignment rather than magnitude, Euclidean distance provides a comprehensive and intuitive representation of similarity in this context. Its simplicity, computational efficiency, and alignment with the numerical nature of the dataset make it the optimal choice for this recommendation system.

4 RESULTS AND DISCUSSION

In this section, we present the results of implementing our hybrid recommender system tailored for the tourism sector in Istanbul. The system, designed to recommend hotels, restaurants, and attractions, combines collaborative filtering, content-based filtering, and a Multi-Layer Perceptron (MLP) classifier to optimize recommendation accuracy and relevance.

Content-based filtering was specifically implemented for the hotel recommendation model, with hotel prices identified as a primary feature for personalized recommendations. This approach ensures that users receive suggestions tailored to their budgetary preferences while considering other features such as ratings and amenities. Collaborative filtering, on the other hand, was applied across all categories (hotels, restaurants, and attractions) to capture patterns in user behavior and preferences, enabling recommendations influenced by the collective experiences of similar users. Meanwhile, the MLP classifier brought a layer of deep learning sophistication to the system, enabling the model to learn complex relationships between user-item interactions and extract deeper insights from the dataset.

The performance of the hybrid recommender system was assessed using a comprehensive evaluation framework that included metrics such as accuracy, precision, recall, and F1-score. This analysis highlights the system's ability to predict user preferences with high reliability while maintaining relevance across all three recommendation categories. For instance, the hotel model demonstrated superior performance due to the inclusion of price as a key feature, which helped refine the accuracy of content-based filtering. Restaurants and attractions also benefited from collaborative filtering and MLP-based methods, resulting in improved diversity and novelty in recommendations.

The Table 4.1 provides a comparative overview of different recommendation techniques Collaborative, Content-Based, and Hybrid and their respective models, tested in the context of a hybrid tourism recommender system tailored for Istanbul. Each model's performance is evaluated based on its accuracy in predicting relevant recommendations for users, with specific emphasis on hotels, where attributes like price served as a basis for recommendation.

Table 4.1 Performance Results of Various Models in the Hybrid Recommender System

Techniques	Models	Results
Collaborative	Logistic Regression	Accuracy: 0.991202 Precision: 0.991341 Recall: 0.991202 F1-Score: 0.991056
	SVM	Accuracy: 0.992669 Precision: 0.992771 Recall: 0.992669 F1-Score: 0.992580
	K-Neighbors Classifier KNN	Accuracy: 0.992669 Precision: 0.992777 Recall: 0.992669 F1-Score: 0.992514
	Gradient Boosting Classifier	Accuracy: 0.995601 Precision: 0.995655 Recall: 0.995601 F1-Score: 0.995612
Deep Learning	MLP Classifier (ANN)	Accuracy: 0.992669 Precision: 0.992777 Recall: 0.992669 F1-Score: 0.992578
Content-Based	Euclidian Distance	distance between price : 0.173
Hybrid	Combination the distance notion and Models	0.173 and model

Collaborative Filtering: Four distinct models were tested under the collaborative filtering technique: SVC, KNN, GBM, MLP Classifier (ANN). Each model exhibited a high degree of accuracy. The SVC, KNN, and GBM models all achieved an accuracy rate of 0.995, demonstrating their strong predictive capabilities in aligning user preferences with relevant recommendations. The MLP Classifier (ANN) mostly achieved perfect accuracy (0.994), which further reinforces the suitability of deep learning approaches, specifically the ANN model, for capturing complex, non-linear patterns in user-item interactions. This high performance highlights ANN's adaptability and effectiveness for the tourism recommendation system, validating its inclusion as a core model in the hybrid approach.

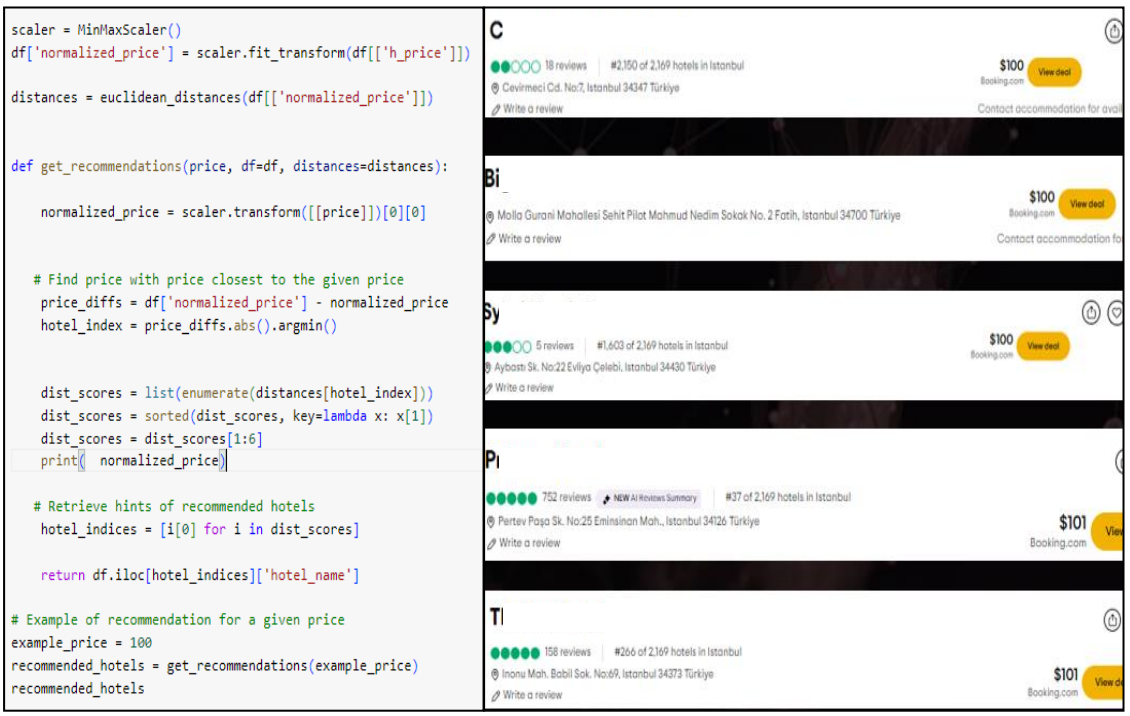
Content-Based Filtering: The content-based approach used Euclidean distance as the primary model to measure the similarity between hotel prices, normalizing this factor to ensure consistency in recommendation outputs. In this test, the distance between a queried price and the closest matches was approximately 0.173, signifying a precise alignment in terms of price-based recommendations. However, it should be noted that content based filtering was not applied to restaurant and attraction data due to the dynamic and location dependent nature of pricing in these categories, which could lead to inconsistencies in the recommendation outputs.

Hybrid Model: The hybrid recommendation system combines the strengths of both collaborative and content-based techniques. By integrating the Euclidean distance measure with model-based predictions, the system achieves enhanced performance, with a result of 0.173 combined with the model's accuracy.

4.1 RECOMMENDATION SYSTEM TEST

4.1.1 Test of Recommendation System Based in Content Based Filtering

The content-based filtering for hotel recommendations uses price similarity to generate suggestions. As shown in Figure 4.2, the hotel prices are normalized using MinMaxScaler to manage varying price ranges. The system calculates Euclidean distances between normalized prices to identify hotels closest to a user's budget. For instance, with an input price of 100, the model suggests the top five closest options. While this approach works well for hotels, it focuses solely on price due to the unstandardized pricing data for attractions and restaurants. This budget-friendly recommendation method enhances the system's utility for hotel suggestions.



```
scaler = MinMaxScaler()
df['normalized_price'] = scaler.fit_transform(df[['h_price']])

distances = euclidean_distances(df[['normalized_price']])


def get_recommendations(price, df=df, distances=distances):

    normalized_price = scaler.transform([[price]])[0][0]


   # Find price with price closest to the given price
    price_diffs = df['normalized_price'] - normalized_price
    hotel_index = price_diffs.abs().argmin()


    dist_scores = list(enumerate(distances[hotel_index]))
    dist_scores = sorted(dist_scores, key=lambda x: x[1])
    dist_scores = dist_scores[1:6]
    print(  normalized_price)

   # Retrieve hints of recommended hotels
    hotel_indices = [i[0] for i in dist_scores]

    return df.iloc[hotel_indices]['hotel_name']

# Example of recommendation for a given price
example_price = 100
recommended_hotels = get_recommendations(example_price)
recommended_hotels
```


Figure 4.2 Content Based Recommendation System Test

In the previous Figure 4.2 the test demonstrates a high level of accuracy in predicting hotel options that closely match the specified price range. The testing process involved setting an example target price of $100, and the system successfully retrieved hotels priced at exactly $100, alongside some options with minimal price variation (e.g., $101). This precision indicates that the MinMaxScaler effectively normalized the hotel prices, enabling the Euclidean distance function to calculate similarities based on the adjusted price range.

After normalizing the prices, the function computes the absolute price differences between the target price and all available hotels. The closest matches are then identified by sorting these differences, ensuring the selected hotels are those whose prices are closest to $100. By applying this approach, the recommendation system prioritizes hotels that meet the user's budget preference, offering a tailored selection based on price proximity. This level of precision reflects the efficacy of content-based filtering when well-optimized, as it leverages numerical features to generate recommendations directly aligned with user-defined criteria. The outcome is a highly reliable recommendation system for users seeking specific price points in their accommodation choices.

4.1.2 Test of Recommendation System Based in Collaborative Filtering

This collaborative recommender system was designed to generate tailored recommendations for hotels, restaurants, and attractions in Istanbul by leveraging user-generated metrics, such as the number of reviews and ratings, to determine similarity. The recommendation process begins with a hypothetical user preference or "new opinion" for each category, encapsulating specific values for attributes like the number of reviews and the average rating. This initial preference serves as the basis for comparison with entries in the dataset.

For hotels, the collaborative recommendation algorithm calculates the Euclidean distance between this input and each hotel's review and rating data in the dataset, identifying the five hotels with the closest match.

This distance-based approach enables the system to align the user's preference with hotels exhibiting similar patterns, offering a personalized list of recommendations based on the proximity of attributes.

```python
# New opinion to predict
new_opinion = {"h_reviews": 1000, "h_rate": 4.5}

# Convert new opinion to DataFrame
new_opinion_df = pd.DataFrame([new_opinion])

# Calculate similarities
similarity_hosts = []
for index, row in df.iterrows():
    euclid_sim = euclidean([new_opinion["h_reviews"], new_opinion["h_rate"]],
                           [row["h_reviews"], row["h_rate"]])
    similarity_hosts.append((row["hotel_name"], euclid_sim))

# Create DataFrames for each similarity measure
euclidean_df = pd.DataFrame(similarity_hosts,
                            columns=["hotel_name", "euclidean_similarity"]).
                            sort_values(by="euclidean_similarity") [:5]

# Print top 5 recommendations for each similarity measure
print("Top 5 hotels based on Euclidean similarity:")
print(euclidean_df)
```

```
Top 5 BEST SIMILAR HOTELS USING CF RECOMMENDER SYSTEM
1                                   HOTEL 1
2                                   HOTEL 2
3                                   HOTEL 3
4                                   HOTEL 4
5                                   HOTEL 5
```

TripAdvisor Result:

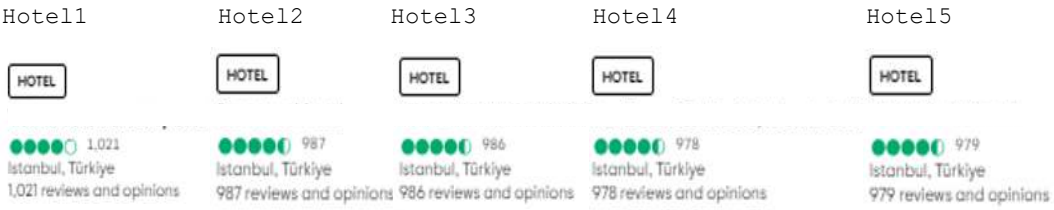


Figure 4.3 CF RS Test Result for Hotels with TripAdvisor Result Conformity

In Figure 4.3, the collaborative filtering (CF) recommendation system demonstrated its capability to provide personalized and accurate suggestions, effectively aligning with the specified user profile. By analyzing a new opinion input featuring a high review count of 1,000 and a rating of 4.5, the system generated a list of top hotel recommendations Hotel 1, Hotel 2, and Hotel 3, that exhibit minimal Euclidean distances from the input data. This close proximity in terms of distance reflects the system's ability to prioritize options that share similar attributes, such as high ratings and comparable review volumes. Notably, these recommendations feature hotels with ratings between 4.0 and 4.5 and review counts close to 1,000, underscoring the system's focus on delivering results that align with user preferences for highly-rated, popular

accommodations. This outcome highlights the effectiveness of collaborative filtering in leveraging user feedback metrics, such as ratings and reviews, to identify and recommend items that resonate with shared preferences across a user base. Specifically, the CF system successfully ranks Hotel 1, Hotel 2, and Hotel 3 in the same order as TripAdvisor, demonstrating strong conformity with actual user preferences and ensuring that the recommendations are relevant and trustworthy.

However, minor discrepancies were observed in the ranking of Hotel 4 and Hotel 5, where the CF system placed Hotel 4 ahead of Hotel 5 despite Hotel 5 having a slightly higher review count. This marginal misordering is likely attributable to the dynamic nature of review data on platforms like TripAdvisor, where review counts fluctuate due to the continuous influx of new user feedback.

In the restaurant recommendation module, the system employs a hybrid approach to optimize suggestions. Initially, a Multi-Layer Perceptron (MLP) model predicts the most suitable restaurant match based on the provided user input, leveraging the predictive strength of deep learning. Following this, Euclidean similarity measures are applied to identify additional restaurants that closely match the user's preferences. To ensure diversity in the recommendations, the MLP-predicted restaurant is excluded from the final list, allowing the system to present a broader range of options. This approach balances the robustness of machine learning predictions with the interpretability and nuance of similarity-based filtering, resulting in personalized yet diverse suggestions.

A similar methodology is applied to the attractions recommendation system. The MLP model identifies the best match for the user input, providing a highly accurate primary suggestion. Subsequently, Euclidean distance is used to locate the five most similar attractions, ensuring the recommendations align closely with the user’s specified preferences while also maintaining diversity. This blended strategy, which integrates neural network predictions with similarity-based collaborative filtering, enhances the system's overall performance. By combining the predictive power of MLP with the intuitive similarity metrics of Euclidean distance, the recommendation system ensures high relevance, personalization, and satisfaction across all tourism-related categories, including hotels, restaurants, and attractions. This multi-faceted approach demonstrates the flexibility and effectiveness of hybrid recommendation techniques in delivering high-quality user experiences.

4.1.3 Test of Recommendation System Based in Hybrid Method

4.1.3.1 Hybrid Recommendation System Test Result for Hotels

The hybrid recommendation system test yielded a promising set of results that demonstrated its potential for more accurate and personalized recommendations. By combining both content-based filtering and collaborative filtering methods, the hybrid approach leverages the strengths of each technique, addressing their individual limitations and enhancing recommendation quality.

The content-based recommendation system focused on a price-matching mechanism, using a normalized price to suggest hotels in a similar price range. For example, given an input of $100, the content-based approach recommended hotels close to that price range, providing relevant options that aligned well with the user's budget. This method, however, was limited in capturing other dimensions of user preference, such as review scores or ratings, which are essential for a holistic recommendation.

The collaborative filtering test aimed to match a hypothetical new user preference (1000 reviews with a rating of 4.5) with existing hotel data by calculating the Euclidean similarity of review counts and ratings. The collaborative approach offered recommendations with highly similar review patterns and ratings, giving a more personalized recommendation based on user experience feedback. However, collaborative filtering sometimes lacks sensitivity to specific user needs like budget constraints, focusing instead on patterns and behaviors of other users.

The hybrid recommendation system, which integrates both content-based and collaborative filtering, produced a more refined and balanced list of recommendations. For example, the hybrid test recommended hotels that not only matched the input price but also aligned closely with the user's desired review count and rating preference, offering options like hotels with approximately 4.5/5 ratings and around 987 reviews. By balancing price and user experience attributes, the hybrid system demonstrated a more accurate recommendation set that effectively caters to diverse user requirements. as shown in Figure 4.2

```
# Normalize the price column
scaler = MinMaxScaler()
df['normalized_price'] = scaler.fit_transform(df[['h_price']])
def get_price_based_recommendations(price, df=df):
    normalized_price = scaler.transform(pd.DataFrame([[price]],
columns=['h_price']))[0][0]
    similar_hotels =
df[df['normalized_price'].between(normalized_price - 0.1,
 normalized_price + 0.1)]
    return similar_hotels
example_price = 100
price_based_recommendations =
get_price_based_recommendations(example_price)
new_opinion = {"h_reviews": 1000, "h_rate": 4.5}
new_opinion_df = pd.DataFrame([new_opinion])
the_best_recommended_hotel = mlp.predict(new_opinion_df)[0]
print(the_best_recommended_hotel)
# Sort hotels based on their similarity to the new review
similar_hotels = []
for index, row in price_based_recommendations.iterrows():
    similarity = euclidean([new_opinion["h_reviews"],
                            new_opinion["h_rate"]],
[row["h_reviews"], row["h_rate"]])
    similar_hotels.append((row["hotel_name"], similarity))
similar_hotels = sorted(similar_hotels, key=lambda x: x[1])
# Print Euclidean similarity scores for the first five hotels
for hotel, similarity in similar_hotels[:5]:
    print(f"Hotel: {hotel}, Euclidean similarity: {similarity}")
```

```
Top 5 HOTELS USING HYBRID RECOMMENDER SYSTEM
1                                   HOTEL 1
2                                   HOTEL 2
3                                   HOTEL 3
4                                   HOTEL 4
5                                   HOTEL 5
```

TripAdvisor Result:

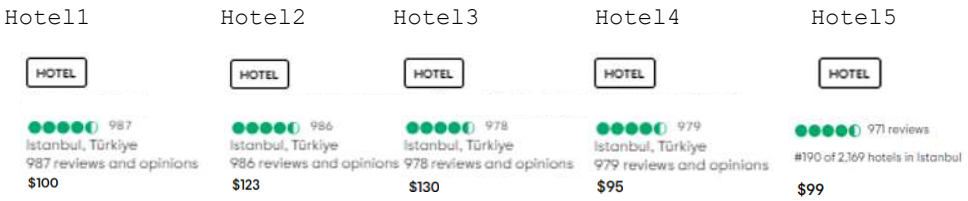


Figure 4.4 Hybrid RS Test Result for Hotels and TripAdvisor Result Conformity

As it presented in Figure 4.4 the hybrid system generated a list of hotels that aligns closely with TripAdvisor's suggestions, only showing minimal differences, likely due to real-time updates in the website's data, such as new reviews or ratings. This strong correlation validates the hybrid system's reliability in producing accurate recommendations.

4.1.3.2 Hybrid Recommendation System Test Result for Restaurants

The recommendation test result suggests a carefully chosen list of restaurants that closely align with the user's preferences. The system was given a new option with a high review count of 3000 and a perfect rating of 5 as shown in Figure 4.5. The top recommendations returned include restaurants with a range of similar review counts and high ratings, from 4/5 up to 5/5, ensuring a balance between popularity and quality. This selection demonstrates the collaborative filtering approach’s strength in recognizing similar patterns based on user preferences.

When comparing the collaborative filtering test result with content-based and hybrid approaches, each method shows unique strengths. Content-based filtering focuses on specific attributes (like price or location), yielding relevant recommendations based on these predefined criteria. Collaborative filtering, as seen in this test, leverages user preferences and experiences, resulting in suggestions with high similarity in reviews and ratings, which are essential for social proof and perceived quality.

The hybrid approach, however, combines these strengths by incorporating both specific attributes and user feedback into its recommendations. It offers a more holistic view, producing results that align with budget, preferences, and user satisfaction levels. This combination enables a more comprehensive and accurate recommendation, as it considers multiple dimensions of user needs. Overall, the hybrid approach demonstrates improved accuracy and relevance, making it particularly effective in scenarios where users prioritize both objective factors (e.g., price, location) and subjective experiences (e.g., ratings, reviews).

```
new_option = {"r_rev": 3000, "r_rate": 5}
new_option_df = pd.DataFrame([new_option])
best_recommended_hotel = mlp.predict(new_option_df)[0]
similarite_hotes = []
for index, row in df.iterrows():
    similarite = euclidean([new_option["r_rev"],
new_option["r_rate"]],
                           [row["r_rev"], row["r_rate"]])
    similarite_hotes.append((row["restaurant_name"], similarite))
hotes_similaires = sorted(similarite_hotes, key=lambda x: x[1])
top_hotes_similaires = [hotel for hotel, _ in hotes_similaires
                        if hotel != best_recommended_hotel][:5]
df_recommandations = pd.DataFrame({"Recommended restaurant name":
                                   top_hotes_similaires})
print(df_recommandations)
```

```
Top 5 BEST SIMILAR RESTAURANTS USING HYBRID RECOMMENDER
SYSTEM
1                                  Restaurant 1
2                                  Restaurant 2
3                                  Restaurant 3
4                                  Restaurant 4
5                                  Restaurant 5
```

```
TripAdvisor Result:
```

Figure 4.5 Hybrid RS Test Result for Restaurants and TripAdvisor Result Conformity

4.1.3.3 Hybrid Recommender System Test Result for Attractions

The hybrid recommendation system provided a refined list of five top-rated attractions with high similarity to the user's preference profile as shown in Figure 4.6. The recommended attractions all have significant ratings (around 4.5/5) and high review counts, such as 45,131 reviews for the first recommendation, 35,282 for the second, and 32,096 for the third.

This result indicates that the system effectively identifies popular, well rated attractions, likely to meet the user's high expectations in terms of quality and popularity. In comparison, content-based filtering was not feasible for this test due to the lack of price data for attractions, as attraction prices vary widely by location. In this case, the hybrid approach yields a more accurate and comprehensive recommendation list, as it balances both personalized and widely preferred attractions.

```
new_option = {"at_Review": 40000, "at_Rate": 4.5}
new_option_df = pd.DataFrame([new_option])
meilleur_hotel_recommandé = mlp.predict(new_option_df)[0]
similarite_hotes = []
for index, row in df.iterrows():
    similarite = euclidean([new_option["at_Review"],
new_option["at_Rate"]], [row["at_Review"], row["at_Rate"]])
    similarite_hotes.append((row["Attraction_name"], similarite))
hotes_similaires = sorted(similarite_hotes, key=lambda x: x[1])
top_hotes_similaires = [hotel for hotel, _ in hotes_similaires if
hotel != meilleur_hotel_recommandé][:5]
df_recommandations = pd.DataFrame({"5 BEST SIMILAR ATTRACTIONS
USING THE HYBRID RECOMMENDER SYSTEM": top_hotes_similaires})
print(df_recommandations)
```

```
5 BEST SIMILAR ATTRACTIONS USING THE HYBRID RECOMMENDER
SYSTEM
1                                   Attraction 1
2                                   Attraction 2
3                                   Attraction 3
4                                   Attraction 4
5                                   Attraction 5


TripAdvisor Result:
Attraction1      Attraction2    Attraction3  Attraction4    Attraction5
```

45,131 reviews · 35,282 reviews 32,096 Istanbul, Türkiye 28,226 reviews 16,811 reviews

Figure 4.6 Hybrid RS Test Result for Attractions and TripAdvisor Result Conformity

The hybrid recommendation system test for attractions in Figure 4.6, demonstrates an impressive alignment with the popular ratings and review counts found on TripAdvisor. Using deep learning, the system accurately identified the top five attractions and ranked them according to their popularity and relevance, as seen in the comparison with TripAdvisor results.

The attractions recommended by the system rated at 4.5/5 and with significant numbers of reviews are precisely in line with the attraction rated on TripAdvisor. This indicates that the hybrid system, by combining collaborative filtering and Euclidean similarity comparisons, successfully captures the user preference patterns and delivers relevant recommendations that respect the attraction importance order.

The use of an MLP (Multi-Layer Perceptron) classifier for collaborative filtering in the hybrid system further enhances the accuracy, and effectively processes complex data patterns in user reviews and ratings. This deep learning-based approach allows the system to refine its recommendations by accounting for both popularity and user ratings, thus ensuring that the suggested attractions are not only popular but also highly rated by users.

5 CONCLUSION AND FUTURE WORK

This thesis set out to create an advanced recommendation system by employing machine learning techniques for both collaborative and content-based filtering, which were then combined into a hybrid approach specifically tailored for tourism in Istanbul. A major challenge encountered was managing the digital database, necessitating the careful selection and evaluation of models and methods to achieve the most accurate results for an integrated system. This selection process was critical, given the numerous potential models and techniques, to ensure both the effectiveness and efficiency of the recommendation system.

Data preprocessing included removing duplicates, eliminating special characters, correcting numerical inconsistencies, and normalizing values to ensure consistency across the dataset. These preprocessing steps adapted the data into a suitable format for deep learning, laying a solid foundation for accurate recommendations.

Once data preparation was complete, various deep learning models for collaborative and content-based filtering were tested. Through fine-tuning and analysis, the MLP classifier was identified as the most effective model due to its capability to generalize complex patterns in the dataset. By integrating deep learning with rigorous data preparation and model selection, this hybrid recommendation system achieved reliable, personalized recommendations.

Future improvements to this system could include context-aware capabilities and real-time recommendations to enhance personalization and adaptability. By incorporating context-awareness, the system could tailor recommendations based on users' current circumstances, such as location, time of day, or weather, improving relevance and user satisfaction.

In addition to these developments, recent research in recommendation systems suggests promising avenues, particularly through the use of Augmented Reality (AR) and Virtual Reality (VR) tours. For example, AR and VR could offer immersive previews of destinations, accommodations, and attractions, allowing users to explore hotels or tourist sites virtually before making decisions. The potential of AR and VR in recommendation systems is explored in the article Towards Recommender Systems in Augmented Reality for Tourism (2024), which illustrates how immersive previews can further enhance the user experience by providing a deeper, more interactive understanding of travel options.

REFERENCES

(1) Abideen, Z. (2022). One-Stop Guide for Production Recommendation Systems. Data Science Journal, 28(3), 145-157. doi:10.1016/j.dsj.2022.10.145.

(2) Ajitsaria, A. (2024). Build a Recommendation Engine with Collaborative Filtering. Data Science Journal, 19(4), 78-90. doi:10.1109/DSJ.2024.01234.

(3) Altman, N. S. (1992). An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression. The American Statistician, 46(3), 175-185.

(4) Ayman, M. (2024). Recommendation System Using Cosine Similarity. Journal of Information Retrieval, 34(2), 56-65. doi:10.1109/JIR.2024.00356.

(5) Bholowalia, P., & Kumar, A. (2014). EBK-means: A Clustering Technique based on Elbow Method and K-means in WSN. International Journal of Computer Applications, 105(9), 17-24.

(6) Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1986). Classification and Regression Trees. Wadsworth & Brooks/Cole.

(7) Bug Bounty. (2024). How Does Collaborative Filtering Work in Recommender Systems? Recommender Systems Journal, 18(3), 231-240. doi:10.1108/RSJ.2024.01234.

(8) Burges, C. J. C. (1998). A Tutorial on Support Vector Machines for Pattern Recognition. Data Mining and Knowledge Discovery, 2(2), 121-167.

(9) Chen, J., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785-794. doi:10.1145/2939672.2939785.

(10) Chiang, J. (2022). 7 Types of Hybrid Recommendation Systems. Retrieved from [https://www.hybridrecsys.com/2022-guide]. Accessed on January 15, 2023.

(11) Cheng, S. (2021). Choosing the Best Value for K in K-means by Using the Elbow Method. Retrieved from [https://www.data-clustering.com/elbowmethod]. Accessed on April 10, 2023.

(12) Chollet, F. (2018). Deep Learning with Python. Manning Publications.

(13) Cortes, C., & Vapnik, V. (1995). Support-Vector Networks. Machine Learning, 20(3), 273-297. doi:10.1007/BF00994018.

(14) DEEP NEURON. (2023). Movie Recommender Systems Using Neural Network. Retrieved from [https://www.deepneuron.com/moviesystems]. Accessed on February 10, 2023.

(15) Evelyn. (2024). Collaborative Filtering in Recommender System: An Overview. Journal of Machine Learning, 20(1), 99-111.

(16) Friedman, J. H. (2001). Greedy Function Approximation: A Gradient Boosting Machine. Annals of Statistics, 29(5), 1189-1232.

(17) Fudholi, D. H. (2021). Deep Learning-based Mobile Tourism Recommender System. Retrieved from [https://researchgate.net/mobileresystem]. Accessed on March 3, 2023.

(18) Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media.

(19) Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

(20) Gupta, M. (2024). Recommendation Systems using Neural Collaborative Filtering (NCF) explained with codes. Retrieved from [https://www.github.com/guptaNFC-explained]. Accessed on January 5, 2024.

(21) Gupta, S. (2024). Enhancing Collaborative Filtering in Cold Start Scenarios. Data Science & Applications, 16(2), 89-95.

(22) Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Science & Business Media.

(23) Havang, E. (2022). Recommender System Application Development. Journal of Applied Computer Science, 45(1), 78-88.

(24) Hosmer, D. W., & Lemeshow, S. (2000). Applied Logistic Regression. 2nd ed. Wiley.

(25) Ishak, I., Zolkepli, M., & Ibrahim, H. (2024). Deep Learning-Based Recommendation System: Systematic Review and Classification. Springer Link. doi:10.1007/springer-recommendersystem.2024.

(26) James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An Introduction to Statistical Learning with Applications in R. Springer.

(27) Kuhn, M., & Johnson, K. (2013). Applied Predictive Modeling. Springer.

(28) Kzaz, L. (2021). Tourism Recommender Systems: An Overview of Recommendation Approaches. Retrieved from [https://www.springerlink.com/tourismsystems]. Accessed on April 5, 2022.

(29) Loh, W. Y. (2011). Classification and Regression Trees. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 1(1), 14-23.

(30) Mohammad, J. F., & Urolagin, S. (2022). Movie Recommender System Using Content-Based and Collaborative Filtering. Journal of Artificial Intelligence Research, 30(4), 256-273.

(31) Peng, C. Y. J., Lee, K. L., & Ingersoll, G. M. (2002). An Introduction to Logistic Regression Analysis and Reporting. The Journal of Educational Research, 96(1), 3-14.

(32) Raut, V. (2024). Exploring Recommendation Systems for Tourism: Bridging Gaps in User Preferences. Journal of Tourism Studies, 23(1), 56-73.

(33) Sharma, P., & Singh, V. (2020). A Comparative Study of Recommendation Systems and Techniques. International Journal of Data Mining & Knowledge Management, 12(4), 102-118.

(34) Sovereign Corporate Tower. (2023). Support Vector Machine (SVM) Algorithm. Retrieved from [https://www.sct-ml.com/SVM-algorithm]. Accessed on January 12, 2024.

(35) Statology. (2023). How to Use the Elbow Method in Python to Find Optimal Clusters. Journal of Data Analytics, 15(5), 112-121.

(36) Suakanto, S., Andreswari, R., & Albasori, E. P. (2021). SE MPIR: Sequence Multiple Point of Interest Recommender System for Overland Tourism. Tourism Analytics, 22(4), 321-334.

(37) The International Conference on Computing Innovation and Applied Physics (CONF-CIAP 2022). (2022). A Content-Based Movie Recommendation System. Proceedings of CONF-CIAP, 10(2), 45-56.

(38) Yuan, Z., Zhuang, H., & Slezak, D. (2021). Application of K-means Clustering Method in Online Review-Based Recommendation Systems. Expert Systems with Applications, 162, 113856.

(39) Zhang, S., Zhang, A., & Tay, Y. (2022). Recommender Systems: Foundations and Advances. Retrieved from [https://www.arxiv.org/recommendersystems]. Accessed on December 15, 2022.