

INTELIGENCIA ARTIFICIAL CAPAZ DE IDENTIFICAR DOENÇAS CARDIACAS ¹

Francisco Gilnece de Menezes Junior ²

RESUMO

Este artigo aborda a relevância da aplicação de técnicas de Machine Learning na detecção precoce de doenças cardíacas, considerando o atual panorama de saúde global. Em particular, propõe-se a criação de uma inteligência artificial capaz de identificar a presença ou ausência de doenças cardíacas com base em métricas específicas. Destacamos a crescente complexidade e prevalência dessas condições, tornando imperativo o desenvolvimento de abordagens inovadoras para enfrentar os desafios associados. A inteligência artificial proposta representa uma ferramenta promissora para aprimorar a prática clínica, oferecendo uma solução avançada e personalizada para a identificação precoce de anomalias cardíacas e a melhoria dos desfechos para os pacientes.

Palavras-chave: IA, Machine-Learnig, Cardiaco, Coração, Google-Colab.

ABSTRACT

This article addresses the relevance of applying Machine Learning techniques in the early detection of heart diseases, considering the current global health scenario. Specifically, it proposes the development of an artificial intelligence capable of identifying the presence or absence of heart diseases based on specific metrics. We emphasize the increasing complexity and prevalence of these conditions, making it imperative to develop innovative approaches to tackle associated challenges. The proposed artificial intelligence represents a promising tool to enhance clinical practice, providing an advanced and personalized solution for the early identification of cardiac anomalies and the improvement of outcomes for patients.

Keywords: AI, Machine Learning, Cardiac, Heart, Google Colab.

¹ Documento apresentando um Comparativo dos tipos de serviços SaaS e IaaS.

1 INTRODUÇÃO

No contexto da saúde cardíaca contemporânea, a detecção precoce e eficiente de doenças assume uma importância crucial para a prevenção de complicações graves e a promoção do bem-estar. A tecnologia de Machine Learning (ML) surge como uma ferramenta promissora para revolucionar a abordagem diagnóstica e prognóstica dessas condições, representando um avanço significativo na identificação precoce de anomalias cardíacas.

As doenças cardíacas, atualmente, figuram entre as principais causas de morbidade e mortalidade global, demandando estratégias inovadoras para enfrentar os desafios impostos por sua crescente prevalência e complexidade. A detecção precoce de anomalias cardíacas torna-se uma prioridade para intervenções oportunas e personalizadas, visando melhorar a qualidade de vida dos indivíduos afetados.

Este artigo propõe a criação de uma inteligência artificial fundamentada em técnicas avançadas de Machine Learning, a qual será desenvolvida com o objetivo específico de identificar a presença ou ausência de doenças cardíacas com base em métricas específicas. A capacidade desses algoritmos em analisar vastos conjuntos de informações, identificar padrões e gerar previsões precisas será explorada como uma abordagem inovadora para o diagnóstico e a estratificação de risco em pacientes cardíacos.

Diante do atual cenário de saúde cardíaca, a aplicação de inteligência artificial torna-se uma promissora aliada na identificação precoce de patologias, proporcionando uma abordagem mais eficaz para a tomada de decisões clínicas. Este artigo busca, assim, contribuir para o avanço da prática médica, apresentando uma solução inovadora que visa não apenas diagnosticar, mas também personalizar o cuidado e melhorar os desfechos para os indivíduos em risco de doenças cardíacas.

2 BASE DE DADOS

Obtemos uma base de dados do Kaggle.com, onde foi feito o download do conjunto de dados Cardiocascular Disease Dataset, onde estão presentes os seguintes dados:

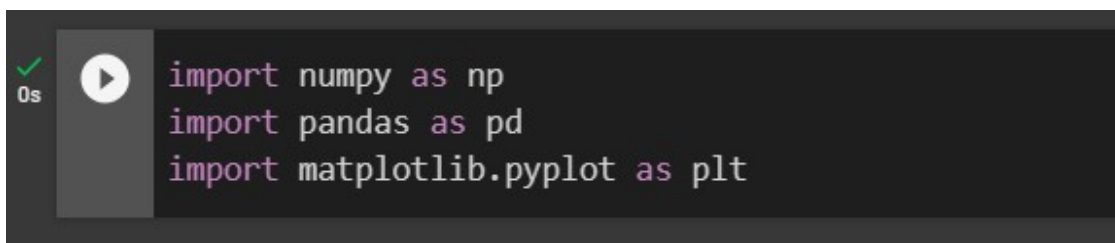
- Idade
- Altura
- Peso

- Gênero
- Pressão arterial sistólica
- Pressão arterial diastólica
- Colesterol
- Glicose
- Tabagismo
- Ingestão de álcool
- Atividade Física

Além disso, há uma variável binária indicando se o usuário possui doença cardiovascular, representada por zeros e uns. Utilizaremos todas as variáveis, exceto a última, para avaliar se a pessoa analisada apresenta algum problema cardíaco. A última variável será crucial como indicativo da presença ou ausência de doença cardiovascular.

Logo após, foi aberto o Google Colab e foram importadas algumas bibliotecas para poder trabalhar e gerenciar a base de dados adquirida anteriormente.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```



No Google Colab, você pode importar arquivos de diversas maneiras. Uma das formas mais comuns é utilizando a biblioteca `google.colab`, que fornece a função `files`. Aqui estão os passos básicos para importar arquivos para o Google Colab:

1. Montar o Google Drive:

Se os arquivos estiverem armazenados no seu Google Drive, você pode

montá-lo no Colab para acessar os arquivos. Execute o seguinte código e siga as instruções para autenticar e montar o Google Drive:

```
from google.colab import drive
drive.mount('/content/drive')
```

2. Upload de arquivos diretamente para o Colab:

Você também pode fazer upload de arquivos diretamente para o Colab. Use o seguinte código para selecionar e fazer o upload de arquivos:

```
from google.colab import files
uploaded = files.upload()
```

Isso abrirá uma opção para você selecionar arquivos do seu dispositivo local e fazer upload para o ambiente do Colab.

3. Acesso ao sistema de arquivos local:

Se você já tem os arquivos em algum lugar acessível pelo sistema de arquivos do Colab (por exemplo, se você faça o download do arquivo diretamente da internet), você pode simplesmente acessar o caminho correspondente.

Exemplo de leitura de um arquivo CSV:

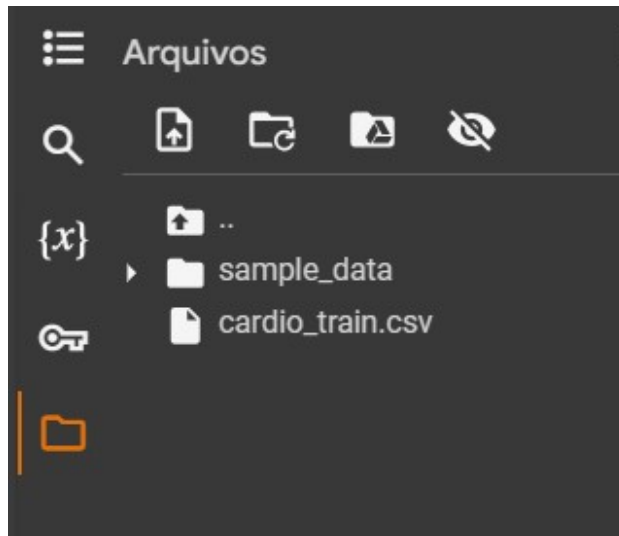
```
df = pd.read_csv('/content/path/do/arquivo/arquivo.csv')
```

4. Baixar arquivos do Colab para o seu dispositivo local:

Se você precisa baixar um arquivo do Colab para o seu dispositivo local, você pode usar o seguinte código:

```
from google.colab import files
files.download('/content/path/do/arquivo/arquivo.csv')
```

Porém o método escolhido foi o Upload de arquivo, onde foi upado a base de dados `cárdio_train.csv`.



3 TRABALHANDO DADOS

Logo após realizar a importação da base para nosso projeto no Google Colab, vamos começar a ler nosso conjunto de dados, lembrando de informar qual o tipo de separação para facilitar no entendimento dos dados, pois estamos utilizando um arquivo CSV, e informar que o índice de coluna é igual a zero para obrigar o compilador a começar pelo primeiro usuário da lista de dados

```
df_cardio = pd.read_csv("cardio_train.csv", sep=";", index_col=0)
```

Logo após iniciar a leitura do arquivo e identificar os parâmetros para facilitar na visualização, chamamos a base de dados e identificamos que ele possui 70.000(setenta mil) linhas e 12 colunas

```

<bound method DataFrame.info of
id
0      18393      2      168      62.0      110      80      1      1      0
1      20228      1      156      85.0      140      90      3      1      0
2      18857      1      165      64.0      130      70      3      1      0
3      17623      2      169      82.0      150     100      1      1      0
4      17474      1      156      56.0      100      60      1      1      0
...      ...      ...      ...      ...      ...      ...      ...      ...
99993  19240      2      168      76.0      120      80      1      1      1
99995  22601      1      158     126.0      140      90      2      2      0
99996  19066      2      183     105.0      180      90      3      1      0
99998  22431      1      163      72.0      135      80      1      2      0
99999  20540      1      170      72.0      120      80      2      1      0

      alco  active  cardio
id
0         0        1        0
1         0        1        1
2         0        0        1
3         0        1        1
4         0        0        0
...      ...      ...      ...
99993     0        1        0
99995     0        1        1
99996     1        0        1
99998     0        0        1
99999     0        1        0

[70000 rows x 12 columns]>

```

Nossa base de dados escolhida conta com informações de 70.000 pessoas tratando-se, assim, de uma ampla base que pode ser utilizada para o ensino de nossa inteligência artificial na identificação de futuros problemas e na assistência ao seu tratamento

Para verificar se os dados foram carregados corretamente foi utilizado a linha de comando ‘df_cardio.info()’, a fim de garantir que nenhuma das variáveis apresentou-se como tipo String e para assegurar que nenhuma coluna de dados estava ausente, uma vez que todas são uteis para a análise final e o diagnóstico usuário testado

No entanto, optamos por adotar uma abordagem tabular a fim de facilitar a visualização e compreensão dos dados. Isso será realizado mediante a execução do comando ‘df_cardio.head()’, que nos permitirá examinar as primeiras linhas da tabela de dados de maneira mais legível.

Essa estratégia se torna imprescindível para proporcionar uma visão inicial dos registros contidos na base possibilitando uma análise preliminar e facilitando o entendimento dos atributos e valores apresentados. Assim tornará mais eficaz o processo de manipulação e interpretação dos dados contidos no conjunto, contribuindo para uma abordagem mais contextualizada e significativa em nossa investigação

df_cardio.head()

	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
id												
0	18393	2	168	62.0	110	80	1	1	0	0	1	0
1	20228	1	156	85.0	140	90	3	1	0	0	1	1
2	18857	1	165	64.0	130	70	3	1	0	0	0	1
3	17623	2	169	82.0	150	100	1	1	0	0	1	1
4	17474	1	156	56.0	100	60	1	1	0	0	0	0

+ Código + Texto

4 ORGANIZAÇÃO DE DADOS

Nesse momento, buscaremos aprimorar a visualização dos dados por meio de ferramentas específicas, sendo a escolha orientada à biblioteca Plotly, uma das mais reconhecidas e amplamente utilizadas na atualidade.

```
from plotly.subplots import make_subplots
import plotly.graph_objects as go
```

Para Iniciar, procederemos à criação de uma figura que contenha 4 linhas e 1 coluna, utilizando o código a seguir:

```
fig = make_subplots(rows=4, cols=1)
```

Esta etapa é fundamental para propiciar uma análise detalhada da distribuição das variáveis contínuas, as quais podem assumir valores diversos. Tais variáveis englobarão os atributos de idade, peso e as duas pressões apresentadas na base

```
fig.add_trace(go.Box(x=df_cardio["age"] / 365, name="Idade"), row=1, col=1)
fig.add_trace(go.Box(x=df_cardio["weight"], name="Peso"), row=2, col=1)
fig.add_trace(go.Box(x=df_cardio["ap_hi"], name="Pressão Sistólica"), row=3, col=1)
fig.add_trace(go.Box(x=df_cardio["ap_lo"], name="Pressão Diastólica"), row=4, col=1)
```

```
✓ 1s ▶ from plotly.subplots import make_subplots
import plotly.graph_objects as go

fig = make_subplots (rows=4, cols=1)

fig.add_trace(go.Box(x=df_cardio["age"] / 365, name="Idade"), row=1, col=1)
fig.add_trace(go.Box(x=df_cardio["weight"], name="Peso"), row=2, col=1)
fig.add_trace(go.Box(x=df_cardio["ap_hi"], name="Pressão sistólica"), row=3, col=1)
fig.add_trace(go.Box(x=df_cardio["ap_lo"], name="Pressão Diastólica"), row=4, col=1)

fig.update_layout(height=700)
fig.show()
```

É importante destacar que foi ajustado o eixo X para exibir a idade em anos, dado a base de dados original utiliza a unidade de dias para representar a idade do usuário, sendo assim foi feita a divisão por 365 dias para aproximar a idade da pessoa analisada

Por fim, efetuaremos a configuração final de layout da figura que será apresentada e a exibição dos dados obtidos

```
fig.update_layout(height=700)
fig.show()
```

Estas medidas visam a aprimorar a análise exploratória dos dados, contribuindo significativamente para a interpretação e compreensão dos padrões e tendências existentes na base de dados


```

from plotly.subplots import make_subplots
import plotly.graph_objects as go

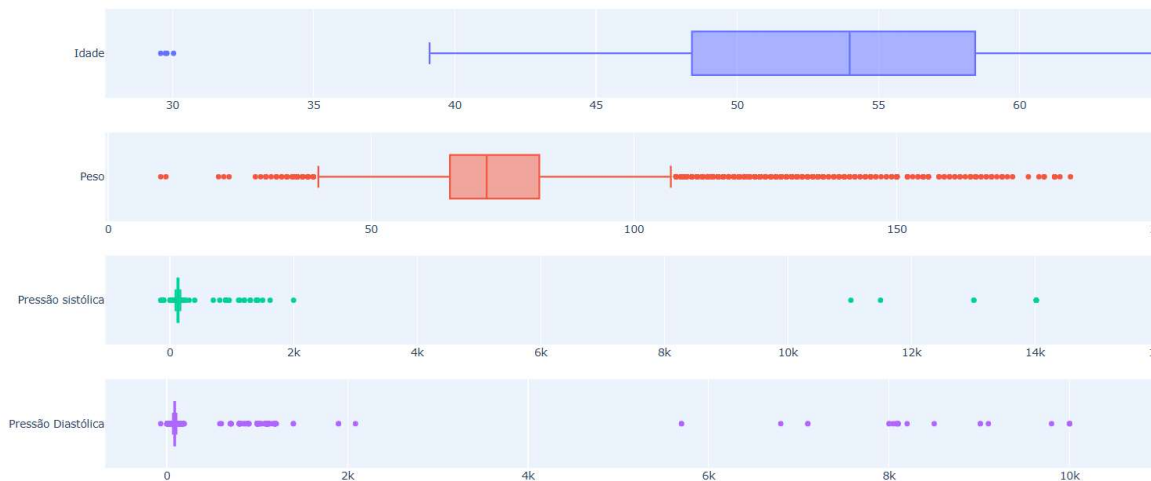
fig = make_subplots (rows=4, cols=1)

fig.add_trace(go.Box(x=df_cardio["age"] / 365, name="Idade"), row=1, col=1)
fig.add_trace(go.Box(x=df_cardio["weight"], name="Peso"), row=2, col=1)
fig.add_trace(go.Box(x=df_cardio["ap_hi"], name="Pressão sistólica"), row=3, col=1)
fig.add_trace(go.Box(x=df_cardio["ap_lo"], name="Pressão Diastólica"), row=4, col=1)

fig.update_layout(height=700)
fig.show()

```

Aqui podemos visualizar a figura gerada com a codagem anterior, onde podemos ver as medias de idade, peso, pressão sistólica e deastólica.



5 VISÃO GLOBAL

A seguir, apresentamos uma visão global da distribuição dos valores, proporcionando uma análise detalhada sobre a divisão de categorias dentro do conjunto de dados

```

from plotly.subplots import make_subplots
fig = make_subplots(rows=2, cols=3)

```

Para proporcionar uma análise detalhada das variáveis categóricas, utilizaremos uma abordagem semelhante à anterior empregada para variáveis contínuas. Desta forma, serão exploradas seis variáveis categóricas.

```

fig.add_trace(go.Bar(y=df_cardio["gender"].value_counts(), x=["Feminino",
"Masculino"], name="Gênero"), row=1, col=1)
fig.add_trace(go.Bar(y=df_cardio["cholesterol"].value_counts(), x=["Normal",
"Acima do Normal", "Muito Acima do Normal"], name="Colesterol"), row=1,
col=2)
fig.add_trace(go.Bar(y=df_cardio["gluc"].value_counts(), x=["Normal", "Acima
do Normal", "Muito Acima do Normal"], name="Glicose"), row=1, col=3)
fig.add_trace(go.Bar(y=df_cardio["smoke"].value_counts(), x=["Não Fumante",
"Fumante"], name="Fumante"), row=2, col=1)
fig.add_trace(go.Bar(y=df_cardio["alco"].value_counts(), x=["Não Alcoólatra",
"Alcoólatra"], name="Alcoólatra"), row=2, col=2)
fig.add_trace(go.Bar(y=df_cardio["active"].value_counts(), x=["Não Ativo",
"Ativo"], name="Ativo"), row=2, col=3)

```

```

[16] from plotly.subplots import make_subplots

fig = make_subplots(rows=2, cols=3)
fig.add_trace(go.Bar(y=df_cardio["gender"].value_counts(), x=["Feminino", "Masculino"], name="Genero"), row=1, col=1)
fig.add_trace(go.Bar(y=df_cardio["cholesterol"].value_counts(), x=["Normal", "acima do normal", "muito acima do normal"],
fig.add_trace(go.Bar(y=df_cardio["gluc"].value_counts(), x=["Normal", "acima do normal", "muito acima do normal"], name="
fig.add_trace(go.Bar(y=df_cardio["smoke"].value_counts(), x=["Não Fumante", "Fumante"], name="Fumante"), row=2, col=1)
fig.add_trace(go.Bar(y=df_cardio["alco"].value_counts(), x=["Não Alcolatra", "Alcolatra"], name="Alcolatra"), row=2, co
fig.add_trace(go.Bar(y=df_cardio["active"].value_counts(), x=["Não Ativo", "Ativo"], name="Ativo"), row=2, col=3)

fig.update_layout(height=700)
fig.show()

```

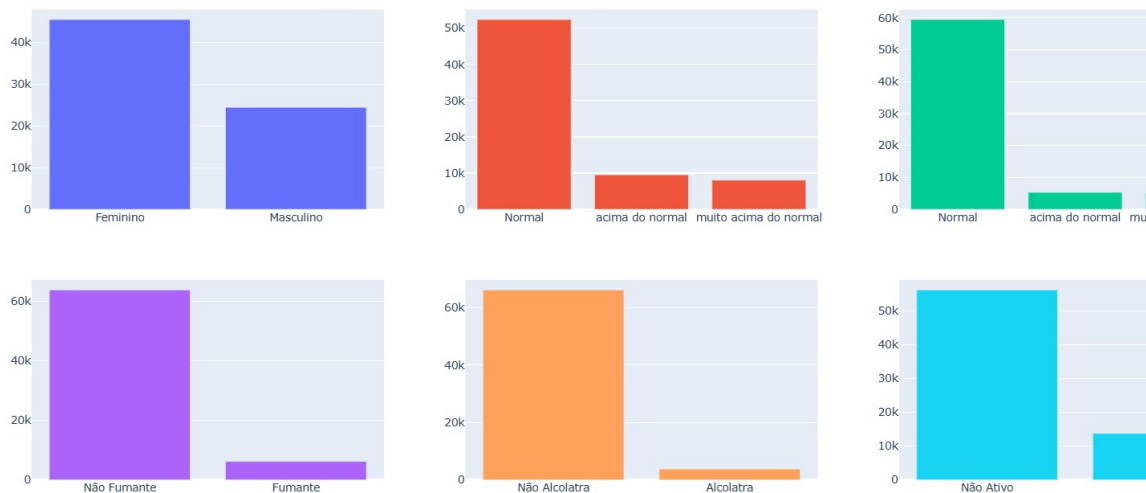
Esse procedimento, aplicado às variáveis categóricas, permite uma melhor compreensão da distribuição dos dados e contribui para identificar padrões e tendências existentes

```

fig.update_layout(height=700)
fig.show()

```

A adoção dessa abordagem visa a aprimorar a exploração e interpretação das variáveis categóricas, enriquecendo a análise exploratória do conjunto de dados. Abaixo podemos visualizar a figura gerada onde podemos ver que a maior parte do publico é do sexo feminino, possui as taxas normais, não são fumantes e nem fazem ingestão de alcool além disso mostra que a grande maioria não pratica nenhum tipo de esporte, sendo assim são pessoas sedentarias!



Por Fim, e não menos significativo, é imprescindível conhecer a proporção de indivíduos que apresentam ou não doença cardiovascular dentro do âmbito desse estudo

```
proporcao_cardiovascular = df_cardio["cardio"].value_counts() /
df_cardio["cardio"].value_counts().sum()
```

A fórmula acima permite calcular e expressar em percentual a proporção de pessoas cardíacas e não cardíacas na base de dados, contribuindo para compreensão mais apurada do perfil cardiovascular do grupo estudado

Este procedimento, além de fornecer informações relevantes sobre a prevalência de doenças cardiovasculares na amostra, constitui um elemento chave para orientar análise mais aprofundadas e estratégicas de intervenção

```
[17] df_cardio["cardio"].value_counts() / df_cardio["cardio"].value_counts().sum()

0    0.5003
1    0.4997
Name: cardio, dtype: float64
```

6 MAPA DE CORRELAÇÃO

A partir de agora vamos relacionar os dados que foram gerados anteriormente com as variáveis categóricas e as variáveis contínuas. Nesse momento, prosseguiremos para a elaboração de um mapa de correlação entre os dados, visando identificar relação estatísticas entre as variáveis em análise. Para realizar essa tarefa, importaremos a biblioteca Seaborn, amplamente utilizada para visualização de dados em Python

```
import seaborn as sns
```

Posteriormente, procederemos à criação do mapa de correlação, utilizando o código a seguir:

```
fig, ax = plt.subplots(figsize=(30, 10))  
sns.heatmap(df_cardio.corr(), annot=True, cmap="RdYlGn")
```

Neste contexto, o método ‘.corr()’ será empregado para calcular a correlação entre todas as variáveis presentes na base de dados. O resultado será representado visualmente por meio de um mapa de calor, no qual as células coloridas indicarão o grau e a direção da correlação entre as variáveis.

Este procedimento é crucial para fornecer insights sobre as relações existentes entre os diferentes atributos, destacando padrões e permitindo uma compreensão mais profunda do comportamento conjunto das variáveis.

```
[18] import seaborn as sns  
  
fig, ax = plt.subplots(figsize=(30, 10))  
sns.heatmap(df_cardio.corr(), annot=True, cmap="RdYlGn")
```

Aqui podemos visualizar a figura do mapa de correlação entre os dados das variáveis categóricas e das variáveis contínuas, sendo assim, podemos passar ao próximo passo do nosso projeto, ou seja, vamos fazer o algoritmo ler e aprender como identificar quem possui doenças cardíacas ou não!



7 LEITURA E TREINAMENTO DA IA

Neste estágio, procederemos ao treinamento do algoritmo para identificar na base de dados indivíduos que apresentam ou não problemas cardíacos. Esse processo envolverá a utilização de 70% do público analisado na base de dados, como conjunto de treinamento, visando capacitar o algoritmo para tomar decisões baseadas em padrões apresentados nesse subconjunto

Posteriormente, avaliaremos a capacidade preditiva do algoritmo ao submetê-lo a à pessoa que nunca foram vistas antes, interessando-nos pela capacidade de classificar se elas apresentam ou não problemas cardíacos

Para efetivar esse processo, realizaremos a separação das variáveis em Y (variáveis alvo) e X (variáveis preditivas), onde Y representa o que desejamos aprender, e X representa as variáveis utilizadas para esse aprendizado.

```
Y = df_cardio["cardio"]
X = df_cardio.loc[:, df_cardio.columns != 'cardio']
```

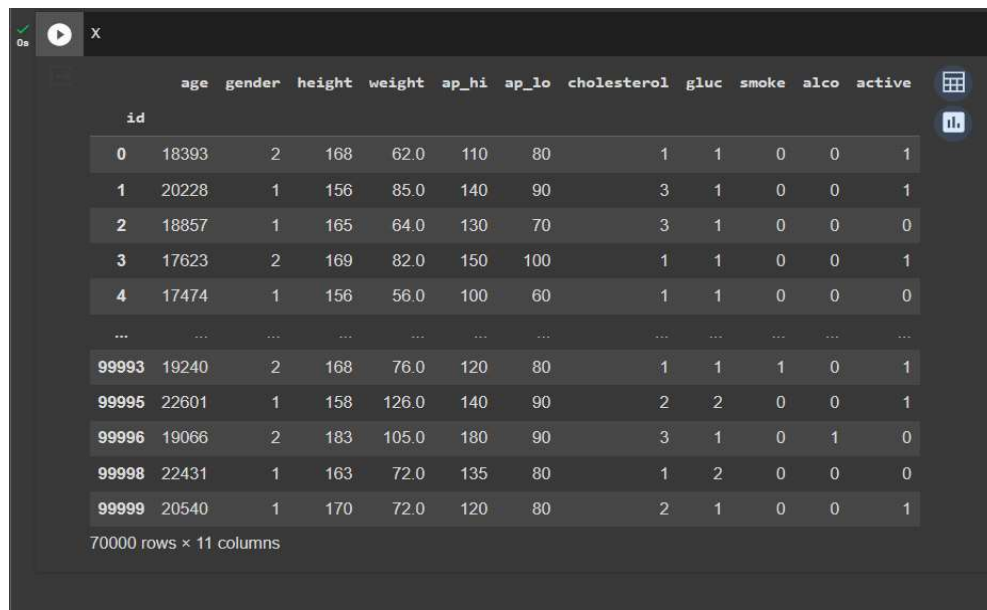
O conjunto Y consistirá na variável 'cardio', indicando a presença ou ausência de problemas cardíacos. Já o conjunto X será formado por todas as outras variáveis existentes na base de dados, exceto a variável alvo

Esta divisão facilitará a implementação de algoritmos de aprendizado de máquina, permitindo que o algoritmo associe as características (x) às suas respectivas classificações de doença cardíaca (y), aprendendo com base nos padrões apresentados no conjunto de treinamento.

[19]

```
Y = df_cardio["cardio"]  
X = df_cardio.loc[:, df_cardio.columns != 'cardio']
```

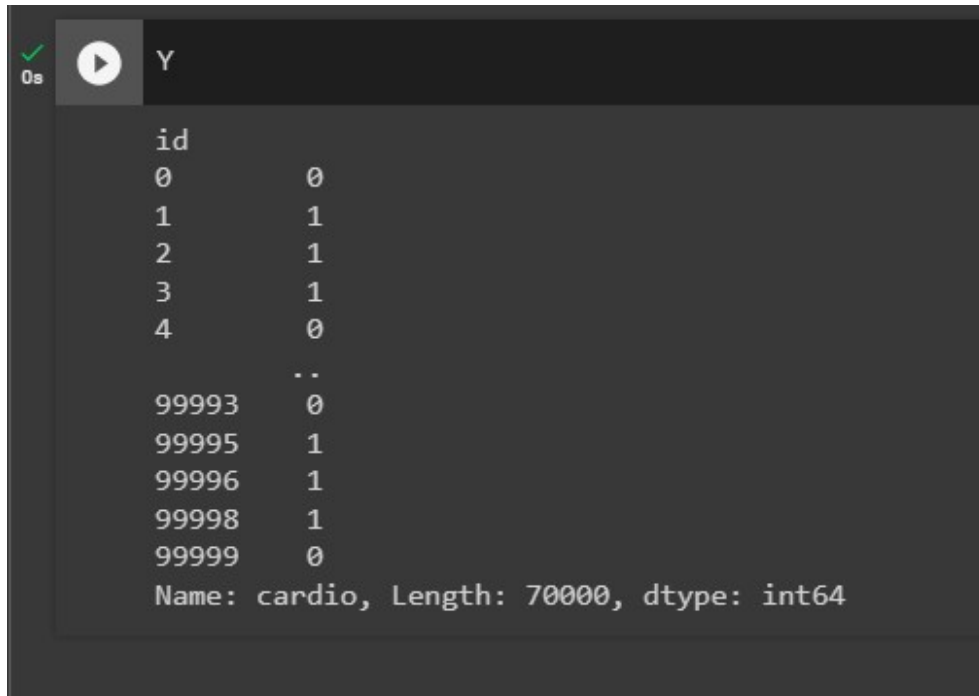
Logo abaixo podemos ver todos os dados que a variável X conseguiu adquirir após o procedimento realizado anteriormente, contendo todos os dados menos o resultado, no caso não possui dados de Y nesse, sendo assim ele será a variável que será utilizada para descobrir se uma pessoa possui doença cardíaca ou não.



	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active
id											
0	18393	2	168	62.0	110	80	1	1	0	0	1
1	20228	1	156	85.0	140	90	3	1	0	0	1
2	18857	1	165	64.0	130	70	3	1	0	0	0
3	17623	2	169	82.0	150	100	1	1	0	0	1
4	17474	1	156	56.0	100	60	1	1	0	0	0
...
99993	19240	2	168	76.0	120	80	1	1	1	0	1
99995	22601	1	158	126.0	140	90	2	2	0	0	1
99996	19066	2	183	105.0	180	90	3	1	0	1	0
99998	22431	1	163	72.0	135	80	1	2	0	0	0
99999	20540	1	170	72.0	120	80	2	1	0	0	1

70000 rows x 11 columns

Já em Y, podemos ver todas as pessoas que foram identificadas com doença cardíaca e pessoas com coração saudável, baseando-se nos dados adquiridos pela variável X, no qual possui todas as métricas necessárias para isso.



id	cardio
0	0
1	1
2	1
3	1
4	0
...	...
99993	0
99995	1
99996	1
99998	1
99999	0

Name: cardio, Length: 70000, dtype: int64

A fim de realizar o treinamento e avaliação do algoritmo, empregaremos uma função disponibilizada pela biblioteca Scikit-Learn (Sklearn), que permite a divisão aleatória dos dados entre conjuntos de treinamento e teste.

```
from sklearn.model_selection import train_test_split
```

A seguinte função será utilizada para realizar o corte aleatório dos dados tendo em vista que buscamos um resultado com a maior eficiência possível para chegar uma precisão acima dos 50%.

```
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.33,
random_state=42)
```

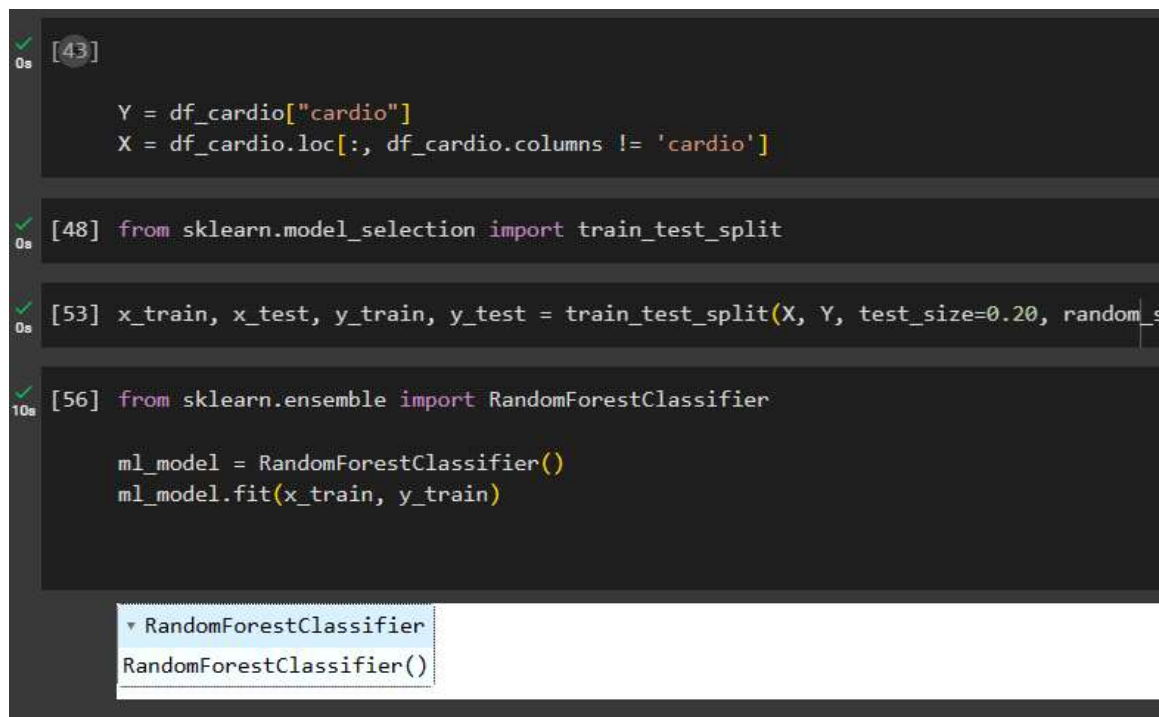
Nesse contexto, 70% do público da base de dados será destinado ao treinamento do algoritmo, enquanto os outros 30% restantes serão utilizados para avaliação da performance do modelo

Posteriormente, para a implementação do algoritmo de aprendizado de máquina, utilizaremos um classificador de floresta aleatória (random forest), disponível na

biblioteca Sklearn.

```
from sklearn.ensemble import RandomForestClassifier
ml_model = RandomForestClassifier()
ml_model.fit(x_train, y_train)
```

O objetivo da instância 'ml_model' é construir um modelo de floresta aleatória e treiná-lo com base nos dados de treinamento (70% da base para realizar previsões em novos dados, visando classificar se os indivíduos apresentam ou não problemas cardíacos. O conjunto de teste será empregado para avaliar a acurácia e a eficácia do modelo



The screenshot shows a Jupyter Notebook interface with a dark background. It displays four code cells, each with a green checkmark icon on the left indicating successful execution. The first cell, labeled [43], contains two lines of code: `Y = df_cardio["cardio"]` and `X = df_cardio.loc[:, df_cardio.columns != 'cardio']`. The second cell, labeled [48], contains `from sklearn.model_selection import train_test_split`. The third cell, labeled [53], contains `x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.20, random_s`. The fourth cell, labeled [56], contains `from sklearn.ensemble import RandomForestClassifier`, `ml_model = RandomForestClassifier()`, and `ml_model.fit(x_train, y_train)`. Below the code cells, a dropdown menu is open, showing the class `RandomForestClassifier` and its constructor `RandomForestClassifier()`.

8 TESTES DE EFICACIA E EFICIENCIA NOS RESULTADOS

Agora que o modelo foi treinado, realizaremos uma previsão para obter um diagnóstico sobre a condição cardiovascular de uma pessoa, selecionada aleatoriamente, utilizando o comando 'predict'

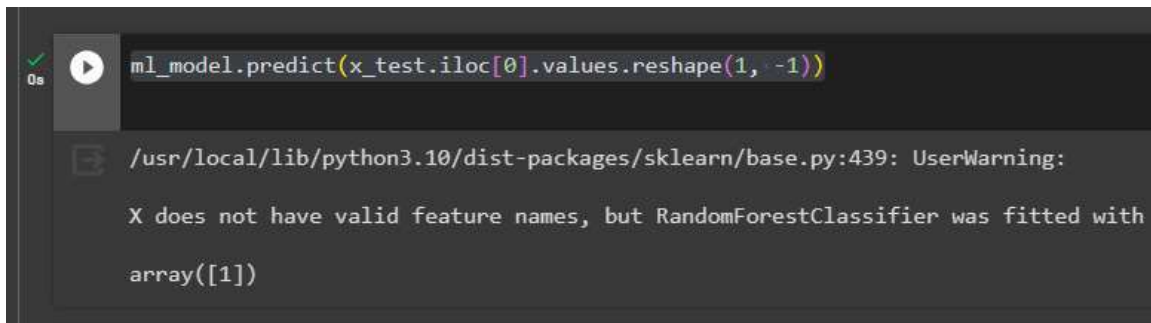
```
ml_model.predict(x_test.iloc[0].values.reshape(1, -1))
```


Aqui, o comando 'predict' é aplicado à primeira pessoa do conjunto de teste('x_test'). O resultado obtido indica a classificação prevista pelo modelo, indicando se o algoritmo acredita que a pessoa em questão apresenta ou não problemas cardíacos

Esta funcionalidade é crucial para entender como o modelo se comporta em situações práticas e para avaliar sua capacidade de generalização a dados não vistos durante o treinamento.

A execução desse comando mostra que o modelo prevê que a primeira pessoa testada está propensa a ter problemas cardíacos. Essa informação, embora útil, deve ser interpretada com cautela e sempre validada com base em conhecimentos médicos e análises mais aprofundadas

Essa abordagem d modelo de aprendizado de máquina permite a rápida avaliação e classificação de dados, fornecendo insights valiosos para aplicação de diagnóstico médico. Contudo, É fundamental compreender os limites do modelo e considerar a orientação de um profissional de saúde para interpretação mais completa e precisa



```
ml_model.predict(x_test.iloc[0].values.reshape(1, -1))
```

/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning:
X does not have valid feature names, but RandomForestClassifier was fitted with
array([1])

Após feito o teste anterior utilizando o 'predict' para saber um possível resultado, foi feito a execução do teste definitivo, analisando o resultado utilizando o comando 'y_test.iloc[0]'. Este comando nos permite acessar a classificação real da primeira pessoa do conjunto de teste, em relação à presença ou ausência de problemas cardíacos.

y_test.iloc[0]

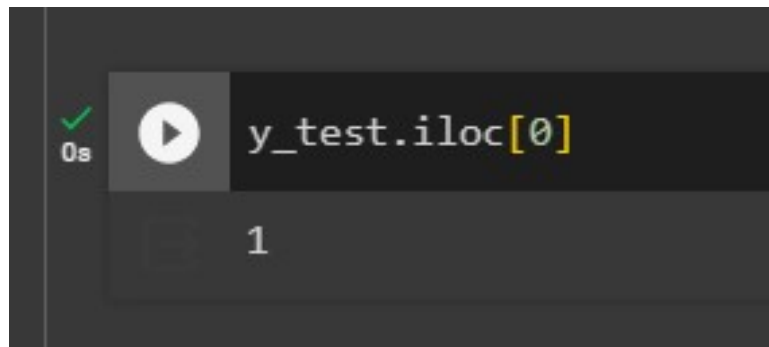
O resultado obtido nos mostra a verdadeira condição cardíaca da primeira pessoa do conjunto de testes, oferecendo uma comparação crucial com a previsão fornecida pelo modelo de aprendizado de máquina. No contexto de avaliação de modelos, essa comparação é fundamental para medir a acurácia e confiabilidade do modelo

Ao observamos que a classificação ela indica a presença de problema cardíacos, e

considerando que a previsão do modelo também concorda com essa condição, temos uma indicação positiva da eficácia em realizar diagnósticos precisos

No entanto, é importante ressaltar que resultados individuais devam ser interpretados com prudência, em cenários médicos, a avaliação do desempenho do modelo deve ser estendida para um conjunto mais amplo de dados e validada por profissionais da saúde.

Esta abordagem de comparação entre resultados reais e previstos destaca a importância de validar a eficácia dos modelos de aprendizado de máquina, especialmente quando aplicados em contextos sensíveis como diagnóstico de condições médicas



9 APRIOMORAMENTO E DESEMPENHO ATINGIDO

```
from sklearn.metrics import classification_report, confusion_matrix
```

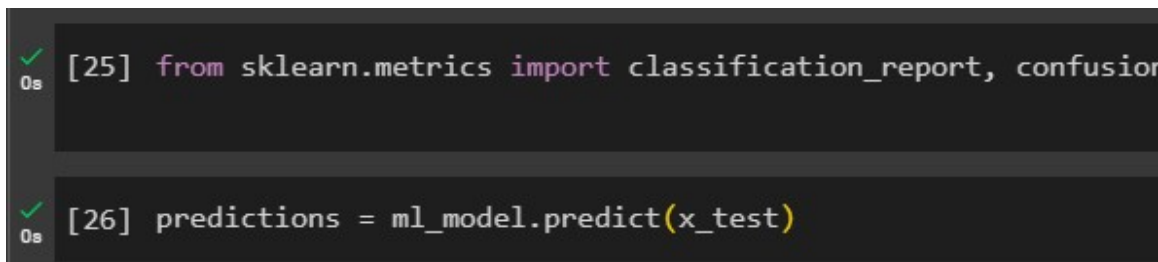
Aqui, estamos importando duas métricas importantes para avaliar a performance de modelos de classificação do Scikit-Learn. A primeira métrica é o ‘classification_report’, que fornece uma análise detalhada das métricas de classificação, como precisão, recall e F1-score, para cada classe no problema de classificação. A segunda métrica é a ‘confusion_matrix’, que cria uma matriz que mostra a quantidade de previsões corretas e incorretas feitas pelo modelo

```
predictions = ml_model.predict(x_test)
```

Neste trecho, utilizamos o modelo de aprendizado de máquina(‘ml_model’) que foi treinado anteriormente. A função ‘predict’ é aplicada ao conjunto de teste (‘x_test’), resultando em um array ‘predictions’ que contém as previsões do modelo para cada exemplo no conjunto de teste.

Essencialmente, após essa execução, temos as previsões do modelo armazenadas na variável 'predictions'. Essas previsões serão utilizadas posteriormente para comparar com os valores reais e avaliar o desempenho do modelo em relação às métricas de classificação e à matriz de confusão

Uma análise mais aprofundada dessas métricas fornecerá insights valiosos sobre a capacidade do modelo em classificar corretamente as instâncias em diferentes classes, sendo fundamental para entender a qualidade e eficácia do modelo de aprendizado de máquina treinado.



```
[25] from sklearn.metrics import classification_report, confusion_matrix

[26] predictions = ml_model.predict(x_test)
```

Neste momento, procederemos à aplicação do modelo treinado sobre todo o conjunto de teste. Utilizando o método 'predict', solicitamos que o algoritmo faça previsões para todas as pessoas presentes na base de dados de teste('x_test'). As previsões resultantes são armazenadas na variável chamada 'predictions'.

```
print(classification_report(y_test, predictions))
print(confusion_matrix(y_test, predictions))
```

Posteriormente, realizaremos uma comparação abrangente entre as previsões geradas pelo modelo ('predictions') e os valores reais contidos na variável alvo ('y_test'). A métrica utilizada para essa análise será o 'classification_report', fornecendo informações detalhadas sobre a precisão, recall e outras métricas de avaliação do modelo.

Adicionalmente, será apresentada uma matriz de confusão ('confusion_matrix'), que permite visualizar quantos exemplos foram classificados corretamente (verdadeiros positivos e verdadeiros negativos) e quantos foram classificados erroneamente (falsos positivos e falsos negativos). Esta matriz oferece uma visão mais detalhada das capacidades do modelo em lidar com diferentes cenários de classificação.

Ambas as informações. O relatório de classificação e a matriz de confusão, são fundamentais para avaliação completa do desempenho do modelo. A interpretação desses resultados pode fornecer insights valiosos sobre a eficácia do modelo em identificar corretamente pessoas com ou sem problemas cardíacos, orientando eventuais ajustes ou melhorias no algoritmo.

	precision	recall	f1-score	s
0	0.71	0.73	0.72	
1	0.72	0.70	0.71	
accuracy			0.72	
macro avg	0.72	0.72	0.72	
weighted avg	0.72	0.72	0.72	
[[8376 3130]				
[3442 8152]]				

10 CONCLUSÃO - RESULTADOS

Na análise das métricas de desempenho, representadas pela tabela de comparação e pela matriz de confusão, obtivemos resultados que refletem a eficácia e precisão do método de aprendizado de máquina desenvolvido. O relatório de comparação revela uma taxa de acerto em torno de 70% indicando que, para a questão de diagnóstico de problemas cardíacos, nosso modelo apresenta uma performance significativa

É válido destacar que, na perspectiva do relatório de comparação, o modelo atinge uma precisão de aproximadamente 70% o que significa que, em aproximadamente 7 de cada 10 avaliações, o modelo realiza previsões corretas. Essa assertividade é crucial em contextos médicos, onde a precisão nas previsões desempenha um papel fundamental na orientação de decisões clínicas.

Analisando a matriz de confusão, podemos identificar que, dos 8376 casos de indivíduos diagnosticados com problemas cardíacos, 3130 foram classificados como falsos positivos. Por outro lado, observamos que 3442 casos de pessoas sem problemas cardíacos e 8152 pessoas foram erroneamente classificados como falsos negativos. Essas informações destacam a importância de considerar tanto os casos positivos quanto os negativos para uma avaliação mais abrangente do modelo.

Em síntese, os resultados obtidos indicam uma promissora capacidade do modelo em realizar diagnósticos relacionados a problemas cardíacos. No entanto, é fundamental reconhecer a necessidade de aprimoramentos contínuos, considerando a complexidade e a delicadeza das decisões médicas. Este trabalho representa um passo significativo na

aplicação de métodos de aprendizado de máquina para apoiar a identificação precoce de condições cardíacas, e sugere direções futuras para refinamentos e expansões na aplicação dessas técnicas na área da saúde.