

WEEK 2 LAB B Activity: Heart Disease Data Cleaning & EDA

Overview

This lab guides you through a complete workflow for handling a messy heart disease dataset. You will:

- Perform EDA **before** cleaning
- Apply a **modular cleaning pipeline**
- Perform EDA **after** cleaning
- Save the cleaned dataset

All steps use Python and are suitable for Jupyter Notebook or VS Code.

1. Import Libraries

```
import pandas as pd
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

2. Load the Dataset

```
def load_data(filepath: str) -> pd.DataFrame:
```

```
    df = pd.read_csv(filepath, na_values=["?", "null", "NULL", "NaN", "nan", ""])
```

```
    return df
```

```
filepath = "heart_disease.csv"
```

```
df_raw = load_data(filepath)
```

3. EDA Utilities

Reusable functions for visual and statistical exploration.

```
def basic_eda(df, title_prefix=""):
```

```
    print(f"{title_prefix} BASIC EDA")
```

```
    print("Shape:", df.shape)
```

```
    print(df.info())
```

```
print(df.isna().sum())
```

```
print(df.describe())
```

Plot Functions

```
def plot_target_distribution(df, title_prefix=""):
```

```
    df['target'].value_counts().sort_index().plot(kind='bar')
```

```
    plt.title(f"{title_prefix}Heart Disease Distribution")
```

```
    plt.xlabel("Target (0 = No Disease, 1 = Disease)")
```

```
    plt.ylabel("Count")
```

```
    plt.show()
```

```
def plot_age_distribution(df, title_prefix=""):
```

```
    df['age'].dropna().astype(float).hist(bins=20)
```

```
    plt.title(f"{title_prefix}Age Distribution")
```

```
    plt.xlabel("Age")
```

```
    plt.ylabel("Frequency")
```

```
    plt.show()
```

```
def plot_chol_vs_target(df, title_prefix=""):
```

```
    plt.scatter(df['chol'], df['target'], alpha=0.7)
```

```
    plt.title(f"{title_prefix}Cholesterol vs Heart Disease")
```

```
    plt.xlabel("Cholesterol")
```

```
    plt.ylabel("Heart Disease (Target)")
```

```
    plt.show()
```

```
def plot_cp_vs_target(df, title_prefix=""):
```

```
    df.groupby('cp')['target'].mean().plot(kind='bar')
```

```
    plt.title(f"{title_prefix}Chest Pain Type vs Heart Disease Rate")
```

```
    plt.xlabel("Chest Pain Type (cp)")
```

```
plt.ylabel("Heart Disease Probability")
plt.show()
```

```
def plot_correlation_matrix(df, title_prefix=""):
    numeric_df = df.select_dtypes(include=[np.number])
    corr = numeric_df.corr()
    plt.imshow(corr, cmap='coolwarm', vmin=-1, vmax=1)
    plt.colorbar()
    plt.title(f"{title_prefix}Feature Correlation Matrix")
    plt.xticks(range(len(corr.columns)), corr.columns, rotation=90)
    plt.yticks(range(len(corr.columns)), corr.columns)
    plt.show()
```

Full EDA Wrapper

```
def full_eda(df, title_prefix=""):
    basic_eda(df, title_prefix)
    plot_target_distribution(df, title_prefix)
    plot_age_distribution(df, title_prefix)
    plot_chol_vs_target(df, title_prefix)
    plot_cp_vs_target(df, title_prefix)
    plot_correlation_matrix(df, title_prefix)
```

4. Cleaning Functions

Standardise Column Names

```
def standardise_column_names(df):
    df = df.copy()
    df.columns = df.columns.str.strip().str.lower()
    return df
```

Clean Categorical Values

```
def clean_categorical_values(df):
```

```

df = df.copy()

if 'sex' in df.columns:
    df['sex'] = df['sex'].astype(str).str.strip().str.lower().replace({
        'm': 'male', 'male': 'male', 'f': 'female', 'female': 'female'
    })

if 'cp' not in df.columns and 'chestpaintype' in df.columns:
    df['cp'] = df['chestpaintype']

if 'cp' in df.columns:
    df['cp'] = df['cp'].astype(str).str.strip().str.upper()

return df

```

Convert Numeric Columns

```

def convert_numeric_columns(df):
    df = df.copy()

    numeric_cols = ['age', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'target']

    for col in numeric_cols:
        if col in df.columns:
            df[col] = pd.to_numeric(df[col], errors='coerce')

    return df

```

Drop Irrelevant Columns

```

def drop_irrelevant_columns(df):
    df = df.copy()

    for col in ['notes', 'extra_col']:
        if col in df.columns:
            df = df.drop(columns=[col])

    if 'cp' in df.columns and 'chestpaintype' in df.columns:
        df = df.drop(columns=['chestpaintype'])

    return df

```

Handle Missing Values

```
def handle_missing_values(df):  
    df = df.copy()  
    df = df.dropna(subset=['target'])  
    key_numeric = ['age', 'trestbps', 'chol', 'thalach']  
    df = df.dropna(subset=[c for c in key_numeric if c in df.columns])  
    return df
```

Remove Duplicates

```
def remove_duplicates(df):  
    df = df.copy()  
    df = df.drop_duplicates()  
    return df
```

Remove Outliers

```
def remove_extreme_outliers(df):  
    df = df.copy()  
    if 'chol' in df.columns:  
        df = df[df['chol'] < 600]  
    if 'age' in df.columns:  
        df = df[(df['age'] >= 18) & (df['age'] <= 100)]  
    return df
```

Full Cleaning Pipeline

```
def full_cleaning_pipeline(df):  
    df = standardise_column_names(df)  
    df = clean_categorical_values(df)  
    df = convert_numeric_columns(df)  
    df = drop_irrelevant_columns(df)  
    df = handle_missing_values(df)  
    df = remove_duplicates(df)
```

```
df = remove_extreme_outliers(df)

return df
```

5. Run EDA Before Cleaning

```
full_eda(df_raw, title_prefix="(Before Cleaning) ")
```

6. Clean the Dataset

```
df_clean = full_cleaning_pipeline(df_raw)
```

7. Run EDA After Cleaning

```
full_eda(df_clean, title_prefix="(After Cleaning) ")
```

8. Save Cleaned Dataset

```
df_clean.to_csv("cleaned_heart_disease.csv", index=False)

print("Cleaned dataset save")
```