import pandas as pd

import lumpy as np

# Load the sensor data

def = pd.read\_csv('bridge\_data.csv')

# Check for missing values

df.isnull().sum()

# Fill or drop missing values

df.fillna(method='ffill', in place=True)

# Convert time-stamps to date time objects if needed

df['timestamp'] = pd.to\_datetime(df['timestamp'])

# Feature Engineering

df['hour'] = df['timestamp'].dt.hour

df['day\_of\_week'] = df['timestamp'].dt.dayofweek

# Normalize or scale the sensor data if necessary

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

df[['sensor\_1', 'sensor\_2', 'sensor\_3']] = scaler.fit\_transform(df[['sensor\_1', 'sensor\_2', 'sensor\_3']])

# Check processed data

print(df.head())

from sklearn.ensemble import RandomForestClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score

# Create features (X) and labels (y) from data

X = df[['sensor\_1', 'sensor\_2', 'sensor\_3', 'hour', 'day\_of\_week']]

y = df['maintenance\_needed'] # Binary target: 1 if maintenance is required, else 0

# Split the data into training and testing sets

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

# Train a RandomForest model

model = RandomForestClassifier(n\_estimators=100, random\_state=42)

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

# Make predictions

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

# Evaluate the model

print("Accuracy:", accuracy\_score(y\_test, y\_pred))