Data Preprocessing in ML – JSON File Exmaple

Aim:

To prepare weather data for machine learning by handling types and scaling.

Steps:

1. Load the structured data.
2. Handle missing values (if any).
3. Encode categorical variables.
4. Normalize or scale numeric features.
5. Display the preprocessed DataFrame.

Expected Output:

A clean, model-ready dataset with numeric features.

Skills Learned:

- Working Handling missing data
- Encoding and scaling
- Data preparation for ML

Raw JSON file

```
{
  "city": {
    "name": "Chennai",
    "country": "IN"
  },
  "list": [
    {
      "dt_txt": "2025-07-22 09:00:00",
      "main": {
        "temp": 301.15,
        "humidity": 80
      },
      "weather": [
        {
          "main": "Rain",
          "description": "light rain"
        }
      ]
    },
    {
      "dt_txt": "2025-07-22 12:00:00",
      "main": {
        "temp": 303.25,
        "humidity": 72
      },
      "weather": [
        {
          "main": "Clouds",
          "description": "scattered clouds"
        }
      ]
    }
  ]
}
```

json structure

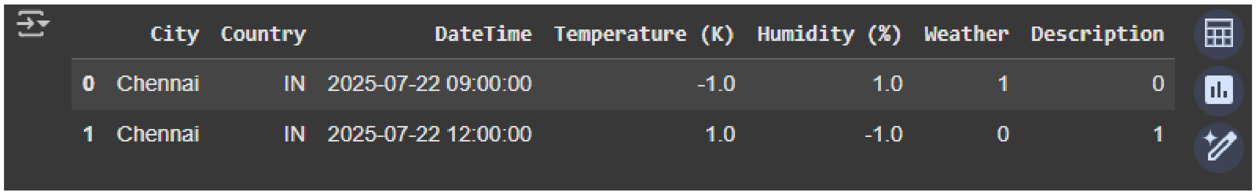
weather. json structure

```
├── city (object)
│   ├── name: "Chennai"
│   └── country: "IN"
├── list (array of objects)
│   ├── [0]
│   │   ├── dt_txt: "2025-07-22 09:00:00"
│   │   ├── main (object)
│   │   │   ├── temp: 301.15
│   │   │   └── humidity: 80
│   │   └── weather (array)
│   │       └── [0]
│   │           ├── main: "Rain"
│   │           └── description: "light rain"
│   └── [1]
│       ├── dt_txt: "2025-07-22 12:00:00"
│       ├── main (object)
│       │   ├── temp: 303.25
│       │   └── humidity: 72
│       └── weather (array)
│           └── [0]
│               ├── main: "Clouds"
│               └── description: "scattered clouds"
│
```

Python Source Code

```
import json
import pandas as pd
from sklearn.preprocessing import LabelEncoder, StandardScaler

# Step 1: Load the JSON properly
with open("weather.json", "r") as f:
    weather_data = json.load(f)

# Step 2: Extract and structure the data
data = []
for entry in weather_data['list']:
    data.append({
        'City': weather_data['city']['name'],
        'Country': weather_data['city']['country'],
        'DateTime': entry['dt_txt'],
        'Temperature (K)': entry['main']['temp'],
        'Humidity (%)': entry['main']['humidity'],
        'Weather': entry['weather'][0]['main'],
        'Description': entry['weather'][0]['description']
    })

df = pd.DataFrame(data)

# Step 3: Label Encoding for categorical columns
le = LabelEncoder()
df['Weather'] = le.fit_transform(df['Weather'])
df['Description'] = le.fit_transform(df['Description'])

# Step 4: Standard Scaling for numerical columns
scaler = StandardScaler()
df[['Temperature (K)', 'Humidity (%)']] =
scaler.fit_transform(df[['Temperature (K)', 'Humidity (%)']])

df
```

OUTPUT:



```
   City     Country  DateTime             Temperature (K)  Humidity (%)  Weather  Description
0  Chennai  IN       2025-07-22 09:00:00  -1.0              1.0          1        0
1  Chennai  IN       2025-07-22 12:00:00   1.0             -1.0          0        1
```

Data Preprocessing in ML – CSV File Exmaple

Aim:
To clean and preprocess student data for machine learning.

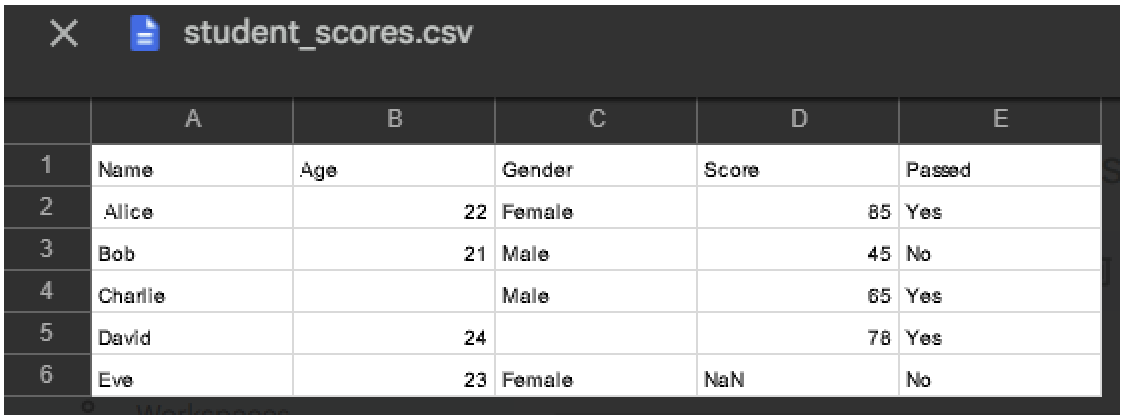
Steps:
1. Load the structured data.
2. Handle missing values (if any).
3. Encode categorical variables.
4. Normalize or scale numeric features.
5. Display the preprocessed DataFrame.

Expected Output:
A clean, model-ready dataset with numeric features.

Skills Learned:

- Handling missing data
- Encoding and scaling
- Data preparation for ML

Sample CSV File

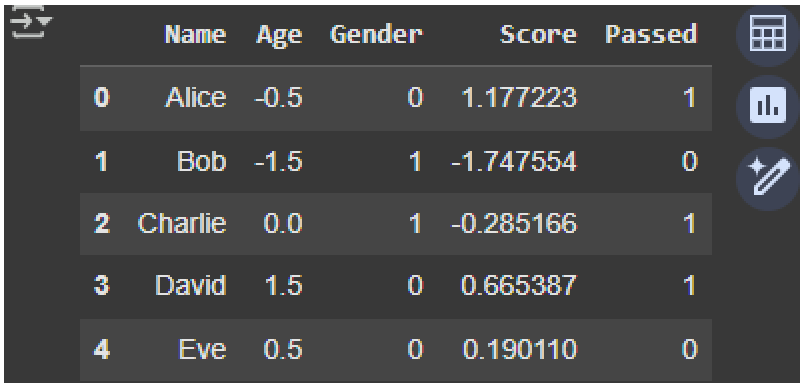

student_scores.csv

	A	B	C	D	E
1	Name	Age	Gender	Score	Passed
2	Alice	22	Female	85	Yes
3	Bob	21	Male	45	No
4	Charlie		Male	65	Yes
5	David	24		78	Yes
6	Eve	23	Female	NaN	No

Python Source Code

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder

df = pd.read_csv("student_scores.csv")

# Strip extra spaces from column names and string fields
df.columns = df.columns.str.strip()
df['Name'] = df['Name'].str.strip()
df['Gender'] = df['Gender'].str.strip()
df['Passed'] = df['Passed'].str.strip()

# Convert Age and Score to numeric, handle missing values
df['Age'] = pd.to_numeric(df['Age'], errors='coerce')
df['Age'] = df['Age'].fillna(df['Age'].mean())
df['Score'] = pd.to_numeric(df['Score'], errors='coerce')
df['Score'] = df['Score'].fillna(df['Score'].median())


# Handle missing gender using mode
df['Gender'] = df['Gender'].fillna(df['Gender'].mode()[0])

# Label Encoding
le = LabelEncoder()
df['Gender'] = le.fit_transform(df['Gender'])
df['Passed'] = le.fit_transform(df['Passed'])

# Feature Scaling
scaler = StandardScaler()
df[['Age', 'Score']] = scaler.fit_transform(df[['Age', 'Score']])

df
```

OUTPUT:



```
      Name   Age  Gender     Score  Passed
0    Alice  -0.5       0  1.177223       1
1      Bob  -1.5       1 -1.747554       0
2  Charlie   0.0       1 -0.285166       1
3    David   1.5       0  0.665387       1
4      Eve   0.5       0  0.190110       0
```

Data Preprocessing in ML – Log File Exmaple

Aim:
To encode and prepare system log data for analysis.

Steps:
1. Load the structured data.
2. Handle missing values (if any).
3. Encode categorical variables.
4. Normalize or scale numeric features.
5. Display the preprocessed DataFrame.

Expected Output:

A clean, model-ready dataset with numeric features.

Skills Learned:

- Handling missing data
- Encoding and scaling
- Data preparation for ML

Pattern matching:

1. `\[(.*?)\]`: Matches and captures the date-time string inside square brackets.

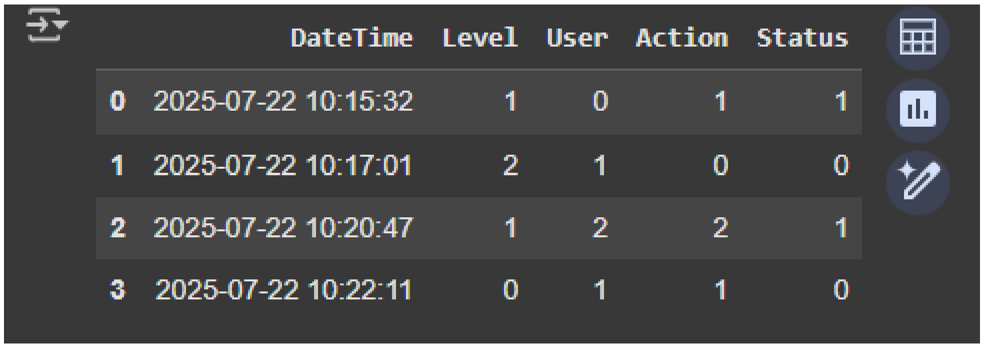
   a) . - any character
   b) * - zero or more times
   c) ? - non-greedy (stop at the first closing bracket)

2. `(\w+)`: Matches and captures the log level (e.g., INFO, ERROR).
3. `User=(\w+)`: Matches "User=" and captures the username that follows.
4. `Action=(\w+)`: Matches "Action=" and captures the user's action (e.g., Login, Upload).
5. `Status=(\w+)`: Matches "Status=" and captures the action result (e.g., Success, Failed).

Python Source Code

```
import pandas as pd
import re
from sklearn.preprocessing import LabelEncoder

with open("access.log") as f:
    lines = f.readlines()

records = []
for line in lines:
    match = re.search(r"\[(.*?)\] (\w+): User=(\w+) Action=(\w+)
Status=(\w+)", line)
    if match:
        dt, level, user, action, status = match.groups()
        records.append({
            'DateTime': dt,
            'Level': level,
            'User': user,
            'Action': action,
            'Status': status
        })

df = pd.DataFrame(records)

# Encode categorical columns
le = LabelEncoder()
df['Level'] = le.fit_transform(df['Level'])
df['User'] = le.fit_transform(df['User'])
df['Action'] = le.fit_transform(df['Action'])
df['Status'] = le.fit_transform(df['Status'])

df
```

OUTPUT:



```
   DateTime             Level  User  Action  Status
0  2025-07-22 10:15:32      1     0       1       1
1  2025-07-22 10:17:01      2     1       0       0
2  2025-07-22 10:20:47      1     2       2       1
3  2025-07-22 10:22:11      0     1       1       0
```

Data Preprocessing in ML – HTML File Exmaple

Aim:
To scale and prepare product details for machine learning models.

Steps:
1. Load the structured data.
2. Handle missing values (if any).
3. Encode categorical variables.
4. Normalize or scale numeric features.
5. Display the preprocessed DataFrame.

Expected Output:
A clean, model-ready dataset with numeric features.

Skills Learned:
- Handling missing data
- Encoding and scaling
- Data preparation for ML

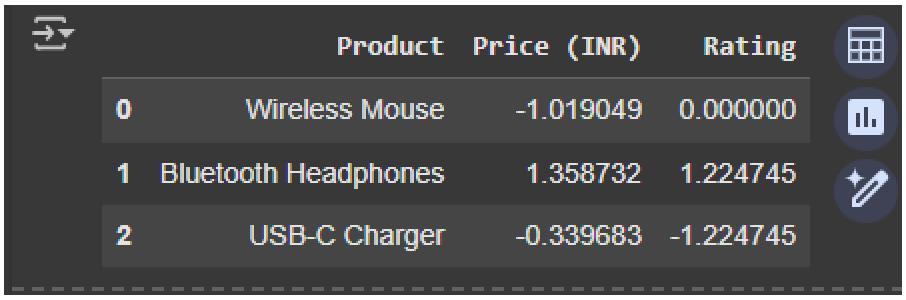
Products.html

```
<html>
  <head><title>Sample Product Page</title></head>
  <body>
    <div class="product">
      <h2 class="title">Wireless Mouse</h2>
      <span class="price">₹799</span>
      <span class="rating">4.3</span>
    </div>
    <div class="product">
      <h2 class="title">Bluetooth Headphones</h2>
      <span class="price">₹1499</span>
      <span class="rating">4.5</span>
    </div>
    <div class="product">
      <h2 class="title">USB-C Charger</h2>
      <span class="price">₹999</span>
      <span class="rating">4.1</span>
    </div>
  </body>
</html>
```

Python Source Code

```
from bs4 import BeautifulSoup
import pandas as pd
from sklearn.preprocessing import StandardScaler

with open("products.html", "r", encoding="utf-8") as f:
    soup = BeautifulSoup(f, "html.parser")

products = soup.find_all("div", class_="product")

records = []
for p in products:
    title = p.find("h2", class_="title").text.strip()
    price = p.find("span",
class_="price").text.strip().replace("₹", "")
    rating = p.find("span", class_="rating").text.strip()
    records.append({
        'Product': title,
        'Price (INR)': int(price),
        'Rating': float(rating)
    })

df = pd.DataFrame(records)

# Feature Scaling
scaler = StandardScaler()
df[['Price (INR)', 'Rating']] = scaler.fit_transform(df[['Price
(INR)', 'Rating']])

df
```

OUTPUT:



```
                  Product  Price (INR)    Rating
0          Wireless Mouse    -1.019049  0.000000
1  Bluetooth Headphones     1.358732  1.224745
2           USB-C Charger    -0.339683 -1.224745
```