Pós Tech Fase 3

O presente documento visa apresentar o projeto da terceira fase do curso de Machine Learnin Engineering. Para tal, foi escolhido o dataset Heart Failure Prediction Dataset, que está disponível no link: https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction/data

Análise exploratória e ajustes do dataset

A análise exploratória do dataset foi realizada seguindo os seguintes passos:

1. Carregamento e inspeção dos dados

Para inspecionar os dados, foram utilizados os comandos df.head(3) (Figura 1), df.info() (Figura 2) e df.describe() (Figura 3).

Na Figura 1, observa-se a presença de variáveis categóricas, indicando a necessidade de tratamento adequado para que o dataset possa ser utilizado corretamente em análises posteriores.

A Figura 2 reforça a identificação das variáveis categóricas e apresenta informações gerais sobre o dataset, como o número de linhas (918) e o número de colunas (12).

Observando a Figura 3, foi possível identificar o seguinte:

1. A linha count indica que não há valores nulos no dataset.
2. A coluna Cholesterol apresenta um desvio padrão elevado, indicando grande variação entre os pacientes e sugerindo a possível necessidade de tratamento ou limpeza desse atributo.
3. As colunas RestingBP, Cholesterol possuem valores mínimos iguais a zero, o que pode indicar erros de medição e reforça a necessidade de limpeza ou ajuste desses atributos.

Na Figura 4, observa-se que as informações das colunas categóricas não apresentam anormalidades, indicando que os valores estão consistentes e dentro do esperado.

```
df = pd.read_csv('dataset/heart-failure.csv')

df.head(3)
```

✓ 0.0s

	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 1 – Primeiras 3 linhas do conjunto de dados

```
df.info()
✓ 0.0s
<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
```

Figura 2 - Saída do comando df.info(), exibindo o número de registros, nomes das colunas, contagem de valores não nulos e tipos de dados presentes no conjunto de dados.

```
df.describe()
✓ 0.0s
```

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease
count	918.000000	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	0.553377
std	9.432617	18.514154	109.384145	0.423046	25.460334	1.066570	0.497414
min	28.000000	0.000000	0.000000	0.000000	60.000000	-2.600000	0.000000
25%	47.000000	120.000000	173.250000	0.000000	120.000000	0.000000	0.000000
50%	54.000000	130.000000	223.000000	0.000000	138.000000	0.600000	1.000000
75%	60.000000	140.000000	267.000000	0.000000	156.000000	1.500000	1.000000
max	77.000000	200.000000	603.000000	1.000000	202.000000	6.200000	1.000000

Figura 3 - Resultado do comando df.describe(), apresentando medidas como média, desvio padrão, valores mínimos, máximos e quartis das variáveis numéricas.

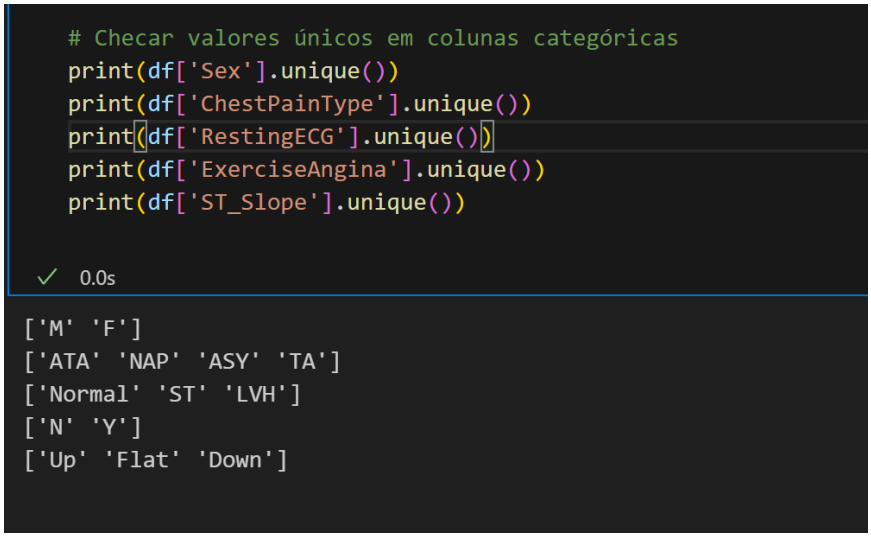

Figura 4 - A figura apresenta os valores distintos encontrados em cada atributo categórico

2. Tratamento de dados e detecção de outliers

Como identificado anteriormente, os atributos RestingBP e Cholesterol possuem valores 0, totalizando um total de 173 linhas juntos, conforme a figura abaixo (Figura 5):

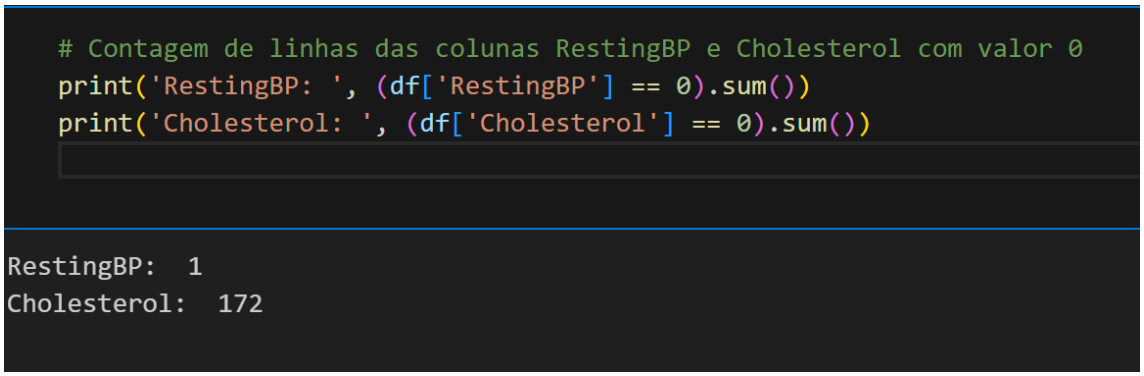

Figura 5 – Contagem dos valores iguais a 0 nos atributos RestingBP e Cholesterol

Para tratar esses valores e, ao mesmo tempo, preservar a quantidade de dados, calculou-se a mediana de cada atributo desconsiderando os valores iguais a zero. Em seguida, os valores zerados foram substituídos pela respectiva mediana como pode ser observado nas Figuras 6 e 7.

```python
# Obater apenas os valores válidos (maiores que 0) para saber a mediana
colesterol_validos = df[df['Cholesterol'] > 0]['Cholesterol']
restingbp_validos = df[df['RestingBP'] > 0]['RestingBP']

# Calular a mediana dos valores válidos
mediana_colesterol = colesterol_validos.median()
mediana_restingbp = restingbp_validos.median()

print('Mediana Colesterol: ', mediana_colesterol)
print('Mediana RestingBP: ', mediana_restingbp)
```

```
Mediana Colesterol:  223.0
Mediana RestingBP:  130.0
```

Figura 6 – Calculo da mediana

```python
# Substitui os valores 0 pela mediana
df['Cholesterol'] = df['Cholesterol'].replace(0, mediana_colesterol)

df['RestingBP'] = df['RestingBP'].replace(0, mediana_restingbp)
print(df.shape)
df.describe()
```

```
(918, 12)
```

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease
count	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000
mean	53.510893	132.538126	240.581699	0.233115	136.809368	0.887364	0.553377
std	9.432617	17.990127	53.982967	0.423046	25.460334	1.066570	0.497414
min	28.000000	80.000000	85.000000	0.000000	60.000000	-2.600000	0.000000
25%	47.000000	120.000000	214.000000	0.000000	120.000000	0.000000	0.000000
50%	54.000000	130.000000	223.000000	0.000000	138.000000	0.600000	1.000000
75%	60.000000	140.000000	267.000000	0.000000	156.000000	1.500000	1.000000
max	77.000000	200.000000	603.000000	1.000000	202.000000	6.200000	1.000000

7 – A figura mostra o resultado da substituição de valores 0 pela mediana nas colunas Cholesterol e RestingBP.

Para verifica a presença de outliers foi usando o método do IQR (Intervalo Interquartil) para os atributos Age, RestingBP, Cholesterol, MaxHR e Oldpeak, conforme as Figuras 8 e 9. E foi feita uma contagem da quantidade de outliers para cada atributo, como mostra a Figura 10.

Por falta de conhecimento na área médica, nenhum tratamento foi aplicado aos outliers, de forma a evitar a exclusão ou alteração de dados que poderiam estar corretos.

```python
# Verificar outliers usando o método do IQR (Intervalo Interquartil)
features = ['Age', 'RestingBP', 'Cholesterol',  'MaxHR', 'Oldpeak']

features_limits = {}

for feature in features:
    q1 = np.percentile(df[feature], 25)
    q3 = np.percentile(df[feature], 75)
    iqr = q3 - q1

    print(f"{feature} Q1: {q1}, Q3: {q3}, IQR: {iqr}")
    # Limite inferior e superior
    limite_inf = q1 - 1.5 * iqr
    limite_sup = q3 + 1.5 * iqr

    # Fora desse intervalo é um outlier
    print(limite_inf, limite_sup)
    print('-------------\n')

    features_limits[feature] = {'inf': float(limite_inf), 'sup': float(limite_sup)}
```

Figura 8 – Código para identificar limites inferior e superior das variáveis numéricas

```
Age Q1: 47.0, Q3: 60.0, IQR: 13.0
27.5 79.5
-------------

RestingBP Q1: 120.0, Q3: 140.0, IQR: 20.0
90.0 170.0
-------------

Cholesterol Q1: 214.0, Q3: 267.0, IQR: 53.0
134.5 346.5
-------------

MaxHR Q1: 120.0, Q3: 156.0, IQR: 36.0
66.0 210.0
-------------

Oldpeak Q1: 0.0, Q3: 1.5, IQR: 1.5
-2.25 3.75
-------------
```

Figura 9 – Resultados do IQR

```
    for feature in features:
        outliers = df[(df[feature] < features_limits[feature]['inf']) |
                      (df[feature] > features_limits[feature]['sup'])]
        print(f"Outliers em {feature}: {len(outliers)} linhas")
```

✓ 0.0s

```
Outliers em Age: 0 linhas
Outliers em RestingBP: 27 linhas
Outliers em Cholesterol: 41 linhas
Outliers em MaxHR: 2 linhas
Outliers em Oldpeak: 16 linhas
```

Figura 10 - Resultado da contagem de valores fora dos limites definidos pelo método do IQR.

Para a normalização do dataset, foi aplicada a técnica de Label Encoding devido à presença de variáveis categóricas. Essa técnica converte valores textuais em representações numéricas inteiras, permitindo a utilização de métodos estatísticos e de aprendizado de máquina que exigem entradas numéricas.

Após a normalização, foi possível gerar o heatmap (Figura 11). Nele, observa-se que os atributos com maior correlação estão concentrados no canto inferior direito.

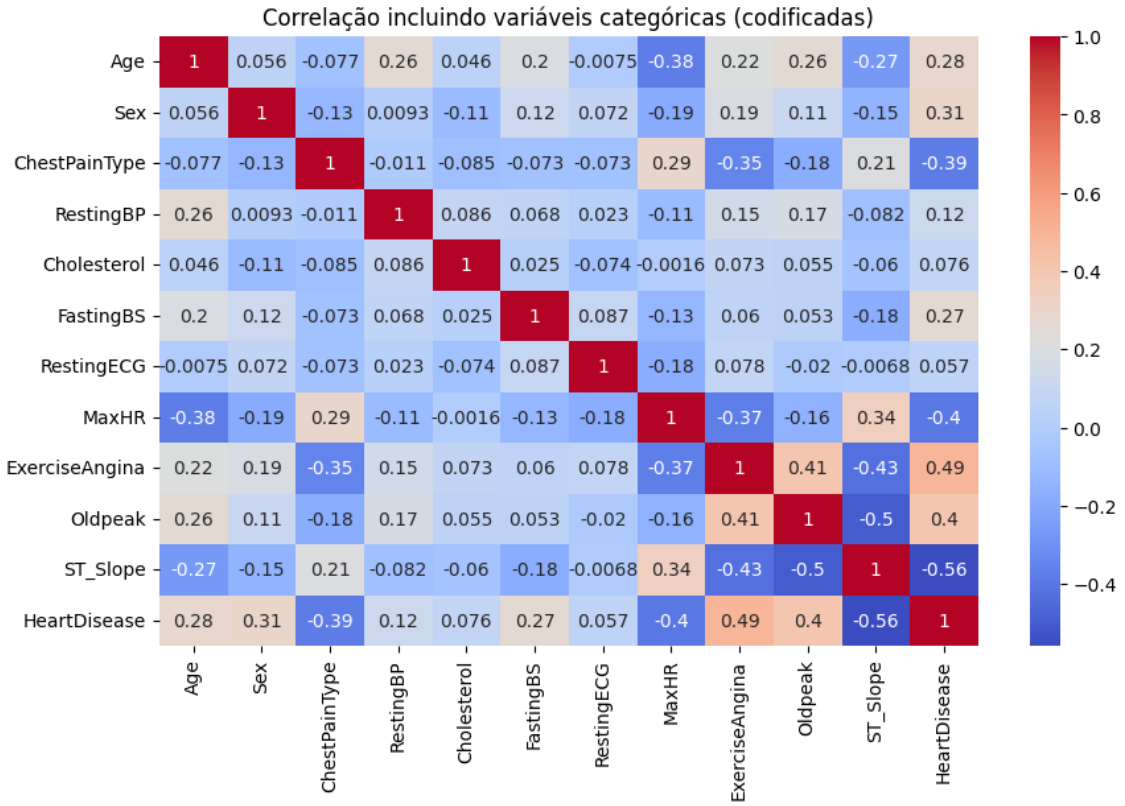

Figura 11 – Heatmap.

```
corr = df.corr()["HeartDisease"].sort_values(ascending=False)
print(corr)
```

✓ 0.0s

```
HeartDisease      1.000000
ExerciseAngina    0.494282
Oldpeak           0.403951
Sex               0.305445
Age               0.282039
FastingBS         0.267291
RestingBP         0.117798
Cholesterol       0.076114
RestingECG        0.057384
ChestPainType    -0.386828
MaxHR            -0.400421
ST_Slope         -0.558771
Name: HeartDisease, dtype: float64
```

Figura 12 - Filtro de correlação dos atributos em relação à variável alvo HeartDisease.

Com base nas imagens apresentadas acima, os atributos considerados foram: ExerciseAngina, Oldpeak, ST_Slope, MaxHR, ChestPainType, pois apresentam correlação mais forte com a variável alvo (HeartDisease). Essa seleção permite reduzir a dimensionalidade do dataset, concentrando a análise em características que possuem maior relevância estatística e potencial impacto no desempenho dos modelos de aprendizado de máquina.

Uso dos Dados em Modelos de ML

Os modelos selecionados para este estudo foram: Regressão Logística, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Árvore de Decisão e XGBoost. Modelos baseados em redes neurais não foram considerados devido ao tamanho reduzido do conjunto de dados, o que poderia comprometer seu desempenho e capacidade de generalização.

Para todos os modelos, o dataset foi dividido em x e y, sendo x composto pelos atributos ExerciseAngina, Oldpeak, ST_Slope, MaxHR, ChestPainType, e y correspondente à variável alvo HeartDisease.

Para todos os modelos, foram adotadas as mesmas abordagens: a realização de dois testes sem alterações nos hiperparâmetros. Na primeira tentativa, o dataset foi dividido em 70% para treinamento e 30% para teste; na segunda, a divisão foi de 90% para treinamento e 10% para teste. Em ambos os casos, utilizou-se random state igual a 45 e aplicou-se o GridSearchCV para a otimização dos hiperparâmetros.

1 – Regressão logística

Os testes realizados sem ajustes nos hiperparâmetros não apresentaram variações significativas nos resultados, observando-se apenas uma diferença de 0,0073.

```
Acurácia: 0.8405797101449275
Precisão: 0.8405797101449275
Recall: 0.8405797101449275
F1-Score: 0.8405797101449275
```

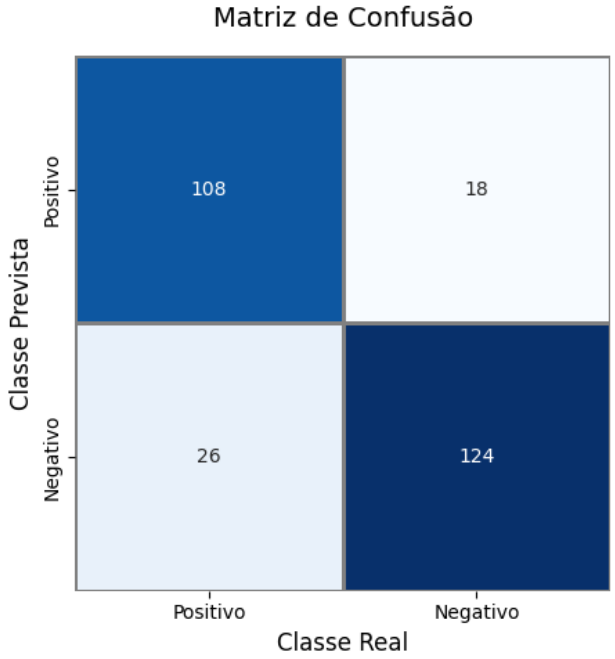

Figura 13 - Resultado, 70% treinamento e 30% teste

```
Acurácia: 0.8478260869565217
Precisão: 0.8478260869565217
Recall: 0.8478260869565217
F1-Score: 0.8478260869565217
```

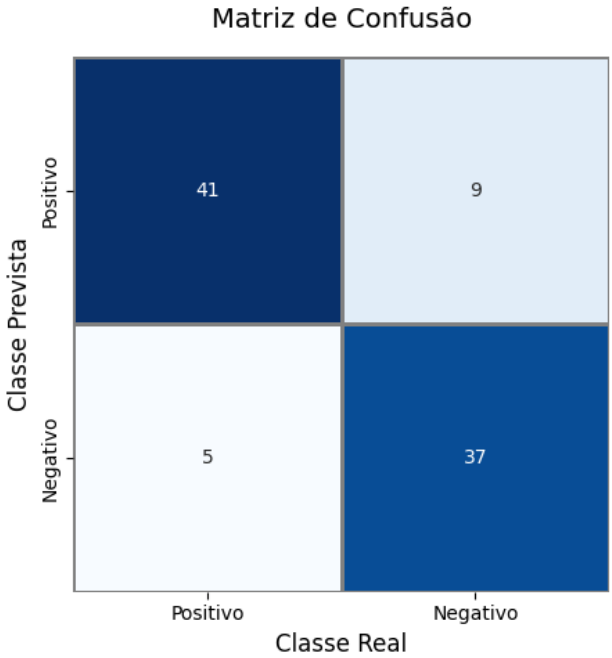

Figura 14 - Resultado, 90% treinamento e 10% teste

Após a aplicação do GridSearchCV para a otimização dos hiperparâmetros, mesmo variando o número de iterações no LogisticRegression(), o melhor resultado obtido apresentou acurácia e precisão de 0,836.

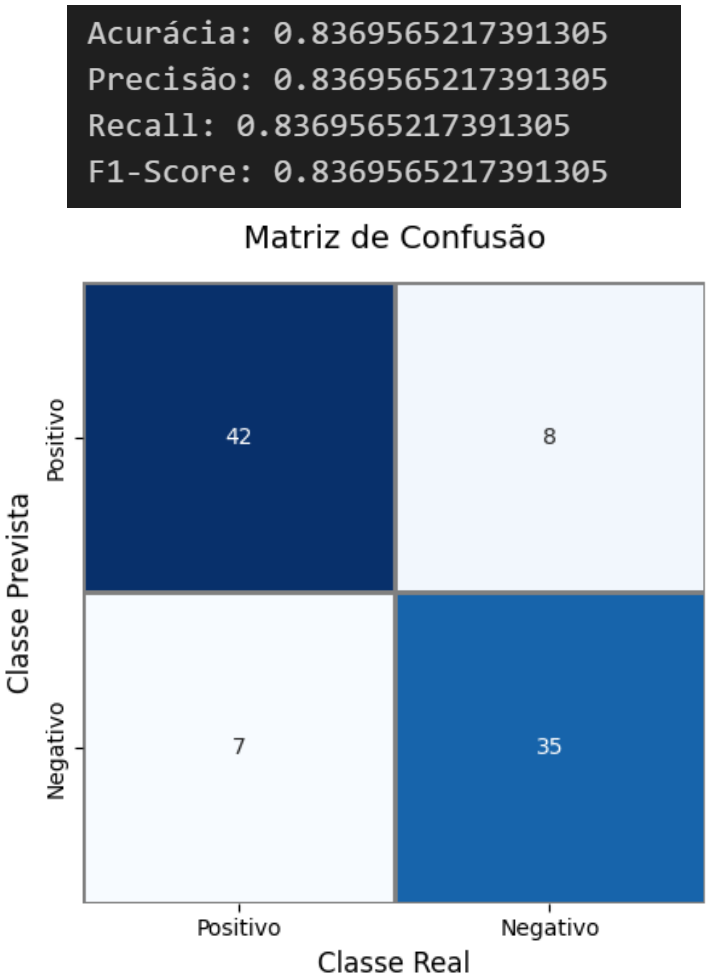

Figura 15 - Resultado, 90% treinamento e 10% teste com o ajuste de hiperparâmetros

2 – K-Nearest Neighbors (KNN)

Para o modelo KNN, os testes realizados sem ajustes nos hiperparâmetros também não apresentaram variações significativas nos resultados, a variação foi de apenas 0,014.

```
Acurácia: 0.8297101449275363
Precisão: 0.8297101449275363
Recall: 0.8297101449275363
F1-Score: 0.8297101449275363
```

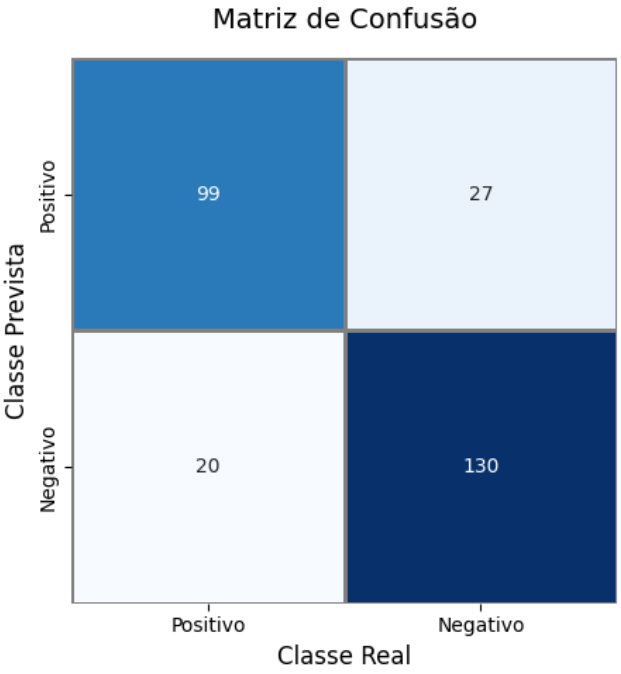

Figura 16 - Resultado, 70% treinamento e 30% teste

```
Acurácia: 0.8152173913043478
Precisão: 0.8152173913043478
Recall: 0.8152173913043478
F1-Score: 0.8152173913043478
```

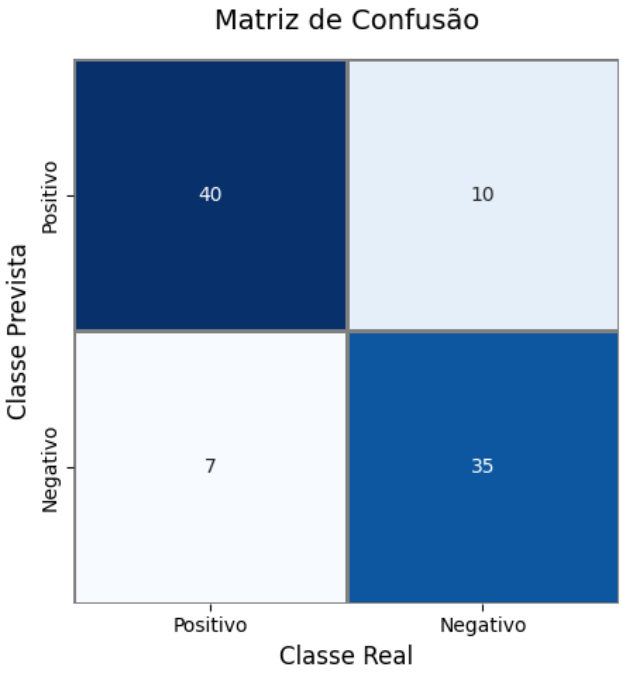

Figura 17 - Resultado, 90% treinamento e 10% teste

Após a aplicação do GridSearchCV para a otimização dos hiperparâmetros, não foi possível observar uma melhora expressiva nas métricas de desempenho.

```
Acurácia: 0.8297101449275363
Precisão: 0.8297101449275363
Recall: 0.8297101449275363
F1-Score: 0.8297101449275363
```

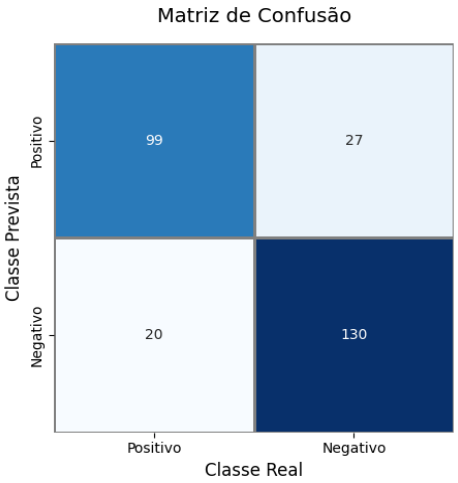

Figura 18 - Resultado, 70% treinamento e 30% teste com o ajuste de hiperparâmetros

3 – Support Vector Machine (SVM)

Os testes realizados sem ajustes nos hiperparâmetros não apresentaram variações significativas nos resultados, a variação foi de apenas 0,029.

```
Acurácia: 0.8442028985507246
Precisão: 0.8442028985507246
Revocação: 0.8442028985507246
F1-Score: 0.8442028985507246
```

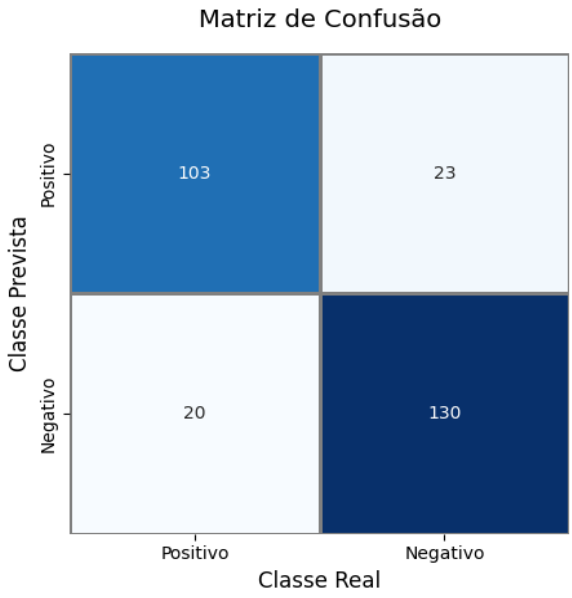

Figura 19 - Resultado, 70% treinamento e 30% teste

```
Acurácia: 0.8152173913043478
Precisão: 0.8152173913043478
Revocação: 0.8152173913043478
F1-Score: 0.8152173913043478
```

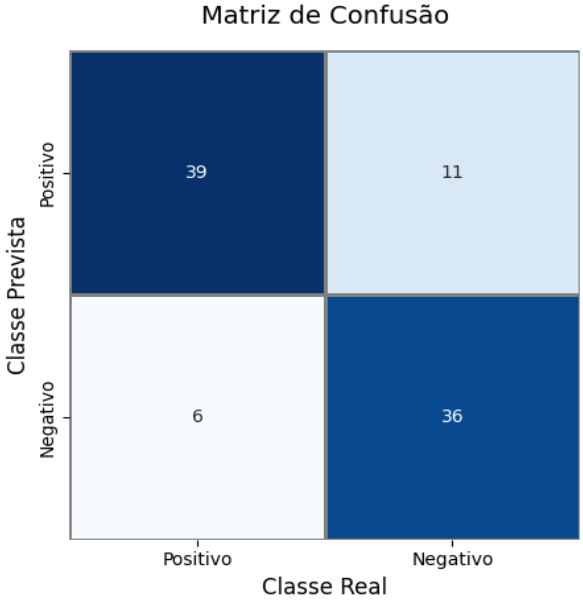

Figura 20 - Resultado, 90% treinamento e 10% teste

Após a aplicação do GridSearchCV para a otimização dos hiperparâmetros, não foi possível observar uma melhora expressiva nas métricas de desempenho.

```
Acurácia: 0.8442028985507246
Precisão: 0.8442028985507246
Revocação: 0.8442028985507246
F1-Score: 0.8442028985507246
```

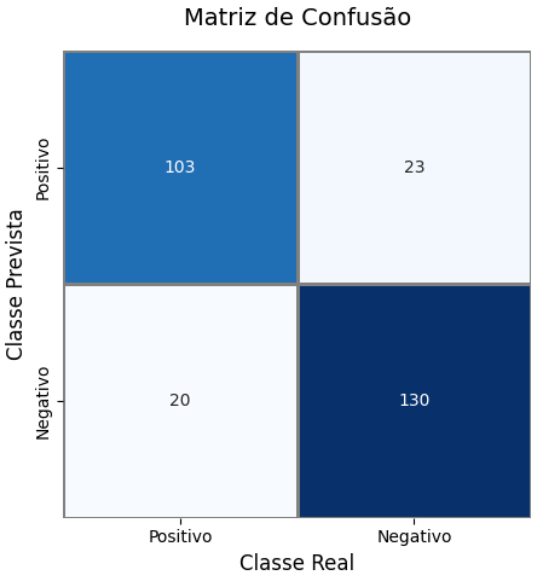

Figura 21 - Resultado, 70% treinamento e 30% teste
com o ajuste de hiperparâmetros

4 – Árvore de decisão

Os testes realizados sem ajustes nos hiperparâmetros não apresentaram variações significativas nos resultados, a variação foi de apenas 0,007.

```
Acurácia: 0.7753623188405797
Precisão: 0.7753623188405797
Recall: 0.7753623188405797
F1-Score: 0.7753623188405797
```

Figura 22 - Resultado, 70% treinamento e 30% teste

```
Acurácia: 0.782608695652174
Precisão: 0.782608695652174
Recall: 0.782608695652174
F1-Score: 0.782608695652174
```

Figura 23 - Resultado, 90% treinamento e 10% teste

Após a aplicação do GridSearchCV para a otimização dos hiperparâmetros, observou-se uma leve melhora nas métricas de desempenho.

```
Acurácia: 0.8260869565217391
Precisão: 0.8260869565217391
Recall: 0.8260869565217391
F1-Score: 0.8260869565217391
```

Figura 24 - Resultado, 70% treinamento e 30% teste com o ajuste de hiperparâmetros.

5 – XGBoost

Os testes realizados sem ajustes nos hiperparâmetros não apresentaram variações significativas nos resultados, a variação foi de apenas 0,0018.

```
Accuracy: 0.8079710144927537
Precision: 0.8211920529801324
Recall: 0.8266666666666667
F1-score: 0.8239202657807309
```

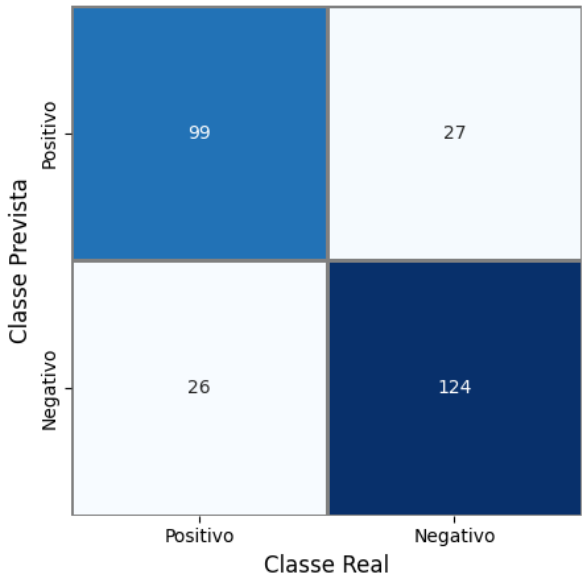

Figura 25 - Resultado, 70% treinamento e 30% teste

```
Accuracy: 0.8260869565217391
Precision: 0.7954545454545454
Recall: 0.8333333333333334
F1-score: 0.813953488372093
```

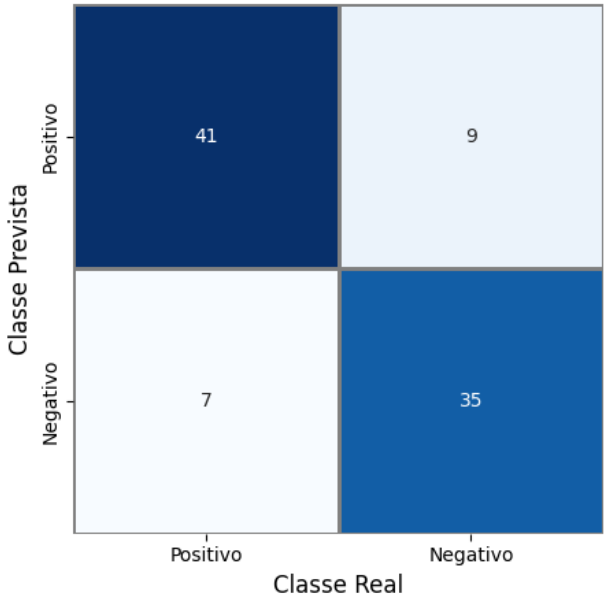

Figura 26 - Resultado, 90% treinamento e 10% teste

Após a aplicação do GridSearchCV para a otimização dos hiperparâmetros, não foi possível observar uma melhora expressiva nas métricas de desempenho.

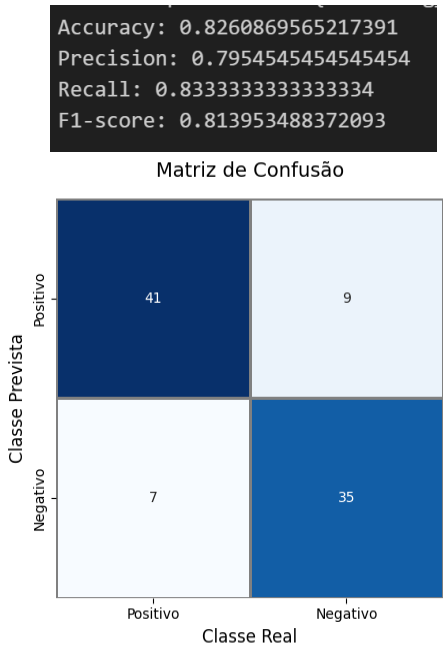

Figura 27 - Resultado, 90% treinamento e 10% teste

Conclusão

A análise exploratória permitiu identificar informações importantes, como variáveis categóricas, valores iguais a zero onde deveriam ser positivos, possíveis outliers e outros padrões relevantes.

Além disso, por se tratar de dados muito específicos sobre doenças cardiovasculares, surgiram diversas dúvidas quanto aos limites superiores e inferiores que os atributos deveriam apresentar para fazer sentido em um contexto médico. Portanto, o tratamento de outliers e o ajuste dos valores dos atributos podem ter influenciado significativamente os modelos aplicados. Vale ressaltar que, por falta de conhecimento especializado na área médica, nenhum valor foi alterado, exceto aqueles iguais a zero, onde deveriam ser positivos.

De acordo com as métricas dos modelos apresentados acima, todos atingiram valores entre 80% e 84%. Alguns modelos, como a Árvore de Decisão, só alcançaram esse desempenho após o ajuste de seus parâmetros por meio do GridSearchCV. Em termos de processamento, também não foram observadas diferenças significativas entre os modelos.

Como a regressão logística apresentou as melhores métricas, ela foi selecionada para aplicação e testes pelos usuários por meio do Streamlit, com o deploy realizado na plataforma Render.