**1. Data Inspection**

* **Objective:** Understand the structure and contents of the dataset.

import pandas as pd

# Load the dataset

data = pd.read\_csv('crop\_data.csv')

# Inspect the first few rows

print(data.head())

# Check the summary of the dataset

print(data.info())

# Check for missing values

print(data.isnull().sum())

**2. Handling Missing Values**

* **Objective:** Address missing values through imputation or removal.

# Fill missing values with a specific value or method

data['Fertilizer\_Usage'] = data['Fertilizer\_Usage'].fillna(data['Fertilizer\_Usage'].median()) # Median imputation

# Or, drop rows with missing values

data\_cleaned = data.dropna() # Drops rows with any missing values

# For categorical data, you might fill missing values with the most frequent value

data['Soil\_Type'] = data['Soil\_Type'].fillna(data['Soil\_Type'].mode()[0])

**3. Handling Outliers**

* **Objective:** Identify and handle outliers to avoid skewed analysis.

# Define a function to detect outliers using Z-score

from scipy import stats

def remove\_outliers(df, column\_name):

z\_scores = stats.zscore(df[column\_name])

abs\_z\_scores = np.abs(z\_scores)

return df[(abs\_z\_scores < 3)]

# Remove outliers from a specific column

data\_cleaned = remove\_outliers(data\_cleaned, 'Rainfall')

**4. Standardizing and Normalizing Data**

* **Objective:** Ensure consistent formats and ranges for numerical features.

from sklearn.preprocessing import StandardScaler

# Standardize numerical features

scaler = StandardScaler()

data\_cleaned[['Fertilizer\_Usage', 'Pesticide\_Usage', 'pH', 'Area', 'Rainfall']] = scaler.fit\_transform(

data\_cleaned[['Fertilizer\_Usage', 'Pesticide\_Usage', 'pH', 'Area', 'Rainfall']]

)

**5. Encoding Categorical Variables**

* **Objective:** Convert categorical variables into numerical format.

python

Copy code

# Convert categorical variables to dummy variables (one-hot encoding)

data\_cleaned = pd.get\_dummies(data\_cleaned, columns=['Soil\_Type'], drop\_first=True) # Drop first to avoid multicollinearity

**6. Removing Duplicate Records**

* **Objective:** Ensure that there are no duplicate rows in the dataset.

# Remove duplicate rows

data\_cleaned = data\_cleaned.drop\_duplicates()

**7. Data Type Conversion**

* **Objective:** Convert data types to appropriate formats for analysis.

# Convert data types if necessary

data\_cleaned['Area'] = data\_cleaned['Area'].astype(float)

data\_cleaned['Crop'] = data\_cleaned['Crop'].astype('category')

**8. Checking for Consistency and Validity**

* **Objective:** Ensure that all data entries adhere to expected formats and constraints.

# Check for any inconsistencies in categorical columns

print(data\_cleaned['Soil\_Type'].value\_counts())

# Validate ranges for numerical features

print(data\_cleaned[['pH', 'Rainfall']].describe())

**Complete Example Code**

Here’s a combined code snippet implementing the data cleanup steps:

import pandas as pd

import numpy as np

from sklearn.preprocessing import StandardScaler

from scipy import stats

# Load the dataset

data = pd.read\_csv('crop\_data.csv')

# Inspect the dataset

print(data.head())

print(data.info())

print(data.isnull().sum())

# Handle missing values

data['Fertilizer\_Usage'] = data['Fertilizer\_Usage'].fillna(data['Fertilizer\_Usage'].median())

data['Soil\_Type'] = data['Soil\_Type'].fillna(data['Soil\_Type'].mode()[0])

data\_cleaned = data.dropna()

# Handle outliers

def remove\_outliers(df, column\_name):

z\_scores = stats.zscore(df[column\_name])

abs\_z\_scores = np.abs(z\_scores)

return df[(abs\_z\_scores < 3)]

data\_cleaned = remove\_outliers(data\_cleaned, 'Rainfall')

# Standardize numerical features

scaler = StandardScaler()

data\_cleaned[['Fertilizer\_Usage', 'Pesticide\_Usage', 'pH', 'Area', 'Rainfall']] = scaler.fit\_transform(

data\_cleaned[['Fertilizer\_Usage', 'Pesticide\_Usage', 'pH', 'Area', 'Rainfall']]

)

# Encode categorical variables

data\_cleaned = pd.get\_dummies(data\_cleaned, columns=['Soil\_Type'], drop\_first=True)

# Remove duplicate records

data\_cleaned = data\_cleaned.drop\_duplicates()

# Convert data types

data\_cleaned['Area'] = data\_cleaned['Area'].astype(float)

data\_cleaned['Crop'] = data\_cleaned['Crop'].astype('category')

# Check for consistency

print(data\_cleaned['Soil\_Type'].value\_counts())

print(data\_cleaned[['pH', 'Rainfall']].describe())

**Final Considerations**

* **Documentation:** Keep a record of all changes made during the cleanup process for reproducibility and transparency.
* **Iteration:** Data cleanup is an iterative process. Regularly revisit the data as you gain more insights or as new data becomes available.

This detailed strategy ensures your data is clean, consistent, and ready for robust analysis.