Artificial Intelligence Assignment

PART- II

Submitted By: 2K23/IT/30 (AREEBA ALI)

TOPIC: CLASSIFICATION

ALGORITHM: Decision Trees

README:

CONTENTS:

1. decision_tree_classifier.py - Main Python implementation
2. high_res_tree.png - Decision tree visualization
3. fictional_characters.xlsx - Dataset used
4. genre_classifier.pkl - Trained model
5. classification_report.txt - Model performance metrics

INSTRUCTIONS TO RUN:

1. Install requirements: pandas, scikit-learn, matplotlib, openpyxl
2. Place all files in the same folder
3. Execute: python decision_tree_classifier.py

Dataset used: Fictional Characters Dataset by Pratyush Puri (https://www.kaggle.com/datasets/pratyushpuri/synthetic-fictional-characters-dataset/discussion?sort=hotness)

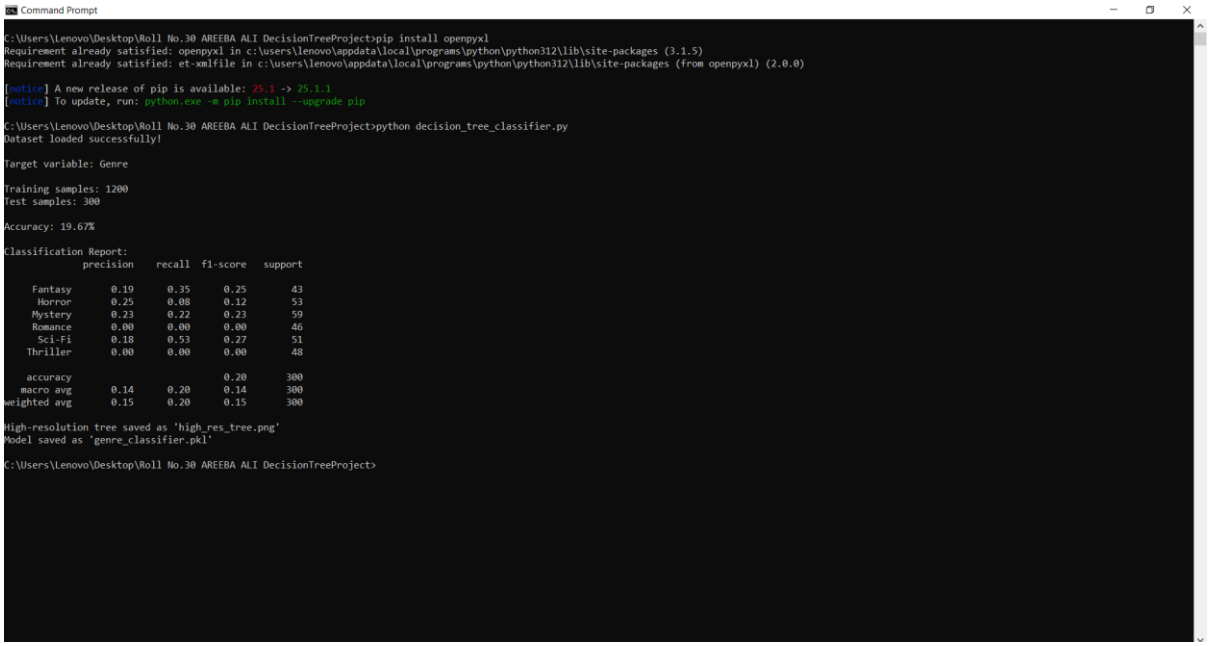
CODE:

```
"""
Fictional Character Genre Classifier
Dataset: fictional_characters.xlsx
Algorithm: Decision Tree Classifier
"""
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt

# --- Data Loading ---
try:
    data = pd.read_excel('fictional_characters.xlsx', engine='openpyxl')
    print("Dataset loaded successfully!\n")
except Exception as e:
    print(f"Error: {e}")
    exit()

# --- Target Selection ---
TARGET = 'Genre'  # Change if needed
print(f"Target variable: {TARGET}\n")

# --- Preprocessing ---
# Encode categorical target
le = LabelEncoder()
y = le.fit_transform(data[TARGET])

# Encode features (simplified approach)
X = data.drop(TARGET, axis=1)
for col in X.select_dtypes(include=['object']).columns:
    X[col] = LabelEncoder().fit_transform(X[col].astype(str))

# --- Train-Test Split ---
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
print(f"Training samples: {len(X_train)}")
```

```
print(f"Test samples: {len(X_test)}\n")

# --- Decision Tree Model ---
model = DecisionTreeClassifier(
   criterion='gini',
   max_depth=3,  # Restricted for interpretability
   random_state=42
)
model.fit(X_train, y_train)

# --- Evaluation ---
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2%}\n")

print("Classification Report:")
print(classification_report(
   y_test, y_pred,
   target_names=le.classes_,
   zero_division=0  # Suppresses warnings
))

# --- Visualization ---
plt.figure(figsize=(25, 15), dpi=300)  # Double the size and resolution
plot_tree(
   model,
   feature_names=X.columns,
   class_names=le.classes_,
   filled=True,
   rounded=True,
   fontsize=10,  # Larger font
   proportion=True,  # Shows percentages
   impurity=False  # Cleaner look
)
plt.tight_layout()  # Prevents label cutoff
plt.savefig('high_res_tree.png', bbox_inches='tight', dpi=300)  # 300 DPI for print quality
print("High-resolution tree saved as 'high_res_tree.png'")

# --- Save Model ---
import joblib
```

joblib.dump(model, 'genre_classifier.pkl')
print("Model saved as 'genre_classifier.pkl'")

OUTPUT:



C:\Users\Lenovo\PycharmProjects\PythonProject.venv\Scripts\python.exe C:\Users\Lenovo\PycharmProjects\PythonProject\decision_tree_classifier.py Dataset loaded successfully!

Target variable: Genre

Training samples: 1200 Test samples: 300

Accuracy: 19.67%

Classification Report: precision recall f1-score support

Fantasy	0.19	0.35	0.25	43
Horror	0.25	0.08	0.12	53
Mystery	0.23	0.22	0.23	59
Romance	0.00	0.00	0.00	46
Sci-Fi	0.18	0.53	0.27	51
Thriller	0.00	0.00	0.00	48
accuracy			0.20	300

macro avg 0.14 0.20 0.14 300 weighted avg 0.15 0.20 0.15 300

High-resolution tree saved as 'high_res_tree.png' Model saved as 'genre_classifier.pkl'

Process finished with exit code 0

MODEL IMAGE: