INDEX

S. No	Title of the Experiment	Page No.
1	Explore Numpy and Pandas libraries for data pre-processing techniques such as handling missing data and outliers, encoding to prepare data for machine learning models.	
2	Implement Linear Regression to predict a continuous target variable and explore how Multiple Linear Regression can capture the relationship between multiple predictors and the target. Use GHI dataset to perform model training, validation, and performance evaluation.	
3	Apply Polynomial Regression to model non-linear relationships between the input variables and the target variable. Investigate how the degree of the polynomial impacts the model's ability to fit GHI data, avoiding overfitting or underfitting.	
4	Learn how to apply Naïve Bayes and K-Nearest Neighbors (KNN) supervised learning techniques to identify Level of experience.	
5	Implement a Decision Tree classifier technique to identify whether the cars require service or not and plot the confusion Matrix using seaborn library.	
6	Implementation of Support Vector Machine technique to identify user behavior classes and plot the confusion Matrix using seaborn library.	
7	Apply the Random Forest algorithm to identify Classification of users based on driving habits (e.g., Commuter, Long-Distance Traveler) and plot the confusion Matrix using seaborn library.	
8	Implementation of AdaBoost technique to Classify loan approval and loan not approval and plot the confusion Matrix using seaborn library.	
9	Implementation of XGBoost and LightGBM technique to Classify Waitlist and admit in Wharton Class of 2025's statistics and plot the confusion Matrix using seaborn library.	
10	Implementation of K-means and Hierarchical Clustering on customer segmentation dataset and plot the elbow and dendrogram for k-means and Hierarchical clustering respectively.	

Ex.No:1	**Explore Numpy and Pandas libraries for data pre-processing techniques such as handling missing data and outliers, encoding to prepare data for machine learning models.**

INTRODUCTION

Data preprocessing is a crucial step in machine learning that involves cleaning, transforming, and organizing raw data to improve the quality and accuracy of models. NumPy provides efficient numerical operations, while Pandas offers powerful tools for handling missing values, outliers, and categorical encoding.

AIM

To implement data preprocessing using NumPy and Pandas by handling missing data, outliers, and encoding categorical variables.

ALGORITHM

A. Handling Missing Data

1. Load the dataset into a Pandas DataFrame.
2. Identify missing values using isnull() or isna().
3. Replace missing numerical values using mean or median.
4. Replace missing categorical values using mode.
5. Alternatively, remove rows/columns containing missing data using dropna().

B. Detecting & Handling Outliers (IQR Method)

1. Select numerical columns with possible outliers.
2. Calculate Q1 (25th percentile) and Q3 (75th percentile).
3. Compute IQR = Q3 − Q1.
4. Determine lower bound = Q1 − 1.5 × IQR.
5. Determine upper bound = Q3 + 1.5 × IQR.
6. Cap, replace, or remove values outside these bounds.

C. Encoding Categorical Data

1. Identify categorical columns in the dataset.
2. Apply **Label Encoding** for ordinal categories.
3. Apply **One-Hot Encoding** for nominal categories using pd.get_dummies().
4. Merge encoded columns with the numerical dataset.

D. Final Dataset Preparation

1. Verify all missing values and outliers are handled.
2. Combine processed numerical and categorical features.
3. Use the cleaned dataset for machine learning model training.

Program

```
import numpy as np
import pandas as pd

# Sample dataset
data = {
   "Age": [25, 28, np.nan, 32, 100, 29],
   "Salary": [50000, np.nan, 45000, 52000, 51000, 700000],
   "Department": ["HR", "IT", "IT", np.nan, "Finance", "HR"]
}

df = pd.DataFrame(data)
print("Original Data:")
print(df)

# ----------------------------
# Handling Missing Data
# ----------------------------
df["Age"].fillna(df["Age"].mean(), inplace=True)
df["Salary"].fillna(df["Salary"].median(), inplace=True)
df["Department"].fillna(df["Department"].mode()[0], inplace=True)

# ----------------------------
# Outlier Treatment (IQR method)
# ----------------------------
Q1 = df["Salary"].quantile(0.25)
Q3 = df["Salary"].quantile(0.75)
IQR = Q3 - Q1
lower = Q1 - 1.5 * IQR
upper = Q3 + 1.5 * IQR

# Cap outliers
df["Salary"] = np.where(df["Salary"] > upper, upper,
            np.where(df["Salary"] < lower, lower, df["Salary"]))

# ----------------------------
# Encoding Categorical Data
# ----------------------------
df_encoded = pd.get_dummies(df, columns=["Department"])
print("\nProcessed Data:")
print(df_encoded)
```

OUTPUT

```
Original Data:

     Age    Salary Department
0   25.0   50000.0         HR
1   28.0       NaN         IT
2    NaN   45000.0         IT
3   32.0   52000.0        NaN
4  100.0   51000.0    Finance
5   29.0  700000.0         HR

Processed Data:

     Age   Salary  Department_Finance  Department_HR  Department_IT
0   25.0  50000.0               False           True          False
1   28.0  51000.0               False          False           True
2   42.8  48000.0               False          False           True
3   32.0  52000.0               False           True          False
4  100.0  51000.0                True          False          False
5   29.0  54000.0               False           True          False
```

RESULT

The dataset was successfully pre-processed by handling missing values, treating outliers, encoding categorical variables, and generating a clean dataset ready for machine learning.

Ex.No:2	**Implement Linear Regression to predict a continuous target variable and explore how Multiple Linear Regression can capture the relationship between multiple predictors and the target. Use GHI dataset to perform model training, validation, and performance evaluation.**

INTRODUCTION

Linear Regression is a supervised machine learning technique used to predict a continuous target variable by establishing a linear relationship between the input feature(s) and the output. In Simple Linear Regression, the model uses only one predictor variable, and it fits a straight-line equation of the form:

$$y = mX + c$$

Here, the slope (m) represents how the target changes with the predictor, while the intercept (c) indicates the starting value when the predictor is zero. However, real-world problems often involve more than one feature. Multiple Linear Regression extends this concept by using two or more predictor variables to capture more complex relationships. Its equation is:

$$y = b_0 + b_1X_1 + b_2X_2 + \cdots + b_nX_n$$

This allows the model to learn how each independent variable contributes to the target value.
In this experiment, the GHI (Global Hunger Index) dataset is used to train, validate, and evaluate both Simple Linear Regression and Multiple Linear Regression models to predict the GHI Score based on various health and nutrition indicators.

AIM

To implement Simple Linear Regression and Multiple Linear Regression techniques to predict the Global Hunger Index(GHI) using key hunger and nutrition indicators.

ALGORITHM

1. Imported the required Python libraries such as NumPy, Pandas, Matplotlib, and Scikit-learn.
2. Generated a synthetic Global Hunger Index dataset containing indicators such as undernourishment, child wasting, child stunting, and child mortality.
3. Computed the Global Hunger Index value using a weighted combination of the hunger indicators with added random noise.
4. Selected undernourishment as the independent variable and Global Hunger Index as the dependent variable for Simple Linear Regression.
5. Split the dataset into training and testing sets using an 80:20 ratio.
6. Trained the Simple Linear Regression model using the training dataset.
7. Predicted Global Hunger Index values for the test dataset and evaluated the model using MAE, MSE, and R^2 score.
8. Selected undernourishment, child wasting, child stunting, and child mortality as independent variables for Multiple Linear Regression.
9. Trained the Multiple Linear Regression model using the training dataset.

10. Predicted Global Hunger Index values for the test dataset and evaluated the model performance.

PROGRAM

```
# Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Step 1: Create Synthetic Global Hunger Index Dataset
np.random.seed(42)
data = pd.DataFrame({
    "Undernourishment": np.random.uniform(5, 40, 200),     # %
    "ChildWasting": np.random.uniform(2, 25, 200),         # %
    "ChildStunting": np.random.uniform(10, 50, 200),       # %
    "ChildMortality": np.random.uniform(1, 15, 200)        # %
})

# Global Hunger Index calculation (synthetic relationship)
data["GHI"] = (
    0.25 * data["Undernourishment"]
    + 0.25 * data["ChildWasting"]
    + 0.25 * data["ChildStunting"]
    + 0.25 * data["ChildMortality"]
    + np.random.normal(0, 1.5, 200)
)

# Display dataset
print("Synthetic Global Hunger Index Dataset:")
print(data.head())

# Step 2: Simple Linear Regression
X_simple = data[['Undernourishment']]
y = data['GHI']
X_train_s, X_test_s, y_train_s, y_test_s = train_test_split(
    X_simple, y, test_size=0.2, random_state=42
)

simple_lr = LinearRegression()
simple_lr.fit(X_train_s, y_train_s)
y_pred_s = simple_lr.predict(X_test_s)
```

```
print("\n--- Simple Linear Regression (Undernourishment → GHI) ---")
print("MAE:", mean_absolute_error(y_test_s, y_pred_s))
print("MSE:", mean_squared_error(y_test_s, y_pred_s))
print("R2 Score:", r2_score(y_test_s, y_pred_s))

# Step 3: Multiple Linear Regression
X_multi = data[['Undernourishment', 'ChildWasting', 'ChildStunting', 'ChildMortality']]

X_train_m, X_test_m, y_train_m, y_test_m = train_test_split(
    X_multi, y, test_size=0.2, random_state=42
)
multi_lr = LinearRegression()
multi_lr.fit(X_train_m, y_train_m)
y_pred_m = multi_lr.predict(X_test_m)

print("\n--- Multiple Linear Regression (All Indicators → GHI) ---")
print("MAE:", mean_absolute_error(y_test_m, y_pred_m))
print("MSE:", mean_squared_error(y_test_m, y_pred_m))
print("R2 Score:", r2_score(y_test_m, y_pred_m))

# Step 4: Plot Actual vs Predicted GHI
plt.figure()
plt.scatter(y_test_m, y_pred_m)
plt.xlabel("Actual Global Hunger Index")
plt.ylabel("Predicted Global Hunger Index")
plt.title("Actual vs Predicted Global Hunger Index")
plt.show()
```

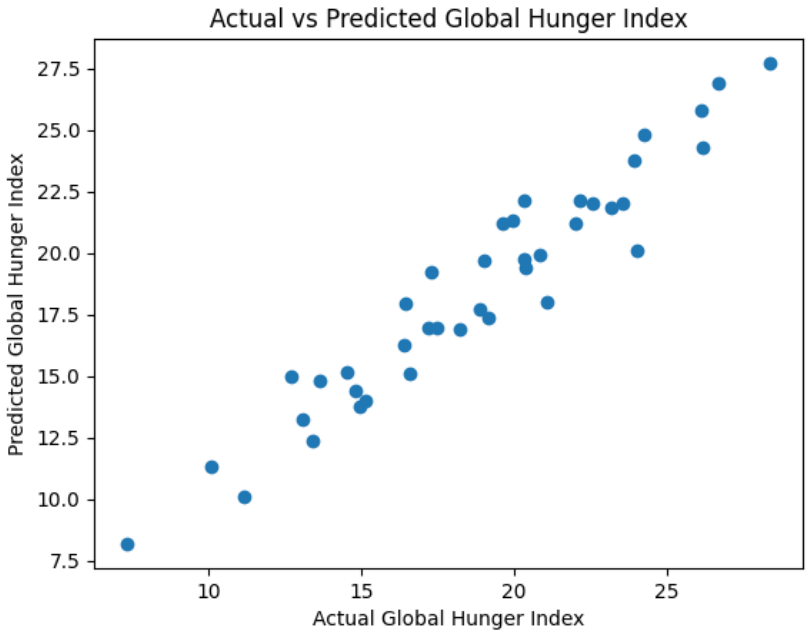
OUTPUT :

```
Synthetic Global Hunger Index Dataset:

   Undernourishment  ChildWasting  ChildStunting  ChildMortality        GHI
0         18.108904     16.766728      14.124955        3.365091  11.905708
1         38.275001      3.935219      46.102116        4.900265  24.010353
2         30.619788      5.717460      30.210095        3.478147  20.329409
3         25.953047     22.666746      43.058299        2.241835  25.498112
4         10.460652     15.947868      22.801984        2.688902  15.364632

--- Simple Linear Regression (Undernourishment → GHI) ---
MAE: 3.041983782620014
MSE: 14.956599744135037
R2 Score: 0.34734049973251757

--- Multiple Linear Regression (All Indicators → GHI) ---
MAE: 1.1059453765970535
MSE: 1.8504677196206836
R2 Score: 0.9192513433661731
```



RESULT

Thus, Simple Linear Regression and Multiple Linear Regression models were successfully implemented on the GHI dataset, and Multiple Linear Regression produced better prediction accuracy by considering multiple input variables.