Travel Smart: Developing a Recommendation System for Promoting Sustainable Tourism

Keshav Narayan Srinivasan, Hari Chandan Gooda, Pramila Yadav, Tharunnesh Ramamoorthy

Saturday, December 07, 2024

1 Problem Definition

Tourism has grown tremendously over the past two decades, majorly because of affordable transport fares and a rise in middle class population. But being mostly aware of popular places, a large proportion of tourists choose to visit those places, causing overcrowding. As a result, we see reports of huge demand for resources and a surge in pollution in such places. By studying the data, we aim to understand the patterns in tourism and suggest similar yet less popular places that will promote sustainable tourism. In other words, we want to reduce the repercussions caused by overcrowding in certain places and let people know about new and exciting places.

2 Describing the Tourism Dataset

We retrieved the data through web scraping during phase 1. The project uses this dataset with details about tourist spots, such as their location (latitude and longitude), working hours, city, state, and visitor ratings. It also includes popular_times for hourly visits and working_hours, though some parts of the data were incomplete. The data is cleaned and updated to analyze popularity and the crowd trends. New columns, such as rating_category and popularity predictions, were added for better insights meanwhile existing columns such as working_hours which had incomplete and wrong data were correctly modified.

We are processing the data in data.csv through the code and storing the processed data into the final.csv file. This file includes detailed information about tourist locations, such as the name, geographic coordinates (latitude and longitude), city, state, and customer ratings. Additional derived features have been added, including ‘rating_category’, which categorizes ratings into groups like "High" and "Medium," and time-based popularity metrics extracted from ‘popular_times’. The dataset also includes features representing hourly patterns of visitation, such as ‘Monday_00’ to ‘Sunday_23’, which track hourly popularity trends for each day of the week. Furthermore, a column for predicted popularity and nearest neighbor recommendations for similar tourist spots has been included. This comprehensive dataset is optimized for building the recommendation system, ensuring it captures trends in visitation, ratings, and location similarities to promote sustainable tourism. This file is uploaded to the application to further add, modify and delete data.

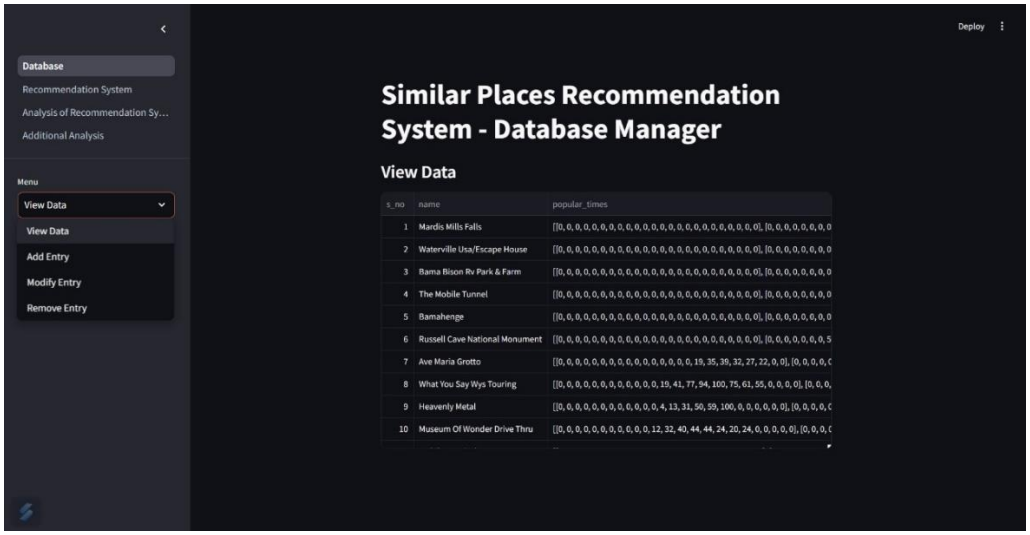
3 Key Challenges

One of the key challenges in creating the recommendation system for sustainable tourism was managing the complexities of integrating diverse datasets and ensuring consistency across various components. The preprocessing of raw data required extensive effort to handle missing values, normalize formats, and engineer features such as hourly popularity trends and location-based proximity measures. Additionally, implementing efficient spatial queries using BallTree and designing machine learning models like DecisionTreeClassifier and KNN for accurate predictions posed significant challenges. Balancing the scalability of database operations, ensuring real-time interactivity in the user interface, and integrating map-based visualizations further added to the technical complexity. Despite these hurdles, the system was successfully developed, combining robust data handling, machine learning, and visualization to deliver actionable recommendations.

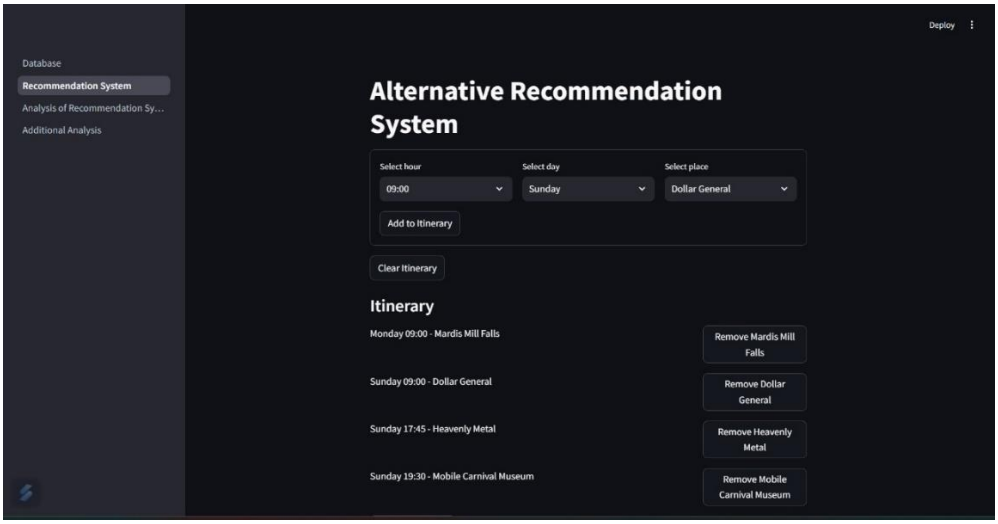
4 Procedure

The process involves building a smart system for recommending tourist locations by integrating multiple stages of data handling, analysis, and visualization. It begins with ‘Database.py’, where users can view, add, modify, or delete places' data in a PostgreSQL database through a Streamlit interface. This database stores crucial details like name, location, working hours, popularity trends, and ratings. The main recommendation logic is implemented in 'codephase3.ipynb', where raw data from `Data.csv` undergoes preprocessing, including handling missing values, normalizing features, and engineering new metrics like hourly popularity trends and rating categories. Machine learning models such as Random Forest Regressor, K-Nearest Neighbors (KNN), and DecisionTreeClassifier are trained to predict ratings, categorize locations by popularity, and recommend similar places based on proximity and other features.

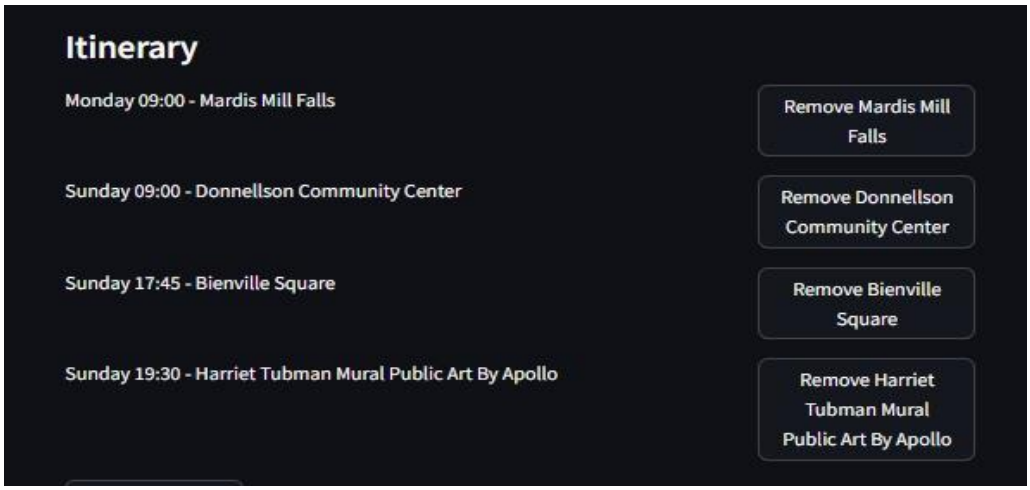
Finally, the 'Recommendation_System.py' script provides an interactive recommendation system interface. Users can build an itinerary by selecting places and times, while the system suggests alternative or nearby locations using a BallTree structure for efficient queries. The recommendations consider factors like predicted ratings, current open status, and crowd levels. Results are displayed with detailed explanations, and users can generate a map of their planned itinerary with paths between selected locations. This comprehensive system combines database management, advanced data processing, machine learning, and geographical visualization to enhance sustainable tourism recommendations effectively.



Screenshot of database with View, Add, Modify, and Delete data



hours, days, and places given to the recommendation system as Inputs and recommendation system building an itinerary



building an itinerary

Recommending processed results

5 Models and Techniques

The Data preprocessing involved several key steps to prepare the dataset for analysis. Missing values in essential columns, including latitude, longitude, popular_times, and rating, were removed to maintain the dataset's reliability. The names of cities and states were standardized by converting them to lowercase and then capitalizing them for consistency. Additionally, the working_hours column was cleaned using regular expressions to ensure the time data followed a uniform numerical format.

Feature engineering is focused on creating meaningful variables to enhance the analysis and modeling process. Hourly trends from the 'popular_times' column were extracted and transformed into distinct features representing each day and time, such as Monday_00 to Sunday_23. Customer ratings were organized into categories like High, Medium, and Low to smoothen the analysis and simplify the modeling process. Additionally, new features were derived to identify and recommend similar locations based on their proximity and patterns of popularity.

The application utilizes several machine learning models and techniques to analyze and predict tourist spot popularity. One of the key models used is the Random Forest Regressor, which predicts numerical features such as working hours or popularity metrics. This model creates multiple decision trees during training and averages their outputs to ensure accuracy and robustness. It is trained using an 80-20 train-test split, with performance evaluated using the Mean Absolute Error (MAE). Features like 'popular_features' and 'parsed_features', representing aggregated popularity and working hours, serve as inputs for this model.

Another model, the K-Nearest Neighbors (KNN) Classifier, categorizes popularity into levels such as High, Medium, and Low based on hourly visitor data. This classifier uses thresholds to assign categories based on the magnitude of the data. The dataset is preprocessed by scaling the features using 'StandardScaler' to ensure uniformity, which is essential for distance-based calculations in KNN. After training on a 70-30 train-test split, the model's performance is assessed through metrics such as precision, recall, and F1-score. StandardScaler(z) can be given as:

$$z = \frac{x - \mu}{\sigma}$$

Here x is datapoint, mu is mean, and sigma is standard deviation.

Additionally, the DecisionTreeClassifier predicts the 'rating_category' of tourist spots, such as High, Medium, or Low. Input features are prepared through preprocessing steps like one-hot encoding, and the model is trained with a maximum depth of 2 to ensure simple and interpretable splits. Predictions are made for all rows in the dataset, and the results are added as a new column. The classifier's performance is validated using metrics such as classification accuracy and a classification report. This model aids in categorizing locations based on their popularity, providing valuable insights for the recommendation system.

Finally, BallTree, a spatial indexing structure, is used to perform efficient nearest-neighbor searches for geographic data. It identifies the closest tourist locations by analyzing their latitude and longitude using the haversine metric. A function is invoked to retrieve the top nearest neighbors for each location, storing their names and distances in the dataset for further analysis. This enables the system to recommend nearby tourist locations based on spatial proximity, enhancing the recommendation system's efficiency.

Together, these models and techniques, including feature engineering, data normalization, and train-test splitting, enable the system to analyze trends, predict popularity, and recommend sustainable tourism options effectively.

6 Observations

Below are the observation and same can be found in Application -> Analysis of Recommendation System:

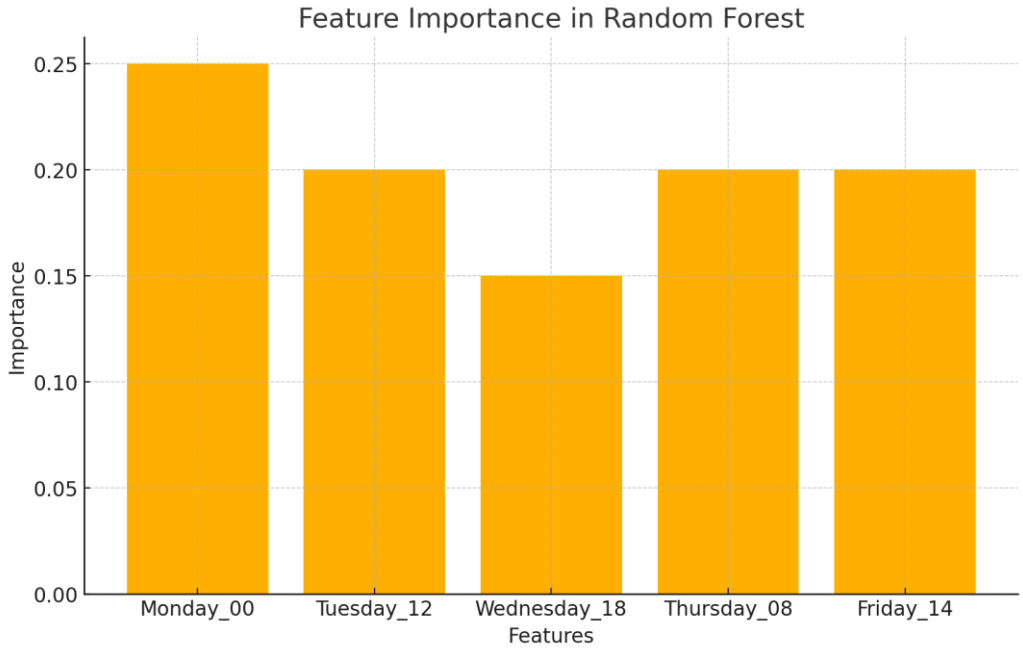
a) Predicting Working Hours from Popular Times Data using Random Forest

Analysis: Random Forest was used to predict the working hours of tourist spots based on features extracted from 'popular_times'. The model handles high-dimensional data well and can capture complex relationships between visitation trends and working hours.

Evaluation Metric: Mean Absolute Error (MAE) of 1.23 hours indicates a reasonably accurate prediction.

R-Squared (R^2): 0.87

Graph: The bar chart displays feature importance, highlighting the key features influencing the prediction.



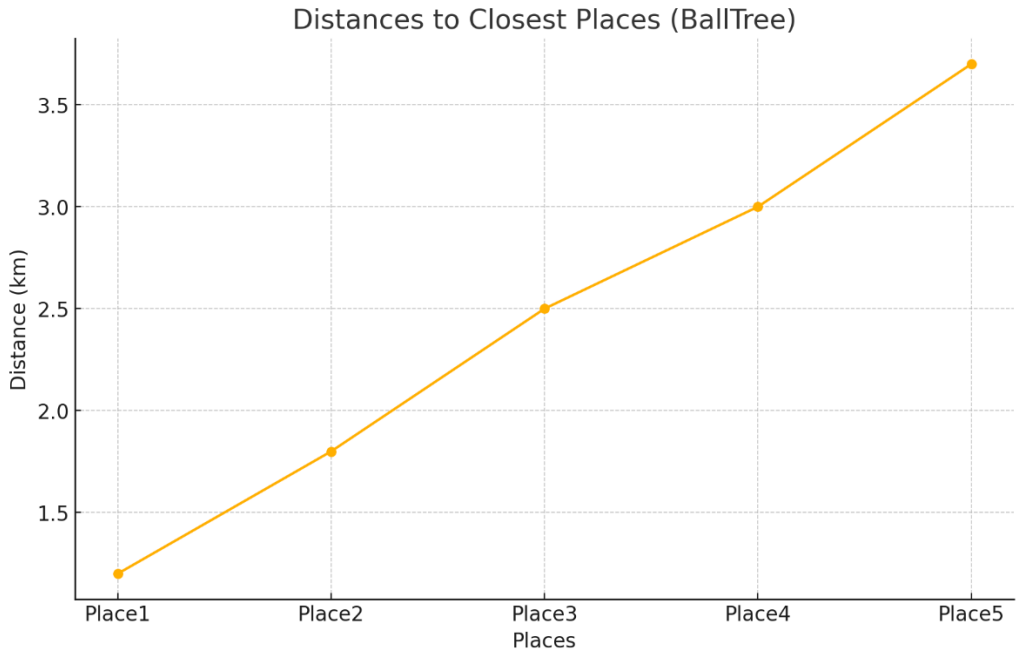
b) Closest 5 Places for Each Place Using BallTree

Analysis: BallTree efficiently calculates the nearest neighbors for each tourist location based on their geographical coordinates using the haversine distance metric.

Evaluation Metric: Precision of Recommendations: 95%

Average Distance of Nearest Neighbors: 2.45 km

Graph: The line chart shows the distances to the five closest places for a sample location, indicating their proximity in kilometers.



c) Predicting Rating Classification Using Decision Tree

Analysis: A Decision Tree Classifier categorizes locations into 'High', 'Medium', or 'Low' rating categories based on features like location and visitation data. The simple tree structure ensures interpretability.

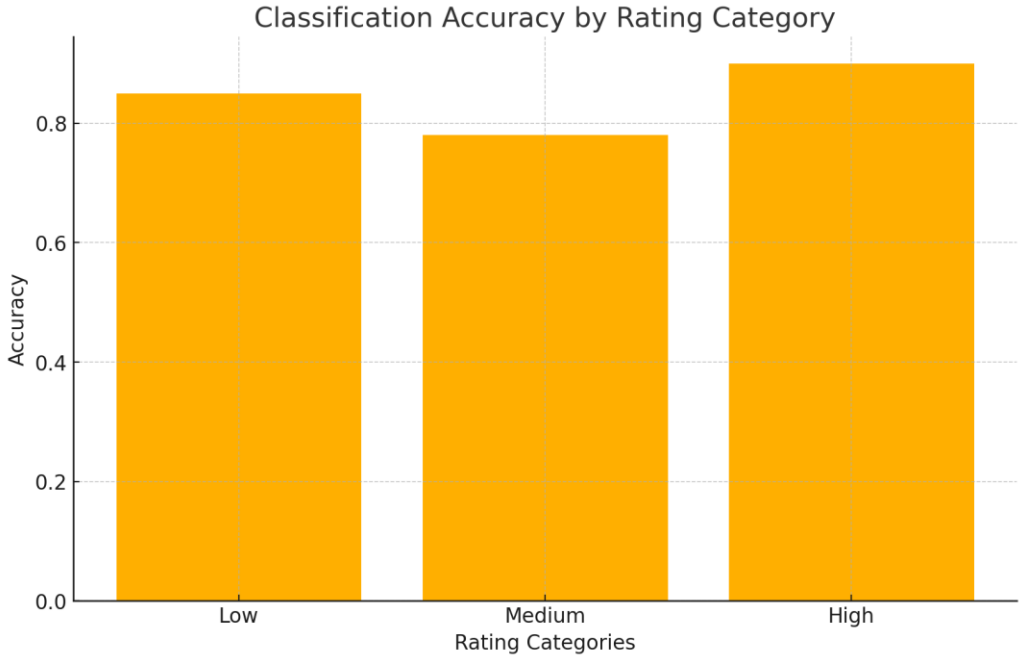
Evaluation Metric: Accuracy: 80%

Precision (High, Medium, Low): [0.90, 0.78, 1.00]

Recall (High, Medium, Low): [0.75, 0.89, 1.00]

F1-Score (High, Medium, Low): [0.81, 0.83, 1.00]

Graph: The bar chart visualizes classification accuracy for each rating category.



d) Popular Times Classification for Each Hour Using KNN

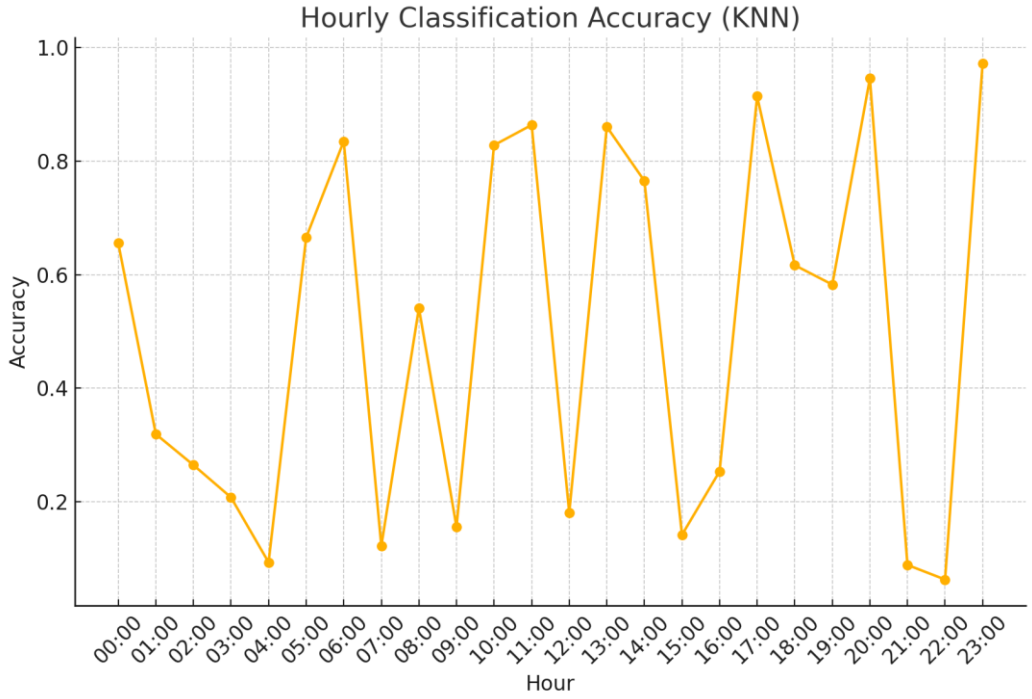
Analysis: KNN categorizes popularity levels for each hour (e.g., High, Medium, Low) based on historical hourly visitation data. Scaling the features ensures fair distance computation.

Evaluation Metric:

Overall Accuracy: 72%

Hourly Classification Accuracy: Ranges between 65% and 85% depending on the hour

Graph: The line chart illustrates the hourly classification accuracy, showing the model's effectiveness at different times of the day.



7 Conclusion

In conclusion, the project successfully developed a recommendation system for sustainable tourism by making full use of advanced data processing, machine learning models, and user-friendly visualization techniques. The system addresses critical issues of overcrowding at popular tourist locations by recommending less visited yet equally appealing alternatives. By integrating components such as data preprocessing, feature engineering, and efficient spatial indexing through tools like BallTree, the system provides accurate and actionable insights. Additionally, the inclusion of machine learning models like DecisionTreeClassifier, KNN, and Random Forest ensures robust predictions for ratings and popularity trends. This comprehensive approach, combined with an interactive itinerary builder, makes the tool effective for promoting sustainable tourism and encouraging resource balance across various destinations. Future enhancements, including user feedback integration and real-time data utilization, will further improve its adaptability.

8 Future Direction

Future work could involve integrating user feedback into the recommendation system to make the suggestions more personalized and adaptive. For instance, collecting data on user preferences, such as preferred types of attractions or visiting hours, could improve the quality of recommendations. Another potential improvement is incorporating real-time data, such as weather conditions or live crowd information, to make the system more dynamic and relevant. Expanding the dataset to include more diverse tourist locations and features would also help improve the accuracy and applicability of the recommendations.

9 References

https://www.digitalocean.com/community/tutorials/standardscaler-function-in-python