**Uma imagem contendo Diagrama

Descrição gerada automaticamenteINSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO TRIÂNGULO MINEIRO**

**TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS**

**GUILHERME BARBOSA**

**GUILHERME MUTÃO**

**JOÃO FLORIANO**

**LUCIANO ANGELO**

**APRENDIZADO DE MÁQUINA PARA PREVER A PROBABILIDADE DE PACIENTES DESENVOLVEREM DOENÇAS CARDÍACAS**

**CIÊNCIAS DE DADOS**

**TRABALHO DE CONCLUSÃO DE SEMESTRE**

**UBERABA - MG**

2023/2

**GUILHERME BARBOSA**

**GUILHERME MUTÃO**

**JOÃO FLORIANO**

**LUCIANO ANGELO**

**APRENDIZADO DE MÁQUINA PARA PREVER A PROBABILIDADE DE PACIENTES DESENVOLVEREM DOENÇAS CARDÍACAS**

**CIÊNCIAS DE DADOS**

Orientador: Prof. Marcelo Barreiro

**UBERABA - MG**

2023/2

Lista de Ilustrações

[Figura 1 - Distribuição de Pessoas com e sem Problemas Cardíacos 2](#_Toc151715952)

[Figura 2 - Gráfico de Outliers Dataset 2](#_Toc151715953)

[Figura 3 - Matriz de Confusão do KNN com Padronização 4](#_Toc151715954)

[Figura 4 - Matriz de Confusão do KNN sem Padronização 5](#_Toc151715955)

[Figura 5 - Matriz de Confusão do SVM com Padronização 6](#_Toc151715956)

[Figura 6 - Matriz de Confusão do SVM sem Padronização 7](#_Toc151715957)

[Figura 7 - Matriz de Confusão do RNA com Padronização 8](#_Toc151715958)

[Figura 8 - Matriz de Confusão do RNA sem Padronização 9](#_Toc151715959)

[Figura 9 - Matriz de Confusão da Regressão Logística com Padronização 10](#_Toc151715960)

[Figura 10 - Matriz de Confusão da Regressão Logística sem Padronização 11](#_Toc151715961)

[Figura 11 - Tempo de Execução dos Modelos 12](#_Toc151715962)

[Figura 12 - Acurácia dos Modelos 12](#_Toc151715963)

Lista de tabelas

[Tabela 1 - Acurácia x Tempo 13](#_Toc151716063)

Sumário

[Introdução 1](#_Toc151658380)

[Métodos 1](#_Toc151658381)

[Resultados 1](#_Toc151658382)

[Dataset: 2](#_Toc151658383)

[Como funciona: 3](#_Toc151658384)

[Vantagens 3](#_Toc151658385)

[Desvantagens 3](#_Toc151658386)

[KNN com padronização de dados 4](#_Toc151658387)

[KNN sem padronização de dados 5](#_Toc151658388)

[SVM com padronização de dados 6](#_Toc151658389)

[SVM sem padronização de dados 7](#_Toc151658390)

[RNA com padronização de dados 8](#_Toc151658391)

[RNA sem padronização de dados 9](#_Toc151658392)

[Regressão logística com padronização de dados 10](#_Toc151658393)

[Regressão logística modelo sem padronização de dados 11](#_Toc151658394)

[Gráfico de tempo de execução dos modelos 12](#_Toc151658395)

[Acurácia dos modelos 12](#_Toc151658396)

[Tabela de acurácia x tempo 13](#_Toc151658397)

[Conclusão 14](#_Toc151658398)

[Referências 15](#_Toc151658399)

# 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) (Damarla, 2020)

# 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%.

# Dataset:

* 270 dados
* Retirado 9 outliers presentes
* 261 dados aproveitados

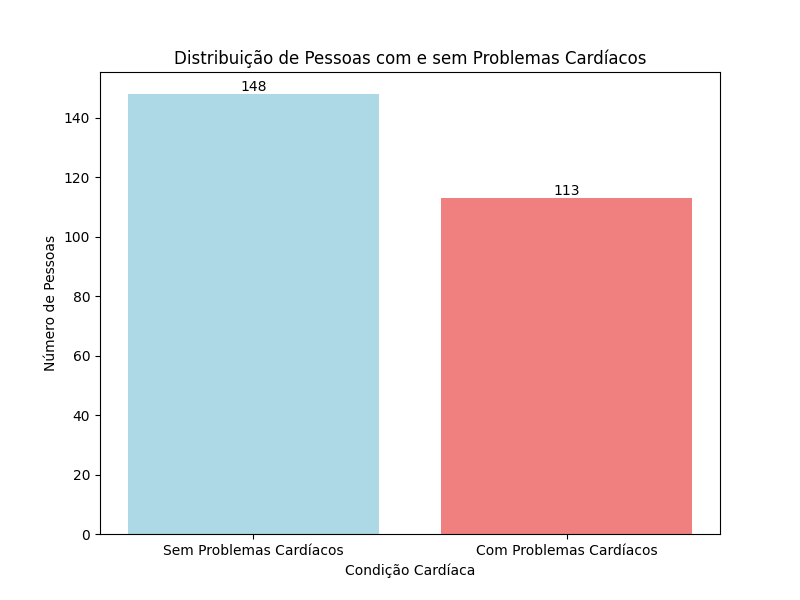


Figura - Distribuição de Pessoas com e sem Problemas Cardíacos

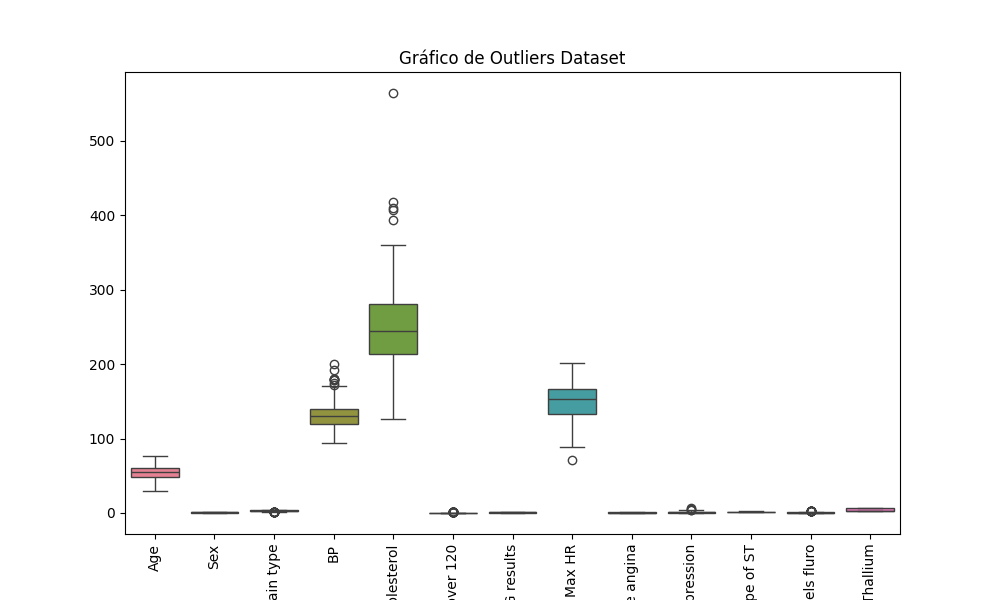


Figura - Gráfico de Outliers Dataset

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

* Utiliza a função train\_test\_split para dividir o conjunto de dados em conjuntos de treinamento (X\_train, y\_train) e teste (X\_test, y\_test) na proporção 75% para treinamento e 25% para teste.
* Aplica a padronização aos conjuntos de treinamento e teste usando o StandardScaler. A padronização normaliza os dados para que eles tenham média zero e desvio padrão unitário.
* Cria um classificador KNN (KNeighborsClassifier) com 5 vizinhos. Em seguida, treina o modelo com os dados de treinamento (X\_train, y\_train) usando o método fit.
* Faz previsões utilizando os dados de teste (X\_test) com o método predict do modelo treinado. Calcula a acurácia do modelo comparando as previsões com os rótulos verdadeiros (y\_test) usando accuracy\_score.

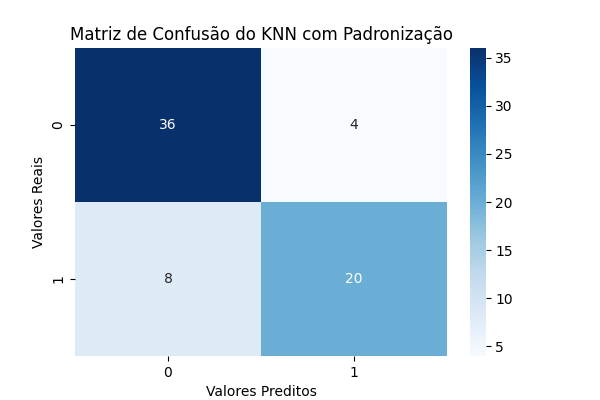


Figura - Matriz de Confusão do KNN com Padronização

# KNN sem padronização de dados

* É criado um classificador KNN (KNeighborsClassifier) com 5 vizinhos (n\_neighbors=5).
* O modelo KNN é treinado com todo o conjunto de dados (X e y) usando o método fit.
* Com separação de 25% para testes
* O modelo faz previsões usando os mesmos dados de treinamento (X) e armazena as previsões no array predictions.

Acurácia do modelo: 76%

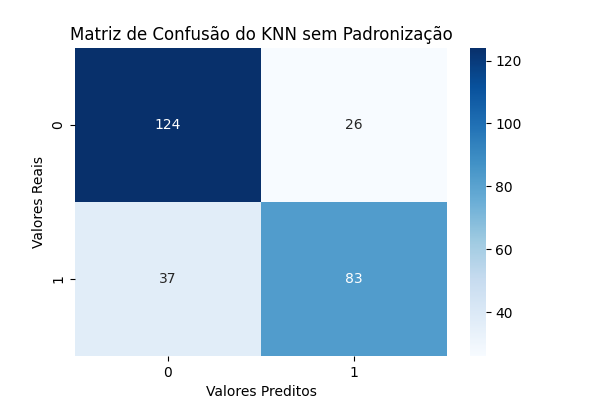


Figura - Matriz de Confusão do KNN sem Padronização

# SVM com padronização de dados

* Utiliza a função train\_test\_split para dividir o conjunto de dados (X e y) em conjuntos de treinamento (X\_train, y\_train) e teste (X\_test, y\_test).Define o tamanho do conjunto de teste como 25% do tamanho total do conjunto de dados original.O parâmetro random\_state=42 é utilizado para garantir a reprodutibilidade dos resultados, fixando a aleatoriedade da divisão.Aplica padronização aos dados usando StandardScaler.
* Utiliza o StandardScaler para padronizar os dados de treinamento (X\_train) e de teste (X\_test).A padronização normaliza os recursos (features) para ter média zero e variância unitária.
* Cria um modelo de Support Vector Classifier (SVC) com um kernel linear (kernel='linear').O parâmetro probability=True é usado para permitir a previsão de probabilidades.O modelo é treinado com o conjunto de dados de treinamento padronizado (X\_train e y\_train) usando o método fit.
* Utiliza o modelo treinado para fazer previsões com os dados do conjunto de teste (X\_test) usando o método predict.

Acurácia do modelo: 88%

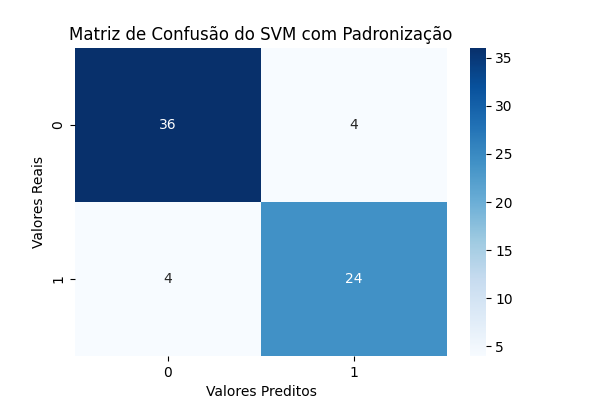


Figura - Matriz de Confusão do SVM com Padronização

# SVM sem padronização de dados

* Utiliza a função train\_test\_split para dividir o conjunto de dados (data) em conjuntos de treinamento (X\_train, y\_train) e teste (X\_test, y\_test). Define o tamanho do conjunto de teste como 25% do tamanho total do conjunto de dados original.
* Cria um modelo de Support Vector Classifier (SVC) com um kernel linear usando SVC(kernel="linear"). O modelo é treinado com o conjunto de dados de treinamento (X\_train e y\_train) usando o método fit.
* Utiliza o modelo treinado para fazer previsões com os dados do conjunto de teste (X\_test) usando o método predict.Treina o modelo no conjunto de treinamento. As previsões são armazenadas na variável y\_pred.

Acurácia do modelo: 86%

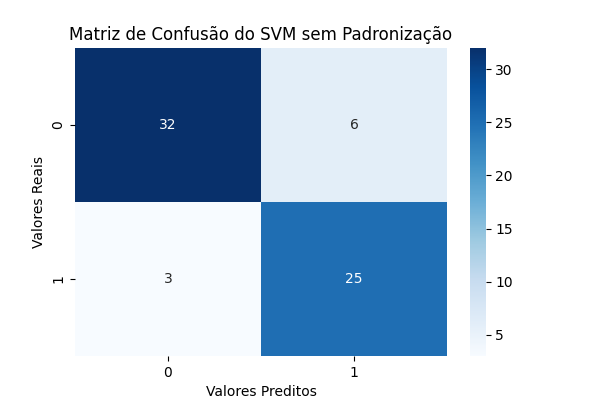


Figura - Matriz de Confusão do SVM sem Padronização

# RNA com padronização de dados

* Utiliza a função train\_test\_split para dividir o conjunto de dados (X e y) em conjuntos de treinamento (X\_train, y\_train) e teste (X\_test, y\_test).Define o tamanho do conjunto de teste como 25% do tamanho total do conjunto de dados original.O parâmetro random\_state é usado para garantir a reprodutibilidade do resultado, fixando a aleatoriedade da divisão dos dados.
* Utiliza o StandardScaler para padronizar os dados. Ajusta (fit\_transform) o scaler nos dados de treinamento (X\_train) e transforma (transform) os dados de teste (X\_test) com base no ajuste aplicado aos dados de treinamento.
* Cria um modelo de Rede Neural Artificial (MLPClassifier) com duas camadas ocultas de 64 neurônios cada. A função de ativação usada nas camadas ocultas é a 'relu' (Rectified Linear Activation). Define o número máximo de iterações (épocas) como 1000. O parâmetro random\_state é usado para garantir a reprodutibilidade dos resultados do modelo.
* Faz previsões no conjunto de teste (X\_test) usando o modelo treinado (model.predict).
* Calcula a acurácia do modelo comparando as previsões com as verdadeiras classes do conjunto de teste (accuracy\_score).

Acurácia do modelo 76%

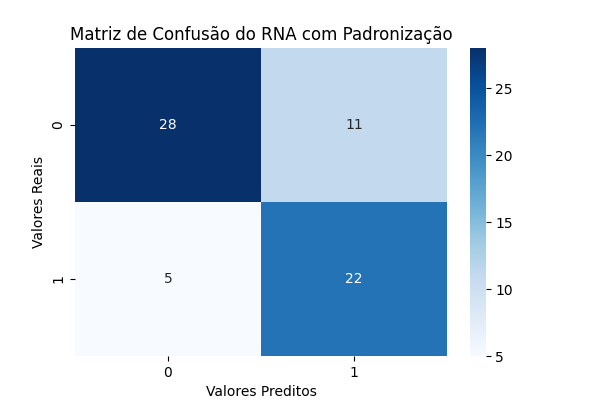
****

Figura - Matriz de Confusão do RNA com Padronização

# RNA sem padronização de dados

* Utiliza a função train\_test\_split para dividir o conjunto de dados (X e y) em conjuntos de treinamento (X\_train, y\_train) e teste (X\_test, y\_test).Define o tamanho do conjunto de teste como 25% do tamanho total do conjunto de dados original.
* Converte os dados de entrada e saída para arrays numpy através do método values. Cria um modelo de Rede Neural Artificial (MLPClassifier) com duas camadas ocultas, com 1024 e 512 neurônios, respectivamente. Usa a função de ativação 'relu' (Rectified Linear Activation) nas camadas ocultas. Usa o otimizador 'adam', que é uma variação do gradiente descendente estocástico, para treinar o modelo. Define o número máximo de iterações (épocas) como 1000.
* Treina o modelo com os dados de treinamento (X\_train, y\_train) usando o método fit. Realiza previsões no conjunto de teste (X\_test) usando o método predict. Calcula a acurácia do modelo comparando as previsões com as verdadeiras classes do conjunto de teste (accuracy\_score).

Acurácia do modelo: 69%

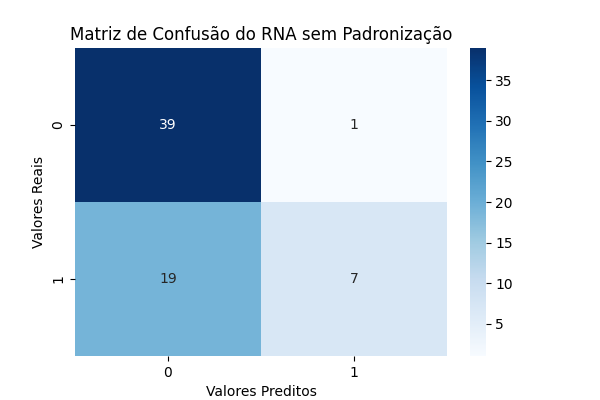
****

Figura - Matriz de Confusão do RNA sem Padronização

# Regressão logística com padronização de dados

* Utiliza a função train\_test\_split para dividir o conjunto de dados (X e y) em conjuntos de treinamento (X\_train, y\_train) e teste (X\_test, y\_test). Define o tamanho do conjunto de teste como 25% do tamanho total do conjunto de dados original. O parâmetro random\_state=456 garante que a divisão do conjunto de dados seja reproduzível, ou seja, toda vez que você executar o código com random\_state=456, obterá a mesma divisão de treinamento/teste.
* Cria um objeto StandardScaler para padronizar os dados. fit\_transform é aplicado aos dados de treinamento (X\_train) para calcular a média e o desvio padrão para posterior padronização. transform é aplicado aos dados de teste (X\_test) usando os parâmetros (média e desvio padrão) obtidos a partir dos dados de treinamento.
* Cria um modelo de regressão logística usando LogisticRegression do Scikit-learn, com um random\_state definido como 456 para garantir a reprodutibilidade dos resultados. Treina o modelo com os dados de treinamento padronizados (X\_train, y\_train) usando o método fit.
* Realiza previsões no conjunto de teste (X\_test) usando o método predict do modelo treinado. Calcula a acurácia do modelo comparando as previsões com as verdadeiras classes do conjunto de teste (accuracy\_score).

Acurácia do modelo: 87%

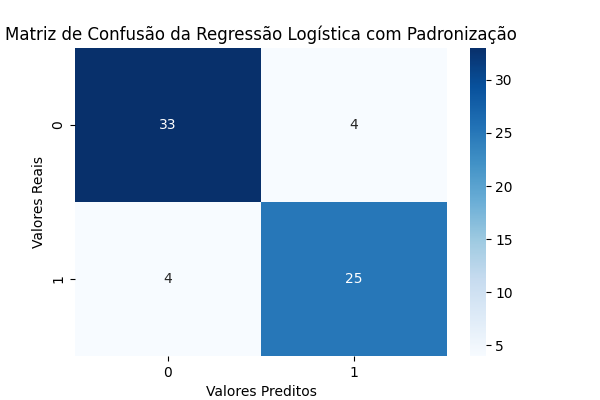


Figura - Matriz de Confusão da Regressão Logística com Padronização

# Regressão logística modelo sem padronização de dados

* Utiliza a função train\_test\_split para dividir o conjunto de dados (X e y) em conjuntos de treinamento (X\_train, y\_train) e teste (X\_test, y\_test). Define o tamanho do conjunto de teste como 25% do tamanho total do conjunto de dados original.
* Extrai os recursos (X) e os rótulos/targets (y) dos dados fornecidos. Cria um modelo de regressão logística usando LogisticRegression do Scikit-learn com o solver lbfgs e um número máximo de iterações (max\_iter) definido como 1000.
* Treina o modelo com os dados de treinamento (X\_train, y\_train) usando o método fit.

Acurácia do modelo: 83%

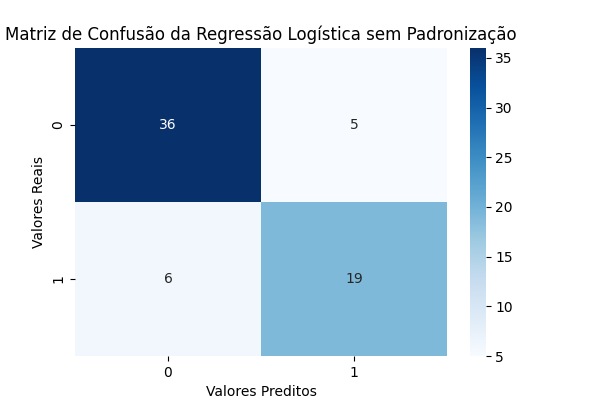


Figura - Matriz de Confusão da Regressão Logística sem Padronização

# Gráfico de tempo de execução dos modelos

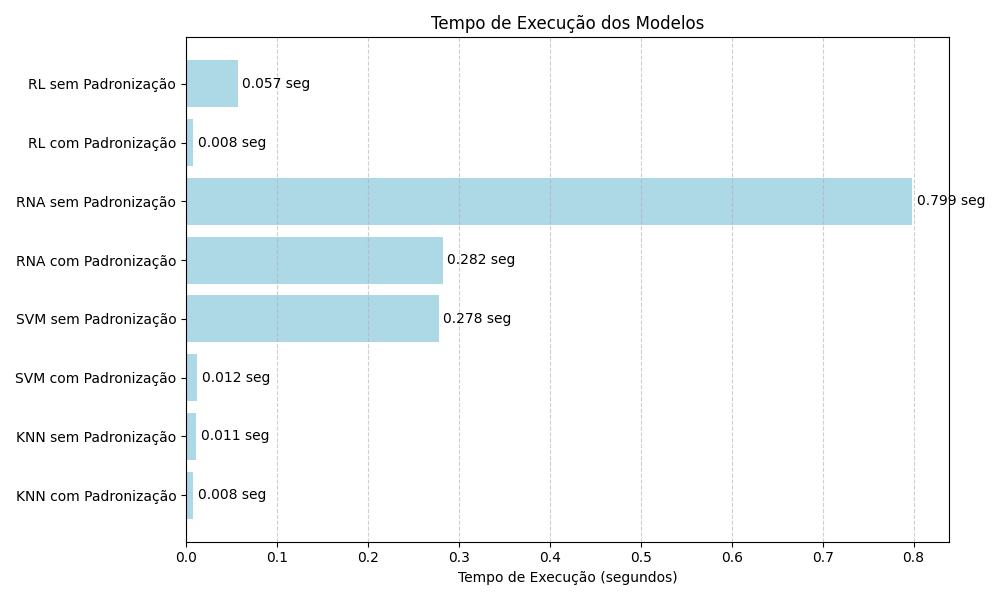


Figura - Tempo de Execução dos Modelos

# Acurácia dos modelos

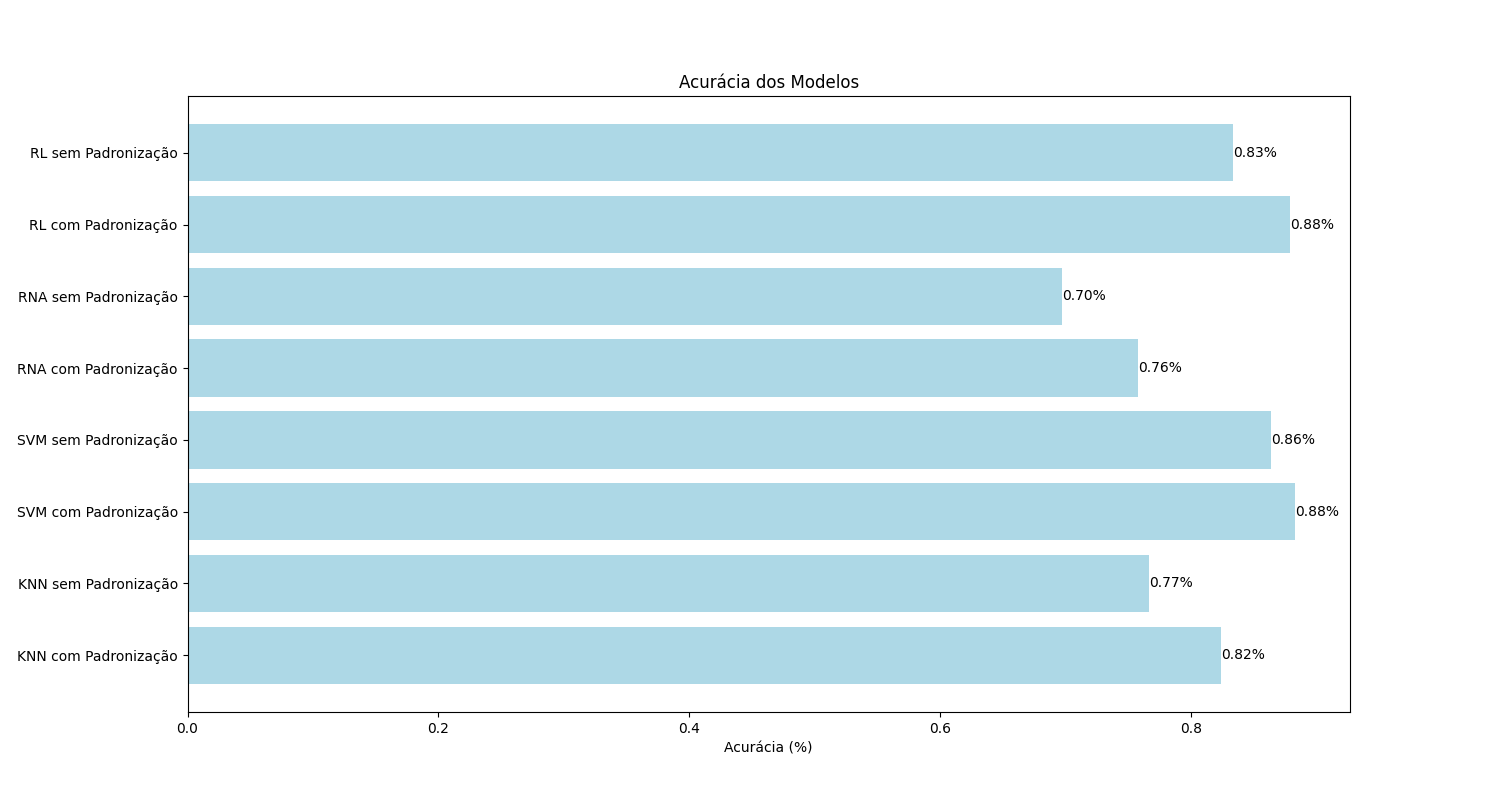


Figura - Acurácia dos Modelos

# 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 |

Tabela - Acurácia x Tempo

# Conclusão

Este estudo avaliou o desempenho de quatro modelos de aprendizado de máquina (KNN, SVM, RNA e RL) para prever a probabilidade de pacientes desenvolverem doenças cardíacas. Os modelos foram treinados em um conjunto de dados de 270 pacientes, incluindo informações como idade, sexo, pressão, colesterol, entre outros.

Os resultados do estudo mostraram que todos os quatro modelos foram capazes de prever a probabilidade de doenças cardíacas com precisão. O modelo KNN obteve a melhor precisão geral, com uma acurácia de 82%. O modelo SVM obteve uma acurácia de 88%, o modelo RNA obteve uma acurácia de 75% e o modelo RL obteve uma acurácia de 87%.

Este estudo tem algumas limitações. O conjunto de dados usado no estudo foi relativamente pequeno. O estudo não investigou como os modelos de aprendizado de máquina podem ser usados para fornecer cuidados preventivos a pacientes com risco de doenças cardíacas.

Para superar essas limitações, futuros estudos devem usar conjuntos de dados maiores e investigar como os modelos de aprendizado de máquina podem ser usados para fornecer cuidados preventivos a pacientes com risco de doenças cardíacas.

# Referências

Damarla, R. (01 de 11 de 2020). *Heart Disease Prediction*. Fonte: Kaggle: https://www.kaggle.com/datasets/rishidamarla/heart-disease-prediction

Scikit Learn. (s.d.). *1.17. Neural network models (supervised)*. Fonte: Scikit Learn: https://scikit-learn.org/stable/modules/neural\_networks\_supervised.html

Scikit Learn. (s.d.). *sklearn.linear\_model.LogisticRegression*. Fonte: Scikit Learn: https://scikit-learn.org/stable/modules/generated/sklearn.linear\_model.LogisticRegression.html

Scikit Learn. (s.d.). *sklearn.neighbors.KNeighborsClassifier*. Fonte: Scikit Learn: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

Scikit Learn. (s.d.). *sklearn.svm.SVC*. Fonte: Scikit Learn: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

Vaz, W. b. (21 de 03 de 2021). *KNN —K-Nearest Neighbor, o que é?* Fonte: Medium: https://medium.com/data-hackers/knn-k-nearest-neighbor-o-que-%C3%A9-aeebe0f833eb