Data Processing and Model Evaluation Tasks

Task 1: Data Processing (Using NumPy)

Data processing is a critical first step in machine learning, ensuring that raw data is cleaned, transformed, and ready for use in AI model training. This task involves handling missing values, encoding categorical features, scaling numerical features, and splitting the data into training and testing sets.

Steps of Implementation:

1. Load the dataset using numpy.genfromtxt or numpy.loadtxt.
2. Handle missing values using SimpleImputer with mean for numerical columns.
3. Remove duplicates using np.unique.
4. Encode categorical variables using LabelEncoder from sklearn.
5. Scale numerical features using StandardScaler from sklearn.
6. Split the data into training and testing sets using train_test_split from sklearn.

Python Code (Task 1):

```
# Import necessary libraries
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.impute import SimpleImputer

# Load the dataset
data = np.genfromtxt('data.csv', delimiter=',', skip_header=1)  # Assuming the file has
```

```
a header


# Step 1: Data Cleaning

imputer = SimpleImputer(strategy='mean')

data[:, :-1] = imputer.fit_transform(data[:, :-1])  # Assuming last column is the target

data = np.unique(data, axis=0)


# Step 2: Encoding Categorical Variables

label_encoder = LabelEncoder()

data[:, -1] = label_encoder.fit_transform(data[:, -1])  # Assuming the last column is

categorical


# Step 3: Feature Scaling

scaler = StandardScaler()

data[:, :-1] = scaler.fit_transform(data[:, :-1])  # Scaling features, not the target


# Step 4: Splitting Data

X = data[:, :-1]

y = data[:, -1]

X_train,  X_test,  y_train,  y_test  =  train_test_split(X,  y,  test_size=0.2,

random_state=42)
```

Task 2: Model Evaluation and Comparison (Using NumPy)

Model evaluation and comparison involve training multiple machine learning models on preprocessed data, then evaluating their performance using standardized metrics such as accuracy, precision, recall, and F1-score. This step helps in selecting the best model for a particular problem.

Steps of Implementation:

1. Initialize models such as Logistic Regression, Decision Tree, and Random Forest using sklearn.
2. Train each model using the training dataset (X_train, y_train).
3. Evaluate each model using the testing dataset (X_test, y_test).
4. Calculate performance metrics like accuracy, precision, recall, and F1-score.
5. Compare the performance of each model to select the best one.

Python Code (Task 2):

```
# Import necessary libraries
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

# Step 1: Initialize Models
logistic_model = LogisticRegression(random_state=42)
decision_tree_model = DecisionTreeClassifier(random_state=42)
random_forest_model = RandomForestClassifier(random_state=42)

# Step 2: Train Models
logistic_model.fit(X_train, y_train)
decision_tree_model.fit(X_train, y_train)
random_forest_model.fit(X_train, y_train)

# Step 3: Evaluate Models
```

```
models = {
    "Logistic Regression": logistic_model,
    "Decision Tree": decision_tree_model,
    "Random Forest": random_forest_model
}

for name, model in models.items():
    y_pred = model.predict(X_test)
    print(f"\n{name} Model Performance:")
    print(f"Accuracy: {accuracy_score(y_test, y_pred):.2f}")
    print("Classification Report:")
    print(classification_report(y_test, y_pred))
```