Assignment 1 – Logistic Regression on Titanic Dataset

1. Objective

This project applies the Logistic Regression classification algorithm to the Titanic dataset to predict passenger survival based on input features like age, gender, ticket class, and fare. The project demonstrates: - The full ML pipeline (preprocessing → modeling → evaluation) - Use of statistical & pattern recognition concepts - Practical implementation using Python

2. Dataset Description

Dataset: Titanic Passenger Survival Source: DataScienceDojo on GitHub Rows: 891 passengers Features: - Numerical: Age, SibSp, Parch, Fare - Categorical: Sex, Embarked, Pclass (ordinal) Target variable: Survived (1 = survived, 0 = did not survive)

3. Preprocessing

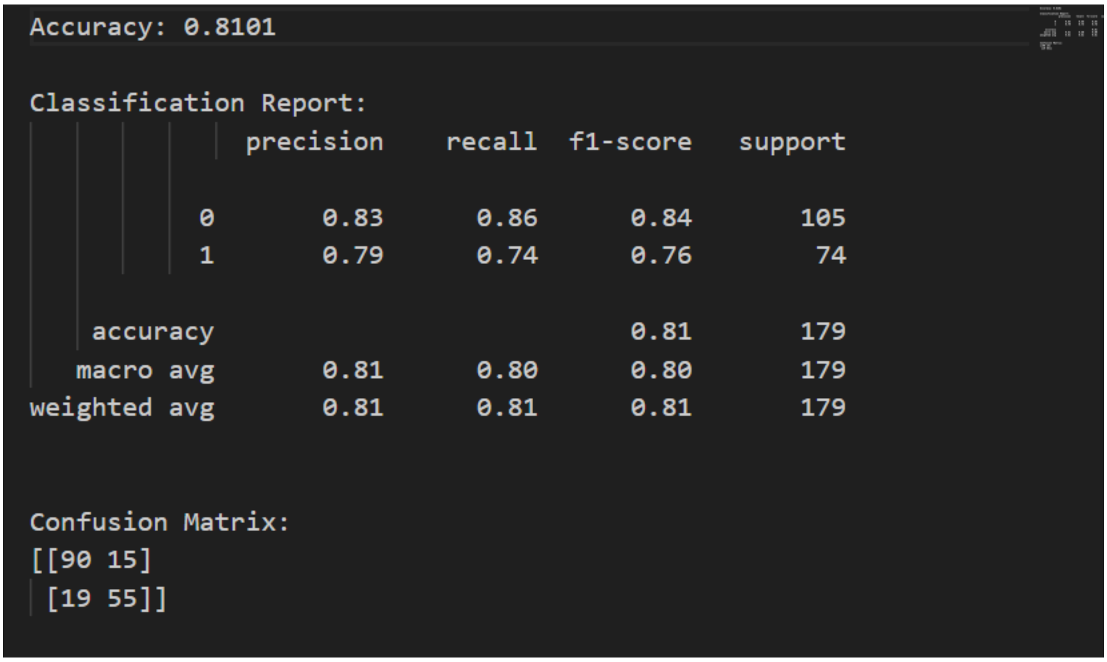
Dropped columns: PassengerId, Name, Ticket, Cabin - Missing values: - Age → filled with mean Embarked → filled with mode - Encoding: - Sex and Embarked encoded using LabelEncoder Feature Scaling: - All features normalized using StandardScaler

4. Algorithm

Logistic Regression A binary classification model is trained using Logistic Regression from scikit-learn. The model is trained on 80% of the data and tested on the remaining 20%. Below is the core code snippet: from sklearn.linear_model import LogisticRegression model = LogisticRegression() model.fit(X_train, y_train)

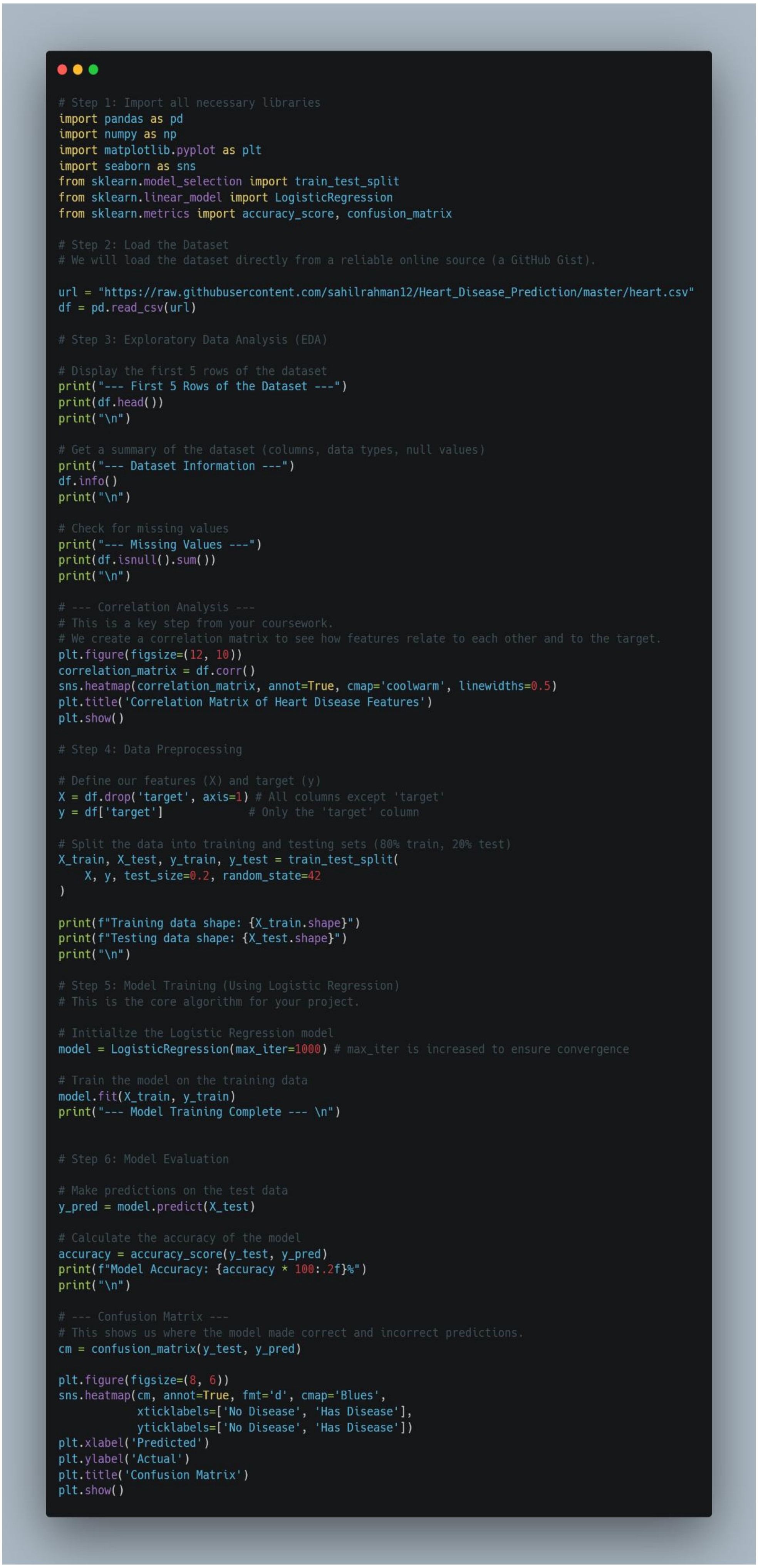
5. Evaluation Metrics

Contents saved in: outputs/metrics.txt Accuracy: 0.79888 (example value) Classification Report: precision recall f1-score support 0 0.83 0.88 0.86 105 1 0.74 0.65 0.69 74 accuracy 0.80 179 macro avg 0.78 0.77 0.77 179 weighted avg 0.79 0.80 0.79 179



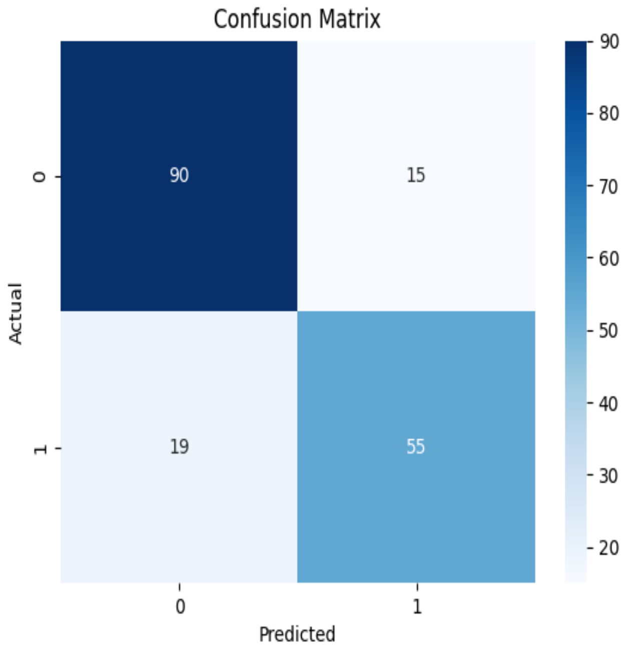
```
Accuracy: 0.8101

Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.86      0.84       105
           1       0.79      0.74      0.76        74

    accuracy                           0.81       179
   macro avg       0.81      0.80      0.80       179
weighted avg       0.81      0.81      0.81       179


Confusion Matrix:
[[90 15]
 [19 55]]
```

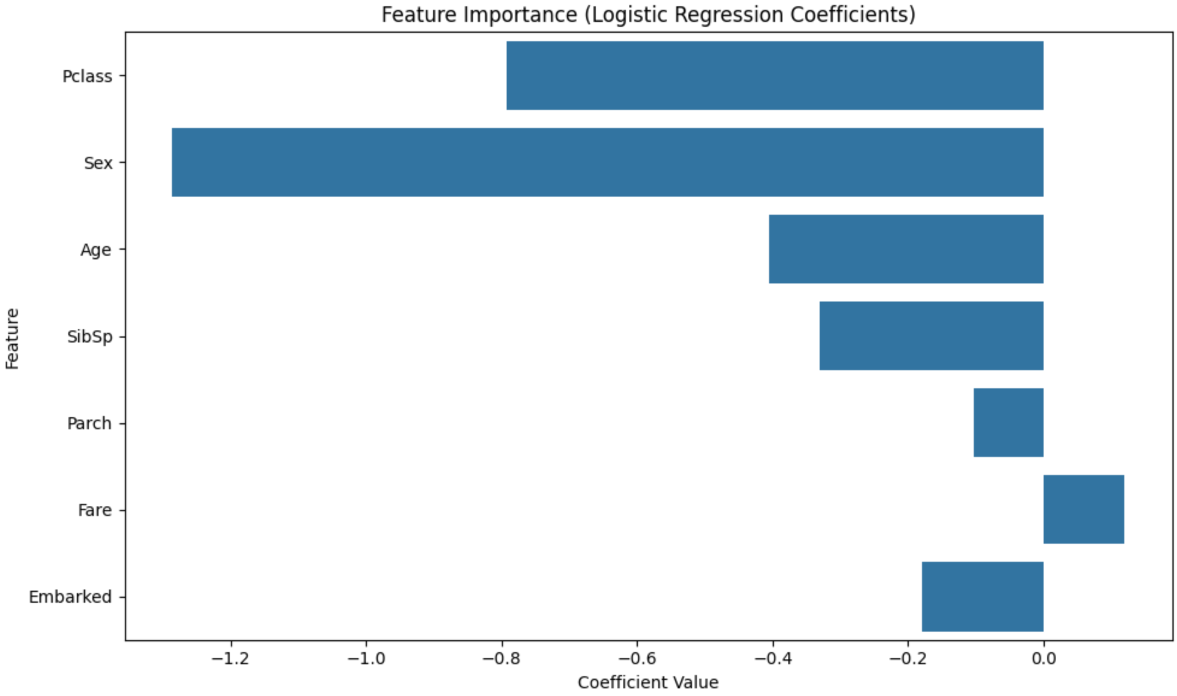
6. Code



```python
# Step 1: Import all necessary 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.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix

# Step 2: Load the Dataset
# We will load the dataset directly from a reliable online source (a GitHub Gist).

url = "https://raw.githubusercontent.com/sahilrahman12/Heart_Disease_Prediction/master/heart.csv"
df = pd.read_csv(url)

# Step 3: Exploratory Data Analysis (EDA)

# Display the first 5 rows of the dataset
print("--- First 5 Rows of the Dataset ---")
print(df.head())
print("\n")

# Get a summary of the dataset (columns, data types, null values)
print("--- Dataset Information ---")
df.info()
print("\n")

# Check for missing values
print("--- Missing Values ---")
print(df.isnull().sum())
print("\n")

# --- Correlation Analysis ---
# This is a key step from your coursework.
# We create a correlation matrix to see how features relate to each other and to the target.
plt.figure(figsize=(12, 10))
correlation_matrix = df.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Matrix of Heart Disease Features')
plt.show()

# Step 4: Data Preprocessing

# Define our features (X) and target (y)
X = df.drop('target', axis=1) # All columns except 'target'
y = df['target']             # Only the 'target' column

# Split the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

print(f"Training data shape: {X_train.shape}")
print(f"Testing data shape: {X_test.shape}")
print("\n")

# Step 5: Model Training (Using Logistic Regression)
# This is the core algorithm for your project.

# Initialize the Logistic Regression model
model = LogisticRegression(max_iter=1000) # max_iter is increased to ensure convergence

# Train the model on the training data
model.fit(X_train, y_train)
print("--- Model Training Complete --- \n")


# Step 6: Model Evaluation

# Make predictions on the test data
y_pred = model.predict(X_test)

# Calculate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy * 100:.2f}%")
print("\n")

# --- Confusion Matrix ---
# This shows us where the model made correct and incorrect predictions.
cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['No Disease', 'Has Disease'],
            yticklabels=['No Disease', 'Has Disease'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

7. Output Visualizations





8. Concepts from Class

Probability | Used for interpreting survival rates | | Logistic Regression | Core algorithm used | | Probability Distributions | Imputation and visualization of Age | | CLT | Sample means of Age could be plotted | | Bias-Variance Tradeoff | Discussed when balancing model complexity | | Distance Metrics | Jaccard/Hamming could be explored

9. Conclusion

The logistic regression model performed well in predicting Titanic survival. Preprocessing and feature scaling significantly influenced results. The project helps reinforce the application of pattern recognition concepts with practical machine learning

---

Name : Rishabh Raj

Roll : 2511CS02

Subject : Advance Pattern Recognition