AIPP LAB 17

Name: MOHAMMED NIZAMUDDIN

Hall Ticket No.: 2503B05144

Date: 04/12/2025 (Week 6)

Completed this Assignment using Google Colab – Gemini Al Integration.

Task 1:

Social Media Data Cleaning

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.

CODE:

```
# ================================================================
# NAME: MOHAMMED NIZAMUDDIN
# HALL TICKET NO. 2503B05144
# DATE: 04/12/2025

# LAB 17 – AI FOR DATA PROCESSING: CLEANING & PREPROCESSING
# ================================================================

# ------------------------------------------------
# INSTALLATION
# ------------------------------------------------
!pip install nltk pandas numpy scikit-learn beautifulsoup4 --quiet

import pandas as pd
import numpy as np
import nltk
import re
from bs4 import BeautifulSoup
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import StandardScaler, LabelEncoder

nltk.download("stopwords")
from nltk.corpus import stopwords
stop_words = set(stopwords.words("english"))
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
# ======================================================================
# ---------------------- TASK 1 ----------------------------------
# SOCIAL MEDIA DATA CLEANING
# ======================================================================

print("\n==================== TASK 1: SOCIAL MEDIA CLEANING ====================")

# Upload dataset in Google Colab
from google.colab import files
uploaded = files.upload()

social_df = pd.read_csv("social_media.csv")

# ----- BEFORE SUMMARY -----
print("\n--- BEFORE CLEANING ---")
print(social_df.head())

# -------- REMOVE PUNCTUATION, STOPWORDS, SPECIAL SYMBOLS ----------
def clean_text(text):
    if pd.isna(text):
        return ""
    text = text.lower()
    text = re.sub(r"[^a-zA-Z\s]", "", text)        # remove punctuation/symbols
    words = [w for w in text.split() if w not in stop_words]
    return " ".join(words)

social_df["clean_post"] = social_df["post_text"].apply(clean_text)
```

```
# -------- HANDLE MISSING VALUES IN likes & shares ----------
social_df["likes"] = social_df["likes"].fillna(social_df["likes"].median())
social_df["shares"] = social_df["shares"].fillna(0)

# -------- TIMESTAMP PROCESSING ----------
social_df["timestamp"] = pd.to_datetime(social_df["timestamp"])
social_df["hour"] = social_df["timestamp"].dt.hour
social_df["weekday"] = social_df["timestamp"].dt.day_name()

# -------- REMOVE DUPLICATE / SPAM ----------
social_df.drop_duplicates(subset="clean_post", inplace=True)

# ----- AFTER SUMMARY -----
print("\n--- AFTER CLEANING ---")
print(social_df.head())

# -------- TEST CASES ----------
assert social_df["clean_post"].isna().sum() == 0
assert social_df["likes"].isna().sum() == 0
assert "hour" in social_df.columns

print("\nTask 1 Passed All Tests ✓")
```

OUTPUT:

```
...  ========== TASK 1: SOCIAL MEDIA CLEANING ==========
     Choose Files No file chosen  Upload widget is only available when the cell has
Saving social_media.csv to social_media (2).csv

--- BEFORE CLEANING ---
   post_id     user                          post_text  likes  shares  \
0        1   user_1  This is a sample POST!!! #fun    20.0     1.0
1        2   user_2          <html>Great Day!</html>    20.0     3.0
2        3   user_3  This is a sample POST!!! #fun    20.0     1.0
3        4   user_4          <html>Great Day!</html>  100.0     NaN
4        5   user_5  This is a sample POST!!! #fun    20.0     5.0

             timestamp
0  2025-01-01 00:00:00
1  2025-01-01 06:00:00
2  2025-01-01 12:00:00
3  2025-01-01 18:00:00
4  2025-01-02 00:00:00

--- AFTER CLEANING ---
   post_id     user                          post_text  likes  shares  \
0        1   user_1  This is a sample POST!!! #fun    20.0     1.0
1        2   user_2          <html>Great Day!</html>    20.0     3.0

             timestamp          clean_post  hour    weekday
0 2025-01-01 00:00:00    sample post fun     0  Wednesday
1 2025-01-01 06:00:00  htmlgreat dayhtml     6  Wednesday

Task 1 Passed All Tests ✓
```

Task 2:

Financial Data Preprocessing

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:

```python
# ======================================================================
# ---------------------- TASK 2 ----------------------------------
# FINANCIAL DATA PREPROCESSING
# ======================================================================

print("\n==================== TASK 2: FINANCIAL DATA ====================")

financial_df = pd.read_csv("financial_data.csv")

# BEFORE SUMMARY
print("\n--- BEFORE ---")
print(financial_df.head())

# Missing values
financial_df["closing_price"] = financial_df["closing_price"].fillna(method="ffill")
financial_df["volume"] = financial_df["volume"].fillna(financial_df["volume"].median())

# Lag features
financial_df["return_1d"] = financial_df["closing_price"].pct_change()
financial_df["return_7d"] = financial_df["closing_price"].pct_change(periods=7)

# Log normalize volume
financial_df["volume_log"] = np.log1p(financial_df["volume"])

# Outlier detection (IQR)
Q1 = financial_df["closing_price"].quantile(0.25)
Q3 = financial_df["closing_price"].quantile(0.75)
IQR = Q3 - Q1
```

```python
lower = Q1 - 1.5 * IQR
upper = Q3 + 1.5 * IQR

financial_df = financial_df[(financial_df["closing_price"] >= lower) &
                            (financial_df["closing_price"] <= upper)]

# AFTER SUMMARY
print("\n--- AFTER ---")
print(financial_df.head())

# TESTS
assert financial_df["volume"].isna().sum() == 0
assert "return_1d" in financial_df.columns
assert financial_df["volume_log"].min() >= 0

print("\nTask 2 Passed All Tests ✓")
```

OUTPUT:

```
==================== TASK 2: FINANCIAL DATA ====================

--- BEFORE ---
         date  closing_price  volume
0  2025-01-01            NaN  5000.0
1  2025-01-02         131.04  2000.0
2  2025-01-03         138.26  2000.0
3  2025-01-04         164.68     NaN
4  2025-01-05         165.06  5000.0

--- AFTER ---
         date  closing_price  volume  return_1d  return_7d  volume_log
1  2025-01-02         131.04  2000.0        NaN        NaN    7.601402
2  2025-01-03         138.26  2000.0   0.055098        NaN    7.601402
3  2025-01-04         164.68  2000.0   0.191089        NaN    7.601402
4  2025-01-05         165.06  5000.0   0.002308        NaN    8.517393
5  2025-01-06         137.99  1500.0  -0.164001        NaN    7.313887

Task 2 Passed All Tests ✓
/tmp/ipython-input-1164313261.py:15: FutureWarning: Series.fillna with 'method'
  financial_df["closing_price"] = financial_df["closing_price"].fillna(method='
```

Task 3:

IoT Sensor Data Preparation

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:

```
# ======================================================================
# ---------------------- TASK 3 ----------------------------------------
# IOT SENSOR DATA
# ======================================================================

print("\n==================== TASK 3: IOT SENSOR ====================")

iot_df = pd.read_csv("iot_sensor.csv")

print("\n--- BEFORE ---")
print(iot_df.head())

# Handle missing → Forward fill
iot_df["temperature"] = iot_df["temperature"].fillna(method="ffill")
iot_df["humidity"] = iot_df["humidity"].fillna(method="ffill")

# Remove drift → Rolling mean
iot_df["temp_smooth"] = iot_df["temperature"].rolling(5, min_periods=1).mean()
iot_df["humidity_smooth"] = iot_df["humidity"].rolling(5, min_periods=1).mean()

# Normalize using standard scaling
scaler = StandardScaler()
iot_df["temp_scaled"] = scaler.fit_transform(iot_df[["temp_smooth"]])
iot_df["humidity_scaled"] = scaler.fit_transform(iot_df[["humidity_smooth"]])

# Encode sensor ID
encoder = LabelEncoder()
iot_df["sensor_encoded"] = encoder.fit_transform(iot_df["sensor_id"])

    print("\n--- AFTER ---")
    print(iot_df.head())

    # TESTS
    assert iot_df["temperature"].isna().sum() == 0
    assert "temp_scaled" in iot_df.columns
    assert iot_df["sensor_encoded"].nunique() > 0

    print("\nTask 3 Passed All Tests ✓")
```

OUTPUT:

```
==================== TASK 3: IOT SENSOR ====================

--- BEFORE ---
             timestamp sensor_id  temperature  humidity
0  2025-02-01 00:00:00         S2         24.0      40.0
1  2025-02-01 01:00:00         S3         30.0       NaN
2  2025-02-01 02:00:00         S1         24.0      50.0
3  2025-02-01 03:00:00         S2         24.0       NaN
4  2025-02-01 04:00:00         S3         23.0      42.0

--- AFTER ---
             timestamp sensor_id  temperature  humidity  temp_smooth  \
0  2025-02-01 00:00:00         S2         24.0      40.0         24.0
1  2025-02-01 01:00:00         S3         30.0      40.0         27.0
2  2025-02-01 02:00:00         S1         24.0      50.0         26.0
3  2025-02-01 03:00:00         S2         24.0      50.0         25.5
4  2025-02-01 04:00:00         S3         23.0      42.0         25.0

   humidity_smooth  temp_scaled  humidity_scaled  sensor_encoded
0        40.000000    -0.031984        -1.922355               1
1        40.000000     2.493079        -1.922355               2
2        43.333333     1.651391        -0.377054               0
3        45.000000     1.230547         0.395597               1
4        44.400000     0.809703         0.117443               2

Task 3 Passed All Tests ✓
/tmp/ipython-input-91793892.py:14: FutureWarning: Series.fillna with 'me
  iot_df["temperature"] = iot_df["temperature"].fillna(method="ffill")
/tmp/ipython-input-91793892.py:15: FutureWarning: Series.fillna with 'me
  iot_df["humidity"] = iot_df["humidity"].fillna(method="ffill")
```

Task 4:

Real-Time Application: Movie Reviews Data Cleaning

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:

```
# ================================================================
# ---------------------- TASK 4 ----------------------------------
# MOVIE REVIEWS
# ================================================================

print("\n==================== TASK 4: MOVIE REVIEWS CLEANING ====================")

movie_df = pd.read_csv("movie_reviews-1.csv")

print("\n--- BEFORE ---")
print(movie_df.head())

# Clean HTML + lowercase
def clean_review(text):
    if pd.isna(text):
        return ""
    text = BeautifulSoup(text, "html.parser").get_text()
    return text.lower()

movie_df["clean_review"] = movie_df["review_text"].apply(clean_review)

# Tokenize + TF-IDF
tfidf = TfidfVectorizer(stop_words="english", max_features=500)
tfidf_matrix = tfidf.fit_transform(movie_df["clean_review"])

# Rating → fill missing
movie_df["rating"] = movie_df["rating"].fillna(movie_df["rating"].median())

# Normalize 0–10 → 0–1
```

```
# Normalize 0–10 → 0–1
movie_df["rating_norm"] = movie_df["rating"] / 10

print("\n--- AFTER ---")
print(movie_df.head())

# TESTS
assert movie_df["clean_review"].isna().sum() == 0
assert movie_df["rating_norm"].max() <= 1
assert tfidf_matrix.shape[0] == len(movie_df)

print("\nTask 4 Passed All Tests ✓")

print("\n==================== ALL TASKS COMPLETED SUCCESSFULLY ====================\n")
```

OUTPUT:

```
==================== TASK 4: MOVIE REVIEWS CLEANING ====================

--- BEFORE ---
   review_id                  review_text  rating
0          1      <p>Amazing movie!</p>     8.0
1          2  Terrible acting & plot!!!     2.0
2          3      <p>Amazing movie!</p>     NaN
3          4  Terrible acting & plot!!!     8.0
4          5      <p>Amazing movie!</p>     5.0

--- AFTER ---
   review_id                  review_text  rating                clean_review  \
0          1      <p>Amazing movie!</p>     8.0              amazing movie!
1          2  Terrible acting & plot!!!     2.0  terrible acting & plot!!!
2          3      <p>Amazing movie!</p>     8.0              amazing movie!
3          4  Terrible acting & plot!!!     8.0  terrible acting & plot!!!
4          5      <p>Amazing movie!</p>     5.0              amazing movie!

   rating_norm
0          0.8
1          0.2
2          0.8
3          0.8
4          0.5

Task 4 Passed All Tests ✓

==================== ALL TASKS COMPLETED SUCCESSFULLY ====================
```