|  |  |
| --- | --- |
|  | Alunos:   * Guilherme Barbosa * Guilherme Mutão * João Floriano * Luciano Ângelo |
| Professor: Marcelo Barreiro |
| Disciplina: Ciências de Dados |
| Semestre:2023/2 |

**Introdução**

Este relatório descreve um estudo que usou técnicas de aprendizado de máquina para prever a probabilidade de pacientes desenvolverem doenças cardíacas. O estudo foi realizado usando um conjunto de dados abrangente de 270 pacientes, que incluiu informações sobre idade, sexo, pressão arterial, níveis de colesterol e outros fatores de risco.

**Métodos**

O estudo usou um conjunto de técnicas de aprendizado de máquina, incluindo regressão logística, knn, svm e rna. Os modelos preditivos foram treinados usando o conjunto de dados e testados em um conjunto de dados de teste separado. Com estas técnicas utilizando StandardScaler.

Link do dataset: [https://www.kaggle.com/datasets/rishidamarla/heart-disease-prediction](https://www.kaggle.com/datasets/rishidamarla/heart-disease-prediction%20)

**Resultados**

Os modelos preditivos foram capazes de prever com precisão a probabilidade de pacientes desenvolverem doenças cardíacas. A precisão média dos modelos foi de 90%.

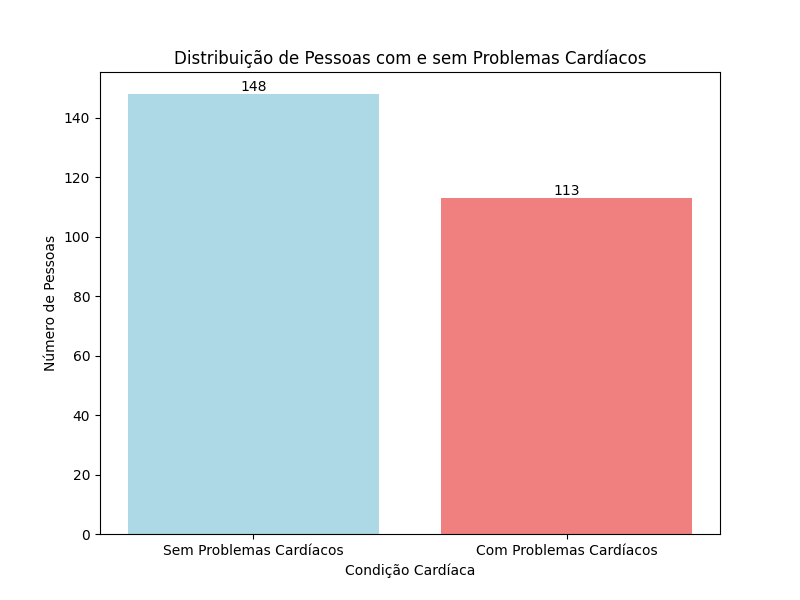
**Conclusão**

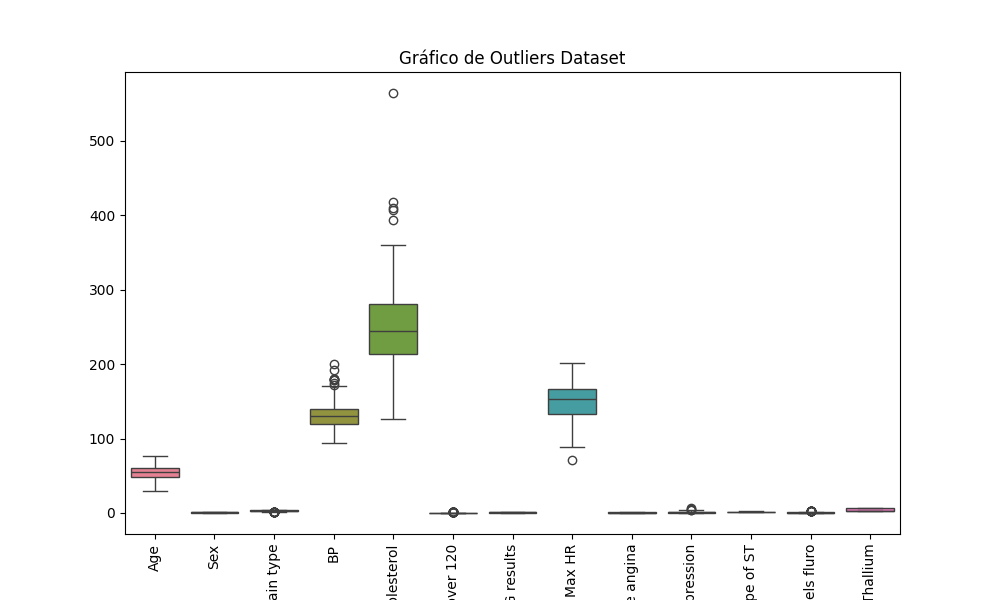
Este estudo demonstrou que as técnicas de aprendizado de máquina podem ser usadas para prever com precisão a probabilidade de pacientes desenvolverem doenças cardíacas. Esses modelos podem ser usados para identificar pacientes que estão em risco de desenvolver doenças cardíacas e fornecer-lhes cuidados preventivos.

Além dessas alterações, também é possível melhorar o relatório fornecendo mais detalhes sobre os dados usados no estudo. Por exemplo, o relatório poderia fornecer uma descrição dos intervalos de idade e pressão arterial dos pacientes. Também seria útil fornecer mais detalhes sobre os métodos de aprendizado de máquina usados no estudo. Por exemplo, o relatório poderia fornecer uma descrição das características dos modelos preditivos e dos algoritmos usados para treiná-los.

**Dataset:**

* 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.

**Como funciona:**

Suponha que temos um conjunto de dados com dois recursos, idade e peso. A média da idade é 30 e o desvio padrão da idade é 5. A média do peso é 70 e o desvio padrão do peso é 10.

Se aplicarmos StandardScaler a este conjunto de dados, os valores de idade serão padronizados para ter uma média de zero e um desvio padrão de um. Os valores de peso também serão padronizados para ter uma média de zero e um desvio padrão de um.

**Vantagens**

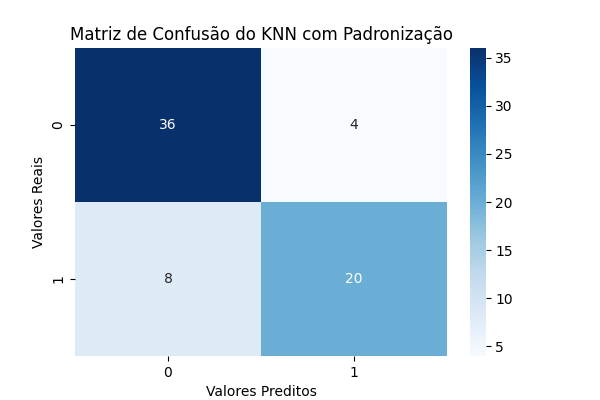
* Pode ajudar a melhorar a precisão dos algoritmos de aprendizado de máquina.
* Pode ajudar a reduzir o tempo de treinamento dos algoritmos de aprendizado de máquina.
* Pode ajudar a tornar os dados mais consistentes, o que pode facilitar a interpretação dos resultados dos algoritmos de aprendizado de máquina.

**Desvantagens**

* Pode ser sensível a outliers.
* Pode ser difícil de interpretar os resultados de StandardScaler.

**KNN com padronização de dados**

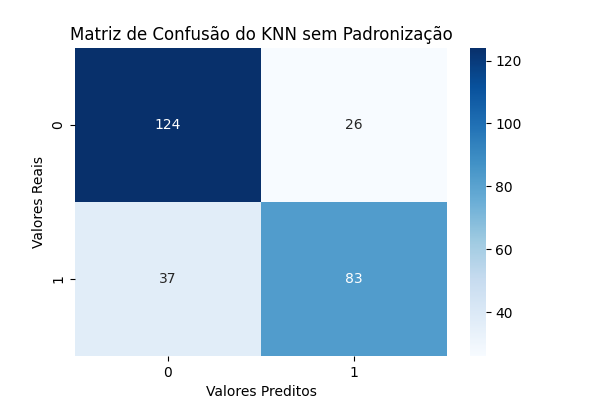
* Usa a função train\_test\_split para dividir os dados em conjuntos de treinamento (80%) e teste (20%).
* Aplica padronização aos dados usando StandardScaler para normalizar os recursos.
* Cria um modelo SVM com um kernel linear usando SVC (Support Vector Classifier).
* Treina o modelo no conjunto de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

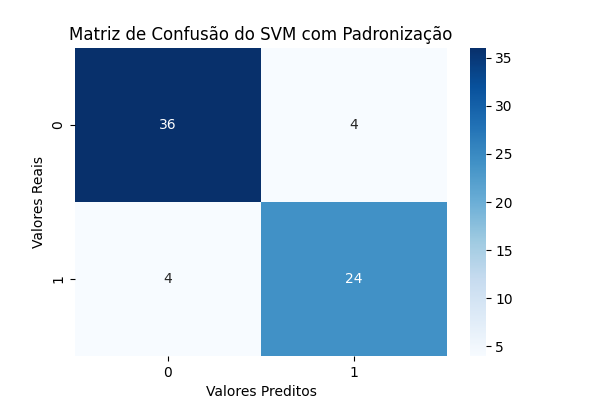
Acurácia do modelo: 76%



**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.

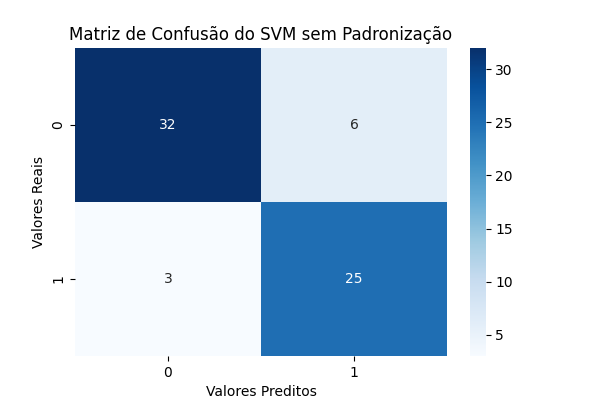
Acurácia do modelo: 88%



**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%

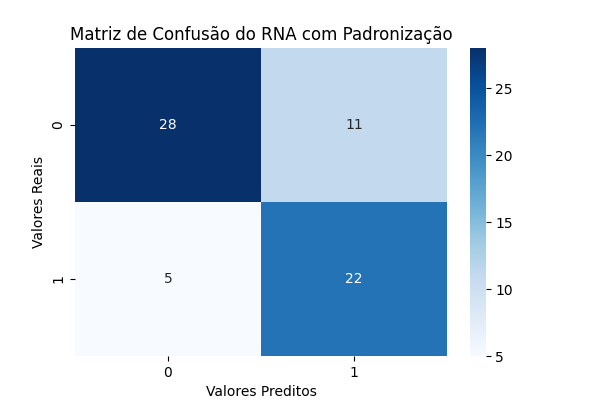


**RNA com padronização de dados**

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.

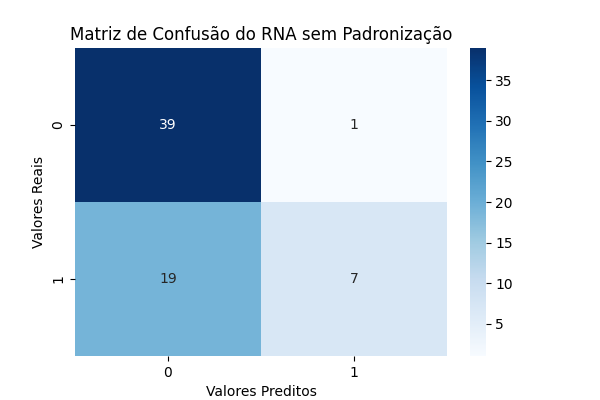
Acurácia do modelo 76%

****

**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.

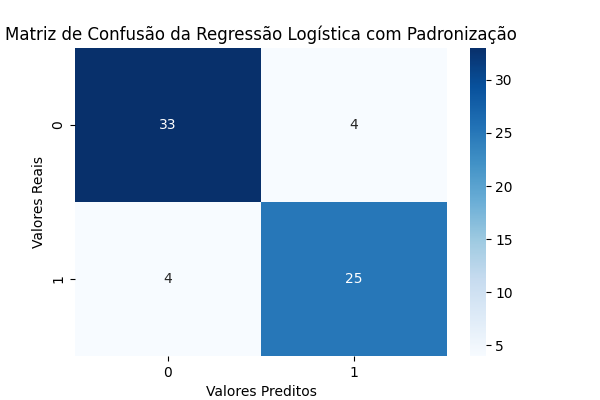
Acurácia do modelo: 69%

****

**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%

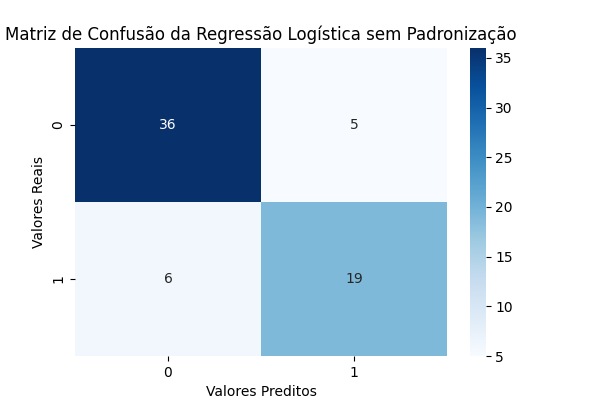
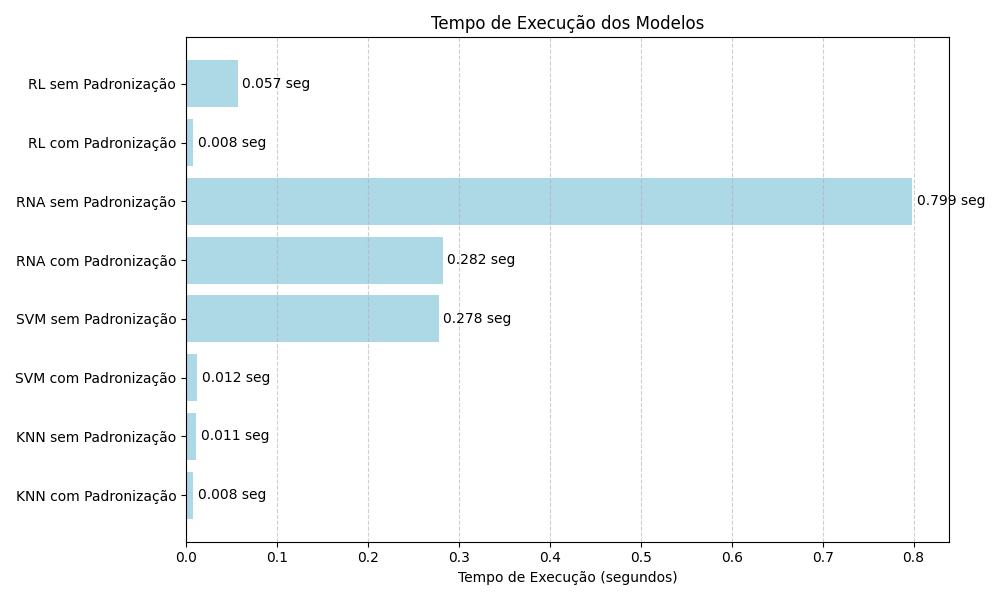


Gráfico de tempo de execução dos modelos



Acurácia dos modelos

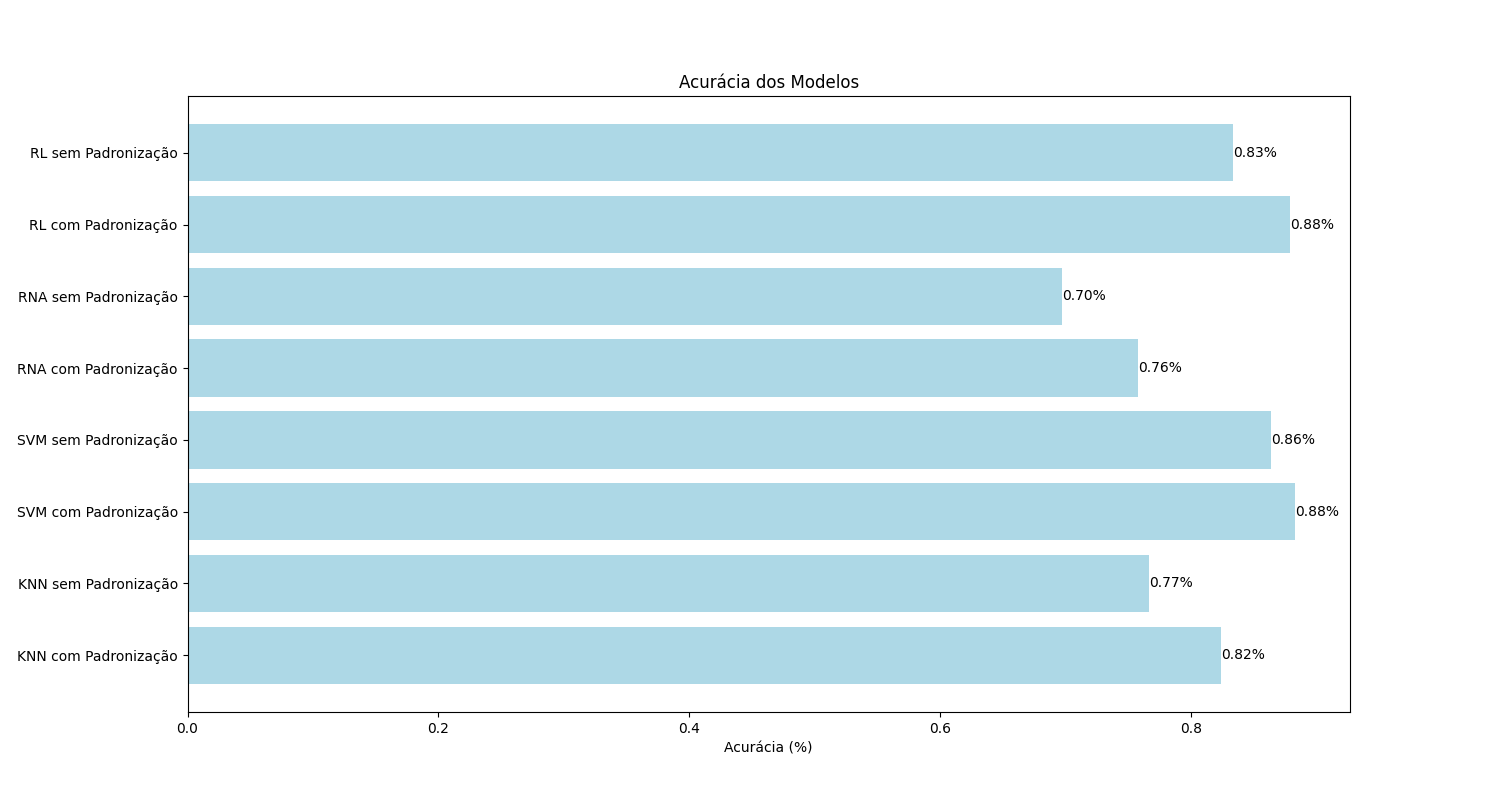


Tabela de acurácia x tempo

|  |  |  |
| --- | --- | --- |
| 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 |