

LAB ASSIGNMENT-17.2

PROGRAMME : BTECH-CSE
SPECIALIZATION : AIML
NAME : NAVADEEP MUNUGOTI
ENROLL.ID : 2403A52015
BATCH.NO : 02

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: To clean a raw dataset of social media posts to prepare it for sentiment and engagement analysis. The cleaning process should include removing stopwords, punctuation, and special characters from the post text. It should also handle missing values in the likes and shares columns appropriately. Additionally, I want to convert the timestamp field into a proper datetime format and extract useful features like the hour of posting and the day of the week. Finally, the script should detect and eliminate any spam or duplicate posts. The expected result is a structured and cleaned dataset ready for further analysis.

Code:

```
C: > Users > thota > Desktop > test_edge_cases.py > ...
1 import pandas as pd
2 import re
3 import nltk
4 from nltk.corpus import stopwords
5 from datetime import datetime
6
7 # Download stopwords if not present
8 try:
9     nltk.data.find('corpora/stopwords')
10 except LookupError:
11     nltk.download('stopwords')
12
13 stop_words = set(stopwords.words('english'))
14
15 def clean_text(text):
16     if pd.isna(text):
17         return ""
18     # Lowercase
19     text = text.lower()
20     # Remove URLs
21     text = re.sub(r'http\S+', '', text)
22     # Remove mentions and hashtags
23     text = re.sub(r'[@#]\w+', '', text)
24     # Remove punctuation and special characters
25     text = re.sub(r'[^\w\s]', '', text)
26     # Remove extra spaces
27     text = re.sub(r'\s+', ' ', text).strip()
28     # Remove stopwords
29     words = text.split()
30     words = [word for word in words if word not in stop_words]
31     return ' '.join(words)
32
33 def clean_dataset(df):
```

```
C: > Users > thota > Desktop > test_edge_cases.py > ...
1 import pandas as pd
2 import re
3 import nltk
4 from nltk.corpus import stopwords
5 from datetime import datetime
6
7 # Download stopwords if not present
8 try:
9     nltk.data.find('corpora/stopwords')
10 except LookupError:
11     nltk.download('stopwords')
12
13 stop_words = set(stopwords.words('english'))
14
15 def clean_text(text):
16     if pd.isna(text):
17         return ""
18     # Lowercase
19     text = text.lower()
20     # Remove URLs
21     text = re.sub(r'http\S+', '', text)
22     # Remove mentions and hashtags
23     text = re.sub(r'@[#]\w+', '', text)
24     # Remove punctuation and special characters
25     text = re.sub(r'[^w\s]', '', text)
26     # Remove extra spaces
27     text = re.sub(r'\s+', ' ', text).strip()
28     # Remove stopwords
29     words = text.split()
30     words = [word for word in words if word not in stop_words]
31     return ' '.join(words)
32
33 def clean_dataset(df):
```

```
C:\> Users > thota > Desktop > test_edge_cases.py > ...
33     def clean_dataset(df):
34         # Clean post_text
35         df['cleaned_text'] = df['post_text'].apply(clean_text)
36
37         # Handle missing values in likes and shares
38         df['likes'] = df['likes'].fillna(0).astype(int)
39         df['shares'] = df['shares'].fillna(0).astype(int)
40
41         # Convert timestamp to datetime
42         df['timestamp'] = pd.to_datetime(df['timestamp'], errors='coerce')
43
44         # Extract hour and weekday
45         df['hour'] = df['timestamp'].dt.hour
46         df['weekday'] = df['timestamp'].dt.day_name()
47
48         # Remove duplicates based on cleaned_text
49         df = df.drop_duplicates(subset=['cleaned_text'])
50
51         # Remove spam: if cleaned_text is empty or too short (<5 chars)
52         df = df[df['cleaned_text'].str.len() >= 5]
53
54     return df
55
56     # Test 1: All missing values
57     print("Testing: All missing values")
58     df1 = pd.DataFrame({
59         'post_text': [None, None, None],
60         'likes': [None, None, None],
61         'shares': [None, None, None],
62         'timestamp': [None, None, None]
63     })
64     df1_clean = clean_dataset(df1)
65     print("Rows after cleaning:", len(df1_clean))
```

```

C:\> Users\thota\Desktop> test_edge_cases.py > ...
110     |     shares : 1, 2, 3,
111     |     'timestamp': ["2023-10-01 12:00:00", "10/02/2023 13:00", "2023-10-03T14:00:00"]
112 }
113 df5_clean = clean_dataset(df5)
114 print("Rows after cleaning:", len(df5_clean))
115 print(df5_clean[['timestamp', 'hour', 'weekday']])
116
117 # Test 6: Non-English stopwords (script uses English, so should still clean)
118 print("\nTesting: Non-English text")
119 df6 = pd.DataFrame({
120     |     'post_text': ["Hola mundo", "Bonjour le monde"],
121     |     'likes': [1, 2],
122     |     'shares': [1, 2],
123     |     'timestamp': ["2023-10-01 12:00:00", "2023-10-01 12:00:00"]
124 })
125 df6_clean = clean_dataset(df6)
126 print("Rows after cleaning:", len(df6_clean))
127 print(df6_clean)
128
129 # Test 7: Larger dataset (generate 1000 rows)
130 print("\nTesting: Larger dataset (1000 rows)")
131 import numpy as np
132 np.random.seed(42)
133 df7 = pd.DataFrame({
134     |     'post_text': ["Post " + str(i) + " with some text!" for i in range(1000)],
135     |     'likes': np.random.randint(0, 1000, 1000),
136     |     'shares': np.random.randint(0, 500, 1000),
137     |     'timestamp': pd.date_range("2023-10-01", periods=1000, freq='H')
138 })
139 df7_clean = clean_dataset(df7)
140 print("Rows after cleaning:", len(df7_clean))
141
142 print("\nAll edge case tests completed.")

```

The screenshot shows a Jupyter Notebook interface with a terminal tab active. The command `PS C:\Users\thota\Desktop> python clean_data.py` is run, followed by the output of the cleaned dataset:

```

Cleaned Dataset:
   post_text  likes  shares      timestamp    cleaned_text  hour  weekday
0  This is a great post! #awesome  100     20  2023-10-01 14:30:00  great post  14  Sunday
1  Wow, amazing content!!! @user     0     15  2023-10-02 09:15:00  wow amazing content  9  Monday
2  Check out this link: http://example.com  200     0  2023-10-03 18:45:00  check link  18  Tuesday
3  Spam spam spam!!!                 5     2  2023-10-04 12:00:00  spam spam spam  12  Wednesday
4  Another post with punctuation!!!  150     30  2023-10-05 16:20:00 another post punctuation  16  Thursday
5  Short                           Short     0     5  2023-10-06 08:10:00  short  8  Friday

```

Observation:

➤ Stopwords, Punctuation, and Special Symbols Removal

Check if the text preprocessing pipeline removes common stopwords (e.g., "the", "is", "at"), punctuation marks, and special characters that do not contribute to sentiment or engagement analysis.

➤ **Handling Missing Values In Likes and Shares**

Observe how the script deals with missing or null values in numeric columns. Does it impute with mean/median, fill with zeros, or drop rows? The strategy should be consistent and context-aware.

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: To preprocess a stock market dataset to prepare it for time-series forecasting. The preprocessing steps should include handling missing values in the closing_price and volume columns, creating lag-based features such as 1-day and 7-day returns, and applying log-scaling to normalize the volume data. Additionally, I want to detect and remove outliers in the closing_price column using the Interquartile Range (IQR) method. The final output should be a clean and structured dataset suitable for building forecasting models.

Code:

```
C: > Users > thota > Desktop > stock_data.csv > data
1 |date,closing_price,volume
2 |2023-01-01,100.5,1000000
3 |2023-01-02,101.2,1100000
4 |2023-01-03,,1200000
5 |2023-01-04,102.8,1300000
6 |2023-01-05,103.1,1400000
7 |2023-01-06,102.9,1500000
8 |2023-01-07,104.5,1600000
9 |2023-01-08,105.0,1700000
10 |2023-01-09,106.2,1800000
11 |2023-01-10,107.0,1900000
12 |2023-01-11,108.5,2000000
13 |2023-01-12,109.2,2100000
14 |2023-01-13,110.0,2200000
15 |2023-01-14,111.5,2300000
16 |2023-01-15,112.0,2400000
17 |2023-01-16,113.5,2500000
18 |2023-01-17,114.0,2600000
19 |2023-01-18,115.5,2700000
20 |2023-01-19,116.0,2800000
21 |2023-01-20,117.5,2900000
22 |2023-01-21,118.0,3000000
23 |2023-01-22,119.5,3100000
24 |2023-01-23,120.0,3200000
25 |2023-01-24,121.5,3300000
26 |2023-01-25,122.0,3400000
27 |2023-01-26,123.5,3500000
28 |2023-01-27,124.0,3600000
29 |2023-01-28,125.5,3700000
30 |2023-01-29,126.0,3800000
31 |2023-01-30,127.5,3900000
32 |2023-01-31,128.0,4000000
33 |2023-02-01,129.5,4100000
```

```
C: > Users > thota > Desktop > test_edge_cases.py > ...
1  import pandas as pd
2  import re
3  import nltk
4  from nltk.corpus import stopwords
5  from datetime import datetime
6
7  # Download stopwords if not present
8  try:
9      nltk.data.find('corpora/stopwords')
10 except LookupError:
11     nltk.download('stopwords')
12
13 stop_words = set(stopwords.words('english'))
14
15 def clean_text(text):
16     if pd.isna(text):
17         return ""
18     # Lowercase
19     text = text.lower()
20     # Remove URLs
21     text = re.sub(r'http\S+', '', text)
22     # Remove mentions and hashtags
23     text = re.sub(r'@[#]\w+', '', text)
24     # Remove punctuation and special characters
25     text = re.sub(r'[^\w\s]', '', text)
26     # Remove extra spaces
27     text = re.sub(r'\s+', ' ', text).strip()
28     # Remove stopwords
29     words = text.split()
30     words = [word for word in words if word not in stop_words]
31     return ' '.join(words)
32
33 def clean_dataset(df):
```

```
C:\> Users > thota > Desktop > 📁 test_edge_cases.py > ...
31     return ' '.join(words)
32
33 def clean_dataset(df):
34     # Clean post_text
35     df['cleaned_text'] = df['post_text'].apply(clean_text)
36
37     # Handle missing values in likes and shares
38     df['likes'] = df['likes'].fillna(0).astype(int)
39     df['shares'] = df['shares'].fillna(0).astype(int)
40
41     # Convert timestamp to datetime
42     df['timestamp'] = pd.to_datetime(df['timestamp'], errors='coerce')
43
44     # Extract hour and weekday
45     df['hour'] = df['timestamp'].dt.hour
46     df['weekday'] = df['timestamp'].dt.day_name()
47
48     # Remove duplicates based on cleaned_text
49     df = df.drop_duplicates(subset=['cleaned_text'])
50
51     # Remove spam: if cleaned_text is empty or too short (<5 chars)
52     df = df[df['cleaned_text'].str.len() >= 5]
53
54     return df
55
56 # Test 1: All missing values
57 print("Testing: All missing values")
58 df1 = pd.DataFrame({
59     'post_text': [None, None, None],
60     'likes': [None, None, None],
61     'shares': [None, None, None],
62     'timestamp': [None, None, None]
63 })
```

```
C:\> Users > thota > Desktop > 📡 test_edge_cases.py > ...
64     df1_clean = clean_dataset(df1)
65     print("Rows after cleaning:", len(df1_clean))
66     print(df1_clean)
67
68     # Test 2: All duplicates
69     print("\nTesting: All duplicates")
70     df2 = pd.DataFrame({
71         'post_text': ["This is a test post", "This is a test post", "This is a test post"]
72         'likes': [10, 10, 10],
73         'shares': [5, 5, 5],
74         'timestamp': ["2023-10-01 12:00:00", "2023-10-01 12:00:00", "2023-10-01 12:00:00"]
75     })
76     df2_clean = clean_dataset(df2)
77     print("Rows after cleaning:", len(df2_clean))
78     print(df2_clean)
79
80     # Test 3: Empty texts
81     print("\nTesting: Empty texts")
82     df3 = pd.DataFrame({
83         'post_text': ["", "", "Hello world"],
84         'likes': [1, 2, 3],
85         'shares': [1, 2, 3],
86         'timestamp': ["2023-10-01 12:00:00", "2023-10-01 12:00:00", "2023-10-01 12:00:00"]
87     })
88     df3_clean = clean_dataset(df3)
89     print("Rows after cleaning:", len(df3_clean))
90     print(df3_clean)
91
92     # Test 4: Very long texts
93     print("\nTesting: Very long texts")
94     long_text = "This is a very long post " * 100
95     df4 = pd.DataFrame({
96         'post_text': [long_text, "Short"]
```

```
C: > Users > thota > Desktop > preprocess_stock.py > ...
1 import pandas as pd
2 import numpy as np
3
4 # Load data
5 df = pd.read_csv('stock_data.csv')
6
7 # Convert date to datetime
8 df['date'] = pd.to_datetime(df['date'])
9
10 # Handle missing values in closing_price and volume
11 df['closing_price'] = df['closing_price'].ffill()
12 df['volume'] = df['volume'].ffill()
13
14 # Create lag features: 1-day and 7-day returns
15 df['return_1d'] = df['closing_price'].pct_change(1)
16 df['return_7d'] = df['closing_price'].pct_change(7)
17
18 # Normalize volume using log-scaling
19 df['volume_log'] = np.log(df['volume'] + 1) # +1 to avoid log(0)
20
21 # Detect outliers in closing_price using IQR method
```

```
C: > Users > thota > Desktop > preprocess_stock.py > ...
1 import pandas as pd
2 import numpy as np
3
4 # Load data
5 df = pd.read_csv('stock_data.csv')
6
7 # Convert date to datetime
8 df['date'] = pd.to_datetime(df['date'])
9
10 # Handle missing values in closing_price and volume
11 df['closing_price'] = df['closing_price'].ffill()
12 df['volume'] = df['volume'].ffill()
13
14 # Create lag features: 1-day and 7-day returns
15 df['return_1d'] = df['closing_price'].pct_change(1)
16 df['return_7d'] = df['closing_price'].pct_change(7)
17
18 # Normalize volume using log-scaling
19 df['volume_log'] = np.log(df['volume'] + 1) # +1 to avoid log(0)
20
21 # Detect outliers in closing_price using IQR method
22 Q1 = df['closing_price'].quantile(0.25)
23 Q3 = df['closing_price'].quantile(0.75)
24 IQR = Q3 - Q1
25 lower_bound = Q1 - 1.5 * IQR
26 upper_bound = Q3 + 1.5 * IQR
27 df['outlier'] = (df['closing_price'] < lower_bound) | (df['closing_price'] > upper_bound)
28
29 # Save preprocessed dataset
30 df.to_csv('preprocessed_stock_data.csv', index=False)
31
32 print("Financial data preprocessing completed. Preprocessed data saved to preprocessed_stock_data.csv")
33 print("\nPreprocessed Dataset (first 10 rows):")
```

```

PS C:\Users\thota\Desktop> python preprocess_stock.py
      Date       Open   Close  Volume  Price_Chg  Price_Pct  IsUp
0  2023-01-02  101.2  1100000  0.006965     NaN  13.910822  False
1  2023-01-03  101.2  1200000  0.000000     NaN  13.997833  False
2  2023-01-04  102.8  1300000  0.015810     NaN  14.077876  False
3  2023-01-05  103.1  1400000  0.002918     NaN  14.151984  False
4  2023-01-06  102.9  1500000 -0.001940     NaN  14.220976  False
5  2023-01-07  104.5  1600000  0.015549     NaN  14.285515  False
6  2023-01-08  105.0  1700000  0.004785  0.044776  14.346139  False
7  2023-01-09  106.2  1800000  0.011429  0.049407  14.403298  False
8  2023-01-10  107.0  1900000  0.007533  0.057312  14.457365  False
PS C:\Users\thota\Desktop>

```

Observation:

➤ **Handling Missing Values**

- Check how missing values in closing_price and volume are treated.
- Are they dropped, forward/backward filled, or imputed with statistical measures like mean or median?
- Ensure the method preserves time-series integrity.

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: To clean and preprocess a dataset containing IoT sensor logs that record temperature and humidity readings. The preprocessing steps should include handling missing values using forward fill, applying a rolling mean to smooth out sensor drift, and normalizing the readings using standard scaling. Additionally, the sensor IDs—if categorical—should be encoded appropriately for modeling. The final output should be a well-structured dataset optimized for anomaly detection tasks.

Code:

```
C: > Users > thota > Desktop > 📜 iot_data.csv > ⏷ data
1 timestamp,sensor_id,temperature,humidity
2 2023-01-01 00:00:00,Sensor_A,25.5,60.2
3 2023-01-01 01:00:00,Sensor_A,26.0,61.0
4 2023-01-01 02:00:00,Sensor_A,,62.5
5 2023-01-01 03:00:00,Sensor_A,27.2,63.1
6 2023-01-01 04:00:00,Sensor_B,24.8,59.8
7 2023-01-01 05:00:00,Sensor_B,25.1,60.5
8 2023-01-01 06:00:00,Sensor_B,25.3,61.2
9 2023-01-01 07:00:00,Sensor_B,25.7,62.0
10 2023-01-01 08: Col 1: timestamp 6.5,63.5
11 2023-01-01 09:00:00,Sensor_C,27.0,64.0
12 2023-01-01 10:00:00,Sensor_C,27.5,64.5
13 2023-01-01 11:00:00,Sensor_C,28.0,65.0
14 2023-01-01 12:00:00,Sensor_A,28.5,65.5
15 2023-01-01 13:00:00,Sensor_A,29.0,66.0
16 2023-01-01 14:00:00,Sensor_A,29.5,66.5
17 2023-01-01 15:00:00,Sensor_A,30.0,67.0
18 2023-01-01 16:00:00,Sensor_B,30.5,67.5
19 2023-01-01 17:00:00,Sensor_B,31.0,68.0
20 2023-01-01 18:00:00,Sensor_B,31.5,68.5
21 2023-01-01 19:00:00,Sensor_B,32.0,69.0
22 2023-01-01 20:00:00,Sensor_C,32.5,69.5
23 2023-01-01 21:00:00,Sensor_C,33.0,70.0
24 2023-01-01 22:00:00,Sensor_C,33.5,70.5
25 2023-01-01 23:00:00,Sensor_C,34.0,71.0
26 2023-01-02 00:00:00,Sensor_A,34.5,71.5
27 2023-01-02 01:00:00,Sensor_A,35.0,72.0
28 2023-01-02 02:00:00,Sensor_A,35.5,72.5
29 2023-01-02 03:00:00,Sensor_A,36.0,73.0
30 2023-01-02 04:00:00,Sensor_B,36.5,73.5
31 2023-01-02 05:00:00,Sensor_B,37.0,74.0
32 2023-01-02 06:00:00,Sensor_B,37.5,74.5
33 2023-01-02 07:00:00,Sensor_B,38.0,75.0
```

```

C:\> Users > thota > Desktop > preprocess_iot.py > ...
1 import pandas as pd
2 import numpy as np
3
4 # Load data
5 df = pd.read_csv('iot_data.csv')
6
7 # Convert timestamp to datetime
8 df['timestamp'] = pd.to_datetime(df['timestamp'])
9
10 # Handle missing values using forward fill
11 df['temperature'] = df['temperature'].ffill()
12 df['humidity'] = df['humidity'].ffill()
13
14 # Remove sensor drift (apply rolling mean with window=5)
15 df['temperature_smooth'] = df['temperature'].rolling(window=5, min_periods=1).mean()
16 df['humidity_smooth'] = df['humidity'].rolling(window=5, min_periods=1).mean()
17
18 # Normalize readings using standard scaling (manual)
19 df['temperature_scaled'] = (df['temperature_smooth'] - df['temperature_smooth'].mean()) / df['temperature_smooth'].std()
20 df['humidity_scaled'] = (df['humidity_smooth'] - df['humidity_smooth'].mean()) / df['humidity_smooth'].std()
21
22 # Encode categorical sensor IDs (manual mapping)
23 sensor_mapping = {'Sensor_A': 0, 'Sensor_B': 1, 'Sensor_C': 2}
24 df['sensor_id_encoded'] = df['sensor_id'].map(sensor_mapping)
25
26 # Save preprocessed dataset
27 df.to_csv('preprocessed_iot_data.csv', index=False)
28
29 print("IoT sensor data preprocessing completed. Preprocessed data saved to preprocessed_iot_data.csv")
30 print("\nPreprocessed Dataset (first 10 rows):")
31 print(df.head(10).to_string(index=False))
32

```

```

PS C:\Users\thota\Desktop> python preprocess_iot.py
969          1
2023-01-01 06:00:00 Sensor_B    25.3    61.2      25.680000    61.420000   -1.245151   -1.306
117          1
2023-01-01 07:00:00 Sensor_B    25.7    62.0      25.620000    61.320000   -1.254505   -1.320
748          1
2023-01-01 08:00:00 Sensor_C    26.5    63.5      25.480000    61.400000   -1.276330   -1.309
043          2
2023-01-01 09:00:00 Sensor_C    27.0    64.0      25.920000    62.240000   -1.207736   -1.186
141          2

```

Observation:➤ **Missing Value Handling**

- Confirm that missing temperature and humidity readings are filled using forward fill (ffill), which maintains temporal continuity.
- Check if the method is applied consistently across all sensors and time intervals.

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

Prompt: A streaming service is aiming to enhance its understanding of customer feedback by cleaning and preparing movie review data for analysis. The task involves several preprocessing steps: first, standardize the review text by converting it to lowercase and stripping out any HTML tags. Next, tokenize the text and apply AI-driven encoding techniques such as TF-IDF or semantic embeddings to convert the reviews into numerical representations. For the ratings, identify any missing values and replace them with the median rating. Then, normalize all ratings from a 0–10 scale to a 0–1 scale to ensure consistency. Finally, produce a summary report comparing the dataset before and after cleaning, highlighting changes in review structure, rating distribution, and sample entries.

Code:

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Movie Reviews Data Cleaning</title>
6      <script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@4.10.0/dist/tf.min.js"></script>
7      <style>
8          body { font-family: Arial, sans-serif; margin: 20px; }
9          h2 { color: #2c3e50; }
10         pre { background: #f4f4f4; padding: 10px; border-radius: 5px; }
11     </style>
12 </head>
13 <body>
14
15 <h2>🎬 Movie Reviews Data Cleaning</h2>
16
17 <div id="before"></div>
18 <div id="after"></div>
19
20 <script>
21     // Sample raw data
22     const reviews = [
23         { text: "<p>Great movie! Loved it.</p>", rating: 9 },
24         { text: "<div>Terrible plot, bad acting.</div>", rating: 2 },
25         { text: "<span>Okay-ish. Not bad.</span>", rating: null },
26         { text: "Amazing visuals! <br> Story was weak.", rating: 7 },
27         { text: "Meh. Could be better.", rating: 5 }
28     ];
29
30     // Step 1: Standardize text
31     function cleanText(html) {
32         const doc = new DOMParser().parseFromString(html, 'text/html');
33         return doc.body.textContent.toLowerCase();
34     }
```

```

1  import pandas as pd
2  import numpy as np
3  import re
4  from sklearn.feature_extraction.text import TfidfVectorizer
5
6  # Sample raw data
7  data = {
8      'review': [
9          "<p>Great movie! Loved it.</p>",
10         "<div>Terrible plot, bad acting.</div>",
11         "<span>Okay-ish. Not bad.</span>",
12         "Amazing visuals! <br> Story was weak.",
13         "Meh. Could be better."
14     ],
15     'rating': [9, 2, None, 7, 5]
16 }
17
18 # Create DataFrame
19 df = pd.DataFrame(data)
20
21 # Step 1: Standardize text
22 def clean_html(text):
23     return re.sub(r'<.*?>', '', text).lower()
24
25 df['cleaned_review'] = df['review'].apply(clean_html)
26
27 # Step 2: Tokenize and encode using TF-IDF
28 vectorizer = TfidfVectorizer()
29 tfidf_matrix = vectorizer.fit_transform(df['cleaned_review'])
30 tfidf_df = pd.DataFrame(tfidf_matrix.toarray(), columns=vectorizer.get_feature_names_out())
31
32 # Step 3: Handle missing ratings
33 median_rating = df['rating'].median()
34 df['rating_filled'] = df['rating'].fillna(median_rating)

```

```

24
25 df['cleaned_review'] = df['review'].apply(clean_html)
26
27 # Step 2: Tokenize and encode using TF-IDF
28 vectorizer = TfidfVectorizer()
29 tfidf_matrix = vectorizer.fit_transform(df['cleaned_review'])
30 tfidf_df = pd.DataFrame(tfidf_matrix.toarray(), columns=vectorizer.get_feature_names_out())
31
32 # Step 3: Handle missing ratings
33 median_rating = df['rating'].median()
34 df['rating_filled'] = df['rating'].fillna(median_rating)
35
36 # Step 4: Normalize ratings (0-10 → 0-1)
37 df['normalized_rating'] = df['rating_filled'] / 10
38
39 # Step 5: Combine and export to CSV
40 final_df = pd.concat([df[['review', 'cleaned_review', 'normalized_rating']], tfidf_df], axis=1)
41 final_df.to_csv('cleaned_movie_reviews.csv', index=False)
42
43 print("✅ Cleaned dataset saved as 'cleaned_movie_reviews.csv'")

```

```
PS C:\Users\thota> PS C:\Users\thota> python clean_review.py  
>>  CSV file 'cleaned_reviews.csv' created successfully.
```

Observation:

- **Before:** Review texts contained inconsistent casing (mix of uppercase and lowercase), and many included HTML tags *which added noise to the data*.
- **After:** *All texts were converted to lowercase, and HTML tags were successfully removed, resulting in cleaner and more uniform textual data.*