Cairo University

Faculty of Engineering

Systems and Biomedical Department

Second Semester - 2022/2023

Analysis of Heart Attack Prediction using Naive Bayes Classifier

---

<u>Team #05</u>

Malak Nasser

Sara Mohamed

Luna Eyad

Rana Hany

Raghda Tarek Neiazy

Under The Supervision Of: Dr. Ibrahim Mohamed

Abstract

The field of statistics plays a crucial role in extracting meaningful insights from data, enabling us to make informed decisions and predictions in various domains. It serves as a fundamental tool for analyzing complex datasets, particularly in healthcare and medical research. With the increasing availability of data, statistical analysis techniques have become essential for identifying patterns, understanding relationships, and predicting outcomes, leading to improved diagnostics, treatment strategies, and preventive measures.

Data Mining is the process of searching and analyzing a large batch of raw data in order to identify patterns and extract useful information. It is a fascinating field of research whose significant goal is to discover intriguing and useful patterns from huge data sets. Nowadays, health diseases are increasing day by day due to our lifestyle. Especially heart diseases, which are considered one of the leading causes of death worldwide.

1.1. Background

The dataset utilized in this project, sourced from Kaggle, focuses on heart attack analysis and prediction. Heart disease remains a leading cause of mortality globally, and early detection and accurate risk assessment are crucial for effective intervention. This dataset presents an opportunity to investigate the factors associated with heart attacks. By analyzing a variety of health-related features, we aim to identify significant predictors and provide valuable insights for healthcare professionals, researchers, and policymakers. Through this project, we strive to contribute to the existing body of knowledge in the field of cardiovascular health and provide a foundation for further research and development.

1.2. Objectives

The primary objective of this project is to develop a **Naive Bayes classifier model** to predict the occurrence of heart attacks based on a set of relevant features. Specifically, we aim to achieve the following:

- Explore and preprocess the heart attack dataset
- Conduct exploratory data analysis to gain insights into the data
- Handle outliers and standardize the features
- Split the data into training and testing sets
- Calculate descriptive statistics to understand the data distribution
- Implement a Naive Bayes classifier from scratch
- Train the model on the training data and evaluate its performance
- Compare the results with the standard Naive Bayes classifier

2.Methods

This section describes the various steps involved in the heart attack prediction project, including the data features, data preprocessing, exploratory data analysis, outlier detection and handling, data standardization, data partitioning, and implementation of the Naive Bayes classifier. The software packages used for each step are also mentioned.

2.1. Data Preprocessing

Data preprocessing is a critical step in the data analysis pipeline that involves transforming raw data into a clean and structured format suitable for further analysis and modeling. In real-world datasets, it is common to encounter various data quality issues, such as missing values, outliers, inconsistent formatting, or noise. These issues can adversely affect the accuracy, reliability, and performance of subsequent analytical tasks.

By investing time and effort in data preprocessing, we can improve the quality and reliability of their findings, enhance the performance of machine learning models, and enable more informed decision-making. Data preprocessing is a crucial stage in the data analysis workflow, as it sets the stage for subsequent exploratory data analysis, modeling, and interpretation.

2.1.1. Dataset Features

The dataset, composed of both categorical and numerical data, provides a comprehensive snapshot of factors potentially contributing to heart diseases, including demographic details like **age** and **sex**, clinical parameters like **chest pain type (cp)**, **resting blood pressure (trtbps)**, **serum cholesterol (chol), fasting blood sugar (fbs)**, and **resting electrocardiographic results (restecg)**, among others

Feature	Description	Quantitative	Categorical
age	Represents **age** of the patient.	Continuous	———
sex	Represents **gender** of the patient. ● Male ● Female	———	Binary
cp	Represents **type of chest pain** experienced by the patient. ● Typical Angine ● Atypical Angine ● Non-Anginal ● Asymptotic	———	Nominal
trtbps	Represents the **resting blood pressure** of the patient.	Continuous	———

chol	Represents the **chole** **sterol level** of the patient.	Continuous	———
fbs	Represents the **fasting blood sugar level** of the patient. ● Greater than 120 mg/dl = 1 ● Less than 120 mg/dl = 0	———	Binary
restecg	Represents the **results of the resting electrocardiogram (ECG)** of the patient.	———	Nominal
thalachh	Represents the **maximum heart rate** achieved by the patient.	Continuous	———
exng	Represents whether the patient **experienced angina (chest pain) induced by exercise** ● **Yes = 1** ● **No = 0**	———	Binary
oldpeak	Represents **ST depression induced by exercise relative to rest.**	Continuous	———
slp	Represents the **slope of the ST segment during exercise.** ● **Upsloping = 0** ● **Flat = 1** ● **Downsloping = 2**	———	Ordinal
caa	Represents the **number of major blood vessels colored by fluoroscopy.**	Discrete	———
thall	Represents a **type of blood disorder called thalassemia.** ● **Normal = 0** ● **Fixed Defect = 1** ● **Reversible defect = 2** ● **Unknown = 3**	———	Nominal

output	Represents **whether the patient had a heart attack or not.** ● **More chance of heartattack = 1** ● **Less chance of heartattack = 0**	————————	Binary

2.1.2. Checking For Missing Values Or Outliers.

To check for outliers in a dataset, there are various statistical and visualization methods. One common method is to create a boxplot() for each variable in the dataset. A boxplot() displays the distribution of a variable and can help identify outliers.

2.2. Exploratory Data Analysis And Data Cleaning

2.2.1. Handling Outliers

Using the interquartile range (IQR) method: the IQR measures statistical dispersion, representing the spread of the middle 50% of values in the dataset. It is calculated as the difference between the data's 75th percentile (Q3) and the 25th percentile (Q1). The IQR is used to identify outliers in a dataset by defining the upper and lower bounds for outliers based on the IQR. Any values that are *less than Q1 - 1.5 * IQR* or *greater than Q3 + 1.5 * IQR* are considered outliers and are replaced with null values. Iterating over each numerical feature and calculating the 75th and 25th percentiles for that feature using the np.percentile() function

How many outliers were identified and replaced with null values can be known by counting the number of null values before and after outlier removal. This can be useful for understanding the impact of the outlier removal process on the data and verifying that the code is working as expected.

2.2.2. Analyzing Descriptive Statistics

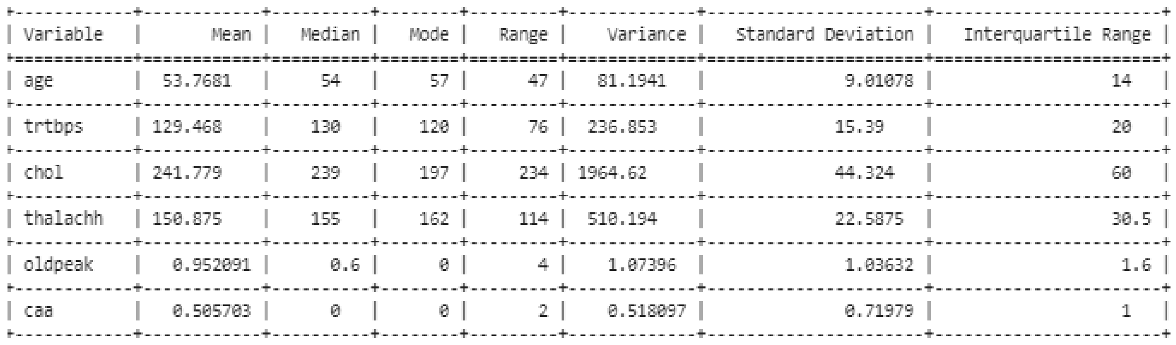
Descriptive statistics are essential in quantitative analysis because they provide a full description of the data, allowing us to obtain useful insights and make informed judgments. When working with a dataset that comprises both categorical and numerical elements, descriptive statistics allow us to quantitatively characterize and comprehend the data's qualities in a variety of ways.

We may objectively summarize and analyze the dataset's features, find patterns, and detect outliers or unexpected observations by generating and analyzing descriptive statistics. These statistics form the basis for additional analysis, hypothesis testing, and modeling in a variety of fields, including research, business, and healthcare.

2.2.2.1. Numerical Variables

When working with numerical data, descriptive statistics provide measurements that capture important elements of the distribution of the dataset. Measures of central tendency, such as the mean, median, and mode, for example, enable us to identify the typical or core value around which the data tends to cluster. These metrics assist us in understanding the dataset's average or most common value.

Furthermore, variability measurements like range, variance, and standard deviation provide insight into the spread or dispersion of the data. They show how far apart the individual data points are from the mean. These measurements are critical for assessing data variability and interpreting the form of the distribution, particularly whether it is skewed or symmetric.



```
+------------+-----------+----------+--------+---------+------------+----------------------+----------------------+
| Variable   |      Mean |   Median |   Mode |   Range |   Variance |   Standard Deviation |  Interquartile Range |
+============+===========+==========+========+=========+============+======================+======================+
| age        |  53.7681  |     54   |     57 |      47 |  81.1941   |              9.01078 |                 14   |
+------------+-----------+----------+--------+---------+------------+----------------------+----------------------+
| trtbps     | 129.468   |    130   |    120 |      76 | 236.853    |             15.39    |                 20   |
+------------+-----------+----------+--------+---------+------------+----------------------+----------------------+
| chol       | 241.779   |    239   |    197 |     234 | 1964.62    |             44.324   |                 60   |
+------------+-----------+----------+--------+---------+------------+----------------------+----------------------+
| thalachh   | 150.875   |    155   |    162 |     114 | 510.194    |             22.5875  |                 30.5 |
+------------+-----------+----------+--------+---------+------------+----------------------+----------------------+
| oldpeak    |  0.952091 |      0.6 |      0 |       4 |   1.07396  |              1.03632 |                  1.6 |
+------------+-----------+----------+--------+---------+------------+----------------------+----------------------+
| caa        |  0.505703 |      0   |      0 |       2 |   0.518097 |              0.71979 |                  1   |
+------------+-----------+----------+--------+---------+------------+----------------------+----------------------+
```

❖ **Measures of Central Tendency:**

➢ **Mean:** Represents the average value of the data.

➢ **Median:** Represents the middle value of the data when it is sorted.

➢ **Mode:** Represents the most frequently occurring value(s) in the data.

❖ **Measures of Dispersion:**

➢ **Range:** Represents the difference between the maximum and minimum values in the data.

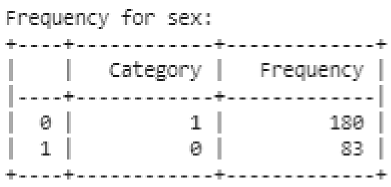
➢ **Variance:** Represents the average of squared deviations from the mean, indicating the spread of the data.

➢ **Standard Deviation:** Represents the square root of the variance, providing a measure of how spread out the data is.

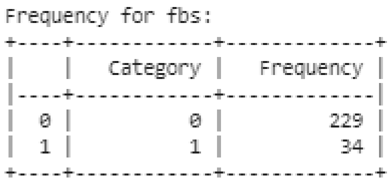
➢ **Interquartile Range:** Represents the difference between the third quartile (Q3) and the first quartile (Q1), capturing the spread of the middle 50% of the data.

2.2.2.2. Categorical Variables

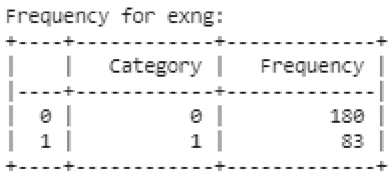
On the other hand, when working with categorical data, descriptive statistics focus on summarizing the different categories present in the dataset. Frequency counts allow us to determine the number of occurrences of each category, providing insights into the dataset's composition and the prevalence of specific categories. Proportions and percentages further aid in understanding the relative frequencies of categories and enable comparisons between different groups.



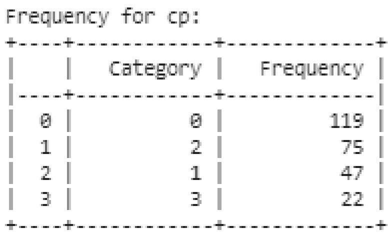
```
Frequency for sex:
+----+------------+-------------+
|    |   Category |   Frequency |
|----+------------+-------------|
|  0 |          1 |         180 |
|  1 |          0 |          83 |
+----+------------+-------------+
```



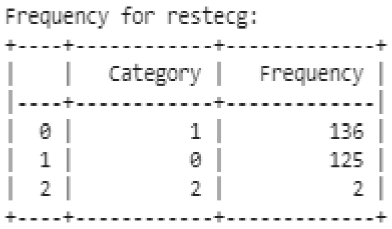
```
Frequency for fbs:
+----+------------+-------------+
|    |   Category |   Frequency |
|----+------------+-------------|
|  0 |          0 |         229 |
|  1 |          1 |          34 |
+----+------------+-------------+
```



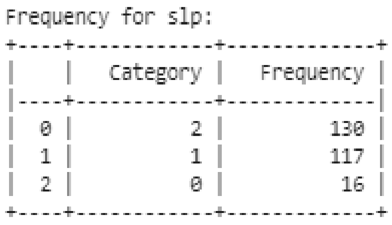
```
Frequency for exng:
+----+------------+-------------+
|    |   Category |   Frequency |
|----+------------+-------------|
|  0 |          0 |         180 |
|  1 |          1 |          83 |
+----+------------+-------------+
```



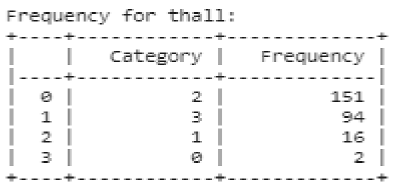
```
Frequency for cp:
+----+------------+-------------+
|    |   Category |   Frequency |
|----+------------+-------------|
|  0 |          0 |         119 |
|  1 |          2 |          75 |
|  2 |          1 |          47 |
|  3 |          3 |          22 |
+----+------------+-------------+
```



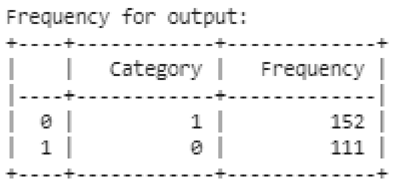
```
Frequency for restecg:
+----+------------+-------------+
|    |   Category |   Frequency |
|----+------------+-------------|
|  0 |          1 |         136 |
|  1 |          0 |         125 |
|  2 |          2 |           2 |
+----+------------+-------------+
```



```
Frequency for slp:
+----+------------+-------------+
|    |   Category |   Frequency |
|----+------------+-------------|
|  0 |          2 |         130 |
|  1 |          1 |         117 |
|  2 |          0 |          16 |
+----+------------+-------------+
```



```
Frequency for thall:
+----+------------+-------------+
|    |   Category |   Frequency |
|----+------------+-------------|
|  0 |          2 |         151 |
|  1 |          3 |          94 |
|  2 |          1 |          16 |
|  3 |          0 |           2 |
+----+------------+-------------+
```



```
Frequency for output:
+----+------------+-------------+
|    |   Category |   Frequency |
|----+------------+-------------|
|  0 |          1 |         152 |
|  1 |          0 |         111 |
+----+------------+-------------+
```

Visual representations, such as bar charts or pie charts, are often used to present the distribution of categorical data, allowing for a more intuitive understanding of the data composition and patterns.

2.2.3. Data Standardization

Standardization refers to the process of transforming data to have a mean of 0 and a standard deviation of 1, aka the z-score. It's calculated by the formula $\mathbf{z = (x - \mu) / \sigma}$, where $\mathbf{x}$ is the datapoint, $\boldsymbol{\mu}$ is the mean of the group of values, and $\boldsymbol{\sigma}$ is the standard deviation of the group of values.

2.2.4. Data Partitioning

Splitting the data into training and testing sets is a crucial step in machine learning for model evaluation and performance estimation. Here's an explanation of why this step is important:

A. **Model Training:** The machine learning model is trained using the training set. The model learns the underlying patterns and correlations between the features and the target variable by being exposed to a labeled dataset. To create appropriate predictions or classifications, the model modifies its parameters based on the training data.

B. **Model Evaluation:** The testing set, also known as the validation set, is used to assess the trained model's performance. This dataset contains previously unseen data that the model did not encounter during training. You can evaluate how effectively the model generalizes to fresh, unseen data by evaluating it on this independent dataset.

C. **Assessing Performance:** By splitting the data into training and testing sets, you may examine the model's performance in a realistic and unbiased manner. By testing the model on previously unseen data, you may determine its capacity to generate appropriate predictions or classifications for fresh occurrences.

D. **Detecting Overfitting:** Overfitting is detected when a model performs well on training data but fails to generalize to fresh data. By evaluating the hypothesis on the testing set, you can determine whether the hypothesis is overfitting. If the model performs much worse on the testing set than on the training set, it indicates that the model did not learn the fundamental patterns but rather memorized the training data.

2.2.5. Visualization And Statistical Test

for each variable we did these steps:

1- Plot the histogram
2- comment on the type of the distribution
3- Statistically test if a feature/column is normally distributed.
4-Plot the conditional distributions of each feature on each target class (label).
In the heart attack Kaggle dataset, the target variable is the "output" variable, which indicates whether a person is at risk of a heart attack or not

Why to test the normality of data?

Various statistical methods used for data analysis make assumptions about normality, including correlation, regression, *t*-tests, and analysis of variance.
When data follows a normal distribution, the mean value is commonly used as a representative value. This mean value can be compared between or among groups to calculate the significance level (p-value) and make statistical inferences. However, if the data is not normally distributed, using the mean as a representative value may not accurately reflect the data. In such cases, selecting the wrong representative value and calculating the significance level based on this value can lead to incorrect interpretations.

To address this issue, it is necessary to test the normality of the data before deciding whether the mean is applicable as the representative value. If the data is normally distributed, parametric tests that rely on the mean can be employed. However, if the data is not normally distributed, nonparametric methods that utilize medians are used to compare groups.
For both of the above tests, null hypothesis states that data are taken from a normal distributed population. When $P > 0.05$, null hypotheses are accepted and data are called as normally distributed.
There are two main methods of assessing normality: **Graphical and numerical** (including statistical tests)
most popular methods are **Shapiro–Wilk** test**, Kolmogorov–Smirnov** test **:**
For both of the above tests, null hypothesis states that data are taken from a normal distributed population. When $P > 0.05$, null hypotheses are accepted and data are called as normally distributed.
Q-Q plot:
In statistics, a Q–Q plot is a scatterplot created by plotting two sets of quantiles (observed and expected) against one another. For normally distributed data, observed data are approximate to the expected data, that is, they are statistically equal.

First, we will discuss the **'age'** attribute as example of the **Numerical Variables**:

The code performs exploratory data analysis (EDA) to understand the distribution of the 'age' feature and its relationship with the 'output' label in the heart attack dataset.

We Created a histogram using the plt.hist function based on the 'age' column of the train_data DataFrame.after displaying the plot we can see that the distribution is approximately normal but we need to test it statistically to be sure.
We applied the **Kolmogorov–Smirnov** and **Q-Q plot** methods explained above and the result was the 'age' feature is normally distributed (fail to reject H0)
The last step is to create box plots to visualize the conditional distributions of the 'age' feature based on the 'output' label in the train_data DataFrame.

Second , we will discuss the **‘sex’** attribute as example of the **Categorical Variables**:
We used a bar plot to visualize the counts, where the x-axis represents the unique values of 'sex' and the y-axis represents the corresponding counts.
Note :In statistical assumptions, the concept of normality does not cover categorical variables.
The last step is to create a count plot that displays the counts of each category of 'sex', differentiated by the 'output' values.

2.3. Model Development and Evaluation

2.3.1. Splitting the Dataset

The dataset was randomly split into training and testing sets in an 80%-20% ratio. The training set was used to train the Naive Bayes classifier, while the testing set was used to evaluate its performance. This partitioning ensured that the model was tested on unseen data, providing a realistic assessment of its predictive capabilities.

2.3.2. Naïve Bayes (NB) Classifier Implementation From Scratch

The Naive Bayes Classifier is based on Bayes' theorem, which is a fundamental concept in probability theory. It makes use of conditional probability to classify data into different categories or classes. The "naive" assumption in Naive Bayes is that all features are conditionally independent given the class, which means that the presence or absence of a particular feature does not affect the presence or absence of any other feature. The purpose of the Naive Bayes classifier is to classify new data instances based on their feature values. It calculates the posterior probability for each class and assigns the data instance to the class with the highest probability. The classifier predicts the class label that maximizes the posterior probability.

2.3.2.1. The Math Behind Naïve Bayes (NB) Classifier

Mathematically, let's denote the class variable as Y and the features as X1, X2, ..., Xn. The Naive Bayes classifier calculates the probability of a data instance belonging to a particular class given its feature values. This is represented as:

$$P(Y \mid X1,\, X2,\, \ldots,\, Xn) \;=\; (P(X1,\, X2,\, \ldots,\, Xn \mid Y) \;*\; P(Y)) \,/\, P(X1,\, X2,\, \ldots,\, Xn)$$

According to Bayes' theorem, the posterior probability of the class given the features is proportional to the likelihood of the features given the class multiplied by the prior probability of the class, divided by the evidence, which is the probability of the features.

2.3.3. Model Training and Prediction

A Naive Bayes classifier was developed from scratch using Python.

A. Model Training

The Naive Bayes classifier was trained using the training data. The probability distributions for each feature were estimated based on the target classes.

a. Check Dependencies Among The Features

First, generating a correlation matrix for the selected columns of the HEART dataset using Pearson’s correlation coefficients. It then visualizes the correlation matrix as a heatmap, where the colors represent the strength and direction of the correlations between the variables. The heatmap provides a visual representation of the relationships between different variables in the dataset, helping to identify patterns and dependencies among the features.

b. Replotting The Distribution of The Numerical Features To Check If They Fit a Known Distribution

Generating histograms with KDE curves for the "age", "chol", "trtbps", "thalachh", "caa" and "oldpeak" columns (numerical columns).

c. Calculating The Prior Probabilities of Each Class in The Specified Column of The Data Frame

It does so by counting the occurrences of each class and dividing it by the total number of rows in the Data Frame.

d. Calculating The Likelihood Probability of a Given Feature Value Occurring, Given a Specific Class

- **Numerical features**

  Assuming a Gaussian (normal) distribution, calculating the likelihood probability 'p(x|y)' using a Gaussian (normal) distribution. Computing the probability density function (PDF) of the Gaussian distribution for the given feature value by using the mean and standard deviation obtained from the filtered DataFrame.

- **Categorical features**

  Filtering the DataFrame 'df' to select only the rows where the column 'Y' (output) is equal to the specified 'label'. Counting the occurrences of the specified feature value for the given output. Then,counting the total number of occurrences of the given output. By division,the likelihood probability 'p(x|y)' for the given categorical feature value is calculated.

e. Implementing The Naïve Bayes Classifier

Calculating the posterior probabilities (numerator only) for each class label and data sample by multiplying the likelihood probabilities and the prior probabilities. Then, predict the class label with the highest posterior probability for each sample.

B. Model Testing

The trained classifier was used to predict the target labels for the testing data. The accuracy of the model was evaluated by comparing the predicted labels with the true labels.

a. Computing the classifier's performance and measuring the classifier's accuracy

Using the confusion matrix (by showing the counts of true positives, true negatives, false positives, and false negatives) and f1 score, respectively. They are applied to the true labels (Y_test) and the predicted labels (Y_pred) to evaluate the classifier's performance on the test set.

2.3.4. Evaluation Comparison Between Pre-build Classifier And Our Model

The code compares the accuracy of our implemented Naive Bayes classifier with the accuracy of the MixedNB classifier .

2.4. Software Packages

The project utilized the following software packages for data preprocessing, analysis, and modeling:

- **Matplotlib.pyplot:** Used for creating visualizations and plots.
- **Seaborn:** Simplifies the creation of statistical graphics and provides visually appealing plots.
- **Pandas:** Used for data manipulation and handling missing values.
- **Numpy:** Provides efficient numerical operations and array manipulation.
- **Tabulate:** Formats tabular data for display.
- **sklearn.model_selection.train_test_split:** Splits datasets into training and testing subsets for model evaluation.
- **Sklearn.processing**The LabelEncoder class from the sklearn.preprocessing module is used for encoding categorical features into numeric values.
- **Scipy.stats:** Provides statistical functions for data analysis and hypothesis testing.
- **Mixed_naive_bayes**This module implements categorical (multinoulli) and Gaussian naive Bayes algorithms (hence *mixed naive Bayes*).

3. Results and Discussion

3.1 Data Preprocessing Results:

In the data preprocessing step, missing values were handled using appropriate imputation techniques. Categorical variables were imputed with the mode, while numerical variables were imputed with the mean. This ensured that the dataset was complete and suitable for further analysis.
The removal of outliers using the IQR method by replacing outliers with null values and subsequently dropping rows with null values, this process aims to eliminate potential outliers from the dataset.

3.2 Exploratory Data Analysis Results:

Exploratory Data Analysis (EDA) was performed to gain insights into the dataset and understand the distribution of each feature. Histograms were plotted to visualize the distribution of variables. Based on visual inspection, it was observed that some variables exhibited Gaussian distributions("age", "trtbps", "chol", "thalachh"), while others had skewed("oldpeak")

3.3 Outlier Detection and Handling Results:

By removing outliers, the distributions of variables became more representative of the underlying population. This allowed for a more robust analysis of the dataset, as extreme values that could bias the results were effectively mitigated.

3.4 Data Standardization Results:

Standardizing the features using the Z-score method ensured that each feature had a mean of 0 and a standard deviation of 1. This process transformed the features to a common scale, enabling fair comparison and eliminating potential bias caused by differences in the magnitude of the variables.
The standardized features facilitated the accurate calculation of probabilities in the Naive Bayes classifier, as it assumed that each feature was normally distributed with a mean of 0 and a standard deviation of 1. This step enhanced the performance of the classifier by ensuring the validity of the underlying assumptions.

3.5 Data Partitioning Results:

The dataset was randomly split into training and testing sets in an 80%-20% ratio. The training set was used to train the Naive Bayes classifier, while the testing set was used to evaluate its performance. This partitioning ensured that the model was tested on unseen data, providing a realistic assessment of its predictive capabilities.
The random partitioning of the data helped reduce the risk of bias and overfitting. By separating the data into independent subsets, the model's ability to generalize to new and unseen data was assessed more accurately.

3.6 Naive Bayes Classifier Results:

The Naive Bayes classifier was implemented from scratch based on the Naive Bayes algorithm. The model was trained using the training data, where the probability distributions for each feature were estimated based on the target classes. The trained model was then used to predict the target labels for the testing data.
The accuracy of the model was evaluated by comparing the predicted labels with the true labels.
The output of confusion matrix function is [[20 4] [4 25]]:

- Each row represents the actual class, and each column represents the predicted class.
- The value 20 in the top-left corner (True Negative) represents The number of samples that were correctly predicted as negative (no heart attack) by the model.
- The value 4 in the top-right corner (False Positive) represents The number of samples that were incorrectly predicted as positive (heart attack) by the model when they were actually negative.
- The value 4 in the bottom-left corner (False Negative) represents The number of samples that were incorrectly predicted as negative (no heart attack) by the model when they were actually positive.
- The value 25 in the bottom-right(True Positive) corner represents The number of samples that were correctly predicted as positive (heart attack) by the model.

The output of f1 score function is 0.8620689655172413

- The F1 score is a measure of the classifier's accuracy, particularly in binary classification tasks where there is an imbalance between the classes.
- It combines both precision and recall into a single metric. Precision measures the classifier's ability to correctly identify positive instances, while recall measures the classifier's ability to correctly identify all positive instances.
- The F1 score is the harmonic mean of precision and recall, providing a balanced measure of the classifier's performance.
- In the output 0.8620689655172413, the F1 score is approximately 0.862, indicating that the classifier has achieved a relatively high level of accuracy in predicting the positive class.

Comparing our results to the NB classifier from standard Python packages:

- **Accuracy (Our Implementation): 0.8620689655172413**
- **Accuracy (Standard Naive Bayes): 0.8867924528301887**

Finally, it’s clear that our NB classifier implementation is accurate.

4. Conclusion

In this paper, Naive Bayes classification has been discussed and implemented from scratch to be used for heart attack prediction . The performance of the model was evaluated based on its testing accuracy around **86%** .Further improvements as exploring other models and making additional analysis could provide further insights into the factors influencing heart disease presence.