Data Cleaning and Preprocessing Techniques in Python

Data cleaning and preprocessing are critical steps in data analysis and machine learning workflows. They ensure that the dataset is accurate, complete, and formatted correctly for analysis.

---

1. Import Required Libraries

python
Copy code

```
import pandas as pd
import numpy as np
```

---

2. Load the Dataset

python
Copy code

```
# Example: Load data from a CSV file
data = pd.read_csv('data.csv')
print(data.head())
```

---

3. Handling Missing Values

Check for Missing Data

python
Copy code

```
# Check for missing values
print(data.isnull().sum())

# Percentage of missing values
print(data.isnull().mean() * 100)
```

Handle Missing Values

Remove Missing Data

python
Copy code

```
# Drop rows with missing values
data_cleaned = data.dropna()
```

```
# Drop columns with missing values
data_cleaned = data.dropna(axis=1)
```

1.

Impute Missing Values
python
Copy code

```
# Fill with a specific value
data['Column1'].fillna(0, inplace=True)

# Fill with the mean, median, or mode
data['Column1'].fillna(data['Column1'].mean(), inplace=True)
data['Column1'].fillna(data['Column1'].median(), inplace=True)
data['Column1'].fillna(data['Column1'].mode()[0], inplace=True)
```

2.

Forward/Backward Fill
python
Copy code

```
data['Column1'].fillna(method='ffill', inplace=True)  # Forward fill
data['Column1'].fillna(method='bfill', inplace=True)  # Backward
fill
```

3.

---

4. Dealing with Outliers

Detect Outliers
Using Boxplot
python
Copy code

```
import matplotlib.pyplot as plt
data.boxplot(column='Column1')
plt.show()
```

1.

Using Z-Score
python
Copy code

```
from scipy.stats import zscore
data['zscore'] = zscore(data['Column1'])
```

```
outliers = data[data['zscore'].abs() > 3]
print(outliers)
```

2.

Using IQR (Interquartile Range)
python
Copy code

```
Q1 = data['Column1'].quantile(0.25)
Q3 = data['Column1'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outliers = data[(data['Column1'] < lower_bound) | (data['Column1'] >
upper_bound)]
print(outliers)
```

3.

Handle Outliers
Remove Outliers
python
Copy code

```
data = data[(data['Column1'] >= lower_bound) & (data['Column1'] <=
upper_bound)]
```

1.

Replace Outliers
python
Copy code

```
data['Column1'] = np.where(data['Column1'] > upper_bound,
upper_bound, data['Column1'])
```

2.

---

5. Encoding Categorical Variables

One-Hot Encoding
python
Copy code

```
# Create dummy variables
data = pd.get_dummies(data, columns=['CategoryColumn'],
drop_first=True)
```

Label Encoding

python
Copy code

```
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
data['CategoryColumn'] =
label_encoder.fit_transform(data['CategoryColumn'])
```

---

6. Scaling and Normalization

Scaling

python
Copy code

```
from sklearn.preprocessing import MinMaxScaler, StandardScaler

# Min-Max Scaling (0 to 1)
scaler = MinMaxScaler()
data[['Column1', 'Column2']] = scaler.fit_transform(data[['Column1',
'Column2']])

# Standardization (mean = 0, std = 1)
scaler = StandardScaler()
data[['Column1', 'Column2']] = scaler.fit_transform(data[['Column1',
'Column2']])
```

Normalization

python
Copy code

```
from sklearn.preprocessing import Normalizer

normalizer = Normalizer()
data[['Column1', 'Column2']] =
normalizer.fit_transform(data[['Column1', 'Column2']])
```

---

7. Handling Duplicates

Check for Duplicates

python
Copy code
```
# Find duplicates
print(data.duplicated().sum())
```

Remove Duplicates

python
Copy code
```
# Remove duplicate rows
data = data.drop_duplicates()
```

---

8. Feature Engineering

Create New Features

python
Copy code
```
# Combine features
data['Total'] = data['Column1'] + data['Column2']

# Extract date components
data['Year'] = pd.to_datetime(data['Date']).dt.year
data['Month'] = pd.to_datetime(data['Date']).dt.month
data['Day'] = pd.to_datetime(data['Date']).dt.day
```

Feature Transformation

python
Copy code
```
# Log Transformation
data['LogColumn'] = np.log1p(data['Column1'])

# Polynomial Features
data['Squared'] = data['Column1'] ** 2
```

---

9. Splitting Data into Train and Test Sets

python
Copy code
```
from sklearn.model_selection import train_test_split
```

```
# Define features and target
X = data.drop(columns=['Target'])
y = data['Target']

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

---

10. Save and Load Processed Data

Save Data

python
Copy code

```
data.to_csv('cleaned_data.csv', index=False)
```

Load Data

python
Copy code

```
data = pd.read_csv('cleaned_data.csv')
```