GitHUb link: -https://github.com/mansi0406/Tourism-Experience-Analytics

Tourism Experience Analytics: Documentation

Summary

This end-to-end Tourism Experience Analytics project builds a personalized travel assistant that ingests multi-table Excel data (Transaction, User, Item, Type, Mode, City/Country/Region/Continent), robustly merges and cleans it (numeric ratings clipped to 1–5, dates constructed from year+month, duplicates removed), engineers key features (user_total_visits, user_avg_rating, attraction_popularity, season/month/year), and generates EDA charts to understand behavior. It then prepares model inputs via consistent one-hot encoding, trains a rating regressor (RandomForestRegressor) and a visit-mode classifier (RandomForestClassifier) with metrics saved, and constructs an item-based recommender using cosine similarity over a user–item matrix, with popularity fallback for cold-start users. All artifacts (cleaned data, feature columns, models, metrics, similarity, EDA images) are saved under outputs/ and surfaced in a styled Streamlit app with three capabilities: predict rating, predict visit mode, and recommend attractions. The result is a reproducible, deployment-ready pipeline and UI that deliver actionable personalization while remaining easy to run (python pipeline.py, streamlit run app.py) and extend (filters, names/images, hybrid recommender, model tuning).

Overview

This document summarizes the end-to-end project: data merge, cleaning, feature engineering, exploratory data analysis (EDA), supervised models, recommendation system, and Streamlit app.

Business Problem

- Increase user satisfaction and retention by suggesting relevant attractions.
- Understand customer behavior across seasons, regions, and trip types.

Data Sources

Tables used: Transaction, User, Item (+ Updated_Item), Type, Mode, City, Country, Region, Continent.

Pipeline Outputs

Key artifacts are produced under outputs/: cleaned_master.csv, feature_columns.json, regression_model.joblib, classification_model.joblib, item_similarity.csv, metrics.json, and eda/ charts.

Model Metrics

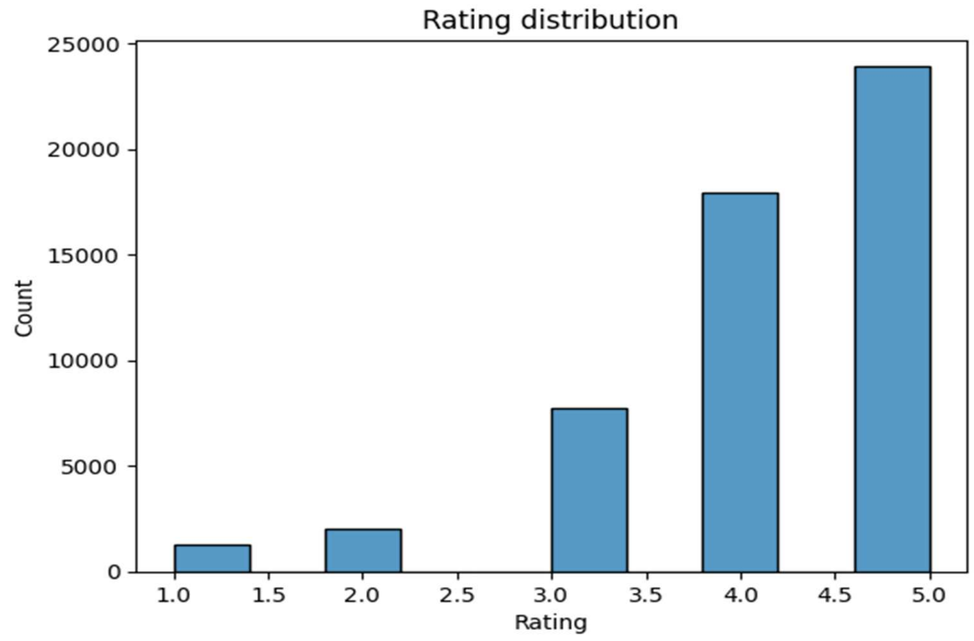
Model	Metric	Value
Regression	R^2	0.6879762779731884
Regression	RMSE	0.5420968492971248
Classification	Accuracy	0.5550727375779331
Classification	F1	0.5359599572061424

GitHUb link: -https://github.com/mansi0406/Tourism-Experience-Analytics

EDA Charts

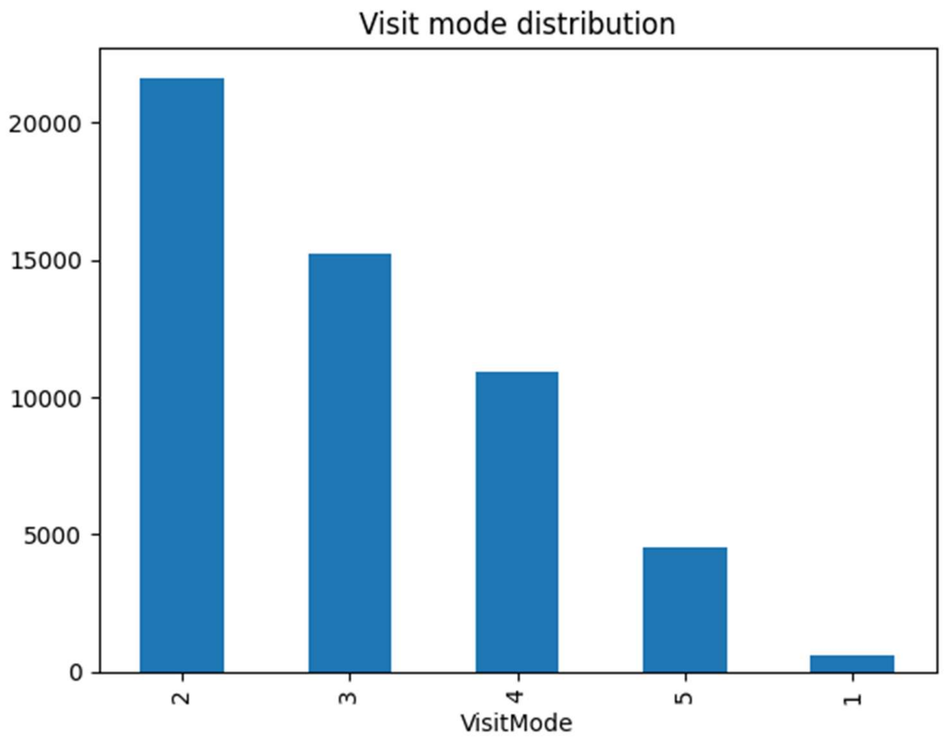
Rating Distribution

- Shows how ratings are spread across 1–5; most are high, indicating satisfied users.
- Models should expect low variance; RMSE around ~1 is reasonable on a 1–5 scale.



Visit Mode Distribution

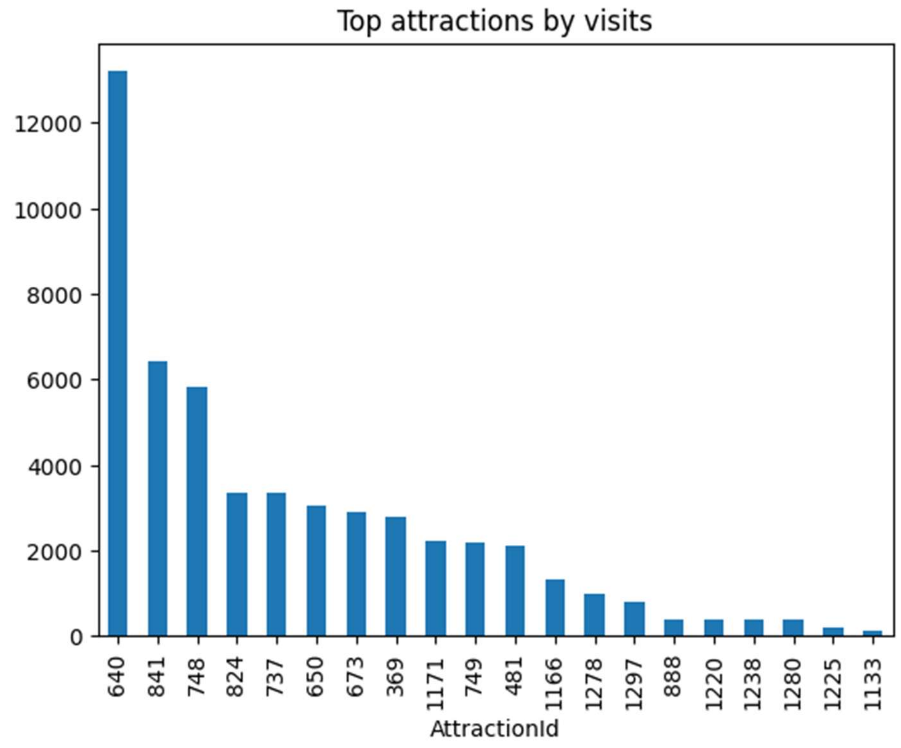
- Displays how often each trip type occurs; reveals class imbalance if one mode dominates.
- Use weighted metrics (F1) and consider class balancing to improve minority predictions.



GitHUb link: -https://github.com/mansi0406/Tourism-Experience-Analytics

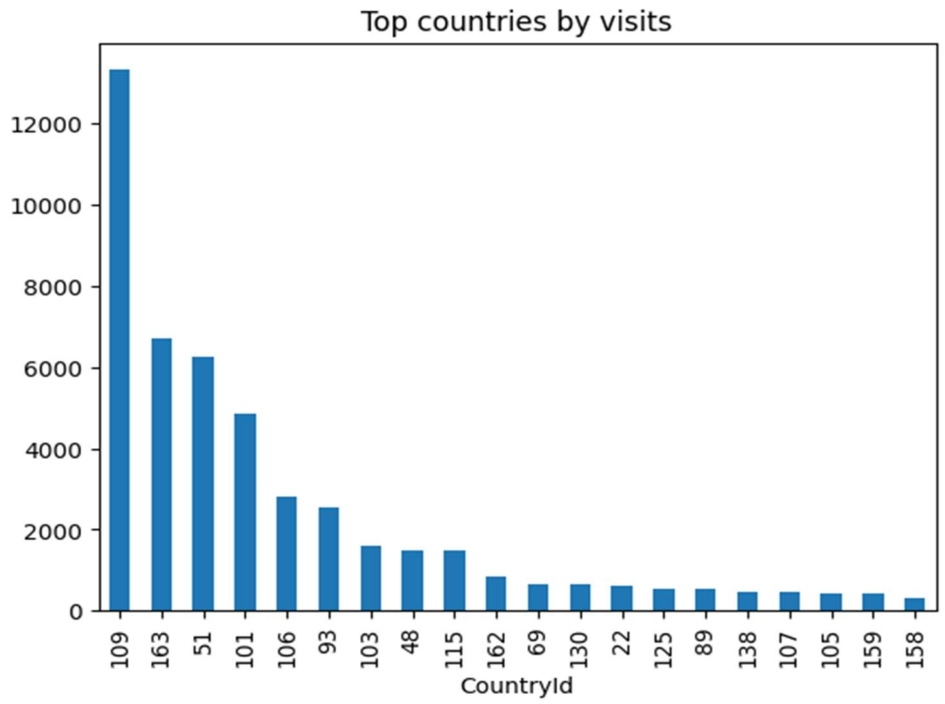
Top Attractions

- Ranks attractions by visits; a few items draw most traffic (popularity bias).
- Useful for cold-start fallback and to highlight crowd favorites.



Top Countries

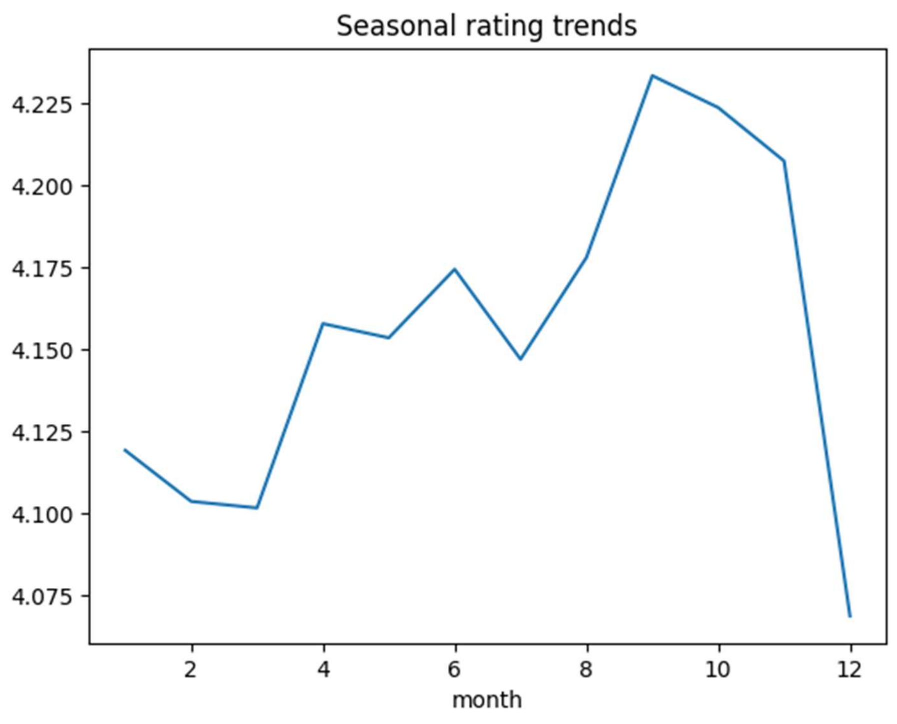
- Shows where visits concentrate geographically; location features matter.
- Helps tailor recommendations and promotions by region.



GitHUb link: -https://github.com/mansi0406/Tourism-Experience-Analytics

Seasonal Trends

- Plots average rating by month; peaks suggest seasonality (e.g., summer).
- Keep month/season as features; they influence both satisfaction and visit mode.



Feature Engineering

Derived features include: user_total_visits, user_avg_rating, attraction_popularity, season, month, year.

Recommendation System

Item-based collaborative filtering using cosine similarity over the user–item rating/interaction matrix. Cold-start and popularity fallbacks are implemented. - User–Item Matrix: ratings when available, interactions otherwise.
- Item Similarity: cosine similarity; IDs normalized to strings for robust lookup.
- Scoring: similarity-weighted sums excluding already seen items.
- Cold-Start: fallback to attraction_popularity or value_counts.

Streamlit App

Multi-page UI to predict rating, predict visit mode, and recommend attractions. Includes styled components and data previews.

Tabs:
- Predict Rating: builds a single feature row, vectorizes, predicts.
- Visit Mode Prediction: enhanced UI; inputs for user, date, region, continent, type; predicts visit mode.

- Recommendations: selects user, returns top-k personalized or popular fallback; robust column detection.

How to Run

- Setup: install Python packages listed in README.
- Pipeline: python pipeline.py produces data, models, metrics, charts.
- App: streamlit run app.py and open http://localhost:8501/

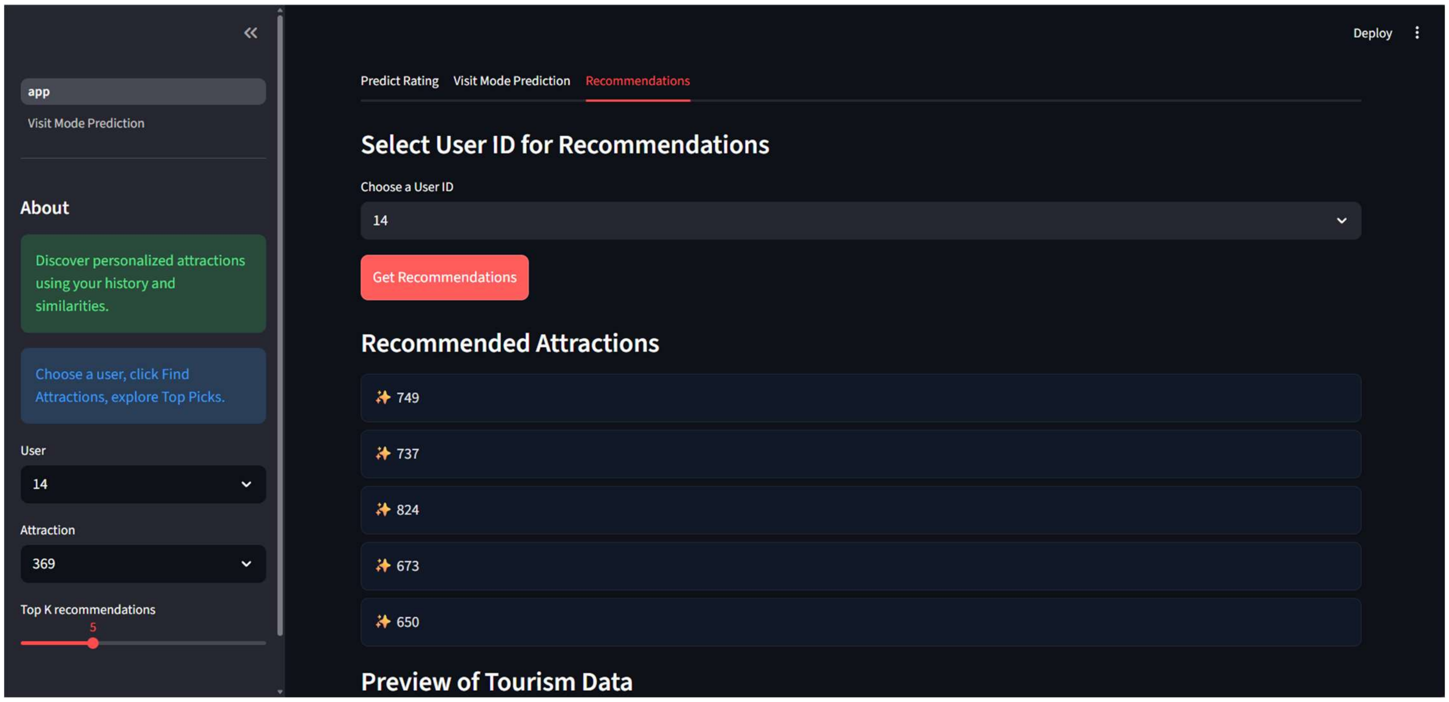
1) python pipeline.py 2) streamlit run app.py 3) Open http://localhost:8501/

- Predict Rating tab

- Takes selected User and Attraction; builds feature row; predicts with regression model.
- Visit Mode Prediction tab
- Inputs for user/date/region/continent/type; updates feature row; predicts visit mode via classifier.

- Recommendations tab

- Select user; compute personalized top-k via similarity.
- Fallbacks: engineered popularity; otherwise value_counts of attraction IDs.
- Always shows a data preview; avoids errors if columns are missing.



About

Discover personalized attractions using your history and similarities.

Choose a user, click Find Attractions, explore Top Picks.

User

14

Attraction

369

Top K recommendations

5

Select User ID for Recommendations

Choose a User ID

14

Recommended Attractions

✨ 749

✨ 737

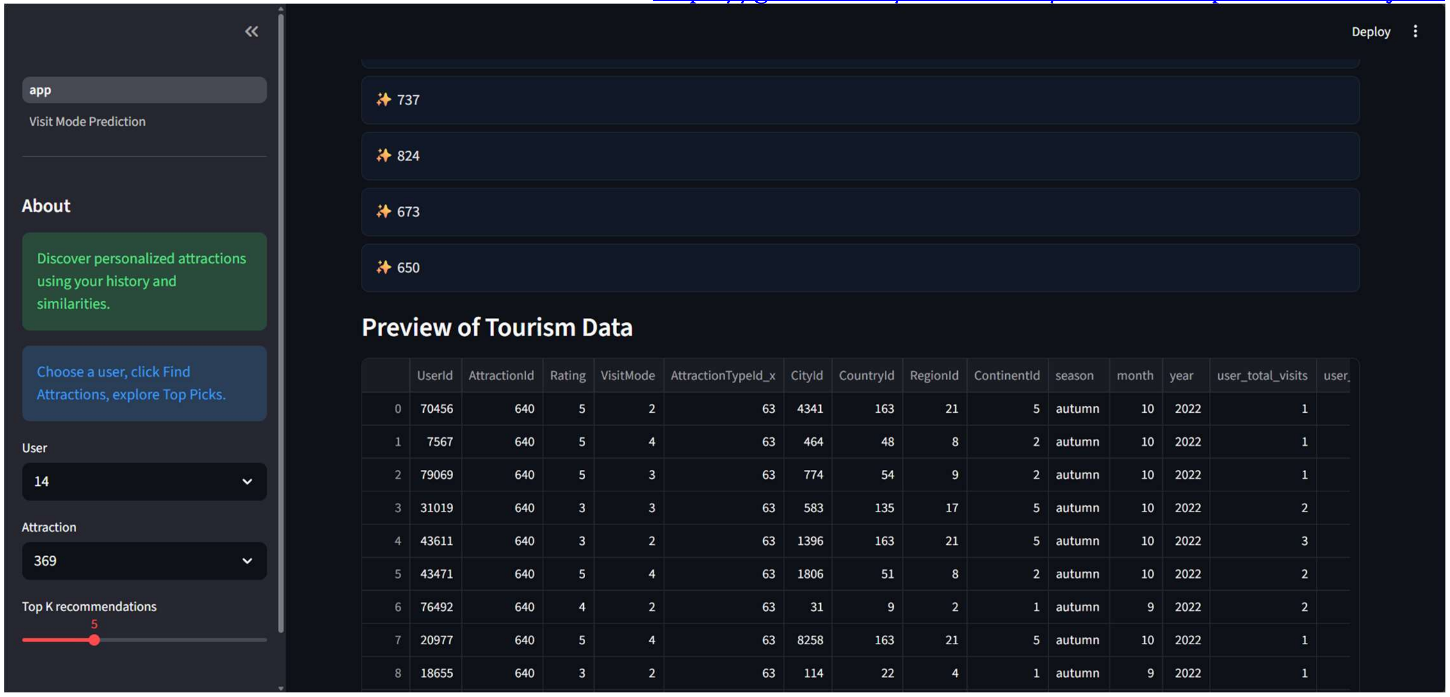
✨ 824

✨ 673

✨ 650

Preview of Tourism Data

GitHUb link: -https://github.com/mansi0406/Tourism-Experience-Analytics



About

Discover personalized attractions using your history and similarities.

Choose a user, click Find Attractions, explore Top Picks.

User

14

Attraction

369

Top K recommendations

5

✨ 737

✨ 824

✨ 673

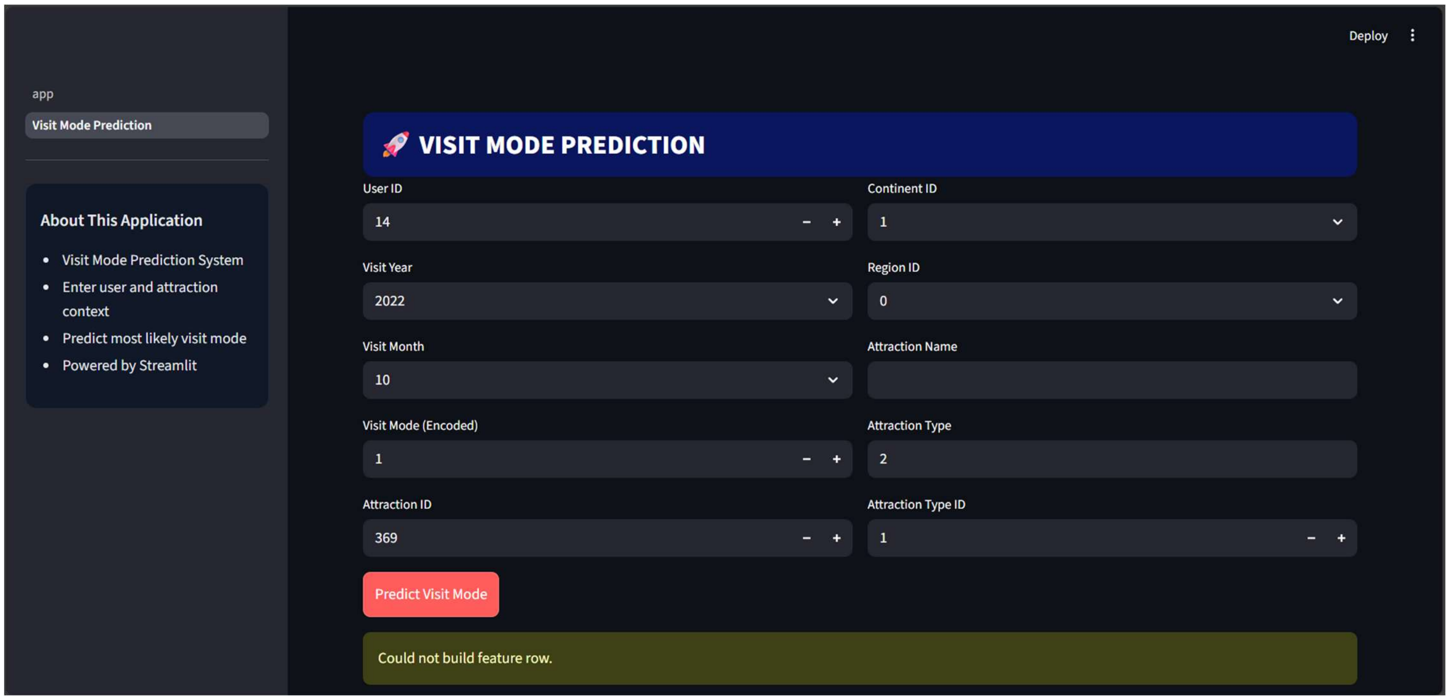
✨ 650

Preview of Tourism Data

	UserId	AttractionId	Rating	VisitMode	AttractionTypeId_x	CityId	CountryId	RegionId	ContinentId	season	month	year	user_total_visits	user_
0	70456	640	5	2	63	4341	163	21	5	autumn	10	2022	1	
1	7567	640	5	4	63	464	48	8	2	autumn	10	2022	1	
2	79069	640	5	3	63	774	54	9	2	autumn	10	2022	1	
3	31019	640	3	3	63	583	135	17	5	autumn	10	2022	2	
4	43611	640	3	2	63	1396	163	21	5	autumn	10	2022	3	
5	43471	640	5	4	63	1806	51	8	2	autumn	10	2022	2	
6	76492	640	4	2	63	31	9	2	1	autumn	9	2022	2	
7	20977	640	5	4	63	8258	163	21	5	autumn	10	2022	1	
8	18655	640	3	2	63	114	22	4	1	autumn	9	2022	1	

Visit Mode page

- Separate Streamlit page mirrors the styled visit-mode form:
  - Loads artifacts, applies CSS, presents inputs for user/context, predicts visit mode, shows data preview.



About This Application

- Visit Mode Prediction System
- Enter user and attraction context
- Predict most likely visit mode
- Powered by Streamlit

🚀 **VISIT MODE PREDICTION**

User ID: 14	Continent ID: 1
Visit Year: 2022	Region ID: 0
Visit Month: 10	Attraction Name:
Visit Mode (Encoded): 1	Attraction Type: 2
Attraction ID: 369	Attraction Type ID: 1

Predict Visit Mode

Could not build feature row.

Interpretation of Metrics

Rating: R^2 indicates explained variance; RMSE shows typical error magnitude (1–5 scale).

GitHUb link: -https://github.com/mansi0406/Tourism-Experience-Analytics

Visit Mode: Accuracy and weighted F1 reflect overall and class-balanced performance.

Limitations

- One-hot encoding can inflate feature space; consider target/category encoders.
- Popularity bias in recommendations; consider hybrid with content features.
- Sparse histories reduce personalized recs; rely on fallbacks.

Security and Privacy

- No PII is exposed in outputs; avoid storing raw sensitive fields in shared artifacts.
- Do not log secrets; keep environment variables out of code.

Reproducibility

- Deterministic seeds set for train/test splits.
- All artifacts written to outputs/ for consistent app runs.

Future Enhancements

- Show attraction names alongside IDs in app and docs.
- Filters: country, type, season; export recommendations to CSV.
- Model tuning: cross-validation, gradient boosting, richer features (text, pricing).
- Hybrid recommender: combine similarity with content signals.

Acknowledgements

- Libraries: pandas, numpy, scikit-learn, Streamlit, seaborn, matplotlib.
- Data schema: hierarchical geography and attraction taxonomy.

Architecture

- End-to-end data flow diagram description (ingest → merge → clean → features → EDA → models → recsys → app).
- Responsibility boundaries for each module (pipeline, app, pages).

Data Dictionary

- Column-by-column table: name, type, meaning, origin table, example values.
- Column name mapping across files (e.g., AttractionId vs ItemId).

Assumptions & Constraints

- Data quality assumptions (ratings 1–5, month/year present).

GitHUb link: -https://github.com/mansi0406/Tourism-Experience-Analytics

- Constraints (join keys required, minimum rows to compute similarity).

Preprocessing Decisions

- Why ratings were clipped; why one-hot encoding; how missing values were handled.
- Duplicate removal criteria; date construction logic (year+month → first of month).

Feature Importance & Explainability

- Model feature importance summary for regression/classification.
- Optional SHAP overview: what features drive predictions; example visual references.

Error Analysis

- Typical regression errors (where predictions underperform).
- Per-class precision/recall for visit mode; confusion matrix narrative.

Training Details

- Train/test split rationale; random seeds; stratification.
- Hyperparameters used; what to tune next and expected trade-offs.

Recommender Mechanics

- Similarity computation explanation; cold-start fallback strategy.
- Example scoring walkthrough for a single user (1–2 sentences).

Evaluation & KPIs

- Primary metrics (R^2, RMSE, Accuracy, weighted F1).
- Recsys offline metrics candidates (Precision@K, Recall@K, NDCG@K) and why they matter.

Reproducibility

- Exact commands to regenerate artifacts and app.
- Folder paths and environment setup; deterministic seeds noted.

Deployment & Operations

- How to package/run on another machine; dependency list.
- Logging/monitoring suggestions (basic health checks, error capture).

GitHUb link: -https://github.com/mansi0406/Tourism-Experience-Analytics

Security & Privacy

- PII handling policy; no secrets in code; where artifacts are safe to share.

User Guide

- Step-by-step app usage: Rating, Visit Mode, Recommendations.
- Common issues and fixes (e.g., rebuild pipeline, hard refresh, popularity fallback shown).

Roadmap

- Attraction names/images in UI, filters and CSV export, model tuning, hybrid recommender.
- Text embeddings for descriptions; seasonal personalization; cohort targeting.

Conclusion

This project delivers a complete, reproducible pipeline and a practical Streamlit app that personalize the tourism experience by predicting ratings, classifying visit mode, and recommending attractions. By robustly merging multi-table data, cleaning and engineering key features, and training baseline models alongside an item-based recommender with cold-start fallbacks, it translates raw records into actionable insights and user-facing value. EDA reveals high rating satisfaction, mode imbalance, geographic hotspots, and seasonality—guiding both modeling choices and product decisions. While the baseline models and similarity approach already provide meaningful outputs, the system is designed for straightforward improvement: richer features (text, recency, pricing), hyperparameter tuning, hybrid content-similarity recommendations, and enhanced app UX (names, images, filters). Overall, the solution establishes a strong foundation for personalized discovery at scale, ready to iterate and deploy.