Project Report: Flood Prediction Tool

1. Summary

This report outlines the end-to-end development of a machine learning model designed to predict flood probability based on a range of environmental and infrastructural factors. The project successfully progressed from initial data exploration and feature engineering to model training, evaluation, and final deployment as an interactive web application using Streamlit. The final model, a fine-tuned LightGBM Regressor, achieved a high-performance metric with an **R-squared score of approximately 0.94**, indicating its strong predictive capability. The project reached a climax in a user-friendly web app that allows for real-time risk assessment by adjusting input parameters.

2. Introduction

2.1. Project Objective

The primary objective of this project was to develop a robust regression model to accurately predict the likelihood of flooding. A secondary goal was to make this model accessible to non-technical users by deploying it as an interactive web application, thereby translating complex data science into a practical tool for risk assessment.

2.2. Scope

The project scope included:

- Analysis of the "Flood Prediction Dataset" from Kaggle.
- Exploratory Data Analysis (EDA) to understand feature relationships.
- Feature engineering to create more insightful variables.
- Training and evaluation of multiple machine learning models (Linear Regression, Random Forest, XGBoost, LightGBM).
- Hyperparameter tuning of the best-performing model.
- Saving the final model and deploying it using Streamlit.

3. Methodology

3.1. Data Collection

The project utilized the **Flood Prediction Dataset**, which contains 21 columns. Twenty of these are independent features representing factors like MonsoonIntensity, Deforestation, and DamsQuality, while the target variable, FloodProbability, is a continuous value representing the likelihood of a flood.

3.2. Exploratory Data Analysis (EDA)

An initial analysis was conducted to understand the dataset's structure. A key step was generating a correlation matrix to investigate potential multicollinearity among the features.

The analysis revealed that the raw features were almost perfectly independent of one another (correlation values near 0.00). This indicated that multicollinearity was not a concern for simpler models like Linear Regression but also suggested that more complex, non-linear relationships might exist, which advanced models could capture more effectively.

3.3. Feature Engineering

To enhance the model's predictive power, two new interaction features were engineered based on logical assumptions:

1. **LandslideRisk**: Created by summing TopographyDrainage and Deforestation. The rationale is that poor topography combined with a lack of forest cover significantly increases landslide risk, which can contribute to flooding.
2. **InadequateInfrastructure**: Created by summing DeterioratingInfrastructure and DrainageSystems. This feature represents the combined risk from failing public works and poor water management systems.

After creating these two features, the four original columns were dropped to avoid data redundancy, resulting in a final feature set of 18 variables for the model.

4. Model Development and Selection

A series of models were trained and evaluated to identify the best performer.

1. **Linear Regression**: Served as a baseline model. It achieved a respectable R-squared score but was limited by its inability to capture non-linear patterns.
2. **Random Forest**: Showed a significant improvement over Linear Regression, demonstrating its ability to handle more complex relationships. Initial tuning brought its R-squared score to ~0.73.
3. **XGBoost & LightGBM**: These advanced gradient boosting models were tested on the engineered feature set. Both performed exceptionally well, with LightGBM showing a slight edge in both accuracy and training speed.
   - **XGBoost R^2 Score**: ~0.934
   - **LightGBM R^2 Score**: ~0.939

4.1. Hyperparameter Tuning

The winning model, **LightGBM**, was further optimized using GridSearchCV. The tuning process searched for the best combination of n_estimators, learning_rate, and num_leaves. The optimal parameters were not identified so used from default parameters

5. Results and Evaluation

The final, tuned LightGBM model, trained on the 18 engineered features, yielded the following outstanding results:

- **Final R-squared (R^2) Score**: **~0.94**
- **Final Mean Squared Error (MSE)**: **~0.00015**

An R-squared score of 0.94 signifies that the model can explain **94% of the variance** in the flood probability data. This is an excellent result, indicating a highly accurate and reliable model suitable for deployment.

6. Deployment

The final trained LightGBM model was saved into a lgbm_flood_prediction_model.joblib file. This model was then deployed as an interactive web application using the **Streamlit** framework.

6.1. Web Application (app.py)

The application features:

- A user-friendly sidebar with sliders for all 20 original input features.
- Backend logic that takes the user's input, performs the exact same feature engineering steps used during training, and feeds the final 18 features into the loaded model.
- A clear display of the predicted flood probability, color-coded for low, moderate, and high risk.

6.2. Deployment Challenges and Solutions

- **Large Model File**: The saved model file exceeded GitHub's 100 MB limit. This was resolved by implementing **Git Large File Storage (LFS)** to handle the large asset.
- **Environment Incompatibility**: Initial deployment on Streamlit Cloud failed due to version conflicts between the local training environment (Python, scikit-learn) and the deployment environment. This was definitively solved by creating a .python-version file to force Streamlit to use the same Python version as the local machine, ensuring perfect compatibility.

7. Conclusion and Future Work

7.1. Conclusion

This project successfully achieved its objectives by developing a high-accuracy flood prediction model and deploying it as a practical, interactive tool. The process highlighted the importance of systematic model evaluation, the significant impact of feature engineering, and the critical need for environment management in deploying machine learning applications.

7.2. Future Work

- **Incorporate Real-Time Data**: Integrate APIs for live weather or river level data to provide predictions based on current conditions.
- **Advanced Feature Engineering**: Explore more complex feature interactions or polynomial features.
- **Geospatial Analysis**: Add a map-based interface where users can select a location to automatically populate relevant geographical features.
- **Model Monitoring**: Implement a system to monitor the model's performance over time and retrain it as new data becomes available.