1. **Implement a program to create a data set**

from sklearn.datasets import make\_classification

import pandas as pd

# Step 1: Create synthetic data

X, y = make\_classification(

n\_samples=1000, # number of data points

n\_features=10, # total number of features

n\_informative=5, # number of informative features

n\_redundant=2, # number of redundant features

n\_classes=2, # binary classification (0 or 1)

random\_state=42 # seed for reproducibility

)

# Step 2: Convert to pandas DataFrame

feature\_names = [f'feature\_{i}' for i in range(X.shape[1])]

df = pd.DataFrame(X, columns=feature\_names)

df['target'] = y

# Step 3: Save dataset to a CSV file

df.to\_csv('synthetic\_classification\_dataset.csv', index=False)

print("Dataset created and saved as 'synthetic\_classification\_dataset.csv'")

**output:-**

| feature\_0 | feature\_1 | feature\_2 | feature\_3 | ... | feature\_9 | target |
| --- | --- | --- | --- | --- | --- | --- |
| -1.062 | 0.365 | 0.547 | 0.148 | ... | -0.312 | 0 |
| 1.237 | -0.789 | -0.421 | -0.912 | ... | 1.021 | 1 |
| -0.214 | 0.598 | 0.301 | 0.408 | ... | -0.689 | 0 |
| ... | ... | ... | ... | ... | ... | ... |

1. **implement a program to read and write the data sets.**

import pandas as pd

# ----------------------

# WRITE DATASET TO CSV

# ----------------------

# Sample data: dictionary of lists

data = {

'feature1': [1.5, 2.3, 3.1],

'feature2': [0, 1, 0],

'target': [1, 0, 1]

}

# Convert to DataFrame

df = pd.DataFrame(data)

# Write to CSV file

df.to\_csv('sample\_dataset.csv', index=False)

print("✅ Dataset written to 'sample\_dataset.csv'")

# ----------------------

# READ DATASET FROM CSV

#

# Read the dataset from CSV

read\_df = pd.read\_csv('sample\_dataset.csv')

# Show contents

print("\n✅ Dataset read from 'sample\_dataset.csv':")

print(read\_df)

**output:-**

✅ Dataset written to 'sample\_dataset.csv'

✅ Dataset read from 'sample\_dataset.csv':

feature1 feature2 target

0 1.5 0 1

1 2.3 1 0

2 3.1 0 1

**3.write a program to impute missing values in machine learning**

import pandas as pd

import numpy as np

from sklearn.impute import SimpleImputer

# ------------------------

# Step 1: Create sample data with missing values

# ------------------------

data = {

'feature1': [1.2, 2.4, np.nan, 4.5, 5.1],

'feature2': [3.3, np.nan, 1.1, 4.4, np.nan],

'target': [0, 1, 0, 1, 0]

}

df = pd.DataFrame(data)

print("Original dataset with missing values:\n", df)

# ------------------------

# Step 2: Impute missing values

# ------------------------

# Create an imputer that replaces missing values with the mean

imputer = SimpleImputer(strategy='mean')

# Apply imputer to feature columns only

df[['feature1', 'feature2']] = imputer.fit\_transform(df[['feature1', 'feature2']])

print("\nDataset after imputation:\n", df)

**output:**-

Original dataset with missing values:

feature1 feature2 target

0 1.2 3.3 0

1 2.4 NaN 1

2 NaN 1.1 0

3 4.5 4.4 1

4 5.1 NaN 0

Dataset after imputation:

feature1 feature2 target

0 1.200 3.3 0

1 2.400 2.933 1

2 3.300 1.1 0

3 4.500 4.4 1

4 5.100 2.933 0

1. **write a Program to handle outliers**

**import numpy as np**

**import pandas as pd**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**from sklearn.datasets import load\_boston**

**from sklearn.linear\_model import LinearRegression**

**from sklearn.model\_selection import train\_test\_split**

**from sklearn.metrics import mean\_squared\_error**

**# Load dataset**

**boston = load\_boston()**

**df = pd.DataFrame(boston.data, columns=boston.feature\_names)**

**df['TARGET'] = boston.target**

**print("Original Data Shape:", df.shape)**

**# Visualize original data**

**sns.boxplot(data=df[['CRIM', 'ZN', 'B', 'TARGET']])**

**plt.title("Boxplot before handling outliers")**

**plt.show()**

**### Method 1: Handle Outliers Using Z-Score**

**from scipy import stats**

**def remove\_outliers\_zscore(data, threshold=3):**

**z\_scores = np.abs(stats.zscore(data.select\_dtypes(include=[np.number])))**

**mask = (z\_scores < threshold).all(axis=1)**

**return data[mask]**

**# Uncomment to use Z-Score method**

**# df\_cleaned = remove\_outliers\_zscore(df)**

**### Method 2: Handle Outliers Using IQR**

**def remove\_outliers\_iqr(data):**

**Q1 = data.quantile(0.25)**

**Q3 = data.quantile(0.75)**

**IQR = Q3 - Q1**

**return data[~((data < (Q1 - 1.5 \* IQR)) | (data > (Q3 + 1.5 \* IQR))).any(axis=1)]**

**# Use IQR Method by default**

**df\_cleaned = remove\_outliers\_iqr(df)**

**print("Data Shape after outlier removal:", df\_cleaned.shape)**

**# Visualize cleaned data**

**sns.boxplot(data=df\_cleaned[['CRIM', 'ZN', 'B', 'TARGET']])**

**plt.title("Boxplot after handling outliers")**

**plt.show()**

**# Machine learning example (simple linear regression)**

**X = df\_cleaned.drop('TARGET', axis=1)**

**y = df\_cleaned['TARGET']**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)**

**model = LinearRegression()**

**model.fit(X\_train, y\_train)**

**predictions = model.predict(X\_test)**

**print("Mean Squared Error:", mean\_squared\_error(y\_test, predictions))**