TOURIST PREDICTION AND ANALYSIS

Project Based Learning (PBL) Report
for the course
STATISTICS FOR MACHINE LEARNING-20MA32001

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

By

A.Pallavi Nagalakshmi	22R11A0596
A.Sravya Sree	22R11A0598
G.Ramya Sree	22R11A05B4

Under the guidance of
Dr.V.S.Triveni

Department of Computer Science and Engineering
Accredited by NBA

Geethanjali College of Engineering and Technology
(UGC Autonomous)
(Affiliated to J.N.T.U.H, Approved by AICTE, New Delhi)
Cheeryal (V), Keesara (M), Medchal.Dist.-501 301.

May-2025

TABLE OF CONTENTS

S.No.	Contents	Page No
	Abstract	**1**
1	**Introduction**	**2**
2	**Literature Review**	**3 - 4**
3	**System Design**	**5**
4	**Implementation**	
	4.1 Code	**6 - 47**
	4.2 Output Screens	**45 - 48**
5	**Conclusion**	**49**
6	**Bibilography**	**50**

ABSTRACT

Tourism plays a crucial role in the economic development of nations, and accurate forecasting of tourist arrivals is essential for effective planning of infrastructure, services, and promotional strategies. This project presents a **Tourist Prediction and Analysis Web Application** designed to forecast international tourist inflow to various countries using historical data and machine learning techniques, specifically the Linear Regression model. The system enables users to select a country and year to predict expected tourist arrivals for the period 2021–2025.

To enhance decision-making, the application provides interactive and informative visualizations including line charts, choropleth maps, and growth comparison tables, offering insights into top source countries, region-wise behavior, and emerging travel trends. The web platform is built using Flask, HTML/CSS, JavaScript, and Chart.js, ensuring a responsive, user-friendly interface.

Designed to support tourism departments, policymakers, and researchers, this tool seamlessly integrates predictive analytics with modern web technologies to deliver a powerful solution for analyzing and anticipating tourism patterns.

1. INTRODUCTION

Tourism is a pivotal sector in the global economy, rapidly expanding and significantly contributing to GDP and employment across various countries. Accurate forecasting of tourist arrivals is essential for governments, tourism departments, businesses, and planners, as it enables better preparation of infrastructure, optimization of services, and effective implementation of marketing strategies. The project titled **"Tourist Prediction and Analysis "** focuses on forecasting tourist inflows through the analysis of historical data, presented via an interactive web-based dashboard.

The primary objective of this project is to develop an application that predicts the number of tourist arrivals for specific countries and years using machine learning techniques. For this purpose, the **Linear Regression model**, a well-known and interpretable supervised learning algorithm, was employed, ideal for forecasting continuous numeric data. The model was trained on extensive tourism datasets spanning multiple years and countries, enabling it to identify patterns and trends in visitor counts.

The application was developed using Flask for the backend, while the frontend was constructed with HTML, CSS, and JavaScript, complemented by visualization frameworks such as Chart.js and Plotly. Users can interact with the system by selecting a country and a target year (from 2021 to 2025), after which the model predicts the expected number of tourist arrivals. The application also includes insightful data visualizations, such as:

1. **Line charts** displaying year-wise arrival trends for the selected country.
2. **Choropleth** world maps highlighting countries based on predicted arrivals.
3. A list of the **top five countries** anticipating the highest growth in tourist numbers.
4. **Regional analyses** examining continental tourist movement trends.

Additionally, the platform presents tables showing the primary tourist source countries for selected destinations. This comprehensive analytical view supports decision-makers in identifying key tourism markets and uncovering potential opportunities for international travel campaigns.

The platform is designed to ensure a smooth and responsive user experience, featuring a clean, modern interface suitable for both professional and public use. The combination of machine learning with interactive data visualization not only enhances the application's utility as a prediction tool but also establishes it as a valuable decision support system.

2. Literature Review

1. **Time Series Models in Tourism Forecasting**
   Traditional statistical approaches, such as the AutoRegressive Integrated Moving Average (ARIMA) and Exponential Smoothing (ETS) models, have been frequently employed to forecast tourist arrivals. These techniques are adept at capturing linear trends and seasonal fluctuations in historical data. For instance, Song et al. (2010) highlighted the utility of ARIMA models for short-term demand forecasting in tourism, particularly when there is a wealth of reliable historical data available. However, these models often fall short when it comes to recognizing complex non-linear interactions or adapting to external shocks like economic downturns or global pandemics.

   Source: *Song, H., Witt, S. F., & Li, G. (2010). The advanced econometrics of tourism demand. Routledge.*

2. **Machine Learning Approaches**
   In recent years, machine learning (ML) techniques have emerged as powerful tools for tourism forecasting due to their ability to model non-linear relationships. Techniques such as Linear Regression, Support Vector Regression (SVR), Random Forest, and Neural Networks have been applied to predict tourism demand with varying degrees of success (Law et al., 2019). Linear Regression, for example, strikes a balance between simplicity and interpretability, making it well-suited for integration into web-based applications. Unlike time series models, ML techniques can leverage a broader range of features, including socio-economic and environmental factors, which improves the overall accuracy of predictions.

   Source: *Law, R., Fuchs, M., & Ricci, F. (2019). Advances in travel and tourism forecasting: Introducing the Journal of Travel Research special issue. Journal of Travel Research, 58(8), 1227-1231.*

3. **Web-Based Tourism Analytics**
   Numerous interactive platforms and dashboards have been created to help stakeholders visualize tourism trends. These tools utilize data visualization libraries such as D3.js, Chart.js, and Python's Plotly, often in conjunction with backend technologies like Flask and Django (Khan & Rahman, 2021). While these applications excel at data representation, few integrate real-time prediction models, which limits their effectiveness in aiding future planning.

   Source: *Khan, S. M., & Rahman, M. M. (2021). Web-based data visualization for tourism analytics using open-source tools. International Journal of Information Management Data Insights, 1(2), 100016.*

4. **Geospatial Analysis and Regional Trends**
   The use of geospatial tools, such as Choropleth maps and GeoJSON data visualization, provides critical insights into regional patterns and trends within tourism. Research has shown that incorporating spatial elements into tourism forecasting enhances the contextual relevance of the data (Gunter & Önder, 2016). This method is particularly beneficial for analyzing regional predictions and comparing tourism growth on a country level.

   Source: *Gunter, U., & Önder, I. (2016). Forecasting international city tourism demand for Paris: Accuracy of uni- and multivariate models employing monthly data. Tourism Management, 53, 1–11.*

5. Challenges and Gaps Identified

Despite significant advancements, the field of tourism forecasting still grapples with various challenges, including data sparsity, sudden global disruptions (such as the COVID-19 pandemic), and inconsistent reporting standards across different countries. The literature underscores the necessity for models that are both adaptable and interpretable, as well as systems that combine predictive capabilities with user-friendly interfaces (Petropoulos et al., 2020).

Source: *Petropoulos, F., Makridakis, S., & Spiliotis, E. (2020). Forecasting tourism: The accuracy of alternative forecasting methods and the role of data aggregation. Annals of Tourism Research, 80, 102737.* This overview highlights the evolution of tourism forecasting methodologies and the ongoing quest for more accurate, adaptable, and user-friendly forecasting systems in an ever-changing global landscape.

3. SYSTEM REQUIREMENTS

1. Hardware Requirements:

- **Processor:** Intel i5 or above (or equivalent)
- **RAM:** 4 GB minimum (8 GB recommended)
- **Hard Disk:** Minimum 500 MB of free space
- **Display:** 1024x768 resolution or higher

2. Software Requirements:

- **Operating System:** Windows 10/11, Linux, or macOS
- **Programming Language:** Python 3.x
- **Frameworks & Libraries:**
    - Flask (for web development)
    - Pandas and NumPy (for data processing)
    - Matplotlib / Chart.js (for data visualization)
    - Scikit-learn (for ML modeling)
    - Web Technologies: HTML5, CSS3, JavaScript
- **IDE:** VS Code / Jupyter Notebook / PyCharm
- **Browser:** Google Chrome / Firefox

4. IMPLEMENTATION

4.1 CODE

app.py

```
from flask import Flask, render_template, request
import pandas as pd
import joblib


app = Flask(__name__)


# Load model and data
model = joblib.load("best_model.pkl")
df = pd.read_csv("cleaned_data.csv")


# Filter out unwanted region rows
df = df[~df["Region"].isin(["Grand Total", "Not Classified elsewhere"])]


# Get updated unique values
countries = sorted(df["Country of Nationality"].unique())
regions = sorted([r for r in df["Region"].unique() if r not in ["Grand Total", "Not Classified Elsewhere"]])




import pycountry
import plotly.express as px


def get_country_code(name):
    try:
        return pycountry.countries.lookup(name).alpha_3
```

```
    except:
        return None


@app.route("/")
def index():
    return render_template("index.html")



@app.route("/prediction", methods=["GET", "POST"])
def prediction():
    prediction = None
    selected_country = ""
    selected_year = ""
    tourist_sources = []

    if request.method == "POST":
        selected_country = request.form["country"]
        selected_year = int(request.form["year"])

        country_row = df[df["Country of Nationality"] == selected_country]
        if not country_row.empty and 2021 <= selected_year <= 2025:
            arrivals = [
                country_row["Number of Arrivals-2017"].values[0],
                country_row["Number of Arrivals-2018"].values[0],
                country_row["Number of Arrivals-2019"].values[0],
                country_row["Number of Arrivals-2020"].values[0]
            ]
            current_arrivals = arrivals.copy()
```

```
            for y in range(2021, selected_year + 1):
                pred = model.predict([current_arrivals[-4:]])[0]
                current_arrivals.append(pred)
            prediction = int(current_arrivals[-1])
        else:
            prediction = "Data not found or year out of range."

        # Get tourist source info (to India from selected country)
        tourist_sources = df[df["Region"] == selected_country][["Country of Nationality", "Number of Arrivals-2021"]] \
    .sort_values(by="Number of Arrivals-2021", ascending=False).head(10).values.tolist()

    return render_template("prediction.html",
                           prediction=prediction,
                           countries=countries,
                           selected_country = selected_country,
                           selected_year=selected_year,
                           tourist_sources=tourist_sources)

@app.route("/trend", methods=["GET", "POST"])
def trend():
    single_prediction = None
    selected_country = None

    compare_predictions = []
    selected_compare = []

    years = [2017, 2018, 2019, 2020, 2021]
```

```
pred_years = [2022, 2023, 2024, 2025]
all_years = years + pred_years

if request.method == "POST":
    if "country" in request.form:
        selected_country = request.form["country"]
        country_row = df[df["Country of Nationality"] == selected_country]

        if not country_row.empty:
            values = [int(country_row[f"Number of Arrivals-{y}"].values[0]) for y in years]
            arrivals = values[-4:]
            for _ in pred_years:
                pred = model.predict([arrivals[-4:]])[0]
                arrivals.append(pred)
            values += arrivals[4:]
            single_prediction = {
                "country": selected_country,
                "years": all_years,
                "values": [int(v) for v in values],
                "zipped": list(zip(all_years, [int(v) for v in values]))
            }

    elif "compare1" in request.form and "compare2" in request.form:
        selected_compare = [request.form["compare1"], request.form["compare2"]]

        for c in selected_compare:
            row = df[df["Country of Nationality"] == c]
            if not row.empty:
```

```
            values = [int(row[f"Number of Arrivals-{y}"].values[0]) for y in years]
            arrivals = values[-4:]
            for _ in pred_years:
                pred = model.predict([arrivals[-4:]])[0]
                arrivals.append(pred)
            values += arrivals[4:]
            compare_predictions.append({
                "country": c,
                "years": all_years,
                "values": [int(v) for v in values],
                "zipped": list(zip(all_years, [int(v) for v in values]))
            })

    return render_template("trend.html",
                           countries=countries,
                           prediction=single_prediction,
                           selected_country=selected_country,
                           compare_predictions=compare_predictions,
                           selected_compare=selected_compare)


@app.route("/growth")
def growth():
    growth_data = []

    for country in countries:
        row = df[df["Country of Nationality"] == country]
        if not row.empty:
```

```
        try:
            # Start from actual 2018–2021
            arrivals = [
                row["Number of Arrivals-2018"].values[0],
                row["Number of Arrivals-2019"].values[0],
                row["Number of Arrivals-2020"].values[0],
                row["Number of Arrivals-2021"].values[0]
            ]

            # Predict 2022–2025
            predicted = arrivals.copy()
            for _ in range(2022, 2026):  # Predict till 2025
                next_pred = model.predict([predicted[-4:]])[0]
                predicted.append(next_pred)

            growth = predicted[-1] - arrivals[-1]  # 2025 - 2021
            growth_data.append({
                "country": country,
                "from_2021": int(arrivals[-1]),
                "to_2025": int(predicted[-1]),
                "growth": int(growth)
            })

        except Exception as e:
            continue  # Skip any problematic rows

    # Sort by growth descending
    top5 = sorted(growth_data, key=lambda x: x["growth"], reverse=True)[:5]
```

```
    return render_template("growth.html", top5=top5)


@app.route("/region", methods=["GET", "POST"])
def region():
    selected_year = None
    region_data = {}

    if request.method == "POST":
        selected_year = int(request.form["year"])
        if selected_year < 2022 or selected_year > 2025:
            return render_template("region.html", error="Please select a year between 2022 and 2025.",
                                   regions=regions, selected_year=selected_year, region_data=None)

        for country in countries:
            row = df[df["Country of Nationality"] == country]
            if not row.empty:
                try:
                    region = row["Region"].values[0]

                    if region in ["Grand Total", "Not Classified Elsewhere"]:
                        continue  # skip unwanted

                    arrivals = [
                        row["Number of Arrivals-2018"].values[0],
                        row["Number of Arrivals-2019"].values[0],
                        row["Number of Arrivals-2020"].values[0],
                        row["Number of Arrivals-2021"].values[0],
```

```
                ]
                current_arrivals = arrivals.copy()

                for y in range(2022, selected_year + 1):
                    pred = model.predict([current_arrivals[-4:]])[0]
                    current_arrivals.append(pred)

                final_prediction = current_arrivals[-1]

                if region not in region_data:
                    region_data[region] = 0
                region_data[region] += final_prediction

            except Exception:
                continue

    return render_template("region.html",
                           selected_year=selected_year,
                           regions=sorted(region_data.keys()),
                           region_data=region_data)

@app.route("/worldmap")
def worldmap():
    data = []

    for country in countries:
        row = df[df["Country of Nationality"] == country]
        if not row.empty:
```

```
        try:
            # Get arrivals from 2018–2021
            arrivals = [
                row["Number of Arrivals-2018"].values[0],
                row["Number of Arrivals-2019"].values[0],
                row["Number of Arrivals-2020"].values[0],
                row["Number of Arrivals-2021"].values[0]
            ]

            # Predict for 2022–2025
            for _ in range(2022, 2026):
                pred = model.predict([arrivals[-4:]])[0]
                arrivals.append(pred)

            predicted_2025 = arrivals[-1]

            iso_code = get_country_code(country)
            if iso_code:
                data.append({
                    "country": country,
                    "iso_code": iso_code,
                    "tourists": int(predicted_2025)
                })

        except Exception:
            continue

# Create DataFrame for Plotly
```

```
    df_map = pd.DataFrame(data)

    fig = px.choropleth(df_map,
                        locations="iso_code",
                        color="tourists",
                        hover_name="country",
                        color_continuous_scale="Blues",
                        projection="natural earth",
                        title="")

    map_html = fig.to_html(full_html=False)

    return render_template("worldmap.html", map_html=map_html)

if __name__ == "__main__":
    app.run(debug=True, use_reloader=False)
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Tourism Dashboard</title>
  <style>
    body {
      margin: 0;
      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
      background-image: url("{{ url_for('static', filename='tourism_bg.png') }}");
      background-size: 100%,100%;
      background-repeat: no-repeat;
```

```
  background-position: top center;
  height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  box-sizing: border-box;
}

h1 {
  margin-top: 60px;
  font-size: 72px;
  color: #004d40;
  text-align: center;
}

.card-container {
  margin-top: auto;
  margin-bottom: 100px;
  display: flex;
  gap: 30px;
  flex-wrap: wrap;
  justify-content: center;
}

.card {
background: rgba(255, 255, 255, 0.3);
backdrop-filter: blur(8px);
-webkit-backdrop-filter: blur(8px);
border: 1px solid rgba(255, 255, 255, 0.4);
border-radius: 15px;
padding: 30px 20px;
```

```
  text-align: center;
  box-shadow: 0 8px 20px rgba(0, 0, 0, 0.15);
  transition: transform 0.3s ease, box-shadow 0.3s ease;
  cursor: pointer;
  text-decoration: none;
  color: #004d40;
  font-weight: bold;
  font-size: 16px;
}



   .card:hover {
     transform: translateY(-8px);
     box-shadow: 0 12px 25px rgba(0, 0, 0, 0.2);
   }

   .icon {
     font-size: 32px;
     margin-bottom: 15px;
     display: block;
   }

   @media (max-width: 600px) {
     h1 {
       font-size: 32px;
       margin-top: 80px;
     }
     .card-container {
       flex-direction: column;
       align-items: center;
       margin-bottom: 60px;
```

```
    }
  }
 </style>
</head>
<body>
 <h1>Tourism</h1>
 <div class="card-container">
  <a href="/prediction" class="card">
   <span class="icon">🔍</span>
   Predict Tourist Arrivals
  </a>
  <a href="/trend" class="card">
   <span class="icon">📈</span>
   View Tourist Trends
  </a>
  <a href="/growth" class="card">
   <span class="icon">🌍</span>
   Top 5 Growing Countries
  </a>
  <a href="/region" class="card">
   <span class="icon">📊</span>
   Predict Region-wise
  </a>
  <a href="{{ url_for('worldmap') }}" class="card">
   <span class="icon">🗺️</span>
   World Map View
  </a>
 </div>
</body>
</html>
```

Prediction.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Tourist Prediction</title>
  <link href="https://fonts.googleapis.com/css2?family=Poppins&display=swap" rel="stylesheet">
  <style>
    body {
      font-family: 'Poppins', sans-serif;
      background: linear-gradient(to bottom right, #cceeff, #e6f7ff);
      padding: 40px;
      margin: 0;
    }

    .container {
      background: rgba(255, 255, 255, 0.95);
      padding: 30px 40px;
      max-width: 600px;
      margin: auto;
      border-radius: 15px;
      box-shadow: 0 10px 30px rgba(0, 0, 0, 0.15);
    }

    h2 {
      text-align: center;
      margin-bottom: 30px;
      color: #004d40;
    }

    label {
      margin-top: 15px;
```

```
  font-weight: 500;
}

select, button {
  width: 100%;
  padding: 12px;
  margin-top: 8px;
  margin-bottom: 20px;
  border: 1px solid #ccc;
  border-radius: 8px;
  font-size: 15px;
}

button {
  background-color: #009688;
  color: white;
  font-weight: bold;
  border: none;
  cursor: pointer;
  transition: background-color 0.3s ease;
}

button:hover {
  background-color: #00796b;
}

.result {
  text-align: center;
  margin-top: 25px;
  font-weight: bold;
  font-size: 18px;
```

```
    color: #333;
  }

  table {
    width: 100%;
    margin-top: 25px;
    border-collapse: collapse;
    background: #ffffff;
    border-radius: 10px;
    overflow: hidden;
    box-shadow: 0 4px 12px rgba(0, 0, 0, 0.05);
  }

  th, td {
    padding: 12px 16px;
    text-align: left;
    border-bottom: 1px solid #e0e0e0;
  }

  th {
    background-color: #e0f7fa;
    font-weight: 600;
  }

  h3 {
    margin-top: 40px;
    text-align: center;
    color: #00695c;
  }
 </style>
</head>
```

```
<body>
  <div class="container">
    <h2>Tourist Count Prediction</h2>
    <form method="POST">
      <label for="country">Select Country:</label>
      <select name="country" required>
        {% for country in countries %}
          <option value="{{ country }}" {% if country == selected_country %}selected{% endif %}>{{ country }}</option>
        {% endfor %}
      </select>

      <label for="year">Select Year:</label>
      <select name="year" required>
        {% for year in range(2021, 2026) %}
          <option value="{{ year }}" {% if year == selected_year %}selected{% endif %}>{{ year }}</option>
        {% endfor %}
      </select>

      <button type="submit">Predict</button>
    </form>

    {% if prediction is not none %}
      <div class="result">
        {% if prediction is string %}
          {{ prediction }}
        {% else %}
          Predicted Tourist Arrivals in {{ selected_year }}: {{ prediction }}
        {% endif %}
      </div>
    {% endif %}
```

```
    {% if tourist_sources %}
      <h3>Top Tourist Source Countries (2021) for {{ selected_country }}</h3>
      <table>
        <tr><th>Country</th><th>Tourist Arrivals</th></tr>
        {% for row in tourist_sources.itertuples() %}
          <tr><td>{{ row[1] }}</td><td>{{ row[2] }}</td></tr>
        {% endfor %}
      </table>
    {% endif %}
  </div>
</body>
</html>
```

Trend.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Tourism Trends</title>
  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
  <style>
    body {
      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
      background: #e0f7ff;
      color: #333;
      margin: 0;
      padding: 0;
    }
```

```
h1, h2, h3 {
  text-align: center;
}

/* Page Heading */
.page-header {
  padding: 60px 20px 20px;
}

.page-header h1 {
  font-size: 36px;
  margin-bottom: 10px;
  color: #333;
}

.page-header p {
  font-size: 18px;
  color: #666;
}

/* Cards */
#trend-choice .card {
  cursor: pointer;
  padding: 30px 25px;
  border-radius: 14px;
  color: white;
  flex: 1;
  text-align: center;
```

```
    transition: transform 0.3s ease, box-shadow 0.3s ease;
    box-shadow: 0 8px 20px rgba(0, 0, 0, 0.15);
    min-width: 280px;
    max-width: 350px;
  }

  #trend-choice .card:hover {
    transform: translateY(-6px);
    box-shadow: 0 12px 24px rgba(0, 0, 0, 0.2);
  }

  .green-card {
  background: linear-gradient(135deg, #43A047, #66BB6A);
  box-shadow: 0 8px 16px rgba(67, 160, 71, 0.3);
  color: #fff;
  font-size: 18px;
  font-weight: 600;
  text-shadow: 1px 1px 2px rgba(0,0,0,0.2);
  transition: background 0.4s ease, box-shadow 0.4s ease, transform 0.3s ease;
}

.green-card:hover {
  background: linear-gradient(135deg, #388E3C, #4CAF50);
  text-shadow: 1px 1px 5px rgba(0,0,0,0.4);
}

.blue-card {
  background: linear-gradient(135deg, #1E88E5, #42A5F5);
```

```
  box-shadow: 0 8px 16px rgba(30, 136, 229, 0.3);
  color: #fff;
  font-size: 18px;
  font-weight: 600;
  text-shadow: 1px 1px 2px rgba(0,0,0,0.2);
  transition: background 0.4s ease, box-shadow 0.4s ease, transform 0.3s ease;
}

.blue-card:hover {
  background: linear-gradient(135deg, #1565C0, #2196F3);
  text-shadow: 1px 1px 5px rgba(0,0,0,0.4);
}



   #trend-choice > div {
     display: flex;
     gap: 30px;
     justify-content: center;
     margin: 40px 20px 60px;
     flex-wrap: wrap;
   }

   /* Forms */
   form {
     max-width: 600px;
     margin: 20px auto;
     background: #fff;
     padding: 25px;
```

```
  border-radius: 10px;
  box-shadow: 0 4px 12px rgba(0,0,0,0.1);
}

label {
  display: block;
  margin-bottom: 10px;
  font-weight: bold;
}

select, .form-control, button {
  width: 100%;
  padding: 10px;
  margin-bottom: 20px;
  font-size: 16px;
  border: 1px solid #ccc;
  border-radius: 8px;
}

button {
  background-color: #009688;
  color: white;
  border: none;
  transition: background 0.3s ease;
  cursor: pointer;
}

button:hover {
```

```
  background-color: #00796b;
}

/* Charts */
.chart-container {
  max-width: 1000px;
  height: 400px;
  margin: 40px auto;
  background: white;
  border-radius: 10px;
  box-shadow: 0 4px 12px rgba(0,0,0,0.08);
  padding: 10px;
}

canvas {
  width: 100% !important;
  height: 100% !important;
}

@media screen and (max-width: 768px) {
  #trend-choice > div {
    flex-direction: column;
    align-items: center;
  }

  #trend-choice .card {
    width: 90%;
    margin-bottom: 20px;
```

```
    }
  }
 </style>
</head>
<body>

<!-- Page Heading -->
<div class="page-header">
 <h1>📊 Tourist Trend Analysis</h1>
</div>

<!-- Card Choices -->
<div id="trend-choice">
 <div>
  <div onclick="showForm('single')" class="card green-card" role="button" tabindex="0">
   <h3>📈 <strong>Country Trend</strong></h3>
   <p>View trend of a single country over the years.</p>
  </div>

  <div onclick="showForm('compare')" class="card blue-card" role="button" tabindex="0">
   <h3>🔀 <strong>Compare Two Countries</strong></h3>
   <p>Compare tourist trends between two countries.</p>
  </div>
 </div>
</div>

<!-- Country Trend Form -->
<div id="single-form" style="display:none;">
```

```
<h2>Tourist Arrival Trend (2017–2025)</h2>
<form method="POST">
<label for="country">Select Country:</label>
<select name="country" class="form-control" required>
{% for country in countries %}
<option value="{{ country }}" {% if selected_country == country %}selected{% endif %}>{{ country }}</option>
{% endfor %}
</select>
<button type="submit">Show Trend</button>
</form>

{% if prediction %}
<h3>Trend for {{ prediction.country }}</h3>
<div class="chart-container">
<canvas id="trendChart"></canvas>
</div>
<script>
const trendCtx = document.getElementById('trendChart').getContext('2d');
const trendYears = {{ prediction["years"] | tojson }};
const trendValues = {{ prediction["values"] | tojson }};

new Chart(trendCtx, {
type: 'line',
data: {
labels: trendYears,
datasets: [{
label: 'Tourist Arrivals',
data: trendValues,
```

```
    borderColor: 'rgba(0, 150, 136, 1)',
    backgroundColor: 'rgba(0, 150, 136, 0.2)',
    fill: true,
    tension: 0.4,
    pointBackgroundColor: '#004d40',
    pointRadius: 4
  }]
},
options: {
  responsive: true,
  maintainAspectRatio: true,
  plugins: {
    legend: {
      display: true,
      position: 'top',
      labels: {
        color: '#004d40',
        font: { weight: 'bold' }
      }
    }
  },
  scales: {
    y: {
      beginAtZero: true,
      ticks: { stepSize: 1000 },
      grid: { color: '#e0f2f1' }
    },
    x: {
```

```
            grid: { color: '#e0f2f1' }
          }
        }
      }
    });
  </script>
 {% endif %}
</div>

<!-- Compare Countries Form -->
<div id="compare-form" style="display:none;">
 <h2>Compare Two Countries</h2>
 <form method="POST">
  <label for="compare1">Select First Country:</label>
  <select name="compare1" required>
   {% for country in countries %}
    <option value="{{ country }}" {% if selected_compare[0] == country %}selected{% endif %}>{{ country }}</option>
   {% endfor %}
  </select>

  <label for="compare2">Select Second Country:</label>
  <select name="compare2" required>
   {% for country in countries %}
    <option value="{{ country }}" {% if selected_compare[1] == country %}selected{% endif %}>{{ country }}</option>
   {% endfor %}
  </select>
```

```
    <button type="submit">Compare Trends</button>
  </form>

  {% if compare_predictions %}
    <h3>Comparison: {{ compare_predictions[0].country }} vs {{ compare_predictions[1].country }}</h3>
    <div class="chart-container">
      <canvas id="compareChart"></canvas>
    </div>
    <script>
      const compareCtx = document.getElementById('compareChart').getContext('2d');
      const compareData = {
        labels: {{ compare_predictions[0].years | tojson }},
        datasets: [
          {
            label: '{{ compare_predictions[0].country }}',
            data: {{ compare_predictions[0]["values"] | tojson }},
            borderColor: 'rgba(255, 87, 34, 1)',
            backgroundColor: 'rgba(255, 87, 34, 0.2)',
            fill: true,
            tension: 0.4,
            pointRadius: 4
          },
          {
            label: '{{ compare_predictions[1].country }}',
            data: {{ compare_predictions[1]["values"] | tojson }},
            borderColor: 'rgba(33, 150, 243, 1)',
            backgroundColor: 'rgba(33, 150, 243, 0.2)',
            fill: true,
```

```
      tension: 0.4,
      pointRadius: 4
    }
  ]
};

new Chart(compareCtx, {
  type: 'line',
  data: compareData,
  options: {
    responsive: true,
    maintainAspectRatio: true,
    plugins: {
      legend: {
        display: true,
        position: 'top',
        labels: {
          font: { weight: 'bold' }
        }
      }
    },
    scales: {
      y: {
        beginAtZero: true,
        grid: { color: '#e0f2f1' }
      },
      x: {
        grid: { color: '#e0f2f1' }
```

```
          }
        }
      }
    });
  </script>
 {% endif %}
</div>
<!-- JS Logic -->
<script>
 function showForm(type) {
  document.getElementById("trend-choice").style.display = "none";
  document.getElementById("single-form").style.display = (type === "single") ? "block" : "none";
  document.getElementById("compare-form").style.display = (type === "compare") ? "block" : "none";
 }

 window.onload = function () {
  {% if prediction %}
   showForm('single');
  {% elif compare_predictions %}
   showForm('compare');
  {% endif %}
 };
</script>

</body>
</html>
```

Growth.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Top 5 Countries by Growth</title>
  <link href="https://fonts.googleapis.com/css2?family=Poppins&display=swap" rel="stylesheet">
  <style>
    body {
      font-family: 'Poppins', sans-serif;
      background: linear-gradient(to right, #e0f7fa, #f1f8e9);
      padding: 40px;
      margin: 0;
    }

    .container {
      max-width: 750px;
      margin: auto;
      background: rgba(255, 255, 255, 0.95);
      padding: 30px;
      border-radius: 15px;
      box-shadow: 0 8px 20px rgba(0, 0, 0, 0.1);
    }

    h2 {
      text-align: center;
      color: #004d40;
      margin-bottom: 30px;
    }
```

```
  table {
    width: 100%;
    border-collapse: collapse;
    border-radius: 10px;
    overflow: hidden;
    box-shadow: 0 4px 12px rgba(0, 0, 0, 0.05);
  }

  th, td {
    padding: 14px;
    text-align: center;
    border-bottom: 1px solid #e0e0e0;
    font-size: 15px;
  }

  th {
    background-color: #00796b;
    color: white;
  }

  tr:nth-child(even) {
    background-color: #f9f9f9;
  }

  tr:hover {
    background-color: #e0f2f1;
  }
</style>
```

```
</head>
<body>
  <div class="container">
    <h2>Top 5 Countries by Tourist Growth (2021 → 2025)</h2>
    <table>
      <thead>
        <tr>
          <th>Rank</th>
          <th>Country</th>
          <th>2021 Arrivals</th>
          <th>2025 Prediction</th>
          <th>Growth</th>
        </tr>
      </thead>
      <tbody>
        {% for entry in top5 %}
        <tr>
          <td>{{ loop.index }}</td>
          <td>{{ entry.country }}</td>
          <td>{{ entry.from_2021 }}</td>
          <td>{{ entry.to_2025 }}</td>
          <td>{{ entry.growth }}</td>
        </tr>
        {% endfor %}
      </tbody>
    </table>
  </div>
</body>
```

```
</html>
```

Region.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Region-wise Tourist Predictions</title>
  <link href="https://fonts.googleapis.com/css2?family=Poppins&display=swap" rel="stylesheet">
  <style>
    body {
      font-family: 'Poppins', sans-serif;
      background: linear-gradient(to right, #e1f5fe, #e8f5e9);
      padding: 40px;
      margin: 0;
    }

    .container {
      max-width: 700px;
      margin: auto;
      background: rgba(255, 255, 255, 0.95);
      padding: 30px;
      border-radius: 15px;
      box-shadow: 0 8px 20px rgba(0, 0, 0, 0.1);
    }

    h2 {
      text-align: center;
      color: #004d40;
      margin-bottom: 30px;
```

```
}

form {
  text-align: center;
  margin-bottom: 30px;
}

select, button {
  padding: 10px 15px;
  font-size: 16px;
  margin: 0 5px;
  border-radius: 8px;
  border: 1px solid #ccc;
  outline: none;
}

button {
  background-color: #00796b;
  color: white;
  border: none;
  cursor: pointer;
  transition: background 0.3s ease;
}

button:hover {
  background-color: #004d40;
}
```

```
table {
  width: 100%;
  border-collapse: collapse;
  margin-top: 20px;
  box-shadow: 0 4px 12px rgba(0, 0, 0, 0.05);
  border-radius: 10px;
  overflow: hidden;
}

th, td {
  padding: 14px;
  text-align: center;
  border-bottom: 1px solid #e0e0e0;
  font-size: 15px;
}

th {
  background-color: #00796b;
  color: white;
}

tr:nth-child(even) {
  background-color: #f9f9f9;
}

tr:hover {
  background-color: #e0f2f1;
}
```

```
    .error {
      color: red;
      text-align: center;
      font-weight: bold;
      margin-top: 20px;
    }
  </style>
</head>
<body>
  <div class="container">
    <h2>Region-wise Tourist Predictions</h2>
    <form method="POST">
      <label for="year">Select Year (2022–2025): </label>
      <select name="year" id="year">
        {% for y in range(2022, 2026) %}
          <option value="{{ y }}" {% if selected_year == y %}selected{% endif %}>{{ y }}</option>
        {% endfor %}
      </select>
      <button type="submit">Predict</button>
    </form>

    {% if region_data %}
      <table>
        <thead>
          <tr>
            <th>Region</th>
            <th>Predicted Tourists ({{ selected_year }})</th>
```

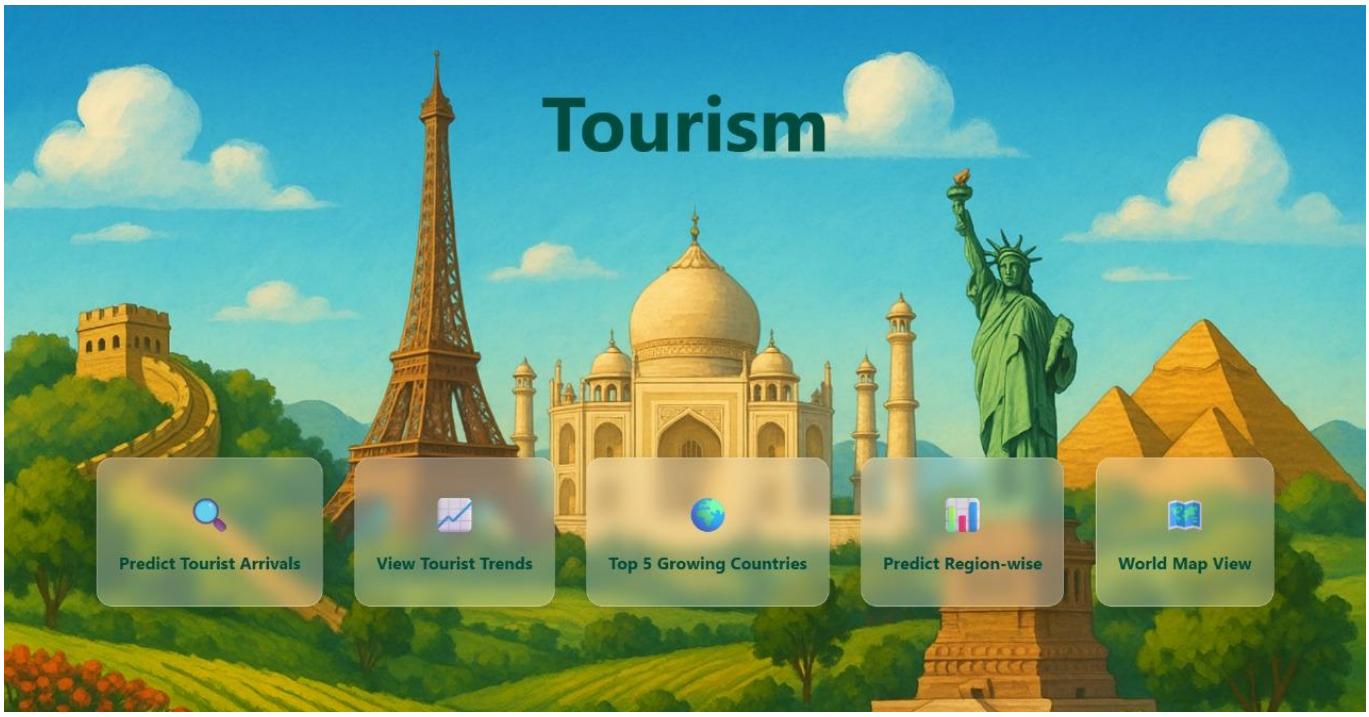
```
        </tr>
      </thead>
      <tbody>
        {% for region, total in region_data.items() %}
          <tr>
            <td>{{ region }}</td>
            <td>{{ total|int }}</td>
          </tr>
        {% endfor %}
      </tbody>
    </table>
  {% elif selected_year %}
    <div class="error">{{ error }}</div>
  {% endif %}
 </div>
</body>
</html>
```

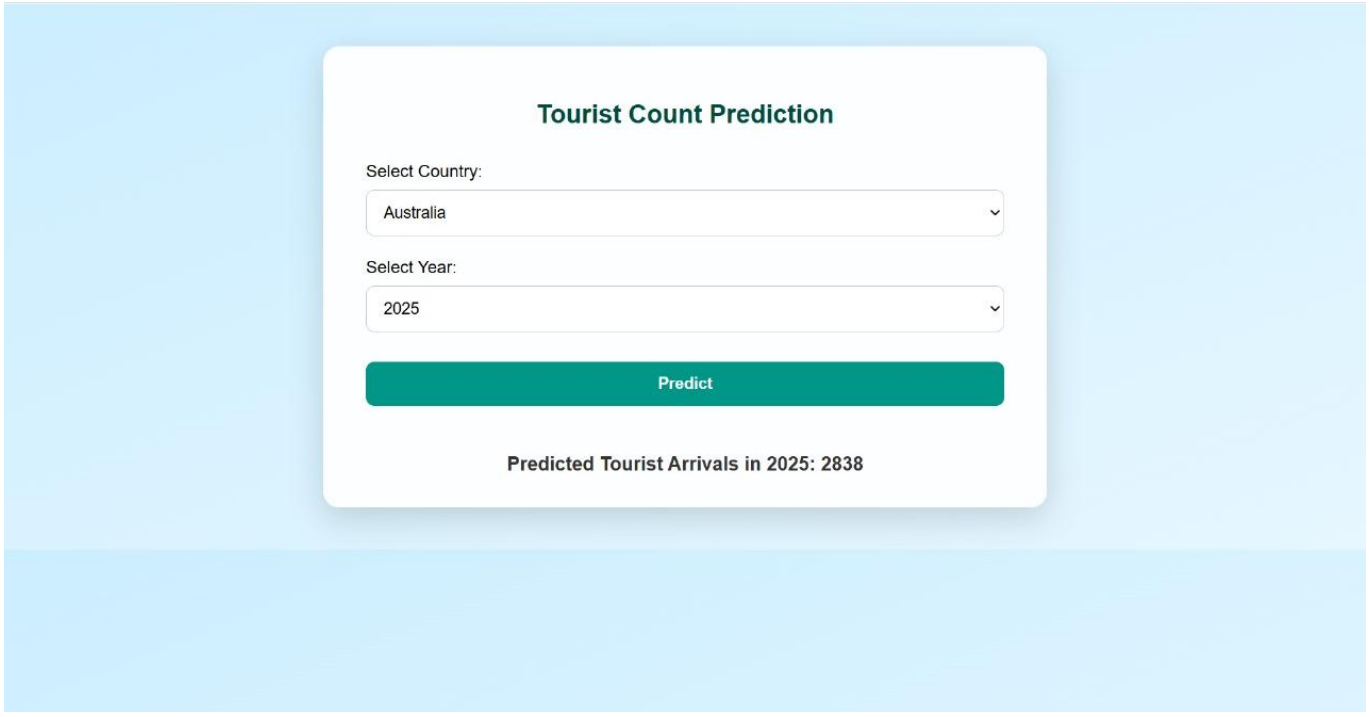
Worldmap.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>World Map - Tourist Predictions</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      padding: 0;
```

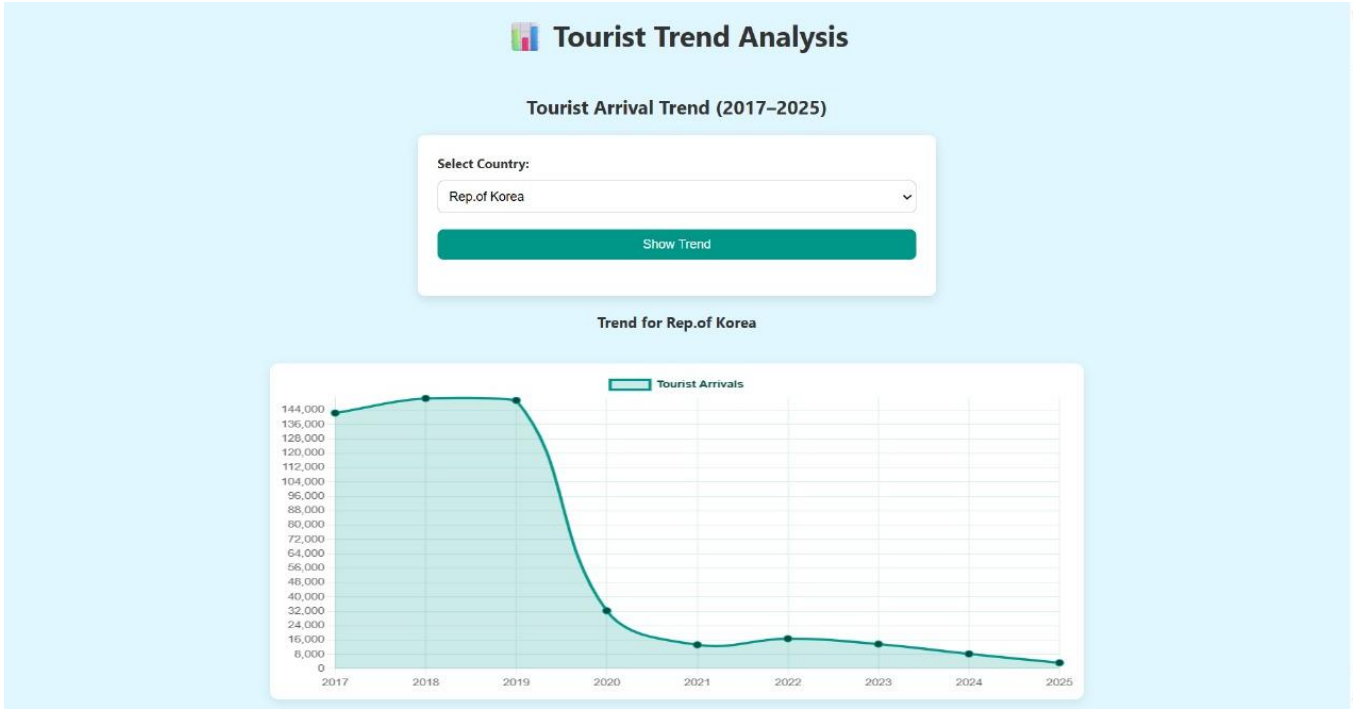
```
            background: linear-gradient(to bottom right, #e0f7fa, #b2ebf2);
            min-height: 100vh;
        }

        h2 {
            text-align: center;
            margin-top: 20px;
            color: #003344;
        }
        .map-container {
            width: 90%;
            margin: 30px auto;
            background: white;
            padding: 20px;
            border-radius: 12px;
            box-shadow: 0 0 20px rgba(0, 0, 0, 0.1);
        }
    </style>
</head>
<body>
    <h2>Predicted Tourist Arrivals (2025)</h2>
    <div class="map-container">
        {{ map_html|safe }}
    </div>
</body>
</html>
```

4.2 OUTPUT SCREENS






Tourist Trend Analysis
Country Trend
View trend of a single country over the years.
Compare Two Countries
Compare tourist trends between two countries.


Tourist Trend Analysis
Tourist Arrival Trend (2017–2025)
Select Country:
Rep.of Korea
Show Trend
Trend for Rep.of Korea
Tourist Arrivals
144,000
136,000
128,000
120,000
112,000
104,000
96,000
88,000
80,000
72,000
64,000
56,000
48,000
40,000
32,000
24,000
16,000
8,000
0
2017
2018
2019
2020
2021
2022
2023
2024
2025

Compare Two Countries

Select First Country:

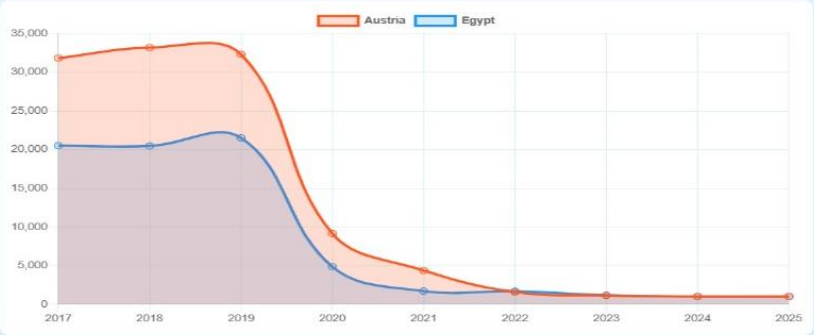
Austria

Select Second Country:

Egypt

Compare Trends

Comparison: Austria vs Egypt



Top 5 Countries by Tourist Growth (2021 → 2025)

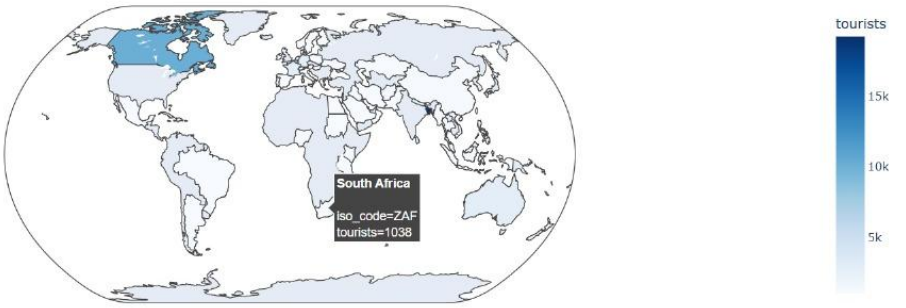
Rank	Country	2021 Arrivals	2025 Prediction	Growth
1	Argentina	530	1038	508
2	Hungary	653	1038	385
3	Mexico	859	1038	179
4	Czech Rep.	983	1038	55
5	Saudi Arabia	1088	1038	-49

Region-wise Tourist Predictions

Select Year (2022–2025): 2025 Predict

Region	Predicted Tourists (2025)
Eastern Europe	8176
South Asia	30914
Central and South America	3115
Australasia	3854
Western Europe	38377
West Asia	8515
North America	26922
East Asia	7463
Africa	7375
South East Asia	9639

Predicted Tourist Arrivals (2025)



5. CONCLUSION

The development of this tourism prediction system demonstrates the potential of machine learning techniques in forecasting and data analysis within the travel and tourism industry. By leveraging historical tourist arrival data and applying a supervised learning model—specifically, Linear Regression—the system is able to generate meaningful insights about future travel trends, regional growth, and country-wise tourist flow. This predictive capability is crucial for policymakers, tourism boards, and private stakeholders, enabling them to make informed decisions related to marketing, infrastructure development, and resource management.

The integration of interactive visualizations using tools like Chart.js enhances the interpretability of results, making the platform accessible not only to technical users but also to decision-makers with non-technical backgrounds. Additionally, the web-based interface built using HTML, CSS, Flask, and Python offers a seamless and user-friendly experience for interacting with the system's predictive functionalities.

This project also emphasizes the importance of data-driven decision-making in a post-pandemic world, where tourism trends are rapidly evolving. The ability to identify top-performing countries and regions in terms of tourist growth helps in directing promotional efforts and optimizing tourism policies. Moreover, the system supports regional-level forecasting, providing granular insights that are often overlooked in national-level analysis.

In summary, the project achieves its core objective of creating a reliable, interpretable, and accessible tourism forecasting platform. With further improvements such as integrating real-time data, experimenting with advanced ML models like Random Forest or ARIMA, and expanding the dataset, the system can be extended into a more robust tool for strategic planning in the tourism sector.

6. BIBLIOGRAPHY

1. Datasets & Open Data Sources

- UNWTO (United Nations World Tourism Organization) – https://www.unwto.org/statistics
- Our World in Data – Tourism Metrics- https://ourworldindata.org/tourism
- World Bank Open Data – International Tourism Arrivals- https://data.worldbank.org/indicator/ST.INT.ARVL
- Google Dataset Search- https://datasetsearch.research.google.com/

4. Research Articles & Academic Journals

- Song, H., & Li, G. (2008). Tourism Demand Modelling and Forecasting—A Review of Recent Research. Tourism Management, 29(2), 203–220. https://doi.org/10.1016/j.tourman.2007.07.016
- Papatheodorou, A. (2006). Tourism Demand Forecasting Using Time Series and Econometric Models. TTRA Annual Conference Proceedings. https://scholarworks.umass.edu/ttra/2006/Presentations/3/
- Zeng, B., & Gerritsen, R. (2014). What do we know about social media in tourism? A review. Tourism Management Perspectives, 10, 27–36. https://doi.org/10.1016/j.tmp.2014.01.001

3. Tools, Frameworks & Libraries

- Scikit-learn (ML Framework in Python) https://scikit-learn.org/stable/
- Flask (Python Web Framework) https://flask.palletsprojects.com
- Chart.js (Data Visualization Library) https://www.chartjs.org/docs/latest/

4. Blogs & Informative Pages

- Bhardwaj, A. (2021). Using Python and Machine Learning for Tourism Statista – Global Tourism Insights https://www.statista.com/topics/9625/tourism-worldwide/