LAB ASSIGNMENT 17.2

Program : B. Tech (CSE)

Specialization : AIML

Course Title : AI Assisted coding

Semester : III

Academic Session : 2025-2026

Name of Student : Thota Akhileshwari

Enrollment No : 2403a52010

Batch No. : 02

Date : 9-11-2025

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 > ...

```python
import pandas as pd
import re
import nltk
from nltk.corpus import stopwords
from datetime import datetime

# Download stopwords if not present
try:
    nltk.data.find('corpora/stopwords')
except LookupError:
    nltk.download('stopwords')

stop_words = set(stopwords.words('english'))

def clean_text(text):
    if pd.isna(text):
        return ""
    # Lowercase
    text = text.lower()
    # Remove URLs
    text = re.sub(r'http\S+', '', text)
    # Remove mentions and hashtags
    text = re.sub(r'[@#]\w+', '', text)
    # Remove punctuation and special characte
    text = re.sub(r'[^\w\s]', '', text)
    # Remove extra spaces
    text = re.sub(r'\s+', ' ', text).strip()
    # Remove stopwords
    words = text.split()
    words = [word for word in words if word n
    return ' '.join(words)

def clean_dataset(df):
```

C: > Users > thota > Desktop > test_edge_cases.py > ...

```python
import pandas as pd
import re
import nltk
from nltk.corpus import stopwords
from datetime import datetime

# Download stopwords if not present
try:
    nltk.data.find('corpora/stopwords')
except LookupError:
    nltk.download('stopwords')

stop_words = set(stopwords.words('english'))

def clean_text(text):
    if pd.isna(text):
        return ""
    # Lowercase
    text = text.lower()
    # Remove URLs
    text = re.sub(r'http\S+', '', text)
    # Remove mentions and hashtags
    text = re.sub(r'[@#]\w+', '', text)
    # Remove punctuation and special characters
    text = re.sub(r'[^\w\s]', '', text)
    # Remove extra spaces
    text = re.sub(r'\s+', ' ', text).strip()
    # Remove stopwords
    words = text.split()
    words = [word for word in words if word not in stop_words]
    return ' '.join(words)

def clean_dataset(df):
```

C: > Users > thota > Desktop > test_edge_cases.py > ...

```python
def clean_dataset(df):
    # Clean post_text
    df['cleaned_text'] = df['post_text'].apply(clean_text)

    # Handle missing values in likes and shares
    df['likes'] = df['likes'].fillna(0).astype(int)
    df['shares'] = df['shares'].fillna(0).astype(int)

    # Convert timestamp to datetime
    df['timestamp'] = pd.to_datetime(df['timestamp'], errors='coerce')

    # Extract hour and weekday
    df['hour'] = df['timestamp'].dt.hour
    df['weekday'] = df['timestamp'].dt.day_name()

    # Remove duplicates based on cleaned_text
    df = df.drop_duplicates(subset=['cleaned_text'])

    # Remove spam: if cleaned_text is empty or too short (<5 chars)
    df = df[df['cleaned_text'].str.len() >= 5]

    return df

# Test 1: All missing values
print("Testing: All missing values")
df1 = pd.DataFrame({
    'post_text': [None, None, None],
    'likes': [None, None, None],
    'shares': [None, None, None],
    'timestamp': [None, None, None]
})
df1_clean = clean_dataset(df1)
print("Rows after cleaning:", len(df1_clean))
```

C: > Users > thota > Desktop > test_edge_cases.py > ...

```
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.")
```

Ln 1, Col 1 Spaces: 4 UTF-8 CRLF { } Python Python 3.11 (64-bit) BLACKBOX Autocomplete

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\thota\Desktop> python clean_data.py

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

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

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

C: > Users > thota > Desktop > test_edge_cases.py > ...

```python
import pandas as pd
import re
import nltk
from nltk.corpus import stopwords
from datetime import datetime

# Download stopwords if not present
try:
    nltk.data.find('corpora/stopwords')
except LookupError:
    nltk.download('stopwords')

stop_words = set(stopwords.words('english'))

def clean_text(text):
    if pd.isna(text):
        return ""
    # Lowercase
    text = text.lower()
    # Remove URLs
    text = re.sub(r'http\S+', '', text)
    # Remove mentions and hashtags
    text = re.sub(r'[@#]\w+', '', text)
    # Remove punctuation and special characters
    text = re.sub(r'[^\w\s]', '', text)
    # Remove extra spaces
    text = re.sub(r'\s+', ' ', text).strip()
    # Remove stopwords
    words = text.split()
    words = [word for word in words if word not in stop_words]
    return ' '.join(words)

def clean_dataset(df):
```

C: > Users > thota > Desktop > test_edge_cases.py > ...

```python
    return ' '.join(words)

def clean_dataset(df):
    # Clean post_text
    df['cleaned_text'] = df['post_text'].apply(clean_text)

    # Handle missing values in likes and shares
    df['likes'] = df['likes'].fillna(0).astype(int)
    df['shares'] = df['shares'].fillna(0).astype(int)

    # Convert timestamp to datetime
    df['timestamp'] = pd.to_datetime(df['timestamp'], errors='coerce')

    # Extract hour and weekday
    df['hour'] = df['timestamp'].dt.hour
    df['weekday'] = df['timestamp'].dt.day_name()

    # Remove duplicates based on cleaned_text
    df = df.drop_duplicates(subset=['cleaned_text'])

    # Remove spam: if cleaned_text is empty or too short (<5 chars)
    df = df[df['cleaned_text'].str.len() >= 5]

    return df

# Test 1: All missing values
print("Testing: All missing values")
df1 = pd.DataFrame({
    'post_text': [None, None, None],
    'likes': [None, None, None],
    'shares': [None, None, None],
    'timestamp': [None, None, None]
})
```

C: > Users > thota > Desktop > test_edge_cases.py > ...

```python
df1_clean = clean_dataset(df1)
print("Rows after cleaning:", len(df1_clean))
print(df1_clean)

# Test 2: All duplicates
print("\nTesting: All duplicates")
df2 = pd.DataFrame({
    'post_text': ["This is a test post", "This is a test post", "This is a test post"],
    'likes': [10, 10, 10],
    'shares': [5, 5, 5],
    'timestamp': ["2023-10-01 12:00:00", "2023-10-01 12:00:00", "2023-10-01 12:00:00"]
})
df2_clean = clean_dataset(df2)
print("Rows after cleaning:", len(df2_clean))
print(df2_clean)

# Test 3: Empty texts
print("\nTesting: Empty texts")
df3 = pd.DataFrame({
    'post_text': ["", "   ", "Hello world"],
    'likes': [1, 2, 3],
    'shares': [1, 2, 3],
    'timestamp': ["2023-10-01 12:00:00", "2023-10-01 12:00:00", "2023-10-01 12:00:00"]
})
df3_clean = clean_dataset(df3)
print("Rows after cleaning:", len(df3_clean))
print(df3_clean)

# Test 4: Very long texts
print("\nTesting: Very long texts")
long_text = "This is a very long post " * 100
df4 = pd.DataFrame({
    'post_text': [long_text, "Short"],
```

C: > Users > thota > Desktop > preprocess_stock.py > ...

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

# Load data
df = pd.read_csv('stock_data.csv')

# Convert date to datetime
df['date'] = pd.to_datetime(df['date'])

# Handle missing values in closing_price and volume
df['closing_price'] = df['closing_price'].ffill()
df['volume'] = df['volume'].ffill()

# Create lag features: 1-day and 7-day returns
df['return_1d'] = df['closing_price'].pct_change(1)
df['return_7d'] = df['closing_price'].pct_change(7)

# Normalize volume using log-scaling
df['volume_log'] = np.log(df['volume'] + 1)  # +1 to avoid log(0)
```

C: > Users > thota > Desktop > preprocess_stock.py > ...

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

# Load data
df = pd.read_csv('stock_data.csv')

# Convert date to datetime
df['date'] = pd.to_datetime(df['date'])

# Handle missing values in closing_price and volume
df['closing_price'] = df['closing_price'].ffill()
df['volume'] = df['volume'].ffill()

# Create lag features: 1-day and 7-day returns
df['return_1d'] = df['closing_price'].pct_change(1)
df['return_7d'] = df['closing_price'].pct_change(7)

# Normalize volume using log-scaling
df['volume_log'] = np.log(df['volume'] + 1)  # +1 to avoid log(0)

# Detect outliers in closing_price using IQR method
Q1 = df['closing_price'].quantile(0.25)
Q3 = df['closing_price'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
df['outlier'] = (df['closing_price'] < lower_bound) | (df['closing_price'] > upper_bound)

# Save preprocessed dataset
df.to_csv('preprocessed_stock_data.csv', index=False)

print("Financial data preprocessing completed. Preprocessed data saved to preprocessed_stock_data.csv")
print("\nPreprocessed Dataset (first 10 rows):")
```

```
PS C:\Users\thota\Desktop> python preprocess_stock.py
2023-01-02         101.2 1100000   0.006965        NaN   13.910822    False
2023-01-03         101.2 1200000   0.000000        NaN   13.997833    False
2023-01-04         102.8 1300000   0.015810        NaN   14.077876    False
2023-01-05         103.1 1400000   0.002918        NaN   14.151984    False
2023-01-06         102.9 1500000  -0.001940        NaN   14.220976    False
2023-01-07         104.5 1600000   0.015549        NaN   14.285515    False
2023-01-08         105.0 1700000   0.004785   0.044776   14.346139    False
2023-01-09         106.2 1800000   0.011429   0.049407   14.403298    False
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

```
timestamp,sensor_id,temperature,humidity
2023-01-01 00:00:00,Sensor_A,25.5,60.2
2023-01-01 01:00:00,Sensor_A,26.0,61.0
2023-01-01 02:00:00,Sensor_A,,62.5
2023-01-01 03:00:00,Sensor_A,27.2,63.1
2023-01-01 04:00:00,Sensor_B,24.8,59.8
2023-01-01 05:00:00,Sensor_B,25.1,60.5
2023-01-01 06:00:00,Sensor_B,25.3,61.2
2023-01-01 07:00:00,Sensor_B,25.7,62.0
2023-01-01 08:00:00,Sensor_B,26.5,63.5
2023-01-01 09:00:00,Sensor_C,27.0,64.0
2023-01-01 10:00:00,Sensor_C,27.5,64.5
2023-01-01 11:00:00,Sensor_C,28.0,65.0
2023-01-01 12:00:00,Sensor_A,28.5,65.5
2023-01-01 13:00:00,Sensor_A,29.0,66.0
2023-01-01 14:00:00,Sensor_A,29.5,66.5
2023-01-01 15:00:00,Sensor_A,30.0,67.0
2023-01-01 16:00:00,Sensor_B,30.5,67.5
2023-01-01 17:00:00,Sensor_B,31.0,68.0
2023-01-01 18:00:00,Sensor_B,31.5,68.5
2023-01-01 19:00:00,Sensor_B,32.0,69.0
2023-01-01 20:00:00,Sensor_C,32.5,69.5
2023-01-01 21:00:00,Sensor_C,33.0,70.0
2023-01-01 22:00:00,Sensor_C,33.5,70.5
2023-01-01 23:00:00,Sensor_C,34.0,71.0
2023-01-02 00:00:00,Sensor_A,34.5,71.5
2023-01-02 01:00:00,Sensor_A,35.0,72.0
2023-01-02 02:00:00,Sensor_A,35.5,72.5
2023-01-02 03:00:00,Sensor_A,36.0,73.0
2023-01-02 04:00:00,Sensor_B,36.5,73.5
2023-01-02 05:00:00,Sensor_B,37.0,74.0
2023-01-02 06:00:00,Sensor_B,37.5,74.5
2023-01-02 07:00:00,Sensor_B,38.0,75.0
```

Col 1: timestamp

C: > Users > thota > Desktop > preprocess_iot.py > ...

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

# Load data
df = pd.read_csv('iot_data.csv')

# Convert timestamp to datetime
df['timestamp'] = pd.to_datetime(df['timestamp'])

# Handle missing values using forward fill
df['temperature'] = df['temperature'].ffill()
df['humidity'] = df['humidity'].ffill()

# Remove sensor drift (apply rolling mean with window=5)
df['temperature_smooth'] = df['temperature'].rolling(window=5, min_periods=1).mean()
df['humidity_smooth'] = df['humidity'].rolling(window=5, min_periods=1).mean()

# Normalize readings using standard scaling (manual)
df['temperature_scaled'] = (df['temperature_smooth'] - df['temperature_smooth'].mean()) / df['temperature_smooth']
df['humidity_scaled'] = (df['humidity_smooth'] - df['humidity_smooth'].mean()) / df['humidity_smooth'].std()

# Encode categorical sensor IDs (manual mapping)
sensor_mapping = {'Sensor_A': 0, 'Sensor_B': 1, 'Sensor_C': 2}
df['sensor_id_encoded'] = df['sensor_id'].map(sensor_mapping)

# Save preprocessed dataset
df.to_csv('preprocessed_iot_data.csv', index=False)

print("IoT sensor data preprocessing completed. Preprocessed data saved to preprocessed_iot_data.csv")
print("\nPreprocessed Dataset (first 10 rows):")
print(df.head(10).to_string(index=False))
```

```
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
PS C:\Users\thota\Desktop>
```

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.

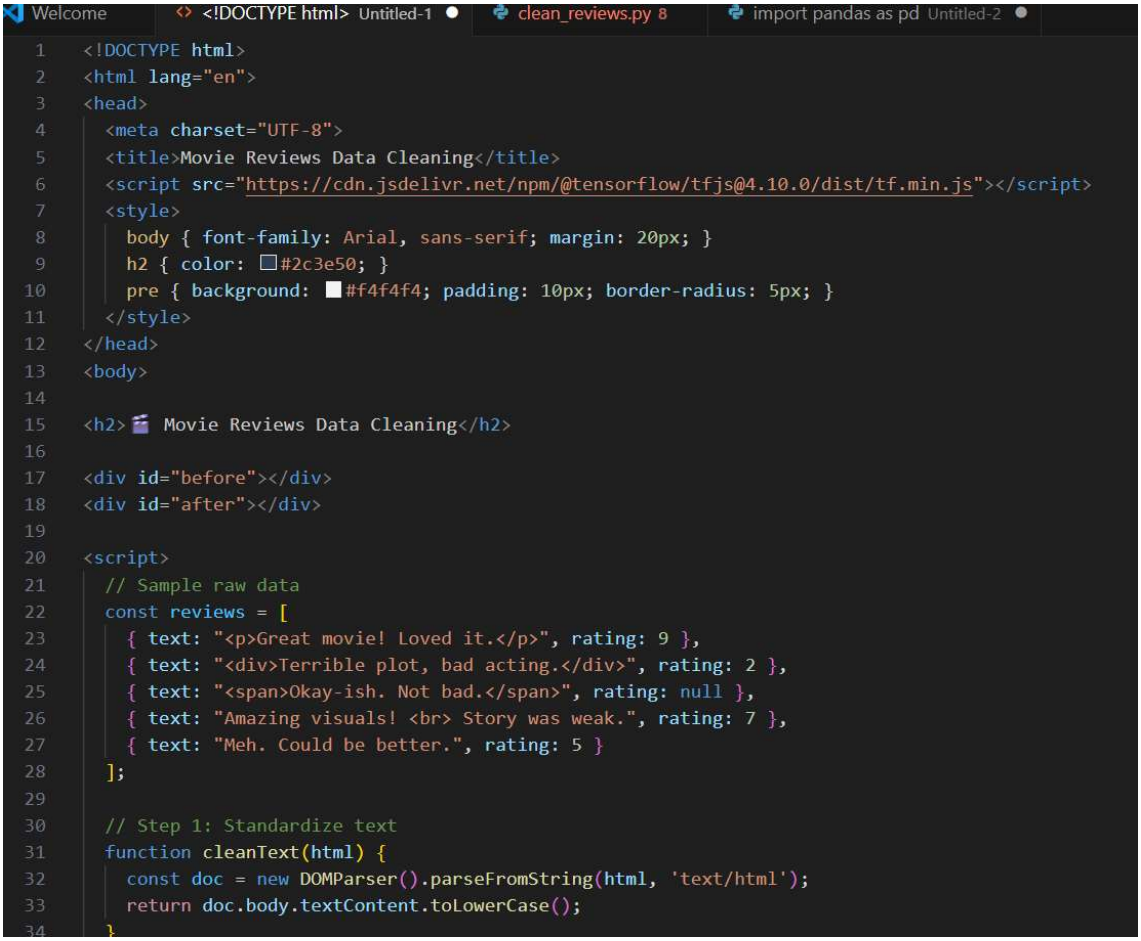
Task 4 – Real-Time Application: Movie Reviews Data Cleaning

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:



```python
import pandas as pd
import numpy as np
import re
from sklearn.feature_extraction.text import TfidfVectorizer

# Sample raw data
data = {
    'review': [
        "<p>Great movie! Loved it.</p>",
        "<div>Terrible plot, bad acting.</div>",
        "<span>Okay-ish. Not bad.</span>",
        "Amazing visuals! <br> Story was weak.",
        "Meh. Could be better."
    ],
    'rating': [9, 2, None, 7, 5]
}

# Create DataFrame
df = pd.DataFrame(data)

# Step 1: Standardize text
def clean_html(text):
    return re.sub(r'<.*?>', '', text).lower()

df['cleaned_review'] = df['review'].apply(clean_html)

# Step 2: Tokenize and encode using TF-IDF
vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform(df['cleaned_review'])
tfidf_df = pd.DataFrame(tfidf_matrix.toarray(), columns=vectorizer.get_feature_names_out())

# Step 3: Handle missing ratings
median_rating = df['rating'].median()
df['rating_filled'] = df['rating'].fillna(median_rating)
```

```python

df['cleaned_review'] = df['review'].apply(clean_html)

# Step 2: Tokenize and encode using TF-IDF
vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform(df['cleaned_review'])
tfidf_df = pd.DataFrame(tfidf_matrix.toarray(), columns=vectorizer.get_feature_names_out())

# Step 3: Handle missing ratings
median_rating = df['rating'].median()
df['rating_filled'] = df['rating'].fillna(median_rating)

# Step 4: Normalize ratings (0–10 → 0–1)
df['normalized_rating'] = df['rating_filled'] / 10

# Step 5: Combine and export to CSV
final_df = pd.concat([df[['review', 'cleaned_review', 'normalized_rating']], tfidf_df], axis=1)
final_df.to_csv('cleaned_movie_reviews.csv', index=False)

print("✅ Cleaned dataset saved as 'cleaned_movie_reviews.csv'")
```

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.*