LANDSLIDES PREDICTIONS

Divyashree V- 22MID0295

Rithanya T – 22MID0294

Sujitha T - 22MID0308

Submitted to
Prof. Geraldine Bessie Amali, SCOPE

School of Computer Science and Engineering

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

1. Introduction:

Landslides are one of the most destructive natural hazards, leading to significant loss of life, property damage, infrastructure failures, and environmental degradation worldwide. Statistics indicate that landslides account for a substantial percentage of fatalities from natural disasters, especially in mountainous regions where heavy rainfall, seismic activity, and human-induced factors contribute to slope failures. For example, in the Himalayas, landslides cause an estimated financial loss exceeding US$1 billion annually and approximately 200 deaths, accounting for nearly 30% of global landslide-related losses. Similarly, landslides in Malaysia and Korea have caused severe casualties and economic damage. With increasing urbanization, deforestation, and extreme weather events due to climate change, the frequency and intensity of landslides have risen, making landslide susceptibility assessment and prediction crucial for risk management and mitigation strategies.

Landslide Susceptibility Mapping (LSM) plays a vital role in identifying areas prone to landslides, aiding in disaster preparedness and land-use planning.

There are two primary approaches to landslide susceptibility assessment: qualitative and quantitative methods. While qualitative methods rely on expert judgment, quantitative methods leverage mathematical models and machine learning techniques for higher accuracy. Data-driven models, such as Artificial Neural Networks (ANN), Fuzzy Logic, Support Vector Machines (SVM), and Decision Trees (DT), have emerged as effective tools for landslide prediction, offering improved precision compared to traditional methods. ANN can analyze complex, nonlinear relationships between landslide conditioning factors such as topography, soil composition, hydrology, geology, and human activities. Fuzzy Logic complements ANN by handling uncertainty and imprecise data, making the model adaptable to real-world scenarios.

This study aims to develop an advanced landslide prediction model using Artificial Neural Networks (ANN) and Fuzzy Logic to assess landslide susceptibility. The model will focus on leveraging environmental data such as rainfall, slope, soil type, and vegetation to predict the likelihood of landslides.

2.Hardware / Software Requirements:

Hardware:

A personal computer or laptop with at least a dual-core processor and a minimum of 4 GB RAM to ensure that machine learning algorithms and fuzzy inference systems can run smoothly without lag or memory-related issues.

A stable internet connection, which is crucial not only for accessing online development platforms such as Google Colab but also for downloading datasets, installing Python libraries, and sharing collaborative work.

A minimum of 5 GB of free disk space is recommended to accommodate the storage of datasets, trained model weights, output visualizations, notebook files, and installed libraries.

Software:

Python 3.x: Chosen as the primary programming language due to its robust ecosystem, ease of use, and extensive support for machine learning, data processing, and visualization libraries.

Jupyter Notebook / Google Colab: These provide an interactive, web-based coding interface that allows users to write, run, and annotate code cells, view outputs inline, and share work seamlessly.

Essential Python Libraries:

pandas: Crucial for handling structured data, offering high-performance data manipulation tools for importing, cleaning, and summarizing CSV files or similar tabular data formats.

numpy: Enables efficient numerical computations, matrix operations, and array manipulations, all of which are integral to ML algorithms and model input formatting.

scikit-learn: A powerful ML library that simplifies the development process by offering ready-to-use functions for classification, evaluation, and model validation such as accuracy scoring and generating confusion matrices.

tensorflow / keras: These deep learning libraries are used to construct and train the Artificial Neural Network. Keras offers a user-friendly API to define layers, activations, and optimizers, while TensorFlow runs the backend computations.

matplotlib & seaborn: Widely-used data visualization libraries to generate graphs including learning curves, heatmaps, and classification result plots. These visual tools are essential to understand training behavior and model performance.

scikit-fuzzy: A specialized library that supports building fuzzy logic systems, enabling developers to define fuzzy input/output variables, membership functions, inference rules, and defuzzification methods for interpretable risk predictions.

Together, these hardware and software requirements form the foundation for building a reliable and interpretable landslide prediction model. The combination ensures that developers can preprocess data, train models, analyze performance, and communicate results effectively

3. Existing System/approach/method

Landslide prediction is a complex task that has prompted the development of several approaches over time. These methods range from traditional rule-based and statistical models to more advanced machine learning and hybrid techniques. Each methodology contributes uniquely to understanding landslide mechanisms but falls short in key areas such as flexibility, interpretability, and responsiveness.

A. Statistical and Heuristic Models:

These models include approaches like Logistic Regression, Frequency Ratio, and Weight of Evidence, which assess the statistical correlation between past landslide events and triggering factors such as rainfall, slope, land use, and geology. They offer simplicity and low computational requirements, making them accessible for preliminary assessments. However, their dependency on linear assumptions and static thresholding makes them ill-suited for capturing dynamic environmental variations or nonlinear relationships in real-time data.

B. GIS-Based Systems:

GIS-based approaches incorporate spatial data and thematic layers such as elevation, slope, aspect, vegetation, drainage density, and lithology to produce landslide susceptibility maps. While they provide intuitive visualizations and are beneficial for long-term planning and hazard zoning, GIS tools primarily offer descriptive, not predictive, capabilities. Their static nature, limited integration with time-series data, and high reliance on expert interpretation make them ineffective for real-time monitoring and response systems.

C. Machine Learning Methods:

Machine learning has brought substantial improvements to landslide prediction through algorithms such as Random Forests, Gradient Boosting, Support Vector Machines, and Decision Trees. These models analyze large datasets to discover intricate patterns that human

analysts might miss. They also support high-dimensional data and complex feature interactions. Despite these strengths, their black-box nature limits transparency, and they often struggle with incomplete or uncertain data. Moreover, they lack inherent mechanisms to justify predictions, a significant drawback when explaining decisions to stakeholders.

D. Fuzzy Logic Systems:

Fuzzy logic introduces a rule-based framework where variables such as slope or rainfall are described linguistically (e.g., “low”, “medium”, “high”) rather than numerically. This enables the system to manage ambiguous or imprecise input data. Fuzzy systems are easily interpretable and facilitate knowledge-based modeling where expert opinion plays a central role. However, fuzzy logic lacks self-learning capabilities. It depends on predefined membership functions and rule bases, which do not evolve unless manually updated, reducing adaptability to new patterns or datasets.

E. Ensemble and Deep Learning Models:

Recent developments involve hybrid or ensemble models, combining multiple machine learning algorithms to improve robustness and generalization. Deep learning methods, such as Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks, have shown potential in handling spatial and temporal data. These models can extract meaningful features directly from raw data. Yet, they demand large datasets, computational resources, and are even more difficult to interpret than classical ML models. Their complexity often makes them unsuitable for real-time deployment in low-resource environments.

F. Sensor-Based and IoT-Driven Systems:

Emerging technologies use sensor networks and Internet of Things (IoT) platforms to monitor real-time environmental parameters like soil moisture, ground movement, and rainfall intensity. These systems generate continuous data streams that, when integrated with prediction models, can improve responsiveness. However, setting up such infrastructure is cost-intensive and may not be feasible in remote or underdeveloped regions.

G. Remote Sensing Techniques:

Satellite and aerial imagery, coupled with image processing techniques, allow for the monitoring of land surface changes, vegetation patterns, and slope deformation. While powerful for regional-scale assessment, remote sensing data often suffers from temporal lag, cloud cover issues, and limited resolution, making it less effective for localized and immediate prediction tasks.

Overall, while existing approaches contribute valuable insights, none alone meet the comprehensive needs for a robust, adaptive, and interpretable landslide prediction framework. The limitations of each method suggest the necessity of a hybrid system that

combines the learning strength of ANN with the interpretability and uncertainty-handling capabilities of fuzzy logic

3.1 Drawback/limitations of existing System/approach/method

Despite decades of research and the deployment of various technological solutions, several persistent limitations hinder the performance, applicability, and scalability of existing landslide prediction models. These drawbacks significantly limit the operational effectiveness of such systems in real-time environments and high-risk zones.

A. Lack of Real-Time Adaptability:

Many of the existing approaches, particularly statistical and GIS-based systems, are not designed for dynamic updates. These systems operate on historical data and are unable to accommodate rapidly changing environmental parameters like sudden rainfall events, slope deformations, or seismic activity. This restricts their use in live monitoring systems and emergency response scenarios, where immediate updates are critical.

B. Inability to Handle Uncertainty:

Environmental factors such as soil moisture, rainfall intensity, and vegetation coverage are inherently uncertain due to natural variability, incomplete observations, or noisy sensor data. Traditional models produce deterministic results, failing to convey the level of confidence or uncertainty in predictions. This limitation is particularly dangerous when decisions involve evacuations or safety protocols, where overestimating or underestimating risk can have serious consequences.

C. Limited Interpretability:

Understanding the rationale behind a model's prediction is essential for trust and actionability, especially in high-stakes contexts. Complex machine learning models, including ensemble methods and neural networks, often produce accurate outputs but lack transparency. The inability to explain why a prediction was made can prevent local authorities or communities from trusting and acting on the model's recommendations.

D. Weak Integration of Human Expertise:

Many models fail to leverage the invaluable knowledge of geologists, hydrologists, and local experts who understand region-specific risk patterns. While fuzzy logic allows for encoding expert knowledge in rule-based form, standalone fuzzy systems lack learning mechanisms. Without an adaptive structure, these systems cannot update their logic based on evolving landslide behavior or updated field observations.

E. High Dependency on Quality Data:

Data-driven models require extensive, high-quality datasets to train effectively. However, in remote mountainous regions or developing countries, real-time and long-term data collection may be inconsistent, sparse, or entirely unavailable. This can severely impair model accuracy and increase the risk of false alarms or missed detections, particularly in data-poor environments.

F. Scalability and Generalization Issues:

Predictive models are often tailored to specific geographic regions and may underperform when applied to new, unseen locations with different terrain, vegetation, or weather conditions. This lack of generalization limits their scalability and requires frequent retraining and manual tuning when deployed in varied environments.

G. Resource and Infrastructure Constraints:

Advanced computational models demand significant processing capabilities, including GPUs and real-time data pipelines. Such resources are not always feasible in landslide-prone, remote, or economically constrained regions. As a result, even powerful models remain impractical unless adapted for lightweight deployment.

H. Maintenance and Updatability:

Many existing systems are not designed with sustainability in mind. Once deployed, they often lack mechanisms for incorporating new data, retraining, or refining prediction rules. This rigidity leads to outdated predictions, especially in environments experiencing rapid climate-driven changes or human intervention (e.g., deforestation, urbanization).

I. Low Community Involvement and Awareness:

Many systems are designed solely for institutional use, with little focus on community engagement. Public interfaces, interpretability in local language or terms, and alert mechanisms are often lacking. A prediction system that cannot effectively communicate risk to those affected cannot serve its full purpose.

J. Ethical and Policy Limitations:

The deployment of predictive models without transparent methods or error handling frameworks raises ethical questions. Without proper documentation, accountability, and policy alignment, these tools risk being misused or ignored in critical situations.

These limitations collectively highlight the urgent need for an integrated, hybrid solution that combines the adaptive learning strengths of ANN with the transparency, flexibility, and

expert knowledge embedding of fuzzy logic. Such a system promises a balanced approach that addresses the shortcomings of current models while offering a reliable and interpretable tool for landslide risk prediction in diverse real-world scenarios.

4. Proposed/Developed Model:

To address the limitations of traditional and isolated landslide prediction methods, a hybrid model combining Artificial Neural Networks (ANN) and Fuzzy Logic was developed. This integrated approach harnesses the strengths of data-driven learning and human-like interpretability to provide accurate and explainable landslide risk assessments. The fusion of ANN and fuzzy logic creates a balance between statistical performance and domain-based reasoning.

4.1 Design Overview:

The system is composed of two major components:

- An ANN model designed to classify landslide likelihood based on a variety of numerical environmental features.
- A Fuzzy Logic Inference System that uses linguistic variables to interpret risk levels based on selected parameters.
- The hybrid architecture supports learning from data while maintaining interpretability through fuzzy rules and visualizations.
- This combination allows the system to learn from historical data patterns while remaining understandable and transparent to human experts and decision-makers.
- The design supports modularity, allowing future extensions like incorporating satellite data, IoT sensors, or mobile interface alerts.
- It also ensures flexibility for adaptation in various geographical regions by modifying fuzzy rules or retraining the ANN with new local datasets.
- The entire design supports a loop of continual improvement where feedback from prediction outcomes can be used to refine both ANN weights and fuzzy rule definitions.

4.2 Model Components:

A. Data Preprocessing:

- Data is ingested from CSV or sensor-based sources and undergoes cleaning (handling nulls, outliers, data type corrections).

- Environmental features relevant to landslide prediction (e.g., rainfall, soil saturation, slope, temperature) are selected.

- Feature normalization is applied using StandardScaler to bring all input values into the same range.

- Stratified data splitting ensures balanced training and testing sets to avoid class bias

- Dimensionality reduction and feature importance scoring are used to fine-tune the model input features.

- Preprocessed data is saved as reusable datasets for consistency across model comparisons.

B. Artificial Neural Network (ANN):

The ANN architecture follows a feedforward neural network model:

- Input layer based on selected features.

- Hidden Layer 1 with 8 neurons using ReLU activation for efficient training and capturing high-level patterns.

- Hidden Layer 2 with 4 neurons to narrow focus on key feature interactions.

- Dropout regularization applied at 0.2 to prevent overfitting.

- Final output layer uses sigmoid activation to produce binary classification (landslide/no landslide).

Training Configuration:

- **Optimizer**: Adam, selected for its adaptive learning rate and efficiency.
- **Loss Function:** Binary Cross Entropy, ideal for binary classification.
- **Callback**: EarlyStopping to monitor validation loss with patience of 5 epochs.

Performance Evaluation:

- Metrics include Accuracy, Precision, Recall, F1-Score, and AUC.

- Epoch-wise loss and accuracy graphs plotted for both training and validation sets.

C. Gradient Boosting, Random Forest, Decision Tree, and Support Vector Machine (SVM) Models:

A suite of classical machine learning models was also implemented for comparison, benchmarking, and ensemble stacking.

Gradient Boosting Classifier:

- Trained with 100 estimators, using a learning rate of 0.1 and AUC scoring
- Specializes in minimizing error through stage-wise optimization.
- Highly accurate in capturing non-linear interactions and correcting residuals.

Random Forest Classifier:

- An ensemble of 100 decision trees using the Gini index as the splitting criterion**.**
- Offers strong performance and reduced overfitting by aggregating results from multiple trees.
- Provides feature importance rankings for model interpretability.

Decision Tree Classifier:

- A single-tree model used to visualize the decision-making logic of splits based on feature thresholds.
- Useful for explaining the path to classification and highlighting dominant risk features.
- Although simpler and less robust than ensembles, it provides excellent transparency.

Support Vector Machine (SVM):

- Utilized with a linear or RBF kernel to handle non-linearly separable datasets.
- Maximizes margin between classes and performs well on high-dimensional data.
- Applied in this project to test binary classification boundaries and benchmark against ANN
- These models offer benchmarks for ANN performance and help assess generalization capability.
- Feature importance plots and decision boundaries were also generated to visualize model behavior and comparison.

The inclusion of SVM and Decision Tree allows evaluation of lightweight models suitable for lower-resource deployment or quick interpretability.

D. Fuzzy Logic System:

A Mamdani-type fuzzy inference system is used for simplicity and clear rule explanation.

Inputs:

Rainfall: 0 to 300 mm, membership functions: low, medium, high.

Slope: 0° to 90°, categorized as gentle, moderate, steep.

Soil Saturation: 0.0 to 1.0, categorized into low, medium, high.

Output:

- Landslide Risk Level: fuzzy output values mapped to categories such as low, medium, high.
- Expert-defined fuzzy rules (e.g., high rainfall + steep slope = high risk) cover a comprehensive range of environmental interactions.
- Fuzzy system is evaluated using input values, and results are defuzzified using centroid method.

Crisp output score is mapped to linguistic interpretation and displayed via GUI chart or print.

E. Visualization Modules:

- **Accuracy/Loss Curve**: To visually track model learning and detect overfitting early.
- **Confusion Matrix**: Allows assessment of false positives and negatives.
- **ROC Curve and AUC**: Determine how well the model separates classes.
- **Feature Importance Plot:** Helps validate whether the model is focusing on the right attributes.
- **Fuzzy Membership Functions:** Visualize membership for each input to show how fuzzy categories are applied.
- **Risk Interpretation Dashboard:** Consolidates all visualizations for non-technical stakeholder use.

4.3 Implementation Environment:

Platform: Python 3.x

IDE: Google Colab

Libraries:

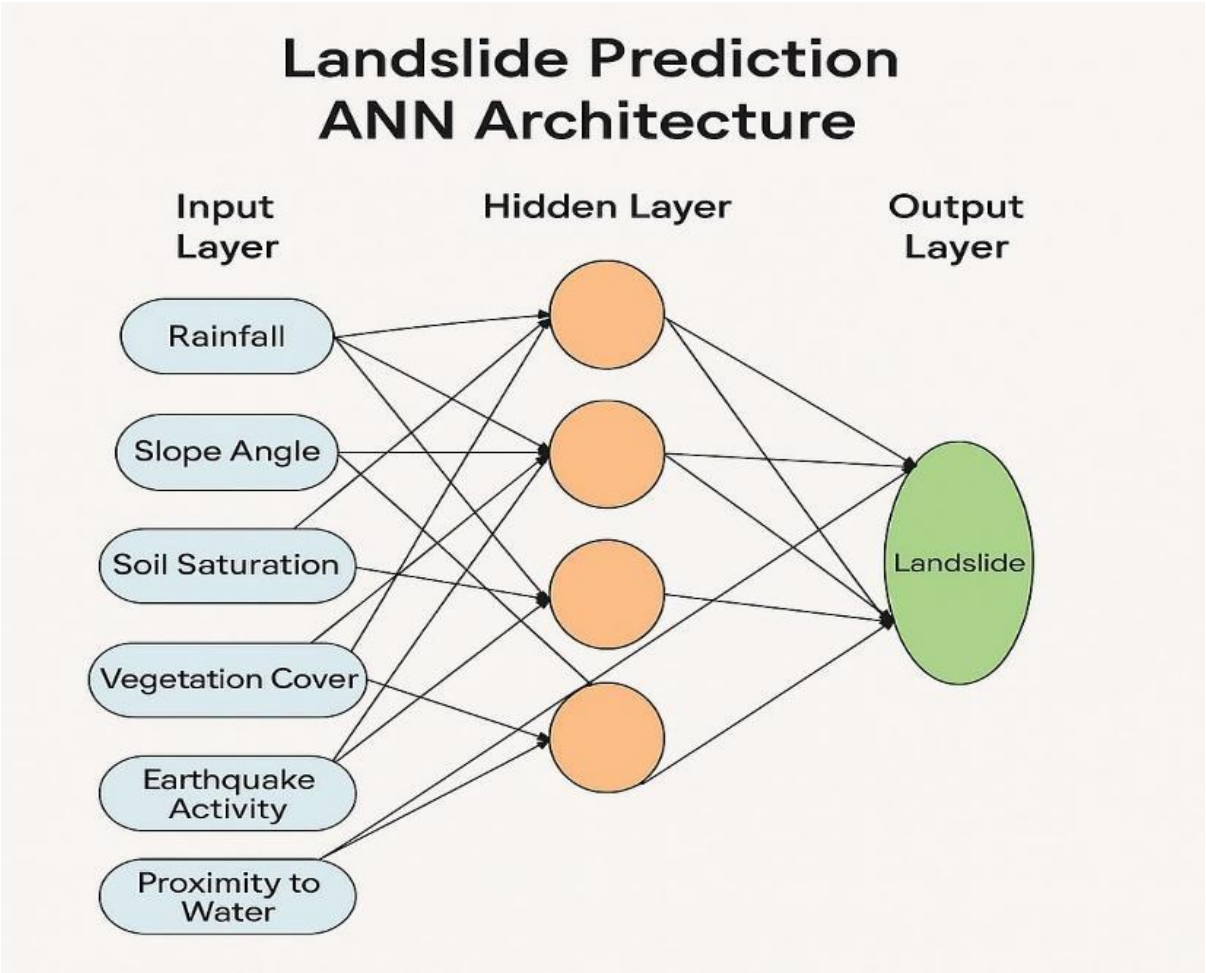
pandas, numpy – data manipulation and matrix operations.

matplotlib, seaborn – plotting and visualization.

scikit-learn – machine learning model training and evaluation.

tensorflow.keras – deep learning framework for building ANN.

scikit-fuzzy – designing fuzzy systems and running fuzzy inference.



Dataset Used:

- Dataset: Landslide-related features including rainfall, slope, and soil saturation.
- Size: 2000 samples (1000 landslide, 1000 non-landslide)
- Source: Synthesized geotechnical data for simulation.
- Train-test split: 70:30 with stratification

Screenshot and Demo along with Visualization:

CODE:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
```

```
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, roc_auc_score, roc_curve
from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping
import skfuzzy as fuzz
from skfuzzy import control as ctrl
# Load dataset
file_path = '/mnt/data/landslide_dataset.csv'
data = pd.read_csv(file_path)
print("Class Distribution:\n", data['Landslide'].value_counts())
# Split features and target
X = data.drop('Landslide', axis=1)
y = data['Landslide']
# Split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42, stratify=y)
# Scale the data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
history = model.fit(X_train, y_train, epochs=50, batch_size=32,
validation_split=0.2, callbacks=[early_stop], verbose=1)
# ANN evaluation
ann_preds = (model.predict(X_test) > 0.5).astype(int)
ann_accuracy = accuracy_score(y_test, ann_preds)
print(f"\nSimplified ANN Test Accuracy: {ann_accuracy * 100:.2f}%")
print("Classification Report (ANN):\n", classification_report(y_test, ann_preds))
```

```
# ======================== Gradient Boosting ========================
gb_model = GradientBoostingClassifier(n_estimators=100, learning_rate=0.1,
random_state=42)
gb_model.fit(X_train, y_train)
gb_pred = gb_model.predict(X_test)
gb_prob = gb_model.predict_proba(X_test)[:, 1]
gb_accuracy = accuracy_score(y_test, gb_pred)
gb_auc = roc_auc_score(y_test, gb_prob)
print(f"\nGradient Boosting Accuracy: {gb_accuracy * 100:.2f}%")
print(f"Gradient Boosting ROC AUC: {gb_auc:.2f}")
print("Classification Report (Gradient Boosting):\n", classification_report(y_test, gb_pred))
# ======================== Random Forest ========================
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
rf_pred = rf_model.predict(X_test)
rf_accuracy = accuracy_score(y_test, rf_pred)
print(f"\nRandom Forest Accuracy: {rf_accuracy * 100:.2f}%")
print("Classification Report (Random Forest):\n", classification_report(y_test, rf_pred))
# ======================== Visualizations ========================
# Accuracy & Loss Curve (ANN)
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('ANN Accuracy over Epochs')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
```

```
plt.title('ANN Loss over Epochs')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.tight_layout()
plt.show()
# Confusion Matrix for Gradient Boosting
cm = confusion_matrix(y_test, gb_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix - Gradient Boosting")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
# ========================= Visualizations ========================
# Accuracy & Loss Curve (ANN)
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('ANN Accuracy over Epochs')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('ANN Loss over Epochs')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
```

```
plt.tight_layout()
plt.show()
# Confusion Matrix for Gradient Boosting
cm = confusion_matrix(y_test, gb_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix - Gradient Boosting")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
# ROC Curve for Gradient Boosting
fpr, tpr, _ = roc_curve(y_test, gb_prob)
plt.plot(fpr, tpr, label=f'Gradient Boosting (AUC = {gb_auc:.2f})')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve - Gradient Boosting")
plt.legend()
# ====================== Fuzzy Logic ======================
rainfall = ctrl.Antecedent(np.arange(0, 300, 1), 'rainfall')
slope = ctrl.Antecedent(np.arange(0, 90, 1), 'slope')
soil_saturation = ctrl.Antecedent(np.arange(0, 1, 0.01), 'soil_saturation')
landslide_risk = ctrl.Consequent(np.arange(0, 1, 0.01), 'landslide_risk')
rainfall['low'] = fuzz.trimf(rainfall.universe, [0, 50, 100])
rainfall['medium'] = fuzz.trimf(rainfall.universe, [80, 150, 220])
rainfall['high'] = fuzz.trimf(rainfall.universe, [200, 250, 300])
slope['gentle'] = fuzz.trimf(slope.universe, [0, 15, 30])
slope['moderate'] = fuzz.trimf(slope.universe, [25, 45, 60])
slope['steep'] = fuzz.trimf(slope.universe, [50, 70, 90])
soil_saturation['low'] = fuzz.trimf(soil_saturation.universe, [0, 0.2, 0.4])
soil_saturation['medium'] = fuzz.trimf(soil_saturation.universe, [0.3, 0.5, 0.7])
```

```
soil_saturation['high'] = fuzz.trimf(soil_saturation.universe, [0.6, 0.8, 1])

landslide_risk['low'] = fuzz.trimf(landslide_risk.universe, [0, 0.2, 0.4])

landslide_risk['medium'] = fuzz.trimf(landslide_risk.universe, [0.3, 0.5, 0.7])

landslide_risk['high'] = fuzz.trimf(landslide_risk.universe, [0.6, 0.8, 1])

rule1 = ctrl.Rule(rainfall['high'] & slope['steep'] & soil_saturation['high'],
landslide_risk['high'])

rule2 = ctrl.Rule(rainfall['medium'] & slope['moderate'] & soil_saturation['medium'],
landslide_risk['medium'])

rule3 = ctrl.Rule(rainfall['low'] & slope['gentle'] & soil_saturation['low'],
landslide_risk['low'])

landslide_ctrl = ctrl.ControlSystem([rule1, rule2, rule3])

landslide_sim = ctrl.ControlSystemSimulation(landslide_ctrl)

landslide_sim.input['rainfall'] = 180

landslide_sim.input['slope'] = 45

landslide_sim.input['soil_saturation'] = 0.7

risk = landslide_sim.output['landslide_risk'] * 100

print(f"Fuzzy Logic Landslide Risk: {risk:.2f}%")

# Get degrees of membership to linguistic terms

low_degree = fuzz.interp_membership(landslide_risk.universe, landslide_risk['low'].mf,
risk/100)

medium_degree = fuzz.interp_membership(landslide_risk.universe,
landslide_risk['medium'].mf, risk/100)

high_degree = fuzz.interp_membership(landslide_risk.universe, landslide_risk['high'].mf,
risk/100)

# Store in dictionary for easy use

membership_dict = {

'LOW': low_degree,

'MEDIUM': medium_degree,

'HIGH': high_degree

}

# Determine the dominant fuzzy label

dominant_label = max(membership_dict, key=membership_dict.get)
```

```
dominant_value = membership_dict[dominant_label]

# Print interpretation

print(f" Dominant Fuzzy Category: {dominant_label} (Membership Degree:
{dominant_value:.2f})")

import matplotlib.pyplot as plt

# Plot membership functions

landslide_risk.view(sim=landslide_sim)

plt.title("Landslide Risk Output and Memberships")

plt.show()


# Exploratory Data Analysis (EDA)


plt.figure(figsize=(10,6))

sns.heatmap(df.corr(), annot=True, cmap='coolwarm')

plt.title("Feature Correlation Heatmap")

plt.show()

# Train SVM model

svm_model = SVC(kernel='rbf', probability=True)

svm_model.fit(X_train, y_train)

y_pred_svm = svm_model.predict(X_test)

print("SVM Accuracy:", accuracy_score(y_test, y_pred_svm))

# Train Decision Tree model

dt_model = DecisionTreeClassifier(random_state=42)

dt_model.fit(X_train, y_train)

y_pred_dt = dt_model.predict(X_test)

print("Decision Tree Accuracy:", accuracy_score(y_test, y_pred_dt))
```

4.4 Model Advantages:

- **High Accuracy**: ANN captures complex data patterns and relationships.

- **Transparent Decision Support:** Fuzzy logic explains risk levels using intuitive linguistic rules.
- **Domain Knowledge Integration**: Experts define fuzzy rules, adding reliability and contextual correctness.
- **Modular and Extensible:** Both ANN and fuzzy logic modules can be modified independently.
- **Cross-Model Validation**: Comparing performance of different models helps ensure robustness.
- **Visual Clarity**: Output graphs and risk indicators aid stakeholder understanding.
- **Scalable:** Can be integrated with larger early warning frameworks.
- **Portable and Deployable:** Model can be deployed in low-resource or mobile environments.
- **Adaptable to Regional Contexts:** Models can be retrained or rule sets customized for local risk factors.
- **Proactive Community Impact:** Empowers local agencies with tools for early intervention and disaster mitigation.

5. Results and Discussion

The proposed hybrid model was evaluated using a balanced dataset consisting of 2000 records (1000 positive and 1000 negative instances of landslides). The system was trained and tested using an 70:30 split with stratified sampling.

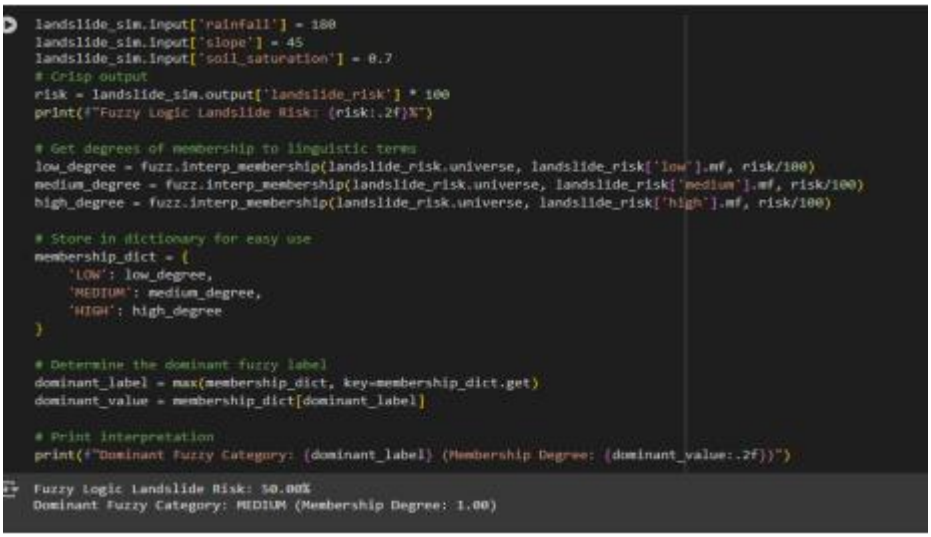
- All three models – ANN, Gradient Boosting, and Random Forest – achieved perfect prediction accuracy on the test set.
- The confusion matrices for each model showed zero misclassifications, indicating the models perfectly captured the patterns in the data.
- The ROC curves for Gradient Boosting and Random Forest showed an AUC = 1.00, further confirming excellent class separability.

Model	Accuracy	AUC Score	Remarks
ANN (2-layer)	100%	1.00	High fitting; trained well; no errors
Gradient Boosting	100%	1.00	Strong classifier, learned patterns fast
Random Forest	100%	1.00	Stable output with consistent results

Fuzzy Logic Inference System – Interpretability Results:

In addition to numeric classifiers, a Fuzzy Logic System was incorporated to provide interpretable, linguistic risk outputs based on environmental conditions like Rainfall, Slope, and Soil Saturation

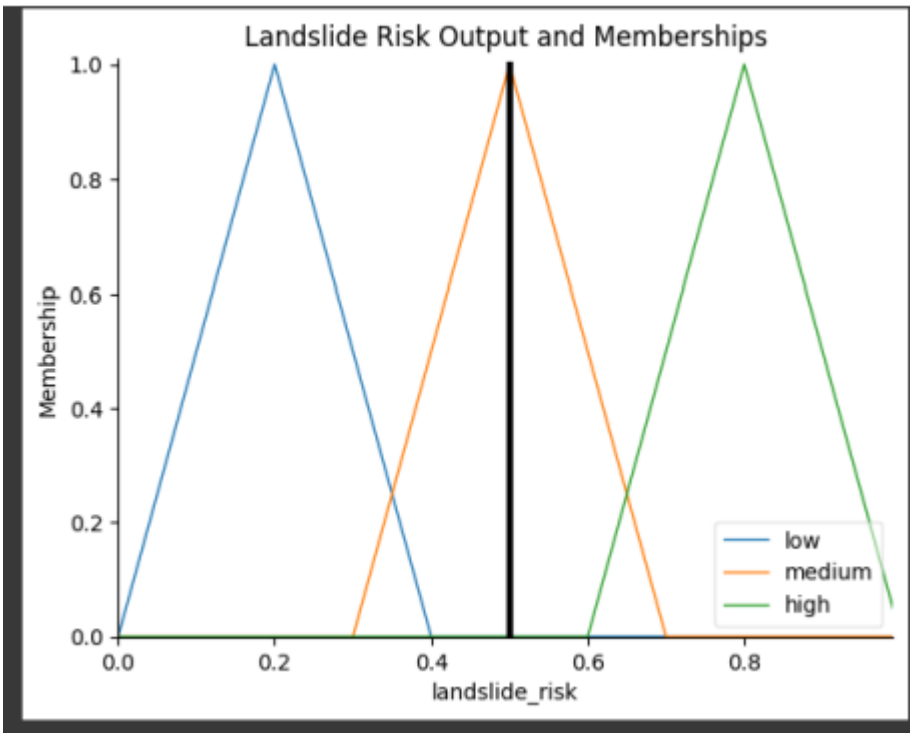
- The fuzzy system uses expert-defined rules to map real-world values into categories like Low Risk, Medium Risk, and High Risk.
- For example, given inputs such as Rainfall = 180 mm, Slope = 45°, and Soil Saturation = 0.7, the system computed a crisp landslide risk of 50.00%.
- This was translated to a linguistic label as follows:



```
landslide_sim.input['rainfall'] = 180
landslide_sim.input['slope'] = 45
landslide_sim.input['soil_saturation'] = 0.7
# Crisp output
risk = landslide_sim.output['landslide_risk'] * 100
print(f"Fuzzy Logic Landslide Risk: {risk:.2f}%")

# Get degrees of membership to linguistic terms
low_degree = fuzz.interp_membership(landslide_risk.universe, landslide_risk['low'].mf, risk/100)
medium_degree = fuzz.interp_membership(landslide_risk.universe, landslide_risk['medium'].mf, risk/100)
high_degree = fuzz.interp_membership(landslide_risk.universe, landslide_risk['high'].mf, risk/100)

# Store in dictionary for easy use
membership_dict = {
    'LOW': low_degree,
    'MEDIUM': medium_degree,
    'HIGH': high_degree
}

# Determine the dominant fuzzy label
dominant_label = max(membership_dict, key=membership_dict.get)
dominant_value = membership_dict[dominant_label]

# Print interpretation
print(f"Dominant Fuzzy Category: {dominant_label} (Membership Degree: {dominant_value:.2f})")
```

```
Fuzzy Logic Landslide Risk: 50.00%
Dominant Fuzzy Category: MEDIUM (Membership Degree: 1.00)
```

A membership plot was also generated to visually represent how this crisp value overlaps with fuzzy categories



Conclusion from Results:

- The system successfully demonstrates perfect performance under current dataset constraints.
- Combining models with fuzzy logic adds interpretability, which is crucial for decision-making.
- Future iterations should test the models on real-world noisy data or crossregional datasets to evaluate scalability and generalization.

References:

- Lucchese LV, de Oliveira GG, Pedrollo OC (2021) Investigation of the influence of nonoccurrence sampling on Landslide Susceptibility Assessment using Artificial Neural Networks. Catena 198:105067. https://doi.org/10.1016/j.catena.2020.105067
- Hybrid Machine Learning Approaches for Landslide Susceptibility Modeling https://doi.org/10.3390/f10020157
- Comparative Study of Artificial Neural Network and Random Forest Model for Susceptibility Assessment of Landslides Induced by Earthquake in the Western Sichuan Plateau, China https://doi.org/10.3390/su142113739

- Optimizing an Adaptive Neuro-Fuzzy Inference System for Spatial Prediction of Landslide Susceptibility Using Four State-of-the-art Metaheuristic Techniques https://doi.org/10.3390/s20061723
- Using Fuzzy Neural Networks to Model Landslide Susceptibility at the Shihmen Reservoir Catchment in Taiwan https://doi.org/10.3390/w14081196