MSCS-634: Project: Deliverable 1 Report: Data Collection, Cleaning, and Exploration

---

Sandesh Pokharel

University of the Cumberlands

MSCS-634 Advanced Big Data and Data Mining

Dr. Satish Penmatsa

July 13, 2025

1. Introduction

In this deliverable, we focus on loading, cleaning, and exploring a healthcare dataset using Python and its popular data analysis libraries like pandas, seaborn, and matplotlib. The selected dataset is the UCI Heart Disease dataset, chosen for its diverse set of numerical and categorical features, presence of missing values, and suitability for classification.

2. Dataset Overview

The dataset contains 920 rows and 14 columns, each representing clinical features relevant to diagnosing heart disease. These features include age, sex, cholesterol, chest pain type, and more. The target column 'num' indicates the presence (1) or absence (0) of heart disease.

3. Justification for Dataset Selection

- Includes both numerical and categorical features
- Contains missing values to practice imputation techniques
- Balanced distribution of target classes
- Suitable for classification, clustering, and regression tasks
- Publicly available and widely used in research for heart disease prediction

4. Data Cleaning and Preprocessing

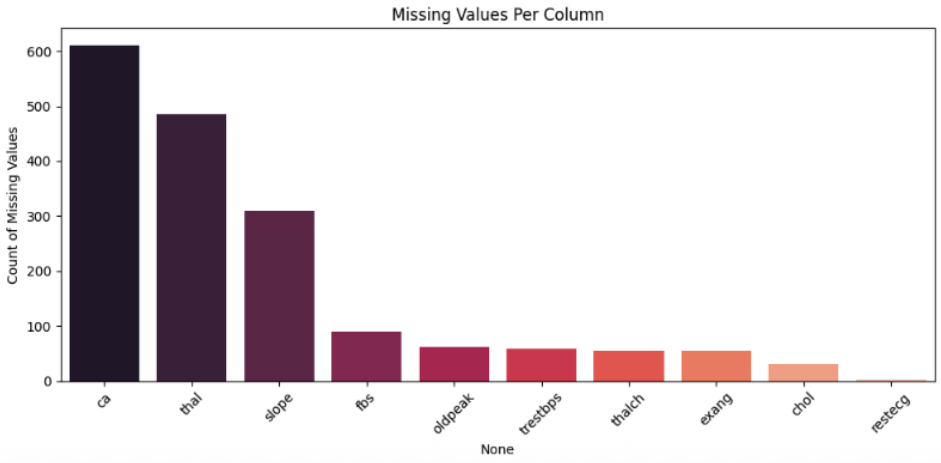
We began by inspecting missing values and data types. The following cleaning steps were taken:

- Converted boolean-like values to integers (e.g., 'True'/'False' to 1/0)
- Dropped columns with too many missing values (e.g., 'ca')
- Imputed categorical columns using mode
- Imputed numerical columns using median
- Converted 'num' to binary (1 = heart disease, 0 = no heart disease)

```
plt.show()
```

```
Missing values per column:

ca          611
thal        486
slope       309
fbs          90
oldpeak      62
trestbps     59
thalch       55
exang        55
chol         30
restecg       2
dtype: int64
```

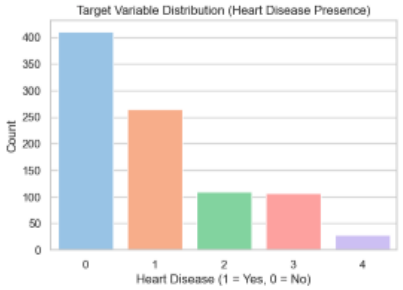
```
/var/folders/bx/n0b8n8wd0ss985_pj3z33n780000gn/T/ipykernel_87430/3955259693.py:14: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=F
alse` for the same effect.

  sns.barplot(x=missing.index, y=missing.values, palette="rocket")
```



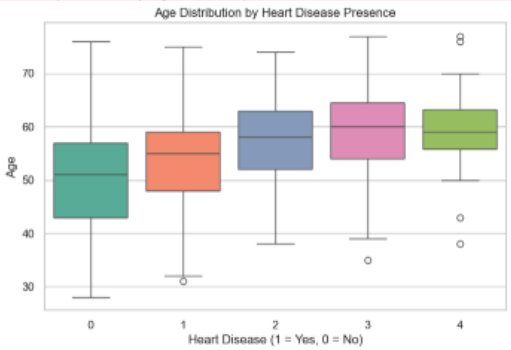

```
[ ]:
```

Microsoft

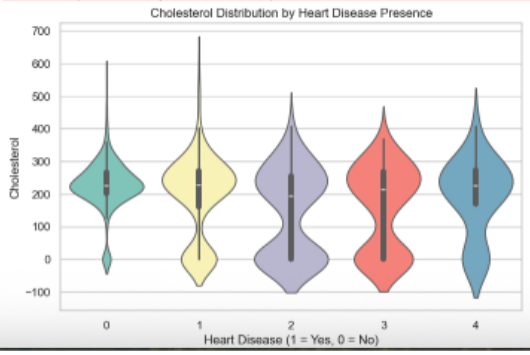
```
/var/folders/bx/m0b8n8wd0ss985_pj3z33n780000gn/T/ipykernel_87430/3235067988.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend
=False` for the same effect.

  sns.countplot(x='num', data=df, palette='pastel')
```




```
/var/folders/bx/m0b8n8wd0ss985_pj3z33n780000gn/T/ipykernel_87430/3235067988.py:11: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend
=False` for the same effect.

  sns.boxplot(x='num', y='age', data=df, palette='Set2')
```




```
/var/folders/bx/m0b8n8wd0ss985_pj3z33n780000gn/T/ipykernel_87430/3235067988.py:19: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend
=False` for the same effect.

  sns.violinplot(x='num', y='chol', data=df, palette='Set3')
```

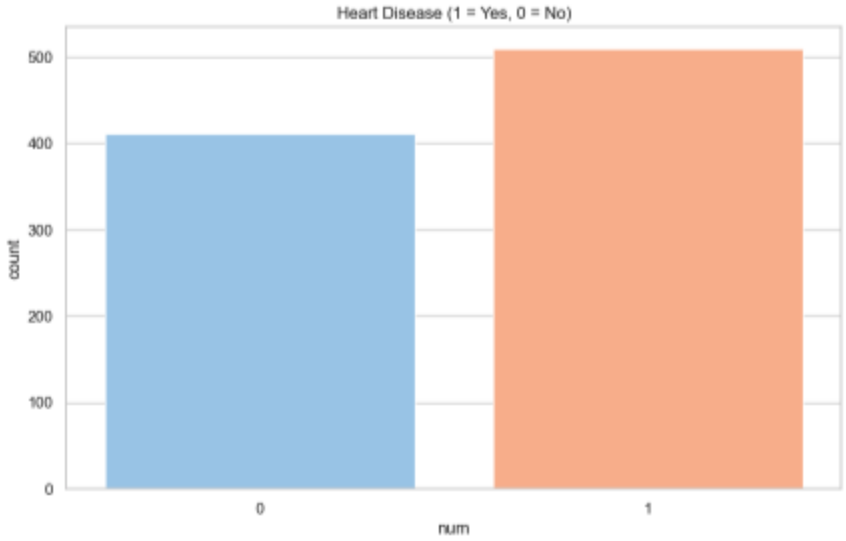


ChatGPT

[17]:
```
# Convert num > 0 to 1 (i.e., any heart disease)
df['num'] = df['num'].apply(lambda x: 1 if x > 0 else 0)
```

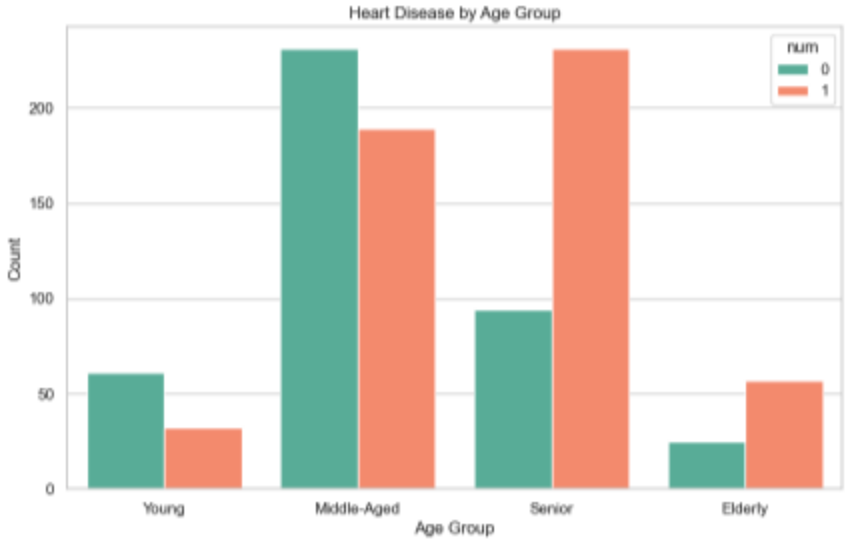
[18]:
```
sns.countplot(x='num', data=df, palette='pastel')
plt.title("Heart Disease (1 = Yes, 0 = No)")
plt.show()
```

```
/var/folders/bx/n0b8n8wd0ss985_pj3z33n780000gn/T/ipykernel_87430/3175312295.py:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend
=False` for the same effect.

  sns.countplot(x='num', data=df, palette='pastel')
```



[ ]:

[19]:
```
# Create age bins
df['age_group'] = pd.cut(df['age'],
                         bins=[0, 40, 55, 65, 100],
                         labels=['Young', 'Middle-Aged', 'Senior', 'Elderly'])

# Check the distribution
sns.countplot(x='age_group', hue='num', data=df, palette='Set2')
plt.title("Heart Disease by Age Group")
plt.xlabel("Age Group")
plt.ylabel("Count")
plt.show()
```



[ ]:

```
[21]: # Drop columns that shouldn't be in correlation (like 'id', 'age_group' which is categorical)
correlation_cols = df.drop(columns=['id', 'age_group']).select_dtypes(include=['int64', 'float64'])

# Compute correlation matrix
correlation_matrix = correlation_cols.corr()

# Plot heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title("Final Correlation Matrix After Preprocessing")
plt.show()
```

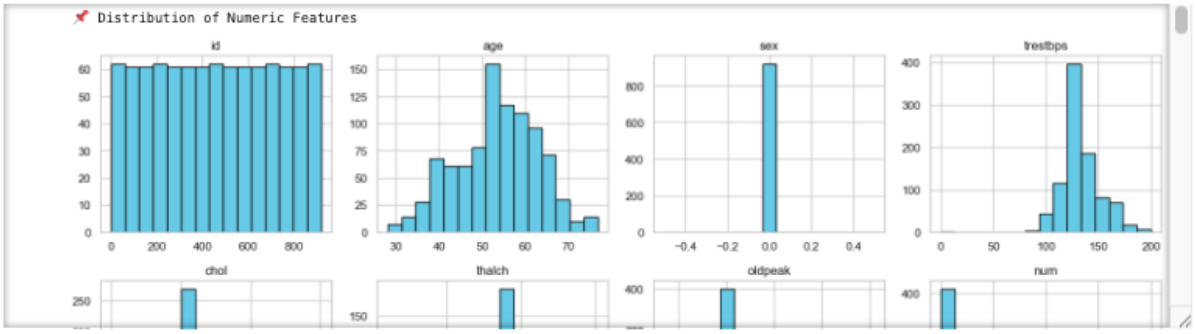

```
[ ]:
```

5. Exploratory Data Analysis (EDA)

Visualizations were used to uncover patterns and relationships in the data. These include:

- Distribution of target variable
- Age group analysis vs heart disease
- Count plots for categorical features
- Histograms and boxplots for numerical features
- Correlation heatmaps

```
[15]: import matplotlib.pyplot as plt
import seaborn as sns

# Set global plot style
sns.set(style="whitegrid")
plt.rcParams['figure.figsize'] = (10, 6)

# 1. Histogram for numerical features
numeric_cols = df.select_dtypes(include=['int64', 'float64']).columns

print("📌 Distribution of Numeric Features")
df[numeric_cols].hist(bins=15, figsize=(15, 12), layout=(4, 4), color='skyblue', edgecolor='black')
plt.tight_layout()
plt.show()

# 2. Boxplots to detect outliers
print("📌 Boxplots for Outlier Detection")
for col in numeric_cols:
    plt.figure()
    sns.boxplot(x=df[col], color='orange')
    plt.title(f'Boxplot: {col}')
    plt.show()

# 3. Correlation Heatmap
print("📌 Correlation Matrix")
plt.figure(figsize=(12, 8))
correlation_matrix = df[numeric_cols].corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title("Correlation Heatmap of Numerical Features")
plt.show()
```




```
[ ]:
```

6. Key Insights from Analysis

- Heart disease is slightly more prevalent in the dataset.
- Senior and elderly groups show higher heart disease rates.
- High cholesterol and lower maximum heart rate (thalach) are linked with heart disease.
- Male patients are more frequently diagnosed with heart disease.

7. Challenges Encountered & Solutions

- Challenge: Mixed data types and missing values across several columns.
  Solution: Used type casting, label encoding, and mode/median imputation depending on the column type.
- Challenge: Column 'ca' had too many missing values.
  Solution: Dropped the column after validating its low utility.
- Challenge: Making EDA comprehensive.
  Solution: Used various seaborn visualizations to explore trends by age, sex, cholesterol, and target class.

[3]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Check for missing values
missing = df.isnull().sum().sort_values(ascending=False)
missing = missing[missing > 0]

# Display missing value count
print("Missing values per column:\n")
print(missing)

# Plot missing values
plt.figure(figsize=(10, 5))
sns.barplot(x=missing.index, y=missing.values, palette="rocket")
plt.title("Missing Values Per Column")
plt.ylabel("Count of Missing Values")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[3], line 2
      1 import matplotlib.pyplot as plt
----> 2 import seaborn as sns
      4 # Check for missing values
      5 missing = df.isnull().sum().sort_values(ascending=False)

ModuleNotFoundError: No module named 'seaborn'
```

[ ]:

]:
```python
# Drop 'ca' column due to too many missing values
df.drop(columns=['ca'], inplace=True)

# Impute categorical columns with mode
categorical_cols = ['thal', 'slope', 'fbs', 'exang', 'restecg']
for col in categorical_cols:
    df[col].fillna(df[col].mode()[0], inplace=True)

# Impute numerical columns with median (corrected 'thalach' spelling)
numerical_cols = ['oldpeak', 'trestbps', 'thalach', 'chol']
for col in numerical_cols:
    df[col].fillna(df[col].median(), inplace=True)

# Confirm all missing values handled
print("Any missing values left?", df.isnull().sum().any())
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[8], line 2
      1 # Drop 'ca' column due to too many missing values
----> 2 df.drop(columns=['ca'], inplace=True)
      4 # Impute categorical columns with mode
      5 categorical_cols = ['thal', 'slope', 'fbs', 'exang', 'restecg']

File ~/Sandesh_Cumberlands_Assignments/Advanced_Big_Data_And_Data_Mining/MSCS-634-Project/venv/lib/python3.13/site-packages/pandas/core/
frame.py:5588, in DataFrame.drop(self, labels, axis, index, columns, level, inplace, errors)
   5440 def drop(
   5441     self,
   5442     labels: IndexLabel | None = None,
   (...)   5449     errors: IgnoreRaise = "raise",
   5450 ) -> DataFrame | None:
   5451     """
   5452     Drop specified labels from rows or columns.
   5453
   (...)   5586             weight  1.0     0.8
   5587     """
-> 5588     return super().drop(
   5589         labels=labels,
   5590         axis=axis,
   5591         index=index,
   5592         columns=columns,
   5593         level=level,
   5594         inplace=inplace,
   5595         errors=errors,
   5596     )

File ~/Sandesh_Cumberlands_Assignments/Advanced_Big_Data_And_Data_Mining/MSCS-634-Project/venv/lib/python3.13/site-packages/pandas/core/
generic.py:4807, in NDFrame.drop(self, labels, axis, index, columns, level, inplace, errors)
   4805 for axis, labels in axes.items():
   4806     if labels is not None:
-> 4807         obj = obj._drop_axis(labels, axis, level=level, errors=errors)
   4809 if inplace:
   4810     self._update_inplace(obj)

File ~/Sandesh_Cumberlands_Assignments/Advanced_Big_Data_And_Data_Mining/MSCS-634-Project/venv/lib/python3.13/site-packages/pandas/core/
generic.py:4849, in NDFrame._drop_axis(self, labels, axis, level, errors, only_slice)
   4847         new_axis = axis.drop(labels, level=level, errors=errors)
   4848     else:
-> 4849         new_axis = axis.drop(labels, errors=errors)
   4850     indexer = axis.get_indexer(new_axis)
   4852 # Case for non-unique axis
   4853 else:

File ~/Sandesh_Cumberlands_Assignments/Advanced_Big_Data_And_Data_Mining/MSCS-634-Project/venv/lib/python3.13/site-packages/pandas/core/
indexes/base.py:7098, in Index.drop(self, labels, errors)
   7096 if mask.any():
   7097     if errors != "ignore":
-> 7098         raise KeyError(f"{labels[mask].tolist()} not found in axis")
   7099     indexer = indexer[~mask]
   7100 return self.delete(indexer)

KeyError: "['ca'] not found in axis"
```

]:

```python
# Drop 'ca' column due to too many missing values
df.drop(columns=['ca'], inplace=True)

# Impute categorical columns with mode
categorical_cols = ['thal', 'slope', 'fbs', 'exang', 'restecg']
for col in categorical_cols:
    df[col].fillna(df[col].mode()[0], inplace=True)

# Impute numerical columns with median
numerical_cols = ['oldpeak', 'trestbps', 'thalch', 'chol']
for col in numerical_cols:
    df[col].fillna(df[col].median(), inplace=True)

# Confirm all missing values handled
print("Any missing values left?", df.isnull().sum().any())
```

```
/var/folders/bx/m0b8n8wd0ss985_pj3z33n780000gn/T/ipykernel_87430/2223433738.py:7: FutureWarning: A value is trying to be set on a copy o
f a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu
es always behaves as a copy.

ollapse Output oing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me
], to perform the operation inplace on the original object.


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

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
File ~/Sandesh_Cumberlands_Assignments/Advanced_Big_Data_And_Data_Mining/MSCS-634-Project/venv/lib/python3.13/site-packages/pandas/core/
indexes/range.py:413, in RangeIndex.get_loc(self, key)
    412 try:
--> 413     return self._range.index(new_key)
    414 except ValueError as err:

ValueError: 0 is not in range

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Cell In[7], line 7
      5 categorical_cols = ['thal', 'slope', 'fbs', 'exang', 'restecg']
      6 for col in categorical_cols:
----> 7     df[col].fillna(df[col].mode()[0], inplace=True)
      9 # Impute numerical columns with median
     10 numerical_cols = ['oldpeak', 'trestbps', 'thalch', 'chol']

File ~/Sandesh_Cumberlands_Assignments/Advanced_Big_Data_And_Data_Mining/MSCS-634-Project/venv/lib/python3.13/site-packages/pandas/core/
series.py:1130, in Series.__getitem__(self, key)
   1127     return self._values[key]
   1129 elif key_is_scalar:
-> 1130     return self._get_value(key)
   1132 # Convert generator to list before going through hashable part
   1133 # (We will iterate through the generator there to check for slices)
   1134 if is_iterator(key):

File ~/Sandesh_Cumberlands_Assignments/Advanced_Big_Data_And_Data_Mining/MSCS-634-Project/venv/lib/python3.13/site-packages/pandas/core/
series.py:1246, in Series._get_value(self, label, takeable)
   1243     return self._values[label]
   1245 # Similar to Index.get_value, but we do not fall back to positional
-> 1246 loc = self.index.get_loc(label)
   1248 if is_integer(loc):
   1249     return self._values[loc]

File ~/Sandesh_Cumberlands_Assignments/Advanced_Big_Data_And_Data_Mining/MSCS-634-Project/venv/lib/python3.13/site-packages/pandas/core/
indexes/range.py:415, in RangeIndex.get_loc(self, key)
    413         return self._range.index(new_key)
    414     except ValueError as err:
--> 415         raise KeyError(key) from err
    416 if isinstance(key, Hashable):
    417     raise KeyError(key)

KeyError: 0
```

8. Conclusion

This deliverable successfully prepared the heart disease dataset by handling missing values, transforming features, and performing thorough exploratory data analysis. The resulting insights provide a solid foundation for building predictive models in