Nome dos integrantes: Guilherme Barbosa, Guilherme Mutão, João Floriano, Luciano Ângelo

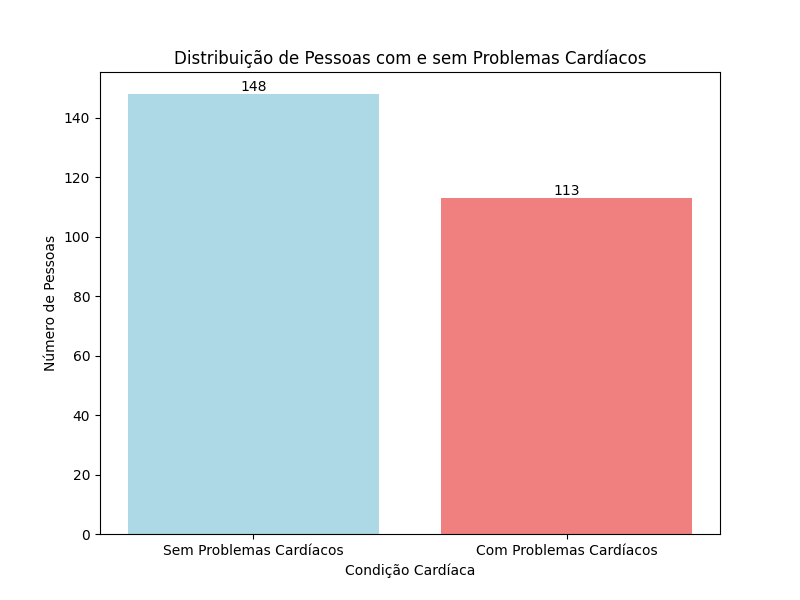
Através de um conjunto de dados abrangente que inclui informações como idade, sexo, pressão arterial, níveis de colesterol e outros fatores de risco, aplicamos técnicas de aprendizado de máquina para criar modelos preditivos.

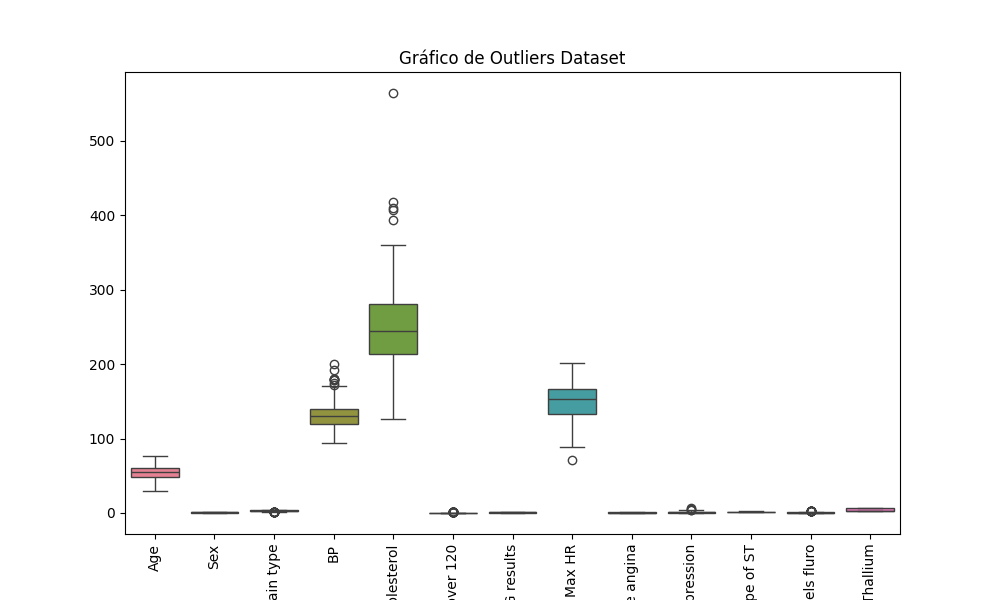
O objetivo é prever quais pacientes têm maior probabilidade de sofrer de uma doença cardíaca em um futuro próximo usando os recursos fornecidos.

Foi utilizado um dataset contendo 270 dados

Retirado 9 outliers presentes

261 dados aproveitados





Foram utilizados 75% de treinamento – 25% de testes

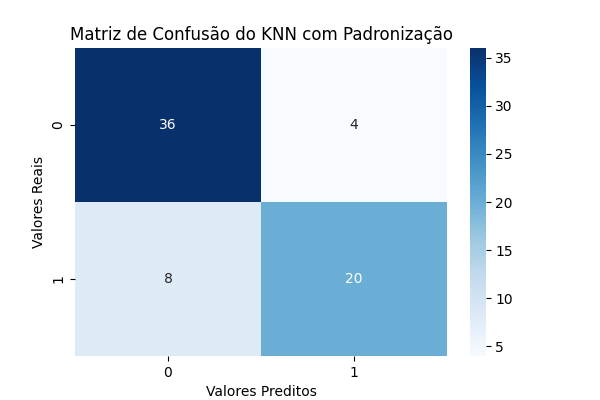
Foi utilizado uma técnica de pré-processamento **StandardScaler**

StandardScaler é uma técnica de pré-processamento de dados amplamente utilizada em aprendizado de máquina e mineração de dados. Ele faz parte do conjunto de técnicas de normalização de recursos que ajudam a tornar os dados mais apropriados para algoritmos de aprendizado de máquina. O principal objetivo do StandardScaler é padronizar os recursos (colunas) de um conjunto de dados para que eles tenham uma média zero e um desvio padrão de um.

* Calcula a média (valor médio) de cada recurso nos dados de treinamento.
* Calcula o desvio padrão (uma medida de dispersão) de cada recurso nos dados de treinamento.
* Subtrai a média de cada valor de recurso.
* Divide cada valor de recurso pelo desvio padrão.

**KNN com padronização de dados**

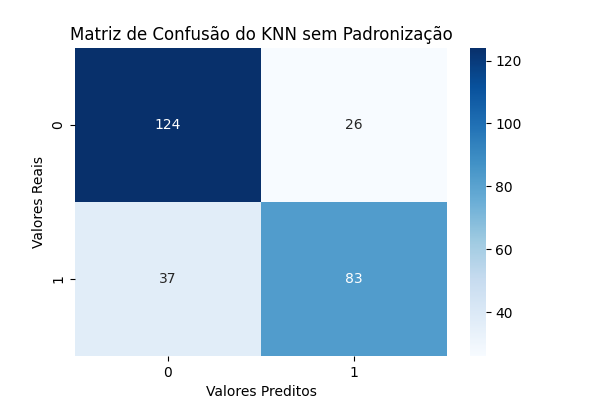
* Usa a função train\_test\_split para dividir os dados em conjuntos de treinamento (80%) e teste (20%).
* Aplica padronização aos dados usando StandardScaler para normalizar os recursos.
* Cria um modelo SVM com um kernel linear usando SVC (Support Vector Classifier).
* Treina o modelo no conjunto de treinamento.
* Faz previsões no conjunto de teste e calcula a acurácia do modelo.

Acurácia do modelo: 82%

**KNN sem padronização de dados**

* Divide os dados em conjuntos de recursos X e rótulos y.
* Usa a função train\_test\_split para dividir os dados em conjuntos de treinamento (75%) e teste (25%).
* Cria um modelo SVM com um kernel linear usando SVC.
* Treina o modelo no conjunto de treinamento.
* Faz previsões no conjunto de teste e calcula a acurácia do modelo.

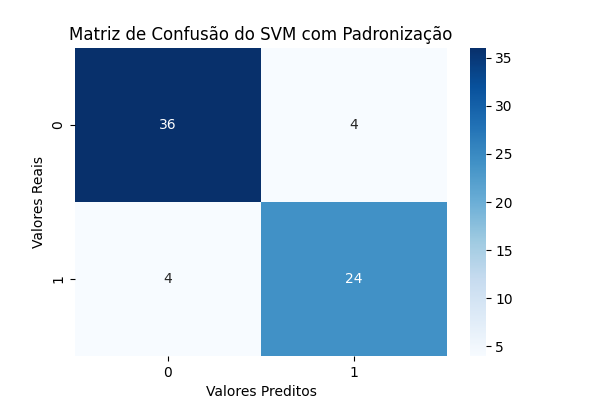
Acurácia do modelo: 76%



**SVM com padronização de dados**

* Divide os dados em conjuntos de treinamento (80%) e teste (20%) usando train\_test\_split.
* Aplica padronização aos dados usando StandardScaler.
* Cria um modelo SVM com um kernel linear usando SVC.
* Treina o modelo no conjunto de treinamento.
* Faz previsões no conjunto de teste e calcula a acurácia do modelo.

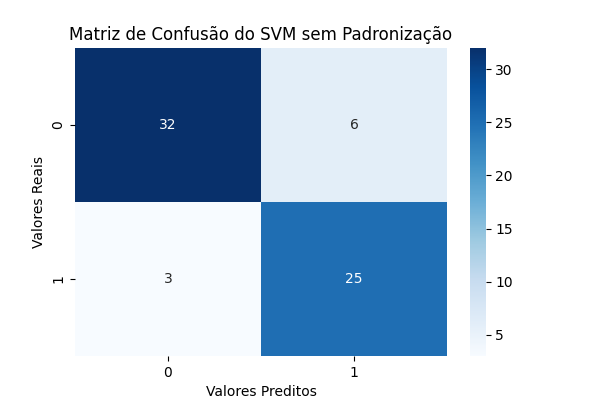
Acurácia do modelo: 88%



**SVM sem padronização de dados**

* Extrai as features (X) e os rótulos (y) diretamente do DataFrame.
* Divide os dados em conjuntos de treinamento (75%) e teste (25%) usando train\_test\_split.
* Cria um modelo SVM com um kernel linear usando SVC.
* Treina o modelo no conjunto de treinamento.
* Faz previsões no conjunto de teste e calcula a acurácia do modelo.

Acurácia do modelo: 86%

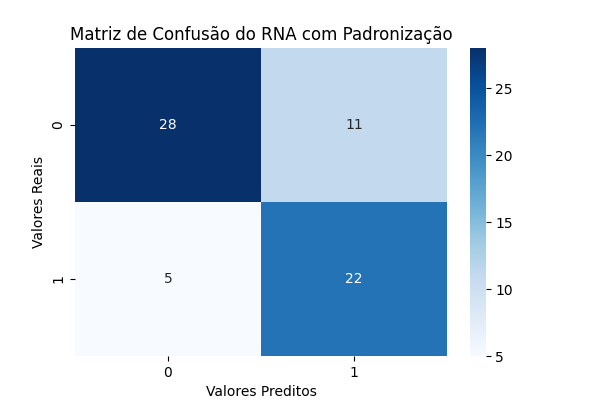


**RNA com padronização de dados**

Divide os dados em conjuntos de treinamento e teste usando train\_test\_split com uma divisão de 80% para treinamento e 20% para teste.

* Aplica padronização aos dados com StandardScaler.
* Cria um modelo de RNA com duas camadas ocultas, cada uma com 64 neurônios.
* Treina o modelo no conjunto de treinamento.

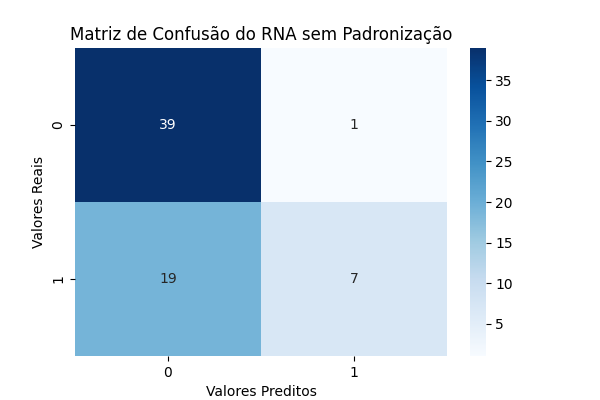
Acurácia do modelo 76%

****

**RNA sem padronização de dados**

* Divide os dados em conjuntos de treinamento e teste usando train\_test\_split com uma divisão de 75% para treinamento e 25% para teste.
* Cria um modelo de RNA com duas camadas ocultas, com tamanhos diferentes (1024 e 512 neurônios), e usa o otimizador "adam".
* Treina o modelo no conjunto de treinamento.

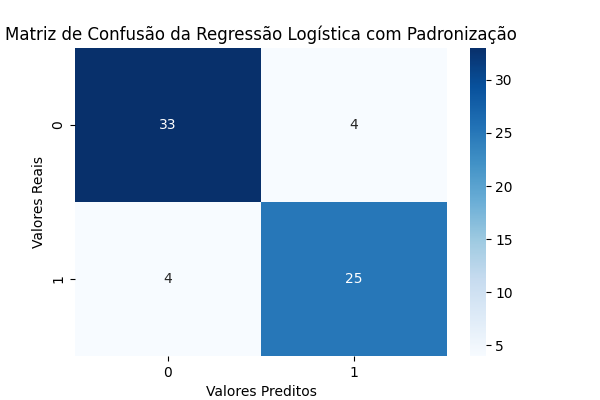
Acurácia do modelo: 69%

****

**Regressão logística com padronização de dados**

* Divide os dados em conjuntos de treinamento e teste usando train\_test\_split com uma divisão de 80% para treinamento e 20% para teste.
* Aplica padronização aos dados com StandardScaler.
* Cria um modelo de Regressão Logística com o solver padrão (geralmente 'lbfgs' ou similar) e um número padrão de iterações (máximo de 100).
* Treina o modelo no conjunto de treinamento.

Acurácia do modelo: 87%



**Regressão logística modelo sem padronização de dados**

* Divide os dados em conjuntos de treinamento e teste usando train\_test\_split com uma divisão de 75% para treinamento e 25% para teste.
* Não aplica padronização aos dados.
* Cria um modelo de Regressão Logística com o solver 'lbfgs' e um número maior de iterações (máximo de 1000).
* Treina o modelo no conjunto de treinamento.

Acurácia do modelo: 83%

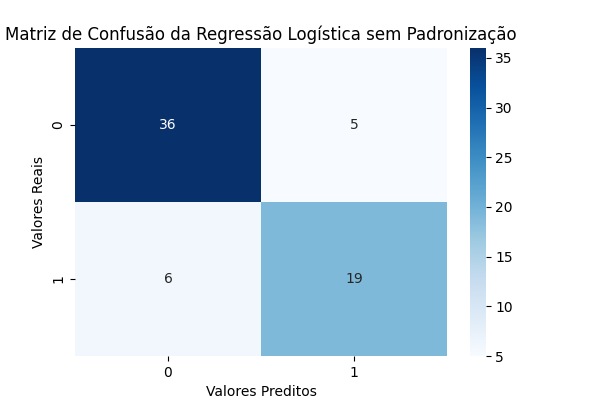
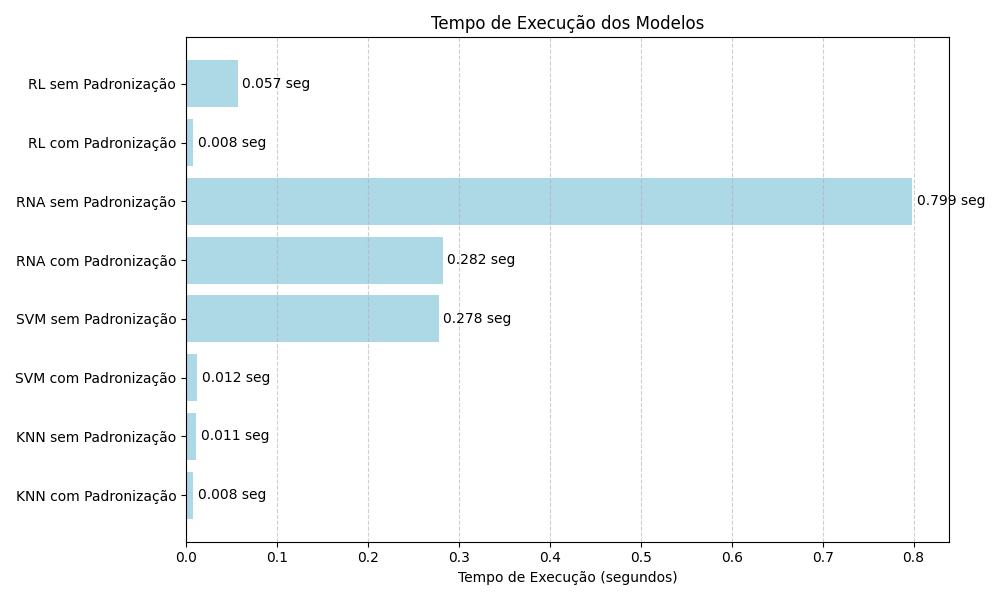


Gráfico de tempo de execução dos modelos



Acurácia dos modelos

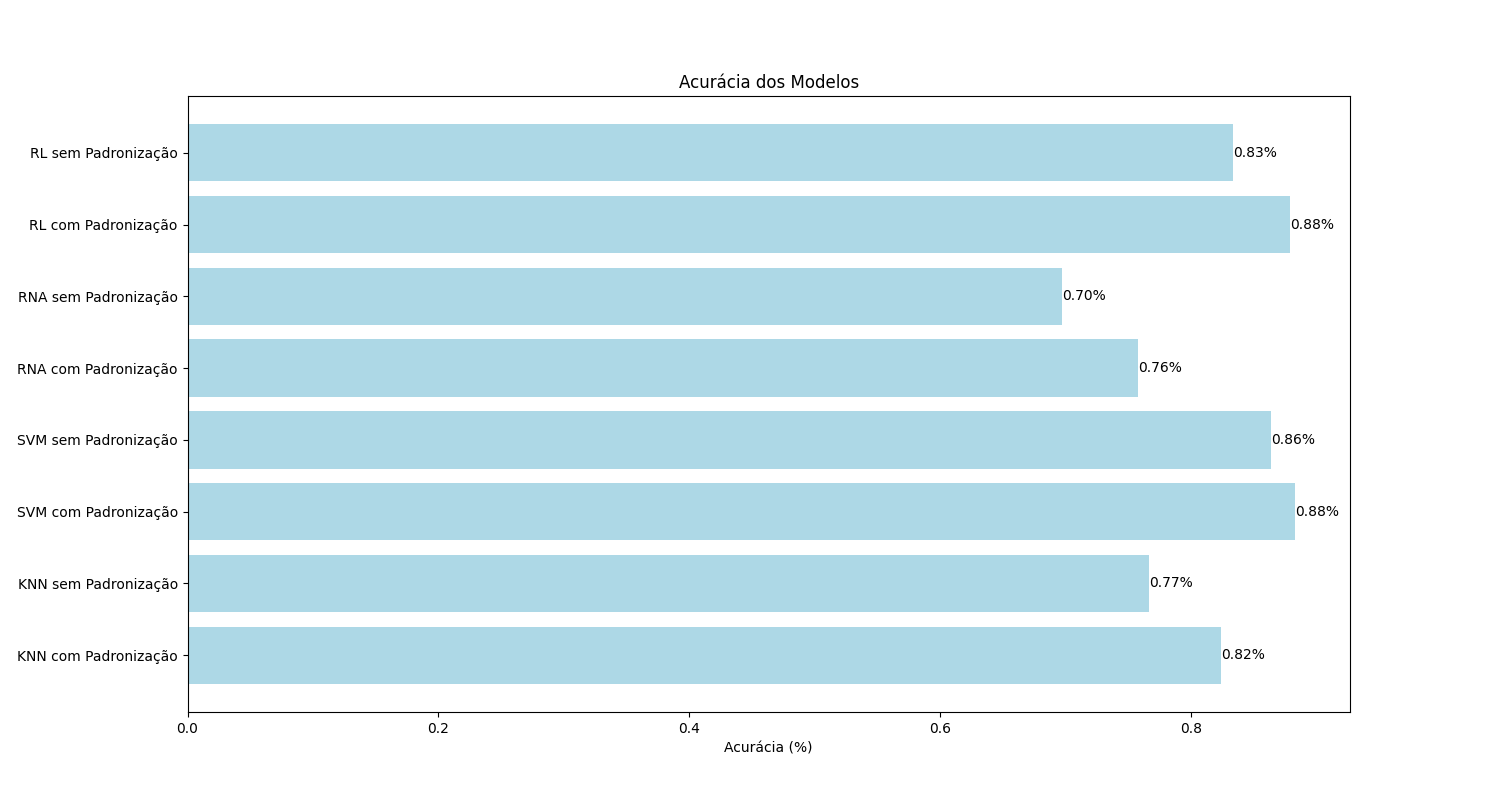


Tabela de acurácia x tempo

|  |  |  |
| --- | --- | --- |
| Modelo | Acurácia | Tempo(segundos) |
| KNN com padronização | 82% | 0.008 |
| KNN sem padronização | 76% | 0.011 |
| SVM com padronização | 88% | 0.012 |
| SVM sem padronização | 86% | 0.278 |
| RNA com padronização | 75% | 0.282 |
| RNA sem padronização | 69% | 0.799 |
| RL com padronização | 87% | 0.008 |
| RL sem padronização | 83% | 0.057 |