Machine Learning Lab

Experiment 2: Data Preprocessing in Machine Learning

For CSE Department, Semester 06

Course Code: U23CM6L2

Compiled by
Mohammed Ufraan

March 24, 2026

Experiment 2

Aim: For a given set of training data examples stored in a CSV file, demonstrate Data Preprocessing in Machine Learning with the following steps.

a) **Getting the Dataset**

Description: This step involves identifying and locating the dataset file (`.csv`) that contains the training data.

Input Format:

- File: `.csv` containing features and target variable.
- Example file: `data.csv`

> **Dataset Download:** The dataset file `data.csv` used in this experiment is available for download at `https://github.com/ufraaan/ml-lab-experiments`. Datasets for all lab experiments are hosted there.

Original Dataset:

Location	Area_sqft	Bedrooms	Price_Lakhs
Urban	800.0	2	40
Suburban	1000.0	2	55
Urban	1200.0	3	65
Rural	NaN	3	60
Suburban	1800.0	4	100

Algorithm:

1. Locate the CSV file.
2. Ensure it has a header row for feature names.
3. Confirm the file path is correct for importing.

Expected Output:

- Dataset file is ready to import.
- File contents:

```
Location  Area_sqft  Bedrooms  Price_Lakhs
Urban     800.0      2         40
Suburban  1000.0     2         55
Urban     1200.0     3         65
Rural     NaN        3         60
Suburban  1800.0     4         100
```

Viva Questions:

1. How do you know if the dataset is suitable for machine learning?
2. Why is having a header row important?
3. Can datasets be in formats other than CSV?

b) **Importing Libraries**

Description: Importing required Python libraries for data manipulation, preprocessing, and splitting datasets.

> **Dataset Download:** Ensure all libraries below are installed before running the experiment. The dataset and any additional setup resources for all experiments are available at `https://github.com/ufraaan/ml-lab-experiments`.

Input Format:

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import LabelEncoder, OneHotEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
```

Listing 1: Importing required libraries

Algorithm:

1. Import `pandas` for dataset handling.
2. Import `NumPy` for numerical operations.
3. Import `sklearn.preprocessing` for encoding and scaling.
4. Import `train_test_split` for splitting the dataset.
5. Import `SimpleImputer` for handling missing data.

Viva Questions:

1. Why do we use pandas for datasets?
2. What is the difference between NumPy arrays and pandas DataFrames?
3. Why is sklearn used in preprocessing?

---

c) **Importing Dataset**

Description: Load the CSV file into Python and separate features and target variables.

Input Format:

- CSV file (`data.csv`) with multiple columns.

Algorithm:

1. Read CSV using `pd.read_csv('data.csv')`.
2. Store features in `X` (all columns except last).
3. Store target variable in `y` (last column).

```
import pandas as pd

dataset = pd.read_csv('data.csv')
X = dataset.iloc[:, :-1].values   # All columns except last (features)
y = dataset.iloc[:, -1].values    # Last column (target variable)

print("Features (X):\n", X)
print("Target (y):\n", y)
```

Listing 2: Importing the dataset and separating features and target

Viva Questions:

1. How do you access features and target separately?
2. Why do we separate X and y?
3. Can we import Excel files similarly?

---

d) **Finding Missing Data**

Description: Identify missing or null values in the dataset and replace them with suitable values.

Input Format:

- Dataset with missing values in numeric columns (e.g., `Area_sqft`).

Algorithm:

1. Detect missing values using `isnull()` or `np.isnan()`.
2. Use `SimpleImputer` to replace missing values with mean/median/mode.
3. Apply the transformation on the feature columns with missing data.

```
from sklearn.impute import SimpleImputer
import numpy as np

imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
imputer.fit(X[:, 1:3])              # Fit on numeric columns
X[:, 1:3] = imputer.transform(X[:, 1:3])

print("Dataset after handling missing values:\n", X)
```

Listing 3: Handling missing data using SimpleImputer

Viva Questions:

1. Why is handling missing data important?
2. What are the different strategies to fill missing data?
3. Can we remove rows with missing data instead of filling?

---

e) **Encoding Categorical Data**

Description: Convert categorical data (e.g., `Location`) into numeric form using encoding.

Input Format:

- Dataset with categorical column `Location` containing values: `Urban`, `Suburban`, `Rural`.

Algorithm:

1. Apply `OneHotEncoder` for features with multiple categories.
2. Apply `LabelEncoder` for the target variable if categorical.
3. Transform the categorical columns into numeric arrays.

```python
from sklearn.preprocessing import OneHotEncoder, LabelEncoder
from sklearn.compose import ColumnTransformer

# OneHotEncode the 'Location' column (index 0)
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [0])],
                       remainder='passthrough')
X = ct.fit_transform(X)

print("Encoded Feature Matrix:\n", X)
```

Listing 4: Encoding categorical data using OneHotEncoder and LabelEncoder

Viva Questions:

1. What is label encoding?
2. What is one-hot encoding?
3. Why should we avoid label encoding for features with no ordinal relationship?

---

f) **Splitting Dataset into Training and Test Set**

Description: Divide the dataset into training and test sets to evaluate model performance.

Input Format:

- Feature matrix `X` and target vector `y`.

Algorithm:

1. Use `train_test_split(X, y, test_size=0.2, random_state=0)`.
2. Assign 80% of data to the training set and 20% to the test set.

```python
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.2,
                                                    random_state=0)

print("Training Set (X_train):\n", X_train)
print("Test Set (X_test):\n", X_test)
```

Listing 5: Splitting dataset into training and test sets

Viva Questions:

1. Why do we split data into training and test sets?
2. What does `random_state` do?
3. Can we use different test sizes?

---

g) **Feature Scaling**

Description: Normalize or standardize numerical features to bring all features to the same scale.

Input Format:

- Numerical columns like `Area_sqft` and `Bedrooms`.

Algorithm:

1. Initialize `StandardScaler()`.
2. Fit the scaler to the training set and transform it.
3. Transform the test set using the same scaler.

```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
X_train = sc.fit_transform(X_train)   # Fit on train, then transform
X_test  = sc.transform(X_test)        # Only transform test (no refit)

print("Scaled Training Set:\n", X_train)
print("Scaled Test Set:\n", X_test)
```

Listing 6: Feature scaling using StandardScaler

Viva Questions:

1. What is feature scaling?
2. Why should test data be scaled using the training scaler?
3. What is the difference between normalization and standardization?

Final Expected Output (after complete pipeline):

```
Model Training Completed
Predicted House Price:
Location: Urban, Area: 1400 sqft, Bedrooms: 3
Predicted Price = 75.20 Lakhs
```