AI ASSISTED CODING

LAB TEST-3

SET:E5

QUESTION-1:

Scenario: In the Finance sector, a company faces a challenge related to data structures with ai.
Task: Use AI-assisted tools to solve a problem involving data structures with ai in this context.
Deliverables: Submit the source code, explanation of AI assistance used, and sample output.

CODE WITH OUTPUT:

```
# finance_graph_risk_ai.py

import networkx as nx
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

# ----------------------------
# Step 1: Create a Graph of Customers and Financial Relationships
# ----------------------------
G = nx.Graph()

# Add customer nodes (10 customers)
for i in range(1, 11):
    G.add_node(i,
               avg_balance=np.random.uniform(1000, 5000),
               num_loans=np.random.randint(1, 5),
               income=np.random.uniform(3000, 10000))

# Add financial relationships (edges)
```

```
edges = [(1,2), (2,3), (3,4), (4,5), (6,7), (7,8), (8,9), (9,10), (2,7),
(5,9)]
G.add_edges_from(edges)

# ----------------------------
# Step 2: Feature Engineering (AI-suggested)
# ----------------------------
# Compute graph-based metrics as features
degree_centrality = nx.degree_centrality(G)
closeness_centrality = nx.closeness_centrality(G)
betweenness_centrality = nx.betweenness_centrality(G)

# Compile features into a DataFrame
data = []
for node, attrs in G.nodes(data=True):
    features = {
        'customer_id': node,
        'avg_balance': attrs['avg_balance'],
        'num_loans': attrs['num_loans'],
        'income': attrs['income'],
        'degree_centrality': degree_centrality[node],
        'closeness_centrality': closeness_centrality[node],
        'betweenness_centrality': betweenness_centrality[node],
        'defaulted': np.random.choice([0, 1], p=[0.8, 0.2])  # 1 = default
    }
    data.append(features)

df = pd.DataFrame(data)

# ----------------------------
# Step 3: AI Model Training (Random Forest)
# ----------------------------
X = df.drop(['customer_id', 'defaulted'], axis=1)
y = df['defaulted']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

# ----------------------------
# Step 4: Evaluation
```

```
# ------------------------------
print("Classification Report:\n")
print(classification_report(y_test, y_pred))

print("\nSample Predictions:")
results = X_test.copy()
results['Actual'] = y_test.values
results['Predicted'] = y_pred
print(results.head())
```

OUTPUT:

```
Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00         3

    accuracy                           1.00         3
   macro avg       1.00      1.00      1.00         3
weighted avg       1.00      1.00      1.00         3


Sample Predictions:
   avg_balance  num_loans       income  degree_centrality  \
8  2890.832507          4  8950.821533           0.333333
1  1164.737484          4  8088.270936           0.333333
5  2709.111621          3  5594.634202           0.111111

   closeness_centrality  betweenness_centrality  Actual  Predicted
8              0.450000                0.333333       0          0
1              0.473684                0.361111       0          0
5              0.346154                0.000000       0          0
```

EXPLANATION:

This notebook presents a financial risk assessment model using graph analysis and machine learning.

It begins by constructing a graph representing customers and their financial connections using the `networkx` library. Nodes in the graph represent customers, and edges signify relationships.

Customer attributes like average balance, number of loans, and income are assigned to each node.

Graph-based features, specifically degree, closeness, and betweenness centrality, are computed to capture the importance and influence of each customer within the network.

These graph features are then combined with the initial customer attributes into a pandas DataFrame.

A 'defaulted' column is added, randomly assigning a default status (1 for default, 0 otherwise) to simulate the target variable.

The data is split into training and testing sets for model development and evaluation.

A Random Forest Classifier is trained on the training data to predict customer defaults.

The model's performance is evaluated using a classification report, showing precision, recall, and f1-score.

Finally, sample predictions are shown for the test set to illustrate the model's output.

QUESTION-2:

Scenario: In the Agriculture sector, a company faces a challenge related to data structures with ai.

Task: Use AI-assisted tools to solve a problem involving data structures with ai in this context.

Deliverables: Submit the source code, explanation of AI assistance used, and sample output.

CODE WITH OUTPUT:

```
# crop_yield_ai.py

import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, r2_score

# ------------------------------
# Step 1: Create Structured Data (Data Structures)
# ------------------------------
# Each record represents a farm with various parameters
data = [
    {"farm_id": 1, "soil_ph": 6.5, "rainfall": 420, "temperature": 28,
"irrigation": 3, "yield_ton_per_ha": 3.4},
    {"farm_id": 2, "soil_ph": 5.8, "rainfall": 380, "temperature": 30,
"irrigation": 2, "yield_ton_per_ha": 2.9},
    {"farm_id": 3, "soil_ph": 7.1, "rainfall": 450, "temperature": 27,
"irrigation": 4, "yield_ton_per_ha": 3.8},
    {"farm_id": 4, "soil_ph": 6.2, "rainfall": 400, "temperature": 29,
"irrigation": 3, "yield_ton_per_ha": 3.2},
]

# Convert list of dictionaries into a DataFrame
df = pd.DataFrame(data)

# ------------------------------
# Step 2: Prepare Data for AI Model
# ------------------------------
X = df[["soil_ph", "rainfall", "temperature", "irrigation"]]
y = df["yield_ton_per_ha"]
```

```
# Split into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

# ------------------------------
# Step 3: Train AI Model
# ------------------------------
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# ------------------------------
# Step 4: Evaluate Model
# ------------------------------
y_pred = model.predict(X_test)

mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Model Evaluation:")
print(f"Mean Absolute Error: {mae:.2f}")
print(f"R² Score: {r2:.2f}")

# ------------------------------
# Step 5: Predict New Crop Yields
# ------------------------------
new_data = pd.DataFrame([
    {"soil_ph": 6.4, "rainfall": 410, "temperature": 28, "irrigation": 3},
    {"soil_ph": 7.2, "rainfall": 460, "temperature": 26, "irrigation": 5},
])

predictions = model.predict(new_data)
print("\nPredicted Crop Yields (tons/ha):")
print(predictions)
```

OUTPUT:

```
Model Evaluation:
Mean Absolute Error: 0.44
R² Score: -8.53

Predicted Crop Yields (tons/ha):
[3.488 3.696]
```

EXPLANATION:

The process begins by creating a structured dataset that simulates farm-level data. Each record in this dataset contains information such as `farm_id`, `soil_ph`, `rainfall`, `temperature`, `irrigation`, and the actual `yield_ton_per_ha`.

This list of dictionaries is then converted into a pandas DataFrame for easier manipulation and analysis.

The data is prepared for the AI model by separating the features (input variables) from the target variable (the variable to be predicted). In this case, `soil_ph`, `rainfall`, `temperature`, and `irrigation` are the features, and `yield_ton_per_ha` is the target.

The dataset is then split into training and testing sets. The training set is used to train the model, while the testing set is used to evaluate its performance on unseen data.

A Random Forest Regressor model is initialized and trained using the training data. The Random Forest algorithm is well-suited for regression tasks and can capture complex relationships between features and the target variable.

After training, the model is used to make predictions on the test set.

The model's performance is evaluated using two common regression metrics: Mean Absolute Error (MAE) and R^2 Score. MAE measures the average absolute difference between the predicted and actual values, while R^2 Score indicates the proportion of the variance in the target variable that is predictable from the features.

Finally, the trained model is used to predict crop yields for new, unseen data points, demonstrating how the model can be applied in practice to estimate yields for different farm conditions.