Heart Disease Risk Prediction using Machine Learning Models

Student ID	Name	Section
206001007	Zehra KOLAT	1
220911757	Kutay ŞAHİNLER	2
2309111082	Göktuğ ŞAHİN	3

Table 1: Team Members

COE305 - Machine Learning

FEMILDA JOSEPHIN JOSEPH SHOBANA BAI

Istinye University
Faculty of Engineering and Natural Sciences

Heart Disease Risk Prediction using Machine Learning Models

206001007 Zehra KOLAT Section 1
220911757 Kutay ŞAHİNLER Section 2
2309111082 Göktuğ ŞAHİN Section 3

23/11/2025

Stage 2 - Dataset & Exploratory Data Analysis (EDA) Report

Contents

1 Dataset Link:

https://www.kaggle.com/datasets/pratyushpuri/heart-disease-dataset-3k-

Kaggle Dataset Link

2 Data Pre-processing

2.1 Handling Missing Values

```
# Importing required libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE
```

```
# Load the dataset
df = pd.read_csv("heart_disease_dataset.csv")  # giving messy db to the data frame
print(df.head()) # this is our first 5 row
```

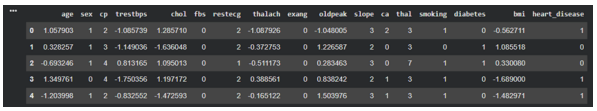
```
...   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope  \
0   67    1   2       111   536    0        2       88      0      1.3      3
1   57    1   3       109   107    0        2      119      0      5.4      2
2   43    1   4       171   508    0        1      113      0      3.7      3
3   71    0   4        90   523    0        2      152      0      4.7      2
4   36    1   2       119   131    0        2      128      0      5.9      3

    ca  thal  smoking  diabetes   bmi  heart_disease
0    2     3        1         0  23.4              1
1    0     3        0         1  35.4              0
2    0     7        1         1  29.9              0
3    1     3        1         0  15.2              1
4    1     3        1         0  16.7              1
```

Figure 1: output

The *df.head()* function displays the first five rows of the dataset. This output was used to verify that the data was loaded correctly and that each column was in the expected format. It was determined that the target variable, heart_disease, was correctly defined as 0 and 1. We've successfully loaded the 'heart_disease_dataset.csv' dataset into a Pandas DataFrame.

2.2 Feature Scaling / Normalization

```
from sklearn.preprocessing import StandardScaler

numerical_features = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak', 'bmi']

scaler = StandardScaler()

df_scaled = df.copy()

df_scaled[numerical_features] = scaler.fit_transform(df_scaled[numerical_features])

df_scaled.head()
df = df_scaled
```



	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	smoking	diabetes	bmi	heart_disease
0	1.057903	1	2	-1.085739	1.285710	0	2	-1.087926	0	-1.048005	3	2	3	1	0	-0.562711	1
1	0.328257	1	3	-1.149036	-1.636048	0	2	-0.372753	0	1.226587	2	0	3	0	1	1.085518	0
2	-0.693246	1	4	0.813165	1.095013	0	1	-0.511173	0	0.283463	3	0	7	1	1	0.330080	0
3	1.349761	0	4	-1.750356	1.197172	0	2	0.388561	0	0.838242	2	1	3	1	0	-1.689000	1
4	-1.203998	1	2	-0.832552	-1.472593	0	2	-0.165122	0	1.503976	3	1	3	1	0	-1.482971	1

Figure 2: output

All numerical features (age, tbps, cholesterol, thalassic, ex-peak, body mass index) were standardized using StandardScaler. This scaling transforms each feature so that its mean is ≈ 0 and its standard deviation is ≈ 1, ensuring that all variables contribute equally to the machine learning models regardless of their original scale.

Positive values indicate measurements above the mean.

Negative values indicate measurements below the mean.

Values close to zero are close to the mean.

Examples:

age = 1.0579 $\rightarrow$ This individual's age is approximately 1 standard deviation above the mean.

cholesterol = -1.6360 $\rightarrow$ This individual's cholesterol is approximately 1.6 standard deviations below the mean.

This normalization reduces the bias that could arise from variables measured at different scales and improves the performance and convergence of machine learning models.

2.3 Check for duplicate rows

```
print(df.info()) # lets look if there any empty cells, as you can see there is not
```

Output 1:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3069 entries, 0 to 3068
Data columns (total 17 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             3069 non-null   int64
 1   sex             3069 non-null   int64
 2   cp              3069 non-null   int64
 3   trestbps        3069 non-null   int64
 4   chol            3069 non-null   int64
 5   fbs             3069 non-null   int64
 6   restecg         3069 non-null   int64
 7   thalach         3069 non-null   int64
 8   exang           3069 non-null   int64
 9   oldpeak         3069 non-null   float64
 10  slope           3069 non-null   int64
 11  ca              3069 non-null   int64
 12  thal            3069 non-null   int64
 13  smoking         3069 non-null   int64
 14  diabetes        3069 non-null   int64
 15  bmi             3069 non-null   float64
 16  heart_disease   3069 non-null   int64
dtypes: float64(2), int64(15)
memory usage: 407.7 KB
None
```

```
print("Number of duplicate rows: ", df.duplicated().sum())
# lets check if there is any duplicate rows, there is not
```

Output 2:

```
Number of duplicate rows:  0
```

As we can see from the two code blocks:

We used the *df.info*() command to understand the overall structure and quality of the dataset. According to the output, our dataset consists of a total of 3069 rows and 17 columns.

The most important finding is that there are no missing data. The 'Non-Null Count' column in the output confirmed that there are 3069 full records across all 17 attributes, a perfect match to the total number of rows.

Furthermore, the data type (Dtype) of all columns appears as int64 or float64. This indicates that all the data is already in numeric format and ready for machine learning models; we won't need to convert text data like 'Male'/'Female'.

2.4 Irrelevant columns

```
# All columns are relevant for predicting heart disease; no columns were removed.
print(df["heart_disease"].value_counts()) # wrong label checking : only 0 and 1 present
```

```
heart_disease
0    1878
1    1191
Name: count, dtype: int64
```

In this project, our goal is to predict the heart_disease column. So, We first needed to check this target column. We ran the *df*['*heart_disease*'].*value_counts*() command to ensure it only contained the values '0' (Not Patient) and '1' (Patient), meaning it wasn't a wrong label.

Fortunately, the result was only 0 and 1.

This command also showed us the class distribution in the dataset. Accordingly:

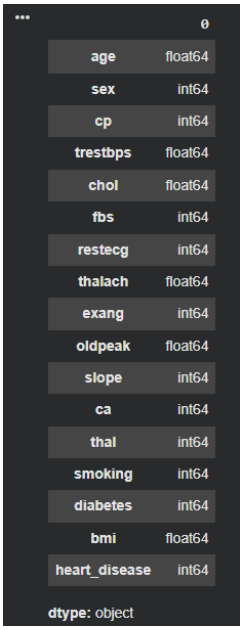
1878 people (61.2%) were labeled 'No Heart Disease' (0).

1191 people (38.8%) were labeled 'Heart Disease' (1).

We can also see the imbalanced data. This is why we will be doing smote in the later stages.

2.5 Encoding Categorical Variables – Label encoder/ One-hot encoder

```
df.dtypes
```



The data type of all columns appears as int64 or float64. This indicates that all the data is already in numeric format and ready for machine learning models; we won't need to convert text data like 'Male'/'Female'. Since all data is in numeric format, there is no need to apply label encoding or one-hot encoding . So no additional encoding step was applied.

2.6 Outliers detection

```
    numerical_cols = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak', 'bmi']

plt.figure(figsize=(15, 10))
for i, col in enumerate(numerical_cols, 1):
    plt.subplot(2, 3, i)
    sns.boxplot(data=df, x=col)
    plt.title(f'{col} Boxplot')
plt.tight_layout()
plt.show()
```

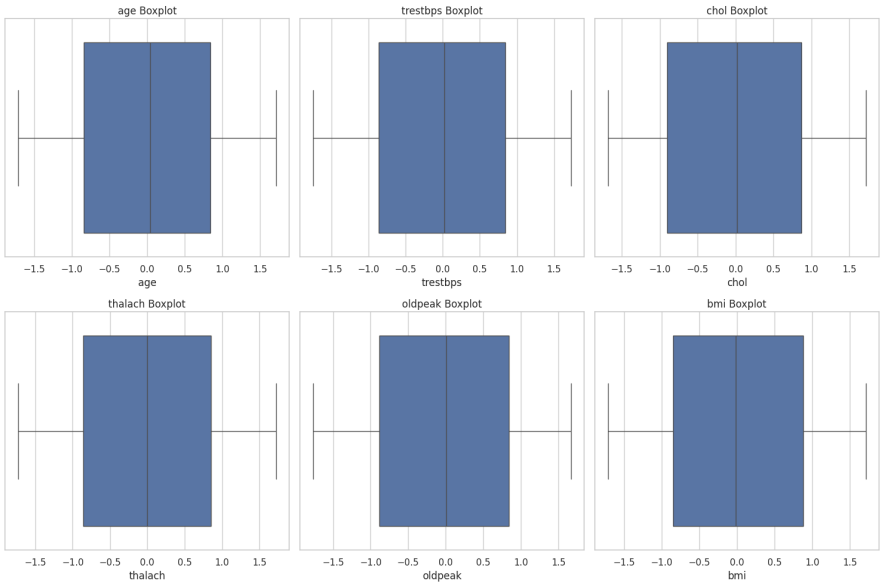


Figure 3: We look for continuous columns:

Outlier detection was performed for all numerical features (age, trestbps, chol, thalach, oldpeak, bmi) using boxplots. The analysis showed that there are no significant outliers in the dataset. Therefore, all numerical values were retained for model training, as the data is clean and does not contain extreme deviations that could bias the results.

3 Split & Smote

We prepare data for the model: split and smote to increase model accuracy.

```
    x = df.drop('heart_disease', axis=1) # delete last lable so x have only datas
y = df['heart_disease'] # y have only last lable 0 & 1

x_train, x_test, y_train, y_test = train_test_split (x, y, test_size = 0.2,
    random_state = 42, stratify = y) # as always we will siplit data to train and test
    with the ratio of 8 - 2

print("Original Training Counts:")
print(y_train.value_counts()) # showing last lable before SMOTE
```

```
    Original Training Counts:
heart_disease
0     1502
1      953
Name: count, dtype: int64
```

The dataset was split into training (80%) and testing (20%) sets. The training data showed class imbalance (0: 1502, 1: 953), which could bias the model. SMOTE will be applied to balance the classes:

```
smote = SMOTE(random_state=42) # in here we create blank smote func, it does not
    nothing yet
x_train_resampled, y_train_resampled = smote.fit_resample(x_train, y_train) # as we
    know fit is training keyword we use it to aplly SMOTE

print("\nBalanced Training Counts (After SMOTE):")
print(y_train_resampled.value_counts()) # showing last lable after SMOTE
```

```
Balanced Training Counts (After SMOTE):
heart_disease
1    1502
0    1502
Name: count, dtype: int64
```

SMOTE was applied to the training set to address class imbalance. After resampling, both classes are balanced with 1502 samples each, ensuring the model can learn equally from both classes.

4 Exploratory Data Analysis (EDA)

4.1 Statistical Summary

```
# Calculating basic statistics for numeric columns
numerical_cols = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak', 'bmi']
stat_summary = df[numerical_cols].describe().T  #

print(stat_summary)

medians = df[numerical_cols].median()
modes = df[numerical_cols].mode().iloc[0]

print("\nMedians:\n", medians)
print("\nModes:\n", modes)
```

```
...            count          mean       std       min       25%       50%  \
    age       3069.0  2.593053e-16  1.000163 -1.714750 -0.839175  0.036399
    trestbps  3069.0  4.306320e-16  1.000163 -1.750356 -0.864200  0.021955
    chol      3069.0 -1.892697e-16  1.000163 -1.683722 -0.900501  0.018934
    thalach   3069.0  2.199464e-17  1.000163 -1.733889 -0.857225 -0.003631
    oldpeak   3069.0 -2.118431e-16  1.000163 -1.769218 -0.881572  0.006074
    bmi       3069.0 -7.177199e-17  1.000163 -1.716470 -0.851150 -0.013301

                  75%       max
    age       0.839009  1.714584
    trestbps  0.844813  1.730969
    chol      0.870262  1.721590
    thalach   0.849963  1.726627
    oldpeak   0.838242  1.670410
    bmi       0.879489  1.717339

    Medians:
     age         0.036399
    trestbps     0.021955
    chol         0.018934
    thalach     -0.003631
    oldpeak      0.006074
    bmi         -0.013301
    dtype: float64

    Modes:
     age         0.109364
    trestbps     0.749868
    chol         0.379897
    thalach     -0.788014
    oldpeak      1.393020
    bmi          0.330080
    Name: 0, dtype: float64
```

Figure 4: output

All numerical features (age, trestbps, arm, height, height of the head, height of the head, body mass index) were standardized using StandardScaler, with a mean around 0 and a standard deviation around 1. Positive values indicate measurements above the mean, negative values indicate measurements below the mean, and values closer to 0 indicate measurements close to the mean. Median values close to 0 confirm that the data are centered, while mode values represent the most frequent observations on the normalized scale. This standardization ensures that all features contribute equally to machine learning models, preventing biases due to different scales.

4.2 Data Visualization

4.2.1 Histograms

```
numerical_cols = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak', 'bmi']

plt.figure(figsize=(15,10))
for i, col in enumerate(numerical_cols, 1):
    plt.subplot(2, 3, i)
    sns.histplot(df[col], kde=True, bins=20)
    plt.title(f'{col} Distribution')
plt.tight_layout()
plt.show()
```

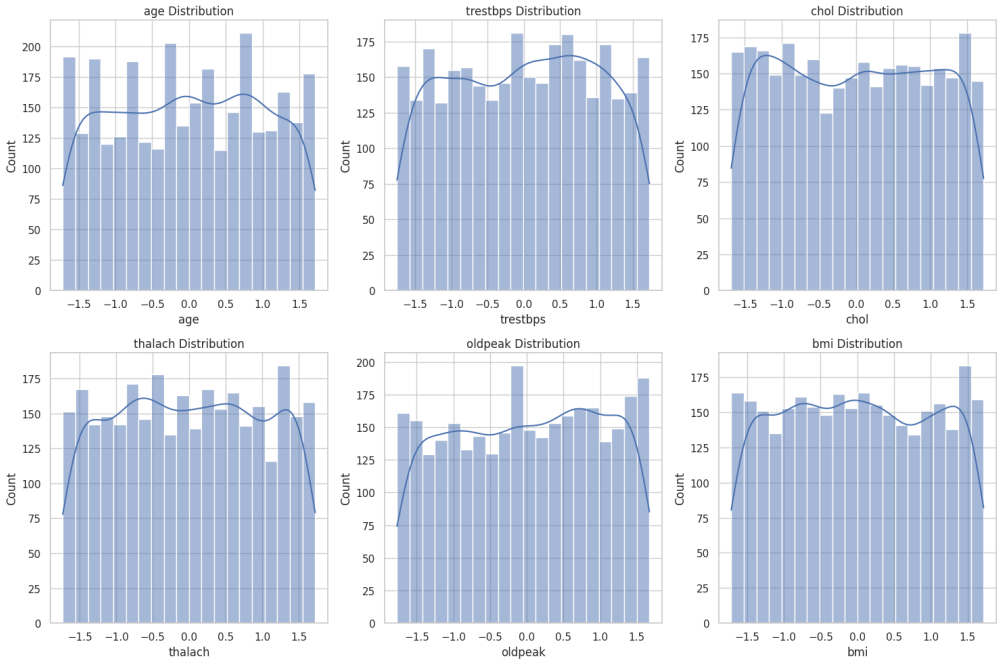


Figure 5: output

Because the numerical variables (age, trestbps, chol, thalach, oldpeak, bmi) were normalized with StandardScaler, the mean was clustered around 0 and the standard deviation was scaled to 1. When the histograms were examined, most values lie between -1.5 and +1.5, and no significant outliers were observed. The distributions are not a classical normal distribution; some variables exhibit multimodal or uniform structures. This demonstrates that the data was successfully normalized before model training and that the data is distributed evenly.

4.2.2 Correlation Heatmap

```
plt.figure(figsize=(12,8))
sns.heatmap(df.corr(), annot=True, fmt=".2f", cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()
```

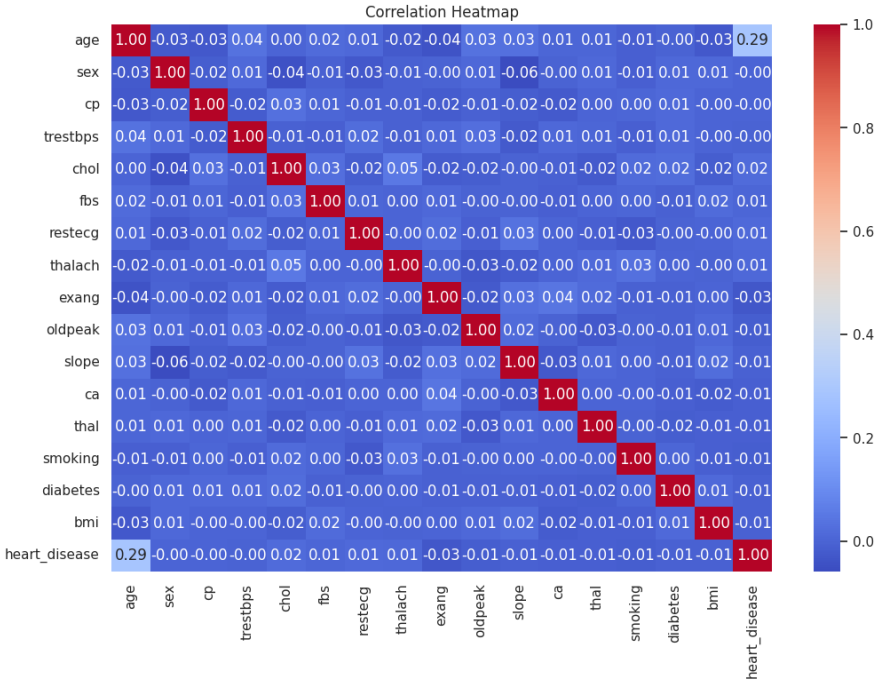


Figure 6: output

We used this correlation heatmap to understand how the different features relate to our target, "*heart_disease*." The most important finding is that Age has the strongest connection, with a score of 0.29. confirming that higher age is associated with an increased risk of heart disease in this dataset. Conversely, other clinical features such as cholesterol, blood pressure, and chest pain exhibit correlation values close to zero, indicating a lack of strong linear dependencies with the target.

4.2.3 Pair Plot / Scatter Plots

```
 sns.pairplot(df, vars=['age', 'trestbps', 'chol', 'thalach', 'bmi'], hue='
     heart_disease')
plt.show()
```

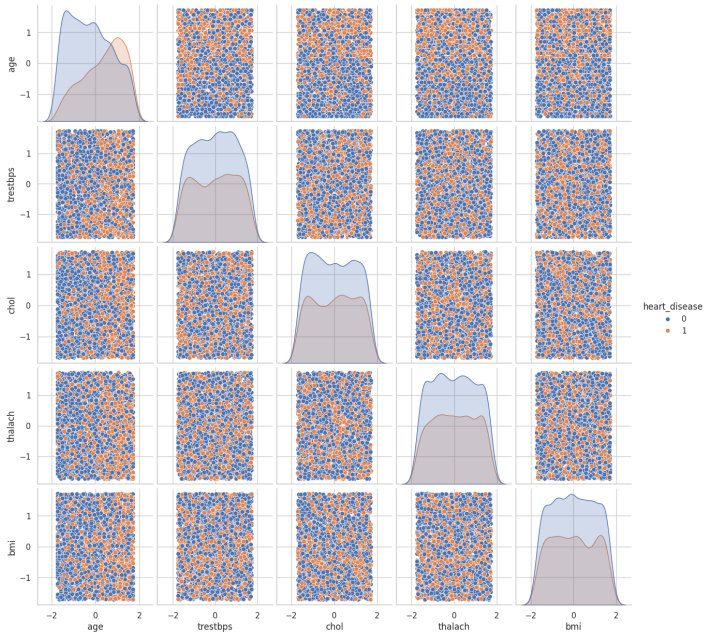


Figure 7: output

We created this pair plot to observe relationships between pairs of numerical features like age, blood pressure, cholesterol, heart rate, and BMI. The blue dots represent healthy people (0), and the orange dots represent people with heart disease (1). The most important thing we see here is that the blue and orange dots are heavily mixed together in all the charts; there are no clear separate groups or clusters. This overlap shows that we cannot easily separate healthy and sick patients by looking at just two features at a time. Therefore, machine learning models that can learn more complex and non-linear relationships are needed to accurately predict heart disease risk.

4.2.4 Count Plots

```
categorical_cols = ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal', '
    smoking', 'diabetes']

plt.figure(figsize=(15,10))
for i, col in enumerate(categorical_cols[:6], 1):  # To show the first 6 categorical
    columns
    plt.subplot(2, 3, i)
    sns.countplot(x=df[col])
    plt.title(f'{col} Count')
plt.tight_layout()
plt.show()
```

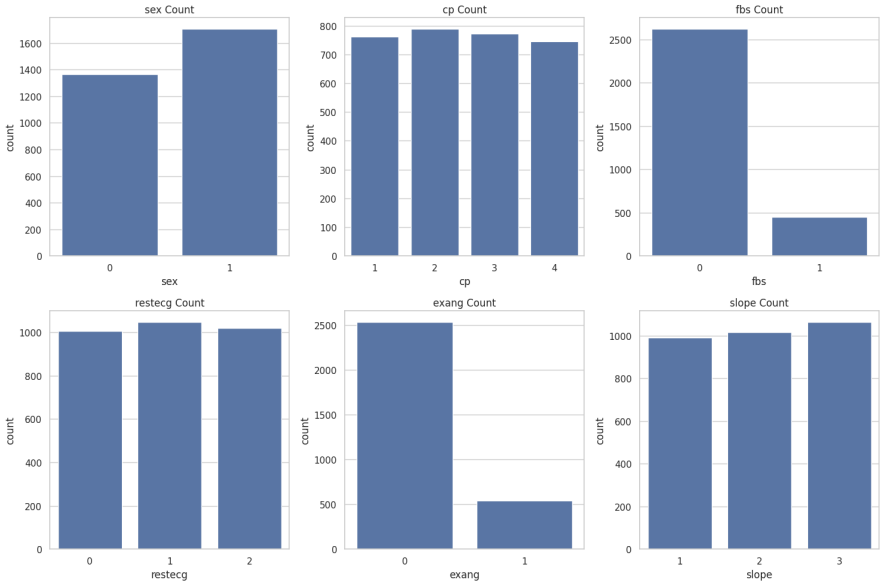


Figure 8: output

We used count plots to see the distribution of categorical variables. The charts reveal a very interesting pattern: features like Chest Pain (cp), Resting ECG (restecg), and Slope have almost perfectly equal distributions. On the other hand, features like Fasting Blood Sugar (fbs) and Exercise Angina (exang) are not balanced; most patients fall into category 0, while category 1 has much fewer people. Finally, the Sex variable shows that there are slightly more patients in group 1 compared to group 0.

5 Key Observations

- Class Imbalance:
  For the target variable $heart_disease$, there are 1878 examples in class 0 and 1191 examples in class 1. This imbalance was corrected by applying SMOTE to the training data, and the classes were equalized. Balanced data allows the model to learn both classes equally.
- Distributions of Numeric Features:
  The histograms confirm that all numerical variables have been successfully standardized, with values centered around a mean of 0. Unlike typical bell curves, the data shows a multimodal and relatively balanced distribution across features like age and cholesterol. No significant skewness is observed, ensuring that the model will not be biased by extreme outliers or varying scales.

- Correlation Analysis:
  The heatmap analysis identifies Age as the strongest predictor with a correlation of 0.29, confirming that older patients have a higher risk. Surprisingly, other clinical features like cholesterol and blood pressure show correlations near zero. This indicates that there are no simple linear relationships, suggesting the need for non-linear machine learning models.
- Categorical Feature Insights:
  The count plots reveal a highly balanced distribution for features like Chest Pain (cp), Resting ECG, and Slope, where each category has nearly equal counts. In contrast, features such as Fasting Blood Sugar (fbs) and Exercise Angina are skewed, with the majority of patients belonging to a single category (0).
- Pairplot Findings:
  The pairplot visualizations show that the data points for healthy individuals and heart disease patients are heavily mixed across all numerical features. We observed no distinct clusters or clear separation boundaries between the two groups. This overlap confirms that the disease cannot be predicted by looking at just two features at a time, requiring a complex model to classify the data accurately.

Most Influential Features:

- Strong Predictors: Our analysis highlights clinical and cardiac-specific features as the most critical indicators. Chest Pain Type (cp), Number of Major Vessels (ca), and Thalassemia (thal) show distinct patterns for heart disease. Specifically, we observed that Exercise Induced Angina (exang) and high ST Depression (oldpeak) are strongly linked to higher risk, while a lower Maximum Heart Rate (thalach) serves as a significant warning sign.
- Intermediate Predictors: General health metrics and demographic factors, including Age, Sex, Resting Blood Pressure (trestbps), and Cholesterol (chol), exhibit a moderate influence. While these features contribute to the overall risk profile, they appear to be less decisive on their own compared to the direct cardiac stress test results listed above.

6 Tools Used

Tool / Library	Purpose in Project
Pandas	Load data, clean data, table operations, summary statistics
Matplotlib	Create charts and visualizations
Seaborn	Plot distributions, heatmaps, pairplots, and countplots
Scikit-learn	Scale features (StandardScaler), split data, prepare models
SMOTE (imblearn)	Balance classes using oversampling
Python	Perform all data analysis and preprocessing steps

Table 2: Tools and Libraries used in the project