Trip planner

Rohan Aditya, 220741, praditya22@iitk.ac.in
Vijay Kumar, 220602, mvijayk22@iitk.ac.in
Maradana Kasi Sri Roshan, 220633, mkasi22@iitk.ac.in
Laveti BhanuPrakash, 220583, lbhanu22@iitk.ac.in

CS685: Data Mining

Abstract

1 Introduction

Problem Statement

The complete codebase for this project is available on GitHub here.

Problem Statement

- **Challenges Faced by Travelers:** Travelers often struggle with deciding the best places to visit, the optimal time for travel, and the most suitable types of attractions based on their preferences.
- **Solution Developed:** A recommendation system was created to provide tailored tourist options by leveraging user inputs such as country, travel mode, and month of visit. The system offers personalized suggestions for attraction types, specific places, and ideal travel months to improve travel planning.

2 Methodology

2.1 Predicting Attraction Types and Places

Input: User's `Country`, `VisitMode`, and `VisitMonth`
Objective:

1. Recommend the top `AttractionTypes` based on user preferences.
2. Based on the selected `AttractionType`, suggest the top `AttractionPlaces`.

Techniques Used:

- **Data Processing**: Label Encoding, One-Hot Encoding

- **Feature Engineering**: Interaction Features (e.g., `VisitMode_VisitMonth`, `VisitMode_Country`)
- **Algorithms**: RandomForest, Logistic Regression, Gradient Boosting, KNN, XGBoost

2.2 Predicting Places and Best Visit Months

Input: User's `Country`, `VisitMode`, and `AttractionType`
Objective:

1. Step 1: Suggest the top `AttractionPlaces` based on user input.
2. Step 2: Recommend the best `VisitMonths` for the chosen attraction.

Techniques Used:

- **Data Processing**: Label Encoding, One-Hot Encoding
- **Feature Engineering**: Interaction Features (e.g., `VisitMode_AttractionType`, `VisitMode_Country`)
- **Algorithms**: RandomForest, Logistic Regression, Gradient Boosting, KNN, XGBoost

Brief Methodology Idea

- **Data Preprocessing**: Handling missing values, outliers, and data discretization to optimize dataset quality.
- **Feature Engineering**: Creating interaction features to capture deeper patterns and improve model accuracy.
- **Model Building**: Training multiple machine learning models to identify the best predictors for attraction types, specific places, and visit months.
- **Evaluation**: Selecting the optimal model based on accuracy and performance metrics, focusing on RandomForest due to its high accuracy and interpretability.

3 Motivation of the Problem

- Tourism plays a vital role in fostering cultural exchange, economic growth, and personal recreation. However, travelers often struggle to make informed decisions due to the vast array of tourist destinations and travel options, especially in regions like Bali, Malang, and Yogyakarta, which offer diverse attractions. This overwhelming variety can lead to suboptimal travel experiences.

- In today's digital era, personalization is key to enhancing user satisfaction, but existing travel planning tools often fail to integrate factors such as travel mode, timing, and destination preferences into a cohesive framework. This project aims to address this gap by developing a recommendation system that provides tailored suggestions for tourist attractions and travel modes based on user inputs. By simplifying decision-making and enhancing user experience, this system not only meets the expectations of modern travelers but also promotes tourism in the target regions.

4 Data Used

For our project, we utilized a dataset from **Mendeley Data**, available at the following link: Mendeley Data Dataset.

The dataset comprises **9 Excel files**, each containing information about users, countries, regions, attraction places, and their corresponding visit details. Below is a detailed description of the data:

1. **City.xlsx**
   Contains information about various cities.
   - **CityId**: Unique identifier for each city.
   - **CityName**: The name of the city (e.g., New York, Paris).
   - **CountryId**: Indicates which country the city belongs to.
2. **Continent.xlsx**
   Provides information about continents.
   - **ContinentId**: Unique identifier for each continent.
   - **Continent**: The name of the continent (e.g., Asia, Europe).
3. **Country.xlsx**
   Contains data related to countries.
   - **CountryId**: Unique identifier for each country.
   - **Country**: The name of the country (e.g., USA, India).
   - **RegionId**: Identifies the region to which the country belongs.
4. **Item.xlsx**
   Stores information about tourist attractions.
   - **AttractionId**: Unique identifier for each attraction place.
   - **AttractionCityId**: The city where the attraction is located.
   - **AttractionTypeId**: Identifies the type of the attraction (e.g., Historical, Nature).
   - **Attraction**: The name of the attraction (e.g., Eiffel Tower, Taj Mahal).
   - **AttractionAddress**: The full address of the attraction.
5. **Mode.xlsx**
   Provides data about the modes of travel used by users.
   - **VisitModeId**: Unique identifier for each visit mode.
   - **VisitMode**: The mode of visit (e.g., Car, Train, Flight).

6. **Region.xlsx**
   Contains data about regions within continents.
   - **RegionId**: Unique identifier for each region.
   - **Region**: The name of the region (e.g., South Asia, Western Europe).
   - **ContinentId**: Indicates which continent the region belongs to.
7. **Transaction.xlsx**
   Captures detailed information about each user's visit to attractions.
   - **TransactionId**: Unique identifier for each transaction.
   - **UserId**: Unique identifier assigned to each user.
   - **VisitYear**: The year the attraction was visited.
   - **VisitMonth**: The month the attraction was visited (e.g., January, June).
   - **VisitMode**: Mode of travel used for the visit (e.g., Flight, Car).
   - **AttractionId**: The specific attraction visited.
   - **Rating**: The rating given by the user for the attraction (scale of 1 to 5).
8. **Type.xlsx**
   Contains data about types of attractions.
   - **AttractionTypeId**: Unique identifier for each attraction type.
   - **AttractionType**: The type of attraction (e.g., Historical, Beach, Cultural).
9. **User.xlsx**
   Provides demographic information about the users.
   - **UserId**: Unique identifier for each user.
   - **ContinentId**: The continent the user is from.
   - **RegionId**: The region within the continent.
   - **CountryId**: The country of the user.
   - **CityId**: The city of the user.

Additional Details:

- The dataset required extensive preprocessing to clean and integrate data from multiple files.
- The **Transaction.xlsx** file was the primary source for building our models, as it contained the relevant fields needed to predict the best attraction types, visit months, and places based on user preferences.
- Features such as **VisitMode**, **AttractionType**, and **VisitMonth** were crucial in developing a robust recommendation system.

5 Methodology

This section outlines the steps involved in the **Data integration**, **Data handling**, **Finding Duplicates**, **Outlier Detection**, **EDA**, **Analysis of Correlation Matrix**, **Association Rule Mining** and **Methodology for Web Application Development Using Machine Learning Mod- els** of the dataset to develop our tourist recommendation models.

5.1 Data Integration

Given the structure of our dataset, which comprised **9 different Excel files**, the primary objective was to consolidate these datasets into a single, unified file for further analysis and model building. The integration was performed using the `Transaction.xlsx` file as it contained the most relevant information for our analysis.

After merging all the datasets, we created a single integrated CSV file named **`Integrated_data.csv`** consiting of **61,949** entries with the following columns:

1. **TransactionId**: Unique identifier for each transaction.
2. **UserId**: Unique identifier assigned to each user.
3. **VisitYear**: The year the visit to the attraction took place.
4. **Rating**: User rating of the attraction (on a scale of 1 to 5).
5. **Continent**: Continent to which the user belongs.
6. **Region**: Region within the continent.
7. **Country**: Country of the user.
8. **City**: City of the user.
9. **Attraction**: The name of the attraction visited.
10. **AttractionAddress**: Address of the attraction.
11. **VisitMode**: Mode of travel used (e.g., Car, Train, Flight).
12. **VisitMonth**: The month when the visit occurred.
13. **AttractionType**: Type of the attraction (e.g., Historical, Nature).

5.2 Data Handling

During the data integration process, we encountered columns with **missing values**:

- **VisitMonth**: 3,117 missing entries.
- **AttractionType**: 1,959 missing entries.

We used the following strategies to handle the missing data:

5.2.1 Handling Missing Data in the VisitMonth Column

1. **Data Type Issue**: The `VisitMonth` column was observed to have an `object` data type instead of `int64`. Upon inspecting the data, we found non-numeric values such as: `[nan, 'August', 'June', 'January', 'March', 'February', 'December', 'November', 'October', 'September', 'May', 'July', 'April']`.
2. **Data Conversion**: We converted the **month names to numeric values**:
   - 'January' $\rightarrow$ 1, 'February' $\rightarrow$ 2, ..., 'December' $\rightarrow$ 12.

   After conversion, we plotted the numeric values against their frequency and identified **out-of-range entries**.
3. **Out-of-Range Data Handling**:
   - Removed data with out-of-range entries.
   - Removed rows with `NaN` values in `VisitMonth`. The missing entries (3,117 rows) were relatively few compared to our total dataset size of **61,949** rows.

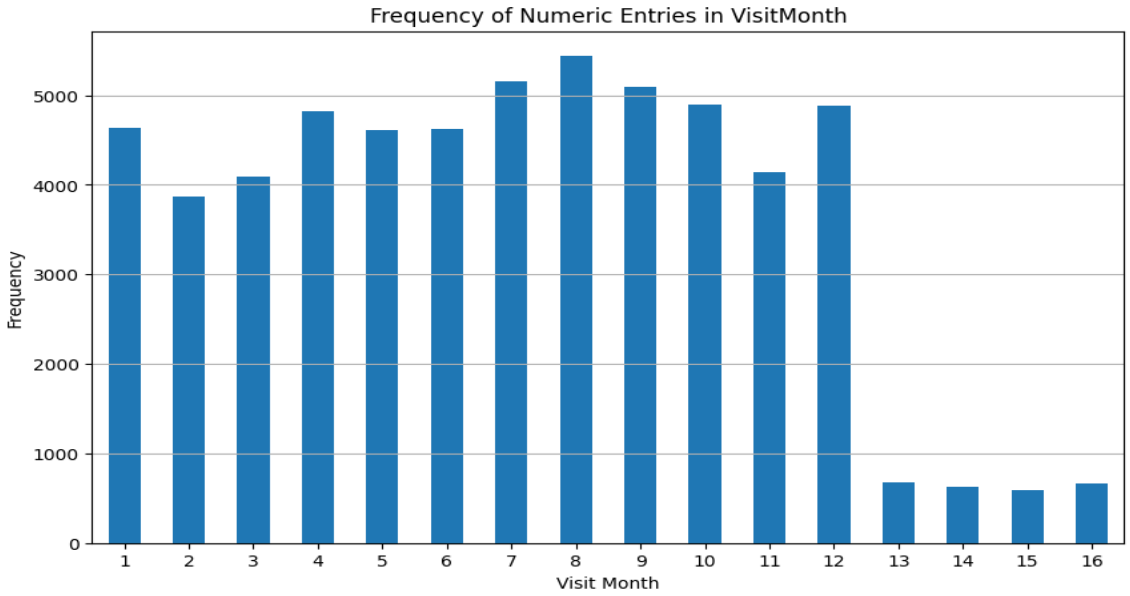

Figure 1: Out-of-Range Data Handling

5.2.2 Handling Missing Data in the AttractionType Column

1. **Mapping Attractions to Attraction Types**: We established a mapping between `Attraction` and `AttractionType` based on existing data, excluding rows with `NaN` in `AttractionType`. We then replaced the `NaN` entries using this mapping.

2. **Final Attraction Types Identified**: Our dataset contains 4 distinct attraction types:

   - **Nature & Wildlife**
   - **Leisure & Entertainment**
   - **Historical & Cultural**
   - **Cultural & Local Exploration**

5.2.3 List of Attractions by Type

- **Cultural & Local Exploration**: Tegalalang Rice Terrace, Jodipan Colorful Village, Malang City Square, Museum Malang Tempo Doeloe, Malioboro Road, Water Castle (Tamansari).
- **Historical & Cultural**: Uluwatu Temple, Tanah Lot Temple, Ratu Boko Temple, Sewu Temple, Ullen Sentalu Museum, Yogyakarta Palace.
- **Leisure & Entertainment**: Waterbom Bali, Bromo Tengger Semeru National Park, Khayangan Reflexology & Massage, Kalibiru National Park, Ramayana Ballet at Prambanan.

- **Nature & Wildlife**: Sacred Monkey Forest Sanctuary, Seminyak Beach, Nusa Dua Beach, Sanur Beach, Tegenungan Waterfall, Kuta Beach - Bali, Mount Semeru Volcano, Sempu Island, Balekambang Beach, Coban Rondo Waterfall, Goa Cina Beach, Jomblang Cave, Merapi Volcano.

5.3 Finding Duplicates

To ensure data quality, we checked for duplicate rows:

- **Number of duplicate rows before removal**: 0

This indicates that our dataset did not contain any duplicate entries, ensuring that the data was clean and ready for analysis.

5.4 Outlier Detection

During our data exploration, we aimed to identify and handle outliers, especially in the `Ratings` column.

1. **Distribution Analysis**: We plotted various graphs, such as "Distribution of Ratings by AttractionType and VisitMode vs Rating" to visualize how ratings were distributed across different categories.
2. **Investigating Outliers**: For entries with ratings of 1 or 2, we investigated potential patterns by plotting multiple graphs:

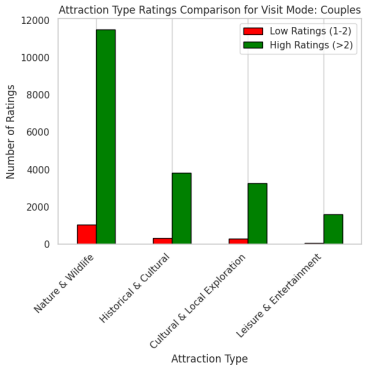

Figure 2: Couples

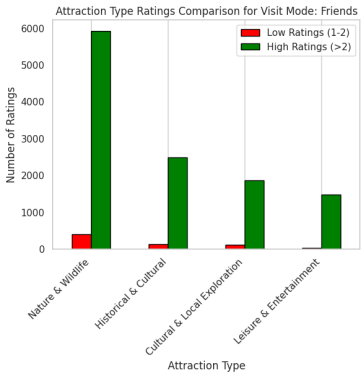

Figure 3: Friends

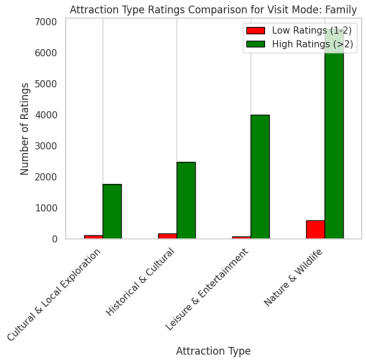

Figure 4: Family

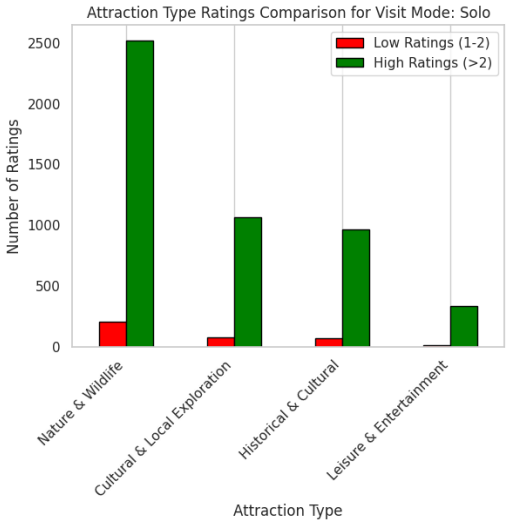

Figure 5: Attraction Type Ratings: Solo

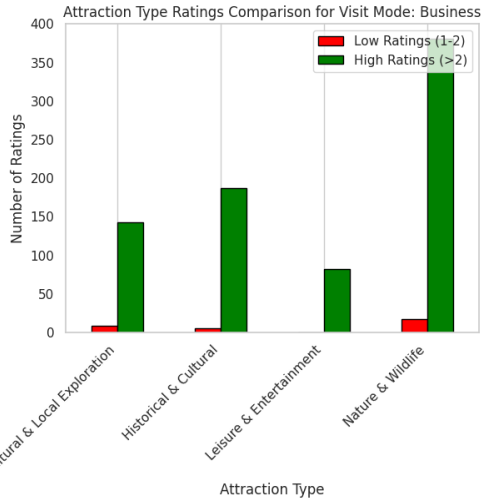

Figure 6: Attraction Type Ratings: Business

After examining these plots, we found no discernible patterns. Hence, we removed rows with ratings less than 3.

3. **Action Taken**:

   - Rows removed: 3,689 (ratings ¡ 3).
   - Original dataset shape after data handling: (56,273 rows, 13 columns).
   - Cleaned dataset shape: (52,584 rows, 13 columns).

5.5 Data Reduction

To simplify our analysis, we reduced the dataset by selecting only the most relevant features:

- **Features to Keep**: VisitMonth, Rating, Country, City, Attraction, VisitMode, AttractionType.
- **Features to Remove**: TransactionId, UserId, Continent, Region, VisitYear, AttractionAddress.
- **Reason**: These columns were either redundant, irrelevant to the prediction task, or contained information that was not necessary for our analysis.

5.6 Data Discretization

1. **Convert VisitMonth to Seasons**: We transformed the `VisitMonth` column into a new `Season` column:
    - Winter: December, January, February.
    - Spring: March, April, May.
    - Summer: June, July, August.
    - Fall: September, October, November.
2. **Convert Rating to Categories**: We simplified the `Rating` column by creating a new `RatingCategory`:
    - Low: Ratings of 1-2.
    - Medium: Ratings of 3-4.
    - High: Ratings of 5.

5.7 Exploratory Data Analysis (EDA)

5.7.1 Number of Tourists Visiting Each Attraction for Different Visit Modes

- Couples: Sacred Monkey Forest Sanctuary is the top attraction, significantly outpacing other destinations preferred by couples.
- Friends: Sacred Monkey Forest Sanctuary and Merapi Volcano dominate as popular spots among friend groups.
- Family: Waterbom Bali stands out as the top destination for families, suggesting a preference for recreational activities.

- Solo: Sacred Monkey Forest Sanctuary and Tegallalang Rice Terrace are favored among solo travelers for their serene ambiance.
- Business: Merapi Volcano and Malioboro Road are leading attractions for business travelers, indicating a blend of cultural and natural interests.

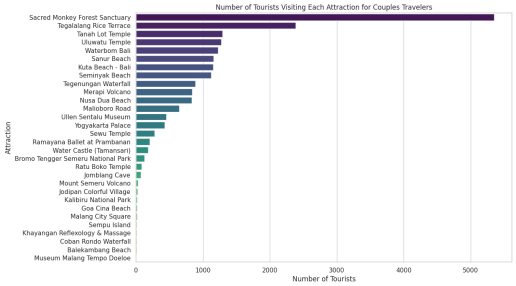

Figure 7: Couples Travelers

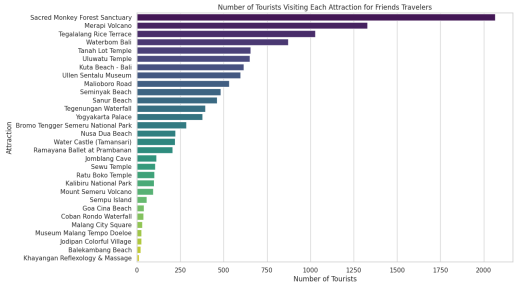

Figure 8: Friends Travelers

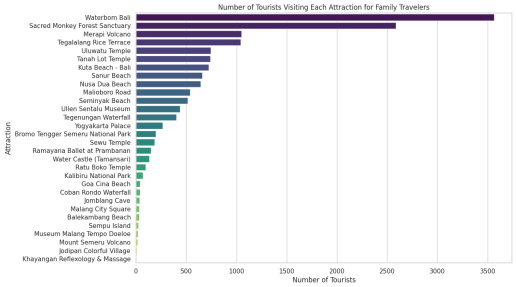

Figure 9: Family Travelers

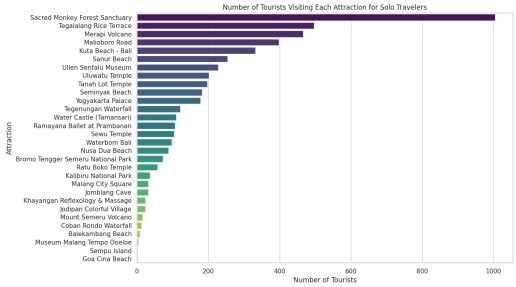

Figure 10: Solo Travelers

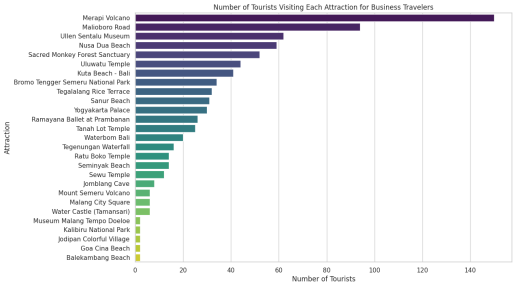

Figure 11: Business Travelers

5.7.2 Average Rating and Number of Ratings for Each Attraction

- **Top Rated Attractions:** Mount Semeru Volcano, Bromo Tengger Semeru National Park, and Waterbom Bali have the highest average ratings, indicating exceptional visitor satisfaction despite having fewer ratings.
- **Highly Reviewed but Lower Ratings:** Attractions like Malioboro Road, Kuta Beach, and Yogyakarta Palace receive a high volume of ratings but have relatively lower average scores.

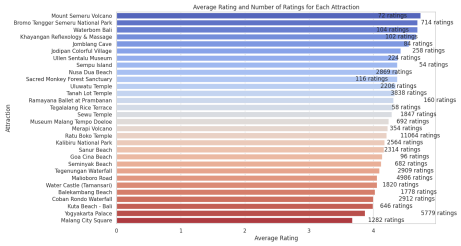

Figure 12: Attraction Ratings Analysis

5.7.3 Top 20 Countries by Visit Count

- **Top Contributors:** Australia, the United States, and the United Kingdom lead in tourist visits, indicating a strong affinity from Western countries.
- **Regional Influence:** Neighboring countries like Singapore, Malaysia, and India also rank highly due to proximity and ease of travel.

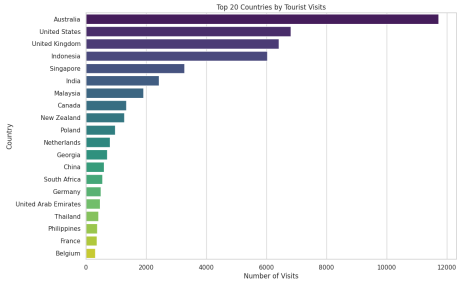

Figure 13: Top Countries by Visit Count

5.7.4 Top 20 Attractions by Visit Count

- **Top Attractions:** Sacred Monkey Forest Sanctuary and Waterbom Bali are the most visited attractions, highlighting a preference for cultural and recreational experiences.
- **Diverse Interests:** The top attractions include a mix of natural and cultural sites, appealing to a wide range of tourist interests.

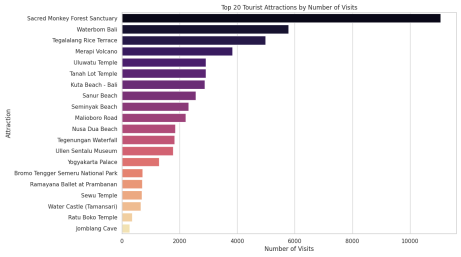

Figure 14: Top Attractions by Visit Count

5.7.5 Visit Mode Preference by Season

- **Couples and Family Travel Dominate:** Across all seasons, couples and family travel modes are most prevalent, especially in fall.
- **Low Business Travel:** Business travel is consistently low, indicating tourism is largely driven by leisure rather than work-related visits.

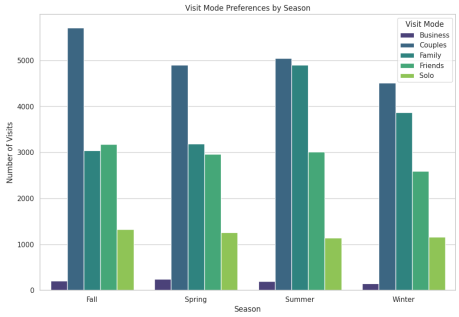

Figure 15: Visit Mode by Season

5.8 Analysis of Correlation Matrix

To better understand the relationships between different features, we generated a **correlation matrix** using heatmaps.

5.8.1 Heatmaps to Understand Feature Relationships

We plotted a heatmap of the correlation matrix to visualize the relationship between attributes such as `Country`, `VisitMonth`, `VisitMode`, `Attraction`, and `AttractionType`.

The correlation matrix reveals that most attributes have **near-zero correlations** with each other, indicating that factors like `Country`, `VisitMonth`, `VisitMode`, and `AttractionType` are largely independent. This independence suggests that these features do not strongly influence one another, which is beneficial for making unbiased predictions in our recommendation models.

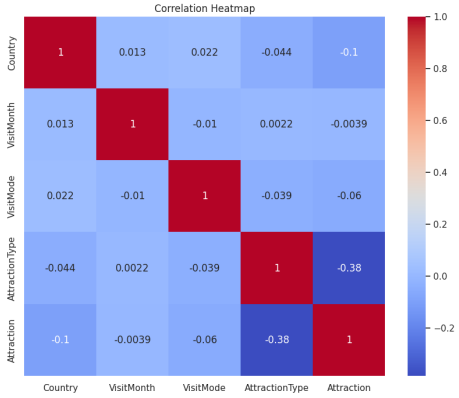

Figure 16: Heatmap of Correlation Matrix

5.8.2 Notable Negative Correlation

A significant observation was the strongest **negative correlation** between:

- **Attraction Type** and **Attraction**: Correlation coefficient of **-0.38**

This suggests that certain types of attractions attract specific kinds of travelers. The preference for an `AttractionType` could influence the choice of specific `Attractions`, which is crucial for enhancing the accuracy of our recommendation system.

5.9 Association Rule Mining

To improve the recommendation system, we applied **Association Rule Mining** to identify patterns and relationships between features. This technique was particularly useful for predicting various attributes:

5.9.1 Predicting AttractionType

- **Key Relationship**: The combination of `VisitMode`, `Country`, and `VisitMonth` was effective for predicting the `AttractionType`.
- **Explanation**: This shows that tourists' travel modes, influenced by their country of origin and the season, play a significant role in determining the type of attraction they are likely to visit.
- **Impact**: Leveraging these features allows for a more **tailored recommendation system**, enhancing predictions for the attraction types tourists might prefer.

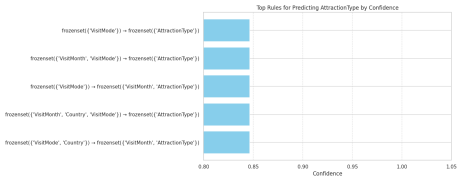

Figure 17: Association Rules for AttractionType

5.9.2 Predicting Specific Attractions

- **Key Relationship**: For predicting specific attractions, combining `VisitMode`, `Country`, and `VisitMonth` has shown **high confidence levels**.
- **Explanation**: This indicates that tourists' preferences for particular attractions are shaped by their travel mode, country of origin, and the season.
- **Impact**: Utilizing these combinations helps in accurately suggesting attractions, optimizing recommendations based on tourists' profiles.

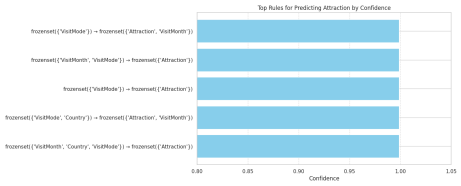

Figure 18: Association Rules for Attractions

5.9.3 Predicting VisitMonth

- **Key Relationship**: The combination of `VisitMode`, `AttractionType`, and `Country` was effective in predicting the preferred `VisitMonth`.
- **Explanation**: This shows that tourists' choices are influenced by their travel mode, the type of attraction they seek, and their country of origin.
- **Impact**: By integrating these attributes, the model can align with **seasonal trends**, enhancing the relevance of visit time recommendations.

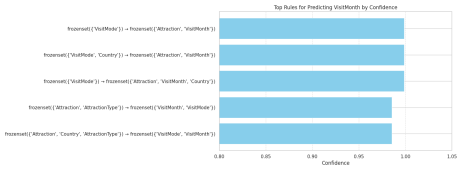

Figure 19: Association Rules for VisitMonth

5.10 Web Application Integration of Machine Learning Models

This section details the methodology for developing a web application that leverages machine learning models to recommend tourist attractions. The application includes both front-end and back-end components, integrating pre-trained models stored as pickle files. The key steps in this structured approach are outlined below.

5.10.1 Front-End Development

The front-end of the web application is built using **HTML**, **CSS**, and **JavaScript**, with additional libraries like **jQuery** and **Select2** for enhanced interactivity. The main goal of the front-end is to provide users with an intuitive and responsive interface to interact with the application.

- **HTML Structure**: The application is structured with multiple dashboards to handle different functionalities, such as recommending attraction types, suggesting the best months to visit, and listing top tourist places. Each dashboard allows users to input parameters like country, visit mode, and attraction type through dropdown menus.
- **CSS Styling**: A consistent and clean user interface is ensured using custom CSS. Styling elements like buttons, forms, and navigation menus enhance the user experience.
- **JavaScript & jQuery**: JavaScript is used to dynamically handle user interactions, such as changing dropdown options and handling button clicks. jQuery is employed for easier DOM manipulation and AJAX requests to communicate with the back-end server. The `Select2` library is utilized for enhancing the dropdown menus, allowing for a smooth and searchable selection experience.

- **Navigation between Dashboards**: The application is designed with multiple dashboards that can be navigated through buttons. The current dashboard is highlighted, and others are hidden to keep the interface clean and focused.

5.10.2 Back-End Development

The back-end is developed using **Python** and **Flask**, a lightweight web framework that handles server-side operations. This layer is responsible for managing user requests, processing data, and returning predictions based on machine learning models.

- **Flask Application**: The Flask server handles multiple endpoints to serve different functionalities, such as predicting attraction types, recommending places, and suggesting the best months to visit. The endpoints are designed to accept JSON data from the front-end, process the inputs, and return predictions.
- **Data Handling**: When a request is received from the front-end, the data is extracted and validated. The inputs (like country, visit mode, and attraction type) are transformed into numerical representations using pre-trained label encoders.
- **Model Integration**: The Flask server loads pre-trained machine learning models stored in `.pkl` files using Python's `pickle` module. These models are utilized to generate predictions based on user inputs. The use of pickled models ensures that the application can serve predictions efficiently without needing to retrain models on the fly.

5.10.3 Machine Learning Models and Pickle Files

To recommend the most relevant tourist attractions, pre-trained **Random Forest** models and other machine learning classifiers are used. These models have been trained offline using historical data and stored as `pickle` files for deployment.

- **Model Training**: The models were initially trained in a Jupyter Notebook using datasets containing information about tourist attractions, visit modes, and seasonal preferences. The training process included data preprocessing, feature engineering, and hyperparameter tuning to optimize model performance.
- **Pickle Serialization**: Once the models were trained, they were serialized using Python's `pickle` module. This involves saving the model along with label encoders and other preprocessing utilities in `.pkl` files. These files are then stored on the server for quick loading during runtime.
- **Model Loading in Flask**: During the initialization of the Flask server, the pickled models and encoders are loaded into memory using `pickle.load()`. This ensures that predictions can be made in real-time as user requests come in, without the overhead of retraining models.

- **Prediction Workflow**:
    1. The Flask server receives input data from the front-end via an AJAX call.
    2. The inputs are encoded using pre-loaded label encoders.
    3. The processed inputs are passed to the relevant pre-trained model to generate predictions.
    4. The predicted outputs are sent back to the front-end, which displays the recommendations to the user.

5.10.4 Data Flow and Communication Between Front-End and Back-End

The web application follows a structured data flow to ensure seamless communication between the front-end and back-end components:

- **AJAX Requests**: The front-end uses AJAX to send data (like selected country, visit mode, etc.) to the Flask server endpoints. These requests are asynchronous, allowing the page to remain responsive while waiting for the server's response.
- **JSON Responses**: The Flask server processes the inputs, runs the necessary predictions using the loaded models, and returns the results as JSON objects. These responses are parsed by the front-end to display the recommended attraction types, places, or months.
- **Error Handling**: Both the front-end and back-end include error handling mechanisms. For example, if the user inputs are invalid or if the server encounters an issue while processing predictions, appropriate error messages are displayed.

5.10.5 Conclusion

The web application integrates machine learning models using Flask for the back-end, HTML/CSS/JavaScript for the front-end, and pre-trained models stored as pickle files. This setup enables efficient real-time predictions, enhancing the user's travel planning experience while ensuring scalability and maintainability.

6 Model Building and Evaluation

In this section, we discuss the models developed to predict various attributes using machine learning techniques. Each model focuses on predicting specific outputs based on different combinations of inputs.

6.1 Predicting Attraction Types (Model-1.1)

Objective: Given a user's `Country`, `VisitMode`, and `VisitMonth`, predict the top 2 `AttractionTypes`.
Steps:

1. **Label Encoding**: Encoded categorical variables: `Country`, `VisitMode`, `AttractionType`, and `Attraction`.
2. **One-Hot Encoding**: Applied one-hot encoding to the encoded data for the feature `Country`.
3. **Feature Engineering**:
    - Created interaction features based on association rule insights:
    - `VisitMode_VisitMonth`: Created by combining `VisitMode` and `VisitMonth`.
    - `VisitMode_Country`: Created by multiplying `VisitMode` with the one-hot encoded `Country`.
4. **Training**:
    - **Input Features** (`X_train_features`):
        - `Country_OneHot`, `VisitMode`, `VisitMonth`, `VisitMode_VisitMonth`, `VisitMode_Country`
    - **Output Feature** (`y_train_feature`): `AttractionType`
5. **Model Evaluation**: Trained models using various algorithms: KNN, XGBoost, Logistic Regression, Gradient Boosting, and RandomForest.

6.2 Predicting Attractions Based on Attraction Type (Model-1.1)

Objective: Given a user's `Country`, `VisitMode`, `VisitMonth`, and selected `AttractionType`, predict the top 3 `Attractions`.
Steps:

1. **Label Encoding**: Encoded categorical variables: `Country`, `VisitMode`, `AttractionType`, `Attraction`.
2. **One-Hot Encoding**: Applied one-hot encoding to the `Country`.
3. **Feature Engineering**: Created interaction features:
    - `VisitMode_VisitMonth`
    - `VisitMode_Country`
4. **Training**:

- **Input Features**: `AttractionType`, `Country_OneHot`, `VisitMode`, `VisitMonth`, `VisitMode_VisitMonth`, `VisitMode_Country`
- **Output Feature**: `Attraction`

5. **Model Evaluation**: Trained models using various algorithms: KNN, XGBoost, Logistic Regression, Gradient Boosting, and RandomForest.

6.3 Predicting Attraction Places Based on Attraction Type (Model-2.1)

Objective: Given a user's `Country`, `VisitMode`, and `AttractionType`, predict the top 3 `AttractionPlaces`.

Steps:

1. **Label Encoding**: Encoded categorical variables: `Country`, `VisitMode`, `AttractionType`, `Attraction`.
2. **Training**:
   - **Input Features**: `Country`, `VisitMode`, `AttractionType`
   - **Output Feature**: `Attraction`
3. **Model Evaluation**: Trained models using various algorithms: KNN, XGBoost, Logistic Regression, Gradient Boosting, and RandomForest. Overall Mean Test Accuracy: 58.80%

6.4 Predicting Visit Month (Model-2.2)

Objective: Given a user's `Country`, `VisitMode`, `AttractionType`, and `Attraction`, predict the top 3 `VisitMonths`.

Steps:

1. **Label Encoding**: Encoded categorical variables: `Country`, `VisitMode`, `AttractionType`, `Attraction`.
2. **One-Hot Encoding**: One-hot encoded the `Country`.
3. **Feature Engineering**: Created interaction features:
   - `VisitMode_AttractionType`
   - `VisitMode_Country`
4. **Training**:
   - **Input Features**: `Country_OneHot`, `VisitMode`, `AttractionType`, `VisitMode_AttractionTy` `VisitMode_Country`

- **Output Feature**: `VisitMonth`

5. **Model Evaluation**: Trained models using various algorithms: KNN, XGBoost, Logistic Regression, Gradient Boosting, and RandomForest.

7 Results

Based on the models developed and the analysis conducted, we summarize the key results as follows:

7.1 Predicting Attraction Types

- **Model-1.1** using `Country`, `VisitMode`, and `VisitMonth` as inputs achieved an accuracy of **53%** using a RandomForest model.
- **Key Insight**: The addition of interaction features (`VisitMode_VisitMonth` and `VisitMode_Country`) improved model performance, highlighting the influence of country and travel season on the type of attractions tourists are likely to prefer.

7.2 Predicting Specific Attractions

- **Model-1.2**For predicting attractions based on the selected `AttractionType`, the RandomForest model achieved varying accuracy levels:
  - Nature & Wildlife: 45%
  - Leisure & Entertainment: 83%
  - Historical & Cultural: 42%
  - Cultural & Local Exploration: 72%
- **Key Insight**: The highest accuracy was observed for `Leisure & Entertainment`, suggesting that tourists' preferences in this category are easier to predict based on their travel mode, country, and the season.

7.3 Predicting Attraction Places

- **Model-2.1** using `Country`, `VisitMode`, and `AttractionType` as inputs the RandomForest model achieved an overall mean accuracy of **58.8%**.
- Accuracy varied significantly by `AttractionType`:
  - Nature & Wildlife: 45.57%
  - Leisure & Entertainment: 81.18%

  - Historical & Cultural: 40.68%
  - Cultural & Local Exploration: 67.78%

- **Key Insight**: The model was most effective for predicting places within the `Leisure & Entertainment` category, suggesting clear patterns in traveler preferences for this attraction type.

7.4 Predicting Visit Month

- **Model-2.2** using `Country`, `VisitMode`, `AttractionType`, and `Attraction` as inputs the RandomForest model resulted in an average accuracy of **26%** across all attractions.
- Best Accuracy: 60% for some attractions, while the lowest accuracy was 9%.
- **Key Insight**: Predicting the optimal `VisitMonth` proved challenging due to seasonal variability, with better performance for attractions that have a strong seasonal preference.

7.5 Correlation Analysis

- The correlation matrix revealed that most attributes have **low correlation**, indicating independence between features such as `Country`, `VisitMonth`, `VisitMode`, and `AttractionType`.
- Notable negative correlation between `AttractionType` and `Attraction` (**-0.38**) suggests that certain types of attractions attract specific travelers, impacting their choices.

7.6 Association Rule Mining

- The combination of `VisitMode` with `Country` and `VisitMonth` was highly effective for predicting `AttractionType` and specific `Attractions`.
- Utilizing features like `VisitMode` and `AttractionType` for predicting the preferred `VisitMonth` enhanced the model's alignment with seasonal trends.
- **Key Insight**: The rules discovered were effective in providing tailored recommendations, indicating that user preferences are influenced by a combination of travel mode, country of origin, and travel season.

7.7 Attraction Type Preferences by Country

Country-wise Preferences:

- Australia shows a strong preference for **Nature & Wildlife** attractions, surpassing other countries, indicating an inclination towards natural explorations.
- In contrast, **Indonesia** and the **United States** exhibit a balanced interest across various attraction types, reflecting diverse tourism preferences.

Attraction Type Dominance:

- **Cultural & Local Exploration** and **Historical & Cultural** sites are favored by tourists from **Indonesia**, **India**, and **Malaysia**, highlighting a cultural affinity in these regions.
- **Leisure & Entertainment** attractions are less popular, suggesting a preference for more authentic experiences over purely recreational ones.

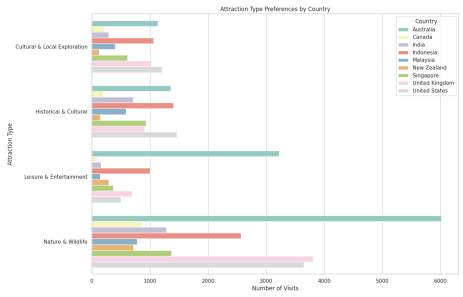

Figure 20: Attraction Preferences by Country

7.8 Popularity of Attraction Types by Season

Dominance of Nature & Wildlife:

- **Nature & Wildlife** remains the most popular attraction type across all seasons, peaking during **summer** and **winter**, showing a consistent draw throughout the year.

Seasonal Variation in Preferences:

- **Cultural & Local Exploration** and **Historical & Cultural** attractions maintain steady popularity in **fall**, **spring**, and **summer**, indicating that interest in these categories is less season-dependent.
- **Leisure & Entertainment** remains the least preferred across all seasons, suggesting tourists prioritize more immersive experiences.

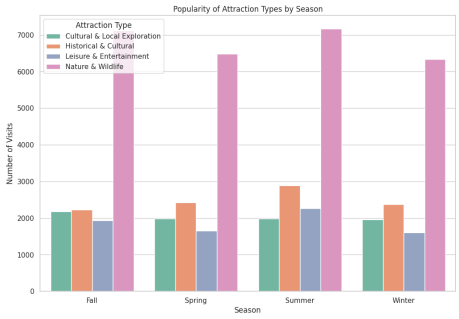

Figure 21: Attraction Type Popularity by Season

7.9 General Observations

- **Sacred Monkey Forest Sanctuary** consistently ranked as a top attraction across multiple traveler categories, showing its universal appeal.
- **Leisure & Entertainment** attractions had the highest predictability, likely due to their consistent popularity and user satisfaction.
- **Tourist Preferences** varied significantly based on travel mode, with families and couples preferring recreational spots, while solo travelers leaned towards serene natural locations.

8 Conclusions and Future Directions

The current system effectively provides personalized recommendations by generating three types of suggestions based on user inputs, specifically:

1. Recommending the top attraction types based on the user's country, visit mode, and preferred month.
2. Offering the best places to visit after selecting an attraction type.
3. Suggesting the most suitable months to visit a place based on the user's preferences.

8.1 Future Development 1

Enhance another model which will suggest the best months to visit based on the user's country, mode of visit, and attraction type. Once a month is selected by the interest of the user, it will recommend top-rated tourist attractions tailored to the chosen inputs.

8.2 Future Development 2

Integrate the address of the attraction places and their ratings into the recommendation models to provide more precise suggestions. This will allow users to choose destinations based on popularity, distance, and user reviews, further improving travel planning.

9 Team Contribution

The success of this project is attributed to the collective efforts of the team members, each of whom contributed significantly to its development. The contributions of each team member are outlined below:

- **Bhanu:**
  - Worked on suggesting the top attraction type based on user inputs (user's country, visit mode, visit month).
  - Contributed to both front-end and back-end tasks.
  - Helped in preparing the project report and resolving coding errors.
- **Vijay:**
  - Focused on suggesting the top attraction places based on user inputs (user's country, visit mode, visit month, and attraction type).
  - Assisted in front-end and back-end work.
  - Participated in report writing and debugging the code.
- **Rohan:**
  - Worked on recommending the top attraction places based on user inputs (user's country, visit mode, and attraction type).
  - Supported front-end and back-end tasks.
  - Helped with report preparation and fixing errors in the code.
- **Roshan:**
  - Developed the functionality to recommend the best visit month based on user inputs (user's country, visit mode, attraction type, and attraction place).

- Contributed to both front-end and back-end development.
- Assisted in report writing and troubleshooting the code.