Artificial Intelligent (Lab)

Task # 10

Submitted To:	**Sir Rasikh Ali**
Submitted By:	**Shumaila Maryam**
Roll no:	**SU92-BSDSM-F24-062**
Section:	**BSDS-3A**

Department of Software Engineering,
Superior University, Lahore

Introduction

This section focuses on the essential stage of **Data Pre-Processing**, where the raw dataset is cleaned, transformed, and prepared for machine learning. Real datasets usually contain missing values, inconsistent formats, unnecessary columns, and mixed data types. If these issues are not fixed, they can negatively affect model accuracy. The purpose of this process is to convert the original, unorganized data into a clean, structured, and machine-readable format. In this project, I applied multiple pre-processing steps such as handling null values, converting data types, extracting numeric values, and encoding categorical features to ensure the dataset becomes fully ready for analysis and model building.

Why I Made This Part

The main goal of performing data pre-processing is to improve the overall quality of the dataset before using it for further analysis. Pre-processing is necessary because:

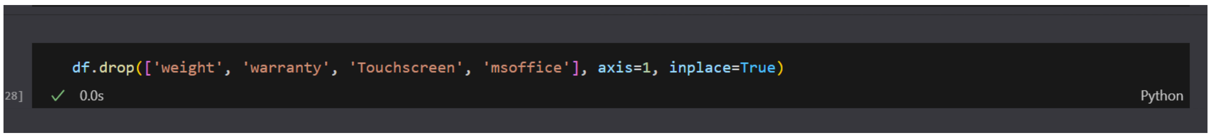
- It fixes missing values to avoid errors during model training.
- It standardizes text-based numeric fields (like "8 GB" → 8).
- It converts categorical labels into numeric values that algorithms can understand.
- It removes unnecessary or irrelevant columns to reduce noise in the dataset.
- It prepares the data for machine learning models by ensuring consistent formats and correct data types.

Overall, this part ensures that the dataset becomes clean, reliable, and suitable for predictive modeling.

How It Works :

1. Dropping Unnecessary Columns

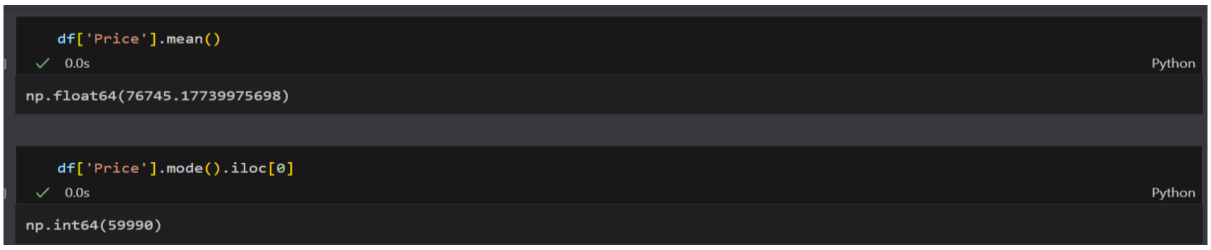
Some columns had too many missing values or no importance for analysis, so they were removed using:



```
df.drop(['weight', 'warranty', 'Touchscreen', 'msoffice'], axis=1, inplace=True)
```
```
28] ✓ 0.0s                                                    Python
```

2. Find mode , mean , median

In data analysis, three of the most important statistical measures are Mean, Median, and Mode. These are also called measures of central tendency, because they help us understand the "center" or typical value of a dataset.



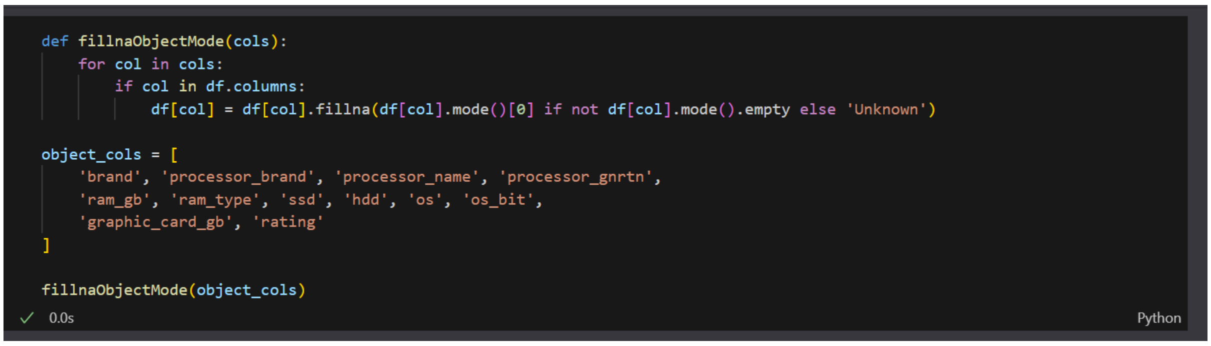
```
df['Price'].mean()
✓ 0.0s                                                        Python
np.float64(76745.17739975698)
```

```
df['Price'].mode().iloc[0]
✓ 0.0s                                                        Python
np.int64(59990)
```

2. Filling Missing Values

2.1 Filling Categorical Columns Using Mode

All object-based columns such as brand, processor_brand, processor_name, ram_type, OS, etc., were filled using the most frequent value:



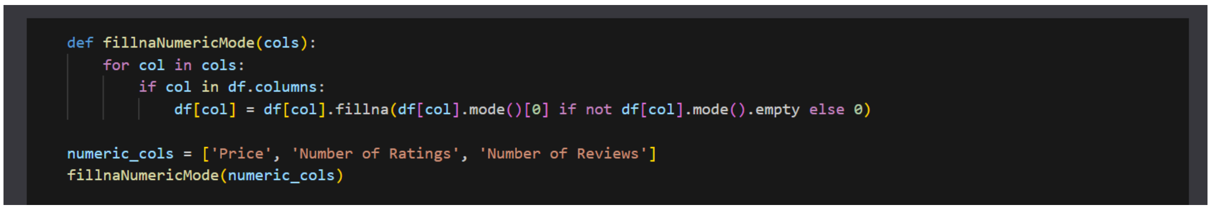
```
def fillnaObjectMode(cols):
    for col in cols:
        if col in df.columns:
            df[col] = df[col].fillna(df[col].mode()[0] if not df[col].mode().empty else 'Unknown')

object_cols = [
    'brand', 'processor_brand', 'processor_name', 'processor_gnrtn',
    'ram_gb', 'ram_type', 'ssd', 'hdd', 'os', 'os_bit',
    'graphic_card_gb', 'rating'
]

fillnaObjectMode(object_cols)
✓ 0.0s                                                        Python
```

2.2 Filling Numeric Columns Using Mode

Columns like Price, Number of Ratings, and Number of Reviews also contained missing values, so they were filled similarly:

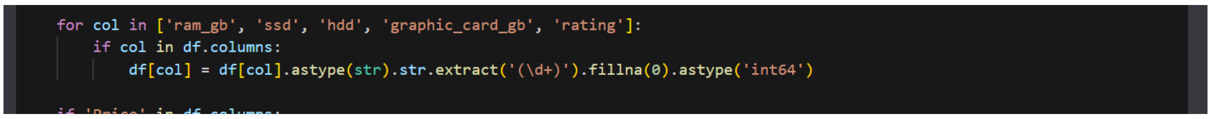


```
def fillnaNumericMode(cols):
    for col in cols:
        if col in df.columns:
            df[col] = df[col].fillna(df[col].mode()[0] if not df[col].mode().empty else 0)

numeric_cols = ['Price', 'Number of Ratings', 'Number of Reviews']
fillnaNumericMode(numeric_cols)
```

3. Extracting Numeric Values From Text Columns

Some columns contained values like "8 GB", "512 SSD", or "2 GB Graphic".
To clean these, I used regex to extract only the digits:

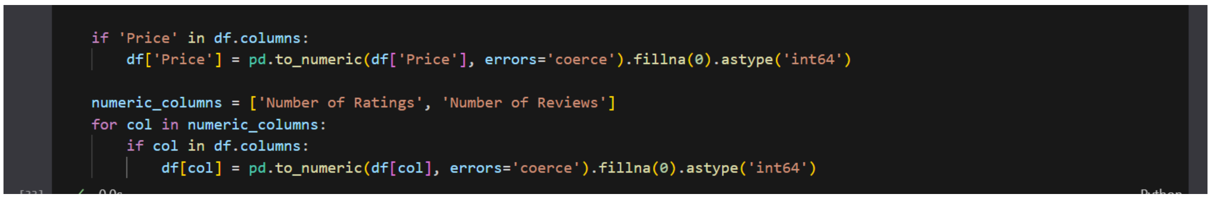


```
for col in ['ram_gb', 'ssd', 'hdd', 'graphic_card_gb', 'rating']:
    if col in df.columns:
        df[col] = df[col].astype(str).str.extract('(\d+)').fillna(0).astype('int64')

if 'Price' in df.columns:
```

4. Converting Data Types

To avoid type conflicts and ensure correct numerical computation, columns were converted to integers:

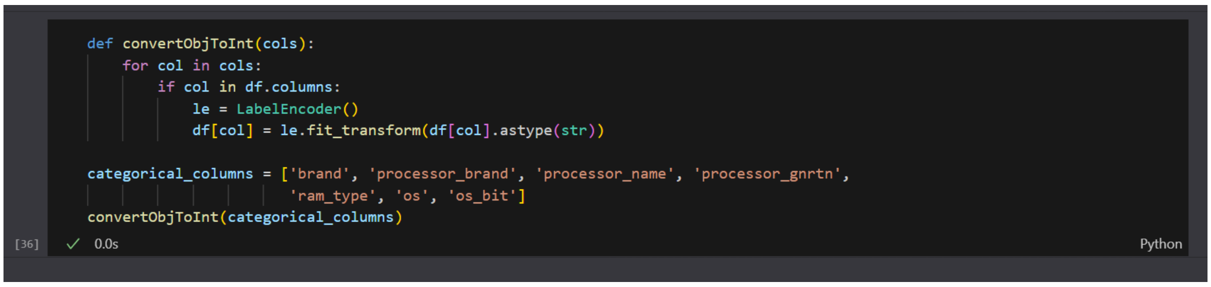


```python
if 'Price' in df.columns:
    df['Price'] = pd.to_numeric(df['Price'], errors='coerce').fillna(0).astype('int64')

numeric_columns = ['Number of Ratings', 'Number of Reviews']
for col in numeric_columns:
    if col in df.columns:
        df[col] = pd.to_numeric(df[col], errors='coerce').fillna(0).astype('int64')
```

5. Encoding Categorical Variables

Machine learning models cannot understand text labels, so they must be converted into numbers.
I used LabelEncoder to transform all major categorical columns:

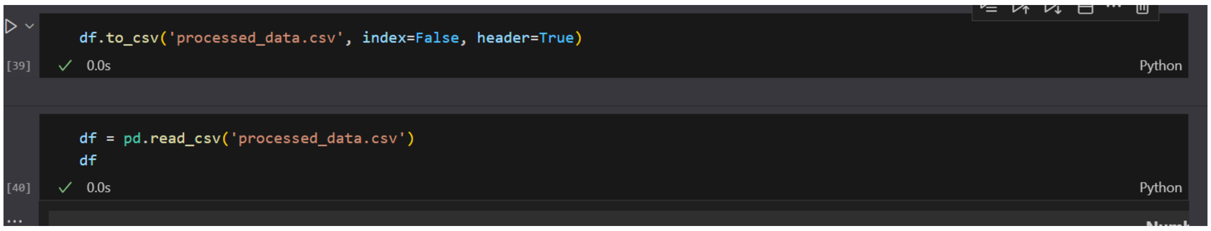


```python
def convertObjToInt(cols):
    for col in cols:
        if col in df.columns:
            le = LabelEncoder()
            df[col] = le.fit_transform(df[col].astype(str))

categorical_columns = ['brand', 'processor_brand', 'processor_name', 'processor_gnrtn',
                       'ram_type', 'os', 'os_bit']
convertObjToInt(categorical_columns)
```

6. Exporting the Final Processed Dataset

After all transformations, the cleaned dataset is saved for future use:



```python
df.to_csv('processed_data.csv', index=False, header=True)
```

```python
df = pd.read_csv('processed_data.csv')
df
```

Summary

In this data pre-processing section, the dataset was fully cleaned, structured, and transformed into a machine-readable format. By removing unnecessary columns, handling missing values, extracting numeric components, converting data types, and encoding categorical features, the dataset became consistent and reliable. The final cleaned file (processed_data.csv) is now ready for model training, visualization, and further analysis. This process ensures high accuracy, reduced errors, and better predictive performance in machine learning tasks.