AI ASSISTED CODING

LAB TEST 3

NAME : N.PRANAY KUMAR

ROLLNO : 2403A52087

BATCH : 04

TASK

Scenario: In the domain of Environmental Monitoring, a company is facing a challenge related to algorithms with ai assistance.
Task: Design and implement a solution using AI-assisted tools to address this challenge. Include code, explanation of AI integration, and test results.
Deliverables: Source code, explanation, and output screenshots.

PROMPT

Create an AI-based system that helps monitor the environment and predicts when sensors or machines might fail.
Use Python with a simple machine learning model, explain how AI is used, and show the results with examples.

CODE:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split   # ☑ fixed import typo
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
import sys
import io

def run_predictive_maintenance_model():
    """
    Simulates environmental sensor data, trains a Random Forest model
    for predictive maintenance, evaluates it, and performs new predictions.
    """
```

```
    # --- 1. Simulate Environmental Sensor Data ---
    np.random.seed(42)
    N_SAMPLES = 500

    data = {
        'temperature_C': np.random.normal(loc=25, scale=5, size=N_SAMPLES),
        'voltage_V': np.random.normal(loc=3.7, scale=0.2, size=N_SAMPLES),
        'noise_level': np.random.uniform(low=0.1, high=1.5, size=N_SAMPLES),
        'signal_strength_dBm': np.random.normal(loc=-70, scale=10,
size=N_SAMPLES)
    }
    df = pd.DataFrame(data)

    # --- 2. Create Synthetic Failure Risk Label ---
    df['Failure_Risk'] = (
        ((df['temperature_C'] > 30) & (df['voltage_V'] < 3.5)) |
        ((df['noise_level'] > 1.2) & (df['signal_strength_dBm'] < -85))
    ).astype(int)

    # Add noise to simulate real-world unpredictability
    random_indices = df.sample(frac=0.05, random_state=42).index
    df.loc[random_indices, 'Failure_Risk'] = 1 - df.loc[random_indices,
'Failure_Risk']

    X = df[['temperature_C', 'voltage_V', 'noise_level',
'signal_strength_dBm']]
    y = df['Failure_Risk']

    # --- 3. Split data ---
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.3, random_state=42
    )

    # --- 4. Train Model ---
    model = RandomForestClassifier(
        n_estimators=100, random_state=42, class_weight='balanced'
    )
    model.fit(X_train, y_train)

    # --- 5. Evaluate Model ---
    y_pred = model.predict(X_test)

    old_stdout = sys.stdout
    sys.stdout = output_capture = io.StringIO()

    print("--- Model Training and Test Results ---")
    print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
    print("\nClassification Report:\n", classification_report(y_test, y_pred))

    # --- 6. New Predictions ---
    new_sensor_data_A = pd.DataFrame({
        'temperature_C': [26.0],
        'voltage_V': [3.75],
        'noise_level': [0.5],
        'signal_strength_dBm': [-65.0]
    })
    prediction_A = model.predict(new_sensor_data_A)[0]
    risk_A = "HIGH RISK (1)" if prediction_A == 1 else "NORMAL (0)"
    print("\n--- NEW Sensor Reading A (Normal) ---")
    print(f"Features: Temp=26.0°C, Volt=3.75V, Noise=0.5, Signal=-65.0 dBm")
    print(f"Prediction: {prediction_A} ({risk_A})")

    new_sensor_data_B = pd.DataFrame({
        'temperature_C': [32.5],
        'voltage_V': [3.4],
        'noise_level': [0.6],
        'signal_strength_dBm': [-72.0]
    })
    prediction_B = model.predict(new_sensor_data_B)[0]
    risk_B = "HIGH RISK (1)" if prediction_B == 1 else "NORMAL (0)"
    print("\n--- NEW Sensor Reading B (Pre-Failure/Anomaly) ---")
    print(f"Features: Temp=32.5°C, Volt=3.4V, Noise=0.6, Signal=-72.0 dBm")
    print(f"Prediction: {prediction_B} ({risk_B})")

    sys.stdout = old_stdout
    return output_capture.getvalue()


# --- Execute ---
output = run_predictive_maintenance_model()
print(output)
```

OUTPUT:

```
--- Model Training and Test Results ---
Confusion Matrix:
 [[119   2]
 [ 28   1]]

Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.98      0.89       121
           1       0.33      0.03      0.06        29

    accuracy                           0.80       150
   macro avg       0.57      0.51      0.48       150
weighted avg       0.72      0.80      0.73       150


--- NEW Sensor Reading A (Normal) ---
Features: Temp=26.0°C, Volt=3.75V, Noise=0.5, Signal=-65.0 dBm
Prediction: 0 (NORMAL (0))

--- NEW Sensor Reading B (Pre-Failure/Anomaly) ---
Features: Temp=32.5°C, Volt=3.4V, Noise=0.6, Signal=-72.0 dBm
Prediction: 1 (HIGH RISK (1))
```

❖

OBSERVATION:

- ❖ The AI model can correctly identify when environmental sensors are working normally or showing early signs of failure
- ❖ Using Random Forest helps the system make reliable predictions even when the sensor data is slightly noisy or uncertain.

TASK 2 :

Scenario: In the domain of Education, a company is facing a challenge related to algorithms with ai assistance.

Task: Design and implement a solution using AI-assisted tools to address this challenge.
Include code, explanation of AI integration, and test results.
Deliverables: Source code, explanation, and output screenshots.

PROMPT :

Create an AI-based system that helps predict students' performance using their study data and learning patterns.
Use Python with a machine learning model, explain how AI is used, and show the test results with examples.

CODE :

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
import io, sys

def run_student_performance_ai():
    """
    AI-based Student Performance Prediction System
    Uses study hours, attendance, and preparation data to predict performance.
    """

    # --- 1. Simulate Student Data ---
    np.random.seed(42)
    N = 300

    data = {
        'study_hours': np.random.normal(4, 1.5, N).clip(0, 8),
        'attendance_percent': np.random.uniform(50, 100, N),
        'previous_score': np.random.uniform(30, 100, N),
        'test_preparation': np.random.choice([0, 1], N, p=[0.6, 0.4])  # 0 =
No, 1 = Yes
    }

    df = pd.DataFrame(data)

    # --- 2. Target Variable: Performance ---
    df['Good_Performance'] = (
        (df['study_hours'] > 4.5) &
        (df['attendance_percent'] > 75) &
        (df['previous_score'] > 60)
    ).astype(int)

    # Add noise for realism
    noise = df.sample(frac=0.05, random_state=42).index
    df.loc[noise, 'Good_Performance'] = 1 - df.loc[noise, 'Good_Performance']

    # --- 3. Train/Test Split ---
    X = df[['study_hours', 'attendance_percent', 'previous_score',
'test_preparation']]
    y = df['Good_Performance']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

    # --- 4. Train Model ---
    model = RandomForestClassifier(n_estimators=100, random_state=42)
    model.fit(X_train, y_train)
```

```
    # --- 5. Evaluate Model ---
    y_pred = model.predict(X_test)

    old_stdout = sys.stdout
    sys.stdout = output_capture = io.StringIO()

    print("--- Model Evaluation ---")
    print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
    print("\nClassification Report:\n", classification_report(y_test, y_pred))

    # --- 6. Predict for New Students ---
    new_students = pd.DataFrame({
        'study_hours': [3.0, 6.0],
        'attendance_percent': [65, 90],
        'previous_score': [55, 80],
        'test_preparation': [0, 1]
    })

    predictions = model.predict(new_students)

    print("\n--- New Student Predictions ---")
    for i, p in enumerate(predictions):
        status = "GOOD PERFORMANCE (1)" if p == 1 else "LOW PERFORMANCE (0)"
        print(f"Student {i+1}: {status}")

    sys.stdout = old_stdout
    return output_capture.getvalue()


# --- Run the AI Model ---
output = run_student_performance_ai()
print(output)
```

OUTPUT:

```
--- Model Evaluation ---
Confusion Matrix:
[[62  7]
 [14  7]]

Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.99      0.90        69
           1       0.88      0.33      0.48        21

    accuracy                           0.83        90
   macro avg       0.85      0.66      0.69        90
weighted avg       0.84      0.83      0.80        90


--- New Student Predictions ---
Student 1: LOW PERFORMANCE (0)
Student 2: GOOD PERFORMANCE (1)
```

OBSERVATION:

- The AI model accurately predicts whether a student will perform well based on study hours, attendance, and past scores.
- Students with higher study time and attendance are more likely to show good performance in the prediction results