Data Cleaning, Preprocessing & Visualization (Healthcare Dataset)

Overview

The project description provides a thorough analysis of a genuine Healthcare dataset, i.e., the completion of all data cleansing and preprocessing tasks, in addition to creating an exploratory visualisation of the resultant data set. Included in this project workflow is the handling of missing or incomplete data, treatment of outliers, encoding of categorical variables, transformation of numerical variables, as well as generating statistical plots that will allow users to understand how their data is arranged.

The final cleaned and formatted data file will include the following columns:

- **Age**
- **Blood_Pressure**
- **Cholesterol**
- **Gender_Male** (boolean: True/False)
- **Condition_Hypertension** (boolean: True/False)

Steps Performed

1. Load Dataset

- Read the healthcare CSV file.

  -> df=pd.read_csv('healthcare_datasets.csv')

- Replaced `"NAN"` strings with actual `NaN`.

  -> df=df.replace("NAN",np.nan)

2. Remove Duplicates

- All duplicate records were dropped to maintain dataset integrity.

  -> df=df.drop_duplicates()

3. Handle Missing Values

- Converted numeric columns to proper numeric types.

  -> df[‘Blood_Pressure’]=pd.to_numeric(df['Blood_Pressure'], *errors*='coerce')

  df[‘Cholesterol’]=pd.to_numeric(df['Cholesterol'], *errors*='coerce')

  df[‘Age']=pd.to_numeric(df['Age'], *errors*='coerce')

- **Numerical values** (Age, Blood Pressure, Cholesterol) → filled using **median**.

  -> numerical_cols=df.select_dtypes(*include*=[‘int64','float64']).columns

  categorical_cols=df.select_dtypes(*include*=[‘object']).columns

  -> *for* col *in* numerical_cols:

  df[col].fillna(df[col].median())

- **Categorical values** (Gender, Condition) → filled using **mode**.

  -> *for* col *in* categorical_cols:

  df[col].fillna(df[col].mode()[0])

4. Remove Columns with Excessive Missing Data

- Columns with **more than 40% missing values** were dropped.

  -> threshold=0.4

  drop_col=df.columns[df.isnull().mean()>threshold]

  df=df.drop(*columns*=drop_col)

  print("Dropped columns: “, drop_col.tolist())

5. Outlier Detection & Treatment

Applied the **IQR (Interquartile Range)** method:

- Calculated Q1, Q3, and IQR.
- Any value outside `[Q1 - 1.5*IQR, Q3 + 1.5*IQR]` was clipped.
- Applied on:
  - Age
  - Blood Pressure
  - Cholesterol

-> def clip_outliers(*column*):

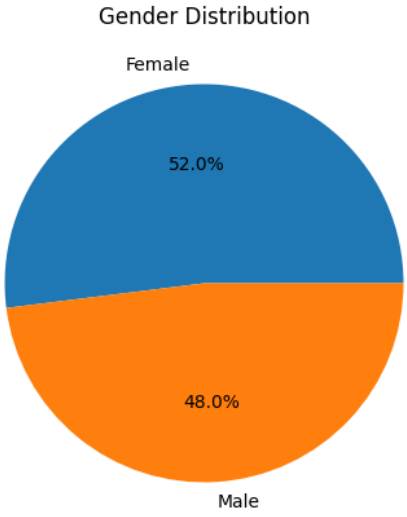
```
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR

    df[column] = np.where(df[column] < lower, lower,
            np.where(df[column] > upper, upper, df[column]))
```

for col *in* ["Age", "Blood_Pressure", "Cholesterol"]:
    clip_outliers(col)

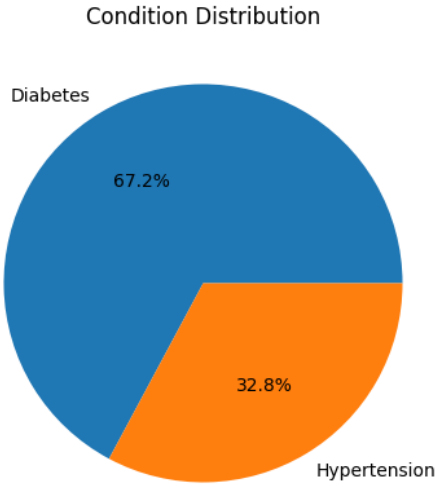
6. Data Visualization

Pie Charts

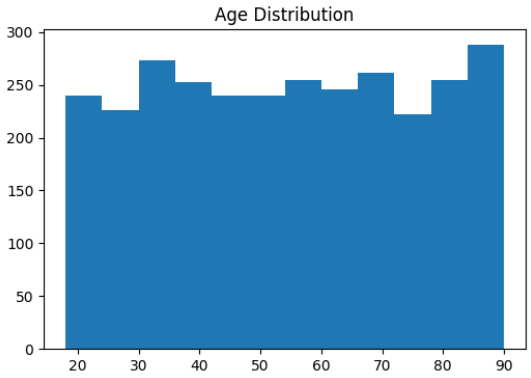
- Gender Distribution

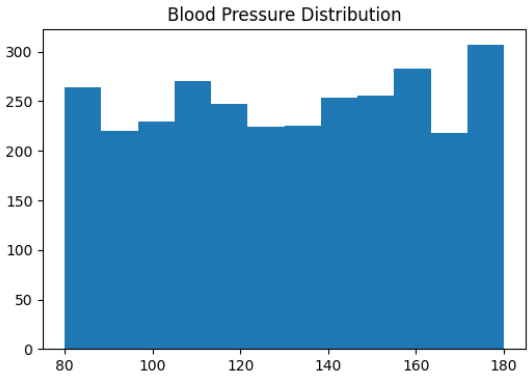


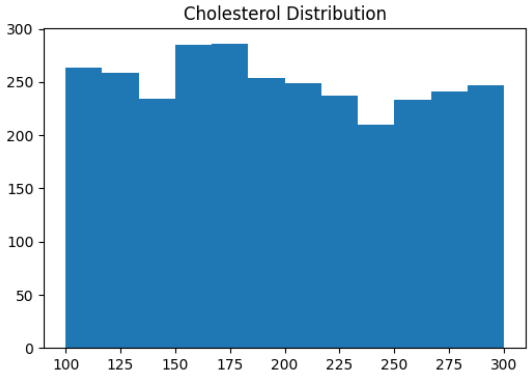
Condition Distribution



Histograms

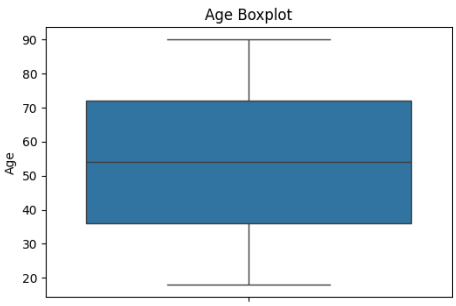


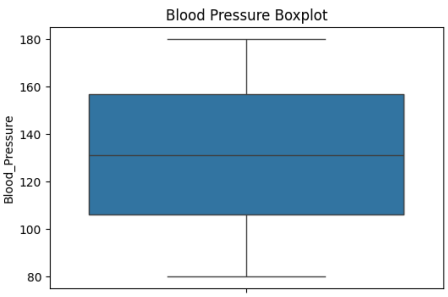


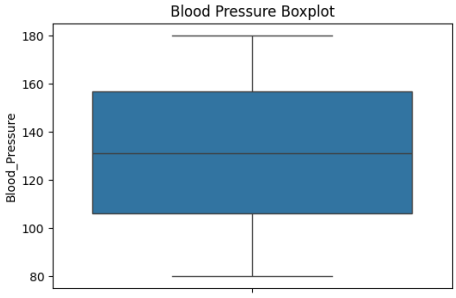


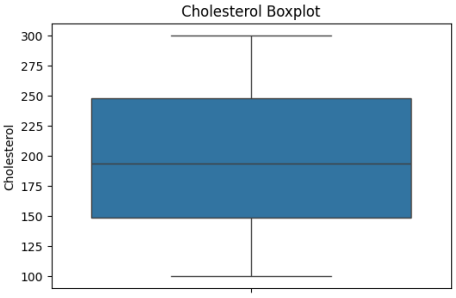
Boxplots

- Used for visualizing outliers in numeric columns.



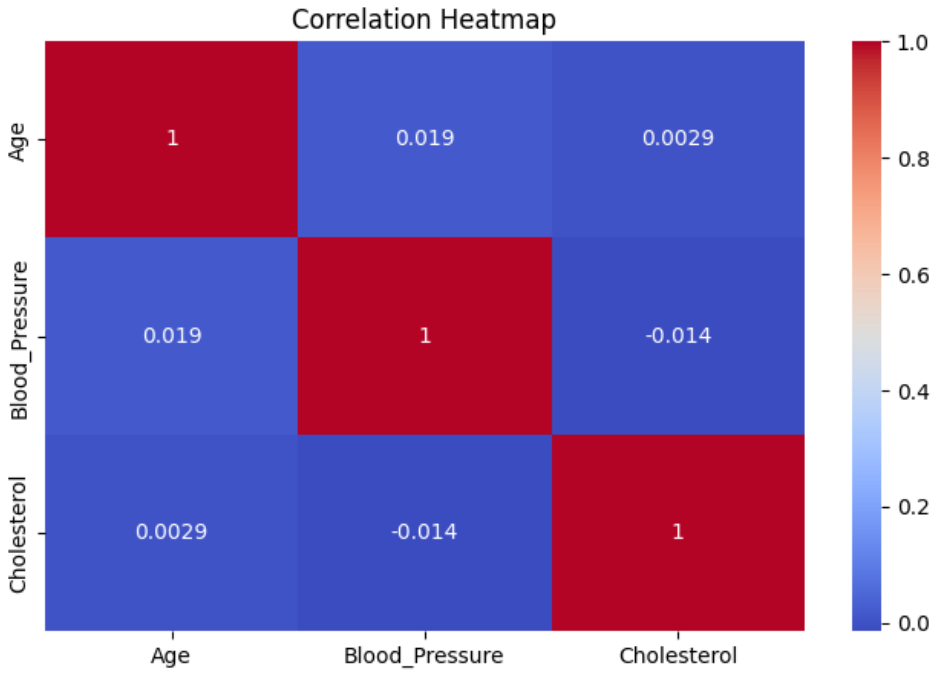






Correlation Heatmap

- Shows relationships between age, blood pressure, and cholesterol.



7. Boolean Encoding (Final Required Format)

Created **two boolean fields**:

- `Gender_Male` → True if gender is male

  -> df["Gender_Male"] = df["Gender"].str.lower().eq("male")

- `Condition_Hypertension` → True if condition is hypertension

  -> df["Condition_Hypertension"] = df["Condition"].str.lower().eq("hypertension")

This keeps encoding simple and readable.

8. data scaling and encoding

```
scaler = StandardScaler()
encoder = OneHotEncoder(drop="first", sparse_output=False)
transformer = ColumnTransformer(
   transformers=[ ("num", scaler, numerical_cols), ("cat", encoder, categorical_cols) ] )
processed_data = transformer.fit_transform(df)
```

9. Final Dataset Creation

Only the following columns are retained:

Column	Type	Description
Age	Numeric	Patient age
Blood_Pressure	Numeric	BP reading
Cholesterol	Numeric	Cholesterol level
Gender_Male	Boolean	True = Male, False = Female
Condition_Hypertension	Boolean	True = Hypertension, False otherwise

Final dataset saved as:

`final_cleaned_dataset.csv`