# Bilenkin540_Term Project_Milestone_2

April 16, 2025

## 0.1 Project Milestone 2: Data Transformation and Cleaning

**Dataset:** heart_failure_clinical_records_dataset.csv

**Objective:** Perform 5+ data transformation/cleaning steps and create a clean, human-readable dataset.

[23]:
```
import pandas as pd
import numpy as np

# Loading the flat dataset
df = pd.read_csv(r"C:\Users\maxim\OneDrive\Desktop\BU\DSC 540\Term␣
↪Project\heart_failure_clinical_records_dataset.csv")

# Displaying the first 5 rows to get insight of the dataset
df.head()
```

[23]:
```
    age  anaemia  creatinine_phosphokinase  diabetes  ejection_fraction  \
0  75.0        0                       582         0                 20
1  55.0        0                      7861         0                 38
2  65.0        0                       146         0                 20
3  50.0        1                       111         0                 20
4  65.0        1                       160         1                 20

   high_blood_pressure  platelets  serum_creatinine  serum_sodium  sex  \
0                    1  265000.00               1.9           130    1
1                    0  263358.03               1.1           136    1
2                    0  162000.00               1.3           129    1
3                    0  210000.00               1.9           137    1
4                    0  327000.00               2.7           116    0

   smoking  time  DEATH_EVENT
0        0     4            1
1        0     6            1
2        1     7            1
3        0     7            1
4        0     8            1
```

### 0.1.1 Step #1 – Rename Column Headers

To make the column names more descriptive and readable, I renamed them using `df.rename()`.

[24]:
```python
df.rename(columns={
    'age': 'Age',
    'anaemia': 'Anemia',
    'creatinine_phosphokinase': 'CreatininePhosphokinase',
    'diabetes': 'Diabetes',
    'ejection_fraction': 'EjectionFraction',
    'high_blood_pressure': 'HighBloodPressure',
    'platelets': 'Platelets',
    'serum_creatinine': 'SerumCreatinine',
    'serum_sodium': 'SerumSodium',
    'sex': 'Sex',
    'smoking': 'Smoking',
    'time': 'FollowUpTime',
    'DEATH_EVENT': 'DeathEvent'
}, inplace=True)
```

### 0.1.2 Step #2 – Check and Remove Duplicates

I used `df.duplicated()` to identify and remove any duplicate records.

[25]:
```python
print("Number of duplicate rows before removal:", df.duplicated().sum())
df.drop_duplicates(inplace=True)
print("Number of duplicate rows after removal:", df.duplicated().sum())
```

```
Number of duplicate rows before removal: 0
Number of duplicate rows after removal: 0
```

### 0.1.3 Step #3 – Standardize Inconsistent Values

The `Sex` column was originally coded as 0 (female) and 1 (male). I converted it to string labels for better readability.

[26]:
```python
df['Sex'] = df['Sex'].map({1: 'Male', 0: 'Female'})
```

### 0.1.4 Step #4 – Handle Missing Values

I checked for missing values. Even though none were found, I demonstrated how to fill numeric missing values using the median.

[27]:
```python
print("Missing values in each column:\n", df.isnull().sum())

# Applying median fill just in case (robust to outliers)
num_cols = df.select_dtypes(include=[np.number]).columns
for col in num_cols:
    df[col] = df[col].fillna(df[col].median())
```

```
Missing values in each column:
 Age                         0
Anemia                      0
CreatininePhosphokinase     0
Diabetes                    0
EjectionFraction            0
HighBloodPressure           0
Platelets                   0
SerumCreatinine             0
SerumSodium                 0
Sex                         0
Smoking                     0
FollowUpTime                0
DeathEvent                  0
dtype: int64
```

### 0.1.5 Step #5 – Detect and Remove Outliers in Age

I used the IQR method to detect outliers in the `Age` column and removed any extreme values.

[28]:
```
Q1 = df['Age'].quantile(0.25)
Q3 = df['Age'].quantile(0.75)
IQR = Q3 - Q1

outliers = df[(df['Age'] < (Q1 - 1.5 * IQR)) | (df['Age'] > (Q3 + 1.5 * IQR))]
print("Number of outliers in 'Age':", len(outliers))

# Removing outliers from dataset
df = df[~((df['Age'] < (Q1 - 1.5 * IQR)) | (df['Age'] > (Q3 + 1.5 * IQR)))]
```

```
Number of outliers in 'Age': 0
```

### 0.1.6 Final – Preview Cleaned Dataset

Below is a snapshot of the fully cleaned and transformed dataset.

[29]:
```
# Displaying the first 5 row of the cleaned dataset
df.head()
```

[29]:
```
    Age  Anemia  CreatininePhosphokinase  Diabetes  EjectionFraction  \
0  75.0       0                      582         0                20
1  55.0       0                     7861         0                38
2  65.0       0                      146         0                20
3  50.0       1                      111         0                20
4  65.0       1                      160         1                20

   HighBloodPressure  Platelets  SerumCreatinine  SerumSodium     Sex  \
0                  1  265000.00              1.9          130    Male
1                  0  263358.03              1.1          136    Male
```

```
2                 0  162000.00               1.3          129    Male
3                 0  210000.00               1.9          137    Male
4                 0  327000.00               2.7          116  Female

   Smoking  FollowUpTime  DeathEvent
0        0             4           1
1        0             6           1
2        1             7           1
3        0             7           1
4        0             8           1
```

### 0.1.7 Ethical Implications of Data Wrangling

In this project, I performed five different cleaning steps to transform the data into a clean and readable format. First, I uploaded the dataset and used the head() method to display the first five rows to better understand the structure and contents of the data.

In total, I completed five transformation steps. In the first step, I renamed the column headers using the df.rename() method to make them more descriptive and readable. I capitalized the first letter of each column name and removed underscores, merging multi-word names using camelCase for better visibility and consistency.

Overall, the dataset was quite clean. I found no duplicate or missing values. I corrected inconsistent values, such as standardizing the gender labels by replacing 0 with "Female" and 1 with "Male" to improve readability for the audience. Additionally, I ran code to detect potential outliers, but none were found. All these steps were carried out to ensure data quality and integrity.

There are no specific legal or regulatory guidelines directly attached to this dataset. However, under the Health Insurance Portability and Accountability Act (HIPAA), any medical health information must be protected and shared only with patient consent. In this case, all data has been anonymized, and no personal identities can be revealed. The dataset was acquired from a public Kaggle repository and contains no personally identifiable information (PII), which suggests that the data was sourced ethically and anonymized appropriately.

The main risk of data transformation is the potential loss of valuable information or the introduction of bias, especially if incorrect assumptions are made. Fortunately, I did not have to make any assumptions in this project because there were no missing values or outliers. However, in healthcare data, outlier removal should always be handled carefully, as extreme values can often reveal critical insights. The dataset appears credible based on its structure and origin. No synthetic or unverifiable data was used.

To mitigate ethical risks, I kept all transformation steps transparent and minimal, avoiding the removal of any significant information. I ensured the changes were reversible, in case any original data points need to be restored for future analysis. Any further use of this data should be handled cautiously to avoid biased or unfair outcomes-especially in a sensitive domain like healthcare.