

Healthcare Data Cleaning for Disease Prediction Accuracy

Title:

Healthcare Data Cleaning: Improving Disease Prediction Accuracy by Handling Missing, Inconsistent, and Noisy Patient Data

Name:

[Sankalp Saxena]

Institution:

[KIET group of institutions]

Date:

[10-03-2025]

2. Introduction

In healthcare, accurate disease prediction is crucial for improving patient care and outcomes. However, the data used to

predict diseases often contains missing, inconsistent, or noisy values. Such issues can adversely affect the performance of prediction models. The purpose of this project is to improve disease prediction accuracy by effectively handling missing, inconsistent, and noisy data in a healthcare dataset.

Data cleaning is a crucial preprocessing step in machine learning workflows. By applying data cleaning techniques such as imputation of missing values, correcting inconsistent data, and reducing noise, we can ensure that the dataset is accurate, consistent, and ready for modeling. This report outlines the methodology for cleaning the dataset and presents the outcomes of the data cleaning process.

3. Methodology

The methodology followed in this project involves the following steps:

Step 1: Data Collection

A **dummy healthcare dataset** was created to simulate real-world healthcare data.

This dataset includes patient information such as age, blood pressure, cholesterol levels, presence of symptoms, and disease outcomes (whether the patient has a disease or not).

Step 2: Data Cleaning

The dataset was cleaned by performing the following tasks:

1. **Handling Missing Data:**
 - **Numerical columns** (age, blood pressure, cholesterol) with missing values were imputed using the **median** value.

- **Categorical columns** (symptom) with missing values were imputed using the **most frequent value**.
- 2. **Data Encoding:**
 - The **categorical column** 'symptom' was encoded using **LabelEncoder**, converting values such as 'Y' and 'N' to numerical values (1 and 0).

Step 3: Data Analysis and Visualization

After cleaning the data, several visualizations were created to gain insights into the distribution of the data and potential patterns:

- **Histograms** for numerical columns (e.g., age, blood pressure, cholesterol).
- **Bar plot** to show the distribution of disease outcomes.
- **Box plots** to observe the spread and outliers of numerical data.

- **Pair plots** to analyze relationships between different variables in the dataset.

Step 4: Implementation

The data cleaning process was implemented using Python, specifically with the following libraries:

- **Pandas** for data manipulation.
- **NumPy** for handling missing values (`np.nan`).
- **Scikit-learn** for imputation (`SimpleImputer`) and encoding (`LabelEncoder`).
- **Matplotlib** and **Seaborn** for data visualization.

4. Code Typed

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
import seaborn as sns
from sklearn.impute import
SimpleImputer # Importing
SimpleImputer

# Create a dummy dataset
data = {
    'age': [25, 30, 35, 40, 45, 50, np.nan,
60, 65, 70],
    'blood_pressure': [120, 130, 140, 150,
np.nan, 160, 170, 180, 190, 200],
    'cholesterol': [200, 220, 240, 250, 260,
270, 280, 290, np.nan, 310],
    'symptom': ['Y', 'N', 'Y', 'Y', 'N', 'Y', 'Y', 'N',
np.nan, 'Y'],
    'disease_outcome': [0, 1, 0, 1, 0, 1, 1, 0,
1, 0] # 0 = No disease, 1 = Disease
}

# Create a DataFrame
```

```
df = pd.DataFrame(data)
```

```
# Save the dummy data to a CSV file  
df.to_csv('healthcare_data.csv',  
index=False)
```

```
# Preview the dataset  
print(df)
```

```
# Data Cleaning: Handling missing values  
(as shown in the previous code)
```

```
# Impute numerical columns with the  
median value  
numerical_cols =  
df.select_dtypes(include=[np.number]).columns
```

```
imputer_num =  
SimpleImputer(strategy='median') # Use  
SimpleImputer to handle missing data
```

```
df[numerical_cols] =  
imputer_num.fit_transform(df[numerical_  
_cols])
```

```
# Impute categorical columns with the  
most frequent value (mode)  
categorical_cols =  
df.select_dtypes(include=[object]).column  
ns  
imputer_cat =  
SimpleImputer(strategy='most_frequent')  
df[categorical_cols] =  
imputer_cat.fit_transform(df[categorical_  
cols])
```

```
# Encoding Categorical Data  
from sklearn.preprocessing import  
LabelEncoder # Importing LabelEncoder  
for encoding categorical data
```



```
encoder = LabelEncoder()
for col in categorical_cols:
    df[col] = encoder.fit_transform(df[col])
```

Now we can create figures to visualize the cleaned dataset.

```
# Set up the plotting environment
plt.figure(figsize=(10, 6))
```

```
# 1. Histogram of numerical columns
plt.subplot(2, 2, 1) # 2 rows, 2 columns,
1st plot
df['age'].plot(kind='hist', bins=10,
color='skyblue', edgecolor='black',
alpha=0.7)
plt.title('Age Distribution')
plt.xlabel('Age')
plt.ylabel('Frequency')
```

2. Bar plot of disease_outcome
distribution

plt.subplot(2, 2, 2) # 2 rows, 2 columns,
2nd plot

```
df['disease_outcome'].value_counts().plot  
(kind='bar', color=['skyblue', 'lightgreen'],  
edgecolor='black')
```

```
plt.title('Disease Outcome Distribution')
```

```
plt.xlabel('Disease Outcome')
```

```
plt.ylabel('Count')
```

3. Box plot of blood_pressure and
cholesterol

plt.subplot(2, 2, 3) # 2 rows, 2 columns,
3rd plot

```
sns.boxplot(data=df[['blood_pressure',  
'cholesterol']], palette='Set2')
```

```
plt.title('Blood Pressure & Cholesterol  
Distribution')
```

```
plt.ylabel('Value')
```

```
# 4. Pair plot to visualize relationships
between variables
plt.subplot(2, 2, 4) # 2 rows, 2 columns,
4th plot
sns.pairplot(df[['age', 'blood_pressure',
'cholesterol', 'disease_outcome']],
hue='disease_outcome',
palette='coolwarm')
plt.title('Pair Plot: Features vs Disease
Outcome')

# Adjust layout for better spacing
plt.tight_layout()

# Show all the plots
plt.show()
```

5. Screenshots / Output Photos

