**Lab 6 – Manual**

**Problem Statement:**

Landslides are a major natural hazard, often triggered by environmental factors such as rainfall, slope angle, and soil moisture. In this lab, you will implement a **Bayesian classifier** to predict landslide occurrences using the **Global Landslide Dataset**.

Bayesian classification is a powerful approach for predicting landslides, leveraging probabilistic models to handle uncertainty and incorporate prior knowledge. Here are some key aspects and recent advancements in this field:

1. **Bayesian Networks**: These are graphical models that represent the probabilistic relationships among various factors influencing landslides, such as rainfall, soil type, and slope angle. They can be used to predict the likelihood of a landslide occurring in a given area
2. **Bayesian Optimization**: This technique is used to fine-tune the hyperparameters of machine learning models, improving their performance in landslide prediction. For example, a study used Bayesian optimization with temporal convolutional networks to predict landslide displacement, achieving superior results compared to traditional methods
3. **Naïve Bayes Classifier**: This is a simple yet effective Bayesian classification method that assumes independence among predictors. It has been applied to landslide susceptibility mapping, where it helps in identifying areas prone to landslides based on historical data
4. **Hybrid Models**: Combining Bayesian methods with other machine learning techniques, such as Random Forests or XGBoost, can enhance prediction accuracy. For instance, Bayesian hyperparameter optimization has been used to improve the performance of these models in landslide susceptibility mapping

**Tasks:**

1. **Dataset Preparation:**
   * Download the **Global Landslide Dataset** from **Kaggle** or **NASA GLC**.
   * Load the dataset into Python using **pandas**.
   * Identify and handle missing values.
   * Convert categorical variables (e.g., "Trigger", "Landslide Size") into numerical form using **Label Encoding or One-Hot Encoding**.
2. **Feature Selection & Preprocessing:**
   * Select relevant features such as **Rainfall**, **Slope Angle**, **Soil Type**, **Elevation**, and **Trigger Type**.
   * Scale the numerical features using **StandardScaler or MinMaxScaler**.
   * Split the dataset into **training (80%)** and **testing (20%)** sets.
3. **Model Implementation:**
   * Train a **Naïve Bayes classifier (GaussianNB)** to classify whether a landslide will occur (Yes=1, No=0).
   * Predict landslide occurrences for the test set.
4. **Model Evaluation:**
   * Calculate **accuracy, precision, recall, and F1-score** of the model.
   * Plot a **confusion matrix heatmap** to visualize prediction performance.
5. **Prediction on New Data:**
   * Use the trained model to predict landslide probability for a **new location** with given environmental conditions:
     + **Rainfall = 250mm**, **Soil Moisture = 40%**, **Slope Angle = 50°**, **Soil Type = Clay**, **Elevation = 1200m**
   * Print the probabilities for **landslide vs. no landslide** and classify the site as **safe or at risk**.
6. **Discussion:**
   * Analyze the model’s strengths and weaknesses.
   * Discuss real-world applications of **Bayesian classification** in **landslide risk prediction**.

**Sample Code:**

# Import required libraries

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 LabelEncoder, StandardScaler

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

# Step 1: Load the dataset

# Replace 'landsilde\_data.csv' with the actual dataset file name

df = pd.read\_csv("landslide\_data.csv")

# Step 2: Explore the dataset

print("Dataset Preview:\n", df.head())

print("\nDataset Information:\n")

print(df.info())

print("\nMissing Values:\n", df.isnull().sum())

# Step 3: Handle missing values (Fill numerical values with mean)

df.fillna(df.mean(), inplace=True)

# Encode categorical variables

label\_encoder = LabelEncoder()

df["Trigger"] = label\_encoder.fit\_transform(df["Trigger"]) # Convert triggers (e.g., Rainfall, Earthquake) to numbers

df["Landslide\_Occurrence"] = label\_encoder.fit\_transform(df["Landslide\_Occurrence"]) # Yes=1, No=0

# Step 4: Define features and target variable

X = df.drop(columns=["Landslide\_Occurrence"]) # Features

y = df["Landslide\_Occurrence"] # Target variable (1 = Landslide, 0 = No Landslide)

# Normalize numerical features

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

# Step 5: Split dataset into training (80%) and testing (20%) sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.2, random\_state=42)

# Step 6: Train Naïve Bayes Model

model = GaussianNB()

model.fit(X\_train, y\_train)

# Step 7: Make Predictions

y\_pred = model.predict(X\_test)

# Step 8: Evaluate Model Performance

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"\nModel Accuracy: {accuracy:.2f}")

# Confusion Matrix

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

print("\nConfusion Matrix:\n", conf\_matrix)

# Classification Report

print("\nClassification Report:\n", classification\_report(y\_test, y\_pred))

# Step 9: Visualize Confusion Matrix

plt.figure(figsize=(6,5))

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=["No Landslide", "Landslide"], yticklabels=["No Landslide", "Landslide"])

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.title("Landslide Classification - Confusion Matrix")

plt.show()

# Step 10: Predict Landslide Probability for a New Location

new\_data = np.array([[250, 40, 50, label\_encoder.transform(["Rainfall"])[0], 1200]]) # Example input

new\_data\_scaled = scaler.transform(new\_data)

probability = model.predict\_proba(new\_data\_scaled)

print(f"\nProbability of No Landslide: {probability[0][0]:.2f}")

print(f"Probability of Landslide: {probability[0][1]:.2f}")

# Classify the new location

if probability[0][1] > 0.5:

print("\nPrediction: High Risk of Landslide!")

else:

print("\nPrediction: Safe from Landslide.")

**How to Use the Code:**

1. **Download** the **Global Landslide Dataset** from **Kaggle or NASA GLC**.
2. **Preprocess the data**: Handle missing values and encode categorical variables.
3. **Train a Naïve Bayes classifier** to predict landslide occurrences.
4. **Evaluate the model** using **accuracy, confusion matrix, and classification report**.
5. **Use the trained model** to predict landslide probability for new locations.