# **Project: Tourist Activity Prediction for Bhutan Tourism**

## **Overview**

In this project, you will develop a machine-learning model to predict tourist activities in Bhutan. You'll work through data collection, analysis, model building, and deployment using a Streamlit application.

**Date of Submission:** 23rd Feb.2025

**Presentation:** 23rd Feb.2025

### **Step 1: Project Setup**

**Install Required Tools:**

Python (preferably 3.8+)

IDE (VS Code, Jupyter Notebook, etc.)

**Install Python Libraries:**  
pip install beautifulsoup4 pandas numpy scikit-learn matplotlib seaborn streamlit plotly

### **Step 2: Data Collection and Preprocessing**

**Web Scraping with Beautiful Soup:**

Identify reliable websites for Bhutan tourism data (e.g., tourist activities, weather, seasonality).

import requests

from bs4 import BeautifulSoup

url = 'https://example.com/bhutan-tourism'

response = requests.get(url)

soup = BeautifulSoup(response.text, 'html.parser')

**Data Cleaning:**

Remove irrelevant data.

Handle missing values and duplicates.

Convert data into structured formats (CSV/applicable format).

### **Step 3: Exploratory Data Analysis (EDA) & Feature Engineering**

**Exploratory Data Analysis:**

* Use pandas, matplotlib, and seaborn (optional) for data visualization.
* Analyze trends, correlations, and patterns in tourist activities.

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

df = pd.read\_csv('Name of data file.csv')

# correlation

sns.heatmap(df.corr(), annot=True, cmap='coolwarm')

plt.show()

**Feature Engineering:**

* Create new features from existing data (e.g., season, weather conditions, expenditure).
* Encode categorical variables (e.g., activity type, location).
* Normalize or standardize data if needed.

### **Step 4: Data Visualization**

Data visualization helps in understanding patterns, trends, and relationships in the data, which can improve your feature selection and model performance.

#### **1. Univariate Analysis (Analyzing a Single Variable)**

**Histograms:** Show the distribution of numerical data.  
  
df['number\_of\_tourists'].hist(bins=20, color='skyblue')

plt.xlabel('Number of Tourists')

plt.ylabel('Frequency')

plt.title('Distribution of Tourist Numbers')

plt.show()

**Bar Charts:** Great for categorical data.  
  
df['activity\_type'].value\_counts().plot(kind='bar', color='green')

plt.title('Popular Tourist Activities')

plt.xlabel('Activity Type')

plt.ylabel('Count')

plt.show()

#### **2. Bivariate Analysis (Comparing Two Variables)**

**Scatter Plots:** To identify relationships between two numerical variables.  
  
plt.scatter(df['average\_temperature'], df['number\_of\_tourists'], alpha=0.5)

plt.title('Tourists vs. Temperature')

plt.xlabel('Average Temperature (°C)')

plt.ylabel('Number of Tourists')

plt.show()

**Box Plots:** To analyze the spread and detect outliers.  
  
sns.boxplot(x='season', y='number\_of\_tourists', data=df)

plt.title('Tourist Activity Across Seasons')

plt.show()

#### **Multivariate Analysis (Comparing Multiple Variables)**

**Heatmap for Correlation Matrix:** To identify the correlation between different variables.  
  
correlation\_matrix = df.corr()

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm')

plt.title('Correlation Heatmap')

plt.show()

**Pair Plot:** To visualize pairwise relationships between variables.  
  
sns.pairplot(df, hue='season')

plt.show()

#### **Advanced Interactive Visualizations with Plotly (optional)**

**Interactive Line Chart:**  
import plotly.express as px

fig = px.line(df, x='month', y='number\_of\_tourists', title='Tourist Trends Over Months')

fig.show()

**Interactive Map (If You Have Geographical Data):**  
  
fig = px.scatter\_geo(df,

lat='latitude',

lon='longitude',

color='activity\_type',

size='number\_of\_tourists',

projection="natural earth",

title='Tourist Activities Across Bhutan')

fig.show()

### **Step 5: Building and Training Machine Learning Models**

**Model Selection:**

Start with basic models like Decision Trees, Random Forests, or Logistic Regression.

**Splitting the Data:**  
from sklearn.model\_selection import train\_test\_split

X = df.drop('target\_activity', axis=1)

y = df['target\_activity']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

**Model Training:**  
  
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier()

model.fit(X\_train, y\_train)

### **Step 6: Evaluating Model Performance**

**Performance Metrics:**

Accuracy, Precision, Recall, F1-Score

Confusion Matrix for classification problems

**Evaluation Example:**  
from sklearn.metrics import classification\_report, confusion\_matrix

y\_pred = model.predict(X\_test)

print(classification\_report(y\_test, y\_pred))

sns.heatmap(confusion\_matrix(y\_test, y\_pred), annot=True, cmap='Blues')

plt.show()

### **Step 7: Building the Streamlit Application (Optional)**

**Install Streamlit (if not done earlier):**  
pip install streamlit

**Create an app.py File:**  
import streamlit as st

import pandas as pd

import pickle

# Load trained model

model = pickle.load(open('model.pkl', 'rb'))

st.title("Bhutan Tourist Activity Prediction")

# Input form

feature1 = st.number\_input("Enter Feature 1")

feature2 = st.selectbox("Choose Feature 2", ["Option1", "Option2"])

if st.button("Predict"):

prediction = model.predict([[feature1, feature2]])

st.success(f"Predicted Activity: {prediction[0]}")

**Run the App:**  
streamlit run app.py

### **Step 8: Final Report and Submission**

1. **Documentation:**
   * Describe your data collection process, analysis, model, and app development.
2. **Submission Package:**
   * Python scripts/notebooks
   * Clean dataset
   * Trained model (.pkl file)
   * Streamlight app (app.py)
   * Project report (PDF/Word)