

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA
Pós-graduação *Lato Sensu* em Inteligência Artificial e Aprendizado de Máquina

Vinício Silva Lima

PREDIÇÃO DE INSUFICIÊNCIA CARDÍACA

Belo Horizonte
Outubro de 2022

Vinício Silva Lima

PREDIÇÃO DE INSUFICIÊNCIA CARDÍACA

Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Inteligência Artificial e Aprendizado de Máquina, como requisito parcial à obtenção do título de *Especialista*.

Belo Horizonte
Outubro de 2022

SUMÁRIO

1. Introdução	4
2. Descrição do Problema e da Solução Proposta	5
3. Canvas Analítico	6
4. Coleta de Dados	6
5. Processamento/Tratamento de Dados	8
6. Análise e Exploração dos Dados	12
7. Preparação dos Dados para os Modelos de Aprendizado de Máquina	23
8. Aplicação de Modelos de Aprendizado de Máquina	25
8.1 Apresentação do Pipeline do Processo	26
9. Discussão dos Resultados	29
9.1 Matriz de confusão	29
9.2 Recall	30
9.3 Precision	30
9.4 F1-Score	30
9.5 Resultados dos modelos preditivos	31
10. Conclusão	32
11. Links	33
12. Referências	33

1. Introdução

O ser humano tem a habilidade de aprender com os problemas e tomar decisões a partir de experiências já vividas. Quando nos referimos a computação, a Inteligência Artificial consegue, por meio de algoritmos complexos de Aprendizado de Máquina, encontrar padrões matemáticos para inferir resultados, de acordo com a necessidade das pessoas. Hoje, com o avanço da Inteligência Artificial, é possível realizar predições se um paciente tem uma pré-disposição a sofrer um ataque cardiovascular baseando-se em dados, tais como: idade, pressão arterial, colesterol, entre outros dados. Apesar disso, por incrível que pareça, ainda sofremos frequentemente com baixas relacionadas a problemas de insuficiência cardíaca, até mesmo em países mais desenvolvidos.

A insuficiência cardíaca aflige ou mata um em cada dois adultos nos Estados Unidos e em outros países desenvolvidos. (Go AS, Mozaffarian D, Roger VL, et al. 2013). Uma das causas dos ataques cardiovasculares é o acúmulo de placas de colesterol nas paredes internas de artérias, conhecido como aterosclerose. Normalmente o desenvolvimento da doença é silencioso e pode desenvolver-se na adolescência ou até mesmo na infância. Quando começa a apresentar sintomas, causa listras esbranquiçadas no revestimento interno das artérias que, com o passar do tempo, transformam-se em bolsas de colesterol que podem inchar-se, dificultando a passagem do fluxo sanguíneo na parede arterial, causando desconforto no peito. Em casos de rompimento das placas de colesterol, pode causar coágulos que, se muito grandes, podem impedir o fluxo sanguíneo e resultar em um ataque cardíaco ou um derrame. (Harvard T.H Chan School of Public Health, 2022).

A prática de atividade física regular reduz o risco de doenças cardíacas, diabetes, acidente vascular cerebral, pressão alta, osteoporose e até mesmo alguns tipos de câncer. (Harvard T.H Chan School of Public Health, 2022). Apesar disso, à medida que vamos envelhecendo, temos uma diminuição de atividade física, nos deixando mais expostos a doenças cardiovasculares (Matthews CE, George SM, Moore SC, et al. 2012).

Com base nisso, entendendo a importância deste tema, este trabalho tem como objetivo aplicar técnicas de Aprendizado de Máquina em uma base de dados de Insuficiência Cardíaca encontrada na plataforma de desafios de Ciência de Dados *Kaggle*. Por meio de modelos de classificação, podemos prever se os pacientes encontrados no *dataset* possuem insuficiência cardíaca ou não, apresentando todo o

processo de desenvolvimento do estudo como um todo e, desta forma, oferecer métricas de classificação efetivas para avaliar os modelos desenvolvidos.

2. Descrição do Problema e da Solução Proposta

Doenças cardiovasculares levam cerca de 17,9 milhões de vidas a cada ano, representando 31% de todas as mortes em todo mundo. Grande parte desses ataques de insuficiência cardíaca estão relacionados a problemas de acúmulo de placas de colesterol nas artérias. Independentemente do tamanho deste armazenamento de colesterol nas artérias, esses pacientes estão sujeitos a rompimentos inesperados nesses vasos, causando dores no peito e até mesmo derrames.

Não existe uma idade específica para um paciente ter o risco de sofrer de um ataque cardíaco. Infelizmente, muitas dessas vítimas sofrem de hipertensão, diabetes ou até mesmo hiperlipidemia.

Para evitar mais baixas de tantas pessoas, é importante que o diagnóstico de pacientes propensos a sofrerem de ataques cardíacos seja feito com antecedência. Neste requisito, modelos de Aprendizado de Máquina podem apoiar na detecção precoce de possíveis vítimas de doenças cardiovasculares, já que conseguem aprender com os padrões encontrados nas bases de dados e predizer se o paciente corre risco de ter um ataque cardíaco ou não.

Para resolver problemas de classificação, os modelos são comumente de Árvores de Decisão ou de Regressão Logística, por exemplo. Com as predições feitas, os modelos são avaliados através de métricas de classificação (Recall, Precision, F1-Score, etc) e os resultados podem ser compartilhados com uma equipe de médicos, a fim de mostrar que a Inteligência Artificial pode ser uma grande aliada na prevenção de possíveis ataques cardiovasculares.

3. Canvas Analítico

Software Analytics Canvas

Project: Predição de Insuficiência Cardíaca








<div> 1. Question</div> <p><i>What is it that we want to know about the software / processes / usage / organization / etc.?</i></p> <ul style="list-style-type: none">• É possível prever se o paciente é propenso a sofrer uma parada cardiovascular com os dados disponíveis?• Será que os níveis de colesterol apresentam uma forte correlação em relação a variável de insuficiência cardíaca?• Será que pessoas que sofreram de insuficiência cardíaca apresentaram comportamentos semelhantes?• Qual será a faixa etária das pessoas mais afetadas por insuficiência cardíaca?	<div> 2. Data Sources</div> <p><i>Which data can possibly answer our question? What information do we need?</i></p> <ul style="list-style-type: none">• Dados relacionados aos resultados dos exames dos pacientes, tais como: Idade, Nível de Colesterol, Dor no Peito, Pressão Arterial, entre outras variáveis• Precisamos ter um dataset com dados consistentes para que o modelo treinado apresente bons resultados• Precisamos da nossa variável-alvo para realizamos inferências e treinarmos o modelo de Aprendizado de Máquina	<div> 3. Heuristics</div> <p><i>Which assumptions do we want to make to simplify the answer to our question?</i></p> <ul style="list-style-type: none">• Encontrar, por meio do estudo dos dados, características que possam resultar em casos de insuficiência cardíaca• Procurar possíveis correlações nos dados em relação a nossa variável-alvo• Documentar e realizar inferências estatísticas com base nos resultados encontrados	<div> 4. Validation</div> <p><i>What results do we expect from our analysis, how are they reviewed and presented in an understandable way?</i></p> <ul style="list-style-type: none">• É esperado uma base com dados consistentes para treinarmos o modelo da melhor forma possível e oferecermos predições relevantes• Os resultados serão analisados por meio métricas de classificação de modelos de Aprendizado de Máquina. Além disso, os resultados serão explicados de forma minuciosa na documentação do projeto, a fim de para facilitar o entendimento
<div> 5. Implementation</div> <p><i>How can we implement the analysis step by step and in a comprehensible way?</i></p> <p>Para realizar o estudo, será aplicada a metodologia do CRISP-DM (Cross Industry Standard Process for Data Mining). Essa metodologia é iterativa, então, apesar de termos as etapas definidas, existe a reflexibilidade de retornamos nas etapas de acordo com a necessidade do negócio e resultados.</p> <p>O CRISP-DM pode ser dividido em algumas etapas:</p> <ul style="list-style-type: none">• Business understanding: Parte de entendimento do negócio e do problema a ser resolvido• Data Understanding: Entendimento inicial dos dados para entendermos como os dados estão distribuídos e quais abordagens serão necessárias para desenvolver o projeto• Data Preparation: Tratamento dos dados e criação de novas variáveis, a fim de potencializar o desempenho do modelo de acordo com as necessidades do negócio• Modeling: Desenvolvimento de modelos de Aprendizado de Aprendizado de Máquina• Evaluation: Análise do desempenho dos modelos desenvolvidos. Nesta etapa, validamos se o modelo treinado é capaz de realizar predições relevantes para solução do problema• Deployment: Será feita a documentação dos resultados obtidos na pesquisa e as perguntas levantadas sendo respondidas baseadas nos estudos realizados	<div> 6. Results</div> <p><i>What are the main insights from our analysis?</i></p> <ul style="list-style-type: none">• Evidenciar que a Inteligência Artificial pode ser uma grande aliada na prevenção de casos de problemas cardiovasculares• Entender quais variáveis foram mais relevantes para prever possíveis ataques de insuficiência cardíaca• Entender se existe correlação entre as variáveis estudadas e nossa variável-alvo	<div> 7. Next Steps</div> <p><i>What follow-up actions can we derive from the findings? Who or what do we need to address next?</i></p> <ul style="list-style-type: none">• Como próximos passos, podemos solicitar novas bases de dados e entrar em contato com profissionais da área da saúde para ver como podemos melhorar mais o modelo preditivo e, como consequência, apoiá-los na prevenção de possíveis casos de insuficiência cardíaca	

Figura 1: Canvas Analítico

4. Coleta de Dados

A base de dados *Heart Failure Prediction Dataset* foi extraída na plataforma Kaggle por meio do seguinte link:

<https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction> . A extração foi feita no dia 20/06/2020.

A base é composta por *datasets* de pacientes dos Estados Unidos, Hungria, França e Suíça. Não foi fornecida a data de extração dos dados e a forma de como a base foi obtida.

Nome do dataset: Heart Failure Prediction Dataset

Descrição: *Dataset* com informações de pacientes americanos, suíços, franceses, húngaros com ou sem histórico de insuficiência cardíaca. O objetivo é analisar as variáveis e desenvolver um modelo capaz de realizar previsões de possíveis pacientes propensos a sofrer de insuficiência cardiovascular.

Data de publicação: 10/2022

Link: <https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction>

Nome do Atributo	Descrição	Tipo
Age	Idade do paciente	Inteiro
Sex	Sexo do paciente	String
ChestPainType	Tipo de dor no peito	String
Cholesterol	Colesterol total	Inteiro
ExerciseAngina	Dor no peito proveniente do exercício	String
FastingBS	Glicose no sangue	Inteiro
MaxHR	Máxima de batimento cardíaco	Inteiro
Oldpeak	Depressão ST induzida por exercício relativamente sossegado	Float
RestingECG	Conclusão do Eletrocardiograma	String
ST_Slope	Taxa de frequência cardíaca	String
HeartDisease	Insuficiência Cardíaca (variável-alvo)	Inteiro

Tabela 1: Descrição das variáveis do dataset

5. Processamento/Tratamento de Dados

Para realizar a análise, foi utilizado o *Jupyter Notebook* e as bibliotecas *pandas*, *numpy* e *category_encoders*

Import das bibliotecas

```
import pandas as pd
import numpy as np
from category_encoders.one_hot import OneHotEncoder
```

executed in 10.4s, finished 00:59:32 2022-06-15

Figura 2: import das bibliotecas

Para começar a análise, é preciso entender como os dados estão distribuídos, conforme imagem abaixo:

```
#Lendo csv
df = pd.read_csv('heart.csv')
#Verificando a quantidade de linhas, colunas, nulos e o tipo de cada variável
df.info()
##Verificando as 3 primeiras linhas do dataframe
df.head(3)
```

executed in 110ms, finished 00:59:36 2022-06-15

```
<class 'pandas.core.frame.DataFrame'>
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)
memory usage: 86.2+ KB
```

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40	M	ATA	140	289	0	Normal	172	N	0.0	Up	0
1	49	F	NAP	160	180	0	Normal	156	N	1.0	Flat	1
2	37	M	ATA	130	283	0	ST	98	N	0.0	Up	0

Figura 3: Leitura e entendimento inicial dos dados

Após importar o csv, é possível verificar que a base não possui variáveis nulas. Como existe atributos do tipo *object*, será preciso separar o *dataframe* em duas partes para trabalhar com as variáveis numéricas separadamente.

Separando as variáveis numéricas

```
#Selecionando as variáveis de formato numérico
atributos_numericos = df.select_dtypes(include=['int64', 'float64'])
#Verificando se as variáveis são numéricas
atributos_numericos.info()
```

executed in 16ms, finished 00:18:26 2022-06-15

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918 entries, 0 to 917
Data columns (total 7 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Age                    918 non-null   int64
1   RestingBP              918 non-null   int64
2   Cholesterol            918 non-null   int64
3   FastingBS              918 non-null   int64
4   MaxHR                  918 non-null   int64
5   Oldpeak                918 non-null   float64
6   HeartDisease           918 non-null   int64
dtypes: float64(1), int64(6)
memory usage: 50.3 KB
```

Figura 4: Separação das variáveis numéricas

Para separar as variáveis numéricas foi utilizada a função `select_dtypes` da biblioteca *pandas* e foi feito um filtro para selecionar as que possuem valores inteiros e *float*. Com o resultado atribuído à variável “atributos_numericos”, foi utilizada a função *info* para conferir os resultados.

Separando as variáveis em formato string e convertendo-as para formato categórico

```
#Selecionando as variáveis de formato object e transformando em categóricas
atributos_categoricos = df.select_dtypes(include=['object']).astype('category')
#Verificando se as variáveis são categóricas
atributos_categoricos.info()
```

executed in 27ms, finished 00:18:26 2022-06-15

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918 entries, 0 to 917
Data columns (total 5 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Sex                    918 non-null   category
1   ChestPainType          918 non-null   category
2   RestingECG             918 non-null   category
3   ExerciseAngina         918 non-null   category
4   ST_Slope               918 non-null   category
dtypes: category(5)
memory usage: 5.3 KB
```

Figura 5: Separação das variáveis categóricas

No caso das variáveis do tipo objeto foi feita uma conversão para formato categórico, pois existem atributos como “Sex” que apresenta resultados provenientes de categorias, como masculino e feminino. Após a atribuição do resultado à variável “atributos_categoricos”, foi realizada a checagem para conferir se o tipo das variáveis foram convertidos para *object*.

Aplicando métricas estatísticas para entender as variáveis numéricas

```
# métricas de tendência central
# transpor para facilitar a visualização
ct1 = pd.DataFrame(atributos_numericos.apply(np.mean)).T # média
ct2 = pd.DataFrame(atributos_numericos.apply(np.median)).T # mediana
# métricas de dispersão
d1 = pd.DataFrame(atributos_numericos.apply(np.std)).T # desvio-padrão
d2 = pd.DataFrame(atributos_numericos.apply(min)).T # mínimo
d3 = pd.DataFrame(atributos_numericos.apply(max)).T # máximo
d4 = pd.DataFrame(atributos_numericos.apply(lambda x: x.max() - x.min())).T # range
d5 = pd.DataFrame(atributos_numericos.apply(lambda x: x.skew())).T # assimetria
d6 = pd.DataFrame(atributos_numericos.apply(lambda x: x.kurtosis())).T # curtose
# concatenar as métricas
m = pd.concat([d2, d3, d4, ct1, ct2, d1, d5, d6]).T.reset_index() # métricas
# nomear as colunas dos atributos concatenados
m.columns = ['Atributos', 'Min', 'Max', 'Range', 'Média', 'Mediana', 'Desvio-Padrão', 'Assimetria',
            'Curtose']
#Mostra tabela com os resultados
m
```

executed in 66ms, finished 00:18:27 2022-06-15

	Atributos	Min	Max	Range	Média	Mediana	Desvio-Padrão	Assimetria	Curtose
0	Age	28.0	77.0	49.0	53.510893	54.0	9.427478	-0.195933	-0.386140
1	RestingBP	0.0	200.0	200.0	132.396514	130.0	18.504067	0.179839	3.271251
2	Cholesterol	0.0	603.0	603.0	198.799564	223.0	109.324551	-0.610086	0.118208
3	FastingBS	0.0	1.0	1.0	0.233115	0.0	0.422815	1.264484	-0.401960
4	MaxHR	60.0	202.0	142.0	136.809368	138.0	25.446463	-0.144359	-0.448248
5	Oldpeak	-2.6	6.2	8.8	0.887364	0.6	1.065989	1.022872	1.203064
6	HeartDisease	0.0	1.0	1.0	0.553377	1.0	0.497143	-0.215086	-1.958008

Figura 6: Métricas estatísticas para entendimento de variáveis numéricas

Utilizando a variável “atributos_numericos” foi feita a declaração de alguns atributos e adicionado medidas de dispersão para entender melhor os dados que estão sendo trabalhados. No caso, é possível observar que a variável de *Cholesterol* apresenta um desvio-padrão bem alto e um valor mínimo de 0, o que pode indicar possíveis valores inválidos na base.

Filtrando as colunas numéricas e entendendo um pouco a distribuição por quartil das variáveis

```
#Entendendo as variáveis numéricas
df.describe(include=['float64', 'int64'], percentiles=[0.01, .1, .25, .50, .75, .9, .99]).T
```

executed in 60ms, finished 00:18:27 2022-06-15

	count	mean	std	min	1%	10%	25%	50%	75%	90%	99%	max
Age	918.0	53.510893	9.432617	28.0	32.00	40.0	47.00	54.0	60.0	65.0	74.00	77.0
RestingBP	918.0	132.396514	18.514154	0.0	95.00	110.0	120.00	130.0	140.0	160.0	180.00	200.0
Cholesterol	918.0	198.799564	109.384145	0.0	0.00	0.0	173.25	223.0	267.0	305.0	411.49	603.0
FastingBS	918.0	0.233115	0.423046	0.0	0.00	0.0	0.00	0.0	0.0	1.0	1.00	1.0
MaxHR	918.0	136.809368	25.460334	60.0	77.17	103.0	120.00	138.0	156.0	170.0	186.00	202.0
Oldpeak	918.0	0.887364	1.066570	-2.6	-0.50	0.0	0.00	0.6	1.5	2.3	4.00	6.2
HeartDisease	918.0	0.553377	0.497414	0.0	0.00	0.0	0.00	1.0	1.0	1.0	1.00	1.0

Figura 7: Analisando as variáveis com quebras por quartil

Para nos aprofundar-se um pouco mais nas variáveis, foi aplicada a função *describe* com quebras por quartis para entender melhor como está a dispersão dos atributos.

Com base nisso, é possível observar que a variável de *Cholesterol* provavelmente possui *outliers*, já que o valor máximo encontrado está muito acima da média.

Concatenando os dados categoricos e numéricos

```
#Juntando as bases das variáveis numéricas e categóricas
df = pd.concat([atributos_categoricos, atributos_numericos], axis=1)
#Verificando as 3 primeiras linhas do dataframe
df.head(3)
```

executed in 33ms, finished 00:18:28 2022-06-15

	Sex	ChestPainType	RestingECG	ExerciseAngina	ST_Slope	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease
0	M	ATA	Normal	N	Up	40	140	289	0	172	0.0	0
1	F	NAP	Normal	N	Flat	49	160	180	0	156	1.0	1
2	M	ATA	ST	N	Up	37	130	283	0	98	0.0	0

Figura 8: Junção de dataframes

Utilizando as variáveis “atributos_categoricos” e “atributos_numericos” pode-se criar um novo *dataframe* concatenando as duas bases. Isso pode ser feito agora sem muitos impeditivos, já que o tipo dos atributos foram tratados anteriormente.

Filtrando colunas categoricas e aplicando o OneHotEncoder para classifica-las numericamente

```
#Filtrando as colunas categoricas
cols_encoding = df.select_dtypes(include='category').columns
#Passando para o OneHotEncoder as colunas categoricas
ohe = OneHotEncoder(cols=cols_encoding)
#Quebrando as variáveis categóricas em colunas numéricas
encoded = ohe.fit_transform(df)
#Verificando as 3 primeiras linhas do dataframe
encoded.head(3)
```

executed in 121ms, finished 00:18:28 2022-06-15

	Sex_1	Sex_2	ChestPainType_1	ChestPainType_2	ChestPainType_3	ChestPainType_4	RestingECG_1	RestingECG_2	RestingECG_3	ExerciseAngina_1	...	HeartDisease
0	1	0	1	0	0	0	1	0	0	1	...	0
1	0	1	0	1	0	0	1	0	0	1	...	1
2	1	0	1	0	0	0	0	1	0	1	...	0

3 rows x 21 columns

Figura 9: Aplicando o OneHotEncoder

Com o novo *dataframe*, faz-se necessário separar os valores categóricos novamente e atribuir a uma variável temporária chamada “cols_encoding”. Feito isso, é passado as variáveis para o *OneHotEncoder* e aplica-se a transformação com o método *fit*, quebrando as colunas de acordo com o número de categorias presentes em cada coluna. No caso de “ChestPainType”, existiam 4 categorias, então foram geradas quatro colunas distintas para separá-las.

6. Análise e Exploração dos Dados

Nesta etapa será dada continuidade a parte de análise descritiva das variáveis e entender melhor o comportamento das variáveis em relação ao *target*. Para facilitar um pouco, será feita distribuições de frequência de forma univariada e depois multivariada. Dependendo dos resultados, será aplicado testes de hipótese, a fim de confirmar ou entender se as distribuições são gaussianas, por exemplo. Desta forma, será possível entender quais tipos de tratamentos serão necessários realizar e quais variáveis são mais adequadas para realizar um modelo preditivo mais assertivo.

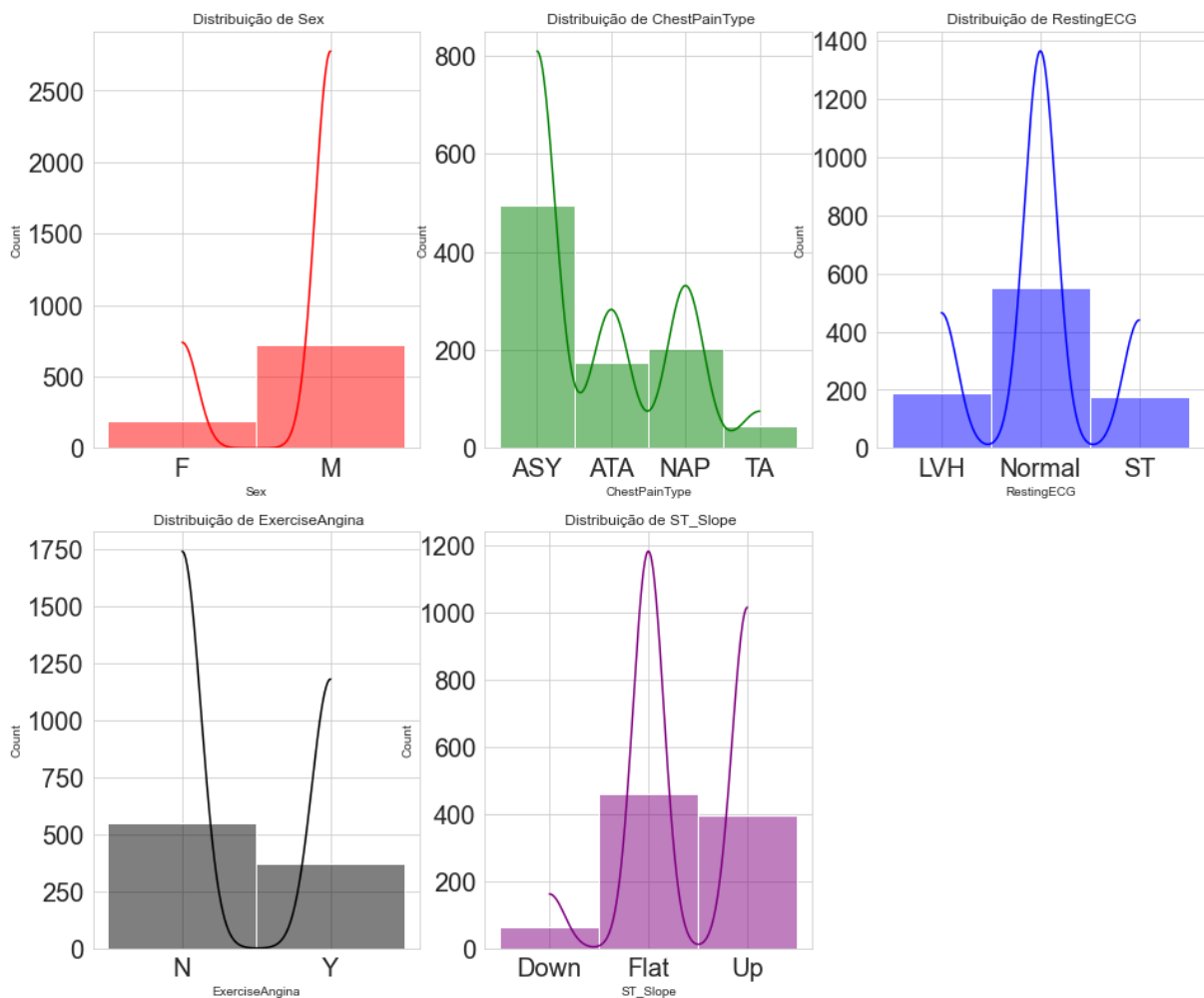


Figura 10: Distribuição variáveis categóricas

Analisando as variáveis categóricas, é possível ver que a maioria dos pacientes da base são do sexo masculino com base na variável *sex*. Quando analisado a variável de *ChestPainType*, apresenta-se uma predominância de pacientes assintomáticos, o

que talvez seja um problema para descobrir se o paciente tem risco ou não de sofrer um ataque cardíaco, por isso faz-se necessário verificar as outras variáveis.

RestingECG apresentou que a maioria dos pacientes não tiveram nenhuma alteração nos exames de eletrocardiograma. Apesar disso, a variável *ExerciseAngina* possui uma distribuição relativamente equilibrada, demonstrando que nem todos os pacientes apresentam desconforto cardíaco.

ST_Slope mostra que a maioria dos pacientes apresentaram aceleração nos batimentos cardíacos quando expostas a atividades físicas, enquanto uma minoria não teve uma resposta muito positiva.

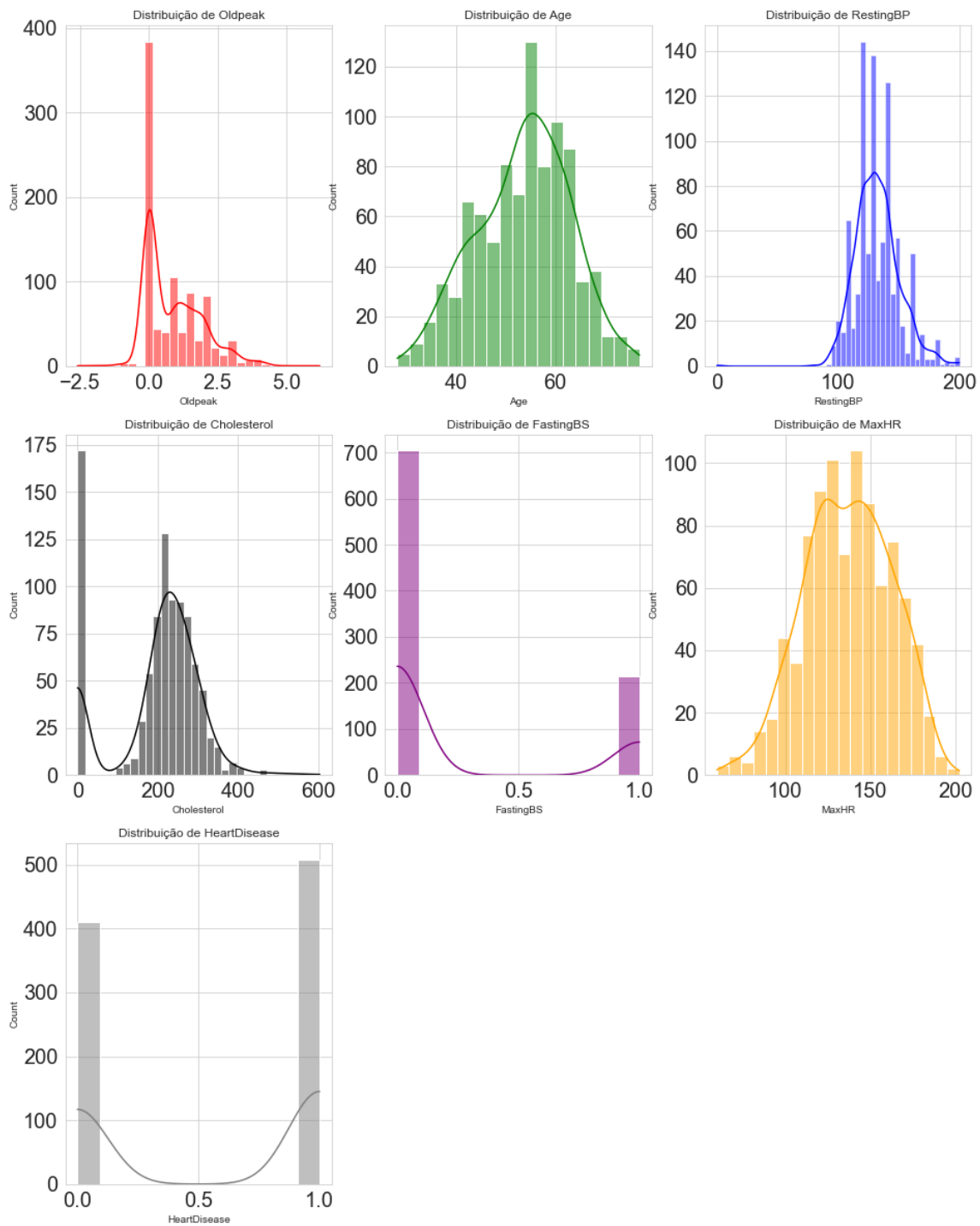


Figura 11: Distribuição variáveis numéricas

Partindo para as variáveis numéricas, aparentemente não existe nenhuma distribuição normal gaussiana, mas é importante comprovar posteriormente aplicando testes de normalidade. Outra informação bem importante é a distribuição de *HeartDisease* (nosso *target*) que possui uma quantidade de pacientes com e sem insuficiência cardíaca bastante balanceada, o que acaba poupando tempo

posteriormente com abordagens de oversampling e undersampling para balancear a nossa variável alvo.

Em relação ao atributo *Cholesterol*, existem alguns valores zerados que provavelmente são indicativos de valores inválidos, conforme visto anteriormente na *Figura 7: Analisando as variáveis com quebras por quartil*. O mesmo acontece com *Oldpeak*, com apenas alguns valores negativos.

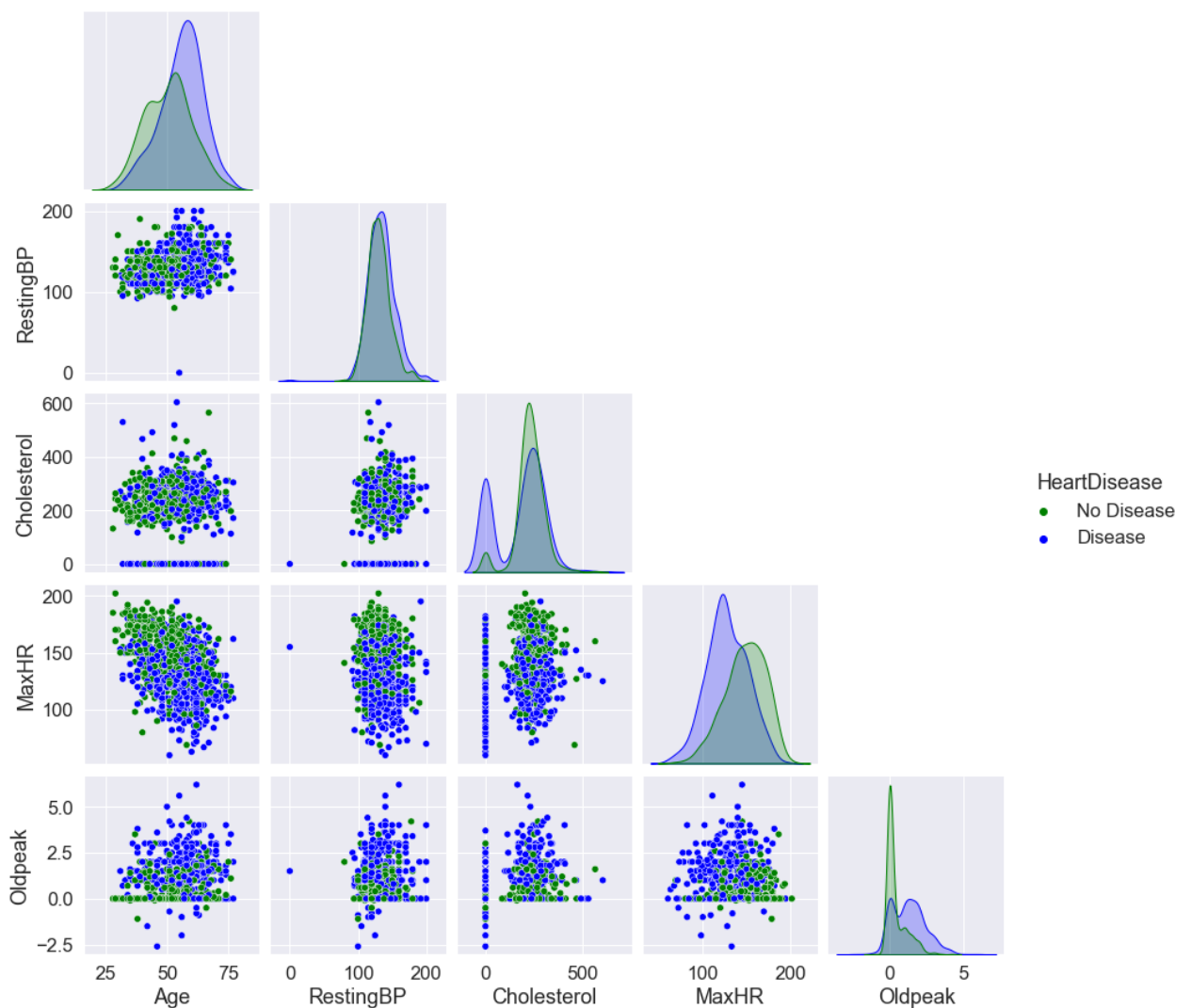


Figura 12: Distribuição variáveis categóricas usando *pairplot*

Utilizando o *scatterplot*, pode-se observar o relacionamento entre as variáveis numéricas em relação a variável de *HeartDisease*, sendo a cor azul representada pela incidência de insuficiência cardíaca e a cor verde em casos de ausência de

incidências. Apesar do *scatterplot* ser uma boa ferramenta para entender um pouco melhor a correlação das variáveis em alguns casos, não foi possível tirar muitas conclusões observando os gráficos, já que os dados estão distribuídos de forma bastante uniforme.

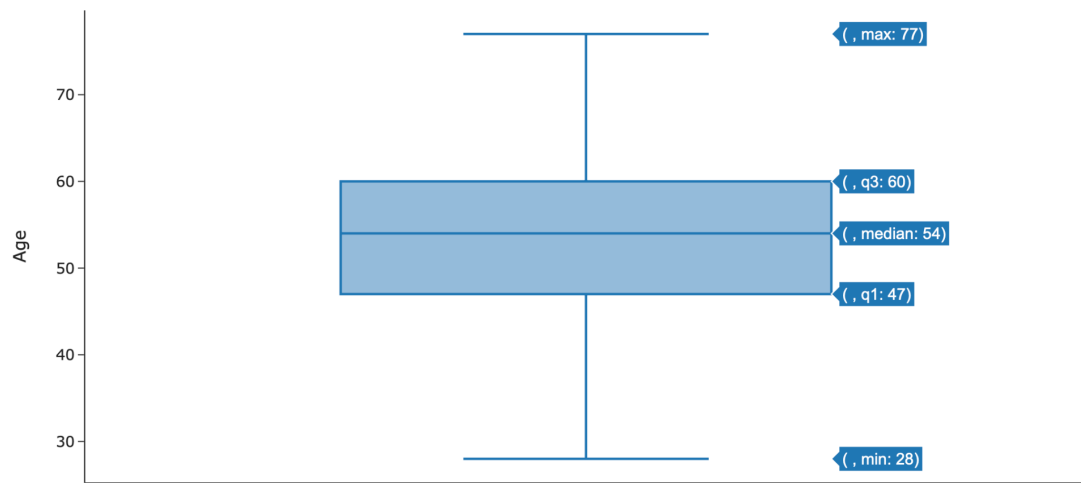


Figura 13: Boxplot variável Age

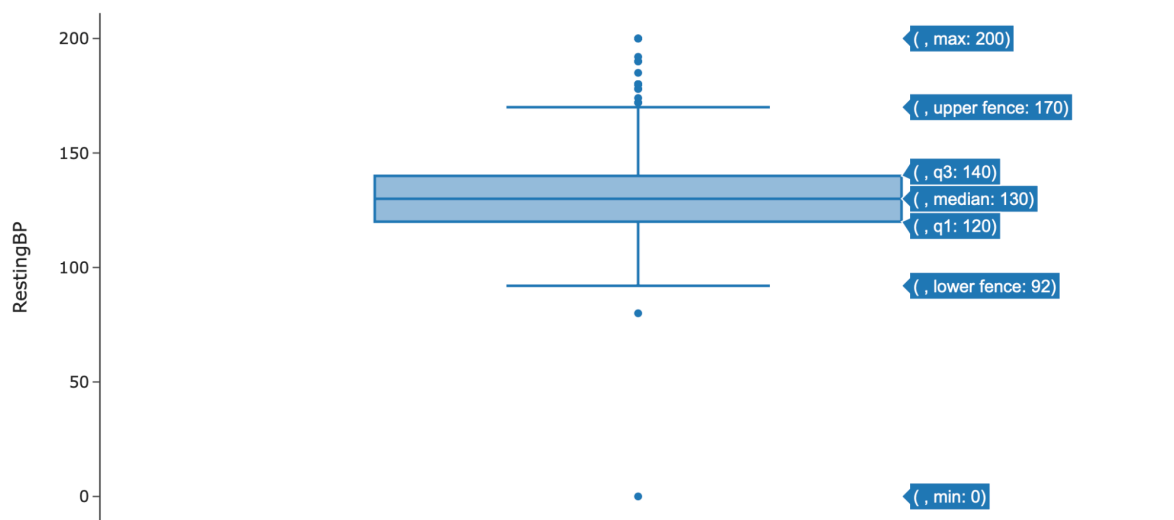


Figura 14: Boxplot variável RestingBP

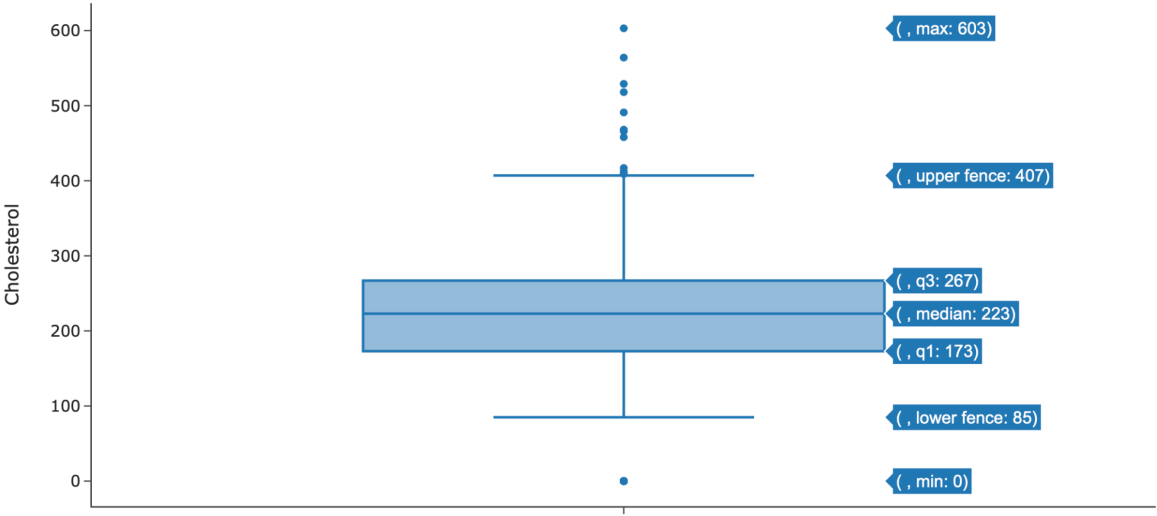


Figura 15: Boxplot variável *Cholesterol*

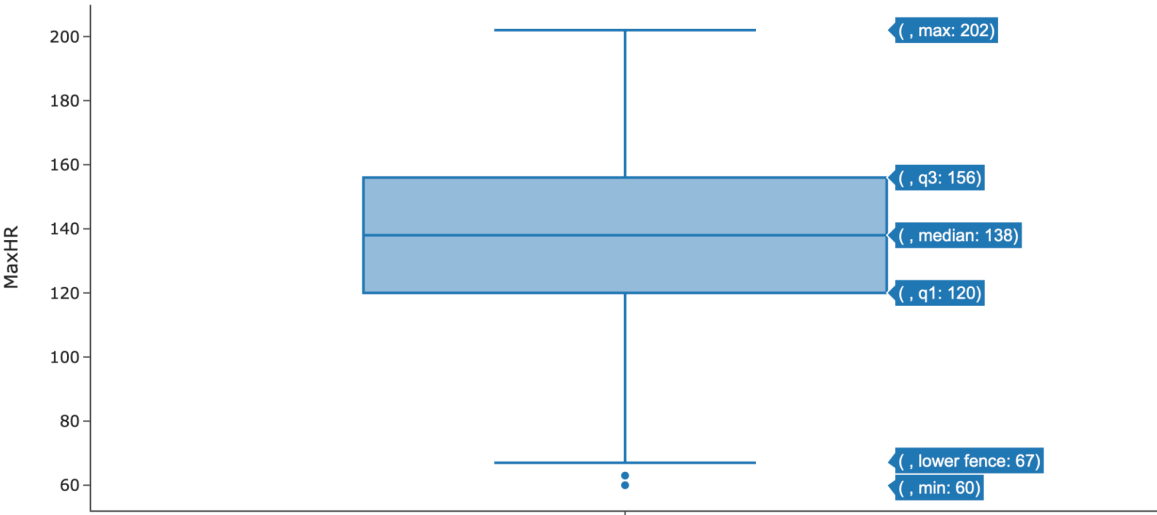


Figura 16: Boxplot variável *MaxHR*

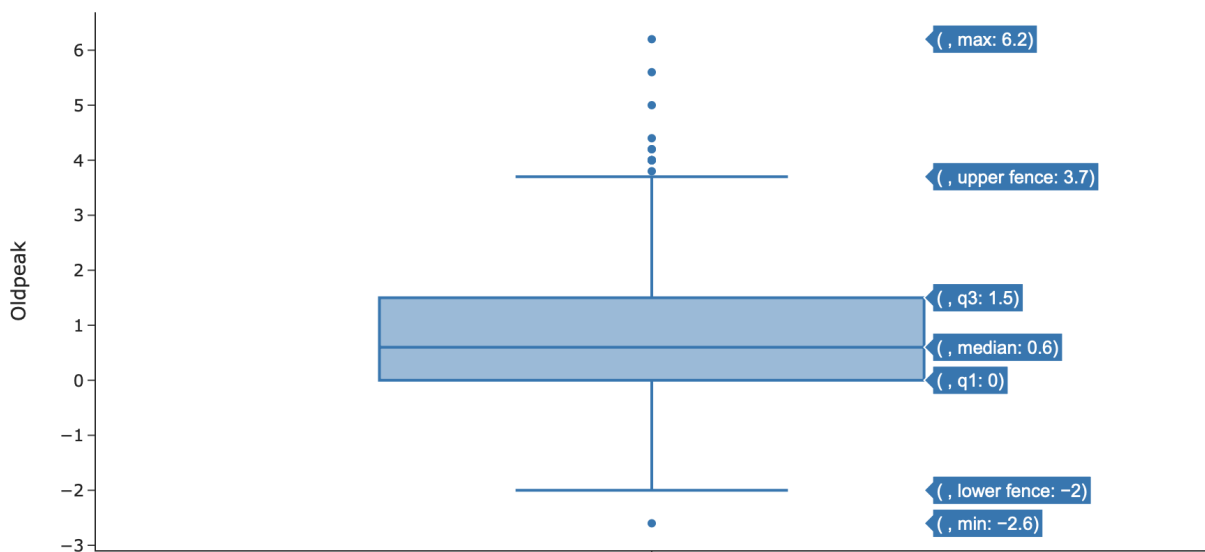


Figura 17: *Boxplot* variável *Oldpeak*

Analisando os boxplots, temos algumas variáveis com outliers, tais como: *RestingBP*, *Cholesterol*, *MaxHR* e *Oldpeak*. Esses outliers podem gerar ruídos nos modelos, prejudicando as previsões, já que o modelo irá aprender com o comportamento apresentado nos dados. Iremos remover os valores inválidos na etapa 7 deste trabalho.

```
from scipy.stats import kstest
from scipy import stats

cols_num = atributos_numericos[['Age', 'RestingBP', 'Cholesterol', 'MaxHR', 'Oldpeak']].copy()
for col in cols_num:
    pvalue = kstest(df[col], stats.norm.cdf).pvalue

    print('-----')
    print('Variável:', col)

    if pvalue < 0.05:
        print('p-value menor que 5%. Hipótese H0 rejeitada')
        print('Não possui distribuição normal')
    else:
        print('p-value menor que 5%. Hipótese H0 Aceita')
        print('Variável possui distribuição normal')

    print('P_value do teste:', pvalue)
```

Figura 18: *Boxplot* variável *Oldpeak*

Observando as distribuições da Figura 11, é possível entender que os atributos não possuem uma distribuição gaussiana. Mesmo assim, é importante aplicar testes estatísticos, a fim de provar matematicamente que as conclusões têm fundamento.

Além disso, é realizado o processo de *Normalizing* ou *Scaling* para ter certeza dessas informações, a fim de evitar equívocos durante a modelagem.

Para realizar este teste de normalidade, foi utilizado o teste de *Kolmogorov-Smirnov* onde são levantadas duas hipóteses:

- H_0 : A variável está normalmente distribuída.
- H_1 : A variável não se encontra na forma normal.

Para que a hipótese alternativa (H_1) seja validada, a hipótese nula (H_0) deve ser rejeitada. Para isso foi aplicado um valor de alpha 5% para critério de avaliação. Neste caso, para que a hipótese nula seja rejeitada, o *p-value* deve ser menor do que o alpha estabelecido.

Segue os resultados abaixo:

Variável	P-value	Resultado
Age	0	Hipótese H0 rejeitada
Cholesterol	0	Hipótese H0 rejeitada
RestingBP	0	Hipótese H0 rejeitada
MaxHR	0	Hipótese H0 rejeitada
OldPeak	4.2004655072318E-200	Hipótese H0 rejeitada

Tabela 2: Teste de *Kolmogorov-Smirnov*

Com base na tabela, pode-se afirmar que todas as variáveis contínuas tiveram a hipótese nula (H_0) rejeitada. Logo, estatisticamente, torna-se evidente que não são distribuições gaussianas.

```

from scipy.stats import mannwhitneyu

cols_num = atributos_numericos[['Age', 'RestingBP', 'Cholesterol', 'MaxHR', 'Oldpeak']].copy()
col_target = df['HeartDisease'] == 1

for col in cols_num:
    positive_heart = df.loc[col_target, col]
    negative_heart = df.loc[~col_target, col]

    pvalue = mannwhitneyu(positive_heart, negative_heart).pvalue

    print('-----')
    print('Variável:', col)

    if pvalue < 0.05:
        print('Hipótese H0 rejeitada. Amostras diferentes')
    else:
        print('Hipótese H0 Aceita. Amostras similares')

    print('P_value do teste:', pvalue)

```

Figura 19: Teste de *Mann-Whitney*

Para analisar se existe mais de uma distribuição para a mesma população, será feito o teste de Mann-Whitney. Este teste trabalha com dois grupos de amostras e tem como objetivo medir o grau de similaridade entre elas.

As hipóteses que compõem o teste são as seguintes:

- H_0 : As amostras são semelhantes e suas variações são decorrentes da aleatoriedade.
- H_1 : As amostras se demonstram diferentes e suas variações se diferenciam de acordo com o acontecimento do fato estudado.

Variável	P-value	Resultado
Age	9.02847069514623E-19	Hipótese H0 rejeitada
Cholesterol	1.14015618177246E-05	Hipótese H0 rejeitada
RestingBP	0.000282403774686077	Hipótese H0 rejeitada
MaxHR	7.53179435979887E-35	Hipótese H0 rejeitada
OldPeak	3.3839225217194E-37	Hipótese H0 rejeitada

Tabela 3: Teste de *Mann-Whitney*

De acordo com o teste de *Mann-Whitney*, todas as variáveis testadas tiveram a sua hipótese nula (H_0) rejeitada. Com base nisso, é possível afirmar que as amostras são estatisticamente divergentes.

Para as variáveis qualitativas. Em adição, será feito o teste de *Qui-Quadrado* para $\alpha < 0.05$, visando testar as proporções das variáveis. Para o teste, temos:

- H_0 as proporções são iguais;
- H_1 as proporções não são iguais.

Variável	P-value	Resultado
ChestPainType@NAP	1.86E-10	Hipótese H_0 rejeitada
ChestPainType@ASY	8.63E-55	Hipótese H_0 rejeitada
ChestPainType@ATA	0.131576751228143	H_0 aceita
ChestPainType@TA	3.38E-37	Hipótese H_0 rejeitada
ExerciseAngina@N	2.91E-50	Hipótese H_0 rejeitada
ExerciseAngina@Y	2.91E-50	Hipótese H_0 rejeitada
RestingECG@LVH	0.809528258440575	H_0 aceita
RestingECG@Normal	0.0067906242525704	Hipótese H_0 rejeitada
RestingECG@ST	0.00250729009841889	Hipótese H_0 rejeitada
Sex@F	4.60E-20	Hipótese H_0 rejeitada

Sex@M	4.60E-20	Hipótese H0 rejeitada
ST_Slope@Down	0.000342178577264425	Hipótese H0 rejeitada
ST_Slope@Flat	8.91E-63	Hipótese H0 rejeitada
ST_Slope@Up	1.03-78	Hipótese H0 rejeitada

Tabela 4: Teste de *Qui-Quadrado*

Pode-se avaliar pelos gráficos e também pelos resultados do teste *Qui-Quadrado*, que todas as variáveis possuem uma diferença estatisticamente significativa em suas proporções, com exceção de *ChestPainType@ATA* e *RestingECG@LVH*.

Para analisar a melhor a correlação entre as variáveis, será utilizado o *heatmap*. O método utilizado foi o *spearman*, já que as variáveis não seguem a distribuição gaussiana.

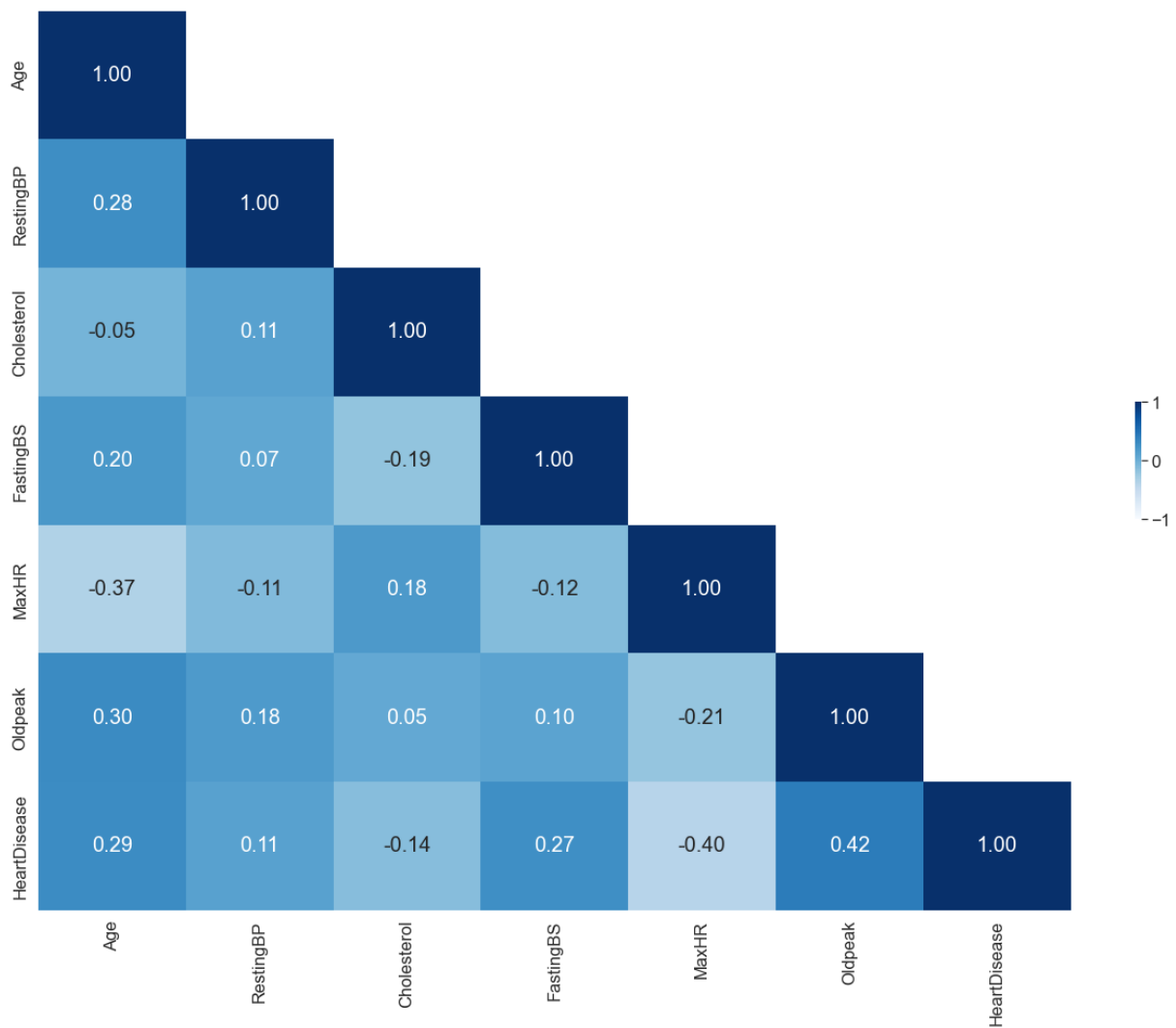


Figura 20: *Heatmap*

As variáveis que possuem mais correlação em relação ao target são *MaxHR* e *Oldpeak*. As outras variáveis possuem correlação abaixo de 50%, o que é considerado uma correlação fraca.

7. Preparação dos Dados para os Modelos de Aprendizado de Máquina

Nesta etapa, será realizado a preparação dos dados e remover os dados inválidos. Além disso, criar variáveis para começar os testes com os modelos preditivos.

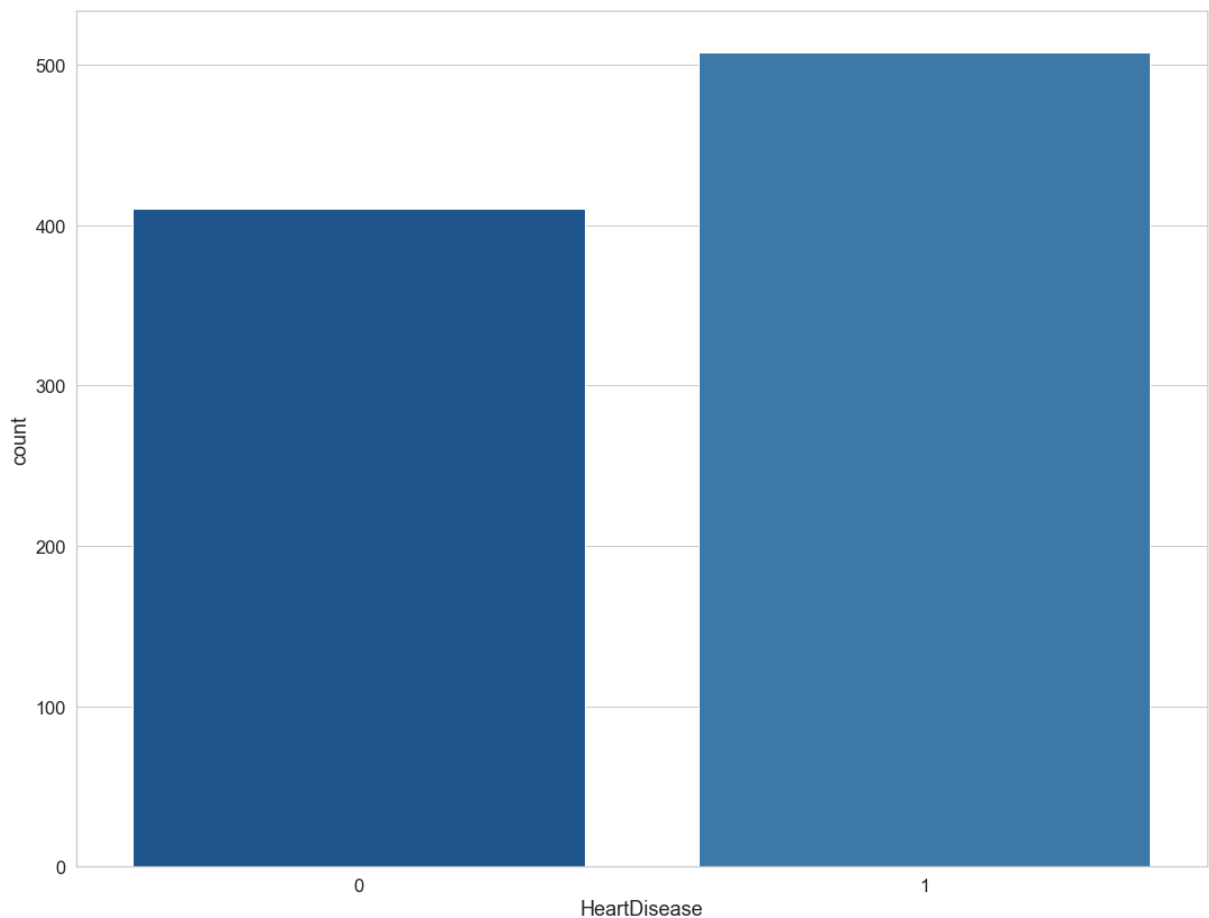


Figura 21: Verificando o balanceamento da variável *target*

Com base no gráfico acima, pode-se afirmar que a variável *target* encontra-se balanceada. Neste caso, não será necessário aplicar nenhuma técnica de balanceamento como *oversampling* ou *undersampling*.

```
def fill_median(df_x, df_y, col, median_normal=None, median_disease=None):
    df_median = pd.concat([df_x, df_y], axis=1)

    if (median_normal == None) & (median_disease == None):
        median_normal = df_median.loc[(df_median[col]!=0) &
                                      (df_median['HeartDisease']==0), col].median()
        median_disease = df_median.loc[(df_median[col]!=0) &
                                       (df_median['HeartDisease']==1), col].median()

    return (df_median.apply(lambda x : x[col]
                            if x[col] != 0
                            else median_normal
                                if x['HeartDisease'] == 0
                                else median_disease,
                            axis=1), median_normal, median_disease)
```

Figura 22: Método para preencher os outliers com a mediana

Foi constatado que existiam muitos dados zerados para a variável de Cholesterol, para não ocorrer falta de dados, será substituído os dados zerados pela mediana. Desta forma, será mantido a mesma quantidade de registros na base.

```
from sklearn.model_selection import train_test_split
X = encoded.drop(columns='HeartDisease')
y = encoded['HeartDisease']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Figura 23: Separação variáveis treino e teste

Nesta última parte, foi passado para o método de *train_test_split* as variáveis tratadas anteriormente com o *OneHotEncoder* para que os atributos categóricos, em conjunto com as features de valores contínuos, sejam aplicados corretamente no modelo preditivo.

8. Aplicação de Modelos de Aprendizado de Máquina

Nesta etapa de modelagem será treinado três modelos de classificação para prever quais pacientes tiveram parada cardiovascular e quem não tem risco de desenvolver a enfermidade, sendo representado por 1 e 0, respectivamente.

Para potencializar o desempenho dos modelos preditivos, será realizado uma otimização de hiperparâmetros aplicando uma biblioteca chamada *RandomizedRandomSearchCV*, que tem como objetivo principal realizar testes com os parâmetros passados e retornar o melhor modelo possível dentro dos ranges estabelecidos. Desta forma, não é utilizado apenas os modelos de forma padronizada e, sim de forma direcionada a encontrar a melhor solução para os dados em estudo. Abaixo, encontra-se os hiperparâmetros que serão testados e os modelos que serão utilizados, são eles: *XGBClassifier*, *RandomForestClassifier* e o *DecisionTreeClassifier*.

```

#Lista com o apelido do modelo, a instância do modelo e os hiperparâmetros a serem testados pelo RandomizedSearchCV
search_models=[
    ('XGBClassifier',
     XGBClassifier(random_state=42),
     {
         "max_depth": [2,5,8],
         "min_child_weight": [2,5,8],
         "subsample": [0.3, 0.7, 0.9],
         "colsample_bytree": [0.3, 0.7, 0.9]
     }),
    ('RandomForestClassifier',
     RandomForestClassifier(random_state=42),
     {
         "max_depth": [2,5,8],
         "min_samples_split": [2, 5, 10],
         "min_samples_leaf": [2, 4, 6],
         "n_estimators": [500]
     }),
    ('DecisionTreeClassifier',
     DecisionTreeClassifier(random_state=42),
     {
         "max_depth": [2,5,8],
         "min_samples_split": [2, 5, 8],
         "min_samples_leaf": [2, 5, 8]
     }),
]

```

Figura 24: Modelos e hiperparâmetros de teste

```

model_list = []

# É passado o apelido do modelo, a instância do modelo e os hiperparâmetros a serem testados, respectivamente
for model_name, model, params in search_models:

    #RandomSearchCV recebendo os modelos por iteração, os hiperparâmetros, cross-validation,
    #o número de processadores a serem utilizados (n_jobs) e a métrica de otimização desejada (scoring)
    random_search_model = RandomizedSearchCV(model, params, cv=5, n_jobs=-1, scoring='f1').fit(X_train, y_train)

    #Salva os melhores hiperparâmetros de cada modelo
    model_list.append(random_search_model.best_estimator_)

    #printa os resultados dos modelos
    print(model_name)
    print("Precision:", precision_score(y_test, random_search_model.predict(X_test)))
    print("Recall:", recall_score(y_test, random_search_model.predict(X_test)))
    print("F1 Score:", f1_score(y_test, random_search_model.predict(X_test)))
    print("ROC AUC Score:", roc_auc_score(y_test, random_search_model.predict(X_test)))
    print('-----')

```

Figura 25: RandomizedSearchCV nos modelos preditivos

Conforme a *Figura 25: RandomizedSearchCV nos modelos preditivos*, os modelos estão sendo otimizados aplicando a função do RandomizedSearchCV com base nos hiperparâmetros estabelecidos. Além disso, é utilizado a técnica de cross-validation na base para quebrar o dataset em cinco partes, a fim de descobrir qual é a melhor amostra para prever os possíveis casos de parada cardiovascular e evitar overfitting.

8.1 Apresentação do Pipeline do Processo

Esta parte do trabalho busca esclarecer as etapas do processo de desenvolvimento do estudo, explicando de forma sucinta cada etapa, conforme a figura abaixo:



Figura 26: Fluxo de atividades do trabalho

Data Source: Esta etapa é composta por uma das partes mais importantes do nosso trabalho: os dados. Foi decidido dividir em duas etapas, pois é importante entender a origem do dataset e quais foram os critérios adotados para que seja feita a seleção da melhor base possível. Tal separação foi feita da seguinte forma:

- **Kaggle:** Plataforma utilizada por profissionais da área de dados e recomendada pela PUC Minas como uma possível fonte de dados para trabalhos acadêmicos. Apesar de possuir muitos datasets, nem todos são muito confiáveis e possuem dados de qualidade. Após bastante pesquisa, foi

selecionado um dataset de pacientes com suspeita ou vítimas de insuficiência cardíaca, conforme descrito na *etapa 4: coleta de dados* deste trabalho.

- **CSV:** Formato fornecido para realizar o trabalho. Arquivos deste formato excel são muito comuns na área de dados, devido a facilidade de manuseio. Apesar disso, não é o mais recomendado para bases de dados muito extensas, já que pode apresentar lentidão e necessitar de máquinas mais robustas para execução. Para bases extensas, é recomendado arquivos em formatos parquet, já que ocupam menos espaço foram desenvolvidos para tal finalidade.

Flow de Desenvolvimento: Este flow de desenvolvimento foi feito pensando na metodologia CRISP-DM (Cross Industry Standard Process for Data Mining), mas de forma simplificada, visto que não há acesso aos stakeholders do projeto e não será feito deploy do modelo preditivo em ambiente de produção até o momento. O estudo pode ser dividido em 3 partes:

- **Análise de Dados:** Nesta etapa, foi desenvolvido uma análise descritiva para entendimento da distribuição dos dados e quais tratamentos seriam necessários para trabalharmos com os dados com a melhor qualidade possível. Após as análises univariadas, foi iniciada a análise diagnóstica onde foram feitas inferências estatísticas e testes de hipóteses, a fim de entender o relacionamento entre os atributos de forma mais assertiva. Para isso, foram utilizadas bibliotecas como pandas, numpy, matplotlib, plotly, entre outros.
- **Preparação dos dados:** Nesta etapa, foi realizado tratamentos nas variáveis serem aplicadas no modelo posteriormente. Para variáveis categóricas, por exemplo, foi feito o OneHotEncoder para categorizar os atributos de forma que seja utilizável pelos modelos preditivos. Além disso, foi feito o preenchimento dos valores faltantes com a mediana, conforme a *figura 22: Separação variáveis treino e teste*. Após a finalização dos tratamentos necessários, a base foi dividida em treino e teste (*figura 23: Separação Variáveis Treino e Teste*) para iniciar-se os testes de modelagem.
- **Modelagem e Validação:** Nesta etapa, foram utilizados três modelos de classificação: XGboostClassifier, DecisionTreeClassifier e o RandomForestClassifier, com o intuito de prever se os pacientes apresentam insuficiência cardíaca ou se estão saudáveis. São feitos testes com três

modelos para fins comparativos e ver qual apresenta melhor performance para resolver o problema. Para extrair o melhor desempenho de cada modelo foi feito um tuning dos hiperparâmetros aplicando o f1-score como otimizador e, posteriormente, os resultados foram validados através das métricas de classificação: Recall, Precision e F1-Score.

9. Discussão dos Resultados

Para avaliar os resultados dos modelos, foram utilizadas três métricas de classificação: *Recall*, *Precision* e *F1-Score*. Faz-se necessário entender como funciona uma matriz de confusão previamente para entender melhor como funcionam as métricas.

9.1 Matriz de confusão

Uma matriz de confusão é uma tabela que indica os erros e acertos do seu modelo, comparando com o resultado esperado. Abaixo, é fornecido um exemplo ilustrar como funciona.

		Detectada	
		Sim	Não
Real	Sim	Verdadeiro Positivo (VP)	Falso Negativo (FN)
	Não	Falso Positivo (FP)	Verdadeiro Negativo (VN)

Figura 27: Matriz de confusão

- **Verdadeiros Positivos:** Classificação correta da classe positivo;
- **Falsos Negativos:** Erro em que o modelo previu a classe Negativo quando, na verdade, o valor correto era da classe positivo;
- **Falsos Positivos:** O modelo previu a classe positivo quando, na verdade, o valor correto seria a classe negativo;

- **Verdadeiros Negativos:** Classificação correta da classe negativo.

De forma geral, costuma-se ler a matriz de confusão por meio da diagonal principal para entender se o modelo realmente está realizando previsões corretas, seja para classe positiva ou negativa.

9.2 Recall

$$Recall = \frac{VP}{VP + FN}$$

O *recall* é muito utilizado em situações em que os Falsos Negativos são considerados mais prejudiciais que os Falsos Positivos. No caso, por exemplo, o modelo pode tentar acertar todos os pacientes vítimas de problemas cardiovasculares, porém, pode ser que ele classifique pacientes saudáveis como doentes (Falso Positivo) no processo. Desta forma, se torna muito arriscado utilizar apenas o recall como métrica de avaliação. É possível interpretar a fórmula acima da seguinte forma: entre todas as situações de classe Positivo como valor esperado, quantas realmente estão corretas?

9.3 Precision

$$Precision = \frac{VP}{VP + FP}$$

A precisão pode ser usada em uma situação em que os Falsos Positivos são considerados mais prejudiciais que os Falsos Negativos. Por exemplo, ao classificar um paciente como doente, faz-se necessário atentar-se a quantidade de erros que o modelo pode gerar, pois é extremamente perigoso diagnosticar um paciente como doente, quando na verdade, ele está saudável. A fórmula acima pode ser compreendida da seguinte forma: dentre todas as classificações de classe Positivo que o modelo fez, quantas estão corretas?

9.4 F1-Score

$$F1Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

O F1-Score é uma média harmônica entre o recall e o precision. No caso, esta métrica é muito interessante para encontrar um valor ideal entre o recall e o precision. Por exemplo, um modelo preditivo pode acertar todos os pacientes vítimas de insuficiência cardíaca (recall alto), mas errar muito no processo (precision baixa). Com o intuito de acertar o máximo possível sem fazer muitas previsões errôneas neste processo, o F1-score acaba sendo uma das métricas mais adequadas.

9.5 Resultados dos modelos preditivos

Nesta etapa, será feita a análise do resultado dos modelos utilizados para prever os pacientes que tiveram insuficiência cardíaca e os que não tiveram. Abaixo, segue uma tabela com o resultado dos modelos.

Modelo	Recall (%)	Precision (%)	F1-Score (%)
XgboostClassifier	77	0.89	0.83
RandomForecastClassifier	83	0.89	0.85
DecisionTreeClassifier	76	0.94	0.84

Tabela 5: Métrica dos modelos

Apesar dos modelos apresentarem resultados parecidos, o modelo que teve uma melhor performance foi o RandomForecastClassifier, pois foi o que apresentou o melhor balanço entre o recall e o precision, resultando consequentemente em um F1-Score mais alto. Quando analisado os outros modelos, fica difícil classificar qual seria o mais interessante, já que o DecisionTreeClassifier conseguiu prever mais pacientes doentes, porém acabou errando mais neste processo. Para classificar o 2 lugar, acaba sendo mais assertivo utilizar o F1-Score para manter os mesmo padrão aplicado para o modelo RandomForecastClassifier. Realizando um ranqueamento de modelos preditivos, os resultados ficam desta forma quando alinhados por F1-Score: RandomForecastClassifier, XgboostClassifier e DecisionTreeClassifier em último lugar.

10. Conclusão

Ao longo deste trabalho foi possível analisar resultados de exames de 918 pacientes de hospitais ao redor do mundo. Entender quais variáveis estão correlacionadas com a incidência de paradas cardiovasculares se torna mais fácil com apoio das ferramentas de Ciência de Dados, mas, mesmo assim, foi desafiador, pois nem todas as variáveis apresentaram o comportamento esperado. Se for feita uma análise referente a matriz de correlação na Figura 20: *Heatmap*, observa-se que o colesterol apresentou uma correlação negativa em relação ao casos de paradas cardiovasculares, o que não é verdade, já que o excesso de gordura nas artérias é um dos principais causadores de problemas cardiovasculares, segundo as pesquisas feitas durante o trabalho.

A falta de stakeholders para auxiliar com os problemas de negócio foi um dos limitadores para imersão maior no estudo, pois um profissional da saúde poderia trazer insights mais assertivos e sugerir estudos mais profundos sobre o tema.

Apesar das dificuldades, foi decidido que a melhor abordagem seria desenvolver modelos preditivos de classificação e utilizá-los aplicando o F1-Score como parâmetro de otimização, já que esta métrica é capaz de ponderar os acertos sem deixar de considerar o número de tentativas, entregando o melhor resultado possível para o negócio.

Os resultados foram muito positivos, pois foi atingido um recall de 87%, um precision de 89% e um F1-Score de 85% aplicando o modelo de classificação RandomForestClassifier. Baseado nessas métricas, é possível afirmar que esta solução tem um alto potencial para ser implementada se fosse fornecida a oportunidade de entrar em contato com os profissionais envolvidos. Mesmo assim, seria necessário um trabalho mais árduo pensando em arquitetura de sistemas, já que a abordagem utilizada foi exploratória, apesar de seguir boas práticas de programação.

Portanto, como conclusão, um próximo passo seria centralizar os dados dos hospitais envolvidos em um banco de dados para automatizar a solução. Desta forma, seria possível extrair os dados e aprimorar o código desenvolvido para rodar em um ambiente da Azure Machine Learning, por exemplo. Dentro da plataforma, seria possível registrar as previsões em um datahub, refletir esses dados em

dashboards para os profissionais da área da saúde e servir como uma ferramenta de apoio para tomada de decisões.

11. Links

O repositório com o presente trabalho e o código de execução pode ser encontrado no link abaixo:

https://github.com/VinicioLima98/TCC_PUC_MG_DATA_SCIENCE

12. Referências

Harvard Health Publishing Harvard Medical School. **Heart Health**. 2022. Disponível em: <<https://www.health.harvard.edu/topics/heart-health>> Acesso em 02/06/2022

Harvard T.H. Chan School of Public Health. **Heart Disease**. 2022. Disponível em: <<https://www.hsph.harvard.edu/nutritionsource/disease-prevention/cardiovascular-disease/>> Acesso em 02/06/2022

GO AS, Mozaffarian D, Roger VL, et al. **Heart disease and stroke statistics**, 2013. National Library of Medicine. Disponível em: <<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5408511/>> Acesso em 02/06/2022;

Matthews CE, George SM, Moore SC, et al. 2012. **Amount of time spent in sedentary behaviors and cause-specific mortality in US adults**. National Library of Medicine. Disponível em: <<https://pubmed.ncbi.nlm.nih.gov/22218159/>> Acesso em 02/06/2022,

Rodrigues, Vitor. **Métricas de Avaliação: acurácia, precisão, recall... quais as diferenças?**. Medium. Disponível em: <<https://vitorborbarodrigues.medium.com/m%C3%A9tricas-de-avalia%C3%A7%C3%A3o-acur%C3%A1cia-precis%C3%A3o-recall-quais-as-diferen%C3%A7as-c8f05e0a513c>>. Acesso em 04/10/2022.