International Scientific Journal of Engineering and Management (ISJEM)
Volume: 05 Issue: 04 | April – 2026
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata
ISSN: 2583-6129
DOI: 10.55041/ISJEM06888

Tourismai: An AI-Driven Tourism Demand Forecasting and Overcrowding Prediction Platform using Machine Learning

Devam Dilipbhai Patel
Dept. of Computer Engineering, PIT
Parul University, Vadodara, Gujarat.

Ms. Kiran Sharma
Faculty Mentor, Asst. Professor, PIT IT services
Parul University, Vadodara,Gujarat, Gujarat
devampatel2004@gmail.com

Ms. Twinkle Shah
Industry Mentor, Infolabz
Ahmedabad,

Abstract— India's tourism sector contributes nearly 9.2% to the national GDP and employs upward of 87 million people; yet the infrastructure surrounding demand prediction, crowd management, and travel planning remains fragmented and largely manual. This paper introduces TourismAI, a full-stack, data-driven web platform developed in Python and Streamlit that attacks this problem from three complementary directions. First, a Random Forest Regressor trained on a curated 20,000-record tourism dataset delivers visitor-count forecasts with an R^2 score of 0.87 and a Mean Absolute Error of approximately 1,820 visitors per destination-date query. Second, a hand-crafted Travel Intelligence Engine (TIE) translates seasonal and destination-specific domain knowledge into interpretable overcrowding risk scores — High, Medium, or Low — achieving 91% classification accuracy across 50 verified test cases. Third, a knowledge-base-driven AI chatbot handles travel guidance queries through a hybrid keyword-matching and dynamic data-aggregation strategy. Beyond these core intelligence modules, TourismAI incorporates a real-time Business Intelligence dashboard with five interactive Plotly charts, a role-based access system distinguishing Travel Agents from regular Users, an interactive Map Explorer, a Budget Trip Planner with day-wise itinerary generation, a Place Comparison Engine, and a high-risk Alerts module. The modular architecture is built for extensibility: swapping in a new dataset or adding a new destination to the intelligence engine requires changes in only one file. TourismAI demonstrates how a relatively small, well-curated dataset combined with thoughtful ML and rule-based design can produce a genuinely useful decision-support tool for tourism stakeholders across India.

Keywords — Tourism Demand Forecasting, Machine Learning, Random Forest Regressor, Overcrowding Prediction, Streamlit, Python, Business Intelligence, AI Chatbot, Seasonal Risk Classification, Smart Tourism, India Tourism, Data-Driven Planning, Scikit-learn, Plotly, Role-Based Access Control

I. INTRODUCTION

1.1 Background and Motivation

India welcomed roughly 1.49 billion domestic tourist visits in 2022–23, a figure that represents not just an economic milestone but also a serious operational challenge for destination managers, state tourism boards, and independent travellers alike [1]. Popular sites such as the Taj Mahal in Agra, the beaches of Goa, the backwaters of Kerala, and the mountain stations of Himachal Pradesh regularly exceed comfortable visitor thresholds during peak weeks, resulting in degraded visitor experience, environmental strain, and logistical bottlenecks. The UNWTO's 2019 global report on overtourism identified India as one of the emerging markets most at risk of unmanaged visitor concentration [2].

At the same time, ordinary travellers — particularly first-generation tourists from Tier-2 and Tier-3 cities — often make trip decisions with limited information. They rely on social media posts, word-of-mouth recommendations, and generic travel blogs, none of which provide data-backed estimates of crowd levels, cost breakdowns, or seasonal advisories. This information asymmetry leads to both overcrowded peak seasons and under-utilised shoulder seasons — an imbalance that harms both the tourist experience and the long-term sustainability of the destination [3].

Machine learning has demonstrated strong potential for demand forecasting across adjacent domains — retail, logistics, energy — and tourism is no exception. However, most academic work in tourism ML has been published in isolation from practical deployment. There is a clear gap between research prototypes that report impressive metrics on cleaned benchmark datasets and end-to-end systems that a travel agent or a curious traveller can actually use. TourismAI is designed to bridge that gap.

1.2 Problem Statement

Four concrete problems motivated the design of this system:

•**Demand unpredictability:** Visitor counts at Indian destinations fluctuate by 300–400% between peak and off-peak periods [1]. Static historical averages are inadequate for trip planning because they mask within-year volatility.

•**Overcrowding risk communication:** Even when aggregate demand data is available, it is rarely translated into actionable, human-readable risk signals for travellers.

•**Fragmented planning tools:** Budget calculators, destination guides, chatbots, and data dashboards exist as separate siloed products. No single platform integrates them with ML-driven intelligence.

•**Stakeholder disparity:** Travel agents need deep analytical tools; casual users need simple interfaces. Existing platforms rarely serve both audiences from a single codebase.

1.3 Objectives

TourismAI was built to achieve the following objectives:

•Train and deploy an ML regression model to predict visitor demand at Indian tourist destinations given a set of trip parameters.

•Design a transparent rule-based Travel Intelligence Engine to classify seasonal overcrowding risk for major destination states.

•Develop a full-stack Streamlit application with role-based dashboards for Travel Agents and general Users.

•Integrate a knowledge-base-driven chatbot for personalised, context-aware travel guidance without the cost and latency of cloud LLM inference.

•Validate all three intelligence components against a 20,000-record real-world dataset spanning 16 Indian states.

•Design for extensibility so that new regions, datasets, or intelligence rules can be added with minimal code changes.

1.4 Scope and Coverage

TourismAI covers 16 major Indian states spanning all six geographic zones: North (Rajasthan, Himachal Pradesh, Uttarakhand, Uttar Pradesh, Punjab), South (Kerala, Karnataka, Tamil Nadu, Telangana), East (West Bengal, Odisha, Assam), West (Goa, Gujarat, Maharashtra), and Northeast (Nagaland). The 20,000-record dataset spans four seasons (Winter, Summer, Monsoon, Festive), six weather types, and six tourist-type categories. The platform is built as a local Streamlit application and is cloud-deployment ready with a single configuration change.

II. LITERATURE REVIEW

2.1 Tourism Demand Forecasting: Historical Perspective

The academic field of tourism demand forecasting has evolved through three broad eras. The first era (1960s–1990s) was dominated by econometric methods — particularly Ordinary Least Squares regression and ARIMA time-series models. Song and Li (2008) reviewed 121 peer-reviewed forecasting studies published between 1995 and 2007 and found that error-correction models and structural VAR models outperformed naive baselines by 20–30% on multi-step horizons, but at the cost of extensive data requirements and expert calibration [4].

The second era (2000s–2015s) introduced Support Vector Machines, neural networks, and hybrid ARIMA-ANN models. Cho (2003) applied an Elman recurrent neural network to Hong Kong tourism arrivals and reported an MAE improvement of 14% over ARIMA [5]. The third and current era is defined by deep learning: LSTM and Transformer-based models have achieved state-of-the-art results on tourism time-series benchmarks, with some studies reporting R^2 values above 0.90 [6]. Random Forest — the ensemble method chosen for TourismAI — has been shown to be competitive with LSTM on datasets of fewer than 50,000 records while being dramatically easier to interpret and deploy [7].

ISSN: 2583-6129
DOI: 10.55041/ISJEM06888

2.2 Machine Learning in Indian Tourism

Research specifically targeting Indian tourism with ML methods is relatively sparse compared to East Asian or European contexts. Sharma and Gupta (2022) built a Random Forest classifier for destination popularity prediction across 18 Indian states using 18 socio-geographic features, achieving an R^2 of 0.81 on a 12,000-record dataset [8]. Their work validated the utility of tree-ensemble methods in this domain but did not extend to a deployable user-facing application. TourismAI extends this direction by adding seasonal risk classification, a chatbot, and a full web UI layer.

Patel et al. (2021) used gradient-boosted trees to predict hotel occupancy rates in Gujarat using booking platform data and reported an MAE of 6.2 occupancy-percentage points [9]. Their feature set included day-of-week, proximity to national holidays, and review scores — features that overlap significantly with TourismAI's feature engineering approach, providing indirect validation of the chosen predictors.

2.3 Overcrowding and Carrying Capacity

The concept of Tourist Carrying Capacity (TCC) defines the maximum number of visitors a destination can accommodate without unacceptable degradation of the physical environment or visitor experience [2]. Physical TCC (space-based) and Perceptual TCC (experience-based) have been studied extensively for UNESCO World Heritage Sites. Rule-based expert systems, which encode domain knowledge as conditional decision trees, have been proposed as a lightweight and transparent complement to ML-based predictions for risk communication to non-technical audiences [10]. TourismAI's Travel Intelligence Engine follows this paradigm.

2.4 Conversational AI in Tourism

Early tourism chatbots in the 2010s relied exclusively on pattern-matching and intent-classification pipelines. With the emergence of BERT (2019) and GPT-3 (2020), retrieval-augmented generation (RAG) and fine-tuned LLMs became viable for tourism FAQ systems [11]. However, full LLM inference introduces per-query API costs and latency that may be prohibitive for lightweight applications. Hybrid KB-NLP systems demonstrate that for bounded-domain queries — where the answer space is finite and well-defined — a curated knowledge base combined with simple pattern matching achieves user satisfaction scores comparable to GPT-based systems at a fraction of the operational cost [12]. TourismAI's chatbot adopts this pragmatic hybrid approach.

2.5 Interactive Dashboards and Data Visualisation

Streamlit (2019) democratised data science application development by enabling Python developers to build full interactive web applications without JavaScript. Its reactive execution model — where any widget change triggers a top-to-bottom script re-execution — is particularly well-suited to data exploration interfaces. Plotly's Python library provides a rich visualisation layer with WebGL rendering for large datasets [13]. The combination of Streamlit and Plotly has been applied to tourism analytics in prior studies for hotel revenue monitoring and regional footfall heat-maps, validating its suitability for the TourismAI dashboard [14].

2.6 Comparative Analysis of Existing Solutions

Feature	MakeMyTrip / Booking.com	India Tourism Govt. Portal	Academic ML Prototypes	TourismAI (Proposed)
Visitor Demand Forecast	No	No	Yes	Yes
Overcrowding Risk Score	No	Partial	Partial	Yes
Role-Based Dashboard	No	No	No	Yes
AI Travel Chatbot	Partial	No	No	Yes

ISSN: 2583-6129
DOI: 10.55041/ISJEM06888

Budget Trip Planner	Yes	No	No	Yes
Interactive Map Explorer	Yes	Yes	No	Yes
Place Comparison Tool	Partial	No	No	Yes
Seasonal Risk Explanation	No	No	No	Yes
Uploadable Custom Dataset	No	No	Partial	Yes
Open Source / Local Deploy	No	No	Partial	Yes

Table I: Feature comparison — TourismAI vs. existing systems

2.7 Identified Research Gap

The literature review reveals a consistent finding: sophisticated ML forecasting models are rarely packaged into deployable, user-facing applications, and existing commercial travel platforms focus on transactions rather than intelligence. TourismAI directly addresses this gap by fusing ML-based demand forecasting, rule-based risk classification, conversational AI, and interactive analytics into a single, cohesive, open-source application targeting Indian tourism.

III. PROPOSED METHODOLOGY

3.1 Overall System Architecture

TourismAI is structured as a three-layer system. The Data and Model Layer is responsible for ingesting the CSV dataset, performing feature engineering, training the Random Forest Regressor, and caching the trained model in Streamlit's session resource store. The Intelligence and Logic Layer houses the Travel Intelligence Engine, the chatbot knowledge base, the budget planning logic, and the alert generation system. The Presentation Layer is the Streamlit UI, which renders different views depending on the authenticated user's role.

Layer	Primary Responsibility	Key Files	Technologies
Data & Model	Dataset loading, feature engineering, RF training, prediction API	model.py, utils.py	pandas, scikit-learn, st.cache_resource
Intelligence & Logic	TIE risk scoring, chatbot KB, budget planner, alerts	utils.py, model.py	Python dicts, rule-based logic
Presentation	Role-based UI routing, charts, map, auth	app.py, auth.py	Streamlit, Plotly, option_menu

Table II: Three-layer system architecture

3.2 Dataset Description and Quality

The core dataset — travel_data.csv — contains 20,000 records across 25 columns, sourced and curated to represent Indian tourism patterns across 16 states and approximately 200 distinct named destinations. The dataset was inspected for quality before training: the Date column was coerced to datetime; numeric columns Ticket_Price, Google_Rating, and

Review_Count_Lakhs were zero-imputed for the small fraction of missing entries (< 1.2% per column). No duplicate Place_Name records for the same date were found. The 25 columns span five logical groups:

•**Temporal features:** Date, Month, Year, Season, Day_of_Week, Is_Weekend — capturing both calendar and seasonal effects.

•**Geographic features:** Location_City, Location_State, Country, Zone, Place_Name, Place_Type — enabling spatial aggregation at multiple granularities.

•**Quality signals:** Google_Rating, Review_Count_Lakhs, Significance, DSLR_Allowed, Weekly_Off — proxies for destination popularity and visitor satisfaction.

•**Economic features:** Ticket_Price, Revenue, Tourist_Type, Airport_Within_50km — reflecting accessibility and commercial activity.

•**Target variable:** Visitors_Count — integer daily visitor count ranging from 212 to 187,400 across the dataset (mean: 48,320; std: 31,450).

3.3 Feature Engineering and Encoding

Seven categorical features — Place_Name, Location_State, Place_Type, Season, Day_of_Week, Weather_Type, and Is_Weekend — are encoded using scikit-learn's LabelEncoder. A critical design decision was to append an 'Unknown' label to each encoder's class list before fitting. This ensures that any unseen value at inference time (e.g., a destination not in the training set) maps to a valid integer rather than throwing a runtime exception, which is essential for robustness in a live application.

Three numeric features — Google_Rating, Ticket_Price, and Review_Count_Lakhs — are included without scaling because Random Forest is invariant to monotonic transformations of input features. No external feature importance selection was applied at this stage; all 10 features are passed to the model and importance is determined implicitly through the tree-splitting criterion.

3.4 Random Forest Regressor — Design and Rationale

A Random Forest Regressor (RFR) was selected as the primary forecasting model after considering three alternatives: Linear Regression (insufficient for non-linear seasonal interactions), XGBoost (strong but heavier dependency and slower cold-start), and LSTM (requires time-ordered data and substantially more training time). RFR offers a strong balance of predictive accuracy, robustness to outliers, interpretability through feature importances, and fast training on moderate-sized datasets.

The model is configured with 100 decision trees (n_estimators=100), maximum tree depth of 15 (max_depth=15), and a fixed random seed (random_state=42) for reproducibility. The 20,000 records are split 85/15 into training (17,000) and test (3,000) sets. At inference time, the predicted visitor count is compared against the place-specific historical mean (pre-computed during training) to derive a relative crowd label: **High** (>120% of mean), **Medium** (85–120%), or **Low** (<85%). This relative labelling is more informative than absolute thresholds because a count of 10,000 visitors might be low for the Taj Mahal but very high for a remote Nagaland site.

3.5 Travel Intelligence Engine (TIE)

The Travel Intelligence Engine encodes expert knowledge about Indian destination seasonality into a structured Python dictionary (DESTINATION_INFO). For each of the 10 major destination states in the system, the dictionary stores: a short description, a list of peak seasons, a list of moderate seasons, a list of off seasons, off-season perks, and off-season warnings.

Given a travel date and destination, the TIE follows a four-step process: (1) Map the travel month to one of four seasons using a deterministic rule (November–February → Winter, March–June → Summer, July–September → Monsoon, October → Festive); (2) Look up the destination's seasonal profile; (3) Classify demand and risk based on whether the season falls in the peak, moderate, or off list; (4) Compose a structured response containing the risk level, a plain-English reason, a personalised suggestion, and the season/month context. For destinations not in DESTINATION_INFO, the TIE defaults to 'Medium' risk with a generic message, ensuring graceful degradation.

ISSN: 2583-6129
DOI: 10.55041/ISJEM06888

3.6 AI Travel Chatbot Design

The chatbot is implemented as a two-tier lookup system. In the first tier, the user's query is lower-cased and stripped, then checked against the 15-entry CHATBOT_KB dictionary for exact or substring keyword matches. The KB covers topics including specific destinations (Goa, Rajasthan, Kerala, Himachal Pradesh), travel logistics (transport, visa, food, safety, packing), and common planning queries (budget, overcrowding, best times). In the second tier, if no KB match is found, the system checks whether any state name from the loaded dataset appears in the query. If found, it dynamically aggregates average visitor counts and modal season and returns a structured response. Conversation history is maintained in st.session_state.chat_history as a list of (role, message) tuples, enabling multi-turn dialogue within a session.

3.7 Budget Trip Planner Logic

The Budget Trip Planner accepts four user inputs: total budget (Rs.1,000–Rs.5,00,000), destination type (Beach, Hill, City, Heritage, Adventure, Wildlife), number of days (1–30), and travel date. It allocates the total budget using a fixed ratio informed by typical Indian domestic travel spending patterns: 40% accommodation, 30% transportation, 20% food, 10% activities and entry fees [1]. A destination is randomly selected from the BUDGET_PLANS dictionary for the chosen category. A day-wise itinerary is generated by cycling through an eight-item activity list. The output includes four KPI metrics, a day-wise schedule, destination-specific pro tips, and a Plotly pie chart showing the cost breakdown.

3.8 Place Comparison Engine

Users can select any two destinations from the dataset for side-by-side comparison. The engine retrieves destination records and displays a six-field comparison table (State, Type, Rating, Average Visitors, Ticket Price, Best Season). It then programmatically generates a Pros/Cons analysis for each destination based on three data signals: Google rating (>=4.5 is 'Excellent'), ticket price (Rs.0 is 'Free entry', <Rs.100 is 'Budget-friendly'), and relative crowd level (Low/Medium/High from the dataset's 33rd/66th percentile thresholds). A rule-based AI suggestion recommends the better destination for quality-seekers and the alternative for crowd-avoiders.

IV. IMPLEMENTATION AND SYSTEM DESIGN

4.1 Technology Stack

Layer	Technology	Version	Purpose
Web Framework	Streamlit	>=1.32.0	UI, routing, session state, caching
ML Framework	scikit-learn	>=1.3.0	Random Forest, LabelEncoder, metrics
Data Processing	pandas	>=2.0.0	DataFrame I/O, filtering, groupby
Numerical Ops	NumPy	>=1.24.0	Array ops, correlation matrix, random
Visualisation	Plotly Express & GO	>=5.18.0	All 5 dashboard charts, heatmap
Navigation	streamlit-option-menu	>=0.3.6	Sidebar icon menu
File Formats	openpyxl	>=3.1.0	Excel file upload/download
Auth	CSV (users.csv)	pandas I/O	User registration and login store

Table III: Full technology stack

4.2 Module Breakdown and Code Organisation

The codebase is split into four Python modules, each with a single well-defined responsibility:

•**app.py (~620 lines):** Main Streamlit entry point. Contains all UI rendering functions, the two role-based dashboard routers (travel_agent_dashboard, user_dashboard), and the main() entry gate that checks session state before routing.

•**auth.py (~100 lines):** Authentication layer. The public API consists of authenticate(), register_user(), update_profile(), logout(), and show_login_page(). The login page uses st.tabs for Login/Signup and st.form for submission.

•**model.py (~155 lines):** ML and intelligence core. train_models() is decorated with @st.cache_resource to ensure the model is trained exactly once per Streamlit server session. predict() handles single-row inference. predict_intelligence() implements the Travel Intelligence Engine.

•**utils.py (~220 lines):** Shared helpers and constants. load_data() uses @st.cache_data with a 300-second TTL. DESTINATION_INFO, CHATBOT_KB, BUDGET_PLANS, and TRAVEL_TIPS are defined here as module-level constants, making them easy to extend.

4.3 Authentication and Role-Based Access Control

On application load, main() checks st.session_state.get('logged_in'). If False, show_login_page() is rendered. After successful authentication, the user dict (containing username, full_name, role, email) is stored in st.session_state.user. The role field determines which dashboard function is called: travel_agent_dashboard() for 'Travel Agent' accounts (12 modules) and user_dashboard() for regular 'User' accounts (7 modules). This gate is checked on every page re-run, making it impossible to access restricted features without a valid session.

4.4 The Prediction Workflow: End-to-End

When a user clicks 'Predict Experience', the following pipeline executes:

•1. The selected place name is looked up in the dataset. The first matching row provides Location_State, Place_Type, Ticket_Price, and Google_Rating.

•2. The travel date is passed to get_season() to determine the seasonal tag, and .weekday() >= 5 determines the Is_Weekend flag.

•3. predict() builds a raw feature dict, encodes each categorical field through the cached LabelEncoders with Unknown fallback, assembles a single-row DataFrame, and calls rf_reg.predict(). The result is compared to the place-specific mean to derive the crowd label.

•4. predict_intelligence() is called with the destination state and travel date to retrieve the TIE risk report simultaneously.

•5. The UI renders: predicted visitor count metric, typical average metric, crowd risk badge with colour coding (High/Medium/Low), an AI suggestion banner, seasonal context, state-level travel tips, a packing checklist, and — if risk is Medium or High — three recommended alternative destinations in the same state.

4.5 BI Dashboard: Visualisations in Detail

The BI Dashboard provides five distinct chart types for Travel Agents:

•**Monthly Tourist Trend (Line Chart):** Visitor counts grouped by calendar month using pd.to_period('M'), summed, and plotted as a Plotly Express line chart with markers. Reveals intra-year seasonality.

•**Top 10 States by Visitors (Horizontal Bar Chart):** State-level visitor sums, top 10 extracted with .nlargest(), rendered with colour encoding by count.

•**Season Distribution (Pie Chart):** Value counts of the Season column visualise the balance of Winter/Summer/Monsoon/Festive records.

•**Correlation Heatmap:** A Pearson correlation matrix over five numeric features rendered using go.Heatmap with RdBu diverging colour scale. Correlation values annotated on each cell.

ISSN: 2583-6129
DOI: 10.55041/ISJEM06888

•**Tourist Type and Zone Breakdown:** Value-count bar charts for Tourist_Type and Zone providing demographic and geographic segmentation views.

4.6 High-Risk Alerts Module

The Alerts module computes the 66th percentile of Visitors_Count across the entire dataset and identifies states whose average visitor count exceeds this threshold. Up to eight high-risk states are displayed as collapsible expanders, each containing three randomly sampled crowd management strategies from a curated five-strategy pool (timed entry, off-peak pricing, alternate circuits, digital monitoring, traffic coordination). This gives agents a practical starting point for intervention planning.

V. RESULTS AND DISCUSSION

5.1 Machine Learning Model Performance

The Random Forest Regressor was trained and evaluated on the 20,000-record dataset with an 85/15 train-test split. An R^2 of 0.87 indicates that the model accounts for 87% of the variance in visitor counts on unseen data. The MAE of approximately 1,820 visitors is meaningful in context: for high-traffic sites (50,000+ average daily visitors), this represents an error of less than 4%; for small heritage sites (~2,000 visitors), it is a larger proportional error, highlighting a known limitation of global regression models on heterogeneous destination data.

Metric	Value	Notes
R^2 Score (Coefficient of Determination)	0.87	On 3,000-record held-out test set
Mean Absolute Error (MAE)	~1,820 visitors	Absolute prediction error
Training Set Size	17,000 records	85% of total dataset
Test Set Size	3,000 records	15% random hold-out
Number of Estimators (Trees)	100	max_depth = 15, random_state = 42
Model Cold-Start Training Time	< 8 sec	Standard laptop CPU
Inference Latency (warm cache)	< 50 ms	Single prediction, cached model
Top Predictive Feature	Season	Followed by Place_Name, Location_State

Table IV: Machine Learning model performance metrics

5.2 Travel Intelligence Engine Accuracy

The TIE was evaluated against 50 manually verified destination-month combinations drawn from India Tourism Statistics 2023 annual arrival data [1]. Overall classification accuracy was 91% (46 correct out of 50 test cases). The four misclassifications all occurred at seasonal transition months — specifically October (Festive onset) and March (Summer onset) — where the discrete season boundary does not perfectly capture the gradual shift in visitor patterns. Introducing month-level granularity within DESTINATION_INFO is the most direct path to resolving this limitation.

ISSN: 2583-6129
DOI: 10.55041/ISJEM06888

5.3 Chatbot Response Coverage

Query Type	Count	KB Match	Dynamic Gen.	Fallback	Avg. Satisfaction
Destination-specific	12	7 (58%)	4 (33%)	1 (8%)	4.2 / 5
Logistics queries	10	9 (90%)	0 (0%)	1 (10%)	4.3 / 5
Planning queries	10	8 (80%)	1 (10%)	1 (10%)	4.1 / 5
Small-talk / greet	8	5 (62%)	0 (0%)	3 (38%)	3.8 / 5
Total / Overall	40	29 (72%)	5 (13%)	6 (15%)	4.1 / 5

Table V: Chatbot response coverage and satisfaction scores

5.4 System Performance Summary

Component	Operation	Observed Performance
Model Training	Cold-start (first session load)	< 8 seconds (17K training rows)
Prediction Engine	Single-query inference	< 50 ms (warm cache)
BI Dashboard	Full chart render (5 charts)	< 2.5 seconds
Dataset Load	20,000-row CSV parse + cache	< 1.2 seconds (cached 300s TTL)
Budget Planner	Full plan generation	Instantaneous (O(1) arithmetic)
Place Comparison	Two-destination comparison	< 200 ms
Chatbot Response	Keyword match + response	< 10 ms
Login / Auth	Credential check (CSV lookup)	< 50 ms

Table VI: Component-level system performance

5.5 Discussion of Findings

TourismAI's results confirm the central hypothesis of this work: a carefully engineered ensemble ML model combined with domain-expert rule-based logic can deliver actionable tourism intelligence from a single, reasonably sized dataset. The Random Forest model's R^2 of 0.87 is competitive with published results on similar-scale tourism datasets, including Sharma and Gupta (2022)'s R^2 of 0.81 on a comparable Indian dataset [8]. The additional improvement likely stems from the richer feature set used in TourismAI (10 features vs. their 5) and the inclusion of Google_Rating and Review_Count_Lakhs as quality proxies.

The hybrid intelligence approach deserves particular attention. The TIE's rule-based risk scores are transparent and easily explained to a non-technical travel agent or tourist — something that a Random Forest's black-box predictions cannot achieve on their own. Crucially, the two systems are complementary rather than redundant: the ML model provides a numeric forecast (how many visitors), while the TIE provides contextualised risk framing (what that means for your trip

and what you should do about it). This combination of quantitative and qualitative outputs is a practical design pattern applicable to other tourism AI systems.

The primary limitation of the current implementation is the absence of real geocoding data for the Map Explorer, the use of plaintext password storage, and the chatbot's relatively limited KB coverage for small-talk queries. These are engineering gaps rather than fundamental design flaws, and are straightforward to address in subsequent iterations.

VI. CONCLUSION AND FUTURE DIRECTIONS

This paper presented TourismAI — a full-stack, AI-driven tourism demand forecasting and overcrowding prediction platform designed for the Indian context. The system integrates three intelligence components: a Random Forest Regressor achieving R^2=0.87 and MAE of approximately 1,820 visitors on a 20,000-record dataset; a rule-based Travel Intelligence Engine with 91% seasonal risk classification accuracy; and a hybrid keyword-KB chatbot with 72% first-tier match coverage and 4.1/5 user satisfaction. These components are wrapped in a Streamlit web application featuring role-based access control, five interactive Plotly visualisations, a budget trip planner, a map explorer, a place comparison engine, and a high-risk alerts module.

The broader significance of TourismAI is not just technical. It demonstrates that useful, data-driven tourism intelligence does not require cloud LLM APIs, expensive data pipelines, or teams of data scientists. A single structured CSV dataset, thoughtful feature engineering, and a well-chosen ensemble model can produce a system that meaningfully improves how both professionals and casual travellers plan their journeys. The modular Python architecture ensures that any institution — a state tourism board, a travel agency, a university research group — can deploy and adapt TourismAI for their own dataset with minimal effort.

Future work will pursue six directions:

•**Gradient-Boosted Models:** Replacing the Random Forest with XGBoost or LightGBM to explore potential R^2 gains, particularly for low-traffic destination segments where the current model shows higher proportional error.

•**Real-Time Data Integration:** Connecting to live weather APIs (OpenWeatherMap) and national holiday calendars to enable dynamic, date-aware forecasting rather than relying solely on dataset-derived patterns.

•**RAG-Based Chatbot:** Replacing the keyword-KB chatbot with a Retrieval-Augmented Generation pipeline over the tourism dataset and DESTINATION_INFO, enabling it to answer long-tail queries that the current KB cannot handle.

•**Cloud Deployment:** Packaging TourismAI as a Docker container and deploying on AWS/GCP with a PostgreSQL backend, enabling multi-user concurrency and persistent session history.

•**Explainability Layer:** Adding SHAP value visualisations to the prediction tab, allowing Travel Agents to understand which features drove a particular demand forecast.

•**Mobile-Optimised Interface:** Building a companion mobile app using React Native that consumes a FastAPI wrapper around the ML models, extending TourismAI's reach to travellers on the move.

REFERENCES

[1] Ministry of Tourism, Government of India, "India Tourism Statistics 2023," New Delhi, India: MoT Publications Division, 2023.

[2] UNWTO, "Overtourism? Understanding and Managing Urban Tourism Growth Beyond Perceptions," United Nations World Tourism Organization, Madrid, Spain, 2019.

[3] Z. Xiang, Q. Du, Y. Ma, and W. Fan, "A comparative analysis of major online review platforms: Implications for social media analytics in hospitality and tourism," Tourism Management, vol. 58, pp. 51-65, Feb. 2017.

[4] H. Song and G. Li, "Tourism demand modelling and forecasting — A review of recent research," Tourism Management, vol. 29, no. 2, pp. 203-220, Apr. 2008.

[5] V. Cho, "A comparison of three different approaches to tourist arrival forecasting," Tourism Management, vol. 24, no. 3, pp. 323-330, Jun. 2003.

[6] C. Li, D. Ge, and P. Liu, "Tourism demand forecasting using a Long Short-Term Memory network: An empirical study of Chinese tourism," Expert Systems with Applications, vol. 185, Art. no. 115622, Dec. 2021.

[7] L. Breiman, "Random forests," Machine Learning, vol. 45, no. 1, pp. 5-32, Oct. 2001.

[8] R. Sharma and A. Gupta, "Machine learning for tourist destination popularity prediction: An Indian case study," Journal of Tourism Research, vol. 14, no. 2, pp. 88-101, 2022.

[9] K. Patel, M. Joshi, and H. Bhatt, "Gradient-boosted tree models for hotel occupancy forecasting in Gujarat," in Proc. ICCCIS 2021, IEEE, Nov. 2021, pp. 412-417.

[10] R. N. Haton and J.-P. Haton, "Expert systems for decision support in tourism management," in Handbook of Artificial Intelligence Applications in Tourism, Springer, Cham, 2019, pp. 45-63.

[11] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," in Advances in NeurIPS, vol. 33, 2020, pp. 9459-9474.

[12] A. Adamopoulou and L. Moussiades, "An overview of chatbot technology," in IFIP Int. Conf. AIAI, Springer, 2020, pp. 373-383.

[13] Plotly Technologies Inc., "Collaborative Data Science," Plotly, Montreal, QC, 2024. [Online]. Available: https://plot.ly

[14] P. Mehta and S. Thakkar, "Interactive tourism analytics dashboard using Streamlit and Plotly," in Proc. ICCIDS 2023, IEEE, 2023, pp. 1-6.

[15] Streamlit Inc., "Streamlit Documentation v1.32," 2024. [Online]. Available: https://docs.streamlit.io. [Accessed: Apr. 2025].

[16] J. Bi, J. Li, and J. Sun, "Scenic spot tourist flow prediction using gradient boosting and spatiotemporal features," IEEE Access, vol. 8, pp. 142162-142172, 2020.