Travel Planner using Machine Learning

Siddhant Pahade

6th May, 2023

Problem statement:

The tourism industry is a significant contributor to the global economy. However, with so many travel options and destinations available, it can be overwhelming for tourists to decide where to go and what to do. To make the travel planning process more efficient and personalized, a tourism app can be developed that uses machine learning (ML) algorithms to suggest travel destinations, activities, and accommodations based on user preferences and behavior.

The main challenge in developing such an app is to design and implement an ML model that can effectively analyze user data and generate personalized recommendations. The ML model needs to be trained on a large dataset of user profiles, travel history, and feedback to learn user preferences and behavior patterns. The model also needs to consider various factors such as weather, season, availability, and price to suggest relevant travel options.

Target customers:

- *Frequent travelers*
- *Vacation planners*
- *Adventure seekers*
- *Budget-conscious travelers*
- *Family / Solo travelers*

External search:

Dataset used: https://www.kaggle.com/datasets/rkiattisak/traveler-trip-data

Applicable constraints:

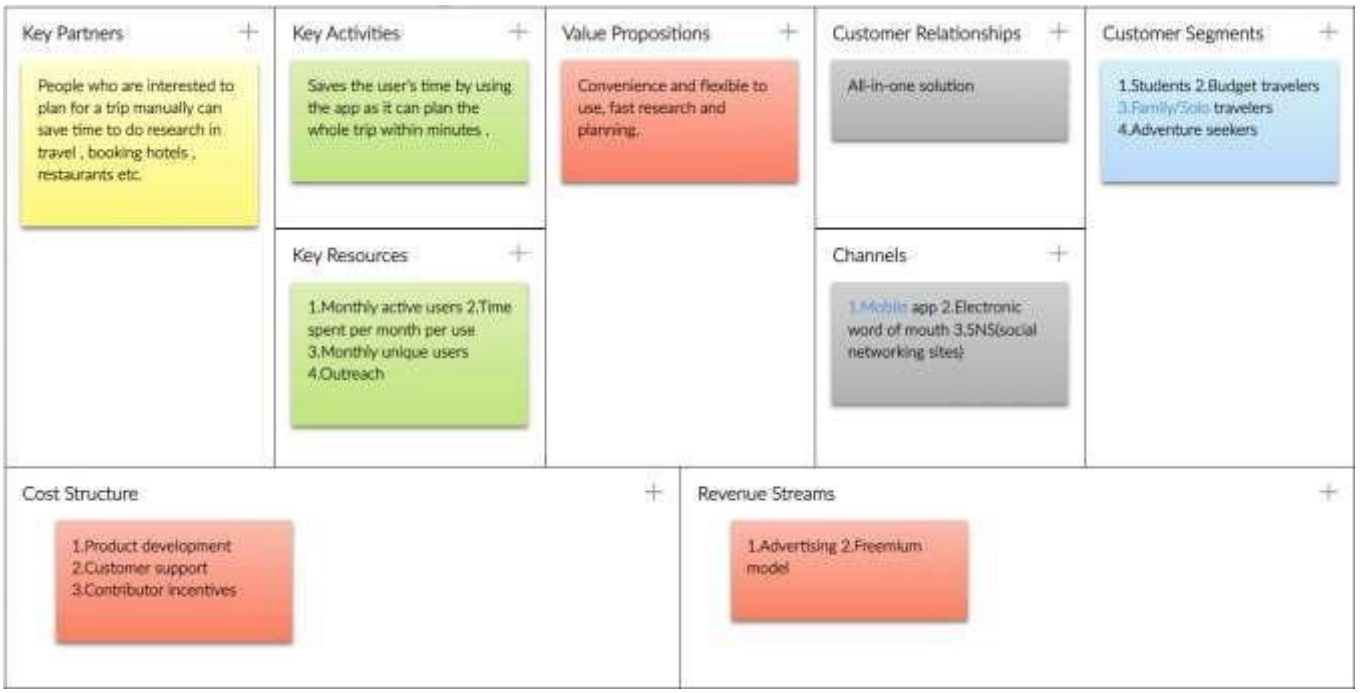
- ***Data availability and quality:*** *The effectiveness of the ML model depends on the quality and quantity of data available. However, data collection and management can be a challenging task in the tourism industry, and there may be limited data available for certain destinations or activities.*
- ***Privacy and security:*** *The app needs to comply with privacy regulations and protect user data from unauthorized access and misuse. The ML model needs to be trained on user data while ensuring anonymity and confidentiality.*
- ***Computational resources:*** *ML algorithms require significant computational resources, and the app needs to be designed to handle large datasets and complex computations efficiently. This may require the use of cloud-based services or dedicated hardware resources.*
- ***Interpretability and explainability:*** *The ML model needs to be transparent and explainable to build user trust and confidence. It should be possible to understand how the recommendations are generated and provide users with explanations and justifications for the recommendations.*

Bench marking:

- ***Personalization:*** *The ML model can analyze user data and behavior to provide personalized recommendations tailored to their interests and preferences. This can enhance the travel experience and increase user satisfaction.*
- ***Efficiency:*** *The app can automate the travel planning process and provide recommendations quickly and efficiently, saving users time and effort.*
- ***Accuracy:*** *The ML model can analyse large datasets and consider multiple factors to provide accurate and relevant recommendations for travel destinations, activities, and accommodations.*
- ***Improved decision making:*** *The app can provide users with comprehensive information and insights on travel options, allowing them to make more informed and confident decisions.*
- ***Competitive advantage:*** *Travel businesses can use the app to differentiate themselves from competitors and offer personalized and innovative services to their customers.*
- ***Data-driven insights:*** *The app can generate valuable insights and trends on user preferences and behaviour that can be used by travel businesses to optimize their services and offerings.*

Overall, a tourism app with ML capabilities can provide significant benefits to users and travel businesses by offering personalized and efficient travel planning services that enhance the travel experience and drive business growth.

Business model:



Applicable patents:

The frameworks used in the model are:

- *Numpy and Pandas : for dataframes and computations*
- *Matplotlib : for visualizations and analysis using plots*
- *Folium : for creating interactive maps and visualizations*

Code implementation & concept development:

Read the data:

```
#importing required libraries
import os
import re
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import seaborn as sns
import folium
from folium.plugins import FastMarkerCluster
```

```
#reading the dataset
df = pd.read_csv("/content/complete_travel_destination_data.csv")
```

```
df.head(5)
```

	Type	Name	Address	Grade	District	AGA Division	PS/MC/UC	Full_Add	Lat	Lon
0	Restaurants	BAVARIAN BARN RESTAURANT	NO. 11, GALLE FACE COURT 02 COLOMBO	A	Colombo	Colombo	Colombo Divisional Secretariat	NO. 11, GALLE FACE COURT 02 COLOMBO, Colombo, ...	80.016670	6.866670
1	Restaurants	LORDS RESTAURANT	80B PORUTHOTA ROAD ETTHUKALA NEGOMBO	A	Gampaha	Negombo	Negombo Divisional Secretariat	80B PORUTHOTA ROAD ETTHUKALA NEGOMBO, Gampaha,...	79.843249	7.206416
2	Restaurants	NIHONBASHI RESTAURANT	NO 11, GALLE FACE TERRACE, COLOMBO 03	A	Colombo	Colombo	Colombo Divisional Secretariat	NO 11, GALLE FACE TERRACE, COLOMBO 03, Colombo...	80.016670	6.866670
3	Restaurants	FREE WIND RESTAURANT & BAR	NO 1285, KANDY ROAD, PALAIYOOTHU, TRINCO MALEE	A	Trincomalee	Uppuveli Trinco Malee	Trincomalee Divisional Secretariat	NO 1285, KANDY ROAD, PALAIYOOTHU, TRINCO MALEE...	80.750000	7.250000
4	Restaurants	EDWIN RESTAURANT	204, LEWIS PLACE, NEGOMBO	B	Gampaha	Negombo	Negombo Divisional Secretariat	204, LEWIS PLACE, NEGOMBO, Gampaha, Sri Lanka	79.843249	7.206416

Firstly , start by importing all the required libraries to visualize the dataset and the second line reads the CSV file into a pandas DataFrame and assigns it to the variable 'df'. The next line prints the first few rows of the DataFrame using the 'head()' function.

```
# check stats of dataset
df.describe()
```

	Lat	Lon
count	1518.000000	1518.000000
mean	80.043295	7.009534
std	2.153123	1.285656
min	-2.701457	5.964618
25%	79.898096	6.866670
50%	80.016670	6.866670
75%	80.016670	7.031163
max	81.849686	53.954905

```
# Checking the null values
df.isnull().sum()
```

```
Type             0
Name             0
Address          0
Grade         1847
District         1
AGA Division     1
PS/MC/UC         1
Full_Add         1
Lat              1
Lon              1
dtype: int64
```

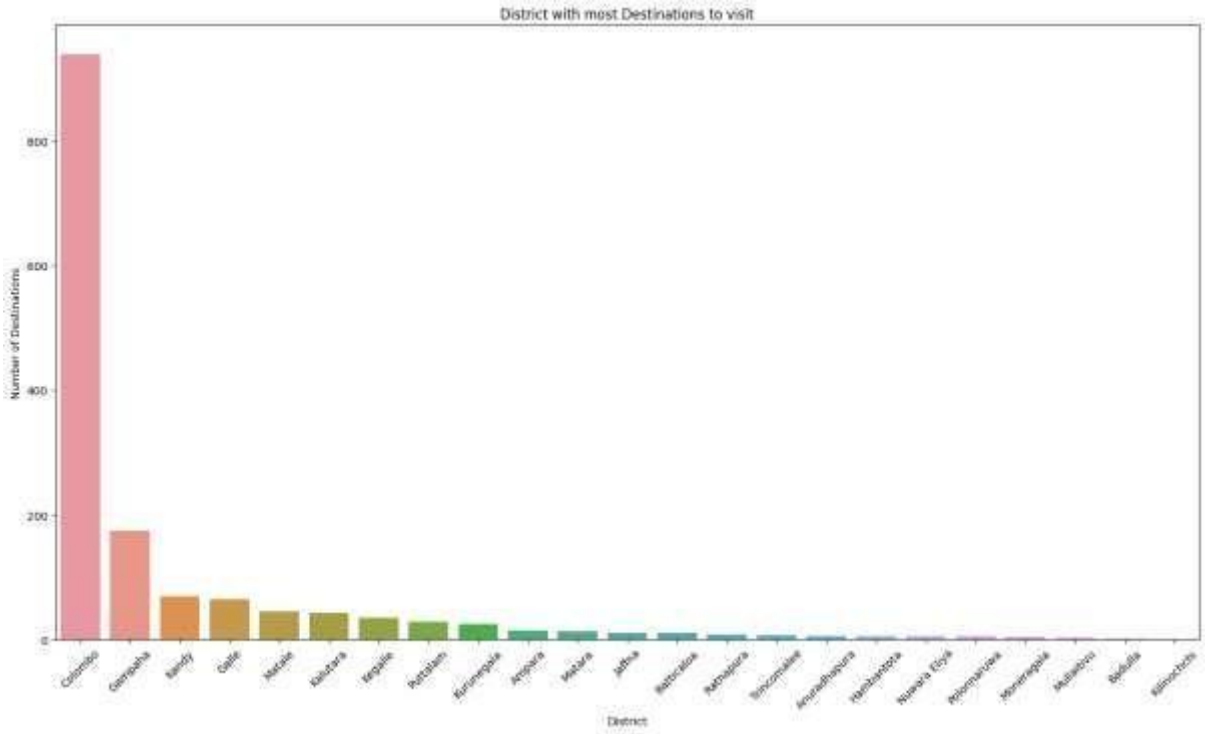
The next line prints summary statistics about the numerical columns in the DataFrame using the 'describe()' function. The next line uses the isnull() function to create a Boolean

DataFrame where True values represent null values and False values represent non-null values. The sum() function is then applied to count the number of null values in each column of the DataFrame.The output will show the number of null values in each column of the DataFrame. If a column has a high number of null values, it may indicate missing data or other issues with the dataset that need to be addressed.

▾ Data Visualization

```
[ ] # District with most Destinations to visit

    top_d_type = df[["Address", "District"]].groupby("District").agg(['count'])['Address']['count'].sort_values(ascending=False)
    ax = sns.barplot( x = top_d_type.index, y = top_d_type.values)
    ax.set_xticklabels(labels=top_d_type.index , rotation=45)
    fig = plt.gcf()
    fig.set_size_inches(10.5, 10, forward=True)
    ax.set_ylabel("Number of Destinations")
    ax.set_xlabel("District")
    ax.set_title("District with most Destinations to visit");
```



This code generates a horizontal bar plot using the Seaborn library (sns.barplot()) to show the number of tourist destinations in each district. Here's a step-by-step breakdown of what the code is doing:

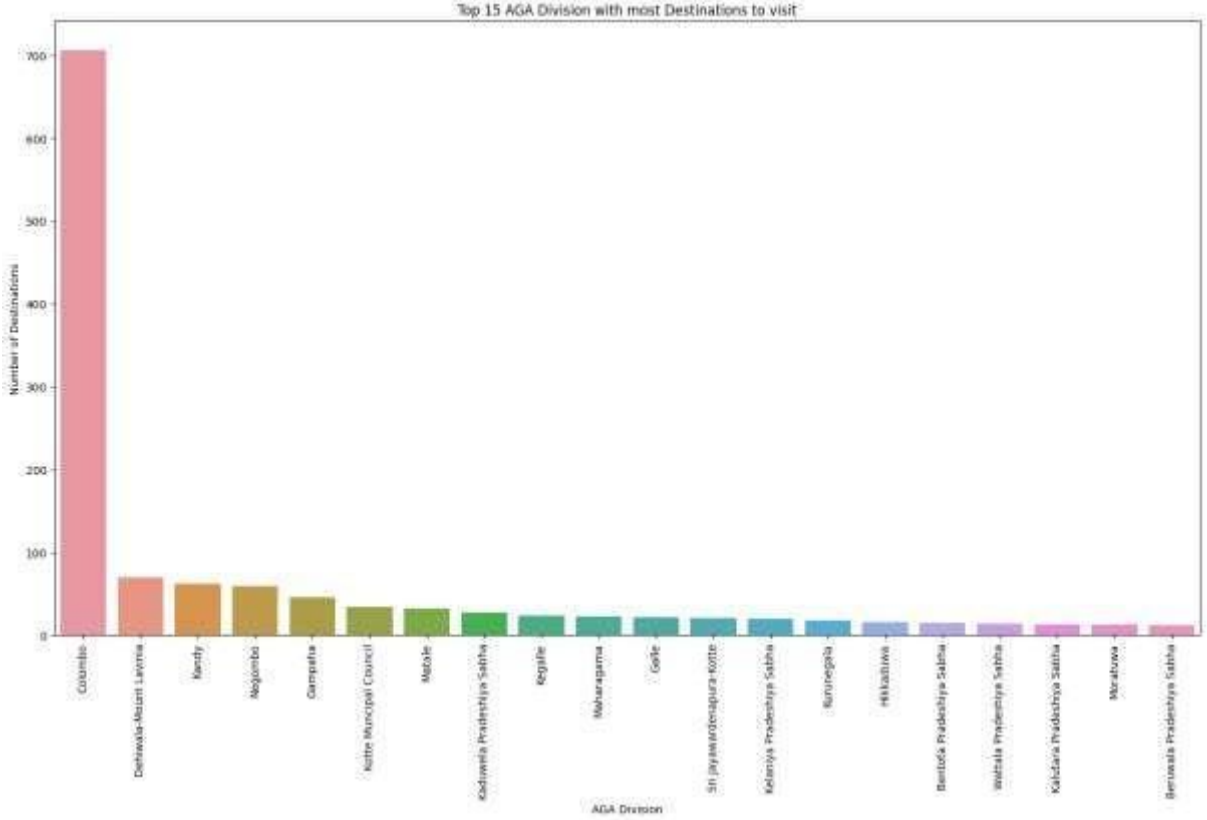
- *The first line of code selects two columns from the pandas DataFrame df: "Address" and "District". The groupby() method is then called to group the data by "District", and the agg() method is called with "count" as the aggregation function. This counts the number of destinations in each district and returns a DataFrame. The indexing is then used to select only the "count" column and assign it to the variable top_d_type. Finally, sort_values() is used to sort the resulting Series in descending order by the count of destinations.*
- *The second line of code creates a horizontal bar plot using Seaborn's sns.barplot() function. The x-axis represents the districts and the y-axis represents the count of destinations in each district. The top_d_type.index and top_d_type.values are passed as arguments to the x and y parameters, respectively.*

- *The third line of code rotates the x-axis labels by 45 degrees using the set_xticklabels() method. This is done to prevent the labels from overlapping and becoming unreadable.*
- *The fourth line of code sets the size of the figure using the set_size_inches() method.*
- *The fifth and sixth lines of code set the y and x axis labels using the set_ylabel() and set_xlabel() methods, respectively.*
- *The last line of code sets the title of the plot using the set_title() method.*

Overall, this code is used to create a horizontal bar plot that shows the number of tourist destinations in each district in a visually appealing and informative way.

```
# Top 20 AGA Division with most Destinations to visit

top_d_type = df[["Address", "AGA Division"]].groupby("AGA Division").agg(['count'])['Address']['count'].sort_values(ascending=False)[:20]
ax = sns.barplot( x = top_d_type.index, y = top_d_type.values)
ax.set_xticklabels(labels=top_d_type.index , rotation=90)
fig = plt.gcf()
fig.set_size_inches(18.5, 10, forward=True)
ax.set_ylabel("Number of Destinations")
ax.set_xlabel("AGA Division")
ax.set_title("Top 15 AGA Division with most Destinations to visit");
```

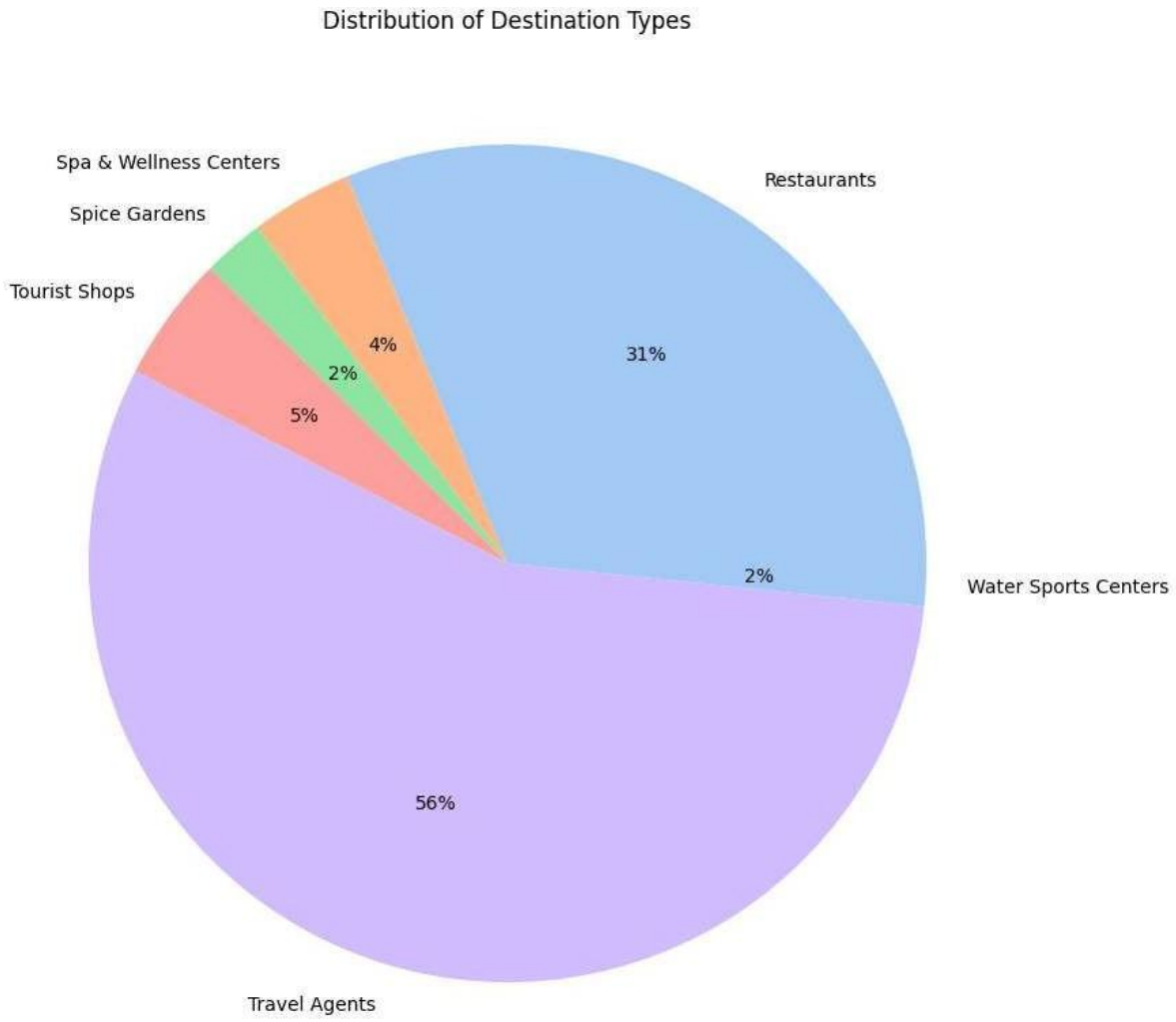


resulting Series in descending order by the count of destinations, and only the top 20 rows are selected using slicing ([:20]).

- *The second line of code creates a horizontal bar plot using Seaborn's sns.barplot() function. The x-axis represents the AGA divisions and the y-axis represents the count of destinations in each AGA division. The top_d_type.index and top_d_type.values are passed as arguments to the x and y parameters, respectively.*
- *The third line of code rotates the x-axis labels by 90 degrees using the set_xticklabels() method. This is done to prevent the labels from overlapping and becoming unreadable.*
- *The fourth line of code sets the size of the figure using the set_size_inches() method.*
- *The fifth and sixth lines of code set the y and x axis labels using the set_ylabel() and set_xlabel() methods, respectively.*
- *The last line of code sets the title of the plot using the set_title() method.*

Overall, this code is used to create a horizontal bar plot that shows the top 20 AGA divisions with the most tourist destinations in a visually appealing and informative way.

```
# Disrtribution of Destination Types
colors = sns.color_palette('pastel')[0:5]
df_dest = df[["Address", "Type"]].groupby("Type").agg(['count'])['Address']['count']
plt.pie(df_dest, labels = df_dest.index, colors = colors, autopct='%.0f%%')
fig = plt.gcf()
fig.set_size_inches(18.5, 10, forward=True)
plt.title("Distribution of Destination Types")
plt.show()
```



This code generates a pie chart using Matplotlib and Seaborn libraries to display the distribution of destination types in the dataset. Here is a step-by-step breakdown of what the code is doing:

- *The first line of code creates a list of 5 pastel colors using Seaborn's color_palette() method and slices the first five colors using indexing. These colors will be used to represent the slices in the pie chart.*
- *The second line of code selects two columns from the pandas DataFrame df: "Address" and "Type". The groupby() method is then called to group the data by "Type", and the agg() method is called with "count" as the aggregation function. This counts the number of destinations in each type and returns a DataFrame. The indexing is then used to select only the "count" column and assign it to the variable df_dest.*
- *The third line of code creates a pie chart using Matplotlib's pie() function. The df_dest and df_dest.index are passed as arguments to the x and labels parameters, respectively. The colors argument is set to the colors list created in the first line, and the autopct parameter is set to format the percentages to no decimal places.*
- *The fourth line of code sets the size of the figure using the set_size_inches() method.*

- *The fifth line of code sets the title of the plot using the title() method.*
- *The last line of code displays the plot using the show() method.*

Overall, this code is used to create a simple and visually appealing pie chart that shows the distribution of destination types in the dataset.

▾ Visualize destinations on map

```
[ ] folium_map = folium.Map(location=[7.8731, 80.7718],
                            zoom_start=8,
                            tiles='CartoDB dark_matter')

    df_map = df[df['Lat'].notna() & df['Lon'].notna() ]

    FastMarkerCluster(data=list(zip(df_map['Lon'].values, df_map['Lat'].values))).add_to(folium_map)
    folium.LayerControl().add_to(folium_map)
    folium_map
```

This code uses the Folium library to create an interactive map displaying the locations of destinations in the dataset. Here is a step-by-step breakdown of what the code is doing:

- *The first line of code creates a Map object with a specified location and zoom level. The location is set to [7.8731, 80.7718], which is a latitude and longitude coordinate that is used to center the map on Sri Lanka. The zoom_start parameter sets the initial zoom level of the map, and the tiles parameter is set to 'CartoDB dark_matter' to choose a dark-colored map style.*
- *The second line of code creates a new DataFrame called df_map that only contains rows where both the "Lat" and "Lon" columns are not null. This is done using the notna() method to filter out rows that have missing latitude or longitude values.*
- *The third line of code creates a FastMarkerCluster layer using the df_map DataFrame. The data parameter is set to a list of tuples containing the latitude and longitude values from df_map. This creates a layer of clustered markers on the map that represent the locations of the destinations in the dataset.*
- *The fourth line of code adds a LayerControl to the map. This allows the user to toggle the visibility of different layers on the map, such as the marker cluster layer.*

- *The last line of code returns the folium_map object, which can be displayed in a Jupyter notebook or saved as an HTML file.*

Overall, this code is used to create an interactive map that displays the locations of destinations in Sri Lanka. The FastMarkerCluster layer is used to improve performance by clustering markers together when zoomed out, and the LayerControl allows the user to easily toggle the visibility of different layers on the map.

By making use of these predictions we can build the app which includes all the features that are mentioned above. The model needs to be trained on a huge amount of data to suggest/plan an itinerary for the whole trip including the cost.

Overall, a travel app can provide users with a convenient and personalized way to plan and book travel, as well as navigate and explore unfamiliar destinations. Incorporating machine learning can make the app even more intelligent and personalized, enhancing the user experience.

***