AI ASSISTED CODING

Lab-17 AI for Data Processing

Name: G Hasini

Roll no: 2503A51L41

Batch no: 24BTCAICSB20

Task_1: Use AI to generate a Python script for cleaning an employee dataset

Prompt: Generate a Python script to clean an employee dataset by handling missing values, formatting joining_date, standardizing department names, and encoding categorical variables.

Code:

```python
import pandas as pd
import os

# Print current working directory
print("Current working directory:", os.getcwd())

# Load dataset
df = pd.read_csv(r"C:\Users\Saniya Tahseen\OneDrive\Desktop\AI assistant coding\employee_data_cleaning\employees.csv")

# Handle missing values (safe assignment)
df = df.fillna({
    'SALARY': df['SALARY'].median(),
    'DEPARTMENT_ID': 'Unknown',
    'HIRE_DATE': '1900-01-01'
})

# Convert HIRE_DATE to datetime with format hint (optional but safer)
df['HIRE_DATE'] = pd.to_datetime(df['HIRE_DATE'], errors='coerce')

# Standardize DEPARTMENT_ID
df['DEPARTMENT_ID'] = df['DEPARTMENT_ID'].astype(str).str.lower().replace({
    'human resources': 'hr',
    'hr': 'hr',
    'finance': 'finance',
    'sales': 'sales',
    'marketing': 'marketing'
})

# Encode categorical variables
df_encoded = pd.get_dummies(df, columns=['DEPARTMENT_ID', 'JOB_ID'])

# Save cleaned data
df_encoded.to_csv("cleaned_employees.csv", index=False)

# Display cleaned DataFrame
print(df_encoded.head())
# Optionally print the raw CSV file contents
```

Output:

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	SALARY	...	JOB_ID_MK_REP	JOB_ID_PR_REP	JOB_ID_PU_CLERK	JOB_ID_PU_MAN	JOB_ID_SH_CLERK	JOB_ID_ST_CLERK	JOB_ID_ST_MAN
198	Donald	OConnell	DOCONNEL	650.507.9833	2007-06-21	2600	...	False	False	False	False	True	False	False
199	Douglas	Grant	DGRANT	650.507.9844	2008-01-13	2600	...	False	False	False	False	True	False	False
200	Jennifer	Whalen	JWHALEN	515.123.4444	2003-09-17	4400	...	False	False	False	False	False	False	False
201	Michael	Hartstein	MHARTSTE	515.123.5555	2004-02-17	13000	...	False	False	False	False	False	False	False
202	Pat	Fay	PFAY	603.123.6666	2005-08-17	6000	...	True	False	False	False	False	False	False

rows x 36 columns]

Observations:

The AI help me clean the database and it handled all the missing value and made the database ready to use

Task_2:

Use AI to generate a script for preprocessing a sales transaction dataset

Prompt:

Preprocess a sales dataset by parsing dates, extracting Month-Year, removing invalid amounts, and normalizing values.

Code:

mean.py > inventory.py > ...

```python
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

# Load dataset
df = pd.read_csv(r"C:\Users\Saniya Tahseen\OneDrive\Desktop\AI assistant coding\sales_data_cleaning\transactions.csv")

# Convert transaction_date to datetime
df['transaction_date'] = pd.to_datetime(df['transaction_date'], format='%Y-%m-%d', errors='coerce')

# Create Month-Year column
df['Month_Year'] = df['transaction_date'].dt.to_period('M').astype(str)

# Remove rows with zero or negative transaction_amount
df = df[df['transaction_amount'] > 0]

# Normalize transaction_amount using Min-Max scaling
scaler = MinMaxScaler()
df['normalized_amount'] = scaler.fit_transform(df[['transaction_amount']])

# Save cleaned data
df.to_csv("cleaned_transactions.csv", index=False)

# Display first few rows
print(df.head())

# Optional: Read and print the saved cleaned file
with open(r"C:\Users\Saniya Tahseen\OneDrive\Desktop\AI assistant coding\sales_data_cleaning\transactions.csv", "r") as f:
    print(f.read())

```

Output:

```
   transaction_id customer_id transaction_date  transaction_amount product_category Month_Year  normalized_amount
0               1        C001       2025-01-15                 250      Electronics    2025-01           0.095238
3               4        C004       2025-02-18                1200      Electronics    2025-02           1.000000
4               5        C005       2025-03-10                 300          Fashion    2025-03           0.142857
5               6        C006       2025-03-15                 450          Grocery    2025-03           0.285714
7               8        C008       2025-04-12                 800          Fashion    2025-04           0.619048
transaction_id,customer_id,transaction_date,transaction_amount,product_category
1,C001,2025-01-15,250,Electronics
2,C002,2025-01-20,0,Fashion
3,C003,2025-02-05,-50,Grocery
4,C004,2025-02-18,1200,Electronics
5,C005,2025-03-10,300,Fashion
6,C006,2025-03-15,450,Grocery
7,C007,2025-04-01,0,Electronics
8,C008,2025-04-12,800,Fashion
9,C009,2025-05-05,150,Grocery
10,C010,2025-05-20,600,Electronics
```

Observation:

- The AI helped me in cleaning the data, handle the missing value and irrelevant information in my database
- It made my database readable and ready to process

Task_3:

Use AI to generate a script for cleaning healthcare patient records**.**

Prompt:

Clean healthcare patient records by imputing numeric means, standardizing height units, fixing gender labels, and dropping IDs.

Code:

```
healthcare_data_cleaning > clean_patient_records.py > ...
import pandas as pd
from io import StringIO

# Step 1: Rebuild the CSV data inside Python
csv_data = """patient_id,gender,height_cm,blood_pressure,heart_rate
101,M,175,120,80
102,Female,160,,72
103,male,180,130,
104,f,165,110,75
105,Male,170,,78
106,F,158,125,70
107,m,172,118,
108,female,168,122,74
109,M,177,135,82
110,F,162,,76
"""

# Step 2: Load CSV data directly (no file issues)
df = pd.read_csv(StringIO(csv_data))

# Step 3: Fill missing values in numeric columns with column mean
numeric_cols = ['blood_pressure', 'heart_rate']
for col in numeric_cols:
    df[col] = df[col].fillna(df[col].mean())

# Step 4: Convert height from cm to meters
if 'height_cm' in df.columns:
    df['height_m'] = df['height_cm'] / 100
    df.drop(columns=['height_cm'], inplace=True)

# Step 5: Standardize gender labels
if 'gender' in df.columns:
    df['gender'] = df['gender'].astype(str).str.lower().replace({
        'm': 'Male',
        'male': 'Male',
        'f': 'Female',
        'female': 'Female'
    })

# Step 6: Drop irrelevant columns
df.drop(columns=['patient_id'], inplace=True)

# Step 7: Save cleaned data
df.to_csv("cleaned_patient_records.csv", index=False)

# Step 8: Display cleaned DataFrame
print("✅ Cleaned data saved as 'cleaned_patient_records.csv'\n")
print(df)
```

Output:

```
✅ Cleaned data saved as 'cleaned_patient_records.csv'

   gender  blood_pressure  heart_rate  height_m
0    Male      120.000000      80.000      1.75
1  Female      122.857143      72.000      1.60
2    Male      130.000000      75.875      1.80
3  Female      110.000000      75.000      1.65
4    Male      122.857143      78.000      1.70
5  Female      125.000000      70.000      1.58
6    Male      118.000000      75.875      1.72
7  Female      122.000000      74.000      1.68
8    Male      135.000000      82.000      1.77
9  Female      122.857143      76.000      1.62
```

Observation:

- It standardized height units from centimeters to meters with a simple conversion, ensuring consistency for downstream analysis like BMI calculation.
- AI corrected inconsistent gender labels (e.g., "M", "Male", "male") into a unified format, improving data quality and enabling reliable categorical encoding.

Task_4:

Use AI to write a script to preprocess a social media text dataset.

Prompt:

Clean and prepare social media text for sentiment analysis by removing noise, normalizing, and lemmatizing

Code:

```python
import pandas as pd
import re

# Step 1: Example dataset (you can replace this with your CSV)
data = {
    'post_id': [1, 2, 3, 4],
    'text': [
        "I loooove this product!!! 😍🔥 Check it out: https://example.com",
        "Ugh, this app keeps crashing :( #annoyed",
        "Best update ever. Totally worth it! 👏👏",
        "Not happy with the service... too slow!!! 😡"
    ]
}

df = pd.DataFrame(data)

# Step 2: Basic stopword list (simple version)
stop_words = {'a', 'the', 'is', 'it', 'this', 'that', 'i', 'with', 'for', 'to', 'and', 'of', 'in', 'on', 'too'}

# Step 3: Text cleaning function (no external libraries)
def clean_text(text):
    # Remove URLs
    text = re.sub(r'http\S+|www\S+', '', text)
    # Remove emojis and non-alphanumeric characters
    text = re.sub(r'[^a-zA-Z0-9\s]', '', text)
    # Convert to lowercase
    text = text.lower()
    # Tokenize by splitting
    words = text.split()
    # Remove stopwords
    words = [w for w in words if w not in stop_words]
    # (Optional) Simple lemmatization-like cleanup for plural forms
    cleaned = []
    for w in words:
        if w.endswith('s') and len(w) > 3:  # crude lemmatization
            w = w[:-1]
        cleaned.append(w)
    # Join back into sentence
    return ' '.join(cleaned)

# Step 4: Apply function
df['clean_text'] = df['text'].apply(clean_text)

# Step 5: Save cleaned data
df.to_csv("cleaned_social_media_posts_simple.csv", index=False)

# Step 6: Display cleaned dataset
print("✅ Cleaned dataset saved as 'cleaned_social_media_posts_simple.csv'\n")
print(df[['text', 'clean_text']])
```

Output:

```
✅ Cleaned dataset saved as 'cleaned_social_media_posts_simple.csv'

                                              text                       clean_text
0  I loooove this product!!! 😍🔥 Check it out: htt...      loooove product check out
1          Ugh, this app keeps crashing :( #annoyed   ugh app keep crashing annoyed
2            Best update ever. Totally worth it! 👏👏  best update ever totally worth
3         Not happy with the service... too slow!!! 😡          not happy service slow
```

Observation:

- AI helped strip out clutter like emojis, URLs, and special characters so your text is clean and analysis-ready.
- AI converted everything to lowercase and removed common stop words to focus on meaningful words.
- AI applied lemmatization to standardize word forms, making your sentiment model smarter and more accurate.

Task_5:

Use AI to create a preprocessing script for a financial dataset

Prompt:

Preprocess financial data by handling missing values, engineering moving averages, normalizing, and encoding categories.

Code:

```
ssignment 17.4 > task5.py > ...
import pandas as pd
from io import StringIO
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer

# Simulated CSV content
csv_data = """date,company_name,sector,stock_price,volume
2025-10-01,AlphaTech,Technology,120.5,10000
2025-10-02,AlphaTech,Technology,121.0,9800
2025-10-03,BetaCorp,Finance,,10500
2025-10-04,BetaCorp,Finance,118.0,
2025-10-05,GammaInc,Healthcare,119.5,11000
2025-10-06,GammaInc,Healthcare,120.0,10800
2025-10-07,AlphaTech,Technology,122.0,10200
2025-10-08,BetaCorp,Finance,123.5,10700
2025-10-09,GammaInc,Healthcare,124.0,10900
2025-10-10,AlphaTech,Technology,125.0,11100
2025-10-11,BetaCorp,Finance,126.5,11200
2025-10-12,GammaInc,Healthcare,127.0,11300
2025-10-13,AlphaTech,Technology,128.0,11400
2025-10-14,BetaCorp,Finance,129.5,11500
2025-10-15,GammaInc,Healthcare,130.0,11600
"""

# Load the CSV from string
df = pd.read_csv(StringIO(csv_data), parse_dates=['date'])

# Handle missing values
imputer = SimpleImputer(strategy='mean')
df['stock_price'] = imputer.fit_transform(df[['stock_price']])
df['volume'] = imputer.fit_transform(df[['volume']])

# Create moving averages
df['MA_7'] = df['stock_price'].rolling(window=7).mean()
df['MA_30'] = df['stock_price'].rolling(window=30).mean()

# Encode categorical variables
df = pd.get_dummies(df, columns=['company_name', 'sector'], drop_first=True)

# Normalize continuous variables
scaler = StandardScaler()
df[['stock_price', 'volume', 'MA_7', 'MA_30']] = scaler.fit_transform(df[['stock_price', 'volume', 'MA_7', 'MA_30']])

# Display the final feature-engineered DataFrame
print(df.head())
```

Output:

	date	stock_price	volume	MA_7	MA_30	company_name_BetaCorp	company_name_GammaInc	sector_Healthcare	sector_Technology
0	2025-10-01	-0.938392	-1.640627	NaN	NaN	False	False	False	True
1	2025-10-02	-0.800103	-2.023440	NaN	NaN	False	False	False	True
2	2025-10-03	0.000000	-0.683594	NaN	NaN	True	False	False	False
3	2025-10-04	-1.629839	0.000000	NaN	NaN	True	False	False	False
4	2025-10-05	-1.214971	0.273438	NaN	NaN	False	True	True	False

Observation:

- AI filled in missing stock price and volume data using column averages, so your model won't stumble on gaps.
- AI added 7-day and 30-day moving averages to give your model trend awareness for smarter predictions.
- AI normalized all numeric features and encoded company and sector labels, making your dataset clean and ML-ready.