Project Report: DarshanAI - A Predictive System for Devotee Footfall at Tirumala

Event: Smart India Hackathon 2025

Team: [Your Team Name]

Problem Domain: Crowd Management & Public Service

Executive Summary

The Tirumala Tirupati Devasthanam (TTD) manages one of the largest daily congregations of devotees in the world, facing immense logistical challenges. This report details the development of **"DarshanAI,"** a machine learning system designed to accurately forecast daily devotee footfall. By analyzing historical darshan data, our **Random Forest Regressor** model learns complex temporal patterns to provide reliable predictions. This data-driven approach empowers the TTD administration to move from reactive crowd control to proactive resource management, directly enhancing devotee safety, optimizing operations, and improving the overall pilgrimage experience.

1. The Problem Statement: Managing a City of Faith

The Tirumala temple is a massive logistical operation, managing millions of pilgrims annually. The inability to accurately predict daily attendance leads to several critical issues:

- **Overcrowding:** Sudden surges in devotees can lead to dangerously long queue times, strain on facilities, and potential safety hazards.
- **Resource Wastage:** Misallocation of essential resources like prasadam, water, accommodation, and sanitation services.
- **Suboptimal Staffing:** Inefficient deployment of security, administrative, and sanitation staff, leading to either understaffing on peak days or overstaffing on lean days.

The core problem is the lack of a reliable, data-driven forecasting tool to anticipate devotee flow.

2. Our Solution: The "DarshanAI" Predictive Engine

"DarshanAI" is an end-to-end machine learning pipeline that transforms raw historical data into actionable daily forecasts. It is designed to be a simple yet powerful decision-making tool for the TTD administration.

The system works in two stages, as detailed in our code:

1. **Data Preprocessing (preprocess.py):** The system first cleans and enriches the historical darshan data. It intelligently engineers new features to help the model understand trends and seasonality.
2. **Model Training (train.py):** A Random Forest model is then trained on this enhanced data. It undergoes rigorous hyperparameter tuning and validation to ensure the highest possible accuracy and reliability before being saved as random_forest_darshans.pkl.

3. Technical Methodology

Our solution is built on a robust and transparent technical foundation.

- **Data Source:** We utilized the "Tirumala Tirupati Devasthanam Darshans Dataset," which contains historical daily attendance figures.
- **Feature Engineering:** We didn't just use the dates; we made them smarter.
    - **Lag Features (lag_1, lag_2):** The model is taught to consider the attendance from the previous one and two days, capturing the strong daily momentum in devotee flow.
    - **Cyclic Temporal Encoding:** We converted the day and month into sine/cosine waves. This allows the model to understand that the end of the month is close to the beginning of the next, capturing weekly and monthly seasonality far more effectively than a simple number.
- **Model Selection:** We chose a **Random Forest Regressor** because it is a powerful and interpretable model. It excels at capturing non-linear patterns in the data and provides a clear breakdown of which factors are most important for its predictions.
- **Rigorous Validation:** To prevent inaccurate or overfitted results, we used a TimeSeriesSplit cross-validation strategy. This mimics a real-world scenario by always training the model on past data to predict the future.

4. How "DarshanAI" Solves the Problem (Practical Usefulness)

"DarshanAI" is more than just a model; it's an operational enhancement tool. By providing accurate forecasts, it directly enables:

- **Optimized Crowd Management:** Administrators can anticipate high-traffic days and proactively manage queue systems, open additional darshan lines, and reduce waiting times significantly.
- **Enhanced Security & Safety:** Security personnel can be allocated based on predicted crowd density, ensuring high-risk areas are adequately monitored and preventing potential stampedes or other incidents.
- **Efficient Resource Allocation:** The TTD can precisely manage inventory for prasadam, Annadanam (free meals), and accommodation, reducing waste and ensuring no devotee is left wanting.
- **Improved Devotee Experience:** A smoother, safer, and more organized pilgrimage allows devotees to focus on the spiritual experience, greatly enhancing satisfaction and goodwill.

5. Future Enhancements & Scalability

The current Random Forest model provides a highly accurate baseline. The following enhancements are proposed to evolve this project into a state-of-the-art, fully integrated predictive system.

- **Advanced Modeling Exploration:** To push the boundaries of accuracy, we will experiment with more sophisticated algorithms specifically designed for complex time-series data. This includes:
    - **Gradient Boosting Machines (XGBoost & LightGBM):** These models are often the top performers in tabular data competitions and could capture more intricate patterns than a Random Forest.
    - **Deep Learning Models (LSTMs):** For longer-term forecasting, Long Short-Term Memory networks could be implemented to better understand long-range seasonal dependencies.
- **Deployment & MLOps Pipeline:** To make the model's predictions accessible and maintain their accuracy over time, we will:
    - **Deploy as a REST API:** The saved `random_forest_darshans.pkl` model will be wrapped in an API (using Flask or FastAPI), allowing other applications to easily request forecasts.

    - **Create an Interactive Dashboard:** A user-friendly web interface will be built for TTD officials to view predictions, analyze trends, and assess model performance visually.
    - **Automate Retraining:** An MLOps pipeline will be established to automatically retrain the model on new data periodically, ensuring it never becomes stale and adapts to changing patterns.
- **Full System Integration & Alerting:** The final step is to embed the model's intelligence directly into TTD's operational workflow by:
    - **Integrating the API** with existing systems for staff scheduling, inventory management, and security allocation.
    - **Building an automated alert system** that notifies key personnel via SMS or email when predicted attendance exceeds a predefined critical threshold.