ImpactSense – Earthquake Impact Prediction & Risk Visualization

Project Title & Team Information

- **Project Name**: ImpactSense – Earthquake Impact Prediction & Risk Visualization
- **Team Members:** Data Science Intern Team
- **Project Duration**: 8 Weeks
- **Version**: 1.0
- **Date**: 8th, October 2025

Executive Summary

ImpactSense is an end-to-end machine learning system that predicts earthquake **Damage Potential**, classifies **Risk Category**, and visualizes geographic risks using an interactive web application powered by Streamlit. The deliverables include data preprocessing pipelines, a trained LightGBM model, explainability via SHAP, and a user-friendly interface for both single-event and batch predictions.

1. Business Context & Problem Statement

1.1 Problem Definition

Decision-makers require consolidated earthquake severity indicators beyond magnitude and depth. ImpactSense addresses this by providing a unified **Damage Potential** score and interactive risk visualizations.

1.2 Business Case

- **Stakeholders**: Disaster response agencies, city planners, utilities, insurers
- **Use Cases**: Rapid risk assessment, resource allocation planning, urban resilience strategies
- **ROI**: Enhanced situational awareness, optimized preparedness efforts

1.3 Success Criteria

- Model accuracy (RMSE, R^2) within predefined thresholds
- Sub-second inference latency for single events
- Clear interpretability via feature-attribution
- Intuitive user interface for non-technical users

2. Technical Architecture & System Design

2.1 System Overview

1. **Data Ingestion**: Upload CSV with earthquake events
2. **Preprocessing**: Pipeline for missing value imputation, feature engineering, encoding
3. **Model Inference**: LightGBM for **Damage Potential** prediction
4. **Visualization**: Statistical views, SHAP explainability, risk map

2.2 Technology Stack

- **Frontend**: Streamlit
- **Backend**: Python, scikit-learn, LightGBM
- **Data Processing**: Pandas, NumPy
- **Visualization**: Plotly, Matplotlib, Seaborn
- **Explainability**: SHAP

3. Data Documentation

3.1 Dataset Description

- **Source**: Global seismic event catalog (e.g., USGS/ISC-GEM)
- **Original Records**: 23,412 → **Earthquakes**: 23,229 → **Model Dataset**: 18,583
- **Core Features**: Latitude, Longitude, Depth, Magnitude, RMS, Magnitude Type, Status

3.2 Preprocessing Pipeline

- **Select Features**: 8 columns including RMS
- **Impute RMS**: RandomForest on Latitude, Longitude, Depth, Magnitude
- **Filter Earthquakes**: Remove non-earthquake events
- **Feature Engineering**: `Damage_Potential = 0.6*Magnitude + 0.2*(700-Depth)/700*10`
- **Encode Categoricals**: One-Hot for Magnitude Type, Status

4.1 Problem Formulation

- **Task**: Regression
- **Target**: Damage_Potential
- **Features**: 5 numeric + 11 encoded categorical

4.2 Model Training

- **Algorithm**: LightGBM Regressor (600 trees, 0.05 learning rate)
- **Split**: 80% train / 20% test
- **Evaluation**: RMSE: `0.014` & R^2: `0.999`
- **Artifacts**: `lgb_damage_model.pkl`, feature list

4.3 Explainability

- **Tool**: SHAP TreeExplainer
- **Usage**: Integrated in Predict page expander

5. Application Features & User Guide

5.1 Pages

- **Data**: Summary stats, histograms, correlation heatmap
- **Predict**: Single-event sliders/dropdowns; Damage Potential, Risk Category, Urban Risk Score; SHAP
- **Map**: Mapbox scatter of risk labels and magnitudes

- **About**: Metric definitions and project purpose

5.2 Usage Steps

1. Upload CSV with required columns
2. Explore data distributions on Data page
3. Switch to Predict page for interactive inference
4. View spatial risk patterns on Map

6. Deployment & Maintenance

6.1 Local Setup

- `pip install -r requirements.txt`
- `streamlit run app.py`

6.2 CI/CD & Docker

- Optional Dockerfile for containerized deployment
- GitHub Actions for pipeline validation

6.3 Monitoring & Retraining

- Log predictions and inputs
- Schedule periodic retraining with new earthquake events

7. Future Roadmap

- Integrate real-time USGS API ingestion
- Population-weighted Urban Risk calibration
- Hyperparameter tuning with Optuna
- Mobile-friendly UI extension

Appendix

A. Repository Structure

```
Project
├── app.py
├── train.py
├── data_preprocessing_pipeline.py
├── data_preprocessing.ipynb
├── data/
│   ├── database.csv
│   ├── processed_trained_data.csv
│   ├── train_data.csv
│   └── test_data.csv
└── models/
    ├── data_preprocessing_pipeline.pkl
    └── lgb_damage_model.pkl
```

B. Key Equations

- **Damage Potential**: `0.6*Magnitude + 0.2*(700-Depth)/700*10`
- **Urban Risk Score**: `Damage_Potential * (1 + (|Latitude|+|Longitude|)/360)`

Generated by ImpactSense team on October 09, 2025