**Sardar Patel Institute of Technology,Mumbai**

**Department of Computer Science Engineering**

**B.E. Sem-VII- PE-IV (2024-2025)**

**IT 24 - AI in Healthcare**

**Experiment5: Data Transformation and preparation for analysis of healthcare data**

**Name: Adwait Purao**  **Date: 7/10/24**

**Objective:**

To understand and apply data transformation techniques for preparing healthcare data for analysis.

Link to Dataset : <https://www.kaggle.com/datasets/prasad22/healthcare-dataset/data>

Link to Notebook:

<https://colab.research.google.com/drive/1w8aKUO0BfcGsN6VhiqxGmCIDwM6mJlG3?usp=sharing>

**1.Introduction to Healthcare Data:**

# **Dataset Information:**

**Each column provides specific information about the patient, their admission, and the healthcare services provided, making this dataset suitable for various data analysis and modeling tasks in the healthcare domain. Here's a brief explanation of each column in the dataset -**

* **Name: This column represents the name of the patient associated with the healthcare record.**
* **Age: The age of the patient at the time of admission, expressed in years.**
* **Gender: Indicates the gender of the patient, either "Male" or "Female."**
* **Blood Type: The patient's blood type, which can be one of the common blood types (e.g., "A+", "O-", etc.).**
* **Medical Condition: This column specifies the primary medical condition or diagnosis associated with the patient, such as "Diabetes," "Hypertension," "Asthma," and more.**
* **Date of Admission: The date on which the patient was admitted to the healthcare facility.**
* **Doctor: The name of the doctor responsible for the patient's care during their admission.**
* **Hospital: Identifies the healthcare facility or hospital where the patient was admitted.**
* **Insurance Provider: This column indicates the patient's insurance provider, which can be one of several options, including "Aetna," "Blue Cross," "Cigna," "UnitedHealthcare," and "Medicare."**
* **Billing Amount: The amount of money billed for the patient's healthcare services during their admission. This is expressed as a floating-point number.**
* **Room Number: The room number where the patient was accommodated during their admission.**
* **Admission Type: Specifies the type of admission, which can be "Emergency," "Elective," or "Urgent," reflecting the circumstances of the admission.**
* **Discharge Date: The date on which the patient was discharged from the healthcare facility, based on the admission date and a random number of days within a realistic range.**
* **Medication: Identifies a medication prescribed or administered to the patient during their admission. Examples include "Aspirin," "Ibuprofen," "Penicillin," "Paracetamol," and "Lipitor."**
* **Test Results: Describes the results of a medical test conducted during the patient's admission. Possible values include "Normal," "Abnormal," or "Inconclusive," indicating the outcome of the test.**

**2.Understanding the Dataset**

**Dataset Description:**

**import pandas as pd**

**def display\_step\_header(step\_name):**

**print(f"\n{'='\*20} {step\_name} {'='\*20}")**

**def load\_data(file\_path):**

**display\_step\_header("1. Data Loading")**

**df = pd.read\_csv(file\_path)**

**print("Dataset Overview:")**

**print(df.info())**

**print("\nSample of the dataset:")**

**print(df.head())**

**print("\nBasic statistics of numerical columns:")**

**print(df.describe())**

**# Display value counts for categorical columns**

**categorical\_columns = df.select\_dtypes(include=['object']).columns**

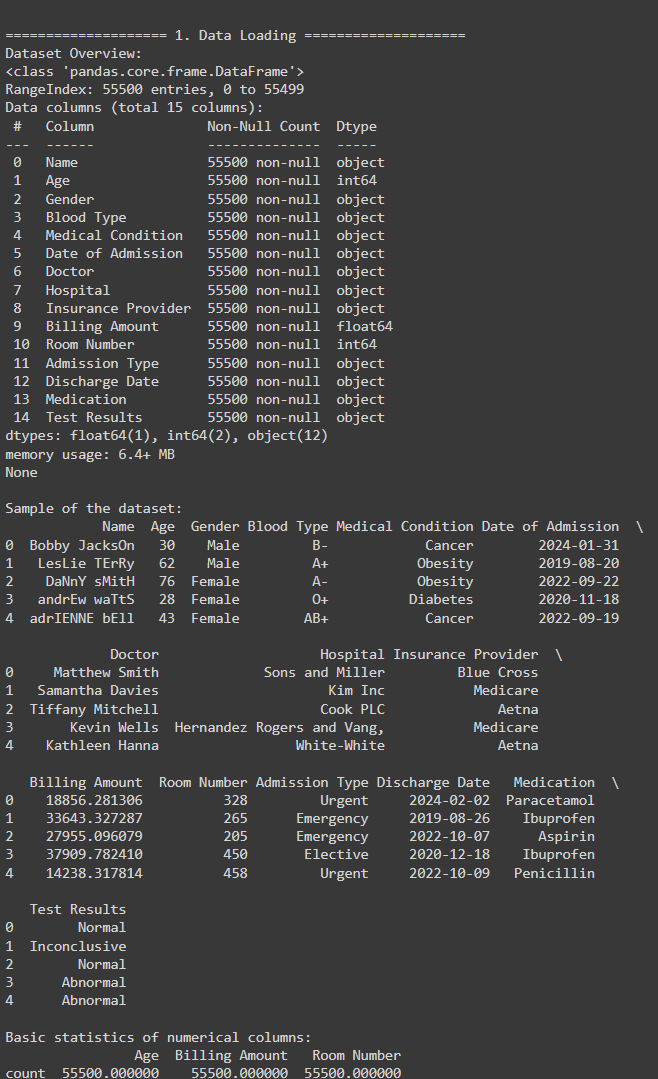
**for col in categorical\_columns:**

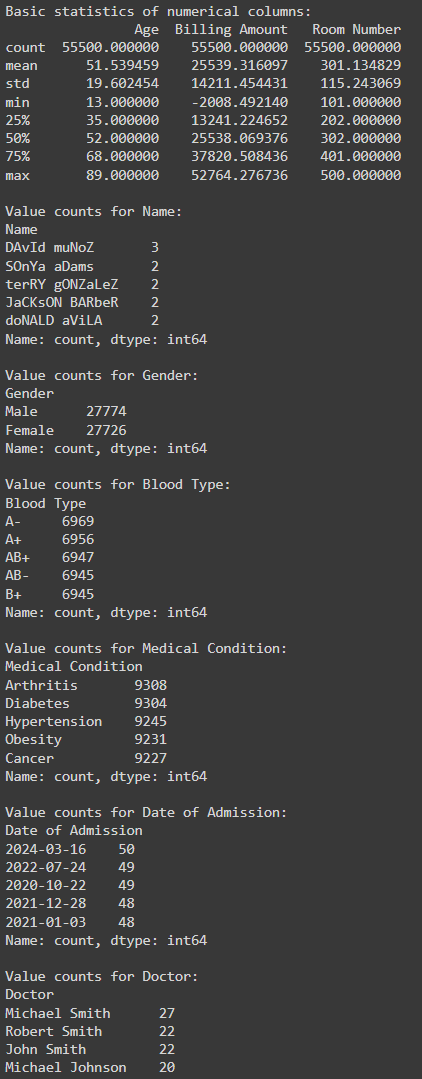
**print(f"\nValue counts for {col}:")**

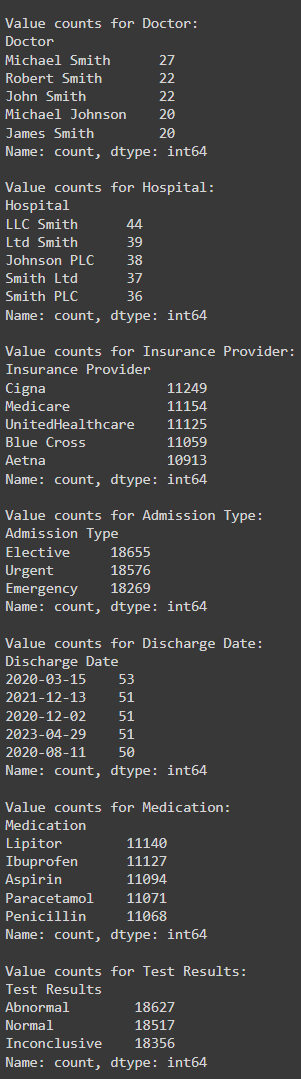
**print(df[col].value\_counts().head())**

**return df**

**df = load\_data('./healthcare\_dataset.csv')**

****

****

****

**3.Handling Missing data in the dataset:**

**Methods of Handling Missing data**

**def handle\_missing\_data(df):**

**display\_step\_header("2. Missing Data Handling")**

**print("Missing values before handling:")**

**print(df.isnull().sum())**

**# For numerical columns, fill with median**

**numeric\_columns = df.select\_dtypes(include=['int64', 'float64']).columns**

**for col in numeric\_columns:**

**df[col].fillna(df[col].median(), inplace=True)**

**# For categorical columns, fill with mode**

**categorical\_columns = df.select\_dtypes(include=['object']).columns**

**for col in categorical\_columns:**

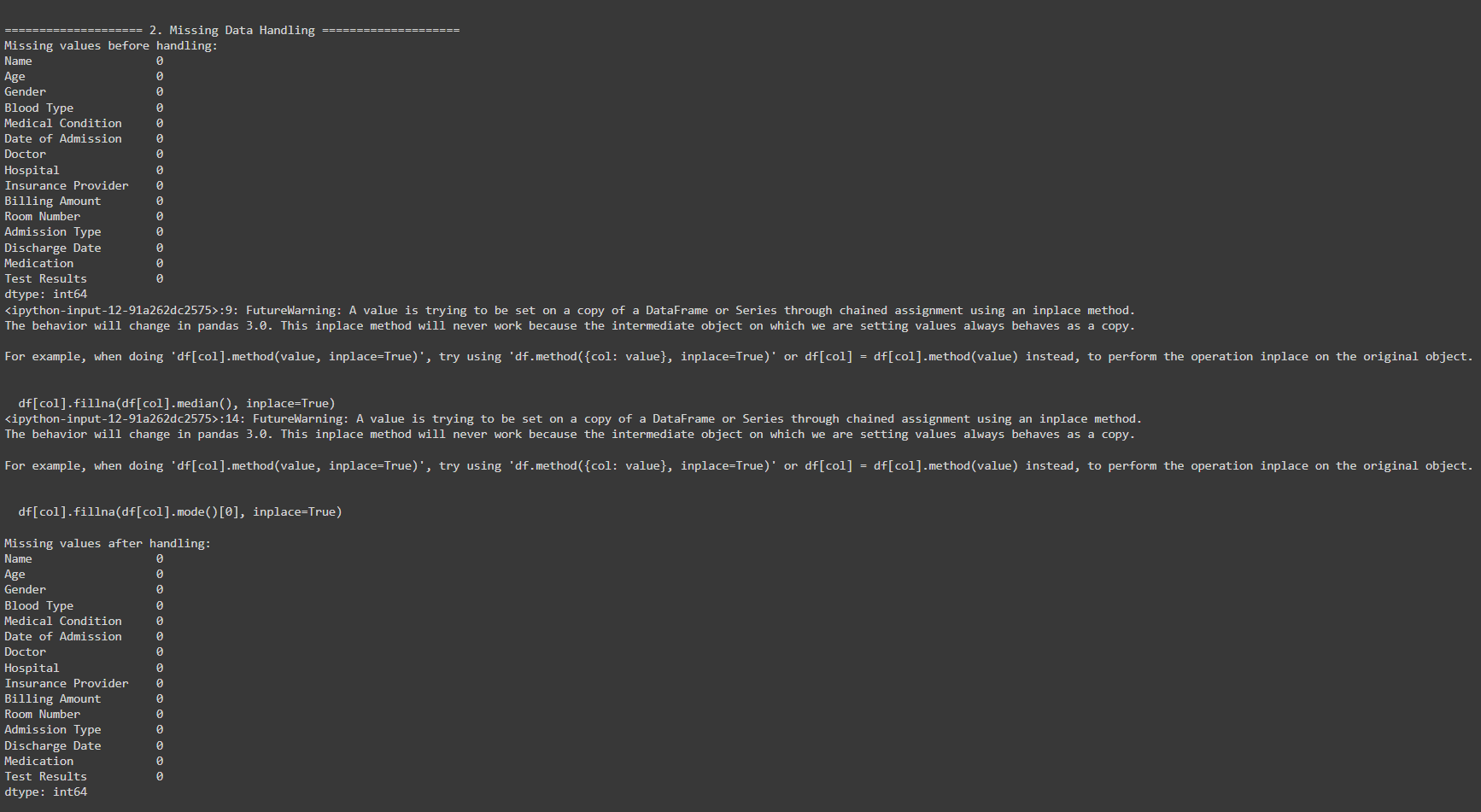
**df[col].fillna(df[col].mode()[0], inplace=True)**

**print("\nMissing values after handling:")**

**print(df.isnull().sum())**

**return df**

**df = handle\_missing\_data(df)**

****

**4.Categorical Data Encoding**

**from sklearn.preprocessing import LabelEncoder**

**def encode\_categorical\_data(df):**

**display\_step\_header("3. Categorical Data Encoding")**

**label\_encoders = {}**

**categorical\_columns = df.select\_dtypes(include=['object']).columns**

**# Create a sample dataframe to show transformations**

**sample\_df = pd.DataFrame()**

**for col in categorical\_columns:**

**label\_encoders[col] = LabelEncoder()**

**df[f'{col}\_encoded'] = label\_encoders[col].fit\_transform(df[col])**

**# Get unique categories and their encodings**

**unique\_categories = df[col].unique()[:3] # Take only first 3 unique values**

**unique\_encodings = [label\_encoders[col].transform([cat])[0] for cat in unique\_categories]**

**# Ensure sample\_df has the correct length**

**sample\_df = pd.DataFrame({col: unique\_categories}) # Create a DataFrame with the correct length**

**sample\_df[f'{col}\_encoded'] = unique\_encodings**

**print(f"\nEncoding example for {col} (showing first 3 categories):")**

**for orig, enc in zip(unique\_categories, unique\_encodings):**

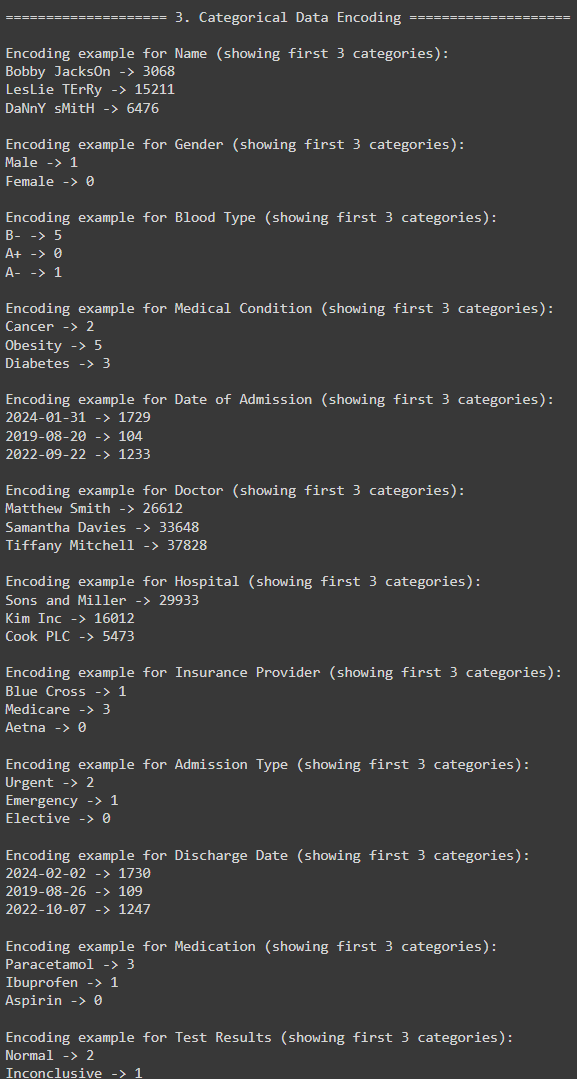
**print(f"{orig} -> {enc}")**

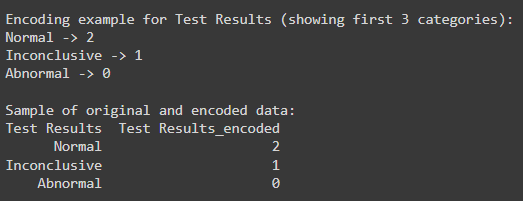
**print("\nSample of original and encoded data:")**

**print(sample\_df.to\_string(index=False))**

**return df, label\_encoders**

**df, label\_encoders = encode\_categorical\_data(df)**

****

****

**5. Outlier Detection and Treatment**

**def handle\_outliers(df, numeric\_columns):**

**display\_step\_header("4. Outlier Detection and Treatment")**

**for col in numeric\_columns:**

**Q1 = df[col].quantile(0.25)**

**Q3 = df[col].quantile(0.75)**

**IQR = Q3 - Q1**

**lower\_bound = Q1 - 1.5 \* IQR**

**upper\_bound = Q3 + 1.5 \* IQR**

**# Print outlier information**

**outliers = df[(df[col] < lower\_bound) | (df[col] > upper\_bound)]**

**print(f"\nOutliers in {col}:")**

**print(f"Number of outliers: {len(outliers)}")**

**print(f"Percentage of outliers: {(len(outliers)/len(df))\*100:.2f}%")**

**print(f"Lower bound: {lower\_bound:.2f}")**

**print(f"Upper bound: {upper\_bound:.2f}")**

**# Cap the outliers**

**df[f'{col}\_cleaned'] = df[col].clip(lower=lower\_bound, upper=upper\_bound)**

**# Display statistics before and after**

**print(f"\nStatistics for {col} before and after outlier treatment:")**

**print(pd.DataFrame({**

**'Original': df[col].describe(),**

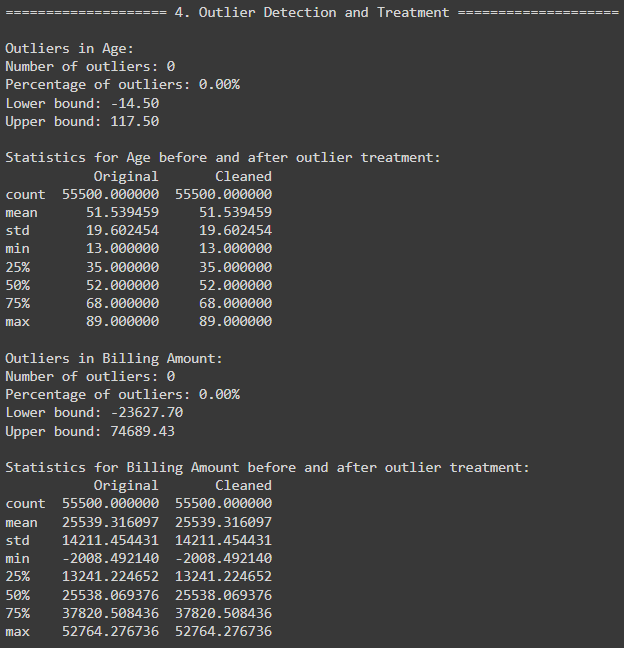
**'Cleaned': df[f'{col}\_cleaned'].describe()**

**}))**

**return df**

**numeric\_columns = ['Age', 'Billing Amount']**

**df = handle\_outliers(df, numeric\_columns)**

****

**6. Feature Scaling and Normalization**

**from sklearn.preprocessing import StandardScaler**

**def scale\_features(df, numeric\_columns):**

**display\_step\_header("5. Feature Scaling")**

**scaler = StandardScaler()**

**scaled\_columns = [col + '\_scaled' for col in numeric\_columns]**

**df[scaled\_columns] = scaler.fit\_transform(df[numeric\_columns])**

**for original, scaled in zip(numeric\_columns, scaled\_columns):**

**print(f"\nScaling results for {original}:")**

**print(pd.DataFrame({**

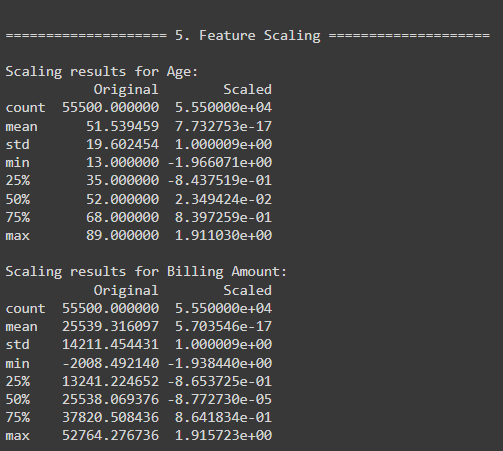
**'Original': df[original].describe(),**

**'Scaled': df[scaled].describe()**

**}))**

**return df, scaler**

**df, scaler = scale\_features(df, numeric\_columns)**

****

**7. Feature Engineering**

* Create new features (e.g., BMI from weight and height).

def engineer\_features(df):

display\_step\_header("6. Feature Engineering")

# Age groups

df['Age\_Group'] = pd.cut(df['Age'], bins=[0, 18, 35, 50, 65, 100],

labels=['0-18', '19-35', '36-50', '51-65', '65+'])

print("\nAge Group Distribution:")

print(df['Age\_Group'].value\_counts())

# Chronic condition flag

df['Is\_Chronic'] = df['Medical Condition'].apply(lambda x: 1 if 'chronic' in str(x).lower() else 0)

print("\nChronic Condition Distribution:")

print(df['Is\_Chronic'].value\_counts(normalize=True))

# Cost per age

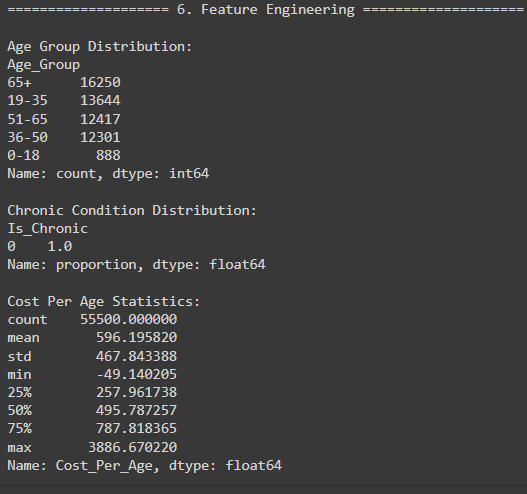
df['Cost\_Per\_Age'] = df['Billing Amount'] / df['Age']

print("\nCost Per Age Statistics:")

print(df['Cost\_Per\_Age'].describe())

return df

df = engineer\_features(df)



**8. Data Preparation for Machine Learning Models:**

#### Train-Test Split

**from sklearn.model\_selection import train\_test\_split**

**def prepare\_for\_ml(df, target\_column='Billing Amount'):**

**display\_step\_header("7. Preparing Data for Machine Learning")**

**numeric\_features = df.select\_dtypes(include=['int64', 'float64']).columns**

**categorical\_features = [col for col in df.columns if '\_encoded' in col]**

**features = list(numeric\_features) + categorical\_features**

**features = [f for f in features if f != target\_column]**

**X = df[features]**

**y = df[target\_column]**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)**

**print("Selected features:")**

**for f in features:**

**print(f"- {f}")**

**print(f"\nTarget variable statistics:")**

**print(y.describe())**

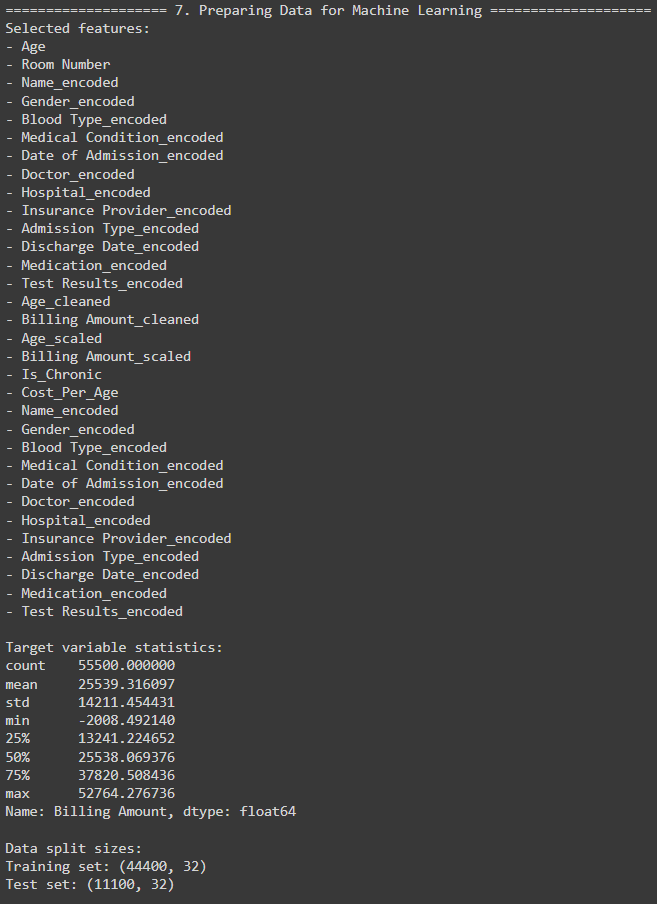
**print("\nData split sizes:")**

**print(f"Training set: {X\_train.shape}")**

**print(f"Test set: {X\_test.shape}")**

**return X\_train, X\_test, y\_train, y\_test**

**X\_train, X\_test, y\_train, y\_test = prepare\_for\_ml(df)**

****

### **Conclusion:**

In this assignment, I learned how to effectively handle real-world data by applying essential preprocessing steps such as handling missing values, encoding categorical variables, detecting and treating outliers, scaling features, and engineering new features. I also gained insights into how these steps impact the data's quality and readiness for machine learning. Through each step, I enhanced my understanding of preparing datasets for analysis and model building, ensuring the data is clean, consistent, and well-structured for successful predictions.