

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

Bruno Defante da Silva

Modelo preditivo para inferência em paradas cardíacas

Belo Horizonte
Outubro de 2022

Bruno Defante da Silva

Modelo preditivo para inferência em paradas cardíacas

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

Introdução.....	5
2. Descrição do Problema e da Solução Proposta	5
3. Canvas Analítico	6
4. Coleta de Dados	7
5. Processamento/Tratamento de Dados.....	8
6. Análise e Exploração dos Dados.....	10
6.1. Visualização de dados	11
6.1.1. Variáveis numéricas.....	11
6.1.2. Variáveis categóricas.....	12
6.1.3. Visualização em pares.....	13
6.1.4. Mapa de calor para correlação dos dados	14
6.1.5. Informação Mútua	15
6.1.6. Análise de anomalias.....	16
6.1.7. Visualização Probabilística	17
6.2. Testes de Hipóteses.....	18
6.2.1. Teste de Normalidade	18
6.2.2. Teste de tendência central	19
6.2.3. Teste de associação	21
7. Preparação dos Dados para os Modelos de Aprendizado de Máquina	22
7.1. Separação dos dados	22
7.2. Pipeline de tratamento de dados.....	23
7.2.1. Classe criada para tratamento de outliers.....	23
7.3. Balanceamento de classes	24
8. Aplicação de Modelos de Aprendizado de Máquina.....	25
8.1. Pipeline de processamento de dados.....	25
8.2. Pipeline de modelos	26
8.3. Execução do tuning do modelo.....	27
8.4. Pipeline completo do desenvolvimento do trabalho	29
9. Discussão dos Resultados	30
9.1. Definição das métricas	30
9.1.1. Matriz de Confusão	30

9.1.2. Recall.....	31
9.1.3. Precision	31
9.1.4. F1 Score	32
9.2. Análise dos modelos	32
10. Conclusão.....	33
11. Links	34
12. Referências	34

Introdução

Algoritmos de aprendizagem de máquina estão se mostrando cada vez mais robustos e confiáveis. O uso destes algoritmos, nos possibilita observar padrões e comportamentos dos quais sozinhos não seríamos capazes. Grandes exemplos dessa evolução, são encontrados, por exemplo, em carros autônomos e algoritmos que são capazes de auxiliar a identificação de possíveis células cancerígenas em exames (Staff, NCI, 2022). Como um exemplo de ferramenta presente no mercado, podemos citar o IBM Watson que possui módulos exclusivos para auxiliar em análises e problemas que estão dentro da área da saúde (IBM, 2022?).

Com essa premissa, este presente trabalho surge com a intenção de estudar quais seriam as principais características que estão relacionadas às doenças do coração, em especial, paradas cardíacas. Em adição, será proposto um modelo preditivo que, baseado no aprendizado em dados históricos, possibilitará classificar novos casos.

2. Descrição do Problema e da Solução Proposta

As doenças cardiovasculares (DCV) são a causa número um de mortes no planeta. Os fatores de risco são variados: desde fumo, diabetes, hipertensão e obesidade, até poluição do ar e condições raras e negligenciadas, como Doença de Chagas e amiloidose cardíaca (Ministério da Saúde, 2022?).

Casos como esses, possuem a necessidade de rápida detecção para que sejam possíveis tratamentos ainda nos primeiros sintomas, visando assegurar que o quadro clínico não chegue a uma possível fatalidade.

Com a finalidade de auxiliar e contribuir com o meio acadêmico e da saúde, este trabalho tem como objetivo entender algumas das características que influenciam no aparecimento de doenças cardiovasculares.








Algoritmos estatísticos serão utilizados para embasar as análises e hipóteses que forem levantadas. Para os modelos preditivos, serão testados alguns modelos de classificação, entre eles, o *XGBoost* e *Random Forest*.

Para o estudo, será utilizado a consolidação de 5 conjuntos de dados independentes, somando 11 variáveis comuns entre eles. Os conjuntos de dados que serão estudados são:

Conjunto de dados	Nº de Observações
Cleveland	303
Hungarian	294
Switzerland	123
Long Beach VA	200
Stalog (Heart) Data Set	270

Tabela 1: lista de conjunto de dados utilizado

3. Canvas Analítico

Software Analytics Canvas		Project: Modelo preditivo para inferência em paradas cardíacas	
<p> 1. Question</p> <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 diferenciar dados de um paciente que teve uma parada cardíaca de alguém que não? • Caso seja possível, podemos aprender este comportamento e, por meio deste, inferir novas paradas cardíacas apenas olhando para os dados referentes ao paciente e ao seu estado clínico? 	<p> 2. Data Sources</p> <p><i>Which data can possibly answer our question? What information do we need?</i></p> <ul style="list-style-type: none"> • Dados relacionados a "quem é o paciente?" e extraídos de exames dos pacientes como (idade, colesterol, pressão sanguínea, etc...) • Os dados necessitam possuir um variável rótulo para que seja possível diferenciar em que circunstâncias é identificada a ocorrência de uma parada cardíaca, assim como o seu oposto. 	<p> 3. Heuristics</p> <p><i>Which assumptions do we want to make to simplify the answer to our question?</i></p> <ul style="list-style-type: none"> • Entender características que podem descrever uma parada cardíaca • Encontrar possíveis correlações dos dados com o nosso alvo • Se os resultados forem positivos podemos criar nosso modelo preditivo e realizar os testes 	<p> 4. Validation</p> <p><i>What results do we expect from our analysis, how are they reviewed and presented in an understandable way?</i></p> <p>Esperamos que seja possível obter boas correlações dos dados</p> <p>Com boas correlações, se torna plausível a utilização de algoritmos de machine learning para aprendizagem de padrões existentes nos dados</p>
<p> 5. Implementation</p> <p><i>How can we implement the analysis step by step and in a comprehensible way?</i></p> <p>A implementação da análise se dará pelo uso da metodologia já consolidada no mercado: CRISP-DM (Cross Industry Standard Process for Data Mining)</p> <p>Os passos a se seguir serão os seguintes:</p> <ul style="list-style-type: none"> • Business understanding: nesta etapa, devemos entender o cenário em que nosso estudo se encontra. • Data understanding: uma vez que tenhamos os dados em mãos, devemos entendê-los para que seja possível saber qual a qualidade do dado que será estudado e quais transformações serão necessárias. • Data preparation: fase em que devemos realizar os tratamentos e devidas transformações que foram devidamente mapeadas durante a fase de entendimento. • Modeling: aqui, será onde daremos início a criação do modelo de aprendizagem de máquina. Devemos, então, testar diversos modelos e técnicas para consolidarmos uma sólida base de comparação. • Evaluation: após criarmos uma base sólida de modelos, é chegada a hora de validarmos qual modelo melhor resolve nosso problema de negócio, o qual estamos estudando. • Deployment: neste estudo, o deployment será dado pelas respostas das perguntas que foram feitas inicialmente e previamente documentadas na parte escrita deste trabalho. 		<p> 6. Results</p> <p><i>What are the main insights from our analysis?</i></p> <ul style="list-style-type: none"> • Em desempenhos positivos, é possível observar como algoritmos de machine learning podem agregar valor como ferramentas analíticas para profissionais da saúde. <p>Estes, podem observar detalhes e comportamentos que um ser Humano pode não perceber por mais que esteja atento.</p>	<p> 7. Next Steps</p> <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, visto a melhoria da performance do modelo escolhido e também a inclusão de profissionais da saúde que possam estar interessados na pesquisa e queiram ajudá-la a evoluir.

Software Analytics Canvas v1.0 designed by Markus Harrer. Visit <https://www.feststelltaste.de/software-analytics-canvas/> for more information. CC BY-SA 4.0

Imagem 1: Cavas analítico

4. Coleta de Dados

Nome do dataset: Heart Failure Prediction Dataset Descrição: 11 clinical features for predicting heart disease events Link: https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction		
Nome do Atributo	Descrição	Tipo
Age	Idade do paciente em anos	Numérico
Sex	Gênero dos pacientes	Carácter
ChestPainType	Tipo de dor no peito sentida	Carácter
RestingBP	Pressão arterial em repouso	Numérico
Cholesterol	Colesterol sérico	Numérico
FastingBS	Açúcar no sangue em jejum	Booleano
RestingECG	Resultado do eletrocardiograma em descanso	Carácter
MaxHR	Frequência cardíaca máxima atingida	Numérico
ExerciseAngina	Angina induzida por exercício	Booleano
Oldpeak	Depressão de ST induzida pelo exercício em relação ao repouso	Numérico
ST_Slope	Inclinação do segmento ST de exercício de pico	Carácter
HeartDisease	Variável alvo (Doença no Coração)	Booleana

Tabela 2: Campos e suas descrições

A partir desses dados, é construída toda a teoria e prática que envolve este projeto, nos possibilitando buscar as respostas para as perguntas motivadoras, nos quais, não somente traçam um objetivo, como também guiam o desenvolvimento.

Os dados encontram-se disponíveis dentro da plataforma destinada ao compartilhamento de bases públicas chamada *kaggle*. A base é disponibilizada sob a licença **Open Data Commons Open Database License (ODbL) v1.0**.

Com esta coleta de dados, é obtido o insumo para que seja desenvolvida toda a análise requerida.

Espera-se, entender e visualizar possíveis correlações entre os dados e o aparecimento de doenças cardíacas, assim como, ser possível a criação de um modelo preditivo que visará facilitar novas identificações através de padrões que reflitam a realidade. Em complemento, uma vez treinado, o modelo estará apto a desempenhar seu papel através da inserção de novos dados que poderão ser coletados.

Será estudada, exaustivamente, possíveis correlações, transformações e principais variáveis que possam acrescentar valor e colaborem positivamente para alcançar os objetivos deste estudo.

Em sua maioria os dados coletados possuem como fonte os Estados Unidos da América, porém temos amostras de outros países como Nova Zelândia e Hungria.

Os dados não possuem marco temporal, uma vez que o problema a ser resolvido não demonstra a necessidade aparente de estar disposto em um grão temporal. Outro fator que dificulta a recuperação desta informação é que, nesta coleta de dados é representada pela junção de outros 5 conjuntos de dados que, podem ser de períodos iguais ou completamente diferentes.

5. Processamento/Tratamento de Dados

Para realizar o pré-processamento dos dados, será utilizado bibliotecas já consolidadas em projetos de ciência de dados, como: Pandas, Scikit-Learn e Numpy.

- **Pandas:** necessário para que seja possível manusearmos os dados em forma de tabelas.
- **Numpy:** possui múltiplas ferramentas estatísticas e auxilia em algumas funções quando queremos ver algumas medidas de dispersão, por exemplo.
- **Scikit-Learn:** é encontrado a maioria das funções para o processamento e tratamento dos dados. Além disso, também possui a maioria dos modelos de aprendizagem de máquina.

Exemplo de uso das bibliotecas:

```
import pandas as pd
import numpy as np
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.pipeline import Pipeline
```

Imagem 1: Importação das bibliotecas

Como citado anteriormente, para manipulação dos dados, será utilizado a biblioteca Pandas.

```
df = pd.read_csv('../data/heart.csv')
✓ 0.2s
```

Imagem 2: Leitura dos dados

Foi realizada uma análise inicial e, com isso, foi constatado que o conjunto de dados possui 2 tipos de variáveis, sendo elas: numéricas e categóricas.

```
df.info()
df.head(3)
✓ 0.8s
```

```
<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

Imagem 3: Características dos dados

Com isso, é preciso tratar o tipo categórico para que seja possível trabalhar com os campos dentro de um modelo preditivo, visto que, modelos, por geral, apenas trabalham com dados numéricos.

Para que se possa ter uma visão um pouco mais descritiva, é possível utilizar a função `describe()`, cuja já está inclusa dentro do Pandas.

Exemplo de dados estatístico que são possíveis obter com esta função são: média dos valores, valor mínimo, valor máximo e os quartis estatísticos.

```
df.describe()
```

✓ 0.3s

	AGE	RESTINGBP	CHOLESTEROL	FASTINGBS	MAXHR	OLDPEAK
count	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000
mean	53.510893	132.396514	198.799564	0.233115	136.809368	0.887364
std	9.432617	18.514154	109.384145	0.423046	25.460334	1.066570
min	28.000000	0.000000	0.000000	0.000000	60.000000	-2.600000
25%	47.000000	120.000000	173.250000	0.000000	120.000000	0.000000
50%	54.000000	130.000000	223.000000	0.000000	138.000000	0.600000
75%	60.000000	140.000000	267.000000	0.000000	156.000000	1.500000
max	77.000000	200.000000	603.000000	1.000000	202.000000	6.200000

Imagem 4: Utilização da função `describe()` para visões estatísticas

Para o tratamento dos dados categóricos, foi definido um *pipeline*, o qual será responsável por conter os passos necessários para realização de todas as transformações nas variáveis contidas no conjunto de dados estudado. A definição do *pipeline* pode ser vista na imagem a seguir:

```
1 cat_feat = df.select_dtypes(np.object_).columns.tolist()
2 preprocess=Pipeline([
3     ('ct', ColumnTransformer([
4         ('onehot', OneHotEncoder(), cat_feat)
5     ],
6     remainder='passthrough'),
7 ])
8 ])
```

Imagem 5: Definição do *Pipeline* para tratamento dos dados

Demais tratamentos nos dados podem ser realizados durante a evolução do estudo e mediante a necessidade.

6. Análise e Exploração dos Dados

A análise de dados deste projeto contará com uma ampla gama de visualizações, no qual, explorará diversos fatores e comportamentos das variáveis estudadas. Será analisado: distribuições, correlações e análise sobre anomalias dos dados, por exemplo.

Além da visualização de dados que é uma etapa fundamental para o entendimento, será testado algumas hipóteses que serão juntamente validadas utilizando testes estatísticos de hipóteses.

6.1. Visualização de dados

6.1.1. Variáveis numéricas

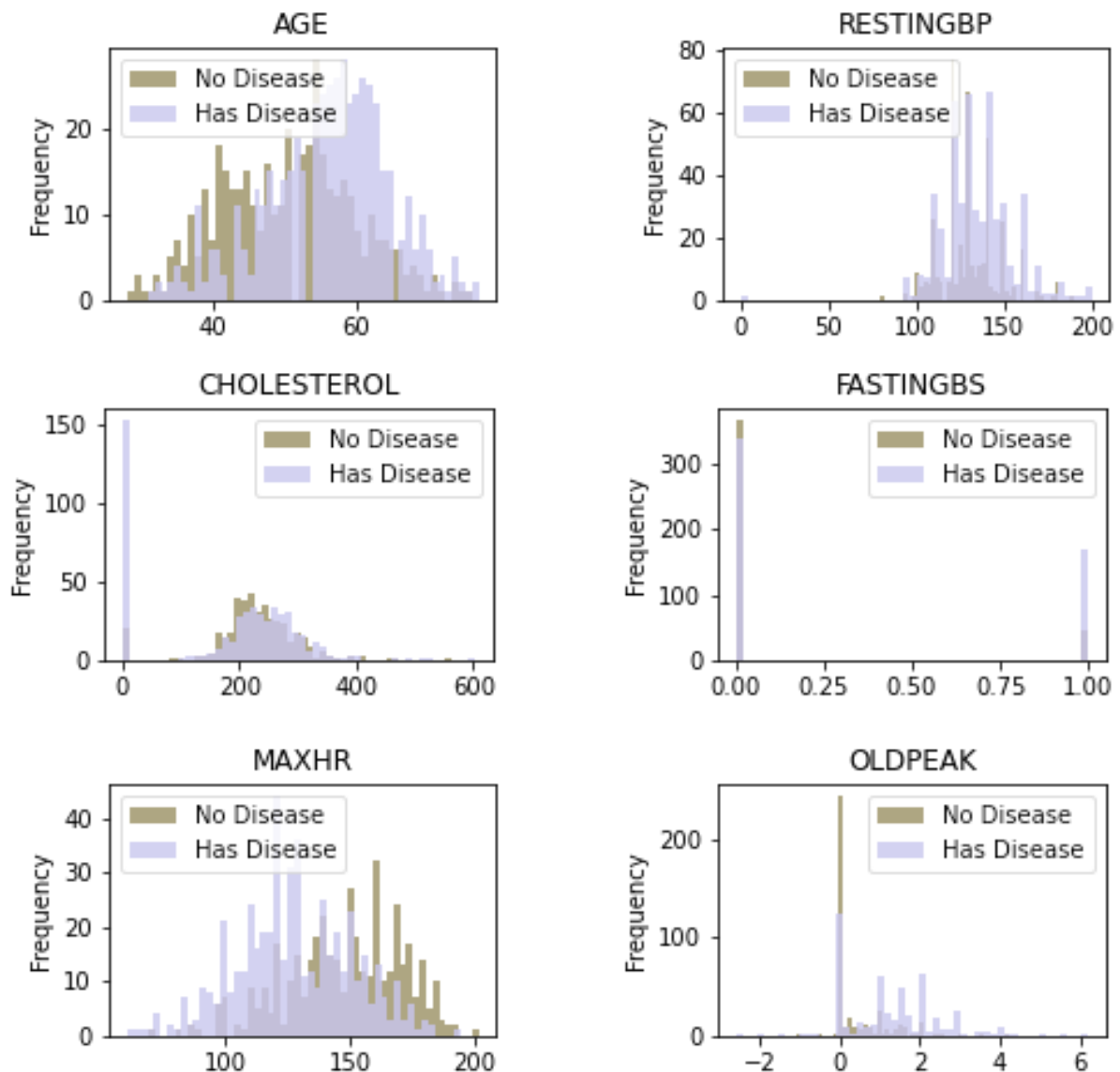


Imagem 6: Distribuições das variáveis numéricas

Através do gráfico de distribuição (histograma) é observado diferentes comportamentos apresentados por cada grupo estudado. Na variável Age, é visto que, o grupo de pessoas com doenças cardíacas está mais concentrado em pessoas de idades sêniores, por exemplo.

Visualmente, é possível observar, que os diferentes grupos, apresentam comportamentos distintos, o que pode ser positivo, pois torna possível a diferenciação do comportamento dos dados em diferentes situações.

Além disso, em algumas variáveis, podem ser observadas certa semelhanças em suas distribuições, quando comparadas com a distribuição Gaussiana, como por exemplo, Age e Cholesterol. Esta hipótese será validada utilizando um teste de normalidade.

6.1.2. Variáveis categóricas

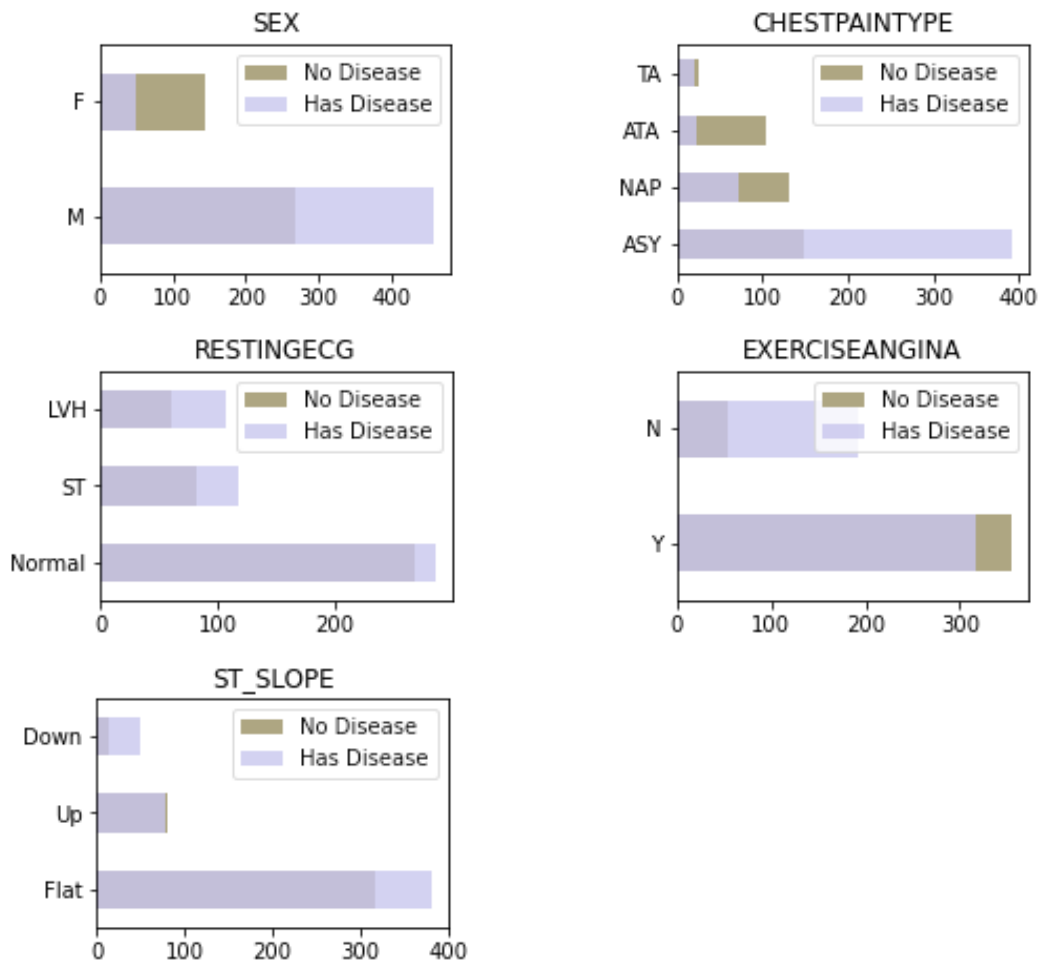


Imagem 7: Distribuições das variáveis categóricas

Através dos gráficos de barras, alguns comportamentos podem ser observados, por exemplo, ao observar os dados por gênero, pode ser notado que pessoas do gênero masculino possuem uma maior probabilidade de desenvolver doenças cardiovasculares em comparação com pessoas do gênero oposto.

Também é observado a forte predominância de certas categorias em algumas variáveis, tais como:

- Sex, onde o gênero masculino é dominante no conjunto de dados.
- ChestPainType, onde pacientes com dores assintomáticas em maioria tiveram problemas no coração.
- ST_SLOPE, onde a maioria das pessoas possuem o segmento de ST na horizontal.

6.1.3. Visualização em pares

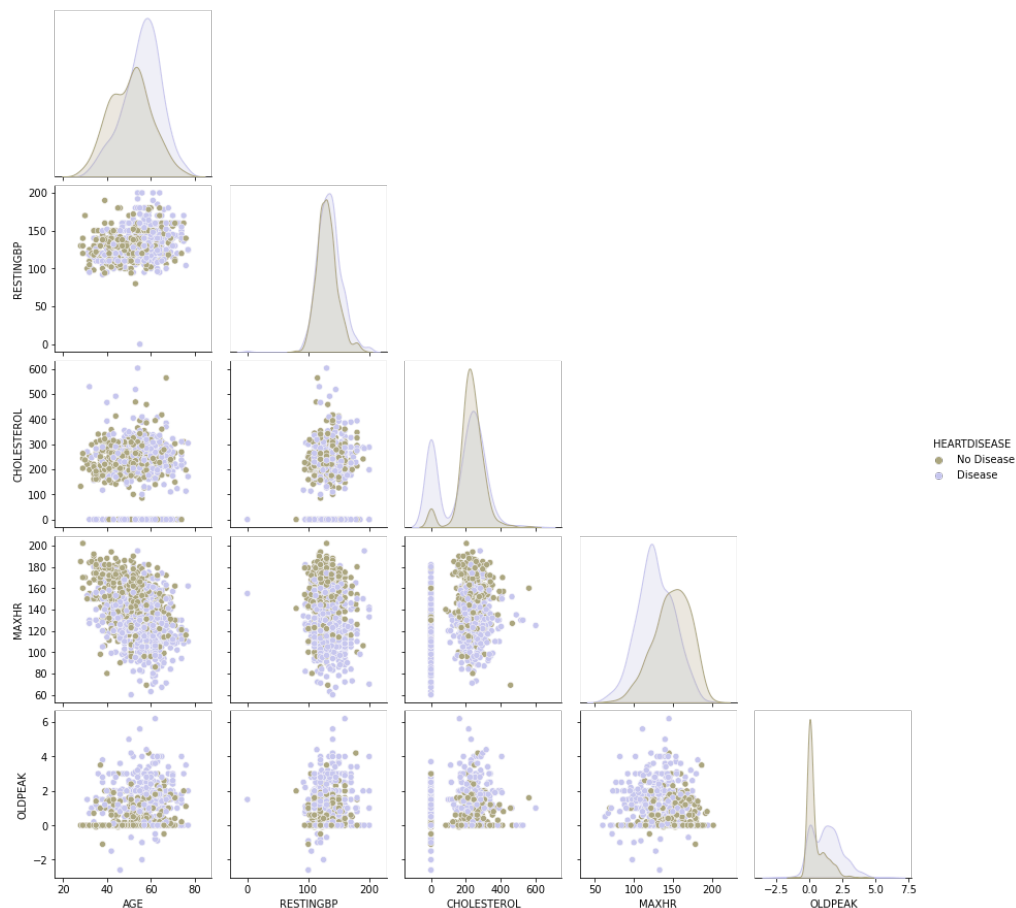
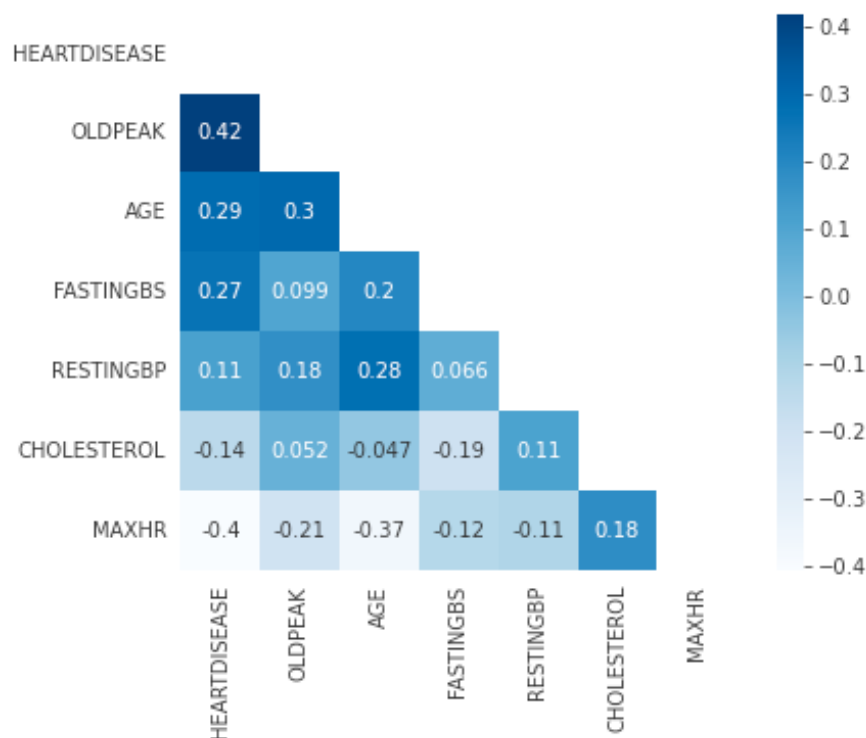


Imagem 8: Gráfico de dispersão por todas as variáveis contínuas

Gráficos de dispersões (*scatters plots*) são muito utilizados para mostrar o comportamento dos dados correlacionando duas variáveis.

Nos gráficos acima, é exibido todas as combinações possíveis entre as variáveis contínuas com as cores sendo caracterizadas pela incidência ou não da variável alvo. Interpretando os gráficos, não há uma correlação expressa que possa ser significativa para a análise. Entretanto, é observado que a distribuição de algumas variáveis no eixo y possui uma separabilidade entre os pontos que são responsáveis pela classe positiva (*Disease*) da classe negativa (*No Disease*).

6.1.4. Mapa de calor para correlação dos dados



A fim de explorar mais a fundo as correlações entre as variáveis, observamos que as únicas correlação aparentes entre os preditos, são as correlações com a variável alvo. Portanto, é confirmado que não há alta correlação entre as variáveis preditoras.

6.1.5. Informação Mútua

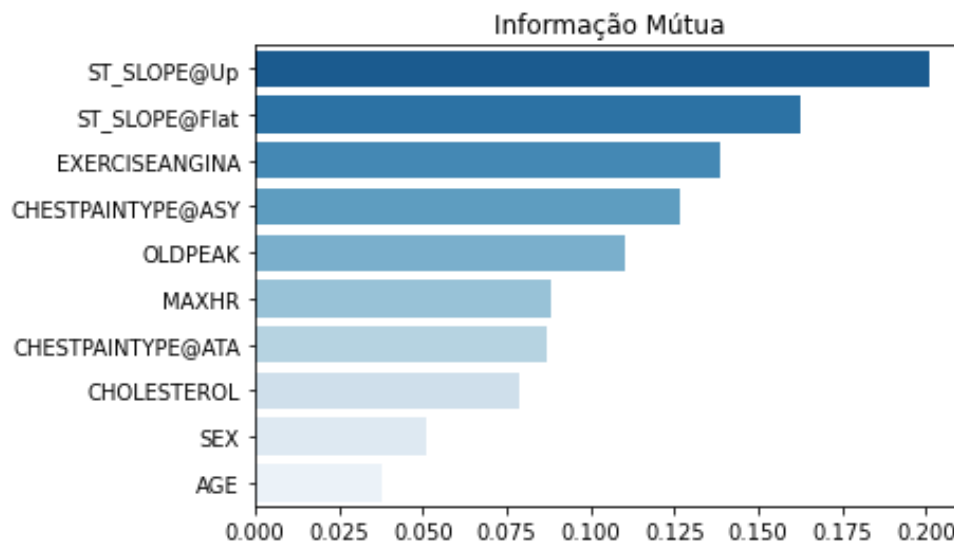


Imagem 10: Gráfico de informação de mútua para as variáveis categóricas e numéricas

Foi executado o algoritmo de informação mútua que visa verificar a dependência das variáveis preditoras perante a variável alvo. No gráfico acima, constam os 10 preditores com os maiores valores.

Variáveis que estão relacionadas ao estado clínico do paciente, por exemplo, dados que apontam para o tipo de comportamento que o gráfico ECG está demonstrando, aparentam ter mais importância do que quando comparadas com gênero ou idade que estão relacionadas com dados demográficos.

6.1.6. Análise de anomalias

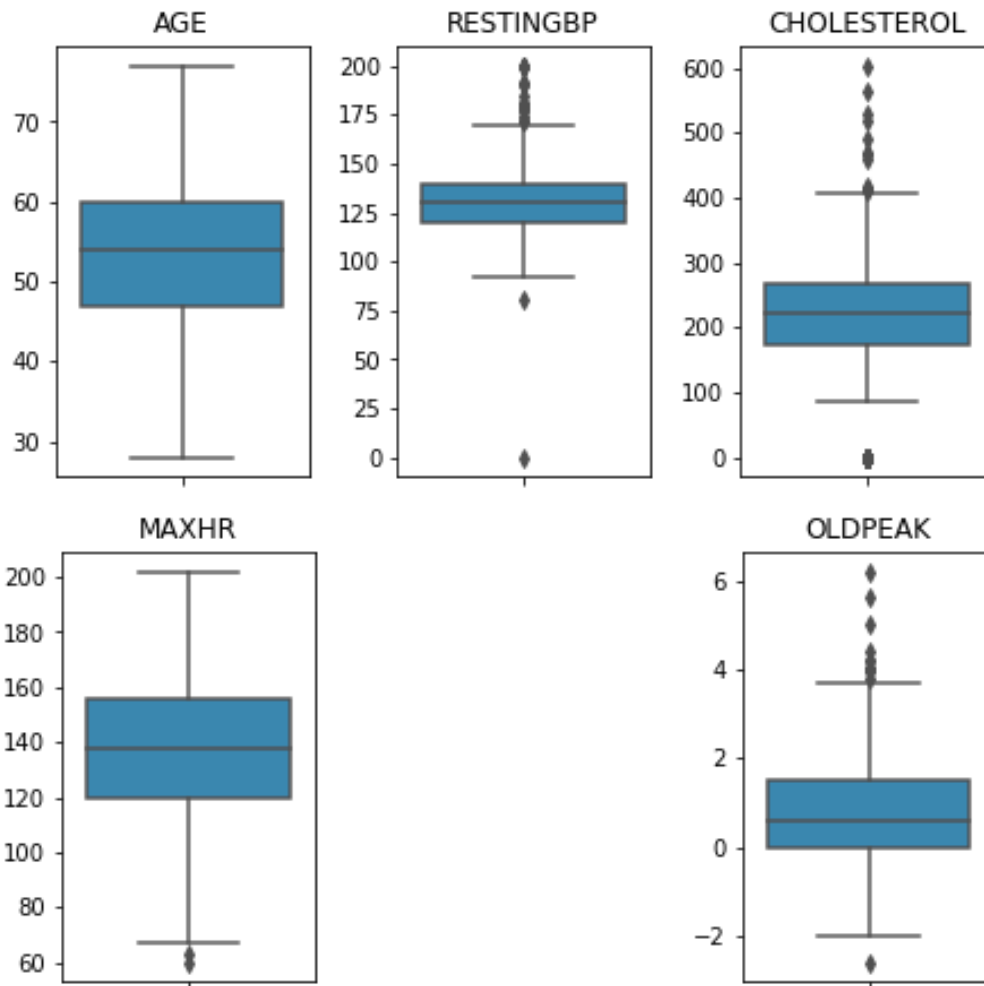


Imagem 11: Gráficos *Box Plot* para detecção de anomalias

Na figura acima, são encontradas anomalias que são representadas por pontos da cor preta e são calculadas seguindo o teste de Tukey. Analisando pelo conhecimento de negócio, valores 0 (zero) nas variáveis de *Cholesterol* e *RestingBP* serão considerados como ruídos e deverão ser tratados na etapa 7. Já os valores que estão acima do limite superior, não há evidências o suficiente para considerarmos ruídos ou erros de registro, por tanto, serão mantidos na análise.

6.1.7. Visualização Probabilística

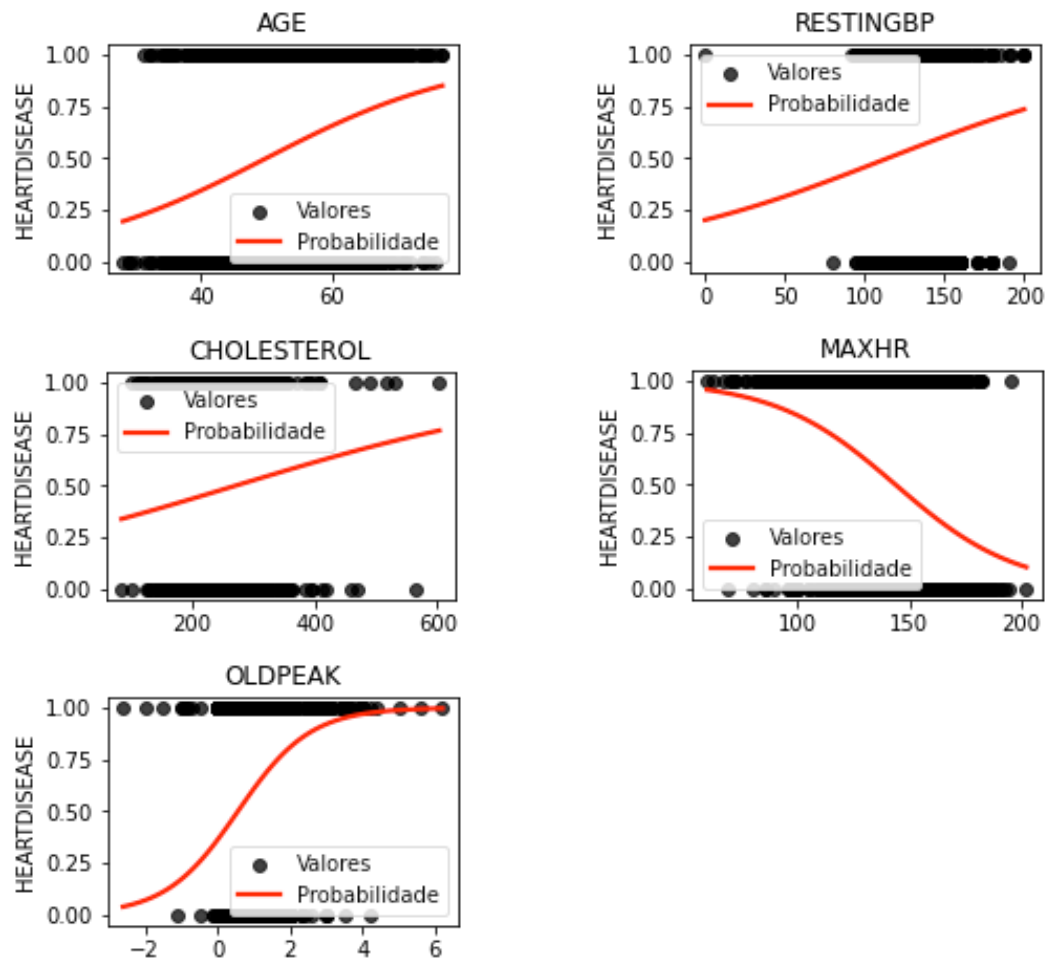


Imagem 12: Visualização probabilística utilizando regressão logística

As visualizações probabilísticas foram criadas utilizando um modelo de regressão logística que é treinado utilizando cada variável de forma separada.

A variável MAXHR traz uma correlação inversa para a probabilidade. Ao interpretar a informação é observado que no momento que foi realizado o exame, quanto maior estivesse o batimento cardíaco, menor a probabilidade do paciente sofrer uma parada cardíaca.

Observando a variável OLDPEAK, é visto uma forte correlação positiva chegando a trazer a probabilidade até os 100%.

6.2. Testes de Hipóteses

6.2.1. Teste de Normalidade

Testar a normalidade de uma determinada variável é importante, visto que, na etapa de tratamentos de dados, alguns tratamentos como, por exemplo, *Scaling* ou *Normalizing*, são melhores aplicados quando se tem conhecimento de qual a distribuição do dado a ser trabalhado. (Toward AI, 2019)

Para se testar a hipótese anteriormente levantada de que algumas das variáveis poderiam seguir a distribuição normal, será utilizado o teste estatístico de hipótese *Kolmogorov-Smirnov*.

Como todo teste de hipótese, este teste é composto por duas hipóteses, são elas:

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

Para critério de aceite da hipótese alternativa (H_1) e, respectivamente a rejeição da hipótese nula (H_0), foi utilizado um valor para *alpha* de 5%.

Efetivamente, para que a hipótese nula seja rejeitada, o *p-value* necessita ser menor do que o valor *alpha* que foi determinado.

Abaixo consta a tabela contendo os resultados do teste.

Variável preditora	Resultado	p-value
Age	Rejeitado H_0	0
RestingBP	Rejeitado H_0	0
Cholesterol	Rejeitado H_0	0
MaxHR	Rejeitado H_0	0
OldPeak	Rejeitado H_0	4.2004655072318036e-200

Tabela 3: Resultados do teste de normalidade

Analisando a tabela, é observado que todas as variáveis contínuas tiveram a hipótese nula (H_0) rejeitada. Portanto, estatisticamente, não há evidências o

suficiente para seja aceito que as variáveis aplicadas ao teste sigam verdadeiramente uma distribuição normal (Gaussiana).

A seguir, o trecho de código escrito na linguagem *Python* e utilizando a biblioteca *SciPy* para a execução e obtenção dos resultados, anteriormente, demonstrados acima:

```

1  from scipy.stats import kstest
2  from scipy import stats
3
4  numeric_columns = df.select_dtypes(np.number).columns
5  for col in numeric_columns:
6      pvalue = kstest(df[col], stats.norm.cdf).pvalue
7
8      print('-'*10)
9      print('Variável a ser testada:', col)
10
11     if pvalue < 0.05:
12         print('Hipótese H0 rejeitada!')
13         print('Variável não possui distribuição normal!')
14     else:
15         print('Hipótese H0 Aceita!')
16         print('Variável possui distribuição normal!')
17
18     print('P_value do teste:', pvalue)

```

Imagem 13: Trecho de código utilizado para obtenção dos resultados do teste de normalidade

6.2.2. Teste de tendência central

Este teste é importante para que seja possível validar a hipótese em que as amostras de ambos os grupos se demonstram diferentes. No objetivo de encontrar um comportamento que possa descrever uma eventual doença cardíaca, é importante trabalhar com dados que demonstrem diferenças nos dois cenários estudados.

O teste que será empregado para esta validação será o teste de Mann-Whitney (*Wilcoxon rank-sum test*). Este, 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.

Abaixo a tabela com os resultados do teste:

Variável preditora	Resultado	p-value
Age	Rejeitado H_0	1.805694139029245e-18
RestingBP	Rejeitado H_0	0.0005648075493721535
Cholesterol	Rejeitado H_0	2.2803123635449238e-05
MaxHR	Rejeitado H_0	1.5063588719598379e-34
OldPeak	Rejeitado H_0	6.767845043438792e-37

Tabela 4: Resultados do teste de tendência central

De acordo com o teste de Mann-Whitney, todas as variáveis testadas tiveram a sua hipótese nula (H_0) rejeitada, isto é, as amostras se demonstraram estatisticamente diferentes quando analisados.

A seguir, o trecho de código escrito na linguagem *Python* e utilizando a biblioteca *SciPy* para a execução e obtenção dos resultados, anteriormente, demonstrados acima:

```

1  from scipy.stats import mannwhitneyu
2
3  numeric_columns = df.select_dtypes(np.number).columns
4  for col in numeric_columns:
5      mask = df['HEARTDISEASE'] == 1
6      grupo_positivo = df.loc[mask, col]
7      grupo_negativo = df.loc[~mask, col]
8
9      pvalue = mannwhitneyu(grupo_positivo, grupo_negativo).pvalue
10
11     print('-'*10)
12     print('Variável a ser testada:', col)
13
14     if pvalue < 0.05:
15         print('Hipótese H0 rejeitada!')
16         print('Amostras semelhantes!')
17     else:
18         print('Hipótese H0 Aceita!')
19         print('Amostras diferentes!')
20
21     print('P_value do teste:', pvalue)

```

Imagem 14: Trecho de código utilizado para obtenção dos resultados do teste de tendência central

6.2.3. Teste de associação

Com o sentido de verificarmos se existe associações das variáveis categóricas com os grupos estudados. O teste de qui-quadrado (*chi-squared*) utiliza da tabela associativa de contingência para calcular sua estatística.

As hipóteses que constituem o teste, são as seguintes:

- H_0 : Não há associação entre os grupos.
- H_1 : Existe associação entre os grupos.

Abaixo a tabela com os resultados do teste:

Variável preditora	Resultado	p-value
CHESTPAINTYPE@ASY	Rejeitado H_0	8.629373889117524e-55
CHESTPAINTYPE@ATA	Rejeitado H_0	1.1525973044710746e-33
CHESTPAINTYPE@NAP	Rejeitado H_0	1.8597575246926209e-10
CHESTPAINTYPE@TA	Aceitado H_0	0.13157675122814316
RESTINGECG@LVH	Aceitado H_0	0.8095282584405754
RESTINGECG@Normal	Rejeitado H_0	0.006790624252570402
RESTINGECG@ST	Rejeitado H_0	0.0025072900984188925
ST_SLOPE@Down	Rejeitado H_0	0.0003421785772644254
ST_SLOPE@Flat	Rejeitado H_0	8.906496025911056e-63
ST_SLOPE@Up	Rejeitado H_0	1.0284928842944619e-78
SEX	Rejeitado H_0	4.5976174508091635e-20
EXERCISEANGINA	Rejeitado H_0	2.907808387659878e-50

Tabela 5: Resultados do teste de associação

Ao contrário do outro teste aplicado nas variáveis contínuas, o teste de qui-quadrado rejeitou duas variáveis: **CHESTPAINTYPE@TA** e **RESTINGECG@LVH**.

Portanto, estas categorias não demonstram evidências o suficiente para acreditar que possuam associações com os grupos estudados e deverão ser retiradas do conjunto de dados.

A seguir, o trecho de código escrito na linguagem *Python* e utilizando a biblioteca *SciPy* para a execução e obtenção dos resultados, anteriormente, demonstrados acima:

```

1  from scipy.stats import chi2_contingency
2
3  hot_encoded = [col for col in df_tratado.columns if '@' in col]
4  for col in hot_encoded + binary_class:
5
6      ctab = pd.crosstab(df_tratado[col], df_tratado['HEARTDISEASE'])
7      _, pvalue, _, _ = chi2_contingency(ctab)
8
9      print('-'*10)
10     print('Variável a ser testada:', col)
11     if pvalue < 0.05:
12         print('Hipótese H0 rejeitada!')
13         print('Há associação entre os grupos!')
14     else:
15         print('Hipótese H0 Aceita!')
16         print('Não há associação entre os grupos!')
17
18     print('P_value do teste:', pvalue)

```

Imagem 15: Trecho de código utilizado para obtenção dos resultados do teste de associações

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

Preparar os dados para o modelo preditivo é um passo importante, tendo em vista que muitas vezes os dados possam estar com ruídos, formatações diferentes e até mesmo necessitando ser transformados em algarismos numéricos.

O tipo de tratamento deve levar em conta o tipo de algoritmo preditivo que se pretende utilizar. Por exemplo, se forem testados modelos que fazem uso de linearidade ou distancia entre pontos, é importante que seja realizada uma padronização (*Standard Scaling*) ou normalização (*Min Max Scaler*) dos dados para que não seja inserido um viés.

Nos próximos tópicos será descrito cada tratamento que for utilizado nos dados estudados, bem como seu propósito de uso.

7.1. Separação dos dados

A separação dos dados é muito importante quando o objetivo é garantir a performance do modelo que está sendo criado. Uma boa prática para isto, é reservar pelo menos 20% dos dados disponíveis para teste e utilizar 80% para aprendizado do algoritmo ou ainda utilizar a técnica de *cross-validation* para ter múltiplas quebras entre dados de treino e teste.

Utilizando a biblioteca *Scikit-learning* será aplicada a função *KFold* que será a responsável por realizar a separação entre treino e teste, essa etapa pode ser vista abaixo:

```
1 X = df.drop(columns='HEARTDISEASE')
2 y = df['HEARTDISEASE']
3
4 kf = KFold(n_splits=5, shuffle=True, random_state=42)
```

Imagem 16: Trecho de código que visa separar os dados em Treino e Teste

7.2. Pipeline de tratamento de dados

Pensando na agilidade e facilidade da implementação de tratamentos adicionais nos dados estudados, é proposto a utilização da classe *Pipeline* pela implementação do *Scikit-learning*, a qual possibilita a inclusão de qualquer classe que implemente os métodos *fit* e *transform*, herdados das classes *Base estimators* e *Transform mixing*.

Abaixo um exemplo de código para criação do pipeline de preparação dos dados:

```
1 preprocess=Pipeline([
2     ('procddata', ProcessData(trans_columns=trans_columns,
3                               clean_inf=True,
4                               clean_sup=False,
5                               outliers_value='Tukey')),
6     ('ct', ColumnTransformer([
7         ('onehot', OneHotEncoder(), cat_feat)
8     ],
9     remainder='passthrough'),
10 ])
11 ])
```

Imagem 17: Trecho de código referente a criação do *pipeline*

7.2.1. Classe criada para tratamento de outliers

Para o tratamento dos valores *outliers*, foi desenvolvida uma classe que possibilita a escolha de quais *outliers* serão tratados em determinada variável. Além disso, é armazenado os valores utilizados, caso seja escolhido o preenchimento pela média, por exemplo. Desta forma, evitamos o problema de *Data Leakage*, pois caso

seja utilizado técnicas de treinamentos, como *cross-validation*, as médias serão calculadas em cada *fold*. Sendo assim, um *fold* não terá conhecimento sobre os dados de outras separações. Garantindo assim, uma maior confiança nas performances que o modelo terá ao verificarmos a performance dele.

Abaixo um exemplo de código para configuração das colunas que serão enviadas à classe:

```

1 trans_columns = {
2     'CHOLESTEROL':{
3         'limit_inf':np.mean,
4         'limit_sup':None
5     },
6     'RESTINGBP':{
7         'limit_inf':np.mean,
8         'limit_sup':None
9     }
10 }
```

Imagem 18: Trecho de código que visa configurar a remoção de *outliers*

7.3. Balanceamento de classes

Durante o estudo, foi identificado que não há a necessidade da realização de técnicas de balanceamento de classes, como: *undersampling* ou *oversampling*.

Balanceamento de Classes

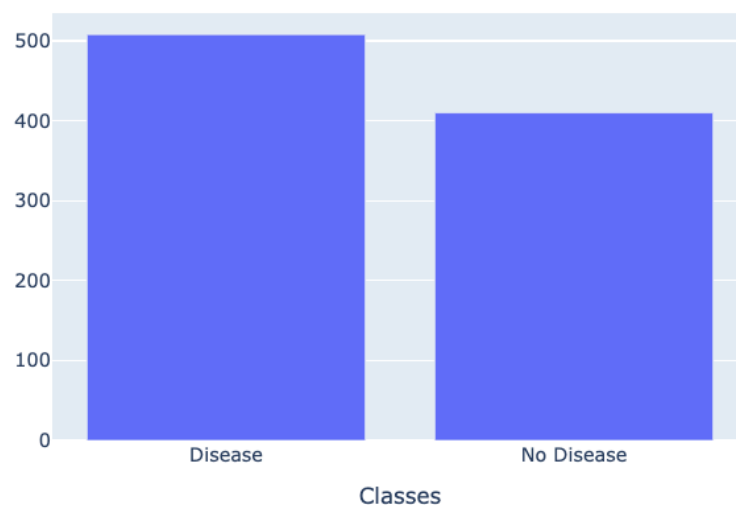


Imagem 19: Verificação do balanço das classes

8. Aplicação de Modelos de Aprendizado de Máquina

A etapa de modelagem é compreendida como a fase onde há o objetivo de testar diferentes modelos e analisar seus resultados. Assim, espera-se que seja possível encontrar o algoritmo preditivo que melhor responda a necessidade do problema a ser resolvido.

O cenário de testes será composto de três algoritmos de Machine Learning, são eles: *XGBoost*, *RandomForest* e *DecisionTree*. Estes modelos podem ser encontrados dentro da biblioteca do *scikit-learning* e são baseados em estruturas de árvores de decisão.

Por serem algoritmos parametrizáveis, contamos com o benefício de poder realizar a técnica chamada de *model tuning* onde é possível ajustar cada parâmetro que compõem o modelo a fim de ter uma melhora de performance quando realizado as suas classificações.

8.1. Pipeline de processamento de dados

A fim de ter uma maior praticidade na etapa de treino & teste dos modelos, é apresentado como solução a utilização de *pipelines* que conterão todo o fluxo de trabalho começando no processamento dos dados até a execução de cada modelo preditivo.

Abaixo, consta o código encarregado de definir o pipeline que realizará o processamento dos dados:

```

1 preprocess=Pipeline([
2     ('procdata', ProcessData(trans_columns=trans_columns,
3                             clean_inf=True,
4                             clean_sup=False,
5                             outliers_value='Tukey')),
6     ('ct', PandasColumnTransformer([
7         ('onehot', OneHotEncoder(handle_unknown='ignore'), cat_feat)
8     ],
9     remainder_cols='passthrough'),
10    ),
11    ('dropfeatures', FunctionTransformer(drop_features,
12                                         kw_args={'cols_todrop':rejected_mann + rejected_chi}))
13 ])
```

Imagem 20: Código do pipeline de processamento de dados

Em ordem:

- **ProcessData**: Classe responsável pelo processamento de outliers;

- **PandasColumnTransformer**: Classe definida que herda *ColumnTransformer* incluída na biblioteca *scikit-learning*. Sua construção se dá para que o tratamento realizado pelo **OneHotEncoder** tenha como retorno um **Pandas-DataFrame**;
- **FunctionTransformer**: Classe contida na biblioteca *scikit-learning* que tem como objetivo tornar possível a utilização de funções dentro de um **pipeline**.

8.2. Pipeline de modelos

Com o *pipeline* de processamento de dados definido, é então, criado o *pipeline* que conterá de fato os modelos a serem testados, assim como, seus parâmetros de busca que será utilizado no momento do *tuning*.

A baixo consta o código responsável pela definição:

```

1 spw = y[y==0].shape[0] / y[y==1].shape[0]
2 search_models=[
3     ('xgb', Pipeline([
4         ('xgb', XGBClassifier(random_state=42,
5                               scale_pos_weight=spw))
6     ]),
7
8     {
9         "estimator__xgb__learning_rate":Real(1e-2, 1),
10        "estimator__xgb__max_depth":Integer(1, 8),
11        "estimator__xgb__min_child_weight":Integer(1,4),
12        "estimator__xgb__subsample":Real(1e-2, 1),
13        "estimator__xgb__colsample_bytree":Real(1e-2, 1)
14    }),
15
16    ('rfc', Pipeline([
17        ('rfc', RandomForestClassifier(random_state=42))
18    ]),
19
20    {
21        "estimator__rfc__n_estimators":Integer(100, 500),
22        "estimator__rfc__max_depth":Integer(1, 8),
23        "estimator__rfc__min_samples_split":Real(1e-1, 1)
24    }),
25
26    ('dtc', Pipeline([
27        ('dtc', DecisionTreeClassifier(random_state=42))
28    ]),
29
30    {
31        "estimator__dtc__max_depth":Integer(1, 8),
32        "estimator__dtc__min_samples_split":Real(1e-1, 1)
33    }),
34 ]

```

Imagem 21: Definição do pipeline junto com os modelos e parâmetros para tuning

No código, é criada uma lista que será composta de *tuplas* e cada posição terá uma finalidade. Por exemplo: (nome_do_modelo, pipeline, espaços_de_buscas).

8.3. Execução do tuning do modelo

Para a execução do *tuning* dos modelos no objetivo de encontrar a melhor solução possível, é utilizado o algoritmo *BayesSearchCV*, incluído na biblioteca *scikit-optimizer*. Este algoritmo realizará a busca dos melhores valores para cada um dos parâmetros disponíveis nos modelos preditivos a partir de espaços de buscas previamente definidos.

A definição do algoritmo será vista na imagem a seguir:

```

1  results=dict(model=[], recall=[], precision=[], f1=[])
2  best_model = list()
3  for m_name, m, ss in search_models:
4      model=Pipeline([
5          ('preprocess', preprocess),
6          ('estimator', m)
7      ])
8
9      model_tuner = BayesSearchCV(
10         estimator=model,
11         cv=kf,
12         search_spaces=ss,
13         n_jobs=-1,
14         n_points=10,
15         n_iter=100,
16         verbose=1,
17         refit=True,
18         random_state=42,
19         scoring='f1'
20     )
21
22     print('Modelo a ser treinado:', m_name)
23     model_tuner.fit(X, y)

```

Imagem 22: Trecho do código necessário para criação

Através da definição da lista de pipelines de modelos na *imagem 21*, será utilizado um laço de repetição para iterar sobre cada pipeline e assim será possível a colheita de resultados separadamente para a escolha do melhor algoritmo com base em sua performance.

O algoritmo de *tuning* **BayesSearchCV**, anteriormente apresentado, receberá cada modelo de forma separada, assim como os parâmetros de busca já definidos. Para o treino, será utilizado a técnica de *cross-validation* que visa de forma exaustiva realizar múltiplas separações dos dados, a fim de ter maior robustez e confiabilidade nas métricas apresentadas.

Após o treino, em cada iteração será realizada a colheita das métricas. Para isto, outro laço de repetição é definido para ser possível obter de forma separada os resultados em em cada quebra que o *cross-validation* realizou.

Este processo pode ser visto na imagem a seguir:

```

25     best_model.append(model_tuner.best_estimator_)
26     y_pred = model_tuner.predict(X_test)
27
28     for train_index, test_index in kf.split(X, y):
29         recall_, precision_, f1_ = list(), list(), list()
30         m = deepcopy(model_tuner.best_estimator_)
31         X_train, y_train = X.loc[train_index], y.loc[train_index]
32         X_test, y_test = X.loc[test_index], y.loc[test_index]
33         m.fit(X_train, y_train)
34         y_pred = m.predict(X_test)
35
36         recall_.append(recall_score(y_test, y_pred))
37         precision_.append(precision_score(y_test, y_pred))
38         f1_.append(f1_score(y_test, y_pred))
39
40     results['model'].append(m_name)
41     results['recall'].append(np.mean(recall_))
42     results['precision'].append(np.mean(precision_))
43     results['f1'].append(np.mean(f1_))

```

Imagem 23: Trecho do código responsável pela colheita de métricas

8.4. Pipeline completo do desenvolvimento do trabalho



Imagem 24: Pipeline do projeto

- **Extração de dados:** A extração de dados é entendida como a etapa mais importante, pois através dela é obtida a matéria-prima do trabalho.
 - **Kaggle:** É uma plataforma aberta que hoje pertence ao Google e possui inúmeras bases de dados que podem ser livremente exploradas por qualquer pessoa.
 - **CSV:** A base de dados trabalhada no desenvolvimento deste projeto é uma base que reúne outras bases de dados coletadas de outros centros medicinais e disponibilizada de forma livre dentro da plataforma Kaggle.
- **Pipeline de Desenvolvimento**
 - **Análise de dados:** esta etapa é a responsável por analisar e provar todas as hipóteses que podem ser criadas durante o desenvolvimento.
 - **Preparação dos dados:** à partir dos dados analisados é possível mapear possíveis tratamentos necessários para ser possível iniciar a etapa de modelagem.
 - **Modelagem e Validação:** é responsável pela estruturação do treinamento e avaliação dos modelos preditivos que se deseja testar.

9. Discussão dos Resultados

A partir do processo anterior de treino e colheita de resultados será utilizado três métricas de avaliação para analisar cada um dos algoritmos testados. As métricas a serem utilizadas serão as seguintes: Recall, Precision, F1 Score.

9.1. Definição das métricas

9.1.1. Matriz de Confusão

A matriz de confusão, no caso de um problema de classificação binária é composta pelos quatro tipos de resultados que se podem ter:

- **Verdadeiro Positivo** (acerto tipo 1): Quando o diagnóstico médico e o resultado dado pelo modelo preditivo convergem à classe positiva do problema estudado: “doença cardíaca”.
- **Falso Positivo** (erro tipo 1): Quando há uma divergência entre o diagnóstico médico e o resultado entregue pelo modelo preditivo. Por exemplo: O modelo classificou o paciente estudado como doença cardíaca, porém o diagnóstico médico foi de paciente saudável.
- **Verdadeiro Negativo** (acerto tipo 2): Quando o diagnóstico médico e o resultado dado pelo modelo preditivo convergem à classe negativa do problema estudado: “paciente saudável”.
- **Falso Negativo** (erro tipo 2): Quando o há uma divergência entre o diagnóstico médico e o resultado entregue pelo modelo preditivo. Por exemplo: O modelo classificou o paciente estudado como “paciente saudável”, porém o diagnóstico médico foi de “doença cardíaca”.

Abaixo pode ser observado a tabela representativa da matriz de confusão:

		Valor Predito y^{\wedge}	
		Negativo (0)	Positivo (1)
Valor Real y	Negativo (0)	Verdadeiro Negativo	Falso Positivo
	Positivo (1)	Falso Negativo	Verdadeiro Positivo

Imagem 24: Representação da Matriz de Confusão

Utilizando a matriz de confusão é possível calcular diversas métricas, porém para este estudo será levado em conta apenas as citadas anteriormente: Recall, Precision e F1 Score.

9.1.2. Recall

Recall é dado pela razão entre eventos preditos corretamente pelo modelo preditivo pela quantidade de eventos existentes. Por exemplo: em uma base onde há 10 pacientes com o status de “doença cardíaca”, caso o modelo testado realize a classificação de 5 pacientes corretamente, nossa métrica Recall será de 50%, pois de 10 possível classificações, o modelo classificou apenas 50% do total. Este cálculo pode ser mostrado com a seguinte formula:

$$Recall = \frac{VerdadeiroPositivo}{VerdadeiroPositivo + FalsoNegativo}$$

9.1.3. Precision

A precisão se define pela razão entre a quantidade de classificações corretas pela quantidade de classificações realizadas somada do total de acertos realizados pelo modelo preditivo. Por exemplo: em uma base onde há 10 pacientes e apenas 5 estão com o status de “doença cardíaca”, o modelo classificou 5 pacientes corretamente com o status de “doença cardíaca”, porém para esta classificação, o

modelo também classificou os outros 5 pacientes com o mesmo status, porém erroneamente. Este cálculo pode ser mostrado com a seguinte formula:

$$Precision = \frac{VerdadeiroPositivo}{VerdadeiroPositivo + FalsoPositivo}$$

9.1.4. F1 Score

Quando o assunto é sobre avaliação de modelos preditivos, é necessário, através do entendimento do negócio, qual métrica se encaixa melhor em determinada situação.

O objetivo deste trabalho é a tentativa de classificar novos pacientes com possíveis doenças cardiovasculares. Portanto, entende-se que ambos, a precision e o recall, se fazem muito importantes como métricas de avaliação.

Pensando nisso, uma possibilidade é utilizar uma outra métrica chamada de **F1 Score**, a qual é calculada utilizando uma média harmonica entre a **precision** e o **recall**.

A formula pode ser encontrada a seguir:

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

9.2. Análise dos modelos

Abaixo é possível encontrar a tabela com os três modelos testados junto com os valores de cada métrica utilizada para teste.

Algoritmo	Recall	Precision	F1 Score
XGBoost	94,56%	84,46%	89,23%
Random Forest	92,39%	77,98%	84,57%
Decision Tree	89,13%	78,09%	83,24%

Ao analisar a tabela acima, o modelo que obteve a maior performance e respectivamente melhor conseguiu generalizar o problema de um modo que conseguiu prever novos casos com baixa taxa de erro, foi o algoritmo XGBoost que obteve as seguintes métricas:

- Recall: 94,56%
- Precision: 84,46%
- F1 Score: 89,23%

10. Conclusão

Durante o desenvolvimento deste trabalho, o objetivo foi entender a possibilidade de desenvolvimento de um algoritmo de inteligência artificial que fosse possível, através de um treinamento prévio, realizar classificações de novos pacientes. Além disso, foi desejado o entendimento através dos dados, das principais variáveis que contribuíssem para uma doença cardiovascular.

Regressando a etapa das análises, algumas variáveis se demonstram com alta capacidade de indicar uma possível doença cardíaca, como por exemplo: o alto índice de colesterol e um valor alto da curva de OldPeak.

Como dificuldades encontradas no desenvolvimento de todo o trabalho, a falta de profissionais formados na área da saúde é algo que acaba por impactar, principalmente durante as análises e conclusões que foram pensadas durante o desenvolvimento.

Pensando em melhorias, o envolvimento de profissionais da área da saúde, seria de extrema importância a fim de revisar todo o trabalho desenvolvido e, também sugerir novas ideias para exploração.

Em questão de transformar o trabalho desenvolvido em um produto que possa ter capacidade de “escalabilidade” para que seja possível executar em outras bases de dados ou com quantidades de dados massivamente maiores, se faz necessário utilização de plataformas que servirão como base para a execução, por exemplo a IBM Cloud, onde é possível a criação de arquiteturas próprias para a execução do algoritmo.

11. Links

Este trabalho pode ser encontrado no seguinte repositório hospedado no GitHub, cujo pode ser acessado pelo seguinte link:

https://github.com/BrunoDefante/TCC_Puc_Minas

12. Referências

National Cancer Institute. Can Artificial Intelligence Help See Cancer in New, and Better, Ways?. 2022. Disponível em: <https://www.cancer.gov/news-events/cancer-currents-blog/2022/artificial-intelligence-cancer-imaging>. Acesso em 05/06/2022

IBM. Por que usar a IA na assistência médica? 2022?. Disponível em: <https://www.ibm.com/watson-health>. Acesso em 05/10/2022

Use o coração para vencer as doenças cardiovasculares. Ministério da Saúde do Brasil. 2022?. Disponível em: [https://bvsmms.saude.gov.br/use-o-coracao-para-vencer-as-doencas-cardiovasculares-29-9-dia-mundial-do-coracao/#:~:text=As%20doen%C3%A7as%20cardiovasculares%20\(DCV\)%20s%C3%A3o,de%20Chagas%20e%20amiloidose%20card%C3%ADaca](https://bvsmms.saude.gov.br/use-o-coracao-para-vencer-as-doencas-cardiovasculares-29-9-dia-mundial-do-coracao/#:~:text=As%20doen%C3%A7as%20cardiovasculares%20(DCV)%20s%C3%A3o,de%20Chagas%20e%20amiloidose%20card%C3%ADaca). Acesso em 06/06/2022

How, When, and why should you Normalize / Standardize / Rescale Your Data?. 2019. Disponível em: <https://towardsai.net/p/data-science/how-when-and-why-should-you-normalize-standardize-rescale-your-data-3f083def38ff>. Acesso em 06/06/2022