UNIVERSIDADE SALVADOR (UNIFACS)
SISTEMAS DE INFORMAÇÃO

PRISCILA SIMAS: 12722123651 NATHALIA LIMA: 12722210839

# 3  PARTE DA A3 (TRABALHO FINAL)

SALVADOR-BA
2024

SUMÁRIO

# INTRODUÇÃO

Este código visa demonstrar um fluxo de trabalho completo para a construção e avaliação de um modelo preditivo utilizando um classificador de árvore de decisão. A escolha desse modelo se deve à sua simplicidade e interpretabilidade, tornando-o uma excelente opção para problemas de classificação.Neste projeto, utilizamos um conjunto de dados (dataset) do site KAGGLE, que que contém informações relevantes, projetado para ajudar na previsão da presença de doenças cardíacas em pacientes com base em uma variedade de características clínicas, este conjunto de dados foi coletado de várias fontes, incluindo hospitais e instituições médicas com 16 colunas e 920 linhas de dados. Os atributos incluem:

Idade: Idade do paciente.

Sexo: Gênero do paciente (masculino ou feminino).

Tipo de dor no peito (CP): Classificação das dores no peito em quatro categorias diferentes.

Pressão arterial em repouso (Trestbps): Medida da pressão arterial do paciente.

Colesterol sérico: Nível de colesterol no sangue.

Glicose em jejum: Indica se a glicose em jejum é superior a 120 mg/dl.

Resultados eletrocardiográficos: Resultados de testes eletrocardiográficos em repouso.

Frequência cardíaca máxima alcançada: A frequência cardíaca mais alta atingida durante o teste.

Angina induzida por exercício: Indica se o paciente experimenta angina durante o exercício.

Oldpeak: Depressão ST induzida por exercício em relação ao repouso.

Inclinação do segmento ST: Inclinação do segmento ST no teste de esforço.

Número de vasos principais coloridos por fluoroscopia: número de vasos sanguíneos visíveis durante o exame.

Thalassemia: Resultados relacionados à condição da thalassemia do paciente.

Por esse dataset ser algo amplo, de fácil acesso e com uma suficiente quantidade de dados, escolhemos ele para implementar em nosso trabalho cujo foco é fazer uma Análise Preditiva que tem como objetivo identificar presença ou ausência de doença cardíaca baseando-se nos atributos

## FUNDAMENTO TEÓRICO

Linguagem: Python

Bibliotecas:
Pandas: Para manipulação e análise de dados.
os: Para interagir com o sistema de arquivos.
matplotlib.pyplot: Para visualização de gráficos.
sklearn.model_selection: Para dividir os dados em conjuntos de treino e teste e realizar busca em grade.
sklearn.tree: Para usar o classificador de árvore de decisão.
sklearn.metrics: Para avaliar o desempenho do modelo.
yellowbrick.classifier: Para visualização da matriz de confusão.
.
Para nossa análise preditiva escolhemos o algoritmo de ÁRVORE DE DECISÃO, que constitui um dos algoritmos mais empregados em aprendizado de máquina por sua fácil compreensão e aplicação prática. Elas operam dividindo os dados em partes menores, baseadas em questões sobre as propriedades dos dados, resultando em decisões em formato de árvore. Nosso trabalho teve essas características:

Os dados foram divididos em conjuntos de treinamento (80%) e teste (20%).

Critério de Divisão: Os critérios, tais como 'gini' e 'entropy', determinam o processo de divisão. A impureza dos nós é avaliada pelo índice Gini, enquanto a entropia avalia a incerteza.

Hiperparâmetros: Variáveis como max_depth, min_samples_split e min_samples_leaf regulam a complexidade do modelo, contribuindo para prevenir o superajuste (overfitting), possibilita alcançar um equilíbrio que potencializa a habilidade de generalização do modelo e aprimora a acurácia dos resultados finais

Validação Cruzada e Busca em Grade

A validação cruzada é um método que possibilita avaliar a performance do modelo em diversos subconjuntos de dados, assegurando que os resultados não sejam restritos apenas ao conjunto de treinamento. A aplicação do algoritmo de busca em grade (GridSearchCV) possibilita a otimização dos hiperparâmetros do modelo por meio da análise sistemática das possíveis combinações.

Scoring: A métrica empregada para avaliar a performance do modelo (neste caso, a acurácia) é crucial para determinar quais hiperparâmetros proporcionam o melhor desempenho.

Separação dos Dados: A divisão dos dados em conjuntos de treinamento e teste é uma prática usual que contribui para assegeurar a boa generalização do modelo para novos dados.
A avaliação do desempenho do modelo é realizada através da matriz de confusão e do relatório de classificação.

## PROJETO DE IMPLEMENTAÇÃO

Começamos o processo de implementação com a limpeza do dataset, com verificação de colunas com muitos valores nulos e sem importância e também com

a análise de dados duplicados, apagando eles. Aquelas com colunas com valores nulos mas não ao ponto de apagar eles fizemos o preenchimento com Mediana ou Média nas colunas numéricas como ‘trestbps’,’chol’,’thalch’, e ‘oldpeak’ e outros preenchemos com valores padrões que fazem sentido no contexto dos dados e para manter a consistência deles

Depois, preparando os dados para a modelagem colocamos a coluna “num”, fora da análise pois ela é a coluna alvo, a coluna que possui o resultado da doença cardiovascular, no dataset original, a coluna “num” possui resultados que vão de 0 a 4, que significa o nível da doença, ou não tem ou é algo que vai se agravando, mas para a nossa análise mudamos esse tipo de resultado e juntamos para só mostrar se os pacientes possuíam ou não a doença cardiovascular, do outro jeito estava dando resultados insuficientes e problemas nas classes, então decidimos juntar, definidos com:

Presença de doença: 1

Nenhuma doença: 0

```
df.loc[df["num"] == 2,"num"]=1
df.loc[df["num"] ==3,"num"]=1
df.loc[df["num"] ==4,"num"]=1
y = df['num']
```

Identificamos  e transformamos  também as colunas categóricas representada em textos em variáveis indicadores porque aprendizagem de máquina se dar melhor com entradas numéricas e podem ser processadas mais precisamente

```
colunas_categoricas = df.select_dtypes(include=['object',
'category']).columns.tolist()
print("Colunas categóricas:", colunas_categoricas)

#Converte variáveis categóricas em variáveis indicadoras para
que possam ser usadas no modelo e evitar problema de
multicolinearidade
```

```
df = pd.get_dummies(df, columns=['sex', 'slope', 'cp', 'fbs',
'restecg', 'exang', 'thal', 'dataset'], drop_first=True)
X = df.drop('num', axis='columns')
```

## CONSIDERAÇÕES FINAIS

Como resultado tivemos:

Acurácia do modelo de 77%

Matriz de Confusão:

[[58 21]

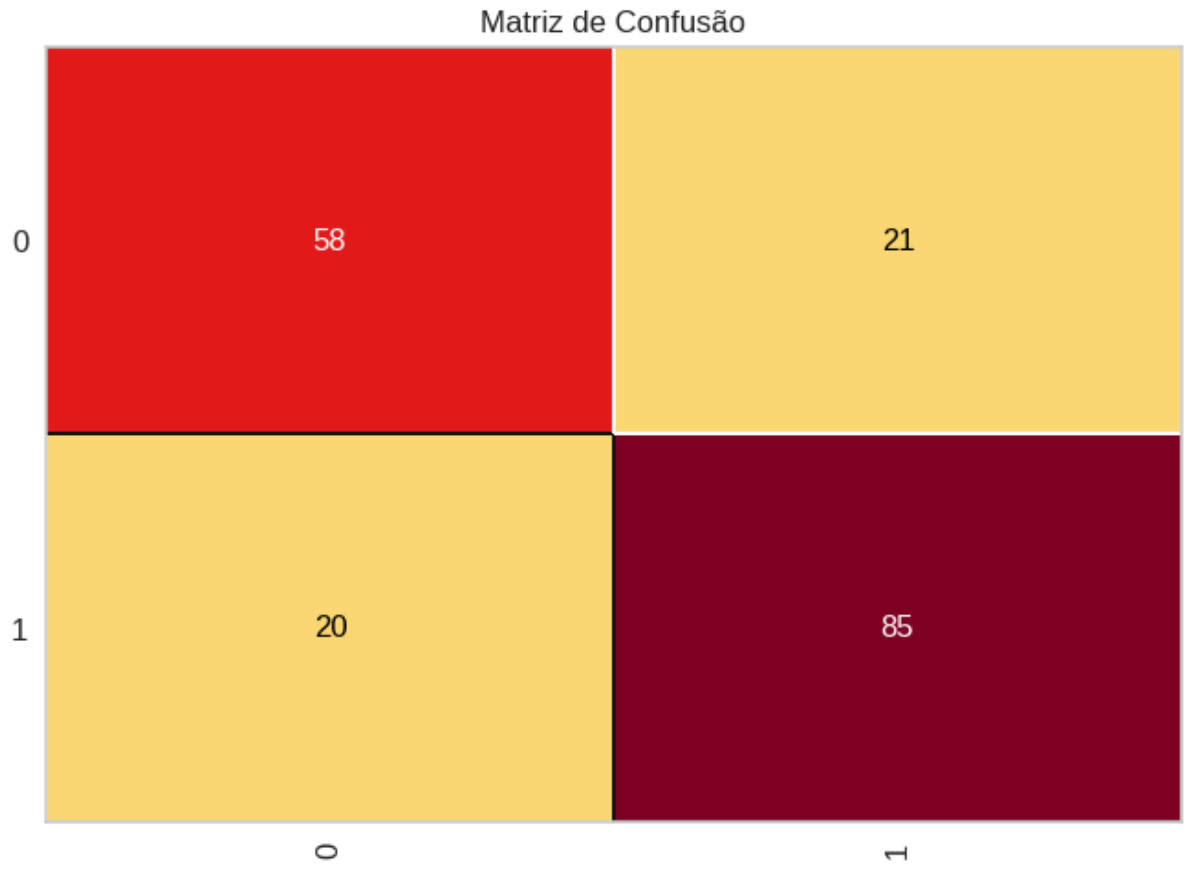
[20 85]]

Relatório de Classificação:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.74 | 0.73 | 0.74 | 79 |
| 1 | 0.80 | 0.81 | 0.81 | 105 |
| accuracy | | | 0.78 | 184 |
| macro avg | 0.77 | 0.77 | 0.77 | 184 |
| weighted avg | 0.78 | 0.78 | 0.78 | 184 |

- Acurácia: Aproximadamente 77,6%, indicando que o modelo acerta a maioria das previsões.
- Precisão: Aproximadamente 80,2%, mostrando que quando o modelo prevê a classe 1, ele está correto em cerca de 80% das vezes.

- Recall: Aproximadamente 81%, indicando que o modelo consegue identificar cerca de 81% das instâncias reais da classe 1.
- F1 Score: Aproximadamente 80,6%, que é uma média harmônica entre precisão e recall, refletindo um bom equilíbrio entre as duas métricas.




O modelo previu corretamente que 58 pacientes não possuem doença (Verdadeiro Positivo)

O modelo previu incorretamente que 21 pacientes possuíam a doença mas na verdade eles não possuíam (Falso Negativo)

O modelo previu incorretamente que 20 pacientes não possuíam a doença mas na verdade eles tinham (Falso Positivo)

O modelo previu corretamente que 85 pacientes tinham a doença (Verdadeira Negativo)

Um dos principais desafios que enfrentamos foi a baixa acurácia do projeto, nas primárias tentativas sempre ficava abaixo de 40%, o que é muito pouco desempenho do código. Tivemos que apagar refazer várias vezes, voltando começo, começou a aumentar quando paramos para dar mais atenção ao fato da limpeza dos dados, acreditamos que antes o algoritmo não conseguia processar certos dados por conta da falta de normalização entre eles, então a acurácia começou a aumentar com os detalhes da limpeza e com mais alguns adicionais como os hiperparâmetro, então conseguimos um resultado satisfatório.
Tentamos também deixar o código mais robusto e complexo mas sempre que mudamos algo, voltava a apresentar um resultado insuficiente.

BIBLIOGRAFIA

https://www.kaggle.com/datasets/redwankarimsony/heart-disease-data/data

https://www.datacamp.com/pt/blog/top-python-libraries-for-data-science

https://www.datageeks.com.br/ajuste-de-hiperparametros/

https://medium.com/@jvsavietto6/machine-learning-m%C3%A9tricas-valida%C3%A7%C3%A3o-cruzada-bias-e-vari%C3%A2ncia-380513d97c95

https://medium.com/@vitor.modesto.leitao/matriz-de-confus%C3%A3o-m%C3%A9tricas-e-trade-off-a60f9d79028b