Simple Sales Data Visualization

Name: Harsh

Unique ID: 202401100300113

Branch: CSE(AI)

Subject: Introduction to AI

Subject Code: AI101B

Healthcare Data Cleaning

Introduction

In today's data-driven world, healthcare analytics plays a crucial role in improving patient care, detecting trends, and making informed medical decisions. However, raw healthcare data often contains missing values, duplicates, and outliers, which can impact the accuracy of analysis. Therefore, proper data preprocessing is essential to ensure reliability and consistency.

To address these challenges, this project focuses on cleaning and standardizing healthcare data, including patient information such as blood pressure, sugar levels, and weight. The process involves handling missing values, removing duplicate entries, identifying and eliminating outliers using the Z-score method, and standardizing numerical features for better analysis. This cleaned and structured data can be further utilized for medical research, predictive analytics, and machine learning applications in healthcare.

Problem Statement

In the healthcare industry, data quality is critical for accurate diagnosis, treatment planning, and medical research. However, raw healthcare data often contains inconsistencies, such as missing values, duplicate records, and outliers, which can lead to inaccurate analysis and flawed decision-making. Some key challenges include:

- **Incomplete Data:** Missing values in patient records can result in biased outcomes and affect predictive analytics.
- **Data Redundancy:** Duplicate entries can skew statistical results and lead to inefficiencies in data processing.
- **Presence of Outliers:** Extreme values in medical parameters like blood pressure and sugar levels can distort trends and impact clinical insights.
- **Lack of Standardization:** Variability in numerical data across different patient records makes comparison and machine learning applications challenging.

To overcome these issues, this project focuses on cleaning and preprocessing healthcare data, ensuring its reliability for further analysis and decision-making.

T This project aims to address these challenges by developing an efficient and user-friendly **healthcare data cleaning and preprocessing tool**. The tool will automatically handle missing values, remove duplicate records, detect and eliminate outliers, and standardize numerical data, ensuring data consistency and accuracy. By automating these preprocessing steps, the project enhances the reliability of healthcare data for further analysis, clinical decision-making, and machine learning applications.

Methodology

To achieve the objectives, we implemented the project using **Python**, a widely used programming language for data processing and analysis. The following steps outline the approach used to clean and preprocess the healthcare data:

1. **Data Collection & Creation**
    - A sample dataset was created containing **PatientID, Age, BloodPressure, SugarLevel, and Weight** attributes.
2. **Handling Missing Values**
    - Missing values, if present, were **replaced with the column mean** to maintain data consistency and avoid bias in analysis.
3. **Removing Duplicate Entries**
    - Duplicate records were **identified and removed** to ensure data integrity and avoid redundant information.
4. **Outlier Detection & Removal**
    - The **Z-score method** was used to detect outliers in numerical attributes (BloodPressure, SugarLevel, and Weight).
    - Any data points with a **Z-score greater than 3** were considered outliers and removed from the dataset.
5. **Data Standardization**
    - The **StandardScaler from Scikit-Learn** was applied to normalize BloodPressure, SugarLevel, and Weight.
    - Standardization ensures that these values have a **mean of 0** and a **standard deviation of 1**, improving the efficiency of machine learning models.
6. **Displaying Cleaned Data**
    - After preprocessing, the cleaned dataset was displayed for verification and further analysis.

This structured methodology ensures that the healthcare dataset is well-prepared for analysis, reducing errors and enhancing the accuracy of any predictive models used in healthcare applications.

Implementation

The project was implemented using **Python** with the following essential libraries:

- **Pandas**: Used for data handling, manipulation, and cleaning.
- **NumPy**: Used for numerical computations, including handling missing values and outlier detection.
- **SciPy**: Used for statistical functions, particularly the **Z-score method** to detect and remove outliers.
- **Scikit-Learn**: Used for **data standardization**, ensuring numerical attributes have a uniform scale.

Implementation Steps:

1. **Dataset Creation:**
   - A **sample dataset** was created containing **PatientID, Age, BloodPressure, SugarLevel, and Weight** values.
2. **Handling Missing Values:**
   - Any missing values were replaced using the **mean of the respective column** to maintain consistency.
3. **Removing Duplicates:**
   - The dataset was checked for **duplicate entries**, and any duplicates found were removed to prevent redundancy.
4. **Outlier Detection & Removal:**
   - The **Z-score method** was used to detect outliers in **BloodPressure, SugarLevel, and Weight** columns.
   - Any value with a **Z-score greater than 3** was considered an outlier and removed from the dataset.
5. **Data Standardization:**
   - The **StandardScaler from Scikit-Learn** was applied to **BloodPressure, SugarLevel, and Weight** to ensure that values have a mean of **0** and a standard deviation of **1**.
6. **Displaying Cleaned Data:**
   - The final cleaned dataset was displayed in the output, ensuring it is **ready for further analysis or machine learning applications**.

This structured approach ensures that the healthcare dataset is clean, consistent, and well-prepared for further data analysis and visualization.

Code

```
import pandas as pd
import numpy as np
from scipy import stats
from sklearn.preprocessing import StandardScaler

# Creating DataFrame
data = {
    "PatientID": range(1, 21),
    "Age": [44, 39, 49, 58, 35, 25, 46, 28, 60, 55, 41, 48, 58, 35, 67, 70,
43, 74, 19, 56],
    "BloodPressure": [118, 109, 149, 121, 109, 129, 132, 93, 145, 125, 143,
141, 93, 145, 176, 109, 148, 122, 147, 119],
    "SugarLevel": [87.89, 177.32, 144.14, 90.35, 126.42, 95.27, 146.60,
109.75, 103.19, 197.72,
                   180.57, 181.97, 181.78, 133.38, 87.00, 193.27, 135.93,
129.41, 125.48, 160.71],
    "Weight": [105.57, 105.70, 77.78, 115.24, 70.38, 119.05, 62.17, 81.79,
94.63, 118.59,
               103.58, 61.45, 50.68, 113.18, 84.93, 77.71, 106.57, 83.30,
74.08, 111.86]
}

df = pd.DataFrame(data)

# 1. Check for Missing Values and Fill if Necessary
df.fillna(df.mean(), inplace=True)  # Filling with mean (if there are missing
values)

# 2. Remove Duplicates
df.drop_duplicates(inplace=True)

# 3. Handle Outliers using Z-score
z_scores = np.abs(stats.zscore(df[['BloodPressure', 'SugarLevel',
'Weight']]))  # Compute Z-score
df_cleaned = df[(z_scores < 3).all(axis=1)]  # Keep values with Z-score < 3
(removes extreme outliers)

# 4. Standardizing Data
scaler = StandardScaler()
df_cleaned[['BloodPressure', 'SugarLevel', 'Weight']] =
scaler.fit_transform(df_cleaned[['BloodPressure', 'SugarLevel', 'Weight']])

# Display cleaned dataset
print("\nCleaned Healthcare Data:\n", df_cleaned)
```

Output

```
Cleaned Healthcare Data:
     PatientID  Age  BloodPressure  SugarLevel    Weight
0            1   44      -0.522960   -1.428130  0.711912
1            2   39      -0.964897    1.050982  0.718226
2            3   49       0.999270    0.131191 -0.637797
3            4   58      -0.375647   -1.359936  1.181566
4            5   35      -0.964897   -0.360030 -0.997201
5            6   25       0.017186   -1.223547  1.366611
6            7   46       0.164499    0.199385 -1.395946
7            8   28      -1.750565   -0.822143 -0.443039
8            9   60       0.802854   -1.003995  0.180576
9           10   55      -0.179230    1.616496  1.344269
10          11   41       0.704645    1.141076  0.615261
11          12   48       0.606437    1.179886 -1.430915
12          13   58      -1.750565    1.174619 -1.953994
13          14   35       0.802854   -0.167090  1.081515
14          15   67       2.325084   -1.452802 -0.290535
15          16   70      -0.964897    1.493136 -0.641197
16          17   43       0.950166   -0.096401  0.760480
17          18   74      -0.326543   -0.277143 -0.369701
18          19   19       0.901062   -0.386088 -0.817499
19          20   56      -0.473855    0.590532  1.017406
PS C:\Users\DELL\OneDrive\Desktop\harshcseai>
```

Conclusion

The **Healthcare Data Processing and Cleaning** project provides an efficient way to handle, analyze, and standardize patient data. By leveraging **Python's Pandas, NumPy, SciPy, and Scikit-Learn libraries**, the project ensures that raw healthcare data is **clean, accurate, and ready for analysis**.

Through **missing value handling, duplicate removal, outlier detection using Z-score, and data standardization**, the dataset becomes more reliable and structured for further medical research or predictive modeling. This approach simplifies **data preprocessing**, making it easier for healthcare professionals and data analysts to derive meaningful insights from patient records.

Future Enhancements

To further improve the **Healthcare Data Processing and Cleaning** project, the following enhancements can be considered:

- **Advanced Missing Value Imputation**: Instead of replacing missing values with the mean, use **machine learning models** or **K-Nearest Neighbors (KNN) imputation** for more accurate predictions.
- **Additional Outlier Detection Methods**: Implement alternative techniques like **Interquartile Range (IQR) method** or **Isolation Forest** for more robust outlier detection.
- **Feature Engineering**: Extract new features from existing data, such as **BMI (Body Mass Index)** from weight and height, to improve healthcare insights.
- **Automated Data Visualization**: Integrate libraries like **Matplotlib and Seaborn** to generate visual reports of blood pressure, sugar levels, and weight distributions.
- **Web-based Interface**: Develop a **user-friendly web application** using Flask or Django to allow easy data uploading, processing, and visualization for non-technical users.
- **Integration with Electronic Health Records (EHRs)**: Enable seamless connection with hospital databases to **automate data retrieval and preprocessing** for real-time analysis.

These improvements will enhance the usability, accuracy, and scalability of the project, making it more valuable for healthcare professionals and researchers.