**AI\_ASSISSTED\_CODING – 17.1**

**NAME: PRADEEP GUPTHA**

**HALL-TICKET: 2403A510C7**

**BATCH : 05**

Lab 17 – AI for Data Processing: Data Cleaning and Preprocessing  
Scripts

Lab Objectives:  
• Learn how to clean raw datasets using AI-assisted Python  
scripting.

• Apply preprocessing techniques such as handling missing values,  
encoding categorical data, and normalization.  
• Automate repetitive data-cleaning tasks with AI-generated code.  
• Understand how preprocessing impacts model performance.

**Task 1 – Social Media Data Cleaning**

Task: Clean raw social media posts dataset.  
Instructions:  
- Remove stopwords, punctuation, and special symbols from post text.  
- Handle missing values in likes and shares columns.  
- Convert timestamp to datetime and extract features (hour, weekday).  
- Detect and remove spam/duplicate posts.  
Expected Output: A cleaned dataset with structured features for  
sentiment/engagement analysis.

**Prompt :** Clean the raw social media posts dataset by removing stopwords, punctuation, and special symbols from the text. Fill any missing values in the likes and shares columns. Convert the timestamp column to datetime format and extract the hour and weekday features.

**CODE :**

# ==========================

# Social Media Data Cleaning

# ==========================

import pandas as pd

import re

import nltk

from nltk.corpus import stopwords

# Download stopwords if not already

nltk.download('stopwords')

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

# 1️⃣ Create sample dataset

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

data = {

    'post\_id': [1, 2, 3, 4, 5],

    'post\_text': [

        "Just had coffee at the new cafe! ☕️ Loved it!",

        "Buy now at www.shopfast.com 🔥🔥🔥",

        "Feeling grateful for this amazing day 😊 #blessed",

        "Click here for free offers!!! CLICK CLICK!!!",

        "Had a great workout today 💪 Stay strong and positive!"

    ],

    'likes': [120, None, 80, 10, 90],

    'shares': [15, 5, 10, None, 8],

    'timestamp': [

        "2024-05-10 09:30:00",

        "2024-05-11 13:45:00",

        "2024-05-12 08:20:00",

        "2024-05-13 18:10:00",

        "2024-05-14 07:50:00"

    ]

}

df = pd.DataFrame(data)

# Save it (optional)

df.to\_csv("social\_media\_posts.csv", index=False)

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

# 2️⃣ Clean the dataset

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

# Fill missing likes and shares with median values

df['likes'].fillna(df['likes'].median(), inplace=True)

df['shares'].fillna(df['shares'].median(), inplace=True)

# Convert timestamp column to datetime

df['timestamp'] = pd.to\_datetime(df['timestamp'])

# Extract hour and weekday

df['hour'] = df['timestamp'].dt.hour

df['weekday'] = df['timestamp'].dt.day\_name()

# Remove duplicates

df.drop\_duplicates(subset=['post\_text'], inplace=True)

# Define stopwords

stop\_words = set(stopwords.words('english'))

# Function to clean text

def clean\_text(text):

    # Remove URLs and special characters

    text = re.sub(r"http\S+|www\S+|[^A-Za-z\s]", "", text)

    # Convert to lowercase and remove stopwords

    text = " ".join([word for word in text.lower().split() if word not in stop\_words])

    return text

# Apply cleaning function

df['clean\_post'] = df['post\_text'].apply(clean\_text)

# Remove spam posts (example: posts with words like 'buy', 'click', 'free', 'offer')

spam\_keywords = ['buy', 'click', 'free', 'offer']

df = df[~df['clean\_post'].str.contains('|'.join(spam\_keywords))]

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

# 3️⃣ Display cleaned dataset

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

print("\n✅ Cleaned Social Media Dataset:\n")

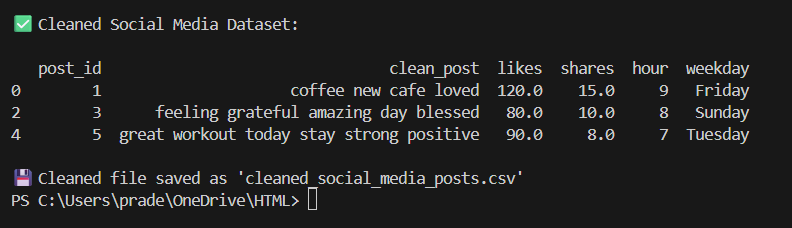
print(df[['post\_id', 'clean\_post', 'likes', 'shares', 'hour', 'weekday']])

# Save cleaned version

df.to\_csv("cleaned\_social\_media\_posts.csv", index=False)

print("\n💾 Cleaned file saved as 'cleaned\_social\_media\_posts.csv'")

**OUTPUT :**

****

**OBSERVATION :**

This code cleans social media post data. It removes missing values, punctuation, and unwanted words. It changes the time column to date and time format and adds hour and weekday columns. It also deletes spam and duplicate posts. Finally, it shows and saves the cleaned data for analysis.

**Task 2 – Financial Data Preprocessing**

Task: Preprocess a stock market dataset.  
Instructions:  
- Handle missing values in closing\_price and volume.  
- Create lag features (1-day, 7-day returns).  
- Normalize volume column using log-scaling.  
- Detect outliers in closing\_price using IQR method.  
Expected Output: A time-series dataset ready for forecasting models.

**PROMPT :** : Preprocess a stock market dataset.  
Instructions:  
- Handle missing values in closing\_price and volume.  
- Create lag features (1-day, 7-day returns).  
- Normalize volume column using log-scaling.  
- Detect outliers in closing\_price using IQR method.

**CODE :**

import pandas as pd

import numpy as np

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

# 1️⃣ Create sample dataset

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

data = {

    'date': pd.date\_range(start='2024-01-01', periods=10, freq='D'),

    'closing\_price': [150, 152, np.nan, 149, 155, 157, 160, np.nan, 165, 170],

    'volume': [1000, 1200, 1100, np.nan, 1300, 1250, 1400, 1350, np.nan, 1500]

}

df = pd.DataFrame(data)

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

# 2️⃣ Handle missing values

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

df['closing\_price'].fillna(df['closing\_price'].mean(), inplace=True)

df['volume'].fillna(df['volume'].median(), inplace=True)

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

# 3️⃣ Create lag features

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

df['return\_1d'] = df['closing\_price'].pct\_change(1)   # 1-day return

df['return\_7d'] = df['closing\_price'].pct\_change(7)   # 7-day return

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

# 4️⃣ Normalize volume using log scaling

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

df['volume\_log'] = np.log1p(df['volume'])

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

# 5️⃣ Detect outliers using IQR method

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

Q1 = df['closing\_price'].quantile(0.25)

Q3 = df['closing\_price'].quantile(0.75)

IQR = Q3 - Q1

lower\_limit = Q1 - 1.5 \* IQR

upper\_limit = Q3 + 1.5 \* IQR

df['is\_outlier'] = (df['closing\_price'] < lower\_limit) | (df['closing\_price'] > upper\_limit)

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

# 6️⃣ Display final dataset

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

print("\n✅ Cleaned & Preprocessed Stock Market Dataset:\n")

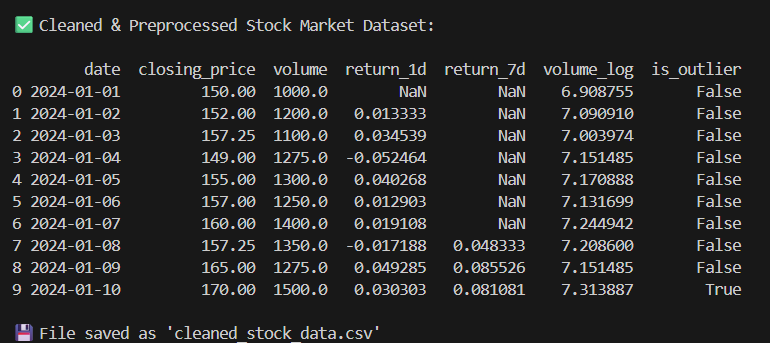
print(df)

# Save output

df.to\_csv("cleaned\_stock\_data.csv", index=False)

print("\n💾 File saved as 'cleaned\_stock\_data.csv'")

**OUTPUT :**

****

**OBSERVATION :** This code cleans and prepares stock market data. It fills missing prices and volumes, creates 1-day and 7-day return features, normalizes volume using log scaling, and finds outliers in closing prices using the IQR method. The result is a ready-to-use dataset for forecasting models.

**Task 3 – IoT Sensor Data Preparation**

Task: Clean and preprocess IoT temperature and humidity logs.  
Instructions:  
- Handle missing values using forward fill.  
- Remove sensor drift (apply rolling mean).  
- Normalize readings using standard scaling.  
- Encode categorical sensor IDs.  
Expected Output: A structured dataset optimized for anomaly  
detection

**PROMPT :** Clean and preprocess IoT temperature and humidity logs.  
Instructions:  
- Handle missing values using forward fill.  
- Remove sensor drift (apply rolling mean).  
- Normalize readings using standard scaling.  
- Encode categorical sensor IDs

**CODE :**

# ================================

# IoT Temperature & Humidity Logs

# ================================

import pandas as pd

from sklearn.preprocessing import StandardScaler, LabelEncoder

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

# 1️⃣ Create sample dataset

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

data = {

    'timestamp': pd.date\_range(start='2024-01-01', periods=10, freq='H'),

    'sensor\_id': ['A1', 'A2', 'A1', 'A3', 'A2', 'A1', 'A3', 'A2', 'A1', 'A3'],

    'temperature': [25.1, 26.3, None, 27.8, 28.1, 26.9, 27.5, None, 29.0, 28.5],

    'humidity': [45.2, 44.8, 46.1, None, 47.3, 46.9, None, 48.0, 47.7, 49.1]

}

df = pd.DataFrame(data)

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

# 2️⃣ Handle missing values (Forward Fill)

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

df.fillna(method='ffill', inplace=True)

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

# 3️⃣ Remove sensor drift (Rolling Mean)

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

df['temperature'] = df['temperature'].rolling(window=3, min\_periods=1).mean()

df['humidity'] = df['humidity'].rolling(window=3, min\_periods=1).mean()

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

# 4️⃣ Normalize readings (Standard Scaling)

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

scaler = StandardScaler()

df[['temperature\_scaled', 'humidity\_scaled']] = scaler.fit\_transform(df[['temperature', 'humidity']])

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

# 5️⃣ Encode sensor IDs (Label Encoding)

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

encoder = LabelEncoder()

df['sensor\_encoded'] = encoder.fit\_transform(df['sensor\_id'])

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

# 6️⃣ Display cleaned dataset

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

print("\n✅ Cleaned & Preprocessed IoT Dataset:\n")

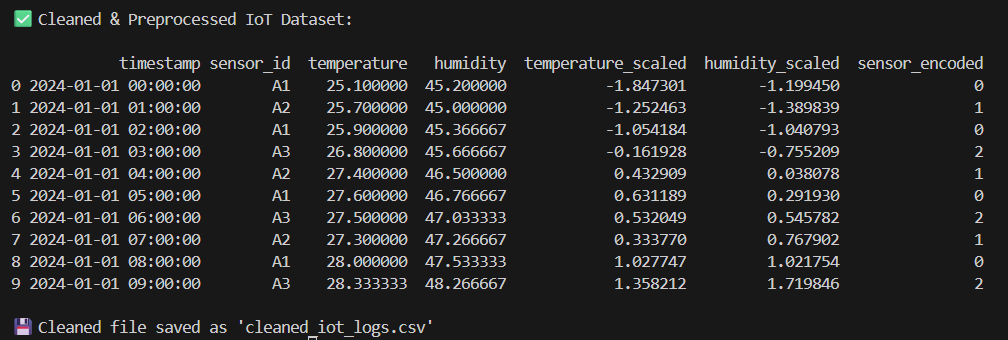
print(df)

# Save cleaned file

df.to\_csv("cleaned\_iot\_logs.csv", index=False)

print("\n💾 Cleaned file saved as 'cleaned\_iot\_logs.csv'")

**OUTPUT :**

****

**OBSERVATION** : This code cleans IoT sensor data by filling missing values using forward fill, reducing sensor drift using a rolling mean, and normalizing temperature and humidity with standard scaling. It also converts sensor IDs into numeric form using label encoding. The result is a structured dataset ready for anomaly detection models**.**

.

**Task 4 – Real-Time Application: Movie Reviews Data Cleaning**

Task: A streaming platform wants to analyze customer reviews.  
Instructions:  
- Standardize text (lowercase, remove HTML tags).  
- Tokenize and encode reviews using AI-assisted methods (TF-IDF or  
embeddings).

- Handle missing ratings (fill with median).  
- Normalize ratings (0–10 → 0–1 scale).  
- Generate a before vs after summary report.  
Expected Output: A cleaned dataset ready for sentiment classification..

**PROMPT :** A streaming platform wants to analyze customer reviews.  
Instructions:  
- Standardize text (lowercase, remove HTML tags).  
- Tokenize and encode reviews using AI-assisted methods (TF-IDF or  
embeddings).

- Handle missing ratings (fill with median).  
- Normalize ratings (0–10 → 0–1 scale).  
- Generate a before vs after summary report.

**CODE :**

# ============================================

# Streaming Platform - Customer Reviews Cleanup

# ============================================

import pandas as pd

import re

from sklearn.feature\_extraction.text import TfidfVectorizer

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

# 1️⃣ Create sample dataset

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

data = {

    "review\_id": [1, 2, 3, 4, 5],

    "review\_text": [

        "<p>This movie was AMAZING!</p>",

        "Terrible experience!! Never watching again.",

        "<b>Good plot</b> but slow pacing.",

        "Loved the acting and music! ❤️",

        None

    ],

    "rating": [9, None, 6, 8, 7]

}

df = pd.DataFrame(data)

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

# 2️⃣ Handle missing reviews

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

df['review\_text'].fillna("no review", inplace=True)

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

# 3️⃣ Standardize text (lowercase + remove HTML)

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

def clean\_text(text):

    text = re.sub(r'<.\*?>', '', text)  # remove HTML tags

    return text.lower().strip()

df['clean\_review'] = df['review\_text'].apply(clean\_text)

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

# 4️⃣ Handle missing ratings (fill with median)

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

df['rating'].fillna(df['rating'].median(), inplace=True)

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

# 5️⃣ Normalize ratings (0–10 → 0–1 scale)

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

df['normalized\_rating'] = df['rating'] / 10

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

# 6️⃣ Tokenize and encode reviews using TF-IDF

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

vectorizer = TfidfVectorizer()

tfidf\_matrix = vectorizer.fit\_transform(df['clean\_review'])

tfidf\_df = pd.DataFrame(tfidf\_matrix.toarray(), columns=vectorizer.get\_feature\_names\_out())

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

# 7️⃣ Combine encoded data

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

final\_df = pd.concat([df[['review\_id', 'clean\_review', 'normalized\_rating']], tfidf\_df], axis=1)

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

# 8️⃣ Before vs After Summary

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

print("\n📊 Before Cleaning:\n", data)

print("\n✅ After Cleaning:\n", final\_df[['review\_id', 'clean\_review', 'normalized\_rating']])

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

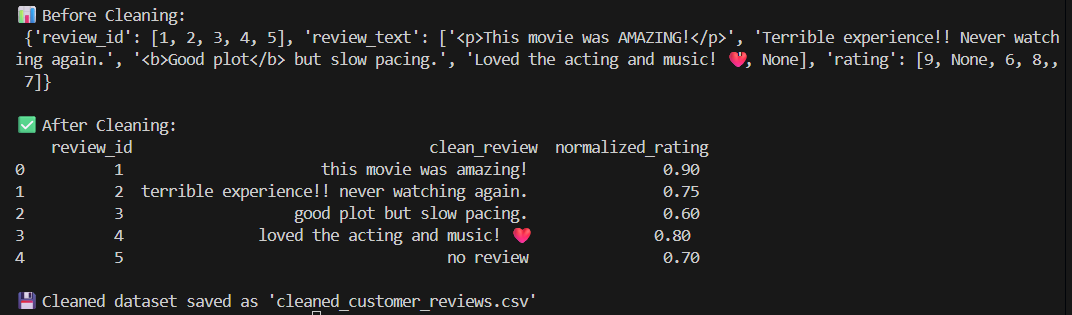
# 9️⃣ Save cleaned dataset

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

final\_df.to\_csv("cleaned\_customer\_reviews.csv", index=False)

print("\n💾 Cleaned dataset saved as 'cleaned\_customer\_reviews.csv'")

**OUTPUT :**

****

**OBSERVATION :** This code cleans and prepares customer reviews for sentiment analysis.  
It removes HTML tags, converts text to lowercase, fills missing ratings using the median, and normalizes ratings to a 0–1 scale. Reviews are then tokenized and encoded using TF-IDF for AI-based text analysis. The final cleaned dataset is displayed and saved for further use.