Cardiovascular Risk Detection using Machine Learning

Juan Francisco Rodríguez

August 30, 2024

Contact Information:

: jf.rdgzgal@gmail.com
: Juan Francisco
: @jfrg99

Contents

Chapter 1

Introduction

In this project, I will explore some risk factors for cardiovascular problems and apply some machine learning models to try to predict or detect the heart disease cases. The dataset for this work was obtained from the *Kaggle* website and can be found here.

Let's briefly discuss the problem. Cardiovascular diseases (CVD) are the leading cause of death worldwide, accounting for about 31% of all deaths. Of these, 4 out of 5 deaths are due to heart attacks and strokes, and one-third of these deaths occur prematurely in people under 70 years of age. People with cardiovascular disease or who are at high cardiovascular risk need early detection.

This dataset, although small, contains some of the most important features that can be used to predict possible heart disease.

This work (the pdf document, the notebook and the .csv datase) can be found in my github.

Chapter 2

Extrapolatory Data Analysis and Feature Engineering

1. Introduction

The first part of this project involves conducting an **Exploratory Data Analysis** (EDA) of our dataset and manipulating it to correct errors, normalize numerical values, and encode categorical variables.

All of this is done with the intention of later applying various machine learning classification models to predict the target variable, thereby creating a model capable of diagnosing whether a patient is at risk of heart failure.

2. Data Oveview

Before applying any model, it is important to perform exploratory data analysis to understand the nature of the data and make better use of machine learning. The features of our dataset are:

- **Age**: age of the patient in years.
- **Sex**: sex of the patient: male (M) or female (F).
- **Chest Pain Type**: Type of chest pain experienced by the patient. It can be:
  - **TA**: Typical Angina.
  - **ATA**: Atypical Angina.
  - **NAP**: Non-Anginal Pain.
  - **ASY**: Asymptomatic.

- **Resting BP**: Resting blood pressure, in $mmHg$. Normal resting blood pressure ranges from 98/60 mm/Hg to 120/80 $mmHg$.
- **Cholesterol**: serum cholesterol in mg/dl.
- **Fasting BS**: Fasting blood sugar level. It is 1 if its over 120 mg/dL; and is 0 otherwise. Normal fasting blood sugar levels are below 100 mg/dL.
- **Resting ECG**: Resting electrocardiogram results. It can be:
  - **Normal**: Normal EGC.
  - **ST**: Having ST-T wave abnormality.
  - **LVH**: Showing probable or definite left ventricular hypertrophy by Estes's criteria.
- **Max HR**: Maximum heart rate achieved during exercise. It is a numeric value typically from 60 to 202 beats per minute.
- **Exercise Angina**: Exercise-induced angina, yes (Y) or no (N).
- **Oldepeak**: Numeric value of the ST depression induced by exercise relative to rest. It indicates the severity induced by exercise relative to rest. It indicates the severity of ischemia.
- **ST Slope**: The slope of the peak exercise ST segment. It can be:
  - **Up**: Upsloping.
  - **Flat**: Flat.
  - **Down**: Downsloping.
- **Heart Disease**: Output class: 1 if the patient has a heart disease, and 0 if the patient has a normal health.

The dataset has a total of 918 entries, representing 918 patients. There are 11 features and 1 target variable, all with 918 non-null values, indicating that the dataset is completely clean and appears to have no missing values.

```
RangeIndex: 918 entries, 0 to 917
Data columns (total 12 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Age     918 non-null    int64
 1   Sex     918 non-null    object
 2   ChestPainType   918 non-null    object
 3   RestingBP       918 non-null    int64
 4   Cholesterol     918 non-null    int64
 5   FastingBS       918 non-null    int64
 6   RestingECG      918 non-null    object
 7   MaxHR   918 non-null    int64
```

```
  8   ExerciseAngina  918 non-null    object
  9   Oldpeak    918 non-null    float64
  10  ST_Slope   918 non-null    object
  11  HeartDisease    918 non-null    int64
dtypes: float64(1), int64(6), object(5)
```

We can distinguish the variables as follows:

- **Numerical feautures**: Age, RestingBP, Cholesterol, MaxHR and Oldpeak.
- **Categorical feautures**: Sex, ChestPainType, RestingECG, ExerciseAngina, STSlope and FastingBS.
- **Target variable**: Heart Disease (categorical).

We can get some statistics of the numerical variables:

	Age	**RestingBP**	**Cholesterol**	**MaxHR**	**Oldpeak**
count	918.0	918.0	918.0	918.0	918.0
mean	53.51	132.4	198.8	136.81	0.89
std	9.43	18.51	109.38	25.46	1.07
min	28.0	0.0	0.0	60.0	-2.6
25%	47.0	120.0	173.25	120.0	0.0
50%	54.0	130.0	223.0	138.0	0.6
75%	60.0	140.0	267.0	156.0	1.5
max	77.0	200.0	603.0	202.0	6.2

The first strange thing we see is that Cholesterol and RestingBP has a minimum of 0, which is incompatible with life. So let's see it more in details studying the outliers of the data.

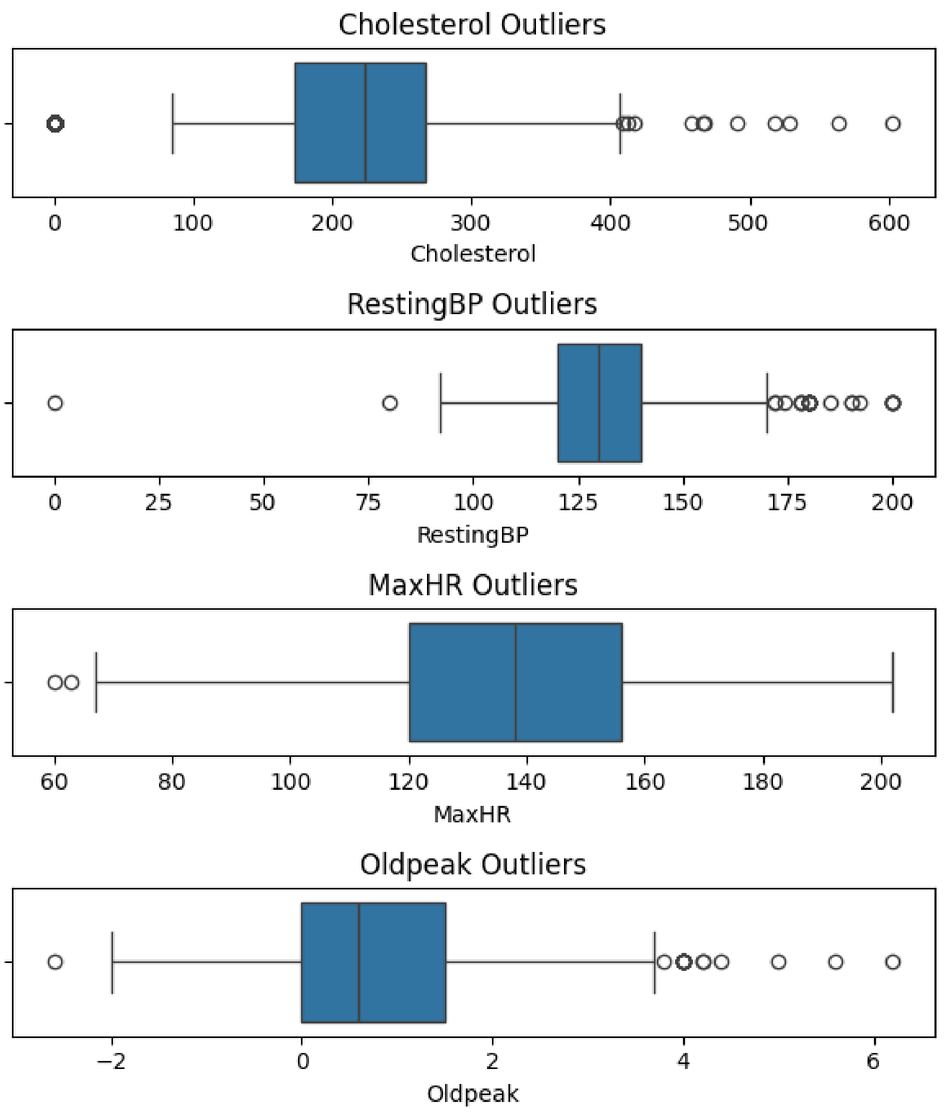
3. Feature Engineering

3.1. Handling Problematic Outliers

We can find the outliers using the interquartile ranges. The total number of outliers are:

Feature	**Outliers_Count**	**Lower_Limit**	**Upper_Limit**
Age	0	27.5	79.5
RestingBP	28	90.0	170.0
Cholesterol	183	32.625	407.625
MaxHR	2	66.0	210.0
Oldpeak	16	-2.25	3.75

We can visualize the using box-plots:



Values upper the limits are plausibles, but the value 0 for Cholesterol and Resting Blood Pressure are not. These values are:

- 172 patients with 0 cholesterol, of which 152 have heart problems.
- Only 1 patient with 0 resting blood pressure.

There are several ways to handle these values:

- **Drop incorrect values**: This is not the best idea in our case because they represent 16.5% of the total dataset, and dropping them would result in a significant loss of other features.

- **Predict the values using some regression models**: I prefer not to do this because the idea of this project is to apply classification methods. Applying a machine learning model over features that were also predicted using the same features could skew the results. If we did not want to apply classification, this would be the best method.
- **Replace them using a general statistic of the rest of the dataset**: This is the best option in my opinion. We can consider the median instead of the mean to reduce the impact on variance, as some ML models are sensitive to it.

Also, as there are more presence of patients with heart disease, I think it would be convenient to assign the median according to this variable to have values that are more in line with reality. So,

- if a patient has these values to 0 and has a heart disease, we replace them using the median of the rest of the patients with a heart disease.
- if a patient has these values to 0 but does not have a heart disease, we replace them using the median of the rest of the patients with no heart disease.

After impute this changes, the summary of the new dataset is:

	Age	**RestingBP**	**Cholesterol**	**MaxHR**	**Oldpeak**
count	918.0	918.0	918.0	918.0	918.0
mean	53.51	132.54	244.58	136.81	0.89
std	9.43	17.99	53.36	25.46	1.07
min	28.0	80.0	85.0	60.0	-2.6
25%	47.0	120.0	214.0	120.0	0.0
50%	54.0	130.0	246.0	138.0	0.6
75%	60.0	140.0	267.0	156.0	1.5
max	77.0	200.0	603.0	202.0	6.2

An interesting insight is that the outliers (excluding the problematic ones seen before) have a high representation of heart disease patients:

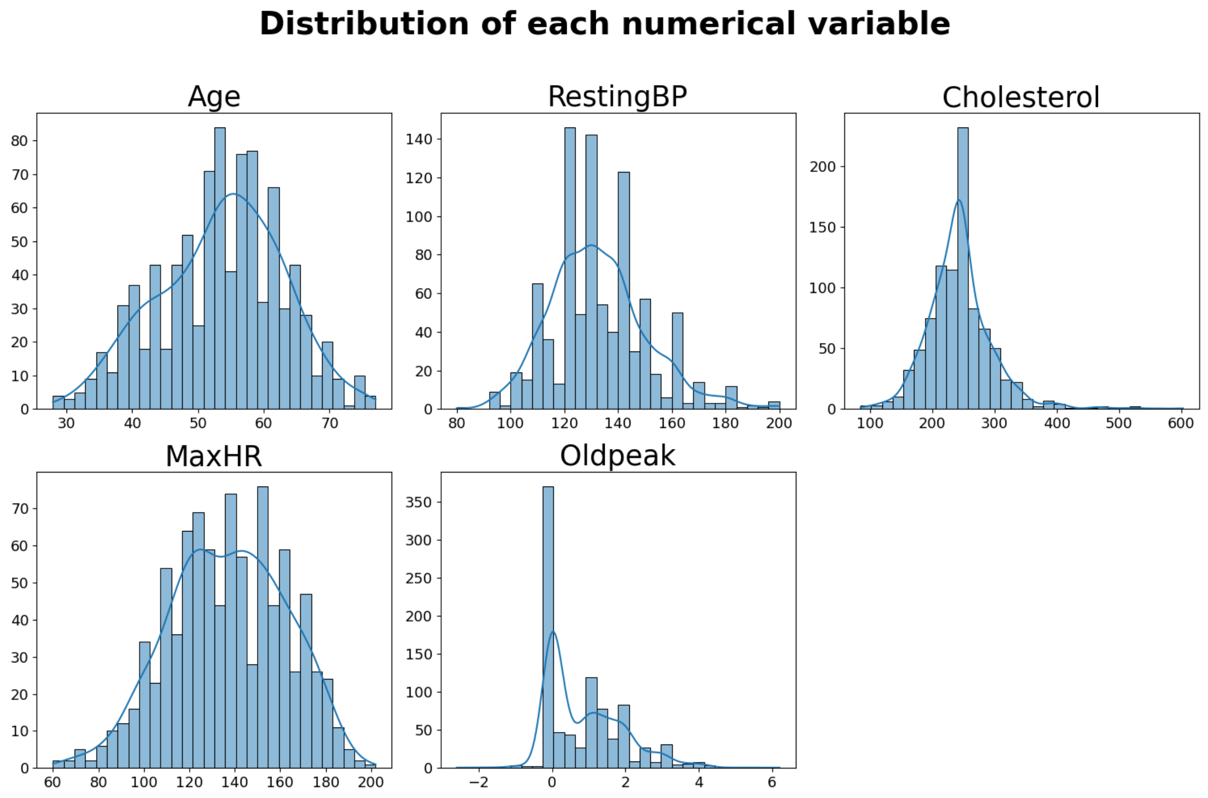
- The proportion of outliers with heart diseases in the Cholesterol variable is 58.33%.
- The proportion of outliers with heart diseases in the RetingBP variable is 68.29%
- The proportion of outliers (only 2) with heart diseases in the MaxRP variable is 100%.
- The proportion of outliers with heart diseases in the Oldpeak variable is 93.75%.

Note: These insights are obtained before data imputation. If we calculate the outliers afterwards, they will be different.

3.2. Scaling Numerical Variables

Normalizing data is an important step in applying machine learning as some alogorithms are based on distances. The library `sklearn` provides two main scalers: `MinMaxScaler()` and `StandardScaler()`. The first one scales the data to the $[0, 1]$ interval, while the second one scales the data centering it to the mean, making it more suitable for data that follows a normal distribution. We need to determine which scaler will work better with our data.

If we see the distribution of the values, we could think they follow a normal distribution:

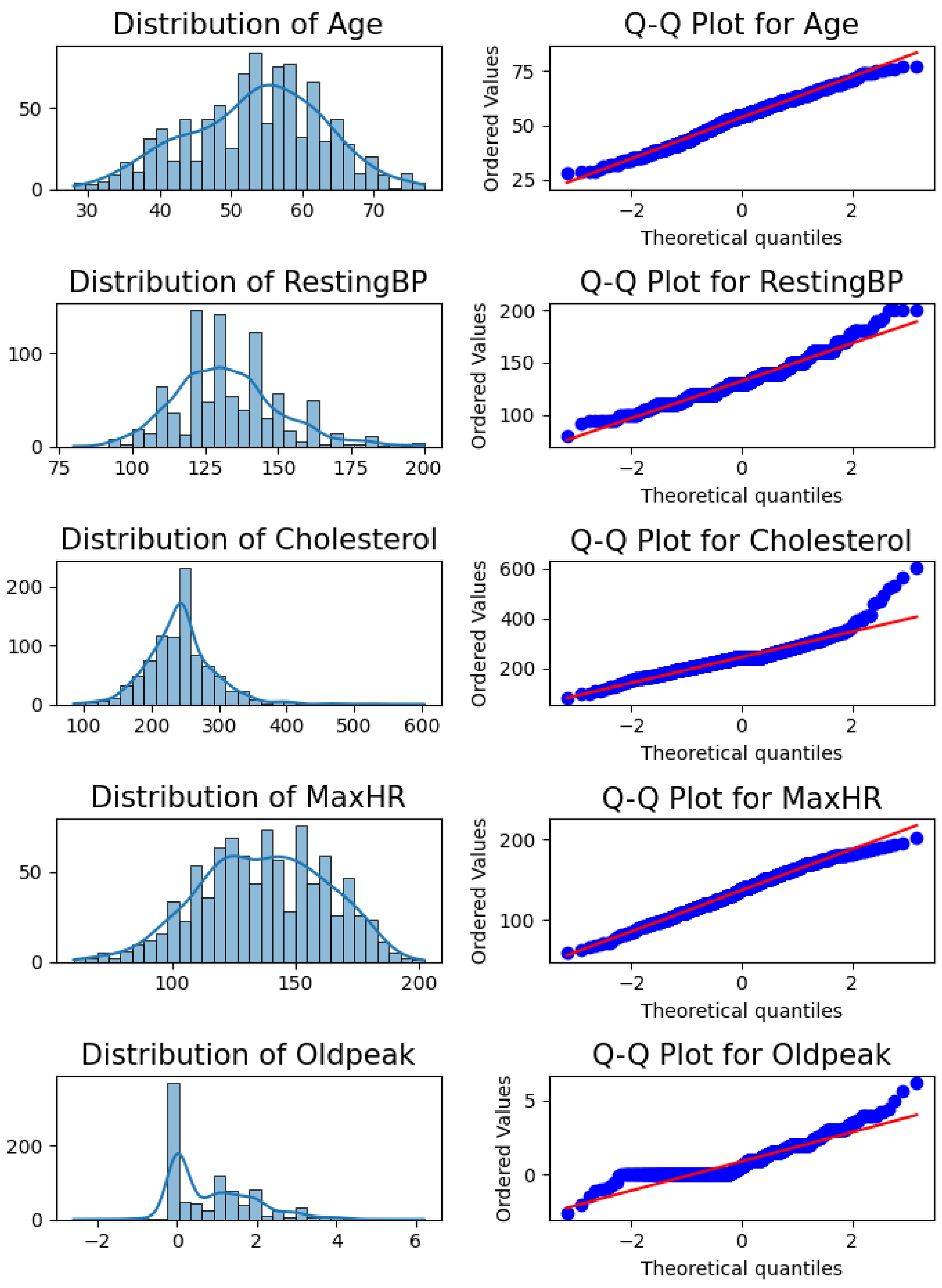


However, there are some quantitative ways to measure if a distribution is a normal distribution such as **Shapiro-Wilk**, **Kolmogorov-Smirnov** and **Anderson-Darling** tests. Also we can consider other metrics such as **Skewness** and **Kurtosis**. The explanation of each one is in the EDA notebook. The results of these tests for a significance level of $\alpha = 0.01$ and metrics are:

Variable	Skewness	Kurtosis	S-W	K-S	A-D
Age	-0.19593	-0.38614	No	No	No
RestingBP	0.60796	0.7962	No	No	No
Cholesterol	1.37352	6.23764	No	No	No
MaxHR	-0.14436	-0.44825	No	Yes	No
Oldpeak	1.02287	1.20306	No	No	No

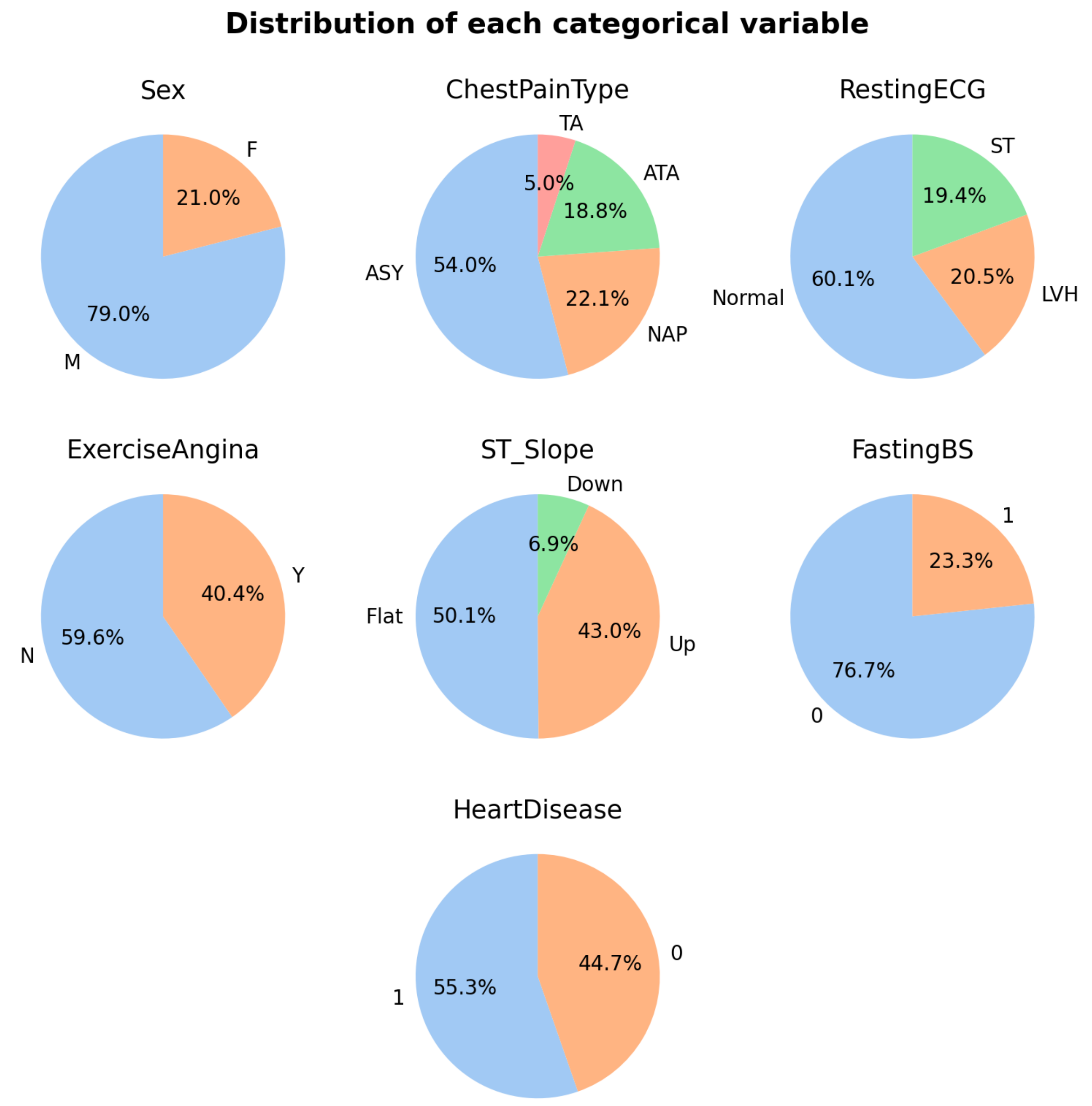
All tests reject the hypothesis that the data follow normal distributions, excepting the Kolmogorov tests, that accepts that MaxHR follows a normal distribution. Also the skewness and kurtosis values also highlight this difference from a normal distribution. Then, we will apply `MinMaxScaler()` to the dataset.

We can also use the Q-Q Plot technique, which visually compares a distribution of points with the normal distribution.



3.3. Encoding Categorical Variables

In order to compare categorical features with the numerical ones and apply machine learning, is necesary to transform them into numerical values. The library 'sklearn' provides many encoders based on different criteria. It's important to understand the nature of this categories.



We can see that:

- HeartDisease and FastingBS are already numerical.
- Sex and ExerciseAngina are binary and cannot be ordered, so we can use `OneHotEncoder`.

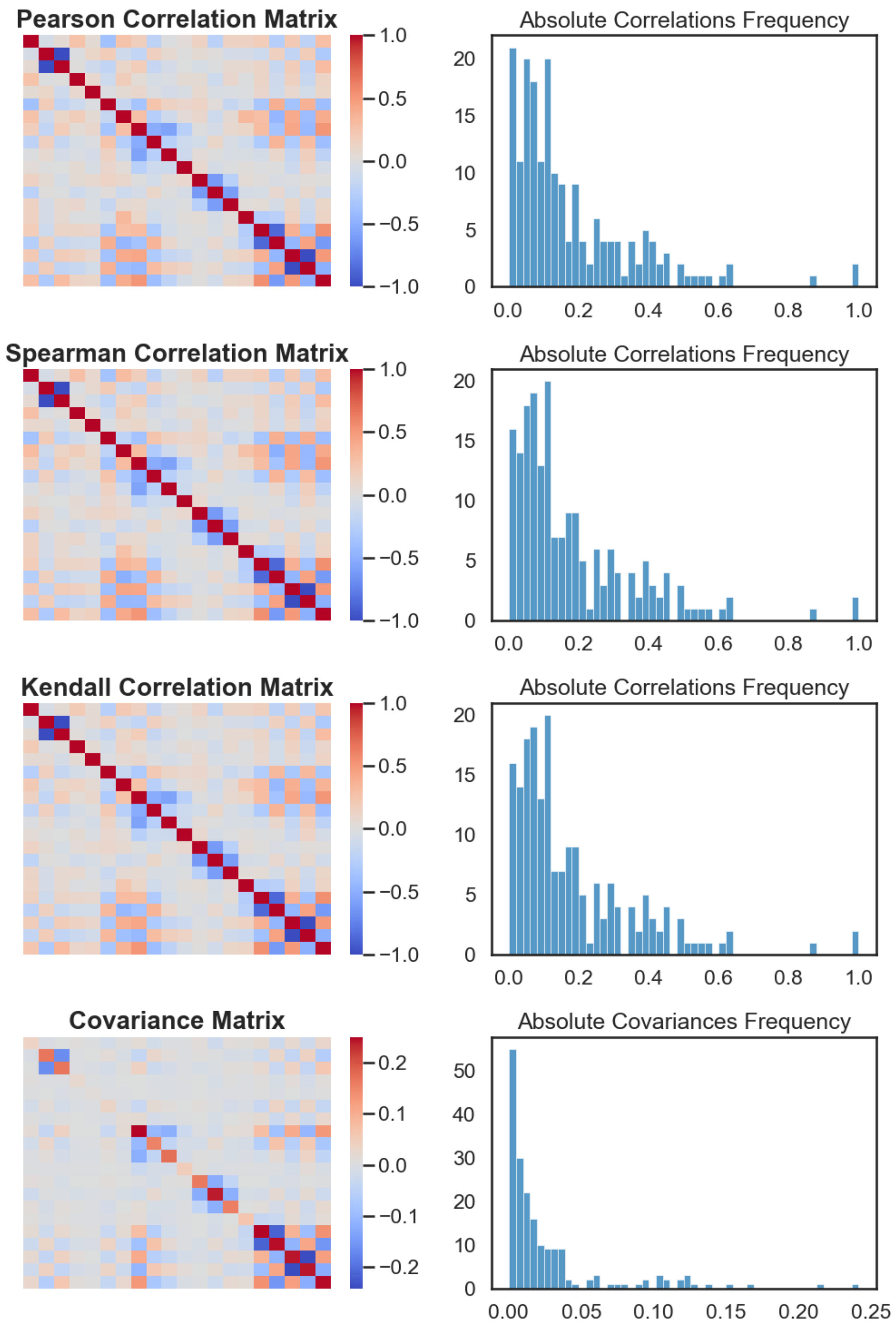
- The features ChestPainType and RestingECG have 4 and 3 categories respectively, but they can't be ordered, so it is not a good idea to encode them using numerical values $0, 1, 2, \ldots$; the best way to proceed is using `OneHotEncoder` again, but it will create new features, one for each category.
- Finally, `ST_Slope` has more than 2 categories. At first, I thought the could be sorted as `Down` < `Flat` < `Up`. But actually it does not rights because, in this context, a flat ST Slope correspond to a health person and both down and up ST Slope indicate a heart disease. So the best encoder is again `OneHotEncoder`.

After apply the scaling and the encoding, out dataset has the form:

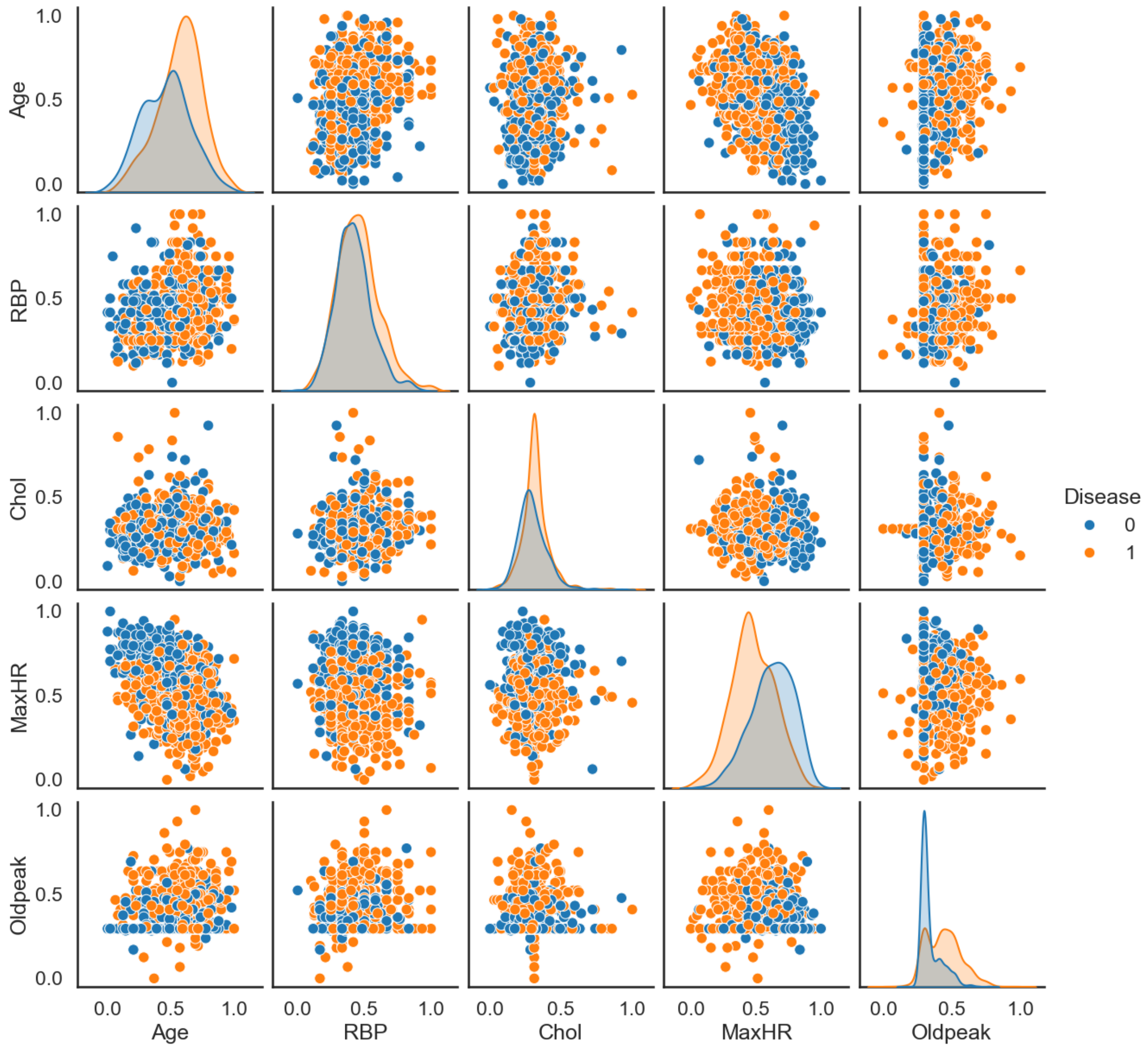
Patient	**0**	**1**	**2**	**3**	**4**	...
Age	0.2449	0.42857	0.18367	0.40816	0.53061	...
F	0.0	1.0	0.0	1.0	0.0	...
M	1.0	0.0	1.0	0.0	1.0	...
RBP	0.5	0.66667	0.41667	0.48333	0.58333	...
Chol	0.39382	0.1834	0.38224	0.24903	0.21236	...
MaxHR	0.78873	0.67606	0.26761	0.33803	0.43662	...
Oldpeak	0.29545	0.40909	0.29545	0.46591	0.29545	...
CPT_ASY	0.0	0.0	0.0	1.0	0.0	...
CPT_ATA	1.0	0.0	1.0	0.0	0.0	...
CPT_NAP	0.0	1.0	0.0	0.0	1.0	...
CPT_TA	0.0	0.0	0.0	0.0	0.0	...
RECG_LVH	0.0	0.0	0.0	0.0	0.0	...
RECG_N	1.0	1.0	0.0	1.0	1.0	...
RECG_ST	0.0	0.0	1.0	0.0	0.0	...
STS_D	0.0	0.0	0.0	0.0	0.0	...
STS_F	0.0	1.0	0.0	1.0	0.0	...
STS_U	1.0	0.0	1.0	0.0	1.0	...
EAY	0.0	0.0	0.0	1.0	0.0	...
EAN	1.0	1.0	1.0	0.0	1.0	...
Disease	0.0	1.0	0.0	1.0	0.0	...

4. Dependency Between Features

There are many ways to measure the dependency between variables. The best known is calculating the linear correlation between the features (Pearson), we can use different criteria of correlation such as Spearman or Kendall. We can also get the covariance relationships between the features.



We can also plot the dispersion of each pair of numerical features to see if there are correlations between them.

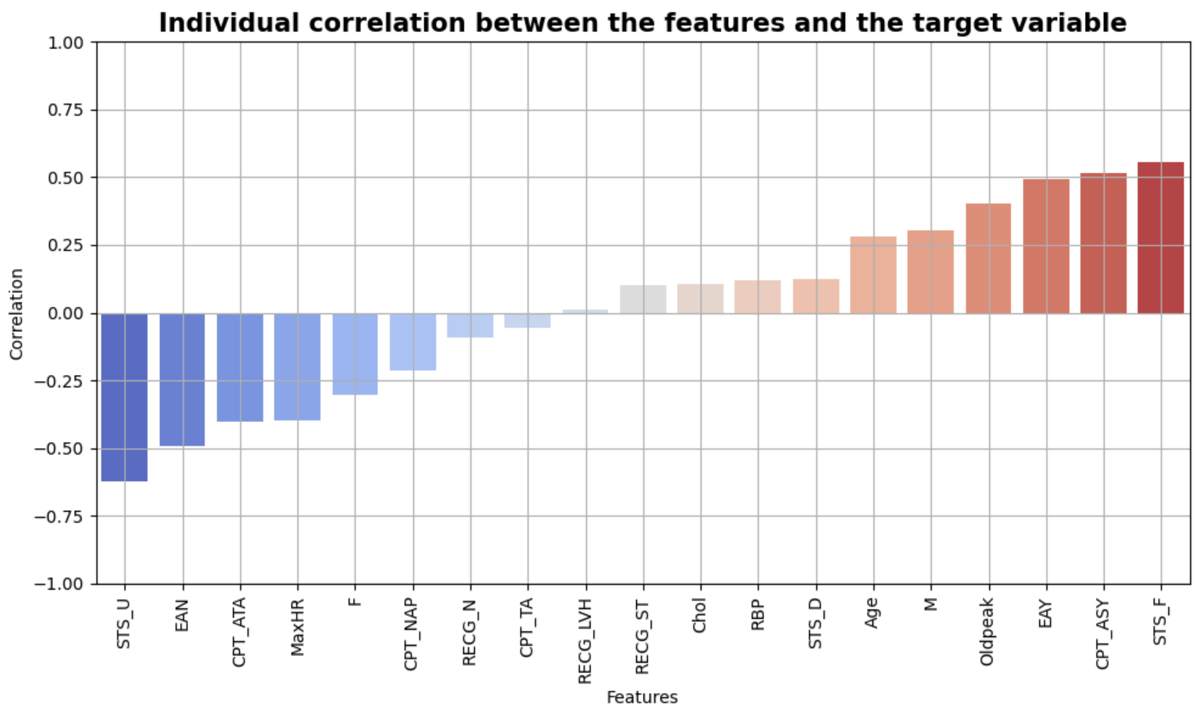


Conclusion

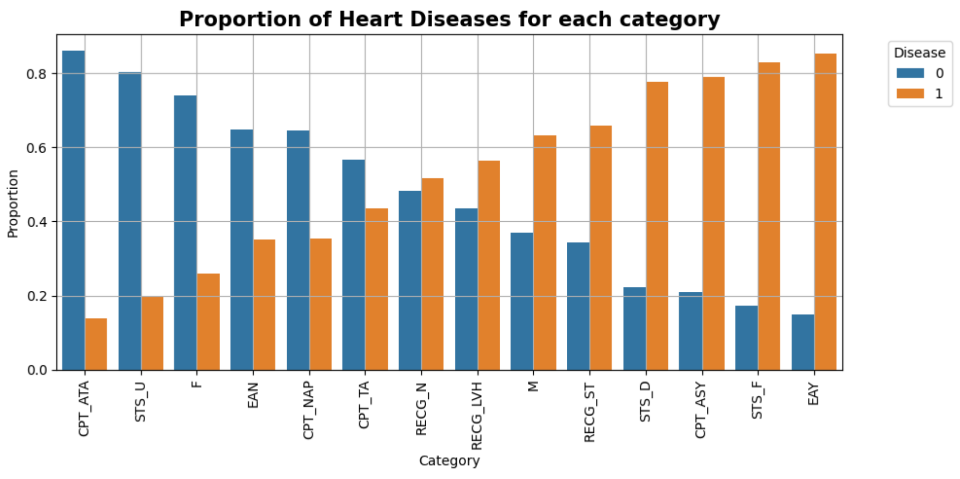
- Based on the **correlation** matrices and the **covariance** matrix we can say that the variables in our dataset are independent. This independence is beneficial for our machine learning models, as it reduces the risk of multicollinearity and ensures that each feature contributes unique information to the model.
- If we look at the **dispersion** pairplot, we can see that features are not correlated. But we can see that there is a clear grouping of Heart Disease classes 0 and 1. This is also a good signal since it indicates the numerical variables has influence on the target variable.

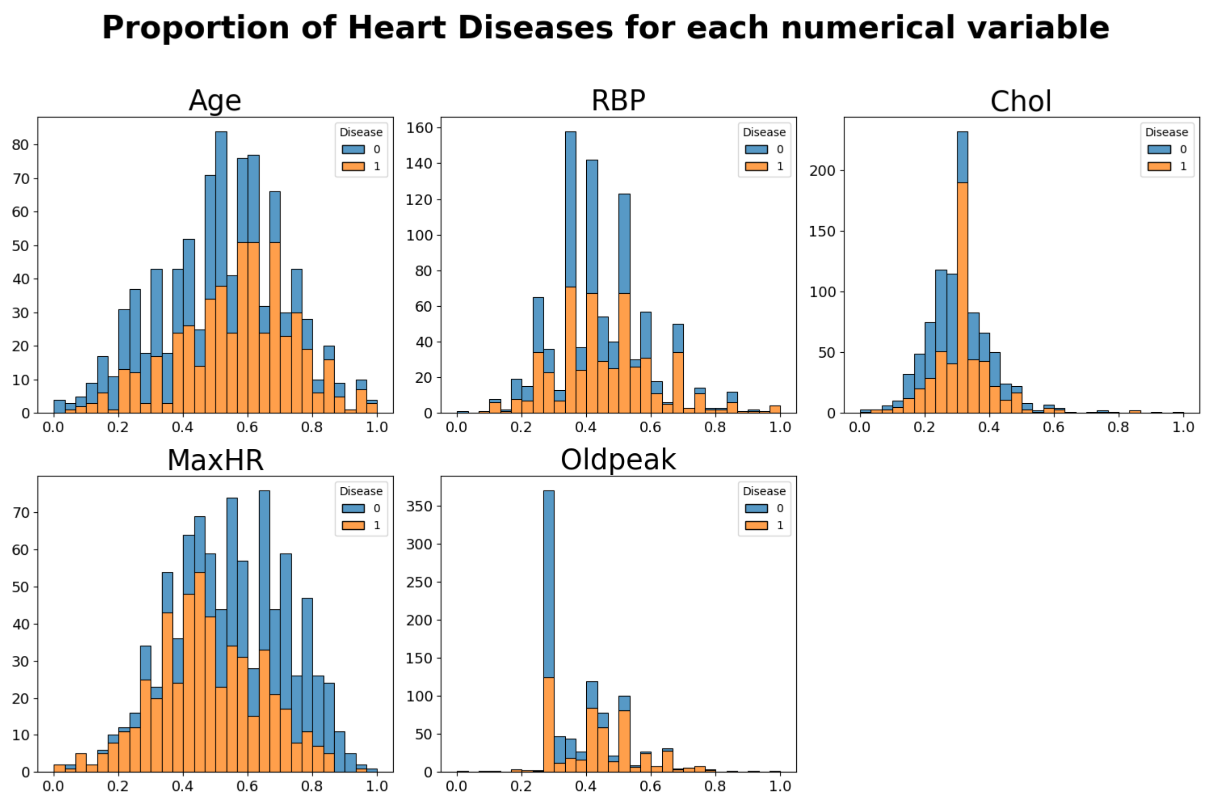
5. Relationships with the Target Variable

In the same way, we can get the linear correlation between each feature and the target variable.



We could see also the proportion of each feature that has heart diesase.





Conclusions

If we look at the *Individual correlation between the features and the target variable* and *Proportion of Heart Diseases for each category* charts, the influence of some categories to the heart disease is clear:

- Features with a **negative correlation** with the target variable has a greater proportion of **class 0** in heart disease: STS_U, EAN, CPT_ATA, F, CPT_NAP.
- Features with a **correlation near to 0** have a similar proportion of 0 and 1: RECG_N, CPT_TA, RECG_LVH.
- Features with a **positive correlation** have a greater proportion of **class 1** in heart disease: M, EAY, CPT_ASY, STS_F.
- Other cases don't follow these tendencies. RECG_ST and STS_D have a (positive) correlation near to 0, and they present greater proportion of the class 1 in heart disease.

Looking at the *Proportion of Heart Diseases for each numerical variable* chart, the relationships are not so clear.

- In MaxHR, there is a greater proportion of patients with no heart disease near to 1.
- In Age, there are more patients older with heart diesases.

Chapter 3

Supervised Machine Learning for Patient Classification

1. Introduction

The second part of this project involves applying various **supervised machine learning classification** models to our preprocessed dataset to predict the target variable, thereby creating a model capable of diagnosing whether a patient is at risk of heart failure.

We will explore and compare the performance of the following classification models: Logistic Regression, K Nearest Neighbors (KNN), Support Vector Machines (SVM) and Decision Trees.

For each kind of model, we will perform hyperparameter tuning to optimize their performance and select the best parameters using their F_1-Score. Then, we will train the best models.

Finally, we will compare the best Logistic Regression, KNN, SVM and Decision Trees models using various evaluation metrics: accuracy, precision, recall, F_1-score, AUC, Log Loss and Cohen's Kappa.

2. Methodology

There are several ways to proceed when applying machine learning models. I will use a cross-validation method to find the best hyperparameters for each model. Specifically, I will use `GridSearchCV`, which automates this process. By default, it implements a `StratifiedKFold` data split. This will generate 5 splits of the data, ensuring that the proportion of the classes in the target variable y is maintained.

Then, we will use `train_test_split`, specifying the `stratify` parameter to the target variable y, for train the final model and get the predictions. This split will be $70 - 30\%$.

3. Some SML models

3.1. Logistic Regression

For Logistic Regression, the we are trying to find the best parameters of the set:

```
param_grid_log = [
    {'penalty': ['l1'], 'C': [0.01, 0.1, 1, 10, 100]},
    {'penalty': ['l2'], 'C': [0.01, 0.1, 1, 10, 100]},
    {'penalty': ['elasticnet'],
     'C': [0.01, 0.1, 1, 10, 100],
     'l1_ratio': [0.1, 0.3, 0.5, 0.7, 0.9]} # l1_ratio is exclusive
                                            # to elasticnet
]
```

The best combination of parameter is

```
best_params_log = {
    'C': 0.1,
    'l1_ratio': 0.9,
    'penalty': 'elasticnet'
}
```

`elasticnet` as penalty with a `l1_ratio` of 0.9. The regularization parameter is `C=0.1`.

3.2. K Nearest Neighbors

For KNN, the we are trying to find the best parameters of the set:

```
param_grid_knn = {
    'n_neighbors': [3, 5, 7, 9, 11],
    'weights': ['uniform', 'distance'],
    'metric': ['euclidean', 'manhattan']
}
```

The best combination of parameter is

```
best_params_knn = {
    'metric': 'manhattan',
    'n_neighbors': 11,
    'weights': 'uniform'
}
```

3.3. Support Vector Machines

For SVM, the we are trying to find the best parameters of the set:

```
param_grid_svc = {
    'C': [0.1, 1, 10, 100],
    'kernel': ['linear', 'poly', 'rbf', 'sigmoid'],
    'gamma': ['scale', 'auto'],
    'degree': [2, 3, 4],
    'coef0': [0.0, 0.1, 0.5]
}
```

The best combination of parameter is

```
best_params_svm = {
    'C': 0.1,
    'coef0': 0.1,
    'degree': 2,
    'gamma': 'auto',
    'kernel': 'sigmoid'}
```

3.4. Decision Trees

For Decision Trees, the we are trying to find the best parameters of the set:

```
param_grid_dt = {
    'criterion': ['gini'],
    'max_depth':range(1, 9, 1),
    'max_features': range(1,  18)
    }
```

The best combination of parameter is

```
best_params_dt =  {
    'max_depth': 3,
    'max_features': 15
}
```

4. Model Comparison

5. The Best Model in Detail

6. Conclusion