



**King Saud University**  
**College of Computer and Information Sciences**  
**Department of Information Technology**

**IT326: Data Mining**  
1<sup>st</sup> Semester 1445 H

# **Cardiovascular Disease**

## **Report**

Section #	NAME	ID
56534	Aljazzi Alhassan	444201096
56534	Ghala Musallam	444200807
56534	Reuof Alanazi	444200528
56534	Modi Albassam	444200496
56534	Maha Albakr	444201108

**Supervised By:** Ms. Hanan Al Tamimi

# 1.Problem Introduction

- **Problem Statement:**

Cardiovascular diseases are a major cause of death worldwide. With more people at risk due to unhealthy lifestyles and genetics, it's important to find ways to identify those at high risk. This project focuses on using data mining techniques to analyze a dataset related to cardiovascular diseases. The goal is to classify individuals based on their risk factors, helping to predict who may develop CVDs.

- **Significance:**

This project is significant because it addresses the growing issue of cardiovascular diseases. By analyzing data, we can better understand the risk factors associated with CVDs and create models that help healthcare providers identify patients at risk. Early detection can lead to timely interventions, improving health outcomes and potentially lowering healthcare costs. The insights gained from this project could also support public health efforts to reduce CVD risk in communities.

## 2.Data Mining Task

- **Classification Task:**

**Task Definition:** Classification involves predicting the categorical class labels for given input data based on historical data.

**Class Attribute:** In this dataset, the class attribute is `cardio`, which indicates whether the individual has cardiovascular disease (1) or not (0).

### Goals of Classification:

- To build a predictive model that can accurately classify individuals as having cardiovascular disease or not based on their health metrics and lifestyle choices.
- To identify the significant factors contributing to cardiovascular disease, which can help in preventive healthcare measures.

- **Clustering Task:**

Task Definition: Clustering involves grouping a set of objects in such a way that objects in the same group (or cluster) are more similar to each other than to those in other groups.

### Goals of Clustering:

- To identify natural groupings of individuals based on their health metrics and lifestyle factors, which may reveal patterns or segments within the population.
- To explore underlying structures in the data that could inform healthcare strategies, resource allocation, and targeted interventions for different groups.

## 3.Data Description

- **Dataset Overview:**

Source: <https://www.kaggle.com/datasets/akshatshaw7/cardiovascular-disease-dataset>

Number of Objects: 70000

Number of Attributes: 14

- **Attribute Details:**

Data types: Integer, Decimal data

Missing Values: No missing values

- **Statistical Analysis:**

## Five number summary:

```
[ ] #Statistical summaries
summary = df.describe()
variance = df.var()
print("Statistical Summary:\n", summary)
print("\nVariance:\n", variance)
```

Statistical Summary:

	Unnamed: 0	id	age	gender	height
count	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000
mean	34999.500000	49972.419900	19468.865814	0.349571	164.359229
std	20207.403759	28851.302323	2467.251667	0.476838	8.210126
min	0.000000	0.000000	10798.000000	0.000000	55.000000
25%	17499.750000	25006.750000	17664.000000	0.000000	159.000000
50%	34999.500000	50001.500000	19703.000000	0.000000	165.000000
75%	52499.250000	74889.250000	21327.000000	1.000000	170.000000
max	69999.000000	99999.000000	23713.000000	1.000000	250.000000

	weight	ap_hi	ap_lo	cholesterol	gluc
count	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000
mean	74.205690	128.817286	96.630414	0.366871	0.226457
std	14.395757	154.011419	188.472530	0.680250	0.572270
min	10.000000	-150.000000	-70.000000	0.000000	0.000000
25%	65.000000	120.000000	80.000000	0.000000	0.000000
50%	72.000000	120.000000	80.000000	0.000000	0.000000
75%	82.000000	140.000000	90.000000	1.000000	0.000000
max	200.000000	16020.000000	11000.000000	2.000000	2.000000

	smoke	alco	active	cardio
count	70000.000000	70000.000000	70000.000000	70000.000000
mean	0.088129	0.053771	0.803729	0.499700
std	0.283484	0.225568	0.397179	0.500003
min	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	1.000000	0.000000
50%	0.000000	0.000000	1.000000	0.000000
75%	0.000000	0.000000	1.000000	1.000000
max	1.000000	1.000000	1.000000	1.000000

Variance:

	Unnamed: 0
id	8.323976e+08
age	6.087331e+06
gender	2.273745e-01
height	6.740617e+01
weight	2.072378e+02
ap_hi	2.371952e+04
ap_lo	3.552189e+04
cholesterol	4.627405e-01
gluc	3.274933e-01
smoke	8.036307e-02
alco	5.088079e-02
active	1.577512e-01
cardio	2.500035e-01

dtype: float64

-This section displays the key statistical measures for each column in the dataset, including the count, mean, standard deviation (std), minimum (min), 25th percentile (25%), 50th percentile (50%), 75th percentile (75%), and maximum (max) values.

-For example, the 'age' column has a mean of 19468.865814, a standard deviation of 2467.256667, and a range from 10798.0 to 23713.0.

-The 'gender' column has a range from 0.0 to 1.0, representing binary gender values.

-The 'height' column has a mean of 164.359229 and a standard deviation of 8.210126.

## **Impact on Data Preprocessing Decisions:**

- **identifying Data Distribution:**

- The statistical summary helps identify the distribution of values in each column. For instance, if the mean and median have a significant difference, it indicates skewness, suggesting that transformations (e.g., log transformations) may be necessary.

- **Detecting Outliers:**

- Large differences between the minimum/maximum values and the quartiles can indicate potential outliers. This information prompts further investigation into these extreme values, which may need to be removed or treated to improve data quality.

- **Assessing Feature Variability:**

- Columns with very low variance may not contribute meaningful information to models, leading to decisions on whether to exclude these features during modeling. Conversely, highly variable features may warrant further examination to understand underlying patterns.

- **Handling Missing Values:**

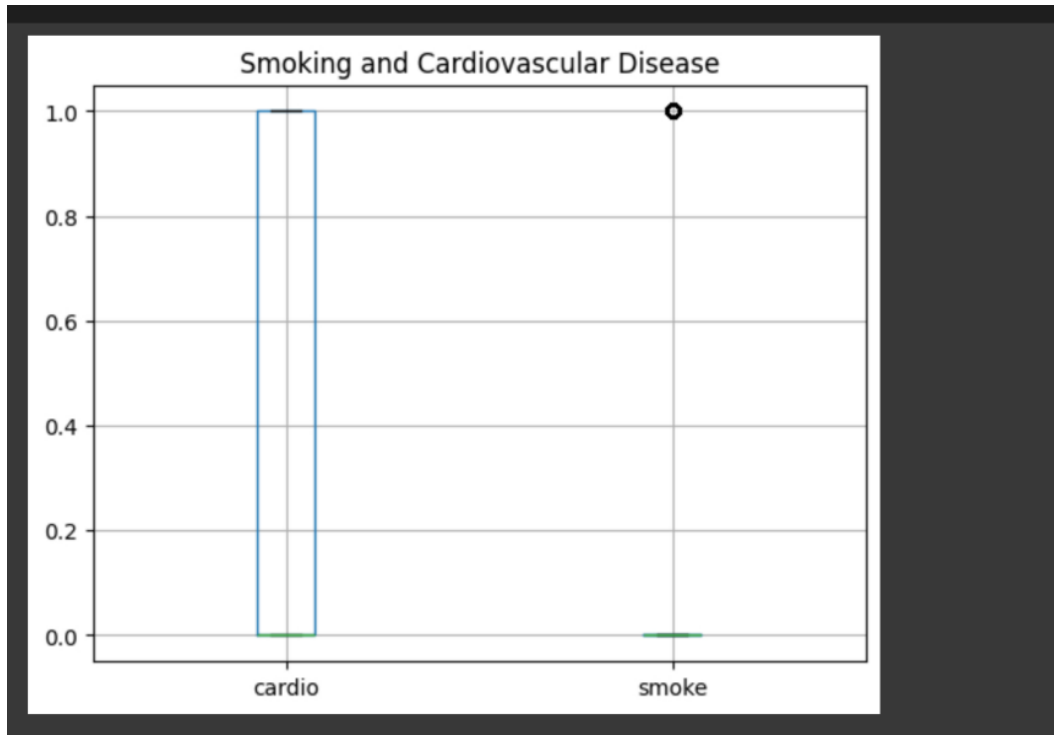
- The count of non-null observations helps identify any missing data issues. If any columns have significantly fewer counts than others, this could necessitate data imputation or removal of those features.

- **Understanding Relationships Between Variables:**

- The summary provides a foundation for exploring relationships between different features. For example, high standard deviation in certain health metrics (like cholesterol levels) may indicate a need for stratification or segmentation in analyses.

## Boxplots:

```
#BoxPlot
df[['cardio', 'smoke']].boxplot()
plt.title('Smoking and Cardiovascular Disease')
plt.show()
```



The boxplot indicates a balanced distribution of cardiovascular disease but shows that smoking is less common, with fewer individuals marked as smokers. With the Correlation Coefficient

## Impact on Data Preprocessing Decisions:

- **Assessing Feature Relevance:**

- If the box plot shows a notable difference in cardiovascular disease status between smokers and non-smokers, it suggests that smoking is a relevant feature for predictive modeling. Conversely, if there is little difference, it may prompt further investigation.

- **Outlier Treatment:**

- The presence of outliers in the box plot indicates individuals with unusual cardiovascular disease statuses. These outliers may need to be addressed during preprocessing, either by removing them or applying techniques to minimize their impact on the model.

- **Potential Data Transformation:**

- If the distributions appear skewed or have a heavy tail, it might suggest the need for transformations (e.g., normalization or scaling) to ensure that the data meets the assumptions of the modeling technique being used.

## Outliers:

```
[ ] #Detect and remove outliers (remove noise)

outlier_threshold = 1.5

def detect_outliers(column_data):
    # Calculate the first and third quartile
    q1 = np.percentile(column_data, 25)
    q3 = np.percentile(column_data, 75)

    iqr = q3 - q1

    upper_bound = q3 + outlier_threshold * iqr
    lower_bound = q1 - outlier_threshold * iqr

    outliers = (column_data > upper_bound) | (column_data < lower_bound)

    return outliers

numeric_columns = df.select_dtypes(include=[np.number]).columns

cleaned_data = df.copy()

for column in numeric_columns:
    outliers = detect_outliers(cleaned_data[column])

    cleaned_data = cleaned_data[~outliers]

print(f"Original data had {len(df)} rows")
print(f"Cleaned data has {len(cleaned_data)} rows after removing outliers")
```

Original data had 70000 rows.

Cleaned data has 31449 rows after removing outliers.

## Impact on Data Preprocessing Decisions:

- **Identifying Data Quality Issues:**

- The presence of outliers can indicate data quality issues or unusual observations that could skew the results of analyses or models. The code's functionality addresses this concern by systematically removing these outliers.

- **Enhancing Model Performance:**

- By cleaning the dataset of outliers, the model is likely to perform better as it reduces noise and potential bias introduced by extreme values. Cleaner data allows for more accurate predictions and insights.



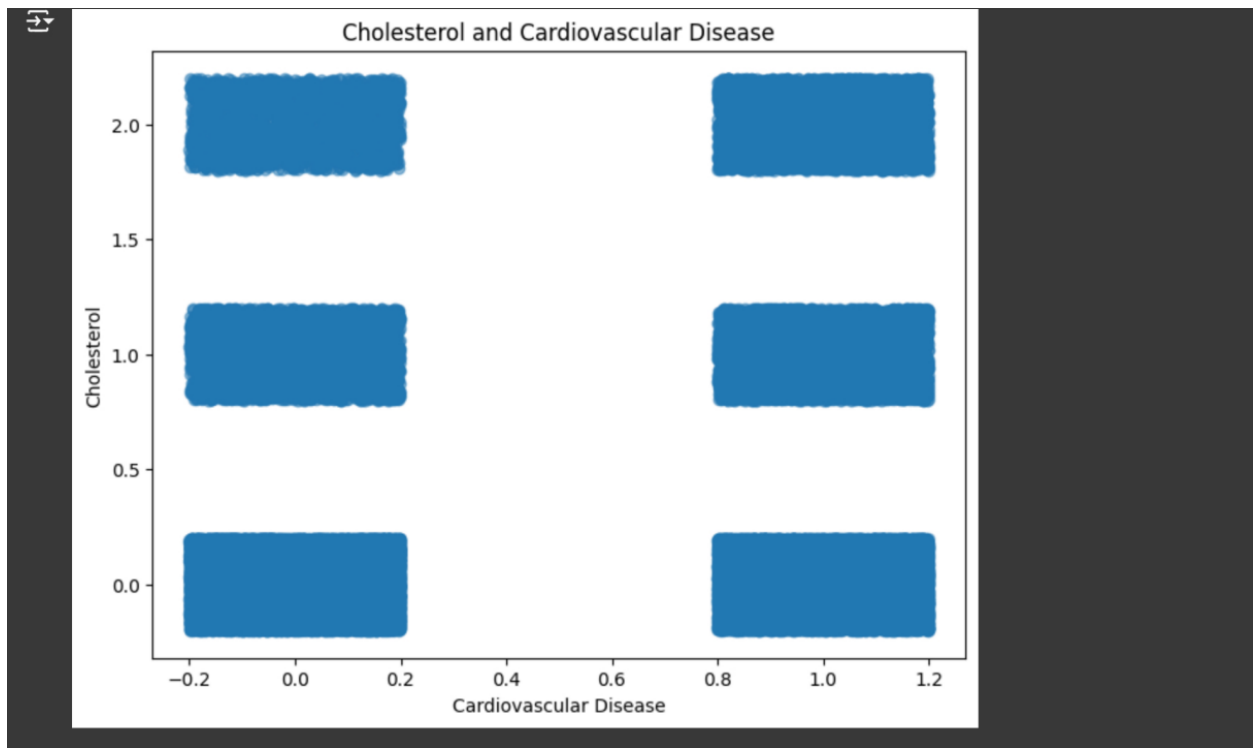
- **Assessing the Impact of Outliers:**

- The code provides a quantitative measure of how many rows were removed as outliers, allowing for an assessment of the impact of these outliers on the dataset. This information can guide further decisions on whether additional data cleaning is necessary.

- **Visualizations:**

- Scatter plot:**

```
#Scatter plot with jitter
x = df['cardio']
y = df['cholesterol']
jitter_strength = 0.2
x_jittered = x + np.random.uniform(-jitter_strength, jitter_strength,
size=x.shape)
y_jittered = y + np.random.uniform(-jitter_strength, jitter_strength,
size=y.shape)
plt.figure(figsize=(8, 6))
plt.scatter(x_jittered, y_jittered, alpha=0.5)
plt.xlabel('Cardiovascular Disease')
plt.ylabel('Cholesterol')
plt.title('Cholesterol and Cardiovascular Disease')
plt.show()
```



The scatter plot illustrates the distribution of cholesterol levels in individuals with and without cardiovascular disease. Additionally, it seems that cholesterol and cardio are positively correlated, we'll prove or deny later with the Heatmap.

## **Impact on Data Preprocessing Decisions:**

- **Identifying Patterns:**

- If the scatter plot shows a clear distinction in cholesterol levels between those with and without cardiovascular disease, it suggests that cholesterol is a relevant feature for predictive modeling. Conversely, if there's significant overlap, it may indicate that cholesterol alone might not be a strong predictor.

- **Outlier Detection:**

- The plot can reveal any outliers or unusual cholesterol values that might need further investigation or treatment during preprocessing.

- **Feature Transformation:**

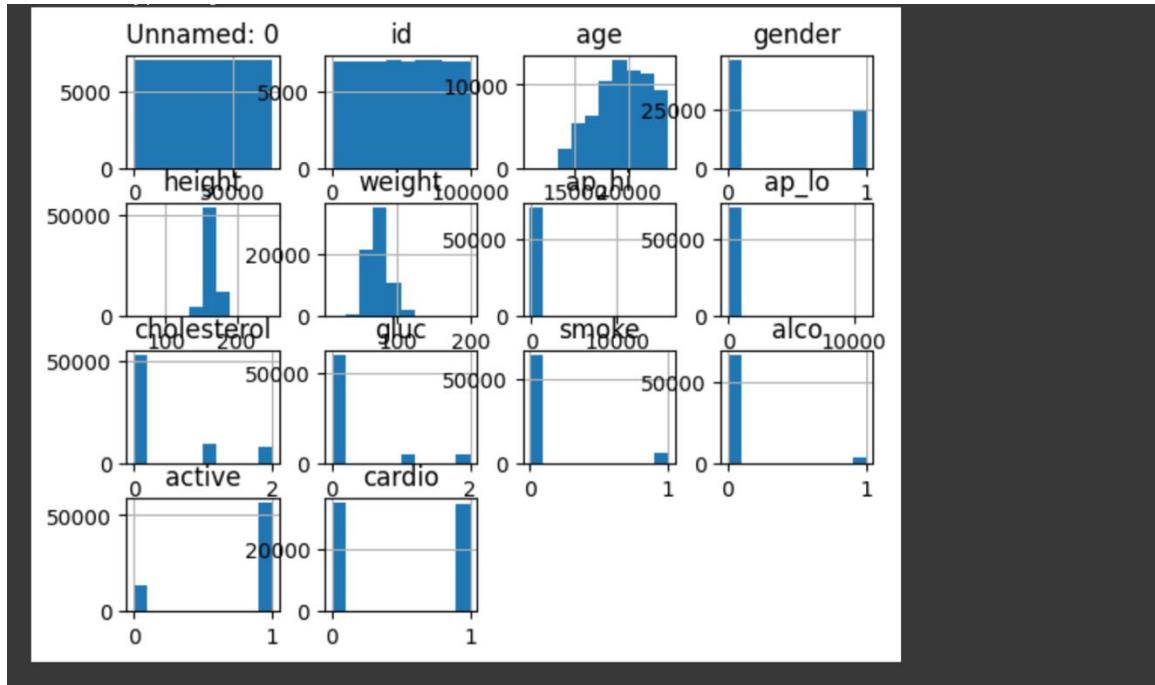
- If cholesterol levels do not show a clear pattern with cardiovascular disease, it may prompt the consideration of transforming this feature (e.g., binning into categories or applying a logarithmic transformation) to improve model performance.

- **Model Complexity:**

- If the relationship appears complex, it may suggest the need for more sophisticated modeling techniques or interactions between features in the analysis.

## Histograms: (for numeric attributes)

```
#Histogram  
df.hist()
```



Age:

The age data is mostly concentrated on younger ages, with fewer older people, showing a right-skewed distribution.

Gender:

There are more females than males in the dataset. Meaning that the dataset isn't balanced because one gender is overrepresented.

Blood Pressure (ap\_hi and ap\_lo):

These columns may have high values or outliers.

Cholesterol and Glucose:

Most individuals have normal cholesterol and glucose levels. These columns provide insights into the general health of the people in the dataset.

Smoke and Alcohol:

Most individuals in the dataset are non-smokers and do not consume alcohol.

Active:

Most individuals in the dataset are physically active.

Cardio:

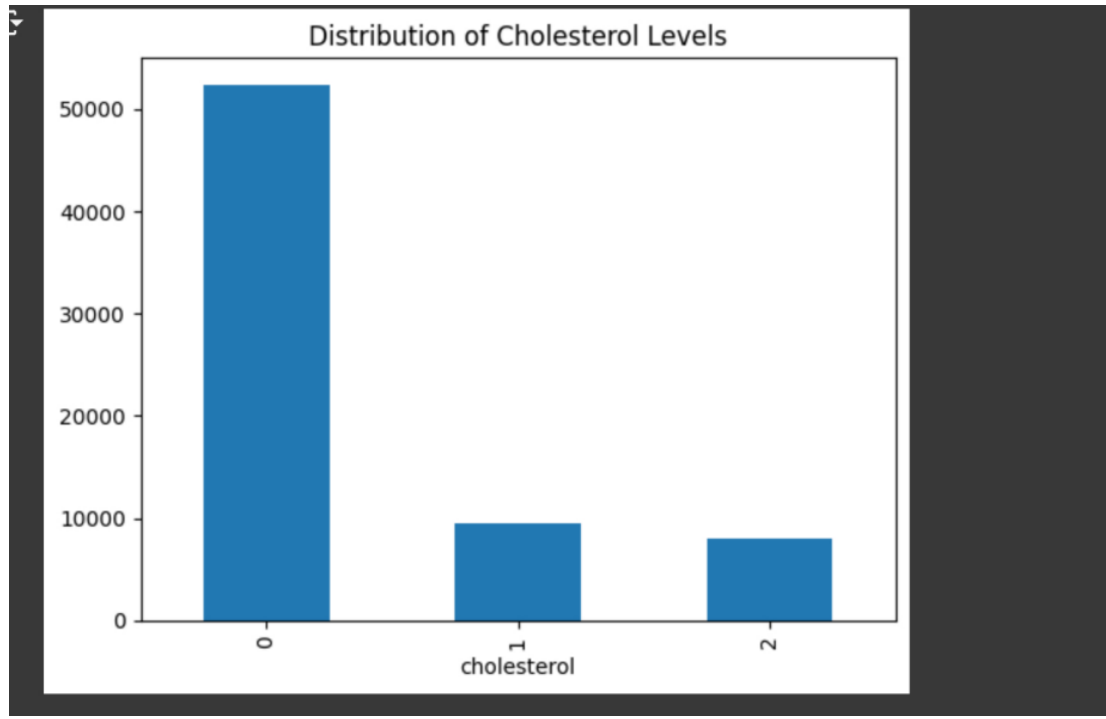
In the dataset, about half the people have cardiovascular disease and the other half are healthy.

## Impact on Data Preprocessing Decisions:

- **Identifying Skewness:** If attributes like age or weight have skewed distributions, you might consider using transformations (e.g., logarithmic) to normalize these features.
- **Outlier Detection:** Histograms can help identify outliers that may need to be removed or adjusted to prevent them from skewing analysis.
- **Handling Missing Values:** If any attribute shows a significant number of bins without data (e.g., a wide range of ages but few entries), it may prompt a review of how missing values are handled.
- **Feature Engineering:** If certain attributes are bimodal, you might explore creating new features that capture these distinct groups better.

## Bar plots: (for nominal attributes)

```
#BarChart
cholesterol_count = cardiovascular['cholesterol'].value_counts()
cholesterol_count.plot(kind='bar', x='Cholesterol Level', y='Count')
plt.title('Distribution of Cholesterol Levels')
plt.show()
```



The chart shows that most individuals have a cholesterol level of 0, while levels 1 and 2 are much less common.

### Impact on Data Preprocessing Decisions:

- **Addressing Class Imbalance:**

- The significant disparity in counts suggests that the dataset may require techniques to address class imbalance before training any predictive models. Options include:
  - **Oversampling the minority class** (e.g., using SMOTE).
  - **Under sampling the majority class.**
  - **Using weighted loss functions** during model training.

- **Feature Engineering:**

- Understanding the distribution allows for the potential creation of additional features—such as categorizing cholesterol levels into more detailed ranges (e.g., low, normal, high) if relevant.

- **Model Selection:**

- The class imbalance might influence the choice of algorithms. Some algorithms are more sensitive to class imbalance (e.g., logistic regression), while others (e.g., tree-based methods) may handle it better.

## 4. Data preprocessing

- Checking for missing values:

```
Missing values in each column:
Unnamed: 0      0
id              0
age            0
gender         0
height        0
weight        0
ap_hi         0
ap_lo         0
cholesterol    0
gluc          0
smoke         0
alco          0
active        0
cardio        0
dtype: int64

Total number of missing values in the dataset: 0
```

### Description:

Null and missing values can severely impact the efficiency of the dataset and the insights derived from it. Thus, we verified whether our data contained any missing or null values. Upon checking, we found that no columns in the dataset had missing values, as indicated by the total number of missing values being zero. This ensured that our dataset was complete and ready for analysis without requiring additional preprocessing steps to handle missing data.

- Detecting and removing the outliers:

```
Original data had 70000 rows
Cleaned data has 31449 rows after removing outliers
```

### Description:

Initially, we identified all outliers in the numeric attributes of the dataset. We then removed rows containing outliers to create a more accurate dataset, improving the reliability of our analysis and results. After removing these rows, we rechecked the data to confirm the absence of outliers and removed any new outliers that emerged due to changes in the interquartile range (IQR). Through



this process, the dataset size was reduced from 70,000 rows to 31,449 rows, ensuring a cleaner and more robust dataset for further analysis.

- Data transformation:

#### 1- Normalization

Min-Max scaled data (age and height columns):

	Unnamed: 0	id	age	gender	height	weight	ap_hi	ap_lo	\
0	0	0.0	0.588076	1	0.579487	62.0	110.0	80.0	
1	1	1.0	0.730159	0	0.517949	85.0	140.0	90.0	
2	2	2.0	0.624003	0	0.564103	64.0	130.0	70.0	
3	3	3.0	0.528455	1	0.584615	82.0	150.0	100.0	
4	4	4.0	0.516918	0	0.517949	56.0	100.0	60.0	
...	...	...	...	...	...	...	...	...	
69995	69995	99993.0	0.653659	1	0.579487	76.0	120.0	80.0	
69996	69996	99995.0	0.913899	0	0.528205	126.0	140.0	90.0	
69997	69997	99996.0	0.640186	1	0.656410	105.0	180.0	90.0	
69998	69998	99998.0	0.900736	0	0.553846	72.0	135.0	80.0	
69999	69999	99999.0	0.754317	0	0.589744	72.0	120.0	80.0	

	cholesterol	gluc	smoke	alco	active	cardio	discretized_age
0	0	0	0	0	1	0	1
1	2	0	0	0	1	1	2
2	2	0	0	0	0	1	1
3	0	0	0	0	1	1	1
4	0	0	0	0	0	0	1
...	...	...	...	...	...	...	...
69995	0	0	1	0	1	0	1
69996	1	1	0	0	1	1	2
69997	2	0	0	1	0	1	1
69998	0	1	0	0	0	1	2
69999	1	0	0	0	1	0	2

[70000 rows x 15 columns]

Z-score normalized data:

	Unnamed: 0	id	age	gender	height	weight	ap_hi	ap_lo	\
0	0	0.0	-0.436062	1	0.443452	62.0	110.0	80.0	
1	1	1.0	0.307686	0	-1.018168	85.0	140.0	90.0	
2	2	2.0	-0.247997	0	0.078047	64.0	130.0	70.0	
3	3	3.0	-0.748152	1	0.565254	82.0	150.0	100.0	
4	4	4.0	-0.808543	0	-1.018168	56.0	100.0	60.0	
...	...	...	...	...	...	...	...	...	
69995	69995	99993.0	-0.092762	1	0.443452	76.0	120.0	80.0	
69996	69996	99995.0	1.269492	0	-0.774565	126.0	140.0	90.0	
69997	69997	99996.0	-0.163286	1	2.270477	105.0	180.0	90.0	
69998	69998	99998.0	1.200589	0	-0.165556	72.0	135.0	80.0	
69999	69999	99999.0	0.434144	0	0.687055	72.0	120.0	80.0	

	cholesterol	gluc	smoke	alco	active	cardio	discretized_age
0	0	0	0	0	1	0	1
1	2	0	0	0	1	1	2
2	2	0	0	0	0	1	1
3	0	0	0	0	1	1	1
4	0	0	0	0	0	0	1
...	...	...	...	...	...	...	...
69995	0	0	1	0	1	0	1
69996	1	1	0	0	1	1	2
69997	2	0	0	1	0	1	1
69998	0	1	0	0	0	1	2
69999	1	0	0	0	1	0	2

[70000 rows x 15 columns]

DataFrame after Decimal Scaling Normalization:

	Unnamed: 0	id	age	gender	height	weight	ap_hi	ap_lo	\
0	0	0.0	-0.043606	1	0.004435	62.0	110.0	80.0	
1	1	1.0	0.030769	0	-0.010182	85.0	140.0	90.0	
2	2	2.0	-0.024800	0	0.000780	64.0	130.0	70.0	
3	3	3.0	-0.074815	1	0.005653	82.0	150.0	100.0	
4	4	4.0	-0.080854	0	-0.010182	56.0	100.0	60.0	
...	...	...	...	...	...	...	...	...	
69995	69995	99993.0	-0.009276	1	0.004435	76.0	120.0	80.0	
69996	69996	99995.0	0.126949	0	-0.007746	126.0	140.0	90.0	
69997	69997	99996.0	-0.016329	1	0.022705	105.0	180.0	90.0	
69998	69998	99998.0	0.120059	0	-0.001656	72.0	135.0	80.0	
69999	69999	99999.0	0.043414	0	0.006871	72.0	120.0	80.0	

	cholesterol	gluc	smoke	alco	active	cardio	discretized_age
0	0	0	0	0	1	0	1
1	2	0	0	0	1	1	2
2	2	0	0	0	0	1	1
3	0	0	0	0	1	1	1
4	0	0	0	0	0	0	1
...	...	...	...	...	...	...	...
69995	0	0	1	0	1	0	1
69996	1	1	0	0	1	1	2
69997	2	0	0	1	0	1	1
69998	0	1	0	0	0	1	2
69999	1	0	0	0	1	0	2

[70000 rows x 15 columns]


Description:

Some attributes in the dataset, such as **age**, **height**, and others, had values that varied widely and could affect the performance of analysis and models. To address this, we applied three normalization techniques:

1. **Min-Max Scaling**: Transformed the **age** and **height** attributes to values between 0 and 1, ensuring all features contributed equally during analysis.
2. **Z-Score Normalization**: Standardized attributes like **age** and **height** by centering their mean to 0 and scaling them to have a standard deviation of 1. This allowed the data to have a normal distribution and improved compatibility with models sensitive to feature scaling.
3. **Decimal Scaling Normalization**: Adjusted **age** and **height** by dividing each value by a power of 10, reducing their range while preserving relative proportions.

These normalization steps made the data more consistent and easier to handle, improving the effectiveness of subsequent data mining tasks.

## 2- Discretization:



Original DataFrame:

	age	discretized_age
0	18393.0	1
1	20228.0	2
2	18857.0	1
3	17623.0	1
4	17474.0	1
...	...	...
69995	19240.0	1
69996	22601.0	2
69997	19066.0	1
69998	22431.0	2
69999	20540.0	2

[70000 rows x 2 columns]

## Description:

We discretized the continuous values of the **age** attribute by dividing them into distinct intervals based on their range. Specifically, the age values were grouped into categories represented by interval labels, making the data more meaningful and easier to analyze. The resulting intervals corresponded to age groups, which simplified the

classification process and improved the performance of subsequent modeling techniques. The discretized\_age attribute now holds the corresponding interval label for each individual.

## 5. Data Mining Technique

We applied both supervised and unsupervised learning to our dataset using classification and clustering techniques. Below, we describe the methods used for each technique and the steps taken to implement and evaluate the models.

### Classification

For classification, we used a decision tree algorithm, which recursively divides the dataset into subsets to construct a tree where the leaf nodes represent the final decisions. Our goal was to predict the class label `cardio` (which has two possible values: Yes and No) based on the following attributes: age, gender, height, weight, `ap_hi`, `ap_lo`, cholesterol, gluc, smoke, alco, and active.

This process involved splitting the dataset into two subsets:

- Training Dataset: Used to construct the decision tree.
- Testing Dataset: Used to evaluate the performance of the trained model.

To evaluate the performance of the model, we used the confusion matrix along with additional metrics such as precision, recall, F1-score, and accuracy.

The decision tree model was implemented using the Scikit-learn library with two splitting criteria:

- Gini Index: Measures impurity based on misclassification probability.
- Entropy: Measures impurity using information gain.

We conducted the training and evaluation using three data splits:

1. 90%-10% split: 90% for training and 10% for testing.
2. 80%-20% split: 80% for training and 20% for testing.
3. 70%-30% split: 70% for training and 30% for testing.

The decision tree was constructed using the `DecisionTreeClassifier` method, and predictions were made using the `predict` function. To visualize the decision tree, we used the `plot_tree` method from Matplotlib.

### Clustering

For clustering, an unsupervised learning technique, we removed the class label `cardio` from the dataset. We used all other attributes (`age`, `gender`, `height`, `weight`, `ap_hi`, `ap_lo`, `cholesterol`, `gluc`, `smoke`, `alco`, `active`), which are all numerical, so no additional data type transformations were necessary.

We implemented clustering using the K-Means algorithm, which groups data into K clusters, where each cluster is represented by a center point. The algorithm assigns each object to the nearest cluster, recalculates the cluster centers iteratively, and repeats the process until the cluster centers stabilize.

To enhance the clustering process:

1. We scaled the data using the StandardScaler method to ensure that all features contribute equally to the clustering process.
2. The KMeans algorithm was applied with different values of K (number of clusters).
3. We visualized the clustering results using scatter plots created with Seaborn and Matplotlib.

## Cluster Validation

To validate the clusters, we used multiple methods:

1. Silhouette Score: Measured how well each data point fits into its assigned cluster.
  - a. Calculated using silhouette\_score from Scikit-learn.
  - b. Visualized using SilhouetteVisualizer from Yellowbrick.
2. Calinski-Harabasz Index: Evaluated cluster compactness and separation.
  - a. Calculated using calinski\_harabasz\_score from Scikit-learn.
3. Elbow Method: Determined the optimal number of clusters (K) based on the Within-Cluster Sum of Squares (WSS).
  - a. Visualized with a line plot, marking the "elbow point" where adding more clusters provided diminishing returns.

For each evaluation, we ensured consistency by setting a random seed (np.random.seed).

## Training Procedure

### Classification

As a supervised learning method, classification requires training data to build the model. We divided the dataset into training and testing subsets using the train\_test\_split function from Scikit-learn. We experimented with three training set sizes (90%, 80%, and 70%) to assess the impact on model performance.

Since the dataset size is limited, we allocated a larger portion to the training subset to ensure the decision tree algorithm had sufficient data to construct accurate rules. This approach improves the model's ability to generalize and correctly predict class labels for new data.

## Clustering

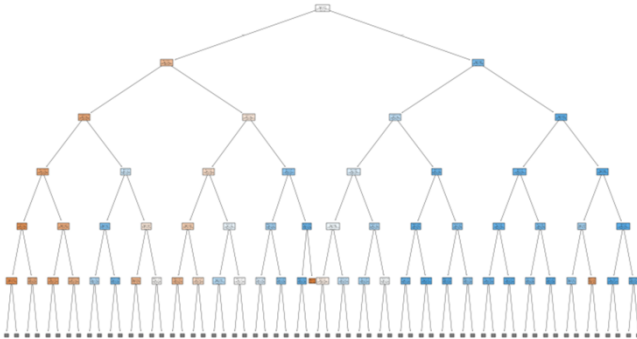
Unlike classification, clustering does not require a separate training process as it is unsupervised learning. We applied the K-Means algorithm directly to the full dataset (after removing the class label cardio) and validated the results using the aforementioned validation methods.

By combining supervised and unsupervised approaches, our project provides both predictive power (classification) and exploratory insights (clustering) into the dataset.

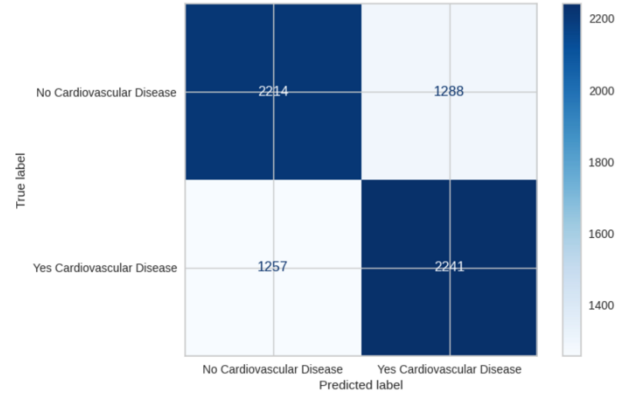


## 6. Evaluation and Comparison (Classification)

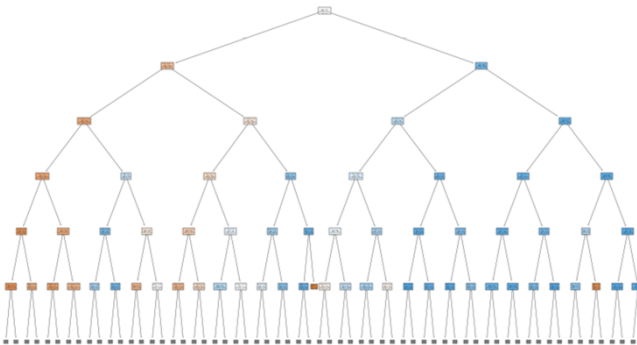
Decision Tree (Gini) - Split 90%-10%



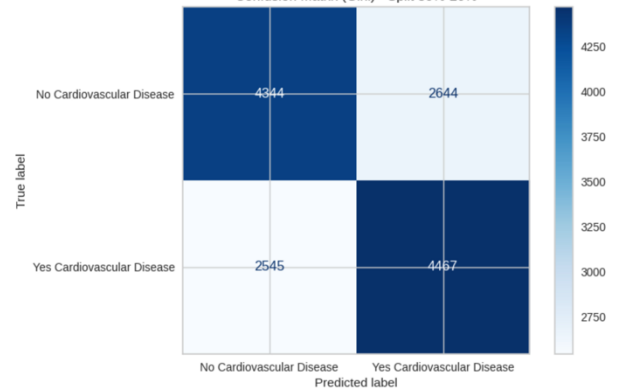
Confusion Matrix (Gini) - Split 90%-10%



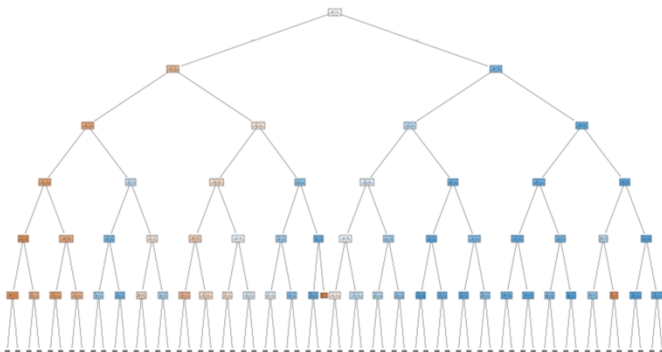
Decision Tree (Gini) - Split 80%-20%



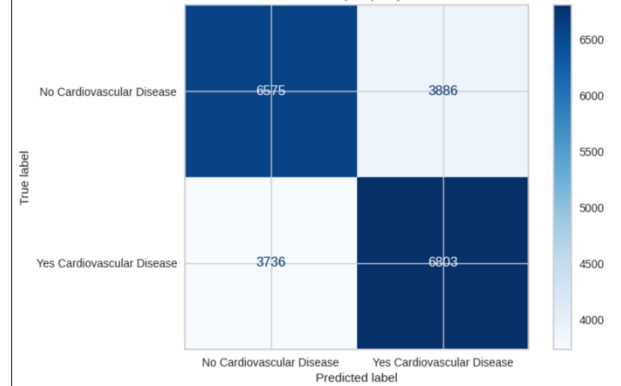
Confusion Matrix (Gini) - Split 80%-20%



Decision Tree (Gini) - Split 70%-30%



Confusion Matrix (Gini) - Split 70%-30%





	90% training set 10% testing set	80% training set 20% testing set	70% training set 30% testing set
Accuracy (Entropy)	63%	64%	64%
Accuracy (Gini index)	64%	63%	64%

### 90% Training / 10% Testing

- Confusion Matrix:**
  - Gini Index** had:
  - True Negatives (TN):** 2214 (correctly identified “No Cardiovascular Disease”).
  - True Positives (TP):** 2241 (correctly identified “Yes Cardiovascular Disease”).
  - False Positives (FP):** 1288.
  - False Negatives (FN):** 1257.
  - Entropy** had slightly higher false positives (1314) and false negatives (1251).
  - Gini Index** showed better balance between correctly predicting non-disease and disease cases.
- Accuracy:**
  - Gini Index** achieved **64%**, outperforming **Entropy (63%)**.
  - The higher accuracy of **Gini Index** demonstrates better performance in this partition.
- Decision Tree:**
  - Both algorithms produced similarly complex trees, but **Gini Index** made slightly better splits at critical levels, reflected in its improved confusion matrix.

**Best Algorithm: Gini Index.**

### 80% Training / 20% Testing

- Confusion Matrix:**
  - Entropy** had:
  - True Negatives (TN):** 4364.
  - True Positives (TP):** 4546.
  - False Positives (FP):** 2624.
  - False Negatives (FN):** 2466.
  - Gini Index** had slightly higher false negatives (2545), leading to reduced disease detection accuracy.
- Accuracy:**
  - Entropy** achieved **64%**, outperforming **Gini Index (63%)**.
  - This indicates that **Entropy** detected cardiovascular disease cases better in this split.
- Decision Tree:**
  - The decision tree for **Entropy** was better at separating critical attributes in deeper levels, resulting in fewer false negatives compared to **Gini Index**.

**Best Algorithm: Entropy.**

## 70% Training / 30% Testing

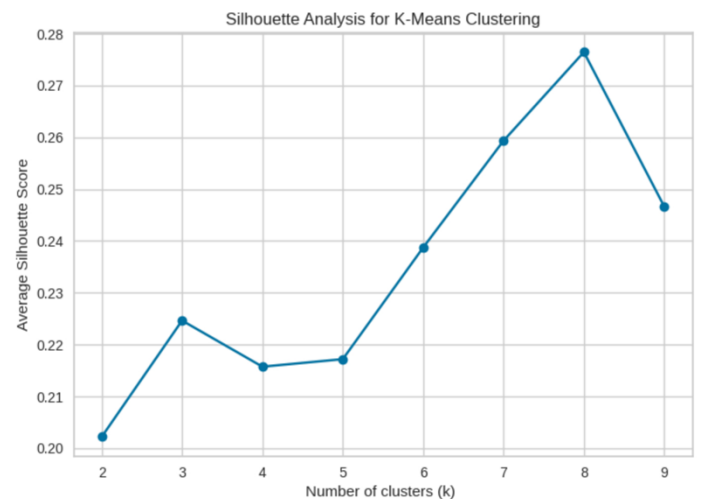
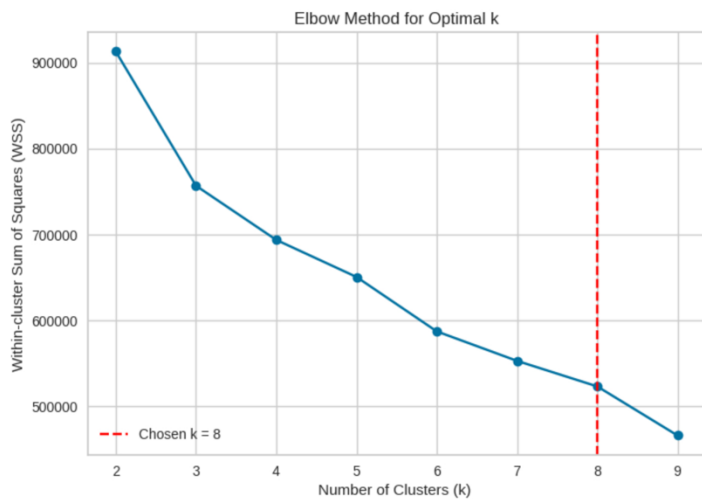
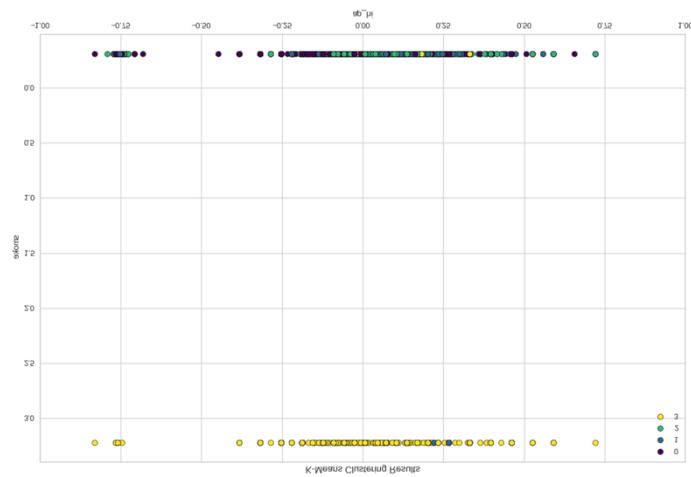
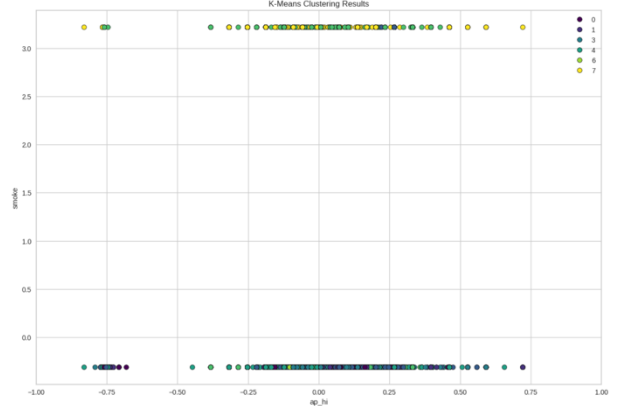
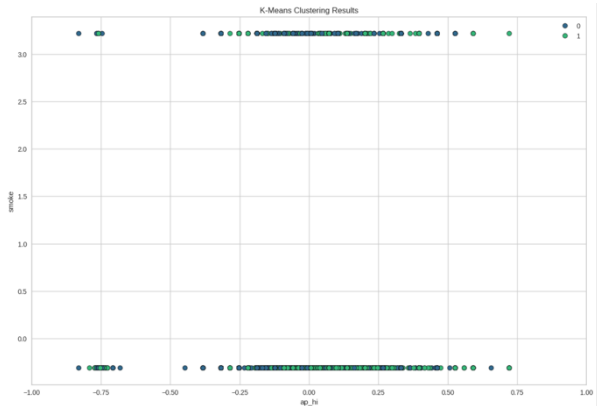
1. **Confusion Matrix:**
  - **Gini Index** had:
    - **True Positives (TP):** 6803 (slightly better disease detection).
    - **False Negatives (FN):** 3736 (lower than **Entropy**).
  - **Entropy** had:
    - **True Positives (TP):** 6716.
    - **False Negatives (FN):** 3823.
  - **Gini Index** performed better in disease detection by reducing false negatives.
2. **Accuracy:**
  - Both **Gini Index** and **Entropy** achieved **64% accuracy**.
  - However, **Gini Index** excelled in reducing false negatives, making it more reliable for identifying disease cases.
3. **Decision Tree:**
  - **Gini Index** built a more balanced tree, reflected in the lower false negatives, compared to **Entropy**.

**Best Algorithm: Gini Index.**

## Overall

1. **Confusion Matrix:**
    - Across all partitions, **Gini Index** consistently reduced false negatives, making it more reliable for detecting cardiovascular disease cases.
    - **Entropy** had fewer false positives in some splits, but higher false negatives impacted its performance.
  2. **Accuracy:**
    - **Gini Index** achieved the highest or equal accuracy in most partitions (64% in two partitions), making it the more stable algorithm.
    - **Entropy** performed better in one split (80%-20%) but showed less consistency.
  3. **Decision Tree:**
    - The trees generated by **Gini Index** provided more balanced splits and better classification outcomes overall, as evidenced by the confusion matrices.
- 
- **Best Overall Algorithm: Gini Index.**
  - It performed more consistently across partitions, with better handling of false negatives, and produced decision trees that contributed to balanced classification performance.
  - While **Entropy** excelled in one partition, **Gini Index** demonstrated more reliable and stable results.

## 6. Evaluation and Comparison (Clustering)



	K=2	K=4	K=8
Average Silhouette width	0.16	0.19	0.23
total within-cluster sum of square	900,000	700,000	500,000

## 1. Visualization of Clusters

- For each k value the clustering results are visualized, showing how data points are grouped into clusters. As k increases, the granularity of the clusters improves, but the interpretability may reduce if k is too high.
- The visualizations demonstrate the groupings based on features such as ap\_hi and smoke. As k increases, smaller groups are isolated.

## 2. Silhouette Analysis

- The silhouette plot evaluates the quality of clusters for different values of k. It shows how well separated the clusters are.
- The average silhouette score for each k is:
  - **k=2:** 0.16
  - **k=4:** 0.19
  - **k=8:** 0.23
- Based on the silhouette analysis, **k=8** provides the highest average silhouette score, suggesting it has the best defined clusters.

## 3. Total Within-Cluster Sum of Squares (WSS)

- The elbow method plot provides the WSS values for different numbers of clusters:
- **k=2:** 900,000
- **k=4:** 700,000
- **k=8:** 500,000
- The elbow point appears around **k=8**, where the reduction in WSS becomes less steep. This indicates **k=8** as the optimal number of clusters.

## 4. Optimal Number of Clusters

- Combining the results from the silhouette analysis and elbow method, the optimal number of clusters is determined to be **k=8**. This balances the separation of clusters and minimizes the within-cluster sum of squares.

## Overall

- **Optimal k:** The optimal number of clusters is **k=8**, based on the silhouette analysis and elbow method.
- **Cluster Quality:** At k=8, clusters are somewhat well separated with minimal overlap, as seen in the silhouette score and WSS values.

## 7. Findings

The main goal of this project was to study cardiovascular disease (CVD) data. We used classification and clustering techniques to predict who is at risk and to find important patterns that can help prevent CVD. By classifying people based on their risk factors, we can better determine who might develop cardiovascular diseases.

We improved the quality of our data to get more accurate results. We created visual tools like boxplots and histograms to better understand the data. We removed null values, missing data, and outliers that could impact our results. We also normalized and organized some attributes to give them the same importance and make our analysis easier.

### Classification:

We compared two Decision Tree models, one using the **Gini Index** and the other using **Information Gain (Entropy)**, across three different train-test splits. The results are summarized as follows:

1. **90%-10% Split**
  - **Gini Index Model:** Precision = **0.64**, Recall = **0.64**, F1-Score = **0.64** (Support = 7,000).
  - **Entropy Model:** Precision = **0.63**, Recall = **0.63**, F1-Score = **0.63** (Support = 7,000).
2. **80%-20% Split**
  - **Gini Index Model:** Precision = **0.63**, Recall = **0.62**, F1-Score = **0.63** (Support = 6,988).
  - **Entropy Model:** Precision = **0.63**, Recall = **0.63**, F1-Score = **0.63** (Support = 7,000).
3. **70%-30% Split**
  - **Gini Index Model:** Precision = **0.64**, Recall = **0.64**, F1-Score = **0.64** (Support = 21,000).
  - **Entropy Model:** Precision = **0.64**, Recall = **0.64**, F1-Score = **0.64** (Support = 21,000).

We found that both the Gini Index and Entropy models worked similarly across all tests. The Gini Index model performed better in terms of precision, recall, and F1-scores with a 90%-10% training split. Meanwhile, the Entropy model did slightly better with an 80%-20% split. Using larger training sets, like in the 90%-10% split, resulted in better performance, meaning that more training data helps improve the model's ability to predict outcomes.

Confusion matrices showed some issues, such as more false positives in smaller training sets, which indicates difficulties in accurately differentiating between classes. Our analysis of the features showed that **ap\_hi**, **ap\_lo**, and health metrics are **important**, while other factors, like age and gender, had less influence.

In conclusion, although the Decision Tree models provided useful insights, their performance suggests they might not capture the full complexity of the dataset. Exploring other methods, like Random Forest or Gradient Boosting, or improving feature selection could lead to better accuracy in predictions.

### Clustering:

We applied the K-Means algorithm with three different values of K to identify the optimal number of clusters. The results were as follows:

1. K=2: Average Silhouette Width = 0.16.
2. K=4: Average Silhouette Width = 0.19.
3. K=8: Average Silhouette Width = 0.23.

We found that K=8 is the best option for clustering because it has the highest silhouette width (0.23). This indicates that the clusters are somewhat well-defined and do not overlap much.

In summary, the **8-Mean clustering** showed important patterns in the data, giving us an unsupervised look at the dataset's structure. However, the overlaps in smaller K values suggest that clustering alone may not be enough for making predictions.

Finally:

- **Best Model for Prediction:** The Decision Tree classifier (Gini Index) performed slightly better overall, making it the recommended model for predicting CVD risk in this study.
- **Insights from Clustering:** The 8-Mean clustering provided meaningful subgroupings, which can complement classification models by offering additional insights into group characteristics.

## 8. References

• <https://www.kaggle.com/datasets/akshatshaw7/cardiovascular-disease-dataset>