

PERSONALIZED TRAVEL ITINERARY GENERATOR SYSTEM

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering

By

NITHYASRI PRABHA S (Reg. No: 40110864)
NIVASH V (Reg. No: 40110866)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Category-1 University by UGC

Accredited with Grade "A++" by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHISALAI,
CHENNAI – 600119

APRIL - 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited "A++" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the Bonafide work of **Nithyasri Prabha S (Reg. No. 40110864), Nivash V (Reg. No. 40110866)** who carried out the project entitled "**Personalized Travel Itinerary Generator System**" under my supervision from November 2023 to April 2024.

Internal Guide

Ms. R. Yogitha, M.E., (Ph.D.)

HEAD OF THE DEPARTMENT
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Dr. Lakshmanan M.E., (Ph.D.)
Channel - 600 113.

Submitted for Viva voce Examination held on 15.04.2024

15/4/24
Internal Examiner

V Sathyasri

External Examiner

(Oral And written)

DECLARATION

I, **Nithyasri Prabha S** (Reg. No. 40110864) hereby declare that the Project Report entitled "**Personalized Travel Itinerary Generator System**" done by me under the guidance of **Ms. R. Yogitha, M.E., (Ph.D.)**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE: 12/4/2024

PLACE: CHENNAI



SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., (Ph.D.), Dean**, School of Computing, **Dr. L. Lakshmanan, M.E., Ph.D., Head of the Department of Computer Science and Engineering** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Ms. R. Yogitha, M.E., (Ph.D.)** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

The Personalized Travel Itinerary Generator System is a cutting-edge software solution aimed at simplifying and enhancing the travel planning experience. In an era of increasing travel complexities, this system offers an innovative approach to crafting tailor-made travel itineraries. This project's core objective is to design a user-friendly application that automates the itinerary creation process. Through efficient data scraping, processing, and optimization, it collects and analyzes destination-related information, enabling the creation of optimal travel plans. Flexibility is a cornerstone of this system, allowing users to modify itineraries, adjust schedules, and incorporate personal notes, ensuring that each traveler's unique requirements are met. Furthermore, the integration of external APIs and databases provides real-time access to data about attractions, weather conditions, transportation schedules, and accommodation availability. The Personalized Travel Itinerary Generator System aspires to revolutionize travel planning by offering a convenient, highly personalized, and enjoyable experience, catering to the individual traveler's needs while simplifying the planning process on a global scale. The system promotes flexibility by allowing users to modify itineraries, adjust timings, and incorporate personal notes, ensuring that each traveler's unique needs are met. Planning for a vacation that takes into consideration all the travel preferences of an individual without having to look through at least hundreds of websites is close to impossible. Our motivation behind doing this project is to reduce the time spent on planning for the vacation and help travelers spend more time on a vacation that they will love. Our project provides a Tailor-made Travel Plan for travelers based on their travel details and preferences. We obtain information from users about the destination, travel dates, budget, amenities that they look for in hotels, categories of attractions that users would love to visit and their favorite cuisine type. Using these we generate a travel plan for the entire trip, recommending users where to stay, what to do at which part of the day and where to eat for different meals of a day. This project will be a one-stop tool for travelers planning their vacations, we achieve the same using Data Science tools and techniques which will be explained in detail in the report.

TABLE OF CONTENTS

Chapter No	TITLE	Page No.
	ABSTRACT	
	LIST OF FIGURES	vii
	LIST OF ABBREVIATIONS	viii
1	INTRODUCTION	01
2	LITERATURE SURVEY	04
	2.1 Inferences from Literature Survey	07
	2.2 Open problems in Existing System	08
3	REQUIREMENT ANALYSIS	10
	3.1 Aim of the Project	10
	3.2 Scope of the Project	10
	3.3 Feasibility Studies/Risk Analysis of the Project	11
	3.4 System Requirements Specification Document	12
4	DESCRIPTION OF PROPOSED SYSTEM	14
	4.1 Modules	14
	4.2 Data science pipeline	14
5	IMPLEMENTATION DETAILS	21
	5.1 Model training methodology and evaluation	21
6	RESULTS AND DISCUSSIONS	26
	6.1 Overview of project result	26
7	CONCLUSION	28
	7.1 Conclusion	28
	7.2 Future Work	28

REFERENCES	29
APPENDIX	30
A. SOURCE CODE	30
B. SCREENSHOTS	45
C. RESEARCH PAPER	49
D. CONFERENCE CERTIFICATE	50

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE. NO
1.1	Apache spark	02
1.2	TensorFlow	02
1.3	Natural language toolkit(NLTK)	03
3.1	Windows	12
3.2	Python	13
3.3	Jupyter notebook	13
4.1	Complete technological pipeline of our project	14
4.2	Few Visualizations we created during EDA	18
4.3	Travel details of user	18
4.4	Attraction category rating	19
4.5	Hotel amenity preferences	20
4.6	Restaurant cuisine type and preferences	20
5.1	Hotel recommendations	26
5.2	Attraction recommendations	27
5.3	Restaurant recommendations	27

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
ALS	Alternating Least Square
API	Application programming interface
DRS	Destination recommendation system
DT	Decision tree
ETL	Extraction, Transformation, Loading
ETP	Electronic travel planner
GPG	GNU Privacy Guard
GPS	Global positioning system
HTML	Hyper Text Markup Language
HTTP	Hypertext Transfer Protocol
MF	Matrix factorization
MLP	Multi layer perceptron
NLTK	Natural language toolkit
RBM	Restricted boltzman machine
RMSE	Root Mean Square Error
SVM	Support vector machine

CHAPTER 1

INTRODUCTION

The Travel Itinerary Generator System is an innovative project designed to streamline and enhance the travel planning process. In a world marked by the increasing complexity of travel options, this system offers a solution to the time-consuming task of creating personalized travel itineraries. This project's primary objective is to develop a user-friendly software application that automates itinerary creation. With this system, users will have access to a powerful tool that takes into account their preferences, interests, and constraints to generate customized itineraries. It will employ advanced data scraping, processing, and optimization techniques to efficiently gather information about destinations, activities, and accommodations, resulting in optimal travel plans. Moreover, the system promotes flexibility by allowing users to modify itineraries, adjust timings, and incorporate personal notes, ensuring that each traveler's unique needs are met. Integration with external APIs and databases will provide real-time information on attractions, weather conditions, transportation schedules, and accommodation availability. The Travel Itinerary Generator System aspires to revolutionize travel planning by offering convenience, customization, and an overall more enjoyable travel experience, simplifying the process for travelers worldwide.

1.1. APACHE SPARK:

Apache Spark is an open-source distributed computing system designed for big data processing and analytics. It provides an interface for programming entire clusters with implicit data parallelism and fault tolerance. Spark offers libraries for various tasks such as SQL, machine learning, streaming data processing, and graph analytics. It is known for its speed and ease of use, with support for multiple programming languages including Scala, Java, Python, and R.



1.1: Apache spark

1.2. TENSOR FLOW:

TensorFlow is an open-source machine learning framework developed by Google. It provides a comprehensive ecosystem of tools, libraries, and community resources for building and deploying machine learning models. TensorFlow's core functionality revolves around the creation and training of deep learning models, particularly neural networks. It offers flexibility, scalability, and portability, enabling developers to deploy models across a variety of platforms, including mobile devices and distributed systems.



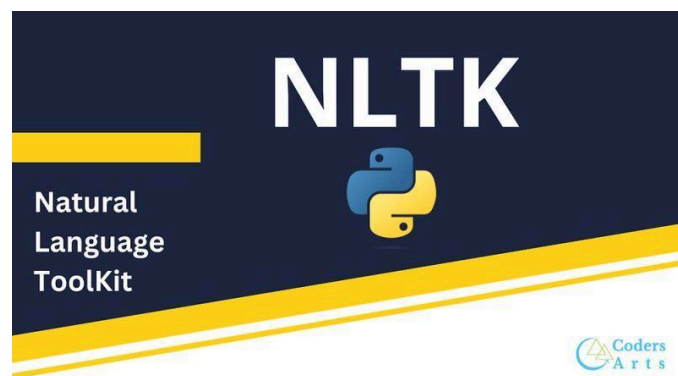
1.2: TensorFlow

1.3. PANDAS:

Pandas is an open-source data analysis and manipulation library for Python. It provides high-performance, easy-to-use data structures and data analysis tools, making it a popular choice for data scientists and analysts. Pandas offers powerful data manipulation capabilities, including data alignment, indexing, filtering, grouping, and aggregation. It also integrates well with other Python libraries, such as NumPy and Matplotlib, for data visualization and analysis.

1.4. NATURAL LANGUAGE TOOLKIT(NLTK):

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources, such as WordNet, along with a suite of text processing libraries for tasks such as tokenization, stemming, tagging, parsing, and classification. NLTK is widely used in educational and research environments for natural language processing (NLP) tasks, including text classification, sentiment analysis, and machine translation.



1.3: NATURAL LANGUAGE TOOLKIT(NLTK)

1.5. SCIKIT - LEARN:

Scikit-learn is an open-source machine learning library for Python. It provides simple and efficient tools for data mining and data analysis, built on top of NumPy, SciPy, and Matplotlib. Scikit-learn offers a wide range of supervised and unsupervised learning algorithms, including classification, regression, clustering, dimensionality reduction, and model selection. It is designed to be easy to use and accessible to non-experts, while still providing advanced functionality for experienced users.

CHAPTER 2

LITERATURE SURVEY

Smart Travel Plan Recommendation System(2022)

Authors: Vaibhav Mandale, Niraj Reddy, Amlan Sahu, Praveenkumar Patel

The internet's remarkable growth and its vast user population have ushered in an era of abundant online knowledge. This treasure trove of information proves invaluable for tailoring recommendations to users' preferences and needs. Recommender systems, pivotal in this landscape, gather user data encompassing preferences, interests, and geographical information. Their development gained prominence with the advent of collaborative filtering techniques. Meeting user preferences is a complex task, yet recommender systems harness data from diverse sources to make predictions and provide personalized item recommendations. This project aims to serve as a one-stop solution for vacation planning, offering travelers tailor-made suggestions aligned with their preferences and desires.

In the quest for a break from the mundane, what's the answer? A vacation! Yet, today, planning a getaway demands a substantial chunk of time. The extensive process involves sifting through countless hotels, attractions, and dining options, all while keeping within a budget. It's a daunting task, and finding a comprehensive source is nearly impossible. Our project aims to streamline this process, allowing travelers to spend more time enjoying their vacation. By collecting travel details and preferences, we create personalized travel plans. We recommend accommodations, activities throughout the day, and dining options to make their trip a memorable one.

Smart Tourist Guide System (2022)

Authors: Rahul Chavan, Manvee Bhoir, Gaurav Sapkale, Anita Mhatre

The Smart Tourist Guide System represents a groundbreaking open-source platform tailored for individuals embarking on journeys across India. This

application serves as a resource to help travelers plan their Indian expeditions, taking into account their preferred number of days, budget, weather considerations, and more. Moreover, it provides valuable insights into the historical background of destinations, showcases traveler reviews, and displays user ratings. The primary objective behind this project is to create a comprehensive tourist guide for Delhi, aiming to address the challenges faced by tourists due to the absence of a reliable guide. Typically, visitors to Delhi resort to hiring professional tourist guides, incurring significant expenses. Often, these guides are not full-time professionals, and their knowledge can be limited, leading to gaps in providing essential information, such as weather conditions, historical significance, and distances from sea level.

The Smart Tourist Guide offers a solution by harnessing modern technology through an Android application. It utilizes Google Maps API, Global Positioning System (GPS), and the internet. By inputting latitude and longitude, it locates destinations on a map and calculates distances between the user's current location and their desired spots. This innovation is designed to revamp the tourism industry, enhancing the travel experience while boosting the tourism sector.

An automated itinerary planning system for holiday travel(2004)

Authors: Simon Dunstall, Mark E.T. Horn, Philip Kilby, Mohan Krishnamoorthy, Bowie Owens, David Sier and Sylvie Thiebaut.

This paper introduces the Electronic Travel Planner (ETP), a prototype travel recommender system geared towards crafting itineraries for tourists. The system relies on traveler preferences and extensive databases filled with tourism and travel data. Its primary functions involve destination selection, tour and attraction recommendations, and the creation of comprehensive travel schedules. These tasks involve intricate optimization challenges, which the prototype tackles using a heuristic problem-solving approach. Computational tests validate the efficacy of these methods, suggesting the potential for automated travel planning in full-scale applications.

The ETP, a web-based prototype, emerged from research conducted between 1999 and 2001 in collaboration with Viator Inc., supported by an Australian Government grant. The system's core code resides in an Internet server, with plans for user interface and Internet connection integration. Commercial implementation discussions are ongoing.

The ETP project's core goal was to revolutionize holiday planning, capturing travelers' preferences and exploring possibilities beyond current capabilities. By offering travel agents and individuals a powerful planning tool, the project aims to enhance satisfaction in the travel planning process and the overall travel experience. These objectives align with the concept of a travel recommender system, embracing various choice models, interactive preference determination, and adaptive recommender systems based on user dialogue.

An Intelligent Destination Recommendation System for Tourists(2018)

Authors: Pree Thiengburanathum

The process of selecting a tourist destination can be daunting, both in the planning stage and during the actual trip. With the advent of recommendation systems, travelers now have the ability to swiftly compare and choose destinations. This involves constructing decision models, predicting user preferences, and interpreting the outcomes.

This research project sets out to create a Destination Recommendation System (DRS) with a strong focus on machine-learning techniques. To develop an effective DRS, an extensive review of existing literature on recommendation systems in the tourism domain was conducted. Subsequently, the thesis introduces a model-based DRS, featuring a two-step filtering method to eliminate irrelevant and redundant information and employing a Decision Tree (DT) classifier for transparency, interpretability, and efficiency in aiding tourists' decision-making. The system's scalability is validated using real-world data from a case-study city. Destination choice models were formulated and tested, demonstrating that the proposed model-based DRS can offer personalized and satisfactory

recommendations to users.

The thesis further suggests an ensemble-based DRS, incorporating weight hybrid and cascade hybrid approaches. Three classification algorithms, including DT, Support Vector Machines (SVMs), and Multi-Layer Perceptrons (MLPs), were explored. Results revealed that the bagging ensemble of MLP classifiers outperformed baseline learners and other combiners.

Lastly, the thesis introduces an Adaptive, Responsive, Interactive Model-based User Interface (ARIM-UI) for the DRS, enhancing the user experience by providing adaptive, informative, and responsive information. This interface empowers tourists to easily interact with the recommended results, thereby improving the overall quality of user engagement with the system.

2.1. INFERENCES FROM LITERATURE SURVEY

The texts highlight the evolving landscape of travel and tourism, where technology is playing a pivotal role in simplifying and enhancing the entire travel experience. In the first text, it's evident that the process of choosing a tourist destination has been historically challenging. With the introduction of recommendation systems, the ability to quickly evaluate and select destinations is now a reality. This underscores the importance of technology in facilitating decision-making by constructing models, predicting user preferences, and interpreting outcomes.

The second and third texts describe projects aimed at revolutionizing travel planning. The first project focuses on developing a Destination Recommendation System (DRS) using machine-learning techniques. It acknowledges the significance of literature review and introduces a model-based DRS with a two-step filtering approach and a Decision Tree (DT) classifier. This research aims to provide personalized and satisfactory recommendations to travelers. The proposal of an ensemble-based DRS further exemplifies the quest for superior recommendations.

The Smart Tourist Guide System, discussed in the last text, provides a comprehensive platform for travelers in India. It leverages technology, including

Google Maps API and GPS, to offer real-time, tailored guidance for travelers. It seeks to eliminate the need for traditional tourist guides, which can be expensive and may lack professionalism. This project represents a forward-looking solution, emphasizing the potential for technology to improve the tourism industry and enhance the traveler's experience.

Overall, these texts underscore the transformative role of technology in making travel planning more accessible, efficient, and tailored to individual preferences, while simultaneously addressing common challenges in the industry.

2.2. OPEN PROBLEMS IN EXISTING SYSTEM

Several open problems exist in existing travel itinerary generator systems, which can be addressed to enhance their functionality and user experience:

2.2.1. Dynamic Recommendations:

Many travel itinerary systems rely on static data, which doesn't account for real-time changes in weather, availability, or closures of attractions. Developing algorithms that can dynamically adapt itineraries based on changing circumstances is a significant challenge.

2.2.2. Multi-Destination Itineraries:

Existing systems often focus on single-destination itineraries. Creating tools that efficiently plan multi-destination trips, which involve coordination and optimization across different locations and time zones, is a complex problem.

2.2.3. User Constraints:

Most systems don't adequately handle user-specific constraints and preferences. Developing methods to integrate a wider range of traveler preferences, such as dietary restrictions, accessibility needs, or pet-friendly options, into itineraries remains an open problem.

2.2.4. Realistic Time Estimations:

Generating itineraries that accurately estimate the time needed for each activity,

including travel between locations, is a non-trivial challenge. Existing systems may not always provide realistic timing, leading to rushed or unrealistic plans.

2.2.5. Group Travel:

Creating itineraries for group travel introduces complexities, such as varying interests and schedules. Developing algorithms that cater to group dynamics and preferences is an open problem.

2.2.6. Integration with Booking:

Many travelers want a seamless experience from itinerary planning to booking. Systems that can integrate itinerary generation with booking accommodations, flights, and activities, and ensure real-time synchronization, are needed.

2.2.7. Data Accuracy and Coverage:

The quality and availability of data are critical for effective itinerary generation. Ensuring that systems have access to comprehensive, reliable, and up-to-date data for global destinations is an ongoing challenge.

2.2.8. Privacy and Data Security:

As systems gather and store personal data, addressing privacy and data security concerns is paramount. Creating systems that protect user information while delivering personalized experiences remains an open problem.

2.2.9. Scalability:

As the number of users and destinations grows, ensuring that the system remains efficient and scalable is crucial.

CHAPTER 3

AIM AND SCOPE OF THE PRESENT INVESTIGATION

3.1. AIM OF THE PROJECT:

The aim of the project is to develop the Personalized Travel Itinerary Generator System, a cutting-edge software solution designed to simplify and enhance the travel planning experience. The primary objective is to create a user-friendly application that automates the itinerary creation process, leveraging advanced algorithms to generate tailor-made travel itineraries based on individual preferences, interests, and constraints. The system aims to revolutionize travel planning by offering highly personalized and enjoyable experiences while simplifying the planning process on a global scale.

3.2. SCOPE OF THE PROJECT:

The scope of the project encompasses the design, development, and implementation of the Personalized Travel Itinerary Generator System. Key components include:

3.2.1. User Interface Design:

Designing an intuitive and user-friendly interface for travelers to input their preferences, constraints, and travel details.

Algorithm Development: Developing advanced algorithms to analyze user input, collect and process destination-related information, and generate customized travel itineraries.

3.2.2. Data Integration:

Integrating external APIs and databases to access real-time data about attractions, weather conditions, transportation schedules, and accommodation availability.

3.2.3. Flexibility and Customization:

Providing users with the flexibility to modify itineraries, adjust schedules, and incorporate personal notes to meet their unique needs.

3.2.4. Testing and Validation:

Conducting thorough testing to ensure the system's reliability, accuracy, and

performance in generating optimal travel plans.

3.2.5. Deployment and Maintenance:

Deploying the system for use by travelers and providing ongoing maintenance and support to address any issues or updates.

The project aims to deliver a comprehensive and robust software solution that transforms the travel planning process, offering convenience, personalization, and enjoyment to users worldwide.

3.3. FEASIBILITY STUDIES / RISK ANALYSIS OF THE PROJECT

3.3.1 Feasibility Study:

Feasibility studies and risk analysis play pivotal roles in project planning, particularly in evaluating the viability and potential challenges of implementing the Personalized Travel Itinerary Generator System. Technical feasibility hinges on assessing the availability of requisite technologies, expertise, and infrastructure necessary for developing advanced algorithms, integrating external APIs and databases, and creating a user-friendly interface. Financial feasibility entails estimating development, deployment, and maintenance costs, alongside potential revenue streams through detailed cost-benefit analysis. Operational feasibility examines the system's integration into existing travel planning processes, considering factors like user acceptance, training needs, and scalability for seamless operation. Risk analysis identifies potential obstacles such as technical hurdles, data privacy concerns, and user adoption issues, followed by the formulation of risk mitigation strategies including thorough testing, data encryption, and user training. Through comprehensive feasibility studies and risk analysis, the project team can make well-informed decisions to overcome challenges and ensure the successful development and implementation of the Personalized Travel Itinerary Generator System. By evaluating technical, financial, and operational feasibility, stakeholders can assess the project's viability and identify potential challenges. Additionally, conducting a thorough risk analysis enables stakeholders to proactively identify and mitigate potential risks, ensuring the successful development and implementation of the system. By addressing these critical aspects of project planning, stakeholders can increase the likelihood of project success and deliver a valuable solution to meet travelers' needs.

3.4. SYSTEM REQUIREMENTS SPECIFICATION DOCUMENT

3.4.1. Windows OS:

Microsoft's Windows OS dominates the personal computer landscape. Its familiar graphical interface with desktops, windows, and icons makes it user-friendly. Pre-installed on many PCs and laptops, it is a popular choice for home and business users alike. Supporting a vast range of software and hardware, Windows caters to various needs with different editions, from basic home use to powerful workstations. Regular updates provide new features and security patches. While generally user-friendly, some may find navigating menus and settings complex. Security concerns require users to stay updated with antivirus software and practice online caution. Despite competition, Windows 11, the latest version with a revamped interface, continues Windows' reign as a popular and widely used



operating system.

3.1: Windows OS

3.4.2 Python:

Python is a user-friendly programming language used for various tasks like web development, data analysis, and artificial intelligence. It's popular because of its easy-to-read code and versatile applications. Many beginners prefer it due to its simplicity. Python's large community means lots of support, making it a great choice for anyone learning to code or working on diverse projects. Its straightforward syntax and vast capabilities make it one of the most accessible languages for programmers of all levels.



3.2: Python

3.4.3. Jupyter Notebook:

Jupyter Notebook is an open-source interactive web application that allows users to create and share documents containing live code, equations, visualizations, and narrative text. Primarily used for data analysis, scientific computing, and machine learning projects, it supports multiple programming languages through the use of kernels, including Python, R, Julia, and Scala. Jupyter Notebooks are particularly valued for their ability to combine executable code with explanatory text and multimedia resources in a single document, facilitating comprehensive data exploration, visualization, and explanation.



3.3: Jupyter Notebook

CHAPTER 4

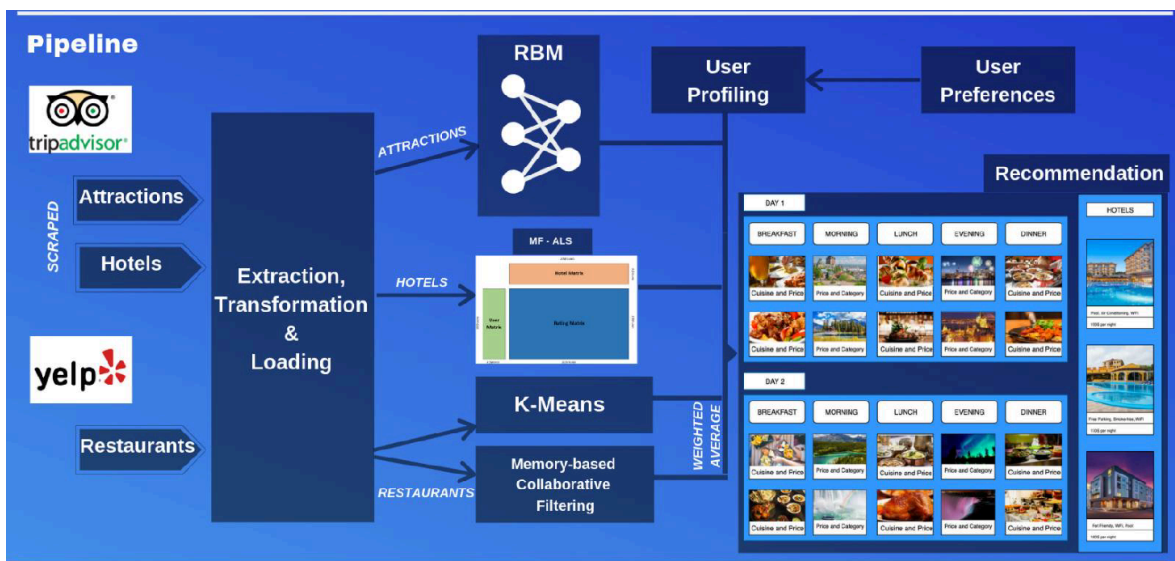
EXPERIMENTAL OR MATERIALS AND METHODS

4.1. MODULES:

The major modules of the project include:

- (i) Data collection through web scraping, cleaning and integration.
- (ii) Hotel recommendation based on user's amenity requirements .
- (iii) Attraction recommendation for Morning and Evening using categories of tours.
- (iv) Restaurant recommendation for different meals of the day using cuisine type and food preferences.
- (v) Integration of all the above mentioned recommendations in-order to provide one-stop travel plan for each day of travel.

4.2. DATA SCIENCE PIPELINE:



4.1: Complete technological pipeline of our project.

4.2.1. DATA COLLECTION:

For the project requirements, we needed datasets containing information about details of attractions, hotels and restaurants and their corresponding reviews. There was no dataset readily available for Hotels and Attractions, thus making Data Collection a very important task for our project. To obtain these datasets, we

crawled through thousands of attractions and hotels on Tripadvisor. We used lxml - a python library along with requests - another python library, for downloading and processing HTML files using ElementTree API. We had to crawl through hotels and attractions in a different manner due to unique HTML page structures of both the categories. Around 3.5k attractions and 35k hotels were collected through scraping tripadvisor. We also created a crawler for scraping the reviews for each attraction and hotel in the dataset and collected reviews close to 33K for attractions and 2.4M for hotels. Since the websites have a limit on the number of hits per day, we performed the data collection in batches. For the restaurant recommendation, we used data from YELP dataset (original). This dataset consists of data related to users, businesses (not restricted to restaurants), reviews, check-in time, user tip reviews and photos of respective

businesses. We used the yelp business, yelp reviews and the yelp photos dataset. The business dataset contained details of all the business on Yelp including restaurants, salons, gas stations, drugstores, etc., without any feature explicitly mentioning the categories of these businesses. The challenge was to extract only the restaurants from the dataset. The techniques used to do the same will be explained later in the report. We got 12K restaurants and close to 5M reviews on the yelp dataset for restaurants.

4.2.2. DATA COLLECTION AND INTEGRATION:

Having collected all the data mentioned above, next big task was cleaning the dataset and integrating them. Due to the inconsistencies in the Tripadvisor website, there were missing values for prices, ratings and details about the location for both the scraped datasets.

i. Hotels: For the hotel dataset, GeoPy - a python library to geolocate a query to an address or coordinates was used to obtain the latitude, longitude positions of a hotel from it's address. Values for which GeoPy wasn't able to output latitude, longitude coordinates from address, we tried to get coordinates based on its locality and further, based on its province. The missing prices were filled hierarchically with the average price of hotels present in a city, then province and finally the entire dataset based on availability of data. This was based on the assumption that hotels in same locality have similar pricing for competency in the market. Missing ratings were filled out in the same manner. Amenities was the

most important feature in hotels dataset and hence, for interpretability, hotel amenities (string) was split, exploded, cleaned and converted into a list of amenities. Continuous unique IDs were provided to users and continuous hotel IDs were provided to hotels.

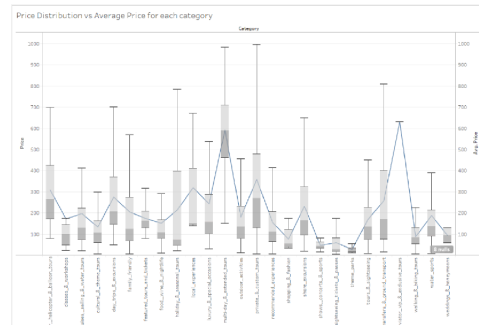
ii. Attractions: For attractions, only the priced tours and sightseeing were collected. The locations of the attractions could not be scraped from the website, and hence the address of the attraction was created from the information scraped from websites, and finally Google API was used to obtain the coordinates of an attraction. The missing coordinate values after this step was handled based on the assumption that similar categories of attractions will be present in the same locality. For example, two different water sports will be present next to each other in a city, there will be an area in the city where all the clubs will be present, etc. The location coordinates of an attraction were assumed to be the average location of attraction categories in a city. Missing prices and ratings were also averaged based on city and category of attraction.

iii. Restaurants: The first level of Data cleaning required us to obtain from the humongous yelp dataset, businesses that pertain to 'restaurants' only. This was achieved using text filtering, we filtered out the businesses for which the 'categories' column of the business dataset had text values unrelated to the food/restaurants industry. Further, we cleaned up the reviews dataset to have only those businesses that are of interest to us (restaurants) and to be aligned with the those in the business dataset. Also, we have removed restaurants that have been marked as closed in the dataset, as recommending 'not-in business' restaurants doesn't make sense. The categories and attributes columns were of utmost important for us from the business dataset. Both of theses columns were textual features and hence, extensive cleaning had to be done to convert them- 'categories' to lists and 'attributes' to dict of dict. We also computed sentiment score, to find out how positive a review for a restaurant is, in-order to provide better personalized recommendations to our users, the details of how we achieved this will be talked about later in this report. Finally, we had to maintain the index (using `reset_index`) of the dataframe as we had used pandas for restaurants dataset- rather than assigning a continuous row ID for businesses and reviews dataset, we chose to exploit the in-built features of pandas. A very common problem in providing recommendations to users is the cold-start problem. That is,

providing recommendation to a new user who does not exist in the database.

4.2.3. EXPLORATORY DATA ANALYSIS(EDA):

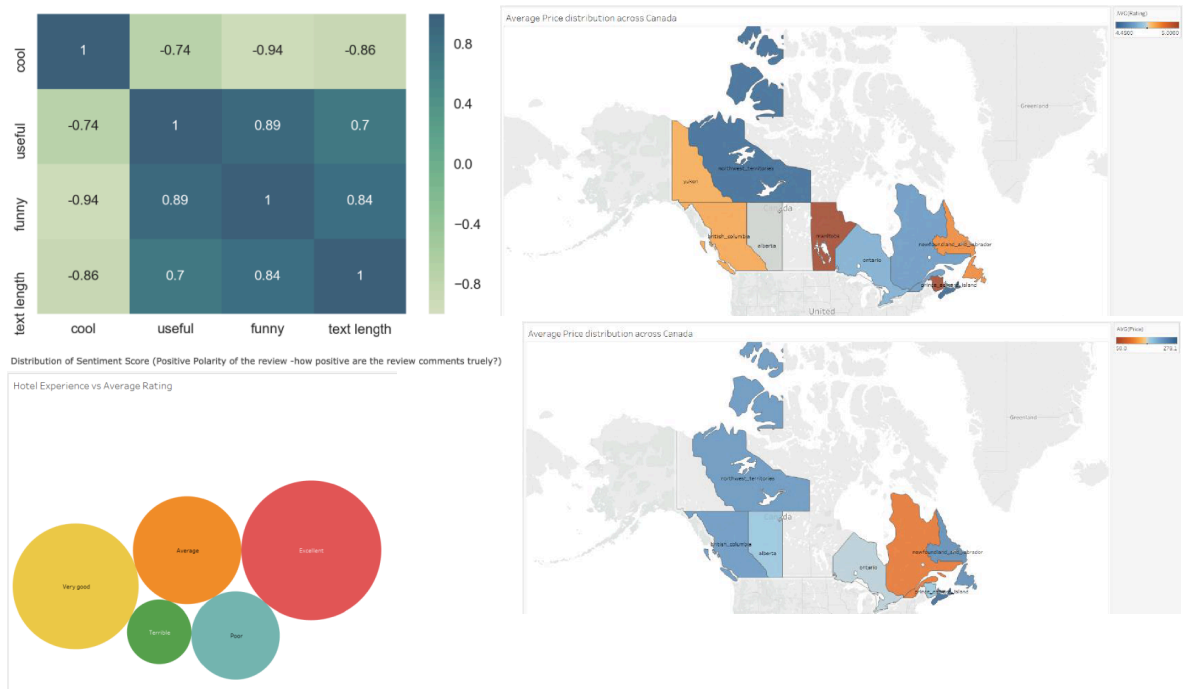
In order to have a better understanding of the data collected and integrated for recommendations, we performed extensive exploratory data analysis on various features of each dataset. This step was highly significant in helping us understand the distribution of prices and ratings across different categories



and other attributes in the dataset. EDA helped us arrive at the decision for using specific attributes from each dataset for user profiling by explaining variance in the data with respect to different attributes of the dataset. The following is the list of some of the important analysis that led us to the decisions we made:

- (i) Average rating for different categories of attraction
- (ii) Rating distribution for attraction categories
- (iii) Average Price and rating distribution across different provinces of Canada
- (iv) Price distribution vs Average price for attraction categories
- (v) Province-wise and city-wise number of attractions
- (vi) Distribution of poorly and highly rated restaurants province-wise
- (vii) Rating distribution for restaurants province-wise
- (viii) Number of restaurants province-wise and city-wise
- (ix) Sentiment Analysis on user reviews and user-tip reviews was performed to better understand the underlying restaurants dataset. We were able to identify that the dataset was highly skewed towards positive reviews. So, we computed the positive to negative ratio and normalized the score across the entire dataset, in-order to understand how truly positive a user review is indeed. The challenge here was that reviews and tips were sometimes written in French. Thus, we first used google translator API to convert them to english sentences before feeding them into VADER sentiment analyzer, which is a rule-based system that is explicitly designed for English language. We have used the compound score metric from VADER to compute the positive-negative ratio. So, we computed the positive to negative ratio and normalized the score across the entire dataset, in-order to understand how truly positive a user review is indeed. So, we computed the positive to negative ratio and normalized the score across the entire dataset,

in-order to understand how truly positive a user review is indeed.



4.2: Few Visualizations we created during EDA

4.2.4. INPUT AND USER PROFILING:

User Name

Nithya

Start Date

09 - 04 - 2024

Destination

toronto

End Date

11 - 04 - 2024

Budget

5 - 555

4.3: Travel details of user

To provide the travel plan to the user, we obtain information about user’s travel dates, their budget and their attraction, hotel and restaurant preferences. One of the major challenges in our project was to create a profile for user based on his preferences, A very common problem in providing recommendations to users is the cold-start problem. That is, providing recommendation to a new user who does not exist in the database. For a new user, there is no information on what he likes to provide him with recommendations. If we have a website, we can take into account the user’s interaction with the website to provide implicit recommendations, but even then, there is no possibility of providing him recommendations he will like before his interactions. In order to solve this problem, we need to profile the user understanding his preferences. Since we have 3 categories of recommendations and we used 3 different for the same, we had to

use different user profiling techniques:

i. Hotels: From EDA, we understood that hotel amenities is the major feature to consider for providing recommendations. We exploded the list containing amenities of a hotel and found top 15 amenities that most hotels have. The user was provided with these amenities to choose from and based on the amenities he chose, hotels were rated on a scale of 5 with rating 5 indicating the hotel containing maximum of user's required amenities. A 10% of this dataset was randomly sampled and combined with the original dataset containing ratings and reviews on hotels. The combined dataset was then passed to the model for training. This user profiling enabled us to match the new user with the closest user in the existing database who had liked similar hotel preferences.

Select atleast 5 amenities:

Nonsmoking hotel	Refrigerator in room	Wheelchair Access
Nonsmoking rooms	Family Rooms	Microwave
Free High Speed Internet W/F	Public Wifi	Breakfast included
Air conditioning	Pets Allowed Dog Pet Friendly	Laundry Service
Free parking	Suites	Kitchenette

Nonsmoking hotel
Select 4 more!

Nonsmoking rooms
Select 3 more!

4.4: Attraction category rating

ii. Attractions: Travelers have unique preferences in visiting different categories of attractions. For example, a travel group might like to visit beaches, water and adventure sports. Another group of travelers might like to visit museums, parks and sightseeing. This observation was leveraged to perform user profiling for Attraction recommendations. 20 Most frequent categories of tours were taken and provided as options to users. User was asked to select and rate at least five of the provided categories. This information then used to match the new user with a user in existing database sharing similar preferences, in essence, rated attracted categories in similar fashion. The recommendation obtained for the matched user

in the database was provided to the new user as his attraction recommendations.

Select and rate atleast 5 categories and rate them:

private_&_custom_tours	food_wine_&_nightlife	day_trips_&_excursions	luxury_&_special_occasions	air_helicopter_&_balloon_tours
tours_&_sightseeing	family_friendly	water_sports	sightseeing_tickets_&_passes	holiday_&_seasonal_tours
recommended_experiences	cultural_&_theme_tours	cruises_sailing_&_water_tours	multi-day_&_extended_tours	shows_concerts_&_sports
outdoor_activities	walking_&_biking_tours	transfers_&_ground_transport	shore_excursions	classes_&_workshops

private_&_custom_tours

Rate 0

Rate 4 more!

4.5: Hotel amenity preferences

iii. Restaurants: People differ widely! For restaurant based user-profiling, we consider the cuisine type, budget and food preferences (like- vegan, gluten-free, etc.,) of the users. We give them a wide variety of popular cuisine types to choose from (15 different options). This information is then used to build the user-profile for an unknown user and three different personalized recommendations lists are provided, one each for each meal of the day. The closest item (restaurants) match to the attributes provided by the user is recommended. After User profiling, the models are trained and evaluated to obtain best performing models. Then the recommendations are filtered and integrated into an Interactive IPython Notebook.

Cuisine type

☐ Asian	☒ Irish	☐ African
☐ Japanese	☒ French	☐ Indian
☒ Pubs	☐ Fusion	☐ Pakistani
☐ Sports Bar	☒ Mexican	☐ Italian
☐ Mediterranean	☒ Canadian	☐ Chinese

Preferences

☒ Vegetarian
☐ Vegan
☐ Halal
☐ Gluten Free

Price Range

☐ Budget
☒ Luxury

4.6: Restaurant cuisine type and preferences

CHAPTER 5

IMPLEMENTATION DETAILS

5.1. MODEL TRAINING METHODOLOGY AND EVALUATION:

As mentioned earlier, 3 different models were used to obtain the recommendations. In this section, we will discuss about the models used in detail, the input for the model, training and evaluation of the model.

5.1.1. Matrix Factorization using Alternating least Squares technique (MF-ALS):

Matrix factorization is one of the most used Collaborative Filtering technique after Netflix Prize Challenge. Matrix Factorization uses the user-item interaction matrix to learn the embedding among the interaction to predict the ratings that will be given by users to unseen/ unused items. Alternating Least Squares (ALS), a matrix factorization algorithm, runs in a parallel fashion. It runs the gradient descent of multiple partitions of training data in parallel. It is also highly scalable and thereby used for large collaborative filtering tasks. It is simple and deals with solving the sparseness issue of the ratings data. Also, ALS can be used in distributed file systems.

INPUT: The hotel data containing information about user, hotel and user's rating for the hotel was used as input for training the MF-ALS model. The dataset also contains 10% sample of the new user data created using his preferences of hotel amenities. The remaining 90% sample used to obtain recommendations for new user.

PARAMETERS: The ALS model was trained with three different ranks (i.e.) 4, 8, 12. Ranks are also known as the latent factors. Latent factors are the features projected from user-item interaction in the lower dimension latent space. To represent user preferences in lower dimension space, latent factors are used. The ALS model was trained with three different ranks, maximum number of iterations as 5, regularization parameter as 0.01 and three attributes namely user id, hotel id and the rating given by user for the hotel.

EVALUATION: Root Mean Square Error (RMSE) was calculated for each case with different ranks and the model with lowest RMSE was used for

recommendations. RMSE is calculated by finding the difference between predicted and existing rating, squaring them and then finding the square root of that value. The model with least RMSE denotes that the user-item interaction embedding has been learnt better and the model is able to predict user ratings to unseen items more accurately. RECOMMENDATION: The saved model was used for predictions. Based on user preference of amenities, ratings were provided to hotels. 10% of the sample data containing user preferences was combined with the actual training data. Remaining 90% of the sample data was used for predictions. The data used for predictions contains the hotels not visited by the user. The trained ALS model is used to predict the ratings that the user will give for the hotels they haven't visited.

5.1.2. Restricted Boltzmann Machine (RBM):

Boltzmann machines (BM) are stochastic and generative neural networks capable of learning internal representations and are able to represent and solve difficult combinatorial problems. RBM's are non-deterministic Deep Learning models with only two types of nodes - visible and hidden nodes. A regular Boltzmann machine has connections between all hidden nodes and visible nodes and amongst themselves. Restricted Boltzmann Machine is a special class of BMs and is restricted in terms of the connection within the same category of nodes and is fully bipartite. There are two other layers of bias units (hidden bias and visible bias) in an RBM. The hidden bias of RBM produces the activation on the forward pass and the visible bias helps RBM to reconstruct the input during a backward pass. The reconstructed input is always different from the actual input as there are no connections among the visible units and therefore, no way of transferring information among themselves. The model is trained to achieve the input after the reconstruction phase.

The inputs are multiplied by weights and hidden bias is added to them. This activation is then passed into a sigmoid activation function. This is the forward-pass. During the backward-pass or reconstruction phase, the activation function outputs act as the new input, multiplied by weights, added to the visible bias and then passed through the sigmoid activation function. The output of the reconstruction phase is then compared with the input to the forward-pass to calculate reconstruction error. The model is trained by reducing the reconstruction

error using a gradient-based Contrastive Divergence algorithm. Contrastive Divergence is the difference between two Kullback-Leibler divergences of the input and reconstructed output. The algorithm tries to bridge the gap between the input and the reconstructed output while updating the weights to do so.

INPUT: The attractions dataset was used to create the user-item interaction matrix, also known as rating matrix. The rating matrix contains n rows for users and m columns for each attraction present in the dataset. The ratings provided by the user i for attraction j was filled at (i,j) position of the matrix. The positions without any user-item interaction was filled using a 0 rating. To avoid overfitting, the dataset was split for training and validation and each row of the matrix was then passed to the visible layer of the RBM model for training with weights, hidden bias and visible bias initialized randomly.

PARAMETERS: The model was trained for different combinations of parameters including number of epochs, learning rates, batch size, number of rows for training and number of hidden units. Trained best model for every combination tried was store for future usage.

EVALUATION: The model performance was evaluated using the Contrastive Divergence algorithm and the model with the best performance, i.e., least divergence between input and reconstructed output at visible layer was saved. The model with lesser divergence from output suggests that RBM has learnt well the embeddings between user and the attraction interactions so that it can replicate the input row.

RECOMMENDATION: The saved model was used to obtain recommendations for all the users in the existing dataset. Then the traveler is matched with the closest user in the dataset based on attraction category preferences and the recommendations are provided obtained for the traveler.

5.1.3. K-Means + Memory based Collaborative Filtering Hybrid technique:

We have used K-Means as content based filtering technique and K-Nearest Neighbors as a memory based Collaborative Filtering technique .K-Means is a clustering technique that uses Euclidean distance in vector-space to measure the proximity of two points (in our case two different restaurants). The algorithm learns to classify the features in vector-space based on the similarity of input features into different clusters, which will be a $M \times N$ matrix, where M denotes the number of

items and N denotes the number of features. It's an unsupervised machine learning technique as the algorithm does not have truth values (labels) to learn from and correct itself. Being a Content Based Filtering Technique, K-Means does not suffer from cold start problem, but is highly sensitive to outliers in the dataset and dimensionality reduction techniques can help make the algorithm more efficient. Thus, we carefully performed EDA to analyze the restaurants dataset, we computed Positive-Negative sentiment ratio per restaurant based on user reviews. This helped weigh down the fact that the dataset was highly skewed towards positive reviews and tips. Also, we performed Principal Component Analysis to reduce the dimensionality, however, we did not scale our features as our input features did not have a linear relationship- scaling these would result in losing important semantic similarities in words. In-order to find the number of components apt for our dataset using PCA, we plotted the variance in features with respect to the number of components and chose the smallest number of components for which the variance was not less than 95%. KNN, for our project, is used as a classifier that helps classify the user-item interaction matrix based on ratings provided. But, unlike the other two techniques, this is an Item-Based collaborative filtering technique as it predicts ratings for unknown users based on items that have obtained similar ratings to the one user is looking for rather than finding a user who is a close match to the current user. This is advantageous as it scales much better than the User-based Collaborative Filtering technique. It scales better because Item features tend to be more stable over user features, user taste changes more dynamically and hence, is very hard to rely on. But, a drawback is that Item-Based techniques will suffer due to the Curse of Dimensionality. Also, these do not suffer from cold-start problem like User-based Collaborative Filtering techniques.

INPUT: For content based approach-the restaurants are filtered based on destination before running K-Means on the dataset. Categories, Attributes, Sentiment score, Star rating, City and State were used as the input features for K-Means algorithm. The Word2Vec model was trained for the vocabulary of words we have obtained after feature extraction and was saved for future use. Also, we had concatenated the input features into a single sentence, thus we had to average out the Word2vec vector obtained per word for each sentence (one sentence per restaurant in the dataset). In-order to obtain three different

recommendations for different meals of the day, we obtained predictions for three different inputs from K-Means simultaneously with appropriate keywords to indicate the meal of the day (breakfast, lunch or dinner) for which we are seeking the recommendation. For KNN algorithm, the restaurants dataset was used to create the user-item interaction matrix, also known as rating matrix. The rating matrix contains n rows for users and m columns for each restaurant present in the dataset. The ratings provided by the user i for attraction j was filled at (i,j) position of the matrix. The positions without any user-item interaction was filled using a 0 rating.

PARAMETERS: The hyper-parameter K for K-means algorithm represents the optimal number of clusters into which the dataset could be divided into. In-order to find the best K value, we iteratively ran K-Means on the entire dataset for two to fifteen times and each time calculated the inertia and silhouette score. KNN takes in the input sample and labels to train on with a parameter to choose the weights you want to assign during prediction to the points and a parameter to say how many recommendations (K) aka nearest neighbors of a point you want the algorithm to predict. Also, you can choose the algorithm and metric that you want to use. We used brute force KNN and cosine similarity as the metric respectively.

EVALUATION: For evaluating K-Means both the Elbow Method and Silhouette Analysis were performed. The best K value (number of clusters) was obtained based on these metrics. Elbow Method: The Elbow curve at a point will have a sharp change, that is the point in the curve (in increasing k value) after which the decrease in Inertia with respect to K isn't substantial gives us the optimal K value. At this optimal value, the curve looks to have an Elbow shape.

RECOMMENDATION: For predictions on new points (new users), we use K-means to assign a cluster to the new point based on the user preference. Then, we find the points within the assigned cluster that are closest to our new point using 'Cosine Similarity'. We go for cosine over euclidean as the input features in our case are word embeddings obtained from Word2Vec, whose semantics will be better captured by using cosine similarity (euclidean will calculate the mere difference in letters of the words). The final recommendation list is a weighted average (based on rating) of the recommendations obtained from the content-based (K-Means) technique and the Memory-Based Collaborative Filtering technique (K-Nearest Neighbors).

CHAPTER 6

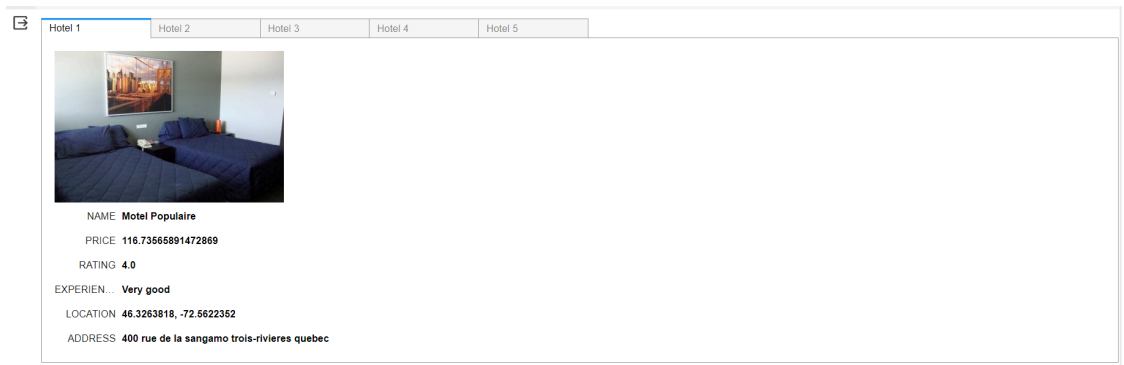
RESULTS AND DISCUSSION

6.1. OVERVIEW OF THE PROJECT RESULT:

The data product for the project was an Interactive IPython Notebook created using ipywidgets – an Ipython library for creating widgets in a notebook. Inputs from traveler containing travel details, budget, their hotel, attraction and restaurant preferences were obtained through interactive widgets as shown in figures 4a, 4b, 4c. The final recommendation for each category of recommendation was handled differently and finally integrated into one single Data Product.

6.1.1. HOTELS:

After obtaining recommendations from the Matrix Factorization model, the result was filtered based on budget and location. The price was multiplied for the number of days of stay and result was sorted and filtered. Details of 5 hotels with top recommendation scores were provided to the traveler as Hotel recommendations to choose from. The result was consistent with the user's amenity requirements. The final output for hotel recommendation is shown in figure 5a.

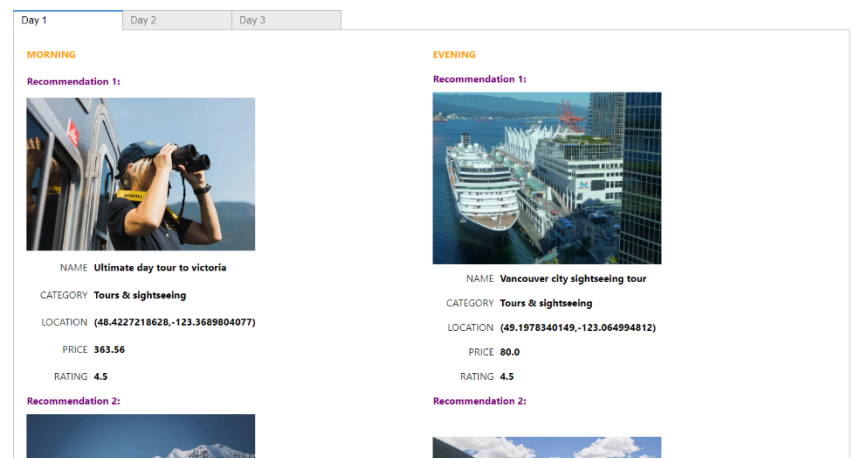


5.1: Hotel recommendations

6.1.2. ATTRACTIONS:

The result from RBM was filtered based on location and budget. Challenges in providing this recommendation was taking into account distance between attractions and the time of day to visit the attraction. Few attractions and tours will be better for morning visits and few tours like night trekking or watching a sunset

would be better to go for in the evening. To solve this problem, NLTK’s Wordnet was used to find the synonyms of evening and night, and these keywords were checked for in the name of the attraction. This enabled us to distinguish Morning and Evening attractions. Secondly, a person going to an attraction in North

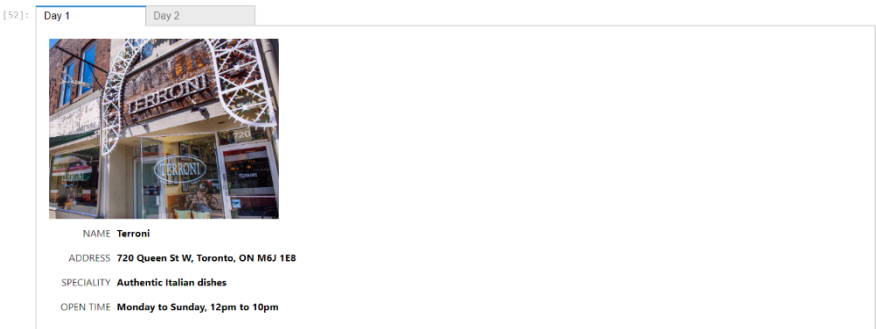


Vancouver would not want to travel all the way to Surrey to visit the second attraction. Hence, distance between consecutive visits for attractions had to be considered. The top recommendation with highest score was chosen, distance was calculated between every attraction to our selection and the closest attraction with next highest recommendation score was chosen for next recommendation. The list of recommendations provided were kept track of so that the recommendations were not repeated. This process repeats for all the days for which the traveler will be staying at the destination and the final output will be provided as shown in figure 5b.

5.2: Attraction recommendations

6.1.3. RESTAURANTS:

After obtaining recommendations (weighted average of the both the algorithms output) using the Hybrid technique, the results are de-duplicated to hold only distinct restaurants for the three different meals (Breakfast, Lunch and Dinner) of a day for the entire duration of travel. It’s also sorted based on the star rating of the recommended restaurant. We recommend two restaurants per meal per day as in figure 5c.



5.3: Restaurant recommendations

CHAPTER 7

SUMMARY AND CONCLUSION

7.1. CONCLUSION:

The project successfully met its goal of creating customized travel plans tailored to users' preferences using advanced Data Science techniques, offering recommendations based on a thorough analysis. However, considering the project's broad applicability, significant opportunities for enhancement and expansion exist. Currently, the dataset encompasses only Canadian destinations, sourced from TripAdvisor. Expanding this dataset to include global destinations would broaden the project's utility, offering a more diverse range of travel recommendations. Moreover, incorporating data from additional sources could enrich the recommendations provided.

7.2. FUTURE WORKS:

Future iterations could also transition from an interactive IPython notebook to a more user-friendly web application, enabling a more engaging user experience and the ability to refine recommendations based on user interactions. Additionally, personalizing recommendations to consider the travelers' composition, such as families with children or solo travelers, could significantly enhance the relevance of suggestions. Lastly, the exploration and integration of alternative recommendation algorithms, including Deep Belief Networks and ensemble methods, promise further improvements in the precision of travel recommendations. Overall, by addressing these areas for development, the project has the potential to evolve into an even more sophisticated tool for personalized travel planning.

REFERENCES: -

- [1] Thiengburanathum, Pree. An intelligent destination recommendation system for tourists. Diss. Bournemouth University, 2018.
- [2] Dunstall, Simon, et al. "An automated itinerary planning system for holiday travel." *Information Technology & Tourism* 6.3 (2003): 195-210.
- [3] Vaibhav Mandale, Niraj Reddy, Amlan Sahu, Praveenkumar Patel. "Smart Travel Plan Recommendation System" (2022).
- [4] Chavan, Rahul, et al. "Smart Tourist Guide System." *Engpaper Journal* (2023).
- [5] Nguyen, Andrew, and Osama Shoubber. "Trippit: An Optimal Itinerary Generator." (2019).
- [6] Meng, Jian, and Neng Xu. "A mobile tourist guide system based on mashup technology." *The 2nd International Conference on Information Science and Engineering*. IEEE, 2010.
- [7] Shi, Xiaoyu, et al. "Tour-guide: Providing location-based tourist information on mobile phones." *2010 10th IEEE International Conference on Computer and Information Technology*. IEEE, 2010.
- [8] Yu, Yao Hua. "Study on intelligent augmented reality tourist guide application based on android smart phone." *Applied Mechanics and Materials* 668 (2014): 1399-1402.
- [9] Sadiku, Peter O., et al. "Design and implementation of an android based tourist guide." *International Journal of Modern Hospitality and Tourism* 1.1 (2019): 1-33.
- [10] Nugraha, Nur Budi. "Mobile application development for smart tourist guide." *Advanced Science Letters* 23.3 (2017): 2475-2477.

APPENDIX

A.SOURCE CODE

PYTHON CODE:

Hotel:

```
# lrm -rf /content/*
pip install --upgrade --no-cache-dir gdown

#https://drive.google.com/file/d/1JV6yAhQaz9CCwQ_p3gwFtxQHnUsg056/view?usp=drive_link
!gdown https://drive.google.com/uc?id=1JV6yAhQaz9CCwQ_p3gwFtxQHnUsg056

Requirement already satisfied: gdown in /usr/local/lib/python3.10/dist-packages (4.7.3)
Collecting gdown
  Downloading gdown-5.1.0-py3-none-any.whl (17 kB)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-packages (from gdown) (4.12.3)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from gdown) (3.13.3)
Requirement already satisfied: requests[socks] in /usr/local/lib/python3.10/dist-packages (from gdown) (2.31.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from gdown) (4.66.2)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-packages (from beautifulsoup4->gdown) (2.5)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (2024.2.2)
Requirement already satisfied: PySocks<1.5.7,>=1.5.6 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (1.7.1)
Installing collected packages: gdown
  Attempting uninstall: gdown
    Found existing installation: gdown 4.7.3
    Uninstalling gdown-4.7.3:
      Successfully uninstalled gdown-4.7.3
Successfully installed gdown-5.1.0
Downloading...
From (original): https://drive.google.com/uc?id=1JV6yAhQaz9CCwQ\_p3gwFtxQHnUsg056
From (redirected): https://drive.google.com/uc?id=1JV6yAhQaz9CCwQ\_p3gwFtxQHnUsg056&confirm=t&uid=5fd7d33e74a3-409e-9dd8-487f18c8110e
To: /content/travel_recc.zip
100% 175M/175M [00:02<00:00, 77.8MB/s]

unzip /content/travel_recc.zip
lrm -rf /content/travel_recc.zip

Archive: /content/travel_recc.zip
  inflating: attraction_etl.ipynb
  inflating: attractions_crawler.ipynb
  inflating: attractions_details_crawler.ipynb
  inflating: attractions_recc.py
  inflating: combine_batches.ipynb
  creating: downloads/
  creating: downloads/rockies classic 4-day summer tour/
  inflating: downloads/rockies classic 4-day summer tour/Image_1.jpg
  inflating: downloads/rockies classic 4-day summer tour/Image_2.jpg
  inflating: downloads/rockies classic 4-day summer tour/Image_3.jpg
  inflating: downloads/rockies classic 4-day summer tour/Image_4.jpg
  inflating: downloads/rockies classic 4-day summer tour/Image_5.jpg
  creating: downloads/4-day yukon aurora viewing tour/
  inflating: downloads/4-day yukon aurora viewing tour/Image_1.jpg
  inflating: downloads/4-day yukon aurora viewing tour/Image_2.jpg
  inflating: downloads/4-day yukon aurora viewing tour/Image_3.jpg
  inflating: downloads/4-day yukon aurora viewing tour/Image_4.jpg
  inflating: downloads/4-day yukon aurora viewing tour/Image_5.jpg
  creating: downloads/Gite Le Vallon/
  inflating: downloads/Gite Le Vallon/Image_1.jpg
  inflating: downloads/Gite Le Vallon/Image_2.jpg
  creating: downloads/60 minute private vacation photography session with local photographer in prague/
  inflating: downloads/60 minute private vacation photography session with local photographer in prague/Image_1.jpg
  inflating: downloads/60 minute private vacation photography session with local photographer in prague/Image_2.jpg
  creating: downloads/120 minute private vacation photography session with local photographer in santorini/
  inflating: downloads/120 minute private vacation photography session with local photographer in santorini/Image_1.jpg
  inflating: downloads/120 minute private vacation photography session with local photographer in santorini/Image_2.jpg
  inflating: downloads/120 minute private vacation photography session with local photographer in santorini/Image_3.jpg
  inflating: downloads/120 minute private vacation photography session with local photographer in santorini/Image_4.jpg
  inflating: downloads/120 minute private vacation photography session with local photographer in santorini/Image_5.jpg
  creating: downloads/120 minute private vacation photography session with local photographer in santorini/.ipynb_checkpoints/
  inflating: downloads/120 minute private vacation photography session with local photographer in santorini/.ipynb_checkpoints/Image_
  creating: downloads/private niagara falls tour in a suv/
  inflating: downloads/private niagara falls tour in a suv/Image_2.jpg
  inflating: downloads/private niagara falls tour in a suv/Image_1.jpg
  creating: downloads/overnight camping and river-rafting trip in squamish/
  inflating: downloads/overnight camping and river-rafting trip in squamish/Image_1.jpg
  inflating: downloads/overnight camping and river-rafting trip in squamish/Image_2.jpg
  creating: downloads/ganga river/
  inflating: downloads/ganga river/Image_1.jpg
  inflating: downloads/ganga river/Image_2.jpg
  creating: downloads/full-day niagara ice wine festival tour with lunch and winery visits/
  inflating: downloads/full-day niagara ice wine festival tour with lunch and winery visits/Image_1.jpg
  inflating: downloads/full-day niagara ice wine festival tour with lunch and winery visits/Image_2.jpg
  creating: downloads/private arrival transfer: calgary international airport to banff/
```

```

inflating: downloads/private arrival transfer: calgary international airport to banff/Image_1.jpg
inflating: downloads/private arrival transfer: calgary international airport to banff/Image_2.jpg
creating: downloads/vancouver island atv tour/
inflating: downloads/vancouver island atv tour/Image_1.jpg
inflating: downloads/vancouver island atv tour/Image_2.jpg
creating: downloads/vancouver island atv tour/.ipynb_checkpoints/
inflating: downloads/vancouver island atv tour/.ipynb_checkpoints/Image_1-checkpoint.jpg
inflating: downloads/vancouver island atv tour/.ipynb_checkpoints/Image_2-checkpoint.jpg
creating: downloads/60 minute private vacation photography session with local photographer in paris/
inflating: downloads/60 minute private vacation photography session with local photographer in paris/Image_2.jpg
inflating: downloads/60 minute private vacation photography session with local photographer in paris/Image_1.jpg

```

```
!pip install pyspark==3.5.0 google_images_download
```

```

Collecting pyspark==3.5.0
  Downloading pyspark-3.5.0.tar.gz (316.9 MB)
    316.9/316.9 MB 2.5 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Collecting google_images_download
  Downloading google_images_download-2.8.0.tar.gz (14 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.10/dist-packages (from pyspark==3.5.0) (0.10.9.7)
Collecting selenium (from google_images_download)
  Downloading selenium-4.19.0-py3-none-any.whl (10.5 MB)
    10.5/10.5 MB 73.1 MB/s eta 0:00:00
Requirement already satisfied: urllib3[socks]<3,>=1.26 in /usr/local/lib/python3.10/dist-packages (from selenium->google_images_download)
Collecting trio==0.17 (from selenium->google_images_download)
  Downloading trio-0.25.0-py3-none-any.whl (467 kB)
    467.2/467.2 kB 37.3 MB/s eta 0:00:00
Collecting trio-websocket==0.9 (from selenium->google_images_download)
  Downloading trio_websocket-0.11.1-py3-none-any.whl (17 kB)
Requirement already satisfied: certifi>=2021.10.8 in /usr/local/lib/python3.10/dist-packages (from selenium->google_images_download) (2024.7.4)
Requirement already satisfied: typing_extensions>=4.9.0 in /usr/local/lib/python3.10/dist-packages (from selenium->google_images_download) (4.9.0)
Requirement already satisfied: attrs>=23.2.0 in /usr/local/lib/python3.10/dist-packages (from trio==0.17->selenium->google_images_download) (24.3.0)
Requirement already satisfied: sortedcontainers in /usr/local/lib/python3.10/dist-packages (from trio==0.17->selenium->google_images_download) (2.4.0)
Requirement already satisfied: idna in /usr/local/lib/python3.10/dist-packages (from trio==0.17->selenium->google_images_download) (3.10.3)
Collecting outcome (from trio==0.17->selenium->google_images_download)
  Downloading outcome-1.3.0.post0-py2.py3-none-any.whl (10 kB)
Requirement already satisfied: sniffio>=1.3.0 in /usr/local/lib/python3.10/dist-packages (from trio==0.17->selenium->google_images_download) (1.3.1)
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-packages (from trio==0.17->selenium->google_images_download) (1.2.2)
Collecting wsproto==0.14 (from trio-websocket==0.9->selenium->google_images_download)
  Downloading wsproto-1.2.0-py3-none-any.whl (24 kB)
Requirement already satisfied: pysocks<1.5.7,<2.0,>=1.5.6 in /usr/local/lib/python3.10/dist-packages (from urllib3[socks]<3,>=1.26->selenium->google_images_download) (1.5.7)
Collecting h11<1,>=0.9.0 (from wsproto==0.14->trio-websocket==0.9->selenium->google_images_download)
  Downloading h11-0.14.0-py3-none-any.whl (58 kB)
    58.3/58.3 kB 7.5 MB/s eta 0:00:00
Building wheels for collected packages: pyspark, google_images_download
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-3.5.0-py2.py3-none-any.whl size=317425344 sha256=6e6b87ad50254c5376de639ca8b23b148c1b282ca
  Stored in directory: /root/.cache/pip/wheels/41/4e/10/c2cf2467f71c678cfc8a6b9ac9241e5e44a01940da8fbb17fc
  Building wheel for google_images_download (setup.py) ... done
  Created wheel for google_images_download: filename=google_images_download-2.8.0-py2.py3-none-any.whl size=14534 sha256=78124be4041048dc
  Stored in directory: /root/.cache/pip/wheels/5b/f2/64/0eebc22fee5cbc0321d332492ddcc45b03c6373b5616ee95c8
Successfully built pyspark google_images_download
Installing collected packages: pyspark, outcome, h11, wsproto, trio, trio-websocket, selenium, google_images_download
Successfully installed google_images_download-2.8.0 h11-0.14.0 outcome-1.3.0.post0 pyspark-3.5.0 selenium-4.19.0 trio-0.25.0 trio-websocket-0.11.1

```

```

import pandas as pd
import ipywidgets as w
from IPython.display import display, IFrame
import pyspark
from pyspark.sql import SQLContext, functions, types
from pyspark.sql import Row
import matplotlib.pyplot as plt
from hotel_recc import *
%matplotlib inline

```

```

sc=pyspark.SparkContext(appName="hotel_recc")
spark = SQLContext(sc)

```

```

/usr/local/lib/python3.10/dist-packages/pyspark/sql/context.py:113: FutureWarning: Deprecated in 3.0.0. Use SparkSession.builder.getOrCreate().warn()

```

User details and Preferences

```

## Ipywidgets to get user name and destination
name = w.Text(description="User Name")
place = w.Text(description="Destination")
v1 = w.VBox([name,place])

## Ipywidgets to get start and end date of the trip

start = w.DatePicker(description='Start Date',disabled=False)
end = w.DatePicker(description='End Date',disabled=False)
v2 = w.VBox([start,end])

out = w.HBox([v1,v2])
display(out)

## Reading file containing hotel details after removing duplicates
del_dup = spark.read.json('etl/del_dup')

## Reading file containing hotel details after removing duplicates and exploding amenities
newh_df = spark.read.json('etl/newh_df')

del_dup.createOrReplaceTempView('del_dup')
newh_df.createOrReplaceTempView('newh_df')

## Finding top 15 amenities to ask users to select inorder to provide hotel recommendations based on amenities chosen
newh1_df = spark.sql("SELECT amenities,COUNT(amenities) AS tot_count FROM newh_df GROUP BY amenities ORDER BY tot_count DESC")
top_amenities = [x[0] for x in newh1_df.head(15) if x[0] != '']

## Getting user's amenity preferences
amenities_pref = []
def on_button_clicked(b):
    if b.description in amenities_pref:
        return
    print(b.description)
    amenities_pref.append(b.description)
    if(len(amenities_pref) < 5):
        print("Select {x} more!\n".format(x=5-len(amenities_pref)))

but_layout = w.Layout(width='100%', height='100px')
but_items = [w.Button(description=c, layout=but_layout) for c in top_amenities]
on_clk = [item.on_click(on_button_clicked) for item in but_items]
r1 = w.VBox([i for i in but_items[:5]])
r2 = w.VBox([i for i in but_items[5:10]])
r3 = w.VBox([i for i in but_items[10:]])
buttons = w.HBox([r1,r2,r3])
print("Select atleast 5 amenities:")
display(buttons)
Select atleast 5 amenities:

```

Nonsmoking hotel	Refrigerator in room	Wheelchair Access
Nonsmoking rooms	Family Rooms	Microwave
Free High Speed Internet WiFi	Public Wifi	Breakfast included
Air conditioning	Pets Allowed Dog Pet Friendly	Laundry Service
Free parking	Suites	Kitchenette

Nonsmoking hotel
Select 4 more!

Family Rooms
Select 3 more!

Public Wifi
Select 2 more!

Pets Allowed Dog Pet Friendly
Select 1 more!

Breakfast included

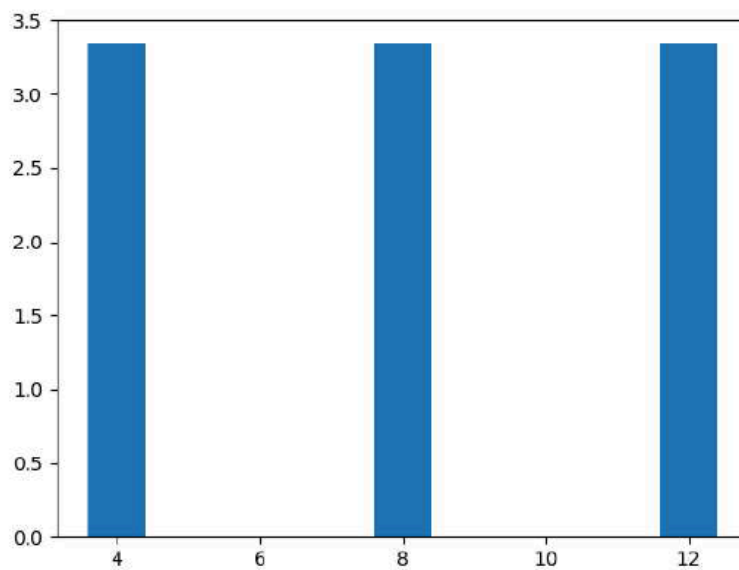
```
amenities_pref
```

```
[' Nonsmoking hotel',  
 ' Family Rooms',  
 ' Public Wifi',  
 ' Pets Allowed Dog Pet Friendly ',  
 ' Breakfast included']
```

Model training and evaluation

```
usr_rating = amenities_rating(spark, amenities_pref, newh_df)  
rank, error, errors, usrid_s2 = model_train(spark, usr_rating)  
print("best rank : ",rank)  
print("best RMSE:" +str(error))  
plt.bar([4,8,12], height=errors)
```

```
best rank : 12  
best RMSE:3.3402154922752136  
<BarContainer object of 3 artists>
```



Getting Recommendation

```
u_tempdf = get_hotel_recc(spark, usrid_s2)  
hotel_df = del_dup.join(u_tempdf, "id").withColumn("address",functions.lower(functions.col("address")))  
# user_location = place.value.lower()  
# hotel_sugg = hotel_df.where(hotel_df.address.contains(user_location))  
recc = hotel_df.dropna().toPandas()
```

```
recc
```

29	5723	street east virten manitoba	['Free High Speed Internet (Wi...	Canada	Average	Virden Motel
30	4399	643 rue roy la tuque quebec	['Free High Speed Internet (WiFi)', 'Golf Co...	Canada	Excellent	Gite aux Trois Pignons
31	4548	1606 east hillcrest drive alberta	['Free High Speed Internet (WiFi)', 'Free pa...	Canada	Excellent	SpringBreak Flower Farm B&B
32	2705	1493 monseigneur- de laval blvd baie-st-paul qu...	['Room service', 'Free parking', 'Restauran...	Canada	Average	Maison Chez Laurent
33	4416	60 principale st edmundston new brunswick	['Free parking', 'Restaurant', 'Pool', 'Ro...	Canada	Average	Motel Le Brayon & Chalets
34	5093	401 route 172 nord sacre- coeur- saguenay quebec	['Restaurant', 'Bar/Lounge', 'Free High Spe...	Canada	Average	Hotel-Motel Coronet
35	4742		['Pool', 'Free parking', 'Free High Speed I...	Canada	Excellent	Gite Ancestral
36	955	120 langley rd vesuvius british columbia	['Free parking', 'Free High Speed Internet (...	Canada	Excellent	Beachfront at Vesuvius
37	4643		['Pool', 'Free High Speed Internet (WiFi)', ...	Canada	Very good	Motel Le Radisson De Val-david
38	315	6271 kispiox valley road british columbia	['Room service', 'Free parking', 'Restauran...	Canada	Excellent	Bear Claw Lodge
39	5796	6 quebec g	['Free parking', 'Free High Speed Internet (...	Canada	Excellent	Le gite chez Majo

```
!pip install bing-image-downloader
```

```
Collecting bing-image-downloader
```

```
  Downloading bing_image_downloader-1.1.2-py3-none-any.whl (5.9 kB)
```

```
Installing collected packages: bing-image-downloader
```

```
Successfully installed bing-image-downloader-1.1.2
```

```
from bing_image_downloader import downloader
```

```
import os
```

```
def get_image(name):
```

```
    name = ' '.join(name.split("_"))
```

```
    os.makedirs(os.path.join('downloads',name),exist_ok=True)
```

```
    if len(os.listdir(os.path.join('downloads',name))) < 1 :
```

```
        downloader.download(name, limit=2, output_dir='downloads',
                             adult_filter_off=True, force_replace=False, timeout=60)
```

```
    return os.path.join('downloads',name,os.listdir(os.path.join('downloads',name))[-1])
```

Attraction:

```
from attractions_recc import *
from IPython.display import display
import ipywidgets as w
import pandas as pd
import re
```

```
WARNING:tensorflow:From C:\Users\nithy\Desktop\Project\travel_recc\envs\lib\site-packages\keras\src\losses.py:2976: The name tf.losses.s
```

```
att_df = pd.read_json('etl/attractions.json',orient='records')
```

▼ User details and Preferences

```
uname = w.Text(description="User Name")
place = w.Text(value = 'Province', description="Destination")
budget = w.IntRangeSlider(min=att_df.price.min(), max=att_df.price.max(),step=10,value=[att_df.price.min(),att_df.price.max()],description="Budget")
v1 = w.VBox([uname,place,budget])
start = w.DatePicker(description='Start Date',disabled=False)
end = w.DatePicker(description='End Date',disabled=False)
v2 = w.VBox([start,end])
out = w.HBox([v1,v2])
display(out)
```

```
⦿ HBox(children=(VBox(children=(Text(value='', description='User Name'), Text(value='Province', description='Des...
```

```
category_df = att_df.groupby('category').size().reset_index().sort_values([0],ascending=False)[:20]
categories = list(category_df.category.values)
cat_rat = dict()
def on_button_clicked(b):
    if b.description in cat_rat:
        return
    print(b.description)
    slider = w.IntSlider(min=0,max=5,step=1,description='Rate')
    display(slider)
    cat_rat[b.description] = slider
    if(len(cat_rat) < 5):
        print("Rate {x} more!\n".format(x=5-len(cat_rat)))
```

```
but_layout = w.Layout(width='100%', height='100px')
but_items = [w.Button(description=c, layout=but_layout) for c in categories]
on_clk = [item.on_click(on_button_clicked) for item in but_items]
c1 = w.VBox([i for i in but_items[:4]])
c2 = w.VBox([i for i in but_items[4:8]])
c3 = w.VBox([i for i in but_items[8:12]])
c4 = w.VBox([i for i in but_items[12:16]])
c5 = w.VBox([i for i in but_items[16:]])
buttons = w.HBox([c1,c2,c3,c4,c5])
print("Select and rate atleast 5 categories and rate them:")
display(buttons)
```

```
Select and rate atleast 5 categories and rate then:
HBox(children=(VBox(children=(Button(description='private_&_custom_tours', layout=Layout(height='100px', width...
```

```
user_name = re.sub(' ','_',uname.value.lower())
province = re.sub(' ','_',place.value.lower())
(low,high) = tuple([float(i) for i in budget.value])
begin_date = start.value
end_date = end.value
cat_rating = dict()
for key, value in cat_rat.items():
    cat_rating[key] = float(value.value)
```

```
user_name,province,(low,high),begin_date,end_date,cat_rating
```

```
('nithya',
'toronto',
(5.0, 555.0),
datetime.date(2024, 4, 9),
datetime.date(2024, 4, 11),
{'private_&_custom_tours': 0.0,
```

```
'cruises_sailing_&_water_tours': 0.0,
'multi-day_&_extended_tours': 0.0,
'shows_concerts_&_sports': 0.0))
```

Model recommendation and filtering

```
filename, user, rbn_att = get_recc(att_df, cat_rating)
```

```
Reading the data
Extracting num_rows from ratings
Preprocessing the dataset
Similar User: 9445
Model restored
```

```
filename, user, rbn_att
```

```
('e50_r40000_lr0.01_hu128_bs16',
9445,
  user_id  attraction_id  rbn_att_id  rating
0         0           74         48      5
1         0          3285        1309      5
2         0          3507        1513      5
3         0           583         281      4
4         0          2249        1064      5
...      ...          ...          ...      ...
33920     13088         2198        1030      5
33921     13089         2716        1232      5
33922     13090          221         132      5
33923     13091         1782         829      5
33924     13092         3230        1300      5
```

```
[33925 rows x 4 columns])
```

```
def filter_df(filename, user, low, high, province, att_df):
    recc_df = pd.read_csv('recommendations/'+filename+'user(u)_unseen.csv'.format(u=user), index_col=0)
    recc_df.columns = ['attraction_id', 'att_name', 'att_cat', 'att_price', 'score']
    recommendation = att_df[['attraction_id', 'name', 'category', 'city', 'latitude', 'longitude', 'price', 'province', 'rating']].set_index('attraction_id')
    # print(recommendation)
    filtered = recommendation[(recommendation.price >= low) & (recommendation.price <= high)]
    url = pd.read_json('outputs/attractions_cat.json', orient='records')
    url['id'] = url.index
    with_url = filtered.set_index('attraction_id').join(url[['id', 'attraction']].set_index('id'), how='inner')
    return with_url
```

```
with_url = filter_df(filename, user, low, high, province, att_df)
# with_url = with_url.head()
with_url
```

		name	category	city	latitude	longitude	price	province
1355		ultimate_day_tour_to_victoria	tours_&_sightseeing	vancouver	48.422722	-123.368980	363.56	british_columbia
6		whistler_small-group_day_trip_from_vancouver	featured_tours_and_tickets	vancouver	49.197834	-123.064995	145.00	british_columbia
428		ultimate_day_tour_to_victoria	day_trips_&_excursions	vancouver	48.422722	-123.368980	363.56	british_columbia
1303		vancouver_city_sightseeing_tour	tours_&_sightseeing	vancouver	49.197834	-123.064995	80.00	british_columbia
2609		ultimate_day_tour_to_victoria	luxury_&_special_occasions	vancouver	48.422722	-123.368980	363.56	british_columbia
...		...	...	...	...	...	...	...
1801		shared_arrival_transfer_calgary_airport_to_al...	transfers_&_ground_transport	calgary	51.130692	-114.006638	72.45	alberta
2438		vancouver_to_victoria_whale_watching_with_retu...	family_friendly	vancouver	49.284752	-123.139481	569.00	british_columbia
2293		hour_east_vancouver_art_mural_&_craft_brewer...	family_friendly	vancouver	38.825863	-76.919594	45.00	british_columbia
376		explore_the_unexplored_exploration_game_on_el...	cultural_&_theme_tours	mont_tremblant	46.173065	-74.598259	102.33	quebec
2183		private_salmon_fishing_charter_from_vancouver_...	water_sports	vancouver	49.202473	-123.144577	749.77	british_columbia

```
1614 rows x 10 columns
```

```

def find_closest(with_url, loc, tod, final):
    syns1 = wordnet.synsets("evening")
    syns2 = wordnet.synsets("night")
    evening = [word.lemmas()[0].name() for word in syns1] + [word.lemmas()[0].name() for word in syns2]
    distance = list()
    for i in with_url[['latitude', 'longitude']].values.tolist():
        distance.append(math.sqrt((loc[0]-i[0])**2 + (loc[1]-i[1])**2))
    with_dist = with_url
    with_dist["distance"] = distance
    sorted_d = with_dist.sort_values(['distance', 'price'], ascending=[True, False])
    if tod == "Evening":
        mask = sorted_d.name.apply(lambda x: any(j in x for j in evening))
    else:
        mask = sorted_d.name.apply(lambda x: all(j not in x for j in evening))
    final = top_recc(sorted_d[mask], final)
    return final

def top_recc(with_url, final):
    i = 0
    while i < len(with_url):
        first_recc = with_url.iloc[[i]]
        if first_recc['name'].values.T[0] not in final['name']:
            final['name'].append(first_recc['name'].values.T[0])
            final['location'].append(first_recc[['latitude', 'longitude']].values.tolist()[0])
            final['price'].append(first_recc['price'].values.T[0])
            final['rating'].append(first_recc['rating'].values.T[0])
            final['image'].append(get_image(first_recc['name'].values.T[0]))
            final['category'].append(first_recc['category'].values.T[0])
        i += 1
    return final

from bing_image_downloader import downloader
import os
def get_image(name):
    name = ' '.join(name.split("_"))
    os.makedirs(os.path.join('downloads', name), exist_ok=True)
    if len(os.listdir(os.path.join('downloads', name))) < 1 :
        downloader.download(name, limit=2, output_dir='downloads',
                             adult_filter_off=True, force_replace=False, timeout=60)

    return os.path.join('downloads', name, os.listdir(os.path.join('downloads', name))[-1])

name = 'ganga_river'
get_image(name)

'downloads\\ganga river\\Image_2.jpg'

import nltk
nltk.download('wordnet')

[nltk_data] Downloading package wordnet to
[nltk_data] C:\Users\nithy\AppData\Roaming\nltk_data...
[nltk_data] Package wordnet is already up-to-date!
True

```

```

%%capture
final = dict()
final['timeofday'] = []
final['image'] = []
final['name'] = []
final['location'] = []
final['price'] = []
final['rating'] = []
final['category'] = []

for i in range(1,(end_date - begin_date).days+2):
    for j in range(2):
        final['timeofday'].append('Morning')
    for j in range(2):
        final['timeofday'].append('Evening')

for i in range(len(final['timeofday'])):
    if i%4 == 0:
        try:
            final = top_recc(with_url, final)
        except:
            pass
    else:
        print(i)
        try:
            final = find_closest(with_url, final['location'][-1],final['timeofday'][i], final)
        except:
            pass

```

▼ Final Output

```

final

{'timeofday': ['Morning',
'Morning',
'Evening',
'Evening',
'Morning',
'Morning',
'Evening',
'Evening',
'Morning',
'Morning',
'Evening',
'Evening'],
'image': ['downloads\\ultimate day tour to victoria\\Image_2.jpg',
'downloads\\whistler small-group day trip from vancouver\\Image_2.jpg',
'downloads\\vancouver city sightseeing tour\\Image_1.jpg',
'downloads\\whistler sasquatch zipline\\Image_2.jpg',
'downloads\\niagara falls canadian side tour and maid of the mist boat ride\\Image_2.jpg',
'downloads\\niagara falls american-side tour with maid of the mist boat ride\\Image_2.png',
'downloads\\niagara falls canada underground railroad heritage tour\\Image_2.jpg',
'downloads\\shared wonder tour\\Image_2.jpg',
'downloads\\niagara falls canada tour from niagara usa\\Image_2.jpg',
'downloads\\niagara falls american side highlights tour\\Image_2.jpg',
'downloads\\niagara falls open-top jet-boat ride\\Image_2.jpg',
'downloads\\niagara falls domed jet-boat ride\\Image_2.jpg',
'downloads\\best of both niagara falls\\xa0 american/canadian tour\\Image_2.jpg',
'downloads\\bird kingdom all day admission ticket\\Image_2.jpg',
'downloads\\niagara falls helicopter tour\\Image_2.jpg',
'downloads\\ultimate niagara falls tour plus helicopter ride and skylon tower lunch\\Image_2.jpg',
'downloads\\best of niagara falls tour from toronto\\Image_2.jpg',
'downloads\\ultimate toronto tour\\Image_2.jpg',
'downloads\\best of niagara falls tour from niagara falls, ontario\\Image_2.jpg',
'downloads\\toronto to niagara falls & niagara-on-the-lake day tour with free wine tasting\\Image_2.jpg',
'downloads\\niagara falls and boat cruise\\Image_2.jpg',

```

```

def final_output(days, final):
    time = ['MORNING', 'EVENING']
    fields = ['NAME', 'CATEGORY', 'LOCATION', 'PRICE', 'RATING']
    recommendations = ['Recommendation 1:', 'Recommendation 2:']

    box_layout = Layout(justify_content='space-between',
                        display='flex',
                        flex_flow='row',
                        align_items='stretch',
                        )
    column_layout = Layout(justify_content='space-between',
                          width='75%',
                          display='flex',
                          flex_flow='column',
                          )

    days = (end_date - begin_date).days + 1
    display(final_output(days, final))

    Tab(children=(VBox(children=(HBox(children=(VBox(children=(HTML(value="<b><font color='orange'>MORNING</b>"), ...
    # print(images)

```

Restaurant:

```
!pip install langchain openai langchain-openai
```

```

Requirement already satisfied: langchain in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (0.1.14)
Requirement already satisfied: openai in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (1.16.2)
Requirement already satisfied: langchain-openai in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (0.1.1)
Requirement already satisfied: PyYAML>=5.3 in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from langchain) (6.0.1)
Requirement already satisfied: SQLAlchemy<3,>=1.4 in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from langchain)
Requirement already satisfied: aiohttp<4.0.0,>=3.8.3 in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from langchain)
Requirement already satisfied: async-timeout<5.0.0,>=4.0.0 in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from langchain)
Requirement already satisfied: dataclasses-json<0.7,>=0.5.7 in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from langchain)
Requirement already satisfied: jsonpatch<2.0,>=1.33 in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from langchain)
Requirement already satisfied: langchain-community<0.1,>=0.0.30 in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from langchain)
Requirement already satisfied: langchain-core<0.2.0,>=0.1.37 in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from langchain)
Requirement already satisfied: langchain-text-splitters<0.1,>=0.0.1 in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from langchain)
Requirement already satisfied: langsmith<0.2.0,>=0.1.17 in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from langchain)
Requirement already satisfied: numpy<2,>=1 in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from langchain) (1.26.4)
Requirement already satisfied: pydantic<3,>=1 in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from langchain) (2.6)
Requirement already satisfied: requests<3,>=2 in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from langchain) (2.3)
Requirement already satisfied: tenacity<9.0.0,>=8.1.0 in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from langchain)
Requirement already satisfied: anyio<5,>=3.5.0 in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from openai) (4.2.0)
Requirement already satisfied: distro<2,>=1.7.0 in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from openai) (1.9)
Requirement already satisfied: httpx<1,>=0.23.0 in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from openai) (0.26)
Requirement already satisfied: sniffio in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from openai) (1.3.0)
Requirement already satisfied: tqdm>4 in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from openai) (4.66.2)
Requirement already satisfied: typing-extensions<5,>=4.7 in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from openai)
Requirement already satisfied: tiktoken<1,>=0.5.2 in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from langchain-community)
Requirement already satisfied: aiohttp<4.0.0,>=3.8.3 in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from langchain-community)
Requirement already satisfied: frozenlist>=1.1.1 in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from aiohttp<4.0.0,>=3.8.3)
Requirement already satisfied: multidict<7.0,>=4.5 in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from aiohttp<4.0.0,>=3.8.3)
Requirement already satisfied: yarl<2.0,>=1.0 in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from aiohttp<4.0.0,>=3.8.3)
Requirement already satisfied: idna>=2.8 in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from anyio<5,>=3.5.0)
Requirement already satisfied: exceptiongroup>=1.0.2 in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from anyio<5,>=3.5.0)
Requirement already satisfied: marshmallow<4.0.0,>=3.18.0 in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from dataclasses-json)
Requirement already satisfied: typing-inspect<1,>=0.4.0 in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from dataclasses-json)
Requirement already satisfied: certifi in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from httpx<1,>=0.23.0)
Requirement already satisfied: httpcore==1.* in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from httpx<1,>=0.23.0)
Requirement already satisfied: h11<0.15,>=0.13 in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from httpcore==1.*)
Requirement already satisfied: jsonpointer>=1.9 in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from jsonpatch<2.0,>=1.33)
Requirement already satisfied: packaging<24.0,>=23.2 in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from langchain-community)
Requirement already satisfied: orjson<4.0.0,>=3.9.14 in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from langsmith)
Requirement already satisfied: annotated-types>=0.4.0 in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from pydantic)
Requirement already satisfied: pydantic-core==2.16.3 in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from pydantic)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from requests)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from requests)
Requirement already satisfied: greenlet!=0.4.17 in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from SQLAlchemy<3,>=1.4)
Requirement already satisfied: regex>=2022.1.18 in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from tiktoken<1,>=0.5.2)
Requirement already satisfied: colorama in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from tqdm>4) (0.4)
Requirement already satisfied: mpy-extensions>=0.3.0 in c:\users\nithy\desktop\project\travel_recc\envs\lib\site-packages (from typing-extensions)

```

```

from IPython.display import display
import ipywidgets as w

country = w.Text(description="Country name")
city = w.Text(description="City name")
start = w.DatePicker(description='Start Date',disabled=False)
end = w.DatePicker(description='End Date',disabled=False)
v1 = w.VBox([country,city])
v2 = w.VBox([start,end])
out = w.HBox([v1,v2])
display(out)

HBox(children=(VBox(children=(Text(value='', description='Country name'),
Text(value='', description='City name')

begin_date = start.value
end_date = end.value
(end_date - begin_date).days

user_country = country.value
user_city = city.value

user_country,user_city

('canada', 'toronto')

import ipywidgets as widgets
from IPython.display import display

# Common restaurant cuisines
cuisines = ['Italian', 'Mexican', 'Chinese', 'Indian', 'Japanese', 'American', 'Mediterranean']

# Create a multi-select dropdown widget
multi_select = widgets.SelectMultiple(
    options=cuisines,
    description='Cuisines',
    disabled=False
)

# Variable to store selected cuisines
selected_cuisines = []

# Function to handle multi-select change event
def on_multi_select_change(change):
    global selected_cuisines
    selected_cuisines = list(change['new'])

# Assigning change event handler to the multi-select dropdown
multi_select.observe(on_multi_select_change, names='value')

# Display the multi-select dropdown
display(multi_select)

SelectMultiple(description='Cuisines', options=('Italian', 'Mexican', 'Chinese',
'Indian', 'Japanese', 'American', 'Mediterranean'))

selected_cuisines

['Italian']

user_cuisine = selected_cuisines[0]
user_cuisine

'Italian'

import os
from langchain_openai import ChatOpenAI
from langchain.prompts import (
    PromptTemplate,
    ChatPromptTemplate,
    SystemMessagePromptTemplate,
    HumanMessagePromptTemplate,
)
from langchain.output_parsers import StructuredOutputParser, ResponseSchema

```

```

def get_prompt_template(system_prompt, system_prompt_variables, human_prompt, human_prompt_variables):
    system_prompt = PromptTemplate(input_variables = system_prompt_variables, template=system_prompt)
    system_message_prompt = SystemMessagePromptTemplate(prompt=system_prompt)

    human_prompt = PromptTemplate(input_variables=human_prompt_variables, template=human_prompt)
    human_message_prompt = HumanMessagePromptTemplate(prompt=human_prompt)

    chat_prompt = ChatPromptTemplate.from_messages([system_message_prompt, human_message_prompt])
    return chat_prompt

def get_output_parser(d):
    response_schemas = []
    for name, desc in d.items():
        response_schemas.append(ResponseSchema(name=name, description=desc))

    output_parser = StructuredOutputParser.from_response_schemas(response_schemas)
    format_instructions = output_parser.get_format_instructions()

    return output_parser, format_instructions

def get_chat_model(openai_api_key, model_name, temperature, max_retries=5):
    chat_model = ChatOpenAI(model_name=model_name, openai_api_key=openai_api_key, temperature=temperature, max_retries = 5)
    return chat_model

openai_api_key = 'sk-ObyGz14Dcpps6soJCFnZT3B1bkFJK584IYWlwcK82PjuQyz0'
model_name = 'gpt-3.5-turbo-0125'
temperature = 0.9
chat_model = get_chat_model(openai_api_key, model_name, temperature, max_retries=5)
chat_model

ChatOpenAI(client=<openai.resources.chat.completions.Completions object at 0x00000218B3704C10>, async_client=
<openai.resources.chat.completions.AsyncCompletions object at 0x00000218B3705C00>, model_name='gpt-3.5-turbo-0125', temperature=0.9,
openai_api_key=SecretStr('*****'), openai_proxy='', max_retries=5)

response_schema_dict = {}
response_schema_dict['name'] = "name of the restaurant"
response_schema_dict['location'] = "location of the restaurant"
response_schema_dict['speciality'] = "speciality of the restaurant"
response_schema_dict['open_time'] = "day and time when the restaurant is open for eg. monday to friday, 10pm to 7pm"

output_parser, format_instructions = get_output_parser(response_schema_dict)
print(format_instructions)

The output should be a markdown code snippet formatted in the following schema, including the leading and trailing ```json" and ```

```json
{
 "name": string // name of the restaurant
 "location": string // location of the restaurant
 "speciality": string // speciality of the restaurant
 "open_time": string // day and time when the restaurant is open for eg. monday to friday, 10pm to 7pm
}
...

system_prompt = "You are a helpful assistant"
system_prompt_variables = []

human_prompt = "Your task is to suggest restaurant given country, city and cuisine"
human_prompt += '\n' + '###' * 30
human_prompt += "\n<<Country>>:\n{country}\n"
human_prompt += '\n' + '###' * 30
human_prompt += "\n<<City>>:\n{city}\n"
human_prompt += '\n' + '###' * 30
human_prompt += "\n<<Cuisine>>:\n{cuisine}\n"
human_prompt += '\n' + '###' * 30
human_prompt += '\n(format_instructions)\n'
human_prompt_variables = ['country', 'cuisine', 'city', 'format_instructions']

chat_prompt = get_prompt_template(system_prompt, system_prompt_variables, human_prompt, human_prompt_variables)
chat_prompt

ChatPromptTemplate(input_variables=['city', 'country', 'cuisine', 'format_instructions'], messages=
[SystemMessagePromptTemplate(prompt=PromptTemplate(input_variables=[], template='You are a helpful assistant')),
HumanMessagePromptTemplate(prompt=PromptTemplate(input_variables=['city', 'country', 'cuisine', 'format_instructions'], template='Your
task is to suggest restaurant given country, city and
cuisine\n#####\n<<Country>>:\n{country}\n\n#####

```

```
{}
```

```
d['country'] = user_country
d['city'] = user_city
d['cuisine'] = user_cuisine
d['format_instructions'] = format_instructions
results = chat_prompt.invoke(d)
for i in results.messages:
 print(i.content)
```

```
You are a helpful assistant
Your task is to suggest restaurant given country, city and cuisine
#####
<<Country>>:
canada

#####
<<City>>:
toronto

#####
<<Cuisine>>:
Italian
```

```
#####
The output should be a markdown code snippet formatted in the following schema, including the leading and trailing ```json" and "```":
```json
{
    "name": string // name of the restaurant
    "location": string // location of the restaurant
    "speciality": string // speciality of the restaurant
    "open_time": string // day and time when the restaurant is open for eg. monday to friday, 10pm to 7pm
}
...

```

```
chain = chat_prompt | chat_model | output_parser
results = chain.invoke(d)
results

{'name': 'Terroni',
 'location': '720 Queen St W, Toronto, ON M6J 1E8',
 'speciality': 'Authentic Italian cuisine',
 'open_time': 'Monday to Sunday, 11:30am to 11:00pm'}
```

```
!pip install bing-image-downloader

Requirement already satisfied: bing-image-downloader in c:\users\nithy\desktop\project\tavel_recc\envs\lib\site-packages (1.1.2)
```

```
from bing_image_downloader import downloader
import os
def get_image(name):
    os.makedirs(os.path.join('downloads',name),exist_ok=True)
    if len(os.listdir(os.path.join('downloads',name))) < 1 :
        downloader.download(name, limit=2, output_dir='downloads',
            adult_filter_off=True, force_replace=False, timeout=60)

    return os.path.join('downloads',name,os.listdir(os.path.join('downloads',name))[-1])

name = f"{results['name']}, {results['location']}"
get_image(name)
```

```
[%] Downloading Images to C:\Users\nithy\Desktop\Project\tavel_recc\downloads\Terroni, 720 Queen St W, Toronto, ON M6J 1E8

[!]Indexing page: 1

[%] Indexed 2 Images on Page 1.

=====

[%] Downloading Image #1 from https://i.pinimg.com/originals/b8/82/3c/b8823c5b34abaec7f981f443a1a3b2e.jpg
[%] File Downloaded !

[%] Downloading Image #2 from https://assets.simpleviewinc.com/simpleview/image/fetch/c_limit,h_1200,g_75,w_1200/https://assets.simplevi
[%] File Downloaded !

[%] Done. Downloaded 2 images.
'downloads\\Terroni, 720 Queen St W, Toronto, ON M6J 1E8\\Image 2.jpg'
```

```

from tqdm.auto import tqdm

day = []
name = []
address = []
img_path = []
speciality = []
open_time = []

for i in tqdm(range((end_date - begin_date).days)):

    chain = chat_prompt | chat_model | output_parser
    results = chain.invoke(d)

    try:
        img = get_image(f"{results['name']}, {results['location']}")
        day.append(f"Day {i+1}")
        name.append(results['name'])
        img_path.append(img)
        address.append(results['location'])
        speciality.append(results['speciality'])
        open_time.append(results['open_time'])
    except:
        pass

    0%|          | 0/2 [00:00<?, ?it/s]
[%] Downloading Images to C:\Users\nithy\Desktop\Project\travel_recc\downloads\Terroni, 720 Queen St W, Toronto, ON M6J 1E8, Canada

[!!]Indexing page: 1

[%] Indexed 2 Images on Page 1.

=====

[%] Downloading Image #1 from https://i.pinimg.com/originals/b8/82/3c/b8823c5b34abaec7f981f4434a1a3b2e.jpg
[%] File Downloaded !

[%] Downloading Image #2 from https://media-cdn.tripadvisor.com/media/photo-s/16/eb/56/3d/photo2.jpg.jpg
[%] File Downloaded !

[%] Done. Downloaded 2 images.

import pandas as pd
data = {
    'Day': day,
    'Name': name,
    'Image_Path': img_path,
    'Address': address,
    'Speciality': speciality,
    'Open_Time': open_time
}

# Create DataFrame from the dictionary
df = pd.DataFrame(data)

# Display the DataFrame
df

```

	Day	Name	Image_Path	Address	Speciality	Open_Time
0	Day 1	Terroni	downloads\Terroni, 720 Queen St W, Toronto, ON...	720 Queen St W, Toronto, ON M6J 1E8	Authentic Italian dishes	Monday to Sunday, 12pm to 10pm
1	Day 1	Terroni	downloads\Terroni, 720 Queen St W, Toronto, ON...	720 Queen St W, Toronto, ON M6J 1E8	Traditional Southern Italian	Monday to Friday, 11:30am to 10pm

```

import re
import ipywidgets as widgets
from ipywidgets import Layout, VBox, HBox

def final_output(df):
    fields = ['NAME', 'ADDRESS', 'SPECIALITY', 'OPEN TIME']

    box_layout = Layout(justify_content='space-between',
                        display='flex',
                        flex_flow='row',
                        align_items='stretch',
                        )
    column_layout = Layout(justify_content='space-between',
                          width='75%',
                          display='flex',
                          flex_flow='column',
                          )

    tab = []
    for i in range(len(df)):
        # Extracting data from the DataFrame
        name = df.loc[i, 'Name']
        img_path = df.loc[i, 'Image_Path']
        address = df.loc[i, 'Address']
        speciality = df.loc[i, 'Speciality']
        open_time = df.loc[i, 'Open_Time']

        # Assuming you have the images already loaded, otherwise you might need to modify this part
        image_widget = widgets.Image(value=open(img_path, "rb").read(), format='jpg', width=300, height=400)

        tab.append(VBox(children=
            [HBox(children=
                [VBox(children=[
                    image_widget,
                    widgets.HTML(description=fields[0], value=f"<b><font color='black'>{name}</b>", disabled=True),
                    widgets.HTML(description=fields[1], value=f"<b><font color='black'>{address}</b>", disabled=True),
                    widgets.HTML(description=fields[2], value=f"<b><font color='black'>{speciality}</b>", disabled=True),
                    widgets.HTML(description=fields[3], value=f"<b><font color='black'>{open_time}</b>", disabled=True)
                ])
            ])
        ])

    df.shape[0]

    2
    tab_recc = widgets.Tab(children=tab)
    final_output(df)

    Tab(children=(VBox(children=(HBox(children=(VBox(children=(Image(value=b'\xff\xd8\xff\xe0\x00\x10JFIF\x00\x01\...
```

B.SCREENSHOTS

User inputs:

- Users give travel details as input.

User Name	Nithya	Start Date	09-04-2024	
Destination	toronto	End Date	11-04-2024	
Budget	 5 – 555			

- Users can choose amenities they are looking for in hotels.

Select atleast 5 amenities:

Nonsmoking hotel	Refrigerator in room	Wheelchair Access
Nonsmoking rooms	Family Rooms	Microwave
Free High Speed Internet WiFi	Public Wifi	Breakfast included
Air conditioning	Pets Allowed Dog Pet Friendly	Laundry Service
Free parking	Suites	Kitchenette

Nonsmoking hotel
Select 4 more!

Nonsmoking rooms
Select 3 more!

- Users can give their attraction preferences.

Select and rate atleast 5 categories and rate them:

private_&_custom_tours	food_wine_&_nightlife	day_trips_&_excursions	luxury_&_special_occasions	air_helicopter_&_balloon_tours
tours_&_sightseeing	family_friendly	water_sports	sightseeing_tickets_&_passes	holiday_&_seasonal_tours
recommended_experiences	cultural_&_theme_tours	cruises_sailing_&_water_tours	multi-day_&_extended_tours	shows_concerts_&_sports
outdoor_activities	walking_&_biking_tours	transfers_&_ground_transport	shore_excursions	classes_&_workshops

private_&_custom_tours

Rate 0

Rate 4 more!

- Type of cuisine user prefers is given as input.

Cuisine type

☐ Asian	☒ Irish	☐ African
☐ Japanese	☒ French	☐ Indian
☒ Pubs	☐ Fusion	☐ Pakistani
☐ Sports Bar	☒ Mexican	☐ Italian
☐ Mediterranean	☒ Canadian	☐ Chinese

Preferences

☒ Vegetarian
☐ Vegan
☐ Halal
☐ Gluten Free

Price Range

☐ Budget
☒ Luxury

Outputs:

Hotel recommendations:

Hotel 1

Hotel 2

Hotel 3

Hotel 4

Hotel 5



NAME

Motel Populaire

PRICE

116.73565891472869

RATING

4.0

EXPERIEN...

Very good

LOCATION

46.3263818, -72.5622352

ADDRESS

400 rue de la sangamo trois-rivieres quebec

Attraction recommendations:

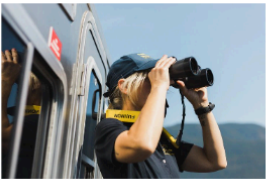
Day 1

Day 2

Day 3

MORNING

Recommendation 1:



NAME

Ultimate day tour to victoria

CATEGORY

Tours & sightseeing

LOCATION

(48.4227218628,-123.3689804077)

PRICE

363.56

RATING

4.5

Recommendation 2:



EVENING

Recommendation 1:



NAME

Vancouver city sightseeing tour

CATEGORY

Tours & sightseeing

LOCATION

(49.1978340149,-123.064994812)

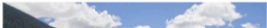
PRICE

80.0

RATING

4.5

Recommendation 2:



Restaurant recommendations:

[52]:

Day 1

Day 2



NAME

Terroni

ADDRESS

720 Queen St W, Toronto, ON M6J 1E8

SPECIALITY

Authentic Italian dishes

OPEN TIME

Monday to Sunday, 12pm to 10pm

C. RESEARCH PAPER:

PERSONALIZED TRAVEL ITINERARY GENERATOR SYSTEM

Nithyasri Prabha. S
Department of Computer Science and
Engineering,
Sathyabama Institute of Science and
Technology, Chennai, India.
nithyasripandian1010@gmail.com

G. Kalaiarasi
Department of Computer Science and
Engineering,
Sathyabama Institute of Science and
Technology, Chennai, India.
kalaiarasi.cse@sathyabama.ac.in

Nivash. V
Department of Computer Science and
Engineering,
Sathyabama Institute of Science and
Technology, Chennai, India.
nivinivash00@gmail.com

M. Selvi
Department of Computer Science and
Engineering,
Sathyabama Institute of Science and
Technology, Chennai, India.
selvi.cse@sathyabama.ac.in

Yogitha. R
Department of Computer Science and
Engineering,
Sathyabama Institute of Science and
Technology, Chennai, India.
yogitha.cse@sathyabama.ac.in

Abstract- The Personalized Travel Itinerary Generator System is designed to enhance vacation planning efficiency, catering to individual preferences. Utilizing user-provided details such as destination, budget, and interests, the system generates bespoke travel plans. Users specify hotel amenities and food preferences, with the system recommending five hotels, two daily attractions, and dining options. Employing a data science pipeline encompassing web scraping, exploratory data analysis, user profiling, and three recommendation algorithms—Restricted Boltzmann Machine, Alternating Least Squares, and a hybrid model—the project exemplifies the integration of cutting-edge technologies. This innovative system not only streamlines itinerary creation but also provides valuable insights into the synergistic application of data-driven methodologies.

Keywords - Travel Itinerary Generator System, Exploratory data analysis, User profiling

I. INTRODUCTION

The Travel Itinerary Generator System project represents an innovative solution designed to simplify and enhance the way individuals plan their journeys. Traveling is an integral part of our lives, whether for leisure, business, or any other purpose. However, crafting a comprehensive and enjoyable itinerary can often be a time-consuming and challenging task. The Travel Itinerary Generator System aims to address these issues and provide travelers with a seamless and personalized planning experience.

In an era where technology has transformed the way we live, work, and travel, this project leverages cutting-edge software and algorithms to streamline the itinerary creation process. Our primary objective is to develop a user-friendly application that

automates the generation of travel itineraries, offering several key benefits to users.

Personalization: One of the core features of the system is its ability to create personalized itineraries. By employing advanced algorithms, the system analyses user preferences, interests, and constraints to generate tailored itineraries that match individual traveler profiles. Whether you're a history buff, a foodie, an adventure seeker, or all of the above, the Travel Itinerary Generator System will cater to your unique needs.

Efficiency: We understand that time is of the essence in today's fast-paced world. To ensure efficient planning, the system utilizes advanced data scraping, processing, and optimization techniques. It gathers destination-related information, including attractions, transportation options, accommodation, and more, to construct optimal travel plans. This means less time spent on research and more time for you to enjoy your trip.

Flexibility: Travel plans are often subject to change. The system allows users to customize their generated itineraries by modifying activities, adjusting timings, adding personal notes, and including specific preferences. Whether you want to squeeze in an extra museum visit or relax on the beach a bit longer, the system accommodates your evolving plans.

Integration: The Travel Itinerary Generator System is designed to seamlessly integrate with external APIs and databases. This integration provides real-time information about attractions, weather conditions, transportation schedules, and accommodation availability, ensuring that your plans are up-to-date and reliable.

By achieving these objectives, the Travel Itinerary Generator System project aims to revolutionize the way travelers plan their trips. It offers convenience, personalization, efficiency, and

flexibility, resulting in a more enjoyable and stress-free travel experience. Whether you're a seasoned globetrotter or a first-time travellers, this system will make planning your journey a breeze, allowing you to focus on creating memorable moments during your adventures.

II. RELATED WORKS

a) An Intelligent Destination Recommendation System (2018) The process of selecting a tourist destination can be daunting, both in the planning stage and during the actual trip. With the advent of recommendation systems, travelers now can swiftly compare and choose destinations. This involves constructing decision models, predicting user preferences, and interpreting the outcomes.

This research project sets out to create a Destination Recommendation System (DRS) with a strong focus on machine-learning techniques. To develop an effective DRS, an extensive review of existing literature on recommendation systems in the tourism domain was conducted. Subsequently, the thesis introduces a model-based DRS, featuring a two-step filtering method to eliminate irrelevant and redundant information and employing a Decision Tree (DT) classifier for transparency, interpretability, and efficiency in aiding tourists' decision-making. The system's scalability is validated using real-world data from a case-study city. Destination choice models were formulated and tested, demonstrating that the proposed model-based DRS can offer personalized and satisfactory recommendations to users.

The thesis further suggests an ensemble-based DRS, incorporating weight hybrid and cascade hybrid approaches. Three classification algorithms, including DT, Support Vector Machines (SVMs), and Multi-Layer Perceptron (MLPs), were explored. Results revealed that the bagging ensemble of MLP classifiers outperformed baseline learners and other combiners.

Lastly, the thesis introduces an Adaptive, Responsive, Interactive Model-based User Interface (ARIM-UI) for the DRS, enhancing the user experience by providing adaptive, informative, and responsive information.

b) An automated itinerary planning system for holiday travel(2004) This paper introduces the Electronic Travel Planner (ETP), a prototype travel recommender system geared towards crafting itineraries for tourists. The system relies on traveler preferences and extensive databases filled with tourism and travel data. Its primary functions involve destination selection, tour and attraction recommendations, and the creation of comprehensive travel schedules. These tasks involve intricate optimization challenges, which the prototype tackles using a

heuristic problem-solving approach. Computational tests validate the efficacy of these methods, suggesting the potential for automated travel planning in full-scale applications.

The ETP, a web-based prototype, emerged from research conducted between 1999 and 2001 in collaboration with Viator Inc., supported by an Australian Government grant. The system's core code resides in an Internet server, with plans for user interface and Internet connection integration. Commercial implementation discussions are ongoing. The ETP project's core goal was to revolutionize holiday planning, capturing travelers' preferences and exploring possibilities beyond current capabilities. By offering travel agents and individuals a powerful planning tool, the project aims to enhance satisfaction in the travel planning process and the overall travel experience. These objectives align with the concept of a travel recommender system, embracing various choice models, interactive preference determination, and adaptive recommender systems based on user dialogue.

c) Smart travel recommendation system(2022) The internet's remarkable growth and its vast user population have ushered in an era of abundant online knowledge. This treasure trove of information proves invaluable for tailoring recommendations to users' preferences and needs. Recommender systems, pivotal in this landscape, gather user data encompassing preferences, interests, and geographical information. Their development gained prominence with the advent of collaborative filtering techniques. Meeting user preferences is a complex task, yet recommender systems harness data from diverse sources to make predictions and provide personalized item recommendations. This project aims to serve as a one-stop solution for vacation planning, offering travelers tailor-made suggestions aligned with their preferences and desires.

In the quest for a break from the mundane, what's the answer? A vacation! Yet, today, planning a getaway demands a substantial chunk of time. The extensive process involves sifting through countless hotels, attractions, and dining options, all while keeping within a budget. It's a daunting task, and finding a comprehensive source is nearly impossible. Our project aims to streamline this process, allowing travelers to spend more time enjoying their vacation. By collecting travel details and preferences, we create personalized travel plans. We recommend accommodations, activities throughout the day, and dining options to make their trip a memorable one.

d) Smart tourist guide system(2022) The Smart Tourist Guide System represents a groundbreaking open-source platform tailored for individuals embarking on journeys across India. This application serves as a resource to help travelers plan their Indian expeditions, taking into account their preferred number of days,

budget, weather considerations, and more. Moreover, it provides valuable insights into the historical background of destinations, showcases traveller reviews, and displays user ratings. The primary objective behind this project is to create a comprehensive tourist guide for Delhi, aiming to address the challenges faced by tourists due to the absence of a reliable guide. Typically, visitors to Delhi resort to hiring professional tourist guides, incurring significant expenses. Often, these guides are not full-time professionals, and their knowledge can be limited, leading to gaps in providing essential information, such as weather conditions, historical significance, and distances from sea level.

The Smart Tourist Guide offers a solution by harnessing modern technology through an Android application. It utilizes Google Maps API, Global Positioning System (GPS), and the internet. By inputting latitude and longitude, it locates destinations on a map and calculates distances between the user's current location and their desired spots. This innovation is designed to revamp the tourism industry, enhancing the travel experience while boosting the tourism sector.

e) An Optimal Itinerary Generator(2019) It presents an innovative approach to solving the challenging problem of generating optimal travel itineraries for tourists. The authors introduce a novel algorithm that efficiently computes the best possible itinerary while taking into account a multitude of factors and constraints.

The system is grounded in graph theory and utilizes a vast database of attractions, their characteristics, and connections between them. The algorithm considers multiple user-defined constraints, such as time limitations, budget constraints, and personal preferences, to generate an itinerary that maximizes the traveler's satisfaction. It employs a heuristic search that evaluates numerous possible paths and selects the one that best aligns with the user's constraints.

One notable feature of this approach is its adaptability. It continuously refines the itinerary as the user provides feedback or changes their preferences, ensuring that the travel plan remains relevant and optimized.

The paper highlights the scalability of the algorithm, emphasizing its ability to handle large datasets, making it suitable for a wide range of destinations. It also discusses the application of machine learning to enhance recommendation accuracy by learning from user interactions and adapting to trends.

The "An Optimal Itinerary Generator" introduces a solution that has the potential to revolutionize the travel planning industry by

providing travelers with personalized, real-time itineraries that cater to their unique needs and preferences. The system's adaptability, scalability, and efficient algorithm make it a promising development in the field of travel itinerary generation. However, real-world implementation and user acceptance would require further exploration, and addressing potential limitations and challenges, such as data accuracy and privacy concerns, is essential for its success.

III. EXISTING SYSTEM

The existing systems exhibit several drawbacks that hinder their efficiency. One significant issue is a lack of personalization, as these systems often rely on generic algorithms that may not consider individual preferences, resulting in suboptimal recommendations. Another drawback is the limited flexibility in accommodating real-time changes or unexpected events, rendering the plans rigid and potentially inconvenient. Additionally, data accuracy and outdated information can be problematic, leading to inaccurate suggestions and potentially causing disruptions during travel. Furthermore, some systems may lack integration with comprehensive travel services, restricting users from accessing a holistic range of options. Overall, these limitations underscore the need for improvements in personalization, flexibility, accuracy, and integration within existing itinerary planner systems.

IV. PROPOSED SYSTEM

The proposed personalized travel itinerary generator system will integrate web scraping, exploratory data analysis, and three recommendation algorithms (Restricted Boltzmann Machine, Alternating Least Squares, and a hybrid model). Users input destination, budget, preferences, hotel amenities, and food choices. The system will generate personalized travel plans, suggesting five hotels, two daily attractions, and meal options. A user profiling module leverages machine learning for behavior analysis. The architecture ensures seamless data flow from collection to recommendation, creating a scalable and user-centric platform for efficient vacation planning.

IV. SYSTEM ARCHITECTURE

The personalized travel itinerary generator system is designed to revolutionize vacation planning by optimizing the process and providing tailored recommendations to users. Leveraging user-provided travel details such as destination, budget, and preferences, this system generates personalized travel plans. The user interface allows travelers to choose hotel amenities and food preferences, and in response, it recommends five hotels, two daily attractions, and restaurants for all meals during the trip. The underlying data science pipeline employs a comprehensive

approach, incorporating web scraping, exploratory data analysis, and user profiling. The recommendation engine is fueled by three sophisticated algorithms: Restricted Boltzmann Machine, Alternating Least Squares, and a hybrid model. This architecture not only enhances the user experience by saving time during vacation planning but also offers valuable insights into the application of cutting-edge technologies, including data scraping, exploratory data analysis, and diverse recommendation algorithms within the travel domain.

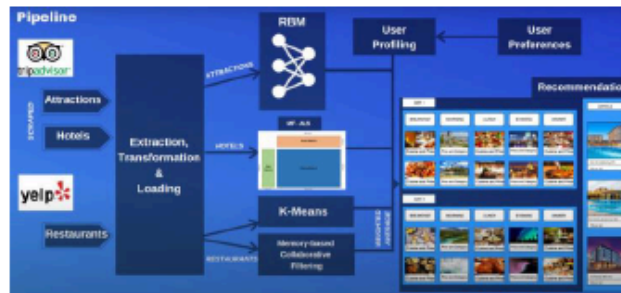


Fig.1. Data science pipeline

V. METHODOLOGY

The methodology employed in the development and functioning of the personalized travel itinerary generator system is structured to efficiently streamline vacation planning, ultimately enhancing the overall user experience. The system's core functionality revolves around utilizing user-provided travel details, including destination preferences, budget constraints, and specific interests. Upon receiving these inputs, it generates personalized travel plans, offering a comprehensive itinerary for users.

Fig.2. User input

The data science pipeline employed in the system encompasses several crucial steps, starting with web scraping. This initial phase involves extracting relevant information from various online sources to build a robust dataset. The extracted data serves

as the foundation for subsequent processes and ensures that the recommendations provided are based on the most up-to-date and accurate information available.

Following web scraping, the system engages in exploratory data analysis (EDA). This step involves a thorough examination of the dataset to identify patterns, correlations, and meaningful insights. EDA not only aids in understanding the characteristics of the data but also informs the subsequent stages of the recommendation process. It plays a pivotal role in shaping user profiling, allowing the system to gain a nuanced understanding of individual preferences and travel behaviour.

User profiling is a crucial aspect of the system's methodology. By analysing the user's historical data and preferences, the system creates a personalized profile that becomes the basis for generating tailored recommendations. This user-centric approach ensures that the travel plans suggested by the system align with the unique preferences and requirements of each user.

The recommendation algorithms utilized in the system are diverse and sophisticated. The three main algorithms employed are the Restricted Boltzmann Machine, Alternating Least Squares, and a hybrid model that combines elements of both. These algorithms work synergistically to analyse user profiles, historical data, and other relevant factors to provide accurate and personalized recommendations. The hybrid model, in particular, leverages the strengths of multiple algorithms, enhancing the system's ability to generate high-quality suggestions.

The overarching methodology of the project not only focuses on the technical aspects of recommendation algorithms but also sheds light on the practical implementation of technologies such as data scraping and exploratory data analysis. By integrating these technologies into the system, it showcases a comprehensive approach to intelligent travel planning. This methodology ensures that the system remains dynamic, efficient, and capable of continuously adapting to evolving user preferences and travel trends. In essence, the system's methodology is designed to empower users with a seamless and personalized vacation planning experience while providing valuable insights into the intersection of technology and travel.

VI. RESULT AND DISCUSSION

The implementation of the Personalized Travel Itinerary Generator System has demonstrated compelling outcomes, showcasing its effectiveness in optimizing vacation planning. Users have reported substantial time savings in itinerary creation, attributing this efficiency to the system's ability to generate tailored recommendations based on individual travel details. The utilization of recommendation algorithms, including the Restricted Boltzmann Machine, Alternating Least Squares, and a

hybrid model, enhances the accuracy of suggested hotels, attractions, and dining options. The seamless integration of data scraping and exploratory data analysis ensures the system's adaptability to evolving user preferences. In summary, the Personalized Travel Itinerary Generator System exhibits considerable potential in elevating the user experience of travel planning.

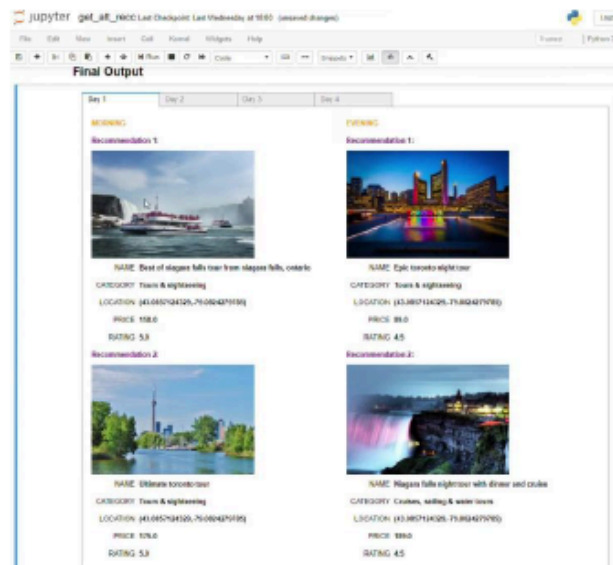


Fig.3. Final output

VI. CONCLUSION

In conclusion, the implementation of the Personalized Travel Itinerary Generator System represents a significant leap forward in the realm of travel planning. The system's success in delivering tailored recommendations, streamlining itinerary creation, and optimizing user experiences underscores its practical value. By leveraging advanced recommendation algorithms and integrating technologies such as data scraping and exploratory data analysis, the system ensures adaptability and accuracy in catering to individual preferences. The positive feedback from users regarding time savings and personalized suggestions attests to the efficacy of this innovative approach. The Personalized Travel Itinerary Generator System stands as a promising solution, offering a seamless and efficient means for travelers to plan and enjoy bespoke travel experiences.

REFERENCES

- [1] An Intelligent Destination Recommendation System (2018), Pree Thiengburanathum, Doctor of Philosophy, Department of Computing and Informatics, Faculty of Science and Technology Bournemouth University, U.K. March 2018.
<https://core.ac.uk/download/pdf/154422294.pdf>
- [2] An automated itinerary planning system for holiday travel Published in Information Technology and Tourism, v6(3), 2004. Submitted June 2003, revised December 2003 Simon Dunstall, Mark E.T. Horn, Philip Kilby, Mohan Krishnamoorthy, Bowie Owens, David Sier and Sylvie Thiebaux. CSIRO Mathematical and Information Sciences.
https://www.researchgate.net/publication/220542946_An_Automated_Itinerary_Planning_System_for_Holiday_Travel
- [3] Smart Travel Plan Recommendation System, Issue 4, April 2022, Vaibhav Mandale, Niraj Reddy, Amlan Sahu, Praveenkumar Patel, Student, Information Technology, Bharati Vidyapeeth College of Engineering, Navi Mumbai, India.ISSN (O) 2278-1021, ISSN (P) 2319-5940
<https://ijarcce.com/wp-content/uploads/2022/04/IJARCCE.2022.11427.pdf>
- [4] Smart Tourist Guide System, 2022, Rahul Chavan, Manvee Bhoir, Gaurav Sapkale, Anita Mhatre.
<https://www.engpaper.com/download/smart-tourist-guide-system.pdf>
- [5] Trippit: An Optimal Itinerary Generator, 6-14-2019, Andrew Nguyen, Osama Shoubber, Department of Computer Engineering Santa Clara University.
https://scholarcommons.scu.edu/cgi/viewcontent.cgi?article=1154&context=cseng_senior
- [6] Jian Meng,Neng Xu ,“A Mobile Tourist Guide System Based on Mashup Technology“ ISBN978-1-4244- 7618-3 /10 ©2010 IEEE.
https://www.academia.edu/37640871/Tourism_Guide_for_Tamil_nadu_Android_Application
- [7] Xiaoyun shi,“Tour-Guide: Providing Location-Based Tourist Information on Mobile Phones “ISBN 978-1-4244-7547-6/10 @2010 IEEE.
http://ijariie.com/AdminUploadPdf/My_City_Guide_Mobile_Application_ijariie9839.pdf
- [8] Y. H. Yu, “Study on Intelligent Augmented Reality Tourist Guide Application Based on Android Smart Phone,” Appl. Mech. Mater., vol. 668–669, pp. 1399–1402, 2014.

<https://iopscience.iop.org/article/10.1088/1742-6596/1430/1/012038>

[9] P. S. S. Pawar, P. Chavhan, A. Lohar, A. Kadam, and P. Ranjane, "Android Based Tourist Guide System," no. 3, pp. 568–570, 2016.

<https://www.scribd.com/document/428819723/100>

[10] N. B. Nugraha, Suyoto, and Pranowo, "Mobile application development for smart tourist guide," *Adv. Sci. Lett.*, vol. 23, no. 3, pp. 2475–2477, 2017.

https://www.researchgate.net/publication/316995202_Mobile_application_development_for_smart_tourist_guide

D. CERTIFICATE:

**VELAMMAL**
INSTITUTE OF TECHNOLOGY
Approved by AICTE New Delhi & Affiliated to Anna University Chennai
CSE, ECE, EEE, IT, MECH Accredited by NBA & NAAC
"Velammal Knowledge Park", Chennai - Koilatta Highway, Pomeri - 601204



Department of Computer Science and Engineering

International Conference on Innovation in Computing and Technology ‘ 24

CERTIFICATE OF APPRECIATION

**ICICT ‘24**

This is to certify that Mr./Ms. Nivash. V of Sathyabama Institute of Science and Technology has participated
and presented a paper titled Personalized travel itinerary generator system
in ICICT’ 24 (International Conference on Innovation in Computing and Technology) organized by the Department of
Computer Science and Engineering on 27th March 2024 at Velammal Institute of Technology.


HoD (CSE)
Dr.V.P.Gladis Pushparathi


Vice Principal
Dr.S.Soundararajan


Principal
Dr.N.Balaji