FERNANDO FELICIANO GODOY

Dispositivo de software que usa aprendizado de máquina para verificação do efeito da realização de exercícios físicos na glicemia de indivíduos acometidos por diabetes *mellitus* tipo 1

São Paulo
2025

FERNANDO FELICIANO GODOY

Dispositivo de software que usa aprendizado de máquina para verificação do efeito da realização de exercícios físicos na glicemia de indivíduos acometidos por diabetes *mellitus* tipo 1

Versão Original

Monografia apresentada ao PECE – Programa de Educação Continuada em Engenharia da Escola Politécnica da Universidade de São Paulo como parte dos requisitos para a conclusão do curso de MBA em Engenharia de Software.

Área de Concentração: Engenharia de Software

Orientador: Prof. Dr. Paulo Sérgio Muniz Silva

São Paulo
2025

Catalogação-na-publicação

Godoy, Fernando

Dispositivo de software que usa aprendizado de máquina para verificação do efeito da realização de exercícios físicos na glicemia de indivíduos acometidos por diabetes mellitus tipo 1 / F. Godoy -- São Paulo, 2025.

69 p.

Monografia (MBA em Tecnologia de Software) - Escola Politécnica da Universidade de São Paulo. PECE – Programa de Educação Continuada em Engenharia.

1.Diabetes mellitus tipo 1 2.Aprendizado de máquina 3.Amostras virtuais I.Universidade de São Paulo. Escola Politécnica. PECE – Programa de Educação Continuada em Engenharia II.t.

Nome: GODOY, Fernando

Título: Dispositivo de software que usa aprendizado de máquina para verificação do efeito da realização de exercícios físicos na glicemia de indivíduos acometidos por diabetes *mellitus* tipo 1

Monografia apresentada ao PECE – Programa de Educação Continuada em Engenharia da Escola Politécnica da Universidade de São Paulo como parte dos requisitos para a conclusão do curso de MBA em Engenharia de Software.

Aprovado em: / /

Banca Examinadora

Prof(a). Dr(a). _____

Instituição: _____

Julgamento: _____

Prof(a). Dr(a). _____

Instituição: _____

Julgamento: _____

Prof(a). Dr(a). _____

Instituição: _____

Julgamento: _____

DEDICATÓRIA

Dedico este trabalho à minha mãe,
por todo o apoio que me deu durante
a minha formação acadêmica.

AGRADECIMENTOS

Agradeço à minha família, especialmente à minha mãe e ao meu avô, ambos incansáveis patrocinadores da minha educação.

Agradeço imensamente ao Professor Paulo Muniz, que aceitou me orientar nesta monografia, me ajudou a manter o foco para cumprir os prazos e contribuiu decisivamente para que eu encontrasse bons caminhos de pesquisa.

Agradeço ao Dr. Carlos Minanni, endocrinologista, especialista no tratamento de diabetes, não só pelos quase 10 anos de cuidado com minha saúde, mas também pela ajuda na identificação de um problema específico para ser abordado neste trabalho e pela revisão de conceitos médicos aqui apresentados.

Aos meus amigos, colegas de trabalho e colegas no curso, Arthur Machado, Luan Sales, Pedro Vieira e Renan Ferreira pela companhia nessa jornada.

À todos os professores do curso do MBA em Engenharia de Software pelos ensinamentos ao longo dos últimos dois anos.

RESUMO

GODOY, Fernando**. Dispositivo de software que usa aprendizado de máquina para verificação do efeito da realização de exercícios físicos na glicemia de indivíduos acometidos por diabetes *mellitus* tipo 1.** 2025. 69 p.. Monografia (MBA em Engenharia de Software). Programa de Educação Continuada em Engenharia da Escola Politécnica da Universidade de São Paulo. São Paulo. 2025.

O objetivo desta monografia é desenvolver um dispositivo de software, utilizando técnicas de aprendizado de máquina, para a predição de hipoglicemias em indivíduos acometidos por diabetes *mellitus* tipo 1 (DM1) após a prática de exercícios físicos, visando alcançar acurácia e revocação superiores a 80% e precisão acima de 60%. Para isso, este trabalho utiliza um protocolo de prova de conceito (PoC) e o auxílio de um especialista na área médica, para a realização de ensaios de exercícios físicos com um indivíduo com DM1 (o próprio autor), cujos dados de glicemia foram coletados por um sensor de monitoramento em tempo real. No desenvolvimento do protótipo do dispositivo de software proposto, foram empregados os algoritmos de k-vizinhos mais próximos, árvore de decisão e floresta aleatória. Diante das restrições para a elaboração do trabalho, obteve-se um conjunto pequeno de amostras. Para contornar tais restrições, utilizou-se uma técnica de geração de amostras virtuais. Os resultados indicaram que essa técnica melhorou significativamente o desempenho dos modelos de árvore de decisão e floresta aleatória. Dentre os modelos avaliados, a árvore de decisão apresentou os melhores resultados, alcançando acurácia média de 66% em validação cruzada, revocação de 75% e precisão de 56%. Apesar da melhoria em relação aos resultados iniciais, as três métricas ficaram abaixo do objetivo estabelecido, sugerindo a necessidade de aprimoramento dos modelos ou de uma coleta de dados maior e mais diversificada, ou mesmo um estudo com especialistas da área médica para avaliarem se os valores estabelecidos para as métricas estão muito restritivos.

Palavras-chave: Diabetes *mellitus* tipo 1. Hipoglicemia. Aprendizado de máquina. Inteligência artificial. K-vizinhos mais próximos. Árvore de decisão. Floresta aleatória. Amostras virtuais. Validação cruzada. Acurácia. Precisão. Revocação.

ABSTRACT

GODOY, Fernando. **Software device using machine learning in order to verify the effect of physical exercise realization on blood glucose levels in individuals with type 1 diabetes *mellitus***. 2025. 69 p.. Monografia (MBA em Engenharia de Software). Programa de Educação Continuada em Engenharia da Escola Politécnica da Universidade de São Paulo. São Paulo. 2025.

The objective of this monograph is to develop a software device, using machine learning techniques, to predict hypoglycemia in individuals with type 1 diabetes *mellitus* (T1DM) after physical exercise, aiming to achieve accuracy and recall above 80% and precision above 60%. To this end, this study used a proof-of-concept (PoC) protocol and the assistance of a medical specialist to conduct physical exercise tests with an individual with T1DM (the author himself), whose blood glucose data was collected using a real-time monitoring sensor. In the development of the prototype of the proposed software device, the k-nearest neighbors, decision tree, and random forest algorithms were employed. Given the limitations of the work, a small dataset was obtained. To overcome these limitations, a virtual sample generation technique was used. The results indicated that this technique significantly improved the performance of the decision tree and random forest models. Among the evaluated models, the decision tree presented the best results, achieving an average accuracy of 66% in cross-validation, recall of 75%, and precision of 56%. Despite the improvement compared to the initial results, all three metrics fell below the established goal, suggesting the need for model refinement or a larger and more diverse data collection, or even a study with medical specialists to evaluate whether the established values for the metrics are too restrictive.

Keywords: Type 1 diabetes *mellitus*. Hypoglycemia. Machine learning. Artificial intelligence. K-nearest neighbors. Decision tree. Random forest. Virtual samples. Cross-validation. Accuracy. Precision. Recall.

LISTA DE ILUSTRAÇÕES

Pág.

LISTA DE TABELAS

Pág.

LISTA DE ABREVIATURAS E SIGLAS

AM	Aprendizado de máquina
API	*Application Programming Interface*
AUC	Área sob a curva (*area uder the curve*)
CEP	Comitê de Ética em Pesquisa
DM1	Diabetes *mellitus* tipo 1
DM2	Diabetes *mellitus* tipo 2
HTTP	*HyperText Transfer Protocol*
IA	Inteligência Artificial
PIB	Produto interno bruto
PoC	Prova de Conceito (*Proof of Concept*)
ROC curve	Curva de característica da operação (*Receiver operating characteristic)*

SUMÁRIO

Pág.

1. INTRODUÇÃO

1.1 Motivações

A diabete *mellitus* é uma doença crônica que atinge milhões de pessoas ao redor do mundo. De acordo com a International Diabetes Federation (2021), 537 milhões de pessoas entre 20 e 79 anos de idade viviam com a doença em 2021, número superior ao triplo da quantidade observada em 2000. Somente no Brasil, o número é de cerca de 16 milhões.

A doença deixa consequências tanto para os pacientes quanto para o sistema de saúde. De acordo com a Sociedade Brasileira de Diabetes (2023), o controle inadequado da doença pode trazer complicações aos pacientes, como retinopatia, doença renal, neuropatia, doença cardiovascular, entre outras. Já os gastos relacionados à diabete nas Américas do Sul e Central são superiores a 1,6% do PIB dessas regiões.

Há dois tipos da doença, chamados de tipo 1 (DM1) e tipo 2 (DM2). A DM2, mais prevalente, atinge cerca de 90% dos casos e está relacionada a causas como obesidade e sedentarismo. Já a DM1, em 2021, acometia cerca de 600 mil pessoas no Brasil e, em 2040, estima-se que deva atingir cerca de 1,8 milhão de pessoas (SOCIEDADE BRASILEIRA DE DIABETES, 2023). A DM1 é uma doença autoimune, cujos portadores produzem pouca ou nenhuma insulina, hormônio que controla as concentrações de glicemia no sangue. Dessa forma, os pacientes acometidos por DM1 necessitam de aplicação de insulina exógena para o controle das taxas e, para isso, precisam calcular a dosagem correta de forma a manter os níveis de glicemia dentro de uma faixa considerada normal. Erros no cálculo da dosagem podem levar às condições de hipoglicemia e hiperglicemia (definidas em mais detalhes no capítulo 2) que, a longo prazo, causam as complicações citadas.

Modelos de cálculo da dose adequada de insulina levam em conta desde a quantidade consumida de carboidratos nas refeições até fatores de sensibilidade característicos de cada pessoa (VETTORETTI *et al.*, 2020). Muitos estudos atuais estão sendo desenvolvidos para reduzir a probabilidade dos pacientes de desenvolverem complicações indesejadas causadas por controle inadequado. Particularmente, em intersecção com o campo de Engenharia de Software, a Inteligência Artificial (IA) tem sido amplamente utilizada. De acordo com Vettoretti *et*

al. (2020), destacam-se as três principais tarefas que os modelos modernos de IA têm se proposto a resolver: cálculo personalizado da dose de insulina bolus, ajuste adaptativo dos parâmetros de cálculo de dose de insulina bolus e a predição futura da glicemia. Particularmente, Khanna, Francon e Miikkulainen (2024) descrevem um trabalho com um dispositivo chamado pâncreas artificial que utiliza um modelo de floresta aleatória para calcular a dosagem de insulina a ser aplicada no paciente.

De fato, existem cada vez mais dispositivos e aplicativos comerciais que visam ajudar os pacientes na gestão da diabetes e na tomada das mais de 100 decisões por dia que eles precisam tomar (KHANNA; FRANCON; MIIKKULAINEN, 2024). No entanto, ainda observa-se uma carência de dispositivos que estudem ou simulem o efeito da realização de exercícios físicos no tratamento dos diabéticos.

1.2 Objetivo

Este trabalho desenvolve um dispositivo de software que implementa modelos de aprendizado de máquina que analisam dados extraídos de ensaios de exercícios físicos de um indivíduo acometido por DM1, em cenários parcialmente controlados, para prever se a realização dos exercícios sob a condição das características observadas leva a um estado de hipoglicemia ou não.

Tendo em vista que o cenário de hipoglicemia requer uma ação mais imediata do paciente, os modelos são treinados para fazerem essa predição. A partir dos modelos treinados, o dispositivo proposto classifica se novas entradas, não presentes no conjunto de treinamento, levam a um estado de hipoglicemia ou não.

Questão de Pesquisa.

Este trabalho pretende responder à seguinte questão: Como implementar um dispositivo de software que possa prever a ocorrência de hipoglicemias em indivíduos acometidos por DM1, após a realização de exercícios físicos?

Hipótese de Pesquisa.

O dispositivo de software implementado, utilizando técnicas de aprendizado de máquina, deve prever a ocorrência de hipoglicemias em indivíduos acometidos por DM1, após a realização de exercícios físicos, com acurácia e revocação superiores a 80%, mantendo a precisão acima de 60%.

1.3 Justificativa

Caso o dispositivo seja bem sucedido, verifica-se que indivíduos com DM1 poderiam utilizá-lo na ocasião da realização de um exercício físico. De posse do resultado previsto pelo dispositivo, caso a previsão seja de um cenário de hipoglicemia, ações preventivas poderiam ser tomadas. Presume-se, portanto, que o dispositivo proposto possa minimizar complicações de saúde aos usuários, e, em consequência, reduzir custos com saúde.

1.4 Método de Pesquisa

O desenvolvimento do dispositivo de software, que implementa modelos de aprendizado de máquina, será feito seguindo um protocolo de uma Prova de Conceito (PoC – *Proof of Concept*).

Um estudo sobre o método de pesquisa de PoC, utilizando uma revisão de literatura sobre o assunto, é descrito por Neto, Borges e Roque (2018, p. 5). Uma das conclusões sobre o que caracteriza uma Prova de Conceito é que ela designa

> uma prática de pesquisa e pode ser caracterizada como um instrumento para a construção de conhecimento com a submissão de objetos de qualquer natureza a um conjunto de atividades, com o objetivo de contribuir com o entendimento e conhecimento desses objetos sob estudo, no desenvolvimento e adoção de novos produtos ou tecnologias por organizações e seus atores. (tradução nossa)

No presente trabalho, um protótipo do software proposto será desenvolvido como PoC, com o objetivo de avaliar a satisfação da hipótese de trabalho. Ainda segundo esses autores, o estudo mostrou que há uma grande variedade de interpretações sobre o significado de PoC, mas que nas pesquisas em Sistemas de Informação ela tende a ser interpretada como uma *prova por demonstração*, seguindo um padrão que articula quatro etapas: problema, hipótese, análise e argumentos de demonstração. No presente trabalho, estabelece-se um protocolo prático para a realização de uma PoC. O protocolo define uma série de passos, descritos em seguida, que procuram atender àquelas etapas:

a) **Passo 1:** as etapas **problema** e **hipótese** são aglutinadas neste primeiro passo, em que se apresenta a definição do problema, o estabelecimento de objetivos e a hipótese de trabalho. Para tanto, procura-se definir os conceitos subjacentes ao escopo do problema. No

presente trabalho, este passo está fragmentado nas seções 1.1 e 1.2 acima, e no capítulo 2 adiante, para seguir o formato estabelecido para o texto desta monografia;

b) **Passo 2:** como se trata da construção de um artefato, neste segundo passo selecionam-se tecnologias e ferramentas a serem utilizadas para permitir a realização da terceira etapa do padrão: a **análise**. As seleções estão justificadas na seção 3.2;

c) **Passo 3:** descreve-se aqui o projeto que permite a criação de um protótipo do artefato visado, utilizando os elementos selecionados no Passo 2. Este passo é apresentado nos capítulos 3 e 4, conforme explicação dada em seguida;

d) **Passo 4:** por fim, neste último passo, realizam-se as etapas de **análise** e **argumentação da demonstração**, que estão compiladas em uma avaliação dos resultados obtidos em ensaios com o protótipo. Este passo é apresentado no capítulo 5.

O Passo 3, que descreve a criação do protótipo, pode ser subdividido em duas etapas: a primeira consiste na obtenção dos dados para treinamento dos modelos de aprendizado de máquina (AM) e a segunda, na codificação do software. A parte de obtenção dos dados está descrita no capítulo 3, enquanto a codificação do software, no capítulo 4. Ressalta-se, como apresentado no capítulo 3, que os dados obtidos são dados do próprio autor (indivíduo acometido por DM1), que, ao autorizar a publicação desses dados, faz com que o presente trabalho não precise ser submetido a um Comitê de Ética em Pesquisa (CEP).

Por fim, para o cumprimento do Passo 4, é necessário, primeiramente, estabelecer critérios para a avaliação dos resultados obtidos com objetivo de concluir se a PoC foi bem sucedida ou mal sucedida, isto é, em que grau os resultados permitem mostrar a satisfação da hipótese de trabalho. As métricas observadas para esse fim são: acurácia, precisão, revocação e F_1 score, definidas no capítulo 2. No capítulo 5, estão presentes as análises dos resultados obtidos.

1.5 Estrutura do Trabalho

O Capítulo 1 INTRODUÇÃO apresenta as motivações, o objetivo, a justificativa, método de pesquisa e a estrutura do trabalho. Neste capítulo, encontram-se descritos os problemas e a hipótese de trabalho, em consonância com o Passo 1 do protocolo de pesquisa estabelecido.

O Capítulo 2 REVISÃO BIBLIOGRÁFICA apresenta conceitos importantes sobre o domínio estudado (diabetes *mellitus* tipo 1), além de conceitos importantes sobre aprendizado de máquina, especialmente sobre os modelos que serão utilizados neste trabalho. Essa revisão conceitual também cumpre requisitos do Passo 1 do protocolo de pesquisa.

O Capítulo 3 PREPARAÇÃO DO AMBIENTE DE SOFTWARE apresenta de forma detalhada o processo de coleta de dados, com o protocolo adotado para a realização de ensaios de exercícios físicos pelo autor (indivíduo com diabetes tipo 1), desenvolvido em parceria com um especialista do domínio. Essa apresentação descreve etapas importantes para a construção do protótipo, como definido no Passo 3 do protocolo de pesquisa. Além disso, este capítulo também justifica as escolhas de tecnologias empregadas no protótipo do artefato de software desenvolvido, de acordo com o que diz o Passo 2.

O Capítulo 4 CONSTRUÇÃO DO ARTEFATO DE SOFTWARE apresenta uma visão arquitetônica do software almejado, além do processo de codificação adotado. Este capítulo complementa a realização do Passo 3 do protocolo de pesquisa.

O Capítulo 5 ANÁLISE DE RESULTADOS descreve os resultados obtidos, as conclusões que podem ser tomadas a respeito desses resultados, além de dificuldades encontradas no trabalho. A discussão sobre os resultados atende ao Passo 4 do protocolo estabelecido. Neste capítulo, também se aprofunda no problema de modelos de aprendizado de máquina com conjuntos amostrais pequenos. Uma solução é apresentada e novos resultados são analisados.

O Capítulo 6 CONSIDERAÇÕES FINAIS apresenta as conclusões e as dificuldades econtradas e propõe trabalhos futuros que complementarão esse tema.

REFERÊNCIAS relaciona o conjunto bibliográfico utilizado na elaboração desta monografia.

APÊNDICE A apresenta uma descrição das premissas adotadas na modelagem do algoritmo de geração de rótulos para as amostras virtuais, introduzidas na seção 5.2.

APÊNDICE B apresenta o conjunto completo de dados coletados, seguindo protocolo definido na seção 3.1.

APÊNDICE C apresenta os trechos de código para a implementação do protótipo do dispositivo de software proposto.

2. REVISÃO BIBLIOGRÁFICA

Este capítulo apresenta um resumo de conceitos básicos de diabetes *mellitus* tipo 1 e de conceitos básicos de modelos de aprendizado de máquina, incluindo uma descrição dos algoritmos utilizados no presente trabalho.

2.1 Conceitos básicos de diabetes *mellitus* tipo 1

A molécula de glicose é o principal combustível da respiração celular, processo responsável pela produção de energia nas células. A insulina é um hormônio produzido em uma região do pâncreas, responsável tanto por facilitar a entrada da glicose disponível no sangue (por produto da digestão ou da queima de reservas) nas células, quanto por transformar os excessos não consumidos em reservas (PETERSEN; SHULMAN, 2018).

A concentração de glicose no sangue recebe o nome de glicemia e normalmente é medida em miligramas por decilitro (mg/dL). O mecanismo de controle dos níveis de glicemia no sangue é complexo e depende da ação de alguns hormônios:

a) a insulina, como descrito anteriormente, age no sentido de **reduzir** as concentrações de glicemia;
b) os hormônios glucagon, adrenalina, cortisol e GH (hormônio do crescimento) são chamados de contrarreguladores, atuando no sentido de **aumentar** as concentrações de glicemia.

A faixa normal de glicemia em jejum é entre 70 mg/dL e 99 mg/dL. Esse valor tende a aumentar de forma controlada após as refeições, retornando ao valor de referência dentro de algumas horas. Quando o nível fica abaixo de 70 mg/dL, ocorre uma condição denominada **hipoglicemia**, que causa sintomas como tremor, suor, calafrios, confusão mental, fome, entre outros (CAMPOS, 2023).

Existe também a condição de **hiperglicemia**, que, no entanto, é um pouco mais complexa de se caracterizar, pois depende se o momento avaliado é pré-prandial (antes de uma refeição) ou pós-prandial (após uma refeição). Por exemplo, a hiperglicemia pré-prandial ocorre com valores de glicemia superiores a 125 mg/dL e a

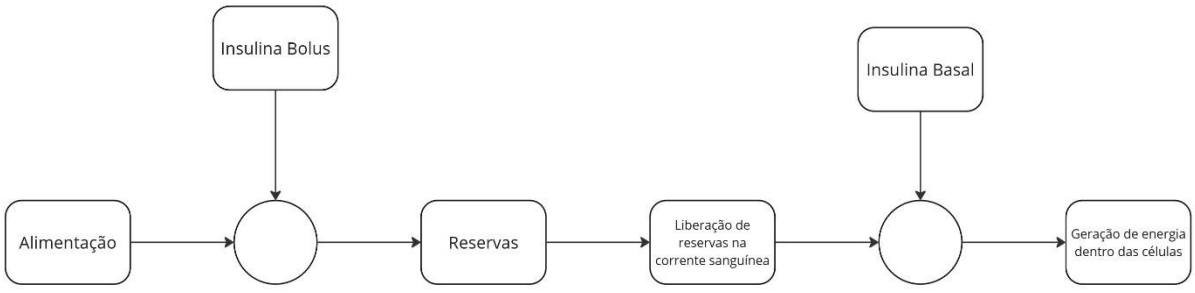
hiperglicemia pós-prandial, com valores superiores a 180 mg/dL (MOURI; BADIREDDY, 2023).

O modelo de aprendizado de máquina proposto nesse trabalho realiza uma classificação de hipoglicemia apenas, seguindo o critério objetivo do limiar de 70 mg/dL.

Como exposto no capítulo 1, a diabete *mellitus* tipo 1 é uma doença autoimune. Isso significa que anticorpos do indivíduo doente atacam células do próprio corpo. Nesse caso, os anticorpos atacam as células específicas do pâncreas responsáveis pela produção do hormônio insulina. Dessa maneira, o mecanismo de regulação dos níveis de glicemia descrito anteriormente para indivíduos saudáveis fica desbalanceado, perdendo o único agente que possibilita a diminuição da glicemia (NEVES *et al.*, 2017).

Em um tratamento comum para DM1, normalmente o paciente faz uso de dois tipos de insulina, uma chamada de basal e outra chamada de bolus, ou prandial. A insulina basal é tomada, em geral, uma vez ao dia e possui uma curva de ação quase plana, com o objetivo de atuar 24 horas por dia, podendo levar glicose das reservas do corpo para dentro das células para produzir energia. Já a insulina bolus é injetada durante as refeições. Possui um tempo de ação de algumas horas, visando atuar no tempo da digestão dos alimentos, onde em geral, geram-se picos de glicemia (NEVES *et al.*, 2017). Essa dose de insulina então atua para a transformação desses picos de glicemia em reservas no organismo. Essa dinâmica está ilustrada na Figura 1.

Figura 1 – Dinâmica de atuação das insulinas exógenas no corpo



Fonte: elaborado pelo autor.

Como a aplicação de insulina ocorre de maneira manual, em geral, precisa-se calcular corretamente a quantidade a ser infundida. A Equação 1 apresenta a fórmula

padrão para cálculo da dose de insulina a ser tomada por pacientes com DM1 (VETTORETTI *et al.*, 2020).

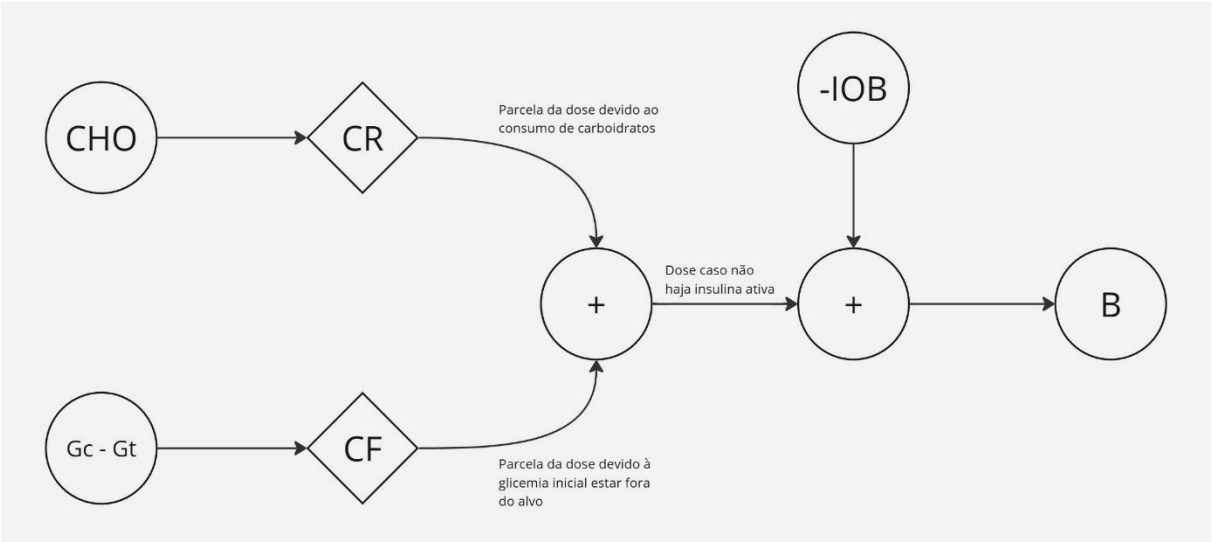
$$B = \frac{CHO}{CR} + \frac{G_C - G_T}{CF} - IOB \quad (1)$$

Os elementos dessa expressão são:

a) B é a dose de insulina bolus a ser aplicada;
b) CHO é a quantidade de gramas de carboidratos consumidas na refeição;
c) CR é um fator de contagem de carboidratos, específico para cada indivíduo;
d) G_C é a glicemia do paciente no início da refeição;
e) G_T é a glicemia alvo do paciente, considerada ideal para controle adequado;
f) CF é um fator de correção, também específico para cada paciente;
g) IOB é a insulina ativa no organismo do paciente.

A dinâmica da lógica do modelo que representa essa equação pode ser observada na Figura 2.

Figura 2 – Modelo conceitual da Equação 1



Fonte: elaborado pelo autor.

Nota-se que, nesse modelo, assume-se que apenas os carboidratos das refeições afetam nos níveis de glicemia, o que se trata de uma simplificação, visto que

outros nutrientes, como proteínas e lipídeos também geram efeitos na glicemia, embora menos diretos. Também assume-se um modelo linear, tanto em relação à parcela da dose devido ao consumo de nutrientes na refeição, quanto em relação à parcela de correção de desvios da glicemia.

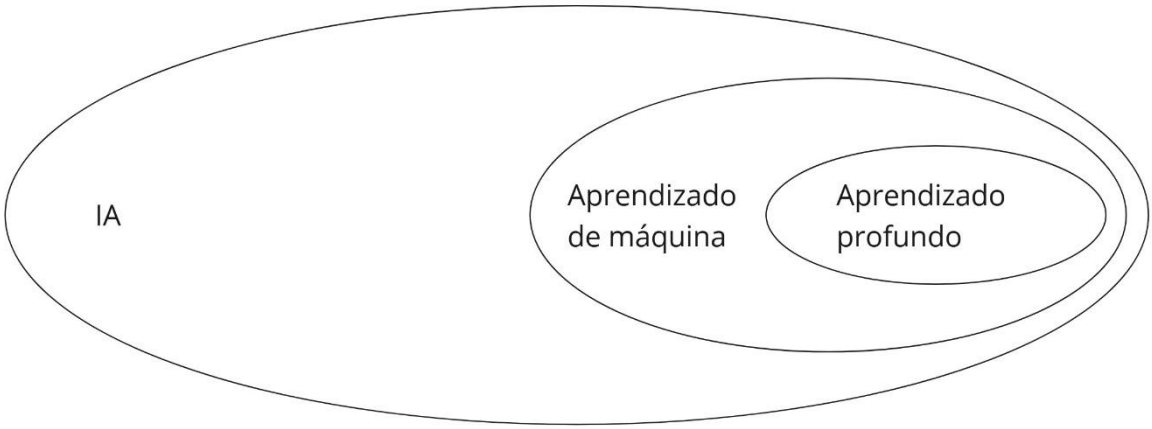
Erros no tratamento já são passíveis de acontecer devido a essas simplificações do modelo. Além disso, os pacientes também estão sujeitos a outros tipos de erros. Por exemplo: pode haver erro dos aparelhos que medem a glicemia no momento da refeição, levando a um erro no cálculo dessa parcela. Também pode haver um erro na estimativa de carboidratos ingeridos na refeição. Uma estimativa maior que a realidade pode levar a uma dose de insulina maior do que a devida. Já uma estimativa menor pode levar a uma dose menor que a devida.

A Equação 1 constitui o modelo padrão de tratamento para DM1. Recentemente, no entanto, um novo tipo de equipamento tem trazido novas oportunidades: os sensores de glicemia em tempo real. Um exemplo é o sensor FreeStyle Libre, da Abbott. Esse sensor, acoplado ao corpo do paciente, afere a glicemia intersticial, nos tecidos, a cada poucos minutos, de forma a conseguir construir uma curva que mostra o comportamento da glicemia ao longo de todo o dia. A partir desses dados é possível extrair a taxa de variação da glicemia (ROC – *rate of change*). Esse parâmetro pode ser extrapolado para se tentar prever a glicemia futura a partir da glicemia instantânea. Visando melhorar o controle de glicemia dos pacientes, os pesquisadores têm tentado utilizar a ROC para melhorar o cálculo da dose de insulina bolus a ser aplicada (VETTORETTI *et al.*, 2020).

2.2 Conceitos básicos de modelos de aprendizado de máquina

O termo *Inteligência Artificial* (IA) remonta à década de 1950, sendo cunhado pelo cientista da computação John McCarthy. O *Aprendizado de Máquina* (AM) é uma subárea do domínio de inteligência artificial, como mostra a Figura 3. O principal objetivo dessa área de estudo é preparar um modelo com dados conhecidos de forma que resultados relevantes sejam gerados quando o modelo receber dados desconhecidos (KNEUSEU, 2024).

Figura 3 – Domínio de conhecimento de IA



Fonte: KNEUSEU, 2024, p. 18.

Uma modelo de aprendizado de máquina funciona como

> uma caixa preta que aceita, geralmente uma coleção de números, e gera uma saída, normalmente um **rótulo** [categórico], como ‘cachorro’ ou ‘gato’, ou um valor contínuo, como a probabilidade se der um ‘cachorro’, ou o valor de um imóvel com as características atribuídas ao modelo (KNEUSEL, 2024, p. 19-20, negrito nosso).

O conjunto de entrada é normalmente expresso por um conjunto de características, ou atributos.

> Em aprendizado de máquina, um *atributo* é um tipo de dado (por exemplo, ‘quilometragem’), ao passo que *característica* assume diversos significados; dependendo do contexto, geralmente significa um atributo mais o seu valor (por exemplo: ‘Quilometragem = 15.000’). Muitas pessoas utilizam as palavras *atributo* e *característica* como sinônimos (GÉRON, 2021, p. 9, itálico do autor).

Dessa forma, uma das maneiras de se classificar os modelos de AM é com relação à presença ou ausência das saídas conhecidas para as amostras de treinamento. Por exemplo, podem ser classificados como **supervisionados**, **não supervisionados**, **semissupervisionados** ou **aprendizado por reforço**. Nos modelos supervisionados, o conjunto de dados de treinamento inclui um rótulo com as soluções desejadas, enquanto nos modelos não supervisionados, não há esse rótulo. Os modelos semissupervisionados são modelos intermediários, que contêm algumas instâncias de treinamento rotuladas e outras não. Por fim, o aprendizado por reforço estabelece uma distribuição de recompensas ou penalidades para que o sistema de aprendizado aprenda sozinho a melhor política a se tomar (GÉRON, 2021).

Dentre os principais tipos de modelos de aprendizado supervisionado estão os modelos de **classificação** e os modelos de **regressão**. Nos modelos classificadores, cada elemento do conjunto de dados de treinamento recebe um rótulo com a categoria a que pertence. A ideia é que quando receber um dado novo, fora do conjunto de treinamento, o modelo consiga classificá-lo de acordo com o que aprendeu dos outros. Já nos modelos de regressão, o rótulo associa cada elemento do conjunto de treinamento a um valor. Dessa forma, ao chegar um elemento novo, o modelo calcula qual deveria ser o valor para ele.

Na realização deste trabalho, o conjunto de dados possui rótulos categóricos, indicando se cada elemento, obtido da realização de exercícios físicos, leva a um cenário com presença ou ausência de hipoglicemia nas duas horas posteriores à realização do exercício, como descrito mais detalhadamente no capítulo 3. A partir desse conjunto de dados, são treinados três modelos de AM para comparação entre eles: **k-vizinhos mais próximos**, **árvore de decisão** e **floresta aleatória**. Os três algoritmos são explicados nas subseções 2.2.1 a 2.2.3.

Uma das formas de se avaliar o desempenho de modelos classificadores é através da **acurácia**. Essa métrica é calculada com a razão entre o número de amostras classificadas corretamente e o número total de amostras. No entanto, outra forma de realizar essa avaliação com classificadores binários é com as métricas de **precisão** e **revocação**, como mostram as Equações 2 e 3 (GÉRON, 2021).

$$precisão = \frac{TP}{TP + FP} \quad (2)$$

$$revocação = \frac{TP}{TP + FN} \quad (3)$$

Nessas equações, TP é o número de verdadeiros positivos, FP é o número de falsos positivos e FN é o número de falsos negativos. Essa terminologia assume que uma das classes é entendida como a classe **positiva** e a outra, como **negativa**. No presente trabalho, a classe com presença de hipoglicemia é considerada a classe positiva. Dessa forma, uma baixa precisão indica que muitas amostras que, na realidade, não são de cenários de hipoglicemias são incorretamente identificadas como hipoglicemias. Enquanto isso, uma baixa revocação indica que uma alta

proporção de hipoglicemias verdadeiras são incorretamente identificadas como não hipoglicemias. Aumentar a precisão pode vir às custas de diminuir a revocação e vice-versa, em um processo que deve fazer uma análise de compensação (*trade-off*) entre precisão e revocação (GÉRON, 2021). Tendo isso em vista, no caso do modelo implementado neste trabalho, manter uma revocação alta é mais importante que manter uma precisão alta, uma vez que o falso negativo pode tirar o paciente de um estado de alerta devido.

Por fim, uma outra métrica para avaliar o desempenho dos classificadores é através da **F_1 score**. De acordo com Géron (2021, p. 75),

> é pertinente combinar a precisão e a revocação em uma única métrica que se chama *F_1 score [pontuação F_1]* [...]. A F_1 score é a *média harmônica* da precisão e da revocação. [...] A média harmônica dá mais importância aos valores mais baixos. Como resultado, o classificador só obterá um score F_1 alto se a revocação e a precisão forem altas.

2.2.1 Modelo dos k-vizinhos mais próximos

O modelo dos k-vizinhos mais próximos é o mais simples dos modelos de AM (KNEUSEU, 2024) . Para k = 1, para cada nova amostra, o modelo encontra a amostra mais próxima no conjunto de treinamento e atribui o rótulo dessa amostra à nova. Para k > 1, a nova amostra recebe o rótulo mais comum entre as k amostras mais próximas. Em caso de empate, o rótulo é escolhido de maneira aleatória entra os rótulos empatados.

Em caso de amostras representadas como vetores n-dimensionais de características numéricas, o conceito de distância normalmente assume o conceito de distância euclidiana (TAULLI, 2020).

2.2.2 Modelo da árvore de decisão

No contexto de classificação, a árvore de decisão é um algoritmo determinístico que, começando de um nó raiz, começa a fazer perguntas sobre as características da entrada. A depender da resposta, o algoritmo escolhe um caminho e assim sucessivamente até chegar no nó folha, que decide a classificação da amostra (KNEUSEU, 2024).

2.2.3 Modelo da floresta aleatória

É comum que as árvores de decisão não funcionem muito bem. Nesse caso, uma composição de várias árvores de decisão, aleatoriamente distintas entre si, constitui uma floresta aleatória.

A técnica para a construção da floresta de forma que nem todas as árvores sejam idênticas constitui-se de três etapas: *bagging*, seleção aleatória de características e *ensemble*.

Bagging é uma técnica utilizada em estatística que consiste na criação de um novo conjunto de dados, a partir do original, por amostragem aleatória, podendo-se ou não selecionar uma amostra mais de uma vez. A seleção aleatória de características é um processo que faz com que cada árvore da floresta leve em conta apenas um subconjunto do total de características. Por fim, *ensemble* consiste em um processo de apuração das saídas de cada árvore da floresta, de modo a encontrar a saída mais votada pela floresta como um todo. Essas três etapas contribuem para a construção de um modelo em geral com melhor de desempenho, porém menos explicável (KNEUSEU, 2024).

3. PREPARAÇÃO DO AMBIENTE DE SOFTWARE

Este capítulo descreve o protocolo de coleta de dados e apresenta a justificativa das escolhas tecnológicas para a implementação do dispositivo de software.

3.1 Protocolo de coleta de dados

Para o desenvolvimento dos modelos de AM propostos neste trabalho, é fundamental a realização de uma coleta de uma base de dados para treinamento e teste. Alguns desafios para o desenvolvimento de modelos já são conhecidos: quantidade insuficiente de dados de treinamento, dados de treinamento não representativos e características irrelevantes são alguns deles (GÉRON, 2021).

De acordo com Géron (2021, p. 20), “normalmente precisa[-se] de centenas de exemplos, e para problemas complexos, como reconhecimento de imagem ou voz, talvez sejam necessários milhões de exemplos”.

Dadas as restrições para o desenvolvimento do trabalho, o número de dados coletados é de apenas algumas dezenas, não sendo, portanto, grande o suficiente para garantir a mitigação desses desafios. No entanto, como o trabalho se propõe à realização de uma PoC, considera-se que esses problemas são aceitáveis. Ressalta-se que o problema de conjuntos de treinamentos pequenos ainda é enfrentado em diversos problemas reais, sendo a área de saúde uma das mais afetadas (KOKOL; KOKOL; ZAGORANSKI, 2022). Uma discussão mais aprofundada sobre essa temática encontra-se na seção 5.2.

Visando eliminar o máximo possível de variáveis que possam influenciar no resultado, deixando a análise do efeito da realização de exercícios ainda mais complexa nesta PoC, estabeleceu-se, em conjunto com um especialista na área médica, um protocolo para a realização da coleta de dados.

O protocolo estabelece:

a) a realização da coleta será realizada com um paciente diabético tipo 1 (o próprio autor);
b) foi realizado sempre o mesmo tipo de exercício físico. Nesse caso, o paciente utilizou uma esteira, em uma academia de condomínio;
c) a duração do exercício foi fixada em 40 minutos;

d) foram estabelecidos dois tipos de intensidade de exercício: **baixa** e **média**, de acordo com o que preconiza a Tabela 1.

Tabela 1 – Descrição das intensidades de exercícios físicos

Intensidade	Tempo de caminhada (6 km/h – 6,6 km/h)	Tempo de corrida (> 9km/h)
Baixa	40 min	0
Média	20 min – 35 min	5 min – 20 min

Fonte: elaborado pelo autor.

Como citado anteriormente, a seleção de características relevantes é um desafio. Para o caso em questão, baseou-se nos parâmetros da Equação 1, além da taxa de variação da glicemia (ROC – *rate of change*), que, como citado no capítulo 2, tem sido utilizada por estudos como uma forma de aperfeiçoamento da Equação 1 (VETTORETTI *et al.*, 2020). Portanto, em síntese, as características do modelo são:

a) glicemia no início da realização do exercício – G_C (em mg/dL);
b) intensidade do exercício – I (baixa ou média);
c) insulina bolus ativa no corpo – IOB (em unidades);
d) carboidratos ingeridos – CHO (em gramas);
e) taxa de variação da glicemia – ROC [em mg/(dL*min)];
f) fator de contagem de carboidratos – CR (gramas de carboidratos/ unidade de insulina);
g) fator de correção – CF [em mg/(dL*unidade de insulina)].

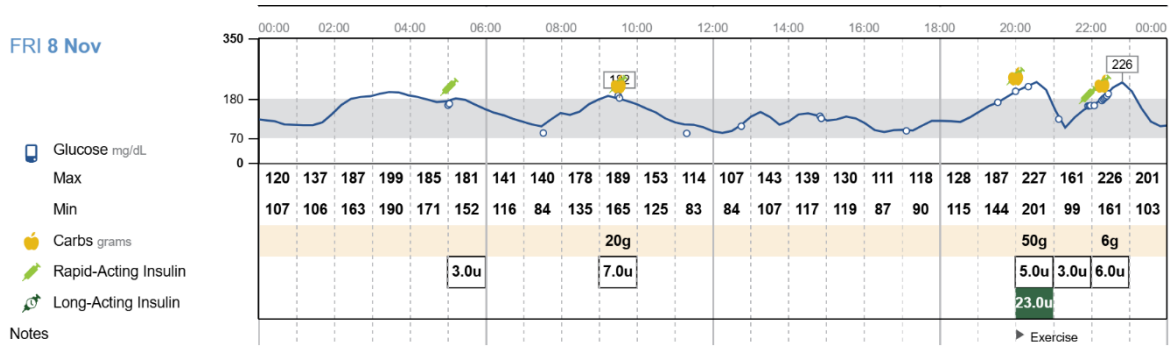
Além disso, o dado de ocorrência ou não de hipoglicemia nas duas horas subsequentes à realização do exercício foi coletado como rótulo para os treinamentos.

Os dados de glicemia são coletados com o uso, pelo paciente, do sensor FreeStyle Libre 2 Plus, que coleta amostras da glicemia intersticial a cada poucos minutos. Essas amostras coletadas são usadas para construir um gráfico da glicemia quase contínuo, como mostra a Figura 4, extraída do relatório de glicose construído pelo aplicativo do sensor.

De posse do gráfico, é possível estimar a ROC da hora anterior à realização de exercício físico, obtendo uma estimativa da derivada da curva neste horário.

Além disso, o paciente também registra as refeições e doses de insulina bolus tomadas.

Figura 4 – Curva diária de glicemia do paciente



Fonte: LibreView.

3.2 Justificativa das escolhas tecnológicas

Para a seleção de tecnologias, destaca-se a biblioteca Scikit-Learn, desenvolvida em Python, que possui fácil usabilidade e implementa muitos algoritmos de aprendizado de máquina de maneira eficiente (GÉRON, 2021). Por esse motivo, essa biblioteca foi selecionada. E, como seu uso se dá em Python, essa mesma linguagem será utilizada no desenvolvimento de outros módulos do artefato de software.

Já a seleção de algoritmos requer uma análise mais detalhada. De acordo com Géron (2021, p. 581), é importante que se:

> 1. Treine muitos modelos rápidos e simples de diferentes categorias (por exemplo, linear, naive Bayes, SVM, floresta aleatória, rede neural etc.) usando parâmetros-padrão.
> 2. Meça e compare seu desempenho [...].
> 3. Analise as variáveis mais significativas para cada algoritmo.
> 4. Analise os tipos de erros que os modelos cometem [...].
> 5. Execute uma rodada rápida de seleção de características e feature engineering.
> 6. Rode mais uma ou duas iterações rápidas das cinco etapas anteriores.
> 7. Faça uma pré-seleção de três a cinco modelos mais promissores, preferindo modelos que cometam diferentes tipos de erros.

Dadas as restrições para a realização deste trabalho, no entanto, a seleção de algoritmos restringiu-se a avaliar os três modelos citados, pelos seguintes motivos:

a) os três modelos se encaixam no problema proposto, por serem **supervisionados** de **classificação**;

b) os três possuem implementação simples com a biblioteca Scikit-Learn, com suporte bibliográfico extenso de como realizar essa implementação.

O teste de mais modelos, como propõe Géron, está listado como trabalho futuro.

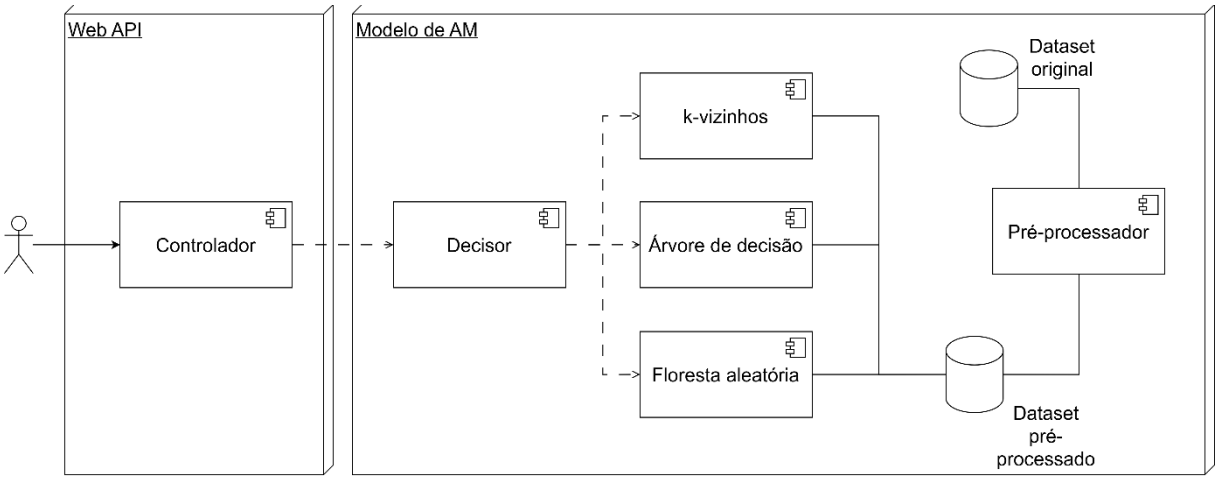
4. CONSTRUÇÃO DO ARTEFATO DE SOFTWARE

Este capítulo apresenta uma descrição da arquitetura do dispositivo de software proposto e do protótipo implementado, do pré-processamento de dados, da validação cruzada dos dados e do treinamento de modelos.

4.1 Arquitetura do dispositivo de software

A arquitetura do software proposto está representada em uma notação semiformal (BASS; CLEMENTS; KAZMAN, 2021, p.331) como mostrado na Figura 5. A visão arquitetônica em foco nessa representação é a de componentes e conectores. Essa visão, de acordo com Bass, Clements e Kazman (2021, p. 335), “mostra elementos que possuem uma presença de [...] processos, serviços, objetos, clientes, servidores ou armazenamento de dados”.

Figura 5 – Arquitetura – Visão de componentes e conectores



Fonte: elaborado pelo autor.

Para a documentação do comportamento do sistema, pode-se utilizar diagramas de atividades UML, que

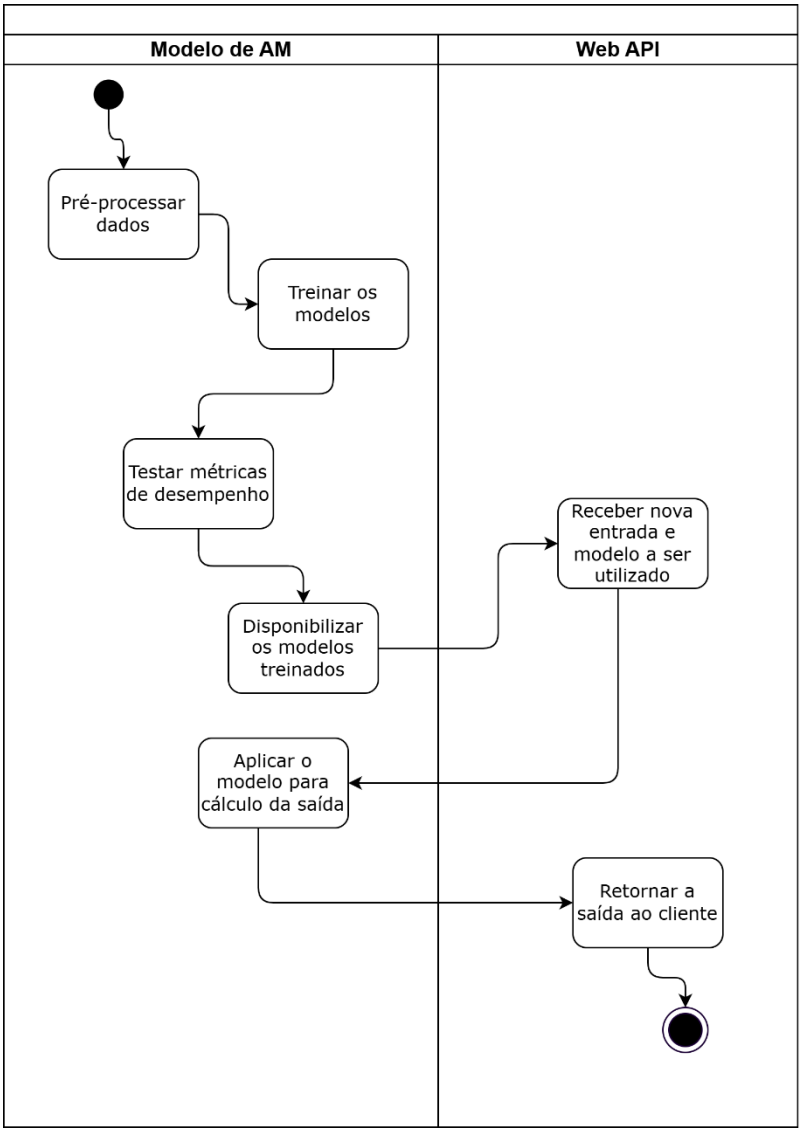
> mostram o processo de negócio como uma sequência de passos (chamados ações) e incluem notação para expressar ramificações condicionais e concorrência, assim como para evidenciar o envio e o recebimento de eventos. Setas entre ações indicam o fluxo de controle (BASS; CLEMENTS; KAZMAN, 2021, p.342).

O comportamento do artefato de software proposto começa com o pré-processamento dos dados coletados, com a codificação de características textuais categóricas e com o escalonamento das outras características. O processo de pré-processamento é detalhado na seção 4.2. Após essa primeira etapa, cada modelo de classificação é treinado com base no conjunto de dados de treinamento. Dessa forma, o resultado são modelos de aprendizado de máquina (AM) na memória da aplicação. Com isso, quando o usuário do software deseja prever o estado final para a uma realização de exercício físico, envia uma requisição HTTP para um controlador na Web API. Dados dessa requisição são repassados ao módulo Decisor, que usa o modelo de AM solicitado pela requisição para a realização da previsão. O diagrama da Figura 6 mostra essa descrição em forma de um diagrama de atividades.

Como mostram as Figuras 5 e 6, cada modelo de AM é implementado em um componente. Esses modelos utilizam a mesma base de dados para treinamento, a qual é uma base de dados pré-processada (de acordo com o descrito na seção 4.2). O Controlador na Web API recebe requisições HTTP e repassa ao Decisor, que identifica o modelo que o usuário quer utilizar e se comunica com o componente específico para utilizá-lo.

O protótipo implementado para esta monografia focaliza o módulo do Modelo de AM, com exceção do componente Decisor, visto que não era necessário para testes de desempenho dos modelos de AM.

Figura 6 – Diagrama de atividades – Visão comportamental do software



Fonte: elaborado pelo autor.

4.2 Pré-processamento de dados

As características do modelo estão apresentadas na Tabela 2, juntamente a uma descrição de seus tipos e valores que se pode esperar de cada uma delas. O rótulo é categórico com valores “SIM” ou “NAO”.

Tabela 2 – Tipos de dados no conjunto de dados original

Característica	Tipo	Valores esperados
G_C	Numérico inteiro	Entre 40 e 400
I	Categórico	{BAIXA, MEDIA}
IOB	Numérico não negativo	Normalmente menor que 10
CHO	Numérico não negativo	Normalmente menor que 50
CR	Numérico não negativo	Normalmente menor que 20
CF	Numérico não negativo	Normalmente menor que 20
ROC	Numérico	Módulo normalmente menor que 5

Fonte: elaborado pelo autor.

A Tabela 3 possui alguns exemplos de valores do conjunto de treinamento.

Tabela 3 – Alguns dados do conjunto de treinamento

G_C	I	IOB	CHO	CR	CF	ROC	Rótulo
175	BAIXA	0,7	0	7	15	0,5	NAO
138	MEDIA	0	15	7	15	0,5	NAO
102	MEDIA	0	15	7	15	0	SIM

Fonte: elaborado pelo autor.

Observa-se, primeiramente, que o atributo **intensidade do exercício** é um atributo textual categórico. De acordo com Géron (2021, p. 18), “a maioria dos algoritmos de AM prefere trabalhar com números”. Dessa forma, a biblioteca Scikit-Learn possui classes que podem converter esses atributos textuais categóricos em números. Como o atributo possui apenas duas categorias possíveis, foi utilizado o OrdinalEncoder de acordo com o fragmento de código apresentado na Figura 7, em Python 3.12.

Figura 7 – Pré-processamento com OrdinalEncoder

```
>>> import pandas as pd
>>> glicemia = pd.read_csv(“./datasets/glicemia/glicemia.csv”,
sep=”;”, thousands=”.”)
>>> glicemia_I = glicemia[[“I”]]
>>> from sklearn.preprocessing import OrdinalEncoder
>>> ordinal_encoder = OrdinalEncoder()
>>> glicemia_I_enconded =
ordinal_enconder.fit_transform(glicemia_I)
```

Fonte: elaborado pelo autor.

Verifica-se que o pré-processamento da coluna “I” para os dados da Tabela 3 resulta nos valores exibidos na Figura 8.

Figura 8 – Resultado do pré-processamento da coluna I

```
array([[0.],
       [1.],
       [1.]])
```

Fonte: elaborado pelo autor.

Verifica-se, portanto, que os valores “BAIXA” foram pré-processados para 0, enquanto os valores “MEDIA”, para 1.

Também é necessário realizar o escalonamento das características, pois, de acordo com Géron (2021, p.55)

> salvo raras exceções, os algoritmos de aprendizado de máquina não funcionam bem quando atributos numéricos de entrada têm escalas muito diferentes. [...] Há duas formas comuns de todos os atributos terem a mesma escala: escalonamento min-max e padronização.
> O escalonamento min-max (que muitos chamam *normalização*) é bastante simples: os valores são deslocados e reescalonados de modo que acabam variando de 0 a 1. [...] A Scikit-Learn fornece um transformador chamado MinMaxScaler para tal. [...] Com a padronização a coisa é diferente: primeiro, ela subtrai o valor médio (por isso os valores padronizados sempre têm média zero) e, em seguida, divide pelo desvio-padrão para que a distribuição resultante tenha variação de unidade. [...] A Scikit-Learn fornece um transformador para padronização chamado StandardScaler.

Neste trabalho, as características G_C, IOB, CHO, CR, e ROC são **normalizadas**, visto que o conjunto de dados de treinamento não apresenta muitos valores atípicos (*outliers*) aparentes. A Figura 9 apresenta o racional para a normalização de G_C.

Figura 9 – Pré-processamento com MinMaxScaler

```
>>> from sklearn.preprocessing import MinMaxScaler
>>> min_max_scaler = MinMaxScaler()
>>> glicemia_Gc = glicemia[["Gc"]]
>>> glicemia_Gc_scaled = min_max_scaler.fit_transform(glicemia_Gc)
```

Fonte: elaborado pelo autor.

4.3 Validação cruzada

Uma maneira de evitar que o modelo implementado se sobreajuste ao conjunto de dados é, de acordo com Géron (2021, p. 59),

> usar o *método k-fold* de validação cruzada da Scikit-Learn. O código [...] divide aleatoriamente o conjunto de treinamento em [...] subconjuntos distintos chamados *folds*, depois treina e avalia o modelo [...], escolhendo sempre um *fold* diferente para avaliação e treinamento dos outros.

O código implementado utilizou a validação cruzada para avaliação dos modelos.

4.4 Treinamento dos modelos

Após a coleta de amostras de acordo com o protocolo apresentado no capítulo 3, o *dataset* (conjunto de dados) possui um conjunto de 32 amostras, das quais 22 são selecionadas para treinamento e 10, para teste.

A Figura 10 apresenta o treinamento para os três modelos selecionados, utilizando a biblioteca sklearn.

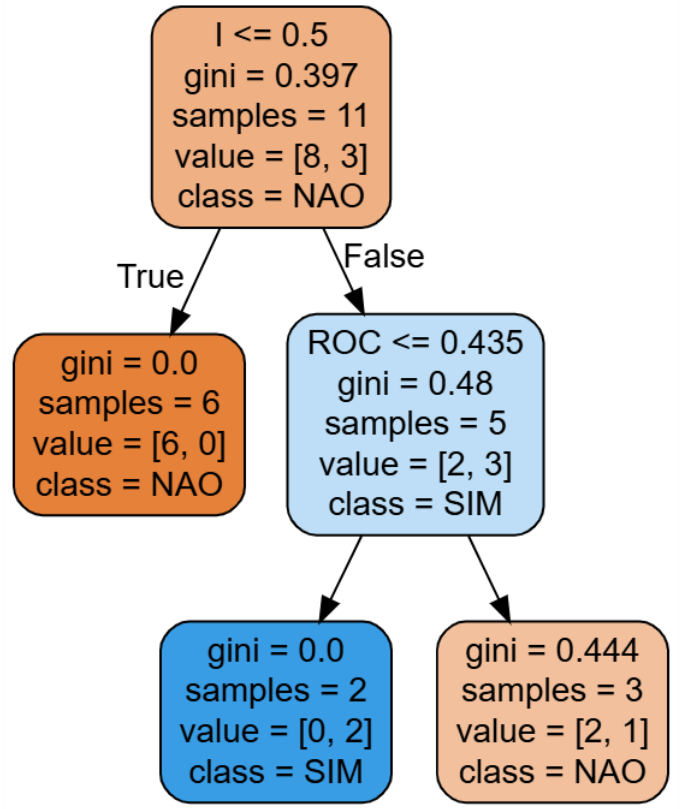
Figura 10 – Treinamento dos modelos

```
>>> from sklearn.neighbors import KNeighborsClassifier
>>> from sklearn.tree import DecisionTreeClassifier
>>> from sklearn.ensemble import RandomForestClassifier
>>>
>>> knn_clf = KNeighborsClassifier()
>>> tree_clf = DecicionTreeClassifier(max_depth=2)
>>> forest_clf = RandomForestClassifier()
>>>
>>> knn_clf.fit(X_train, y_train)
>>> tree_clf.fit(X_train, y_train)
>>> forest_clf.fit(X_train, y_train)
```

Fonte: elaborado pelo autor.

Nota-se que X_train é um vetor que guarda as 22 amostras de treinamento pré-processadas e y_train é um vetor que guarda os rótulos correspondentes. Destaca-se também que, neste exemplo de código, limitou-se a profundidade da árvore de decisão a 2 (max_depth=2). A Figura 11 mostra a árvore de decisão gerada no caso com as amostras normalizadas.

Figura 11 – Árvore de decisão – amostras normalizadas



Fonte: gerado com Graphviz.

5. ANÁLISE DOS RESULTADOS

Este capítulo apresenta a análise dos resultados dos dados coletados e uma discussão, seguida de ensaio, sobre adaptações a serem feitas para conjuntos pequenos de amostras.

5.1 Análise dos resultados dos dados coletados

O conjunto de dados coletados foi de 32 amostras, que estão exibidas no Apêndice B. Essas 32 amostras estão separadas em 22 amostras de treinamento e 10 de teste. A separação no conjunto de testes e treinamento utilizou o método train_test_split da Scikit-Learn.

Devido à baixa quantidade de amostras, os testes de acurácia, precisão e revocação foram realizados com validação cruzada, de acordo com o apresentado na seção 4.3, com valor de k = 2. Esses resultados estão apresentados na Tabela 4 (a linha de acurácia apresenta o resultado de cada um dos conjuntos de validação).

Tabela 4 – Métricas dos modelos no conjunto de treinamento

Métrica	k-vizinhos	Árvore de decisão	Floresta Aleatória
Acurácia	73%, 73%	55%, 73%	55%, 55%
Precisão	60%	43%	0%
Revogação	43%	43%	0%
F_1 score	50%	43%	0%

Fonte: elaborado pelo autor.

As Tabelas 5-7 apresentam as matrizes de confusão para os modelos testados.

Tabela 5 – Matriz de confusão – k-vizinhos mais próximos

	Não hipoglicemia prevista	Hipoglicemia prevista
Não hipoglicemia real	13	2
Hipoglicemia real	4	3

Fonte: elaborado pelo autor.

Tabela 6 – Matriz de confusão – Árvore de decisão

	Não hipoglicemia prevista	Hipoglicemia prevista
Não hipoglicemia real	11	4
Hipoglicemia real	4	3

Fonte: elaborado pelo autor.

Tabela 7 – Matriz de confusão – Floresta aleatória

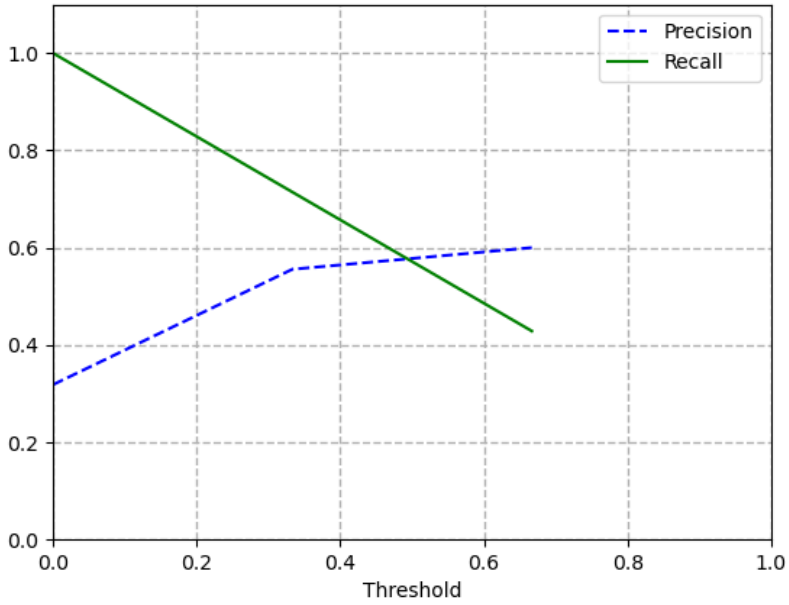
	Não hipoglicemia prevista	Hipoglicemia prevista
Não hipoglicemia real	12	3
Hipoglicemia real	7	0

Fonte: elaborado pelo autor.

Essa primeira análise mostra o modelo dos k-vizinhos mais próximos como o mais promissor dentre os testados para este conjunto de dados de treinamento.

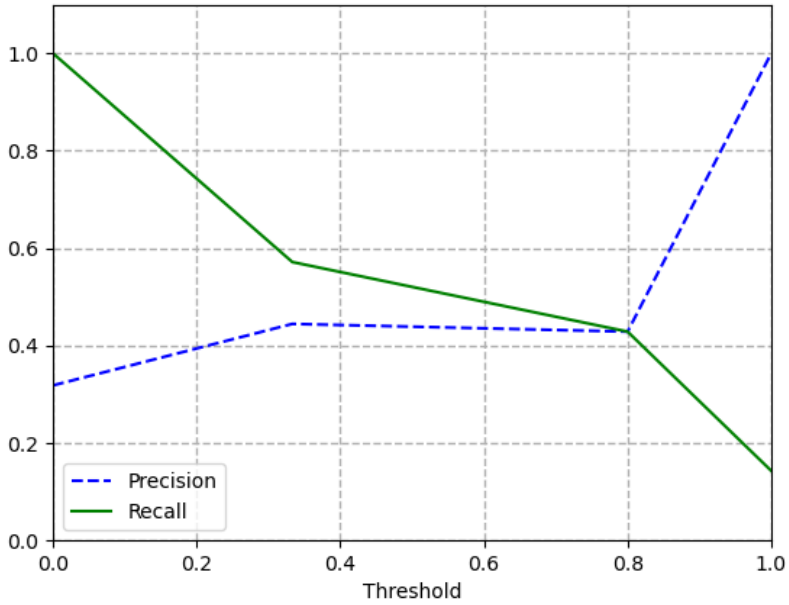
Na seção 2.2, foi apresentada a noção do *trade-off* precisão-revocação, definida por Géron (2021, p.75). Os modelos utilizados (k-vizinhos mais próximos, árvore de decisão e floresta aleatória) atribuem a cada amostra uma probabilidade de pertencer a uma determinada classe e, dessa forma, definem um limiar (*threshold*) a partir do qual passa a considerar que a amostra pertence àquela classe. Pode-se estudar o efeito de um aumento ou diminuição desse limiar na precisão e na revocação. Esse estudo está apresentado para cada modelo nas Figuras 12-14.

Figura 12 – Precisão x revocação – k-vizinhos mais próximos



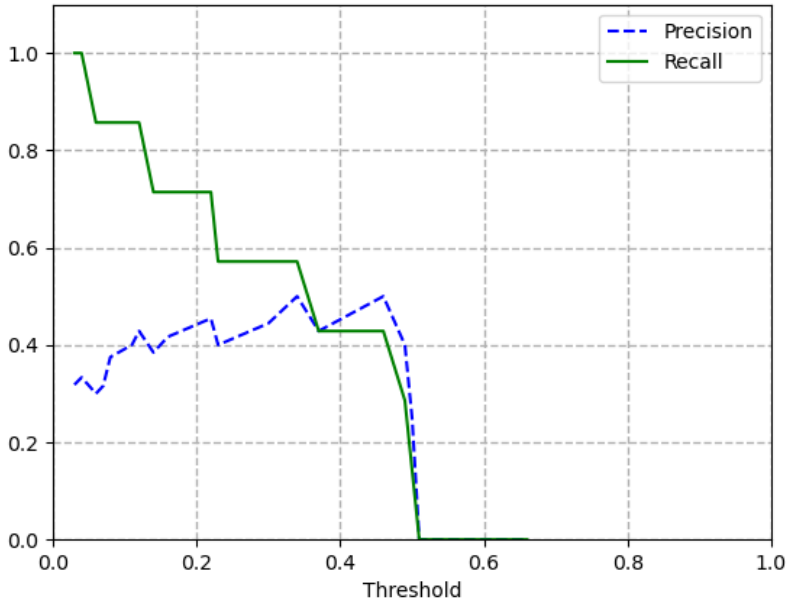
Fonte: elaborado pelo autor.

Figura 13 – Precisão x revocação – Árvore de decisão



Fonte: elaborado pelo autor.

Figura 14 – Precisão x revocação – Floresta aleatória



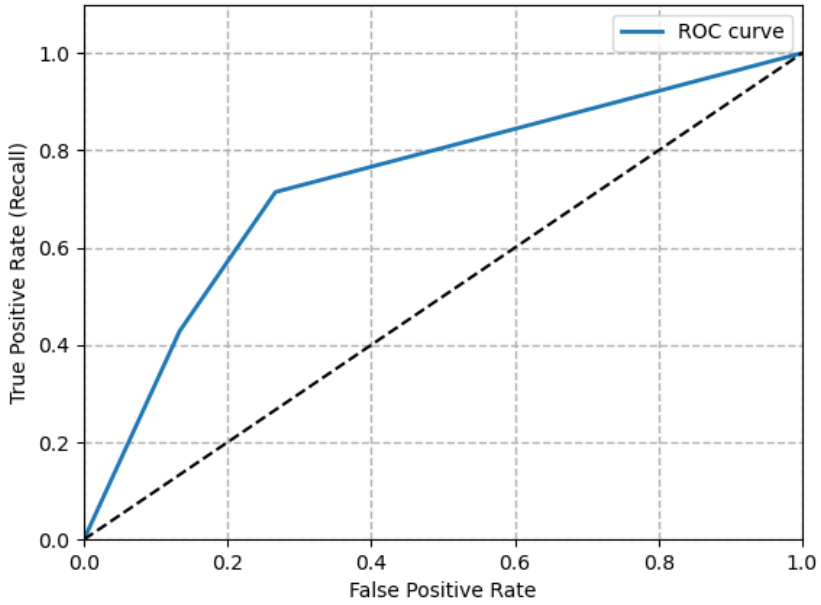
Fonte: elaborado pelo autor.

Géron (2021, p.78-79) apresenta uma outra maneira de analisar o desempenho de classificadores binários: através da *curva de característica de operação* (*receiver operating characteristic – ROC curve*). De acordo com ele,

> [a curva de característica de operação] é muito semelhante à curva de precisão/revocação, mas, em vez de plotar precisão vs. revocação, a curva ROC representa *a taxa de verdadeiros positivos* (outro nome para revocação) em relação à *taxa de positivos falsos* (FPR). O FPR é a proporção de instâncias negativas classificadas incorretamente como positivas. É igual a 1 – a *taxa de verdadeiros negativos* (TNR) [...]. A linha pontilhada representa a curva ROC de um classificador exclusivamente aleatório; um bom classificador fica o mais distante possível dessa linha (em direção ao canto superior esquerdo).

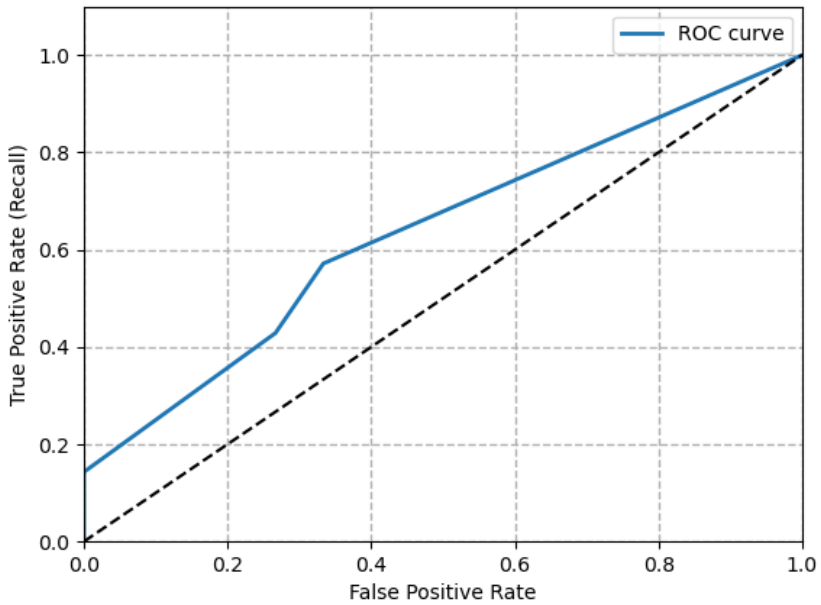
As Figuras 15-17 mostram a curva ROC para cada um dos modelos testados.

Figura 15 – Curva ROC – k-vizinhos mais próximos



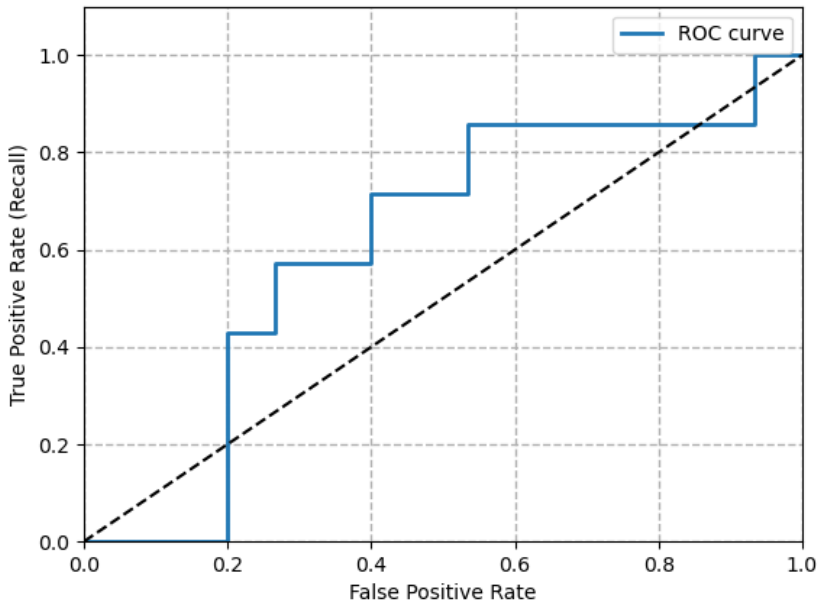
Fonte: elaborado pelo autor.

Figura 16 – Curva ROC – Árvore de decisão



Fonte: elaborado pelo autor.

Figura 17 – Curva ROC – Floresta aleatória



Fonte: elaborado pelo autor.

Também pode-se utilizar a área sob a curva ROC (AUC – *area under the curve*) como uma medida de desempenho do classificador. Enquanto um classificador aleatório possui AUC 0,5, um classificador perfeito possui AUC 1 (GÉRON, 2021). A Tabela 8 mostra o resultado de AUC para cada modelo.

Tabela 8 – AUC dos modelos

Modelo	AUC
k-vizinhos mais próximos	0,73
Árvore de decisão	0,63
Floresta aleatória	0,61

Fonte: elaborado pelo autor.

Nota-se que, também por essa métrica, o modelo dos k-vizinhos mais próximos mostra-se o mais promissor.

Após a análise de desempenho no conjunto de treinamento, faz-se necessário analisar o desempenho dos modelos na generalização para o conjunto de testes. A Tabela 9 mostra o conjunto de rótulos de teste comparando os valores previstos por cada um dos modelos.

Tabela 9 – Rótulos no conjunto de testes vs. previsões dos modelos

Fonte	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
Rótulo real	1	0	0	1	1	0	0	1	1	0
k-vizinhos	0	1	0	1	1	0	1	0	1	1
Árvore de decisão	0	0	0	1	1	0	1	0	1	1
Floresta aleatória	0	0	0	1	0	0	0	0	0	1

Fonte: elaborado pelo autor.

Os resultados da tabela 9 mostram que o modelo dos k-vizinhos mais próximos erra 5 das 10 previsões no conjunto de testes, apresentando, portanto, acurácia de 50% no conjunto de testes. O modelo da árvore de decisão erra 4, com acurácia de 60% no conjunto de testes. Por fim, o modelo da floresta aleatória erra 5, com acurácia de 50%.

Para o conjunto de treinamento apresentado, que possui um número limitado de amostras, pode-se notar que nenhum dos três modelos apresentou um desempenho que satisfaça a hipótese de pesquisa. Mesmo o modelo dos k-vizinhos mais próximos, que, em média, apresentou boa acurácia no conjunto de treinamentos (em média 73%), não generalizou bem no conjunto de testes, com acurácia de apenas 50% nesse conjunto, o que se assemelha a um classificador aleatório.

5.2 Adaptações para conjuntos pequenos de amostras

As restrições de recursos para a elaboração do presente trabalho implicaram um conjunto pequeno de amostras (apenas 32). Contudo, por mais que se possa dispor de mais recursos para a obtenção de um conjunto maior de treinamento e testes, a natureza do problema tende a limitar o tamanho desse conjunto. Isso se deve à dificuldade para obtenção de indivíduos para testes, dificuldade para a reprodução das especificações do protocolo estabelecido no capítulo 3 e ao número limitado de vezes que esses indivíduos se submeteriam a essas condições na realização de exercícios físicos para a obtenção dos dados.

De fato, o problema de se dispor de um conjunto pequeno de amostras tende a acontecer em estudos de aprendizado de máquina aplicados, particularmente, à área da saúde. Existem, no entanto, algumas técnicas utilizadas para lidar com a

limitação de amostras, sendo a técnica de amostras virtuais (*virtual samples*) uma das mais frequentes. Além disso, árvores de decisão e florestas aleatórias estão entre os algoritmos de AM mais utilizados em problemas com conjuntos de dados pequenos, dando mais suporte às escolhas de algoritmos explicadas na seção 3.2 (KOKOL; KOKOL; ZAGORANSKI, 2022).

As amostras virtuais não fazem parte do conjunto de dados reais, mas são derivadas da distribuição dos dados originais ou projetadas a partir de algoritmos de predição (XU *et al.*, 2023).

Para a construção de um conjunto de amostras virtuais, foi desenvolvido um modelo a partir do conhecimento prévio do domínio trabalhado, além de observação do conjunto coletado. As definições desse modelo são as seguintes:

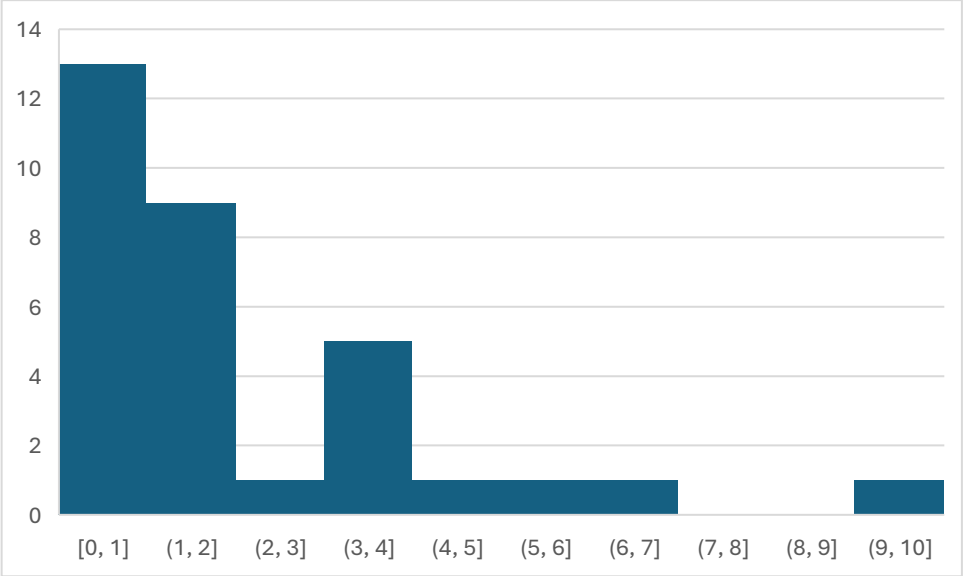
a) observa-se que a média dos valores da glicemia inicial (G_C) no conjunto de amostras reais é de 203 mg/dL, com desvio padrão de 55 mg/dL. No entanto, esses valores estão acima do recomendado por padrões médicos de bom controle, e podem apenas representar um subconjunto de dias em que o paciente estava com glicemia acima do recomendado. Por isso, para a construção das amostras virtuais, optou-se pela geração de números aleatórios com distribuição normal, com média 150 mg/dL e desvio padrão 30 mg/dL. Além disso, os valores foram limitados inferiormente em 90 mg/dL, pois existem recomendações de evitar a realização de exercícios físicos caso a glicemia inicial esteja abaixo desse valor;
b) para a intensidade do exercício (I), optou-se pela geração de metade as amostras em baixa intensidade e metade das amostras em média intensidade;
c) para a insulina bolus ativa no corpo (IOB), ao se montar um histograma com as amostras reais, obtém-se o resultado da Figura 18, onde nota-se que existe uma concentração grande de amostras com IOB próxima a 0, com redução de amostras nas faixas maiores. Acima de 10 unidades, já não se observam amostras. Dessa forma, para a modelagem das amostras virtuais de IOB, optou-se por construir uma distribuição normal, de média zero e desvio padrão 2 unidades, com valores tomados em módulo para evitar a ocorrência de casos negativos;

d) da mesma forma, o histograma para as amostras reais da quantidade de carboidratos ingeridos antes da refeição (CHO) pode ser visto na Figura 19. De maneira similar, também há uma concentração próxima a zero e nenhum valor acima de 60 gramas. Para esse cenário, utilizou-se uma distribuição normal de média zero e desvio padrão 10 gramas, com valores tomados em módulo para evitar a ocorrência de casos negativos;
e) os fatores de contagem de carboidratos (CR) e de correção (CF) dependem do indivíduo e do horário do dia. Como foram realizados ensaios apenas com um indivíduo e em horários parecidos, esses valores foram constantes em toda a base coletada. Os mesmos valores foram mantidos para as amostras virtuais;
f) os valores de taxa de variação de glicemia (ROC) estão distribuídos conforme o histograma apresentado na Figura 20. Nota-se que os valores estão mais concentrados ao redor de zero e a frequência diminui à medida que o valor absoluto de ROC cresce. As amostras virtuais foram modeladas por uma distribuição normal de média zero e desvio padrão 1. Nesse caso, valores negativos são permitidos;
g) para o cálculo da glicemia final resultante da realização do exercício, segue-se uma fórmula apresentada na Equação 4, para casos de exercícios com intensidade baixa, ou na Equação 5, para intensidade média. $\mathbb{N}$ representa um ruído gaussiano de média zero e desvio padrão 10 mg/dL. Uma explicação mais detalhada sobre as premissas adotadas para os termos dessas equações encontra-se no Apêndice A;
h) valores resultantes abaixo de 70 mg/dL são classificados como hipoglicemia, enquanto os outros, como não hipoglicemia.

$$G = \max\left(G_C - 10\,IOB + 3\,CHO + 30\,ROC - 30 + \mathbb{N}, 60\right) \quad (4)$$

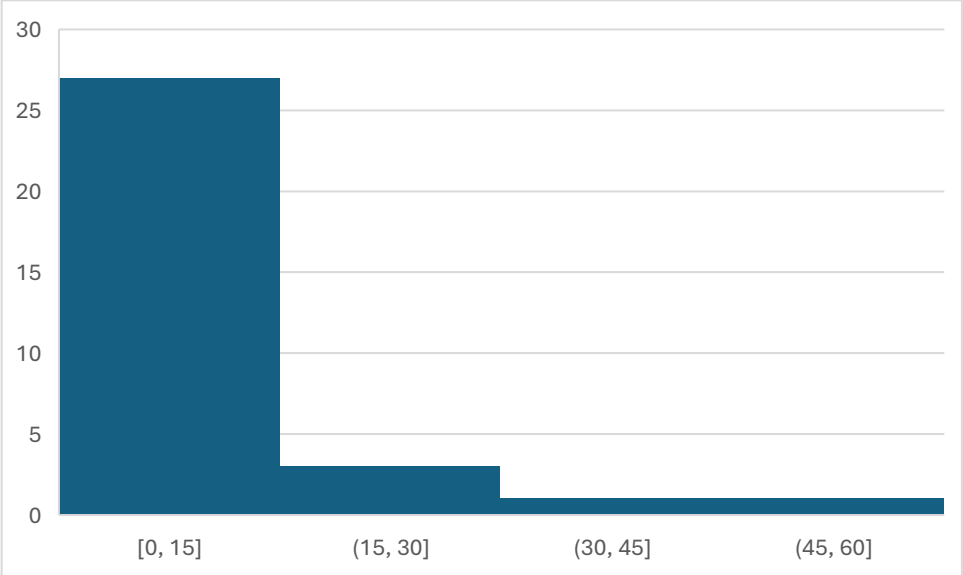
$$G = \max\left(G_C - 20\,IOB + 1{,}5\,CHO + 30\,ROC - 50 + \mathbb{N}, 60\right) \quad (5)$$

Figura 18 – Histograma das amostras reais de IOB



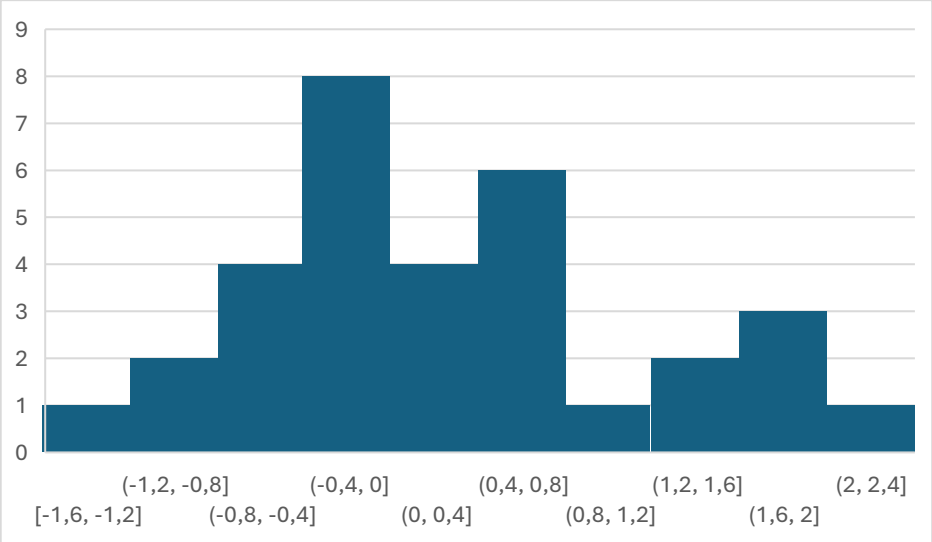
Fonte: elaborado pelo autor.

Figura 19 – Histograma das amostras reais de CHO



Fonte: elaborado pelo autor.

Figura 20 – Histograma das amostras reais de ROC



Fonte: elaborado pelo autor.

Foram então gerados dois conjuntos virtuais de tamanho 50 mil e 1 milhão, respectivamente. Em seguida, foram treinados os três modelos (k-vizinhos mais próximos, árvore de decisão e floresta aleatória). A Tabela 10 resume os resultados de acurácia, precisão, revocação e F_1 score para os dois conjuntos, assim como o conjunto real para o modelo dos k-vizinhos mais próximos. As Tabelas 11 e 12 resumem os mesmos resultados, respectivamente para os modelos árvore de decisão e floresta aleatória.

Tabela 10 – Desempenho dos modelos k-vizinhos com amostras virtuais

Amostras	Acurácia	Precisão	Revocação	F_1 score
N = 32	73%, 73%	60%	43%	50%
N = 50.000	94%, 94%	89%	88%	89%
N = 1.000.000	94%, 94%	90%	89%	89%

Fonte: elaborado pelo autor.

Tabela 11 – Desempenho dos modelos árvore de decisão com amostras virtuais

Amostras	Acurácia	Precisão	Revocação	F_1 score
N = 32	55%, 73%	43%	43%	43%
N = 50.000	79%, 80%	71%	38%	50%
N = 1.000.000	80%, 80%	71%	42%	52%

Fonte: elaborado pelo autor.

Tabela 12 – Desempenho dos modelos floresta aleatória com amostras virtuais

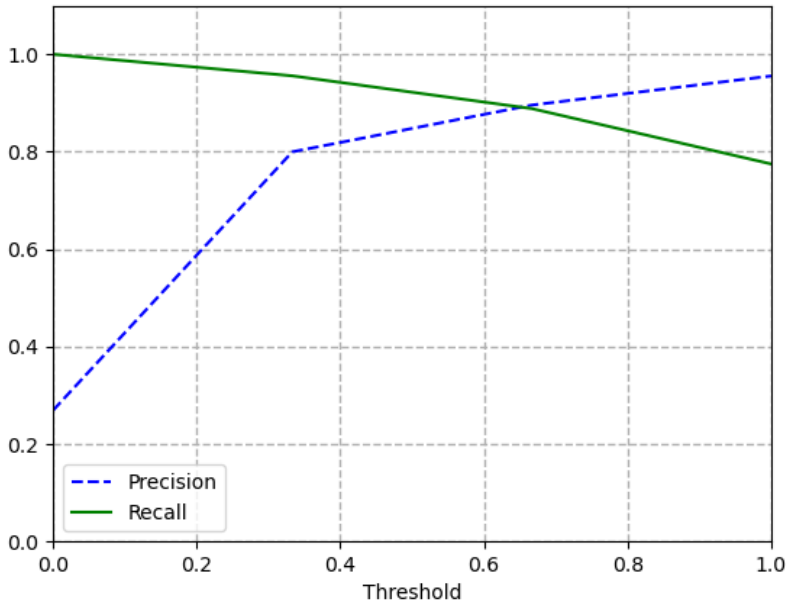
Amostras	Acurácia	Precisão	Revocação	F_1 score
N = 32	55%, 55%	0%	0%	0%
N = 50.000	94%, 94%	90%	88%	89%
N = 1.000.000	94%, 94%	89%	89%	89%

Fonte: elaborado pelo autor.

Das Tabelas 10-12, nota-se que a floresta aleatória apresentou grande ganho de desemprenho com as amostras virtuais. Já a árvore de decisão, que havia sido inicialmente configurada para ter profundidade máxima 2, não apresentou o mesmo nível de melhora. Foi realizado, então, mais um teste com a árvore de decisão, para 1.000.000 de amostras virtuais, aumentando, no entanto, a profundidade máxima para 8. O resultado foi de acurácia para os dois *folds* 92%, 92%, precisão 87% e revocação 84%, resultando em F_1 score de 86%. Esse número se assemelha mais aos outros modelos para N = 1.000.000. Portanto, para as análises posteriores com o modelo de árvore de decisão, foi utilizado este, com profundidade máxima 8.

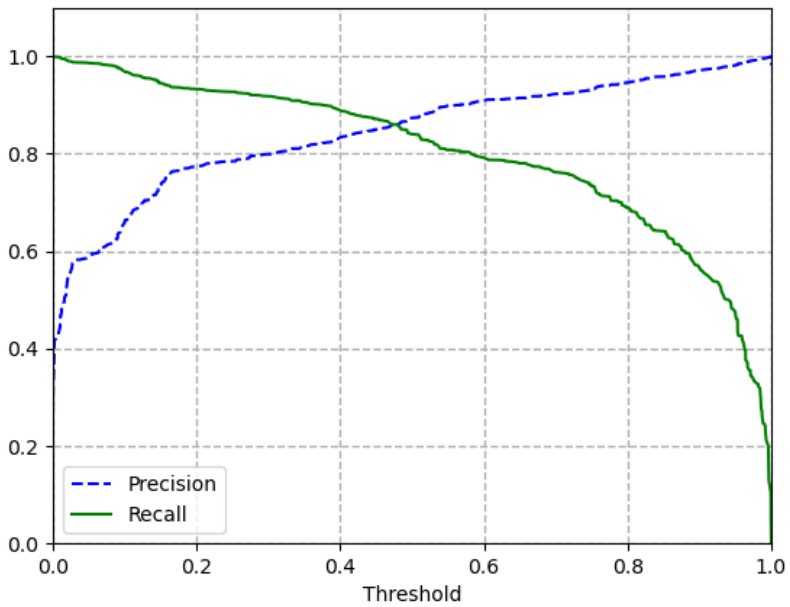
As curvas de precisão *versus* revocação dos modelos k-vizinhos mais próximos, árvore de decisão e floresta aleatória, para N = 1.000.000, estão reproduzidas, respectivamente, nas Figuras 21-23, enquanto as curvas ROC estão reproduzidas nas Figuras 24-26.

Figura 21 – Precisão x revocação – k-vizinhos mais próximos (N = 1.000.000)



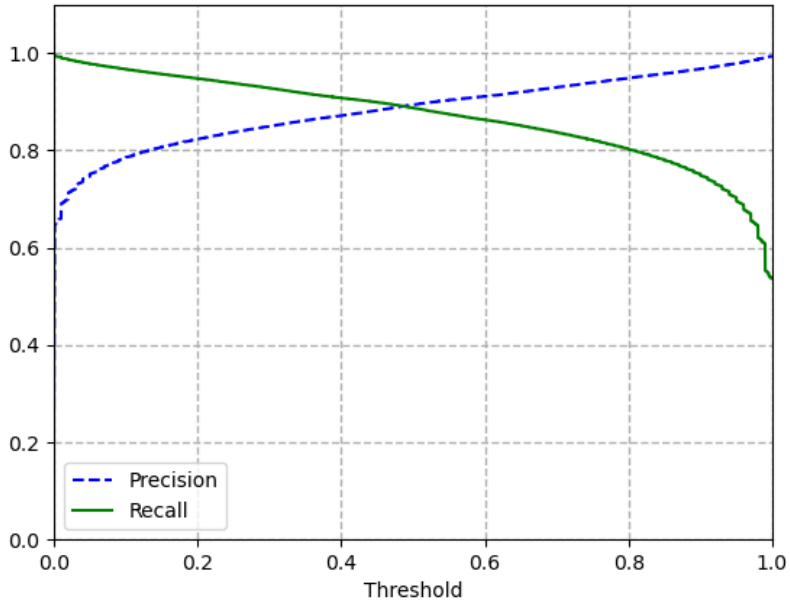
Fonte: elaborado pelo autor.

Figura 22 – Precisão x revocação – Árvore de decisão (N = 1.000.000)



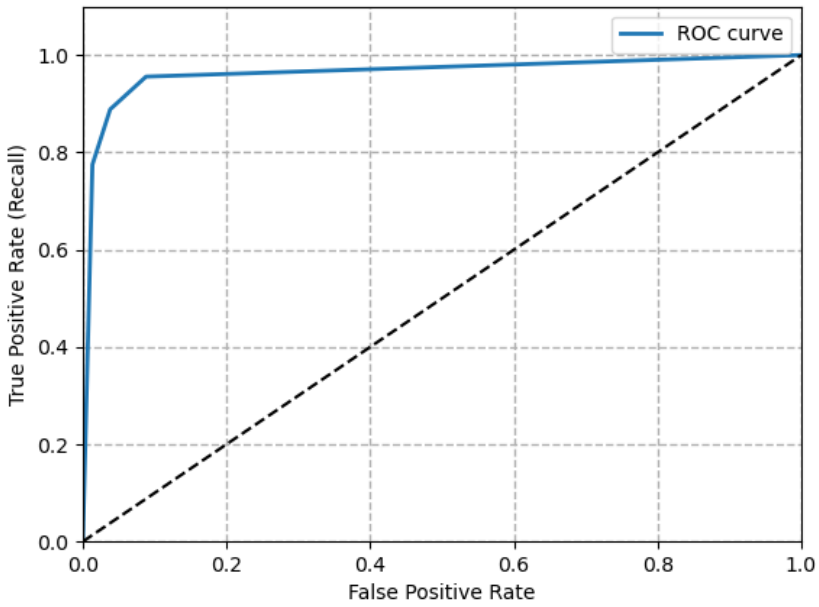
Fonte: elaborado pelo autor.

Figura 23 – Precisão x revocação – Floresta aleatória (N = 1.000.000)



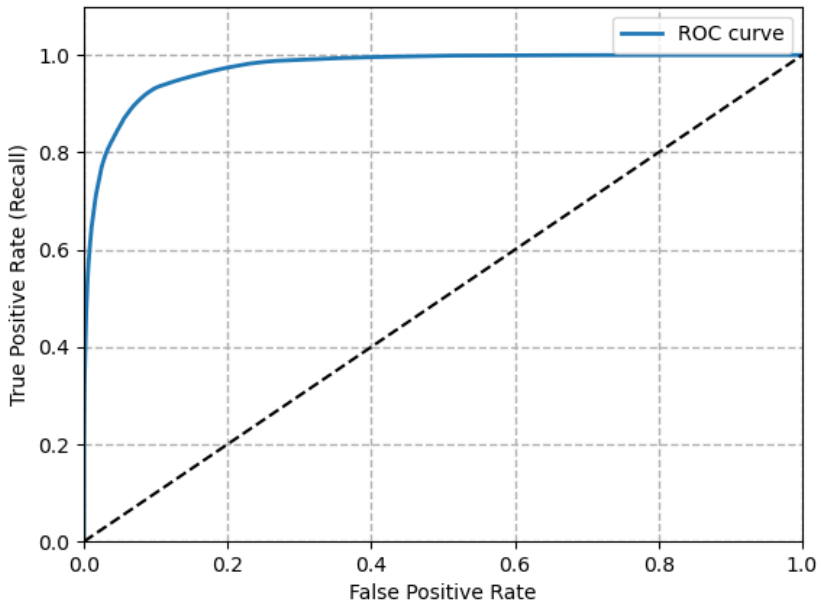
Fonte: elaborado pelo autor.

Figura 24 – Curva ROC – k-vizinhos mais próximos (N = 1.000.000)



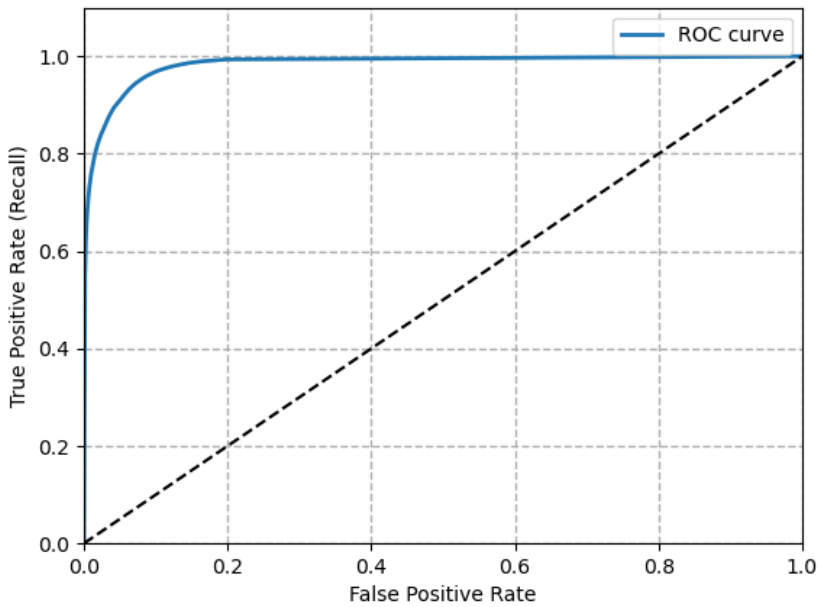
Fonte: elaborado pelo autor.

Figura 25 – Curva ROC – Árvore de decisão (N = 1.000.000)



Fonte: elaborado pelo autor.

Figura 26 – Curva ROC – Floresta aleatória (N = 1.000.000)



Fonte: elaborado pelo autor.

Por fim, foi calculada a área sob a curva ROC (AUC) novamente, agora para N = 1.000.000. Os resultados, comparados o modelo original, com N = 32, estão reproduzidos na Tabela 13. Nota-se que houve uma melhoria dessa métrica nos três modelos analisados, comprovando a visível melhoria nos formatos das curvas ROC (Figuras 24-26).

Tabela 13 – AUC dos modelos com amostras originais vs. virtuais

Modelo	AUC (N=32)	AUC (N=1.000.000)
k-vizinhos mais próximos	0,73	0,96
Árvore de decisão	0,63	0,97
Floresta aleatória	0,61	0,98

Fonte: elaborado pelo autor.

Com o objetivo de entender se o modelo de criação de amostras virtuais melhora o desempenho para as amostras reais, analisou-se a acurácia, precisão, revocação e F_1 score dos três modelos, para N = 1.000.000, apenas nas amostras reais. Os valores encontrados encontram-se na Tabela 14.

Tabela 14 – Análise dos modelos para as amostras reais (N = 1.000.000)

Métrica	k-vizinhos	Árvore de decisão	Floresta aleatória
Acurácia	75%, 56%	56%, 75%	63%, 56%
Precisão	55%	56%	44%
Revocação	50%	75%	33%
F_1 score	52%	64%	38%

Fonte: elaborado pelo autor.

Comparando-se os resultados obtidos na Tabela 14 com aqueles da Tabela 4, nota-se que tanto o modelo da árvore de decisão quanto o modelo da floresta aleatória apresentam desempenho superior com a presença das amostras virtuais. Já o modelo dos k-vizinhos mais próximos não apresentou melhoria de desempenho. Entre eles, o modelo árvore de decisão apresenta, para esse conjunto de amostras, superioridade, tanto na métrica de precisão, quanto na métrica de revocação. Apesar disso, nenhum deles atingiu o objetivo estabelecido na hipótese deste trabalho, de 80% para acurácia e revocação, mantendo a precisão acima de 60%.

Já ao se comparar os resultados da Tabela 14 com aqueles das Tabelas 10-12, nota-se que os modelos estão bem ajustados às amostras virtuais, mas não generalizam tão bem para amostras reais. É possível que ajustes no modelo de criação das amostras virtuais possa melhorar o desempenho da generalização para amostras reais. Isso pode ensejar um trabalho futuro.

6. CONSIDERAÇÕES FINAIS

Este capítulo apresenta as conclusões e contribuições do trabalho, assim como sugestões para trabalhos futuros.

6.1 Conclusões

Com base nas análises apresentadas no Capítulo 5, nota-se que a implementação inicial dos modelos de aprendizado de máquina (AM) para as 32 amostras reais não apresenta um desemprenho que satisfaça a hipótese desta monografia. Devido às restrições de recursos comentadas anteriormente, os modelos foram treinados com uma quantidade limitada de dados (apenas 32 amostras, sendo 22 de treinamento e 10 de testes), o que potencialmente impactou significativamente no desempenho dos modelos. No entanto, como foi ponderado no capítulo 5, por mais que se pudesse dispor de mais recursos para a obtenção de um conjunto maior de treinamento e testes, a natureza do problema tende a limitar o tamanho desse conjunto.

Com base em pesquisas sobre problemas de aprendizado de máquina em conjuntos de dados pequenos, realizou-se uma implementação de um modelo de geração de amostras virtuais, a partir tanto de conhecimentos do domínio estudado, quanto da observação das amostras reais obtidas. O uso das amostras virtuais resultou em modelos que apresentavam desempenho que satisfaziam a hipótese do presente trabalho dentro das próprias amostras virtuais, mas com resultados não tão expressivos na generalização para as amostras reais. Ainda assim, o desempenho dos modelos de árvore de decisão e floresta aleatória apresentou melhoria de desempenho a partir do uso das amostras virtuais.

Os resultados da PoC derivados do pequeno número de amostras sugerem, por um lado, que a metodologia aplicada nesta monografia com o conjunto de dados recolhido não foi suficiente para desenvolver um modelo com o desempenho esperado. Ainda assim, notaram-se melhorias de desempenho com o método de amostras virtuais e é possível que, com a coleta de mais amostras reais e com ajustes no modelo de criação de amostras virtuais, o desempenho esperado seja atingido. Por outro lado, pode ser que a fixação arbitrária de um limiar mínimo de 80% para acurácia e revocação, mantendo a precisão acima de 60%, seja bastante restritiva. Diante

dessas considerações, é necessário, em primeiro lugar, a discussão com especialistas da área médica em questão para uma avaliação desses resultados para, em seguida, aprofundar este estudo em um trabalho futuro.

6.2 Contribuições do Trabalho

Este trabalho propõe uma aplicação de aprendizado de máquina em um problema de saúde. Os modelos propostos tentam prever a ocorrência ou não de hipoglicemia em cenários de realização de exercícios físicos por indivíduos acometidos por diabetes *mellitus* tipo 1.

Este trabalho também desenvolve um método para geração de amostras virtuais para o modelo estudado.

6.3 Trabalhos Futuros

Trabalhos futuros que complementam o tema abordado nesta monografia poderiam:

a) aumentar o conjunto de dados de treinamento e testes, inclusive com a participação de mais indivíduos;
b) aumentar o grau de liberdade na realização de exercícios físicos, por exemplo, fazendo o tempo de duração ser variável e tornando-o característica dos modelos;
c) avaliar o desempenho de outros modelos de aprendizado de máquina;
d) aperfeiçoar o modelo de criação de amostras virtuais;
e) implementar um modelo de aprendizado de máquina com aprendizado incremental, aprendendo com cada nova amostra e aperfeiçoando o modelo;
f) implementar uma aplicação *front-end* para facilitar o uso por indivíduos interessados;
g) integrar a aplicação desenvolvida a monitores de glicemia em tempo real.

REFERÊNCIAS

BASS, L; CLEMENTS, P; KAZMAN, R. **Software Architecture in Practice.** 4. Ed. Addison Wesley, 2021.

CAMPOS, L. Hipoglicemia: como tratar e evitar? **Sociedade Brasileira de Diabetes**, 26 abril 2023. Disponível em: https://diabetes.org.br/hipoglicemia-como-tratar-e-evitar/#:~:text=A%20hipoglicemia%20pode%20ser%20classificada,exijam%20assist%C3%AAncia%20para%20o%20tratamento. Acesso em: 23 novembro 2024.

GÉRON, A. **Mãos à Obra: Aprendizado de Máquina com Scikit-Learn, Keras & TensorFlow:** Conceitos, ferramentas e técnicas para a construção de sistemas inteligentes**.** 2. Ed. Tradução: Cibelle Ravaglia. Rio de Janeiro: Alta Books, 2021. Título original: Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow, ISBN 978-1-492-03264-9.

INTERNATIONAL DIABETES FEDERATION. **IDF Diabetes Atlas: 10th edition.** IDF, 2021. 141p. Disponível em: https://profissional.diabetes.org.br/wp-content/uploads/2022/02/IDF_Atlas_10th_Edition_2021-.pdf. Acesso em: 23 novembro 2024.

KOKOL, P; KOKOL M; ZAGORANSKI, S. Machine Learning on small size samples: A synthetic knowledge synthesis. **Sage Journals: Science Progress.** 2022. Disponível em: https://doi.org/10.1177/00368504211029777. Acesso em: 4 janeiro 2025.

KHANNA, A; FRANCON, O; MIIKKULAINEN, R. Optimizing the Design of an Artificial Pancreas to Improve Diabetes Management. **arXiv.** 2024. Disponível em: https://arxiv.org/pdf/2402.07949. Acesso em: 23 novembro 2024.

KNEUSEU, R. **Como a Inteligência Artificial Funciona: Da Magia à Ciência.** 1. Ed. Tradução: Cibelle Ravaglia. São Paulo: Novatec, 2024. Título original: How AI Works: From Sorcery to Science, ISBN 978-1-718-50372-4.

MOURI, M.; BADIREDDY, M. **Hyperglycemia.** StatPearls, 2023. Disponível em: https://www.ncbi.nlm.nih.gov/books/NBK430900/. Acesso em: 19 janeiro 2025.

NETO, A. J. R.; BORGES, M. M; ROQUE, L. A Preliminary Study of Proof of Concept Practices and treir Connections with Information Systems and Information Science.Em: Proceedings of Sicth International Conference on Technologinal Ecosystems for Enhancing Multiculturality. 2018. Disponível em: https://dl.acm.org/doi/10.1145/3284179.3284226. Acesso em: 23 novembro 2024.

NEVES, C.; NEVES, J. S.; OLIVEIRA, S. C.; OLIVEIRA, A.; CARVALHO, D. Diabetes *Mellitus* Tipo 1. **Revisão Portuguesa de Diabetes.** 2017. Disponível em: https://www.revportdiabetes.com/wp-content/uploads/2018/02/RPD-Vol-12-n%C2%BA-4-Dezembro-2017-Artigo-Revis%C3%A3o-p%C3%A1g-159-167.pdf.pdf. Acesso em: 19 janeiro 2025.

PETERSEN, M.; SHULMAN, G. Mechanisms of insulin action and insulin resistance. **American Physiological Society.** 2018. Disponível em: https://doi.org/10.1152/physrev.00063.2017. Acesso em: 19 janeiro 2025.

SOCIEDADE BRASILEIRA DE DIABETES. Departamento de Saíde Pública. **Dados Epidemiológicos do diabetes *mellitus* no Brasil.** SBD, 2023. 40p. Disponível em: https://profissional.diabetes.org.br/wp-content/uploads/2023/06/Dados-Epidemiologicos-SBD_comT1Dindex.pdf. Acesso em: 23 novembro 2024.

TAULLI, T. **Introdução à Inteligência Artificial: Uma abordagem não técnica.**1. ed. Tradução: Luciana do Amaral Teixeira. São Paulo: Novatec, 2020. Título original: Artificial Intelligence Basics; A Non-Technical Introduction.

VETTORETTI, M; CAPPON, G; FACCHINETTI, A; SPARACINO, G. Advanced Diabetes Management Using Artificial Intelligence and Continuous Glucose Monitoring Sensors. **Sensors.** 2020. Disponível em: https://www.mdpi.com/1424-8220/20/14/3870. Acesso em: 20 novembro 2024.

XU, P; JI, X; LI, M; LU, W. Virtual Sample Generation in machine learning assisted materials design and 59andom59e59. **Journal of Materials Informatics.** 2023. Disponível em: https://www.oaepublish.com/articles/jmi.2023.18. Acesso em: 13 janeiro 2025.

APÊNDICE A – Modelo de geração de rótulos para amostras virtuais

Como discutido na seção 5.2, as Equações 6 e 7 modelam a glicemia do paciente após a realização de exercícios físicos de baixa e média intensidade, respectivamente. Nessas equações, $\mathbb{N}$ representa um ruído gaussiano de média zero e desvio padrão 10 mg/dL.

$$G = \max\left(G_C - 10\,IOB + 3\,CHO + 30\,ROC - 30 + \mathbb{N}, 60\right) \quad (6)$$

$$G = \max\left(G_C - 20\,IOB + 1{,}5\,CHO + 30\,ROC - 50 + \mathbb{N}, 60\right) \quad (7)$$

A explicação para os coeficientes modelados nessas equações são:

a) a glicemia após a realização do exercício é a glicemia inicial subtraída de um valor, logo, ambas as equações partem de G_C;
b) a presença de insulina ativa no corpo (IOB) contribui para a redução da glicemia, como discutido no Capítulo 2. Logo, os coeficientes de IOB são ambos negativos;
c) além disso, assume-se como premissa que a sensibilidade à IOB é maior em casos de exercícios mais intensos, dada a maior redução da glicemia em casos de maior intensidade de exercícios. Logo, o coeficiente de IOB é maior, em módulo, na Equação 7;
d) ainda sobre o coeficiente de IOB, o indivíduo diabético em questão apresenta um fator de correção CF (de acordo com o apresentado no Capítulo 2) com valor 15. Isso significa que, a cada 15 mg/dL que a glicemia se encontra acima do alvo, recomenda-se a aplicação de uma unidade de insulina para que a glicemia retorne ao alvo em um tempo de 3 horas, em geral. Dessa maneira, considerando variações em torno desse valor, o modelo considerou que cada unidade de insulina ativa no corpo reduz, no período de 40 minutos da realização do exercício, 10 mg/dL no caso de exercício de baixa intensidade e 20 mg/dL no caso de exercícios de média intensidade;
e) o consumo de carboidratos (CHO) contribui para o aumento da glicemia, logo, os coeficientes de CHO são positivos em ambas as equações;

f) além disso, assume-se que a sensibilidade a CHO seja reduzida em casos de exercícios de maior intensidade, por isso o coeficiente é menor na Equação 7;
g) ainda sobre os coeficientes de CHO, o paciente apresenta fator de contagem de carboidratos CR com valor 7. De acordo com o apresentado no Capítulo 2, isso significa que a cada 7 gramas de carboidrato ingerido, uma unidade de insulina é requerida para a correção das 7 gramas. Assumindo que cada unidade reduz 15 mg/dL (como discutido anteriormente, dado o valor de CF), toma-se como premissa que 7 gramas de carboidratos elevam em 15 mg/dL a glicemia. Dessa maneira, cada grama de carboidrato eleva em aproximadamente 2 mg/dL a glicemia. Analogamente ao modelo para IOB, foram utilizadas variações em torno desse valor. Tomou-se, no modelo, o valor de 3 para o caso de baixa intensidade e 1,5 para o caso de média intensidade;
h) para os coeficientes de ROC, como essa grandeza representa a taxa de variação da glicemia por minuto no início da realização do exercício, caso essa taxa se mantivesse durante os 40 minutos da realização do exercício, o coeficiente proposto deveria ser 40. Considerou-se, no entanto, que a ROC deveria ter certa atenuação ao longo do tempo e propôs-se o valor de 30, para ambos os casos;
i) ambos os casos possuem uma constante negativa, representando a queda de glicemia causada exclusivamente pela realização do exercício físico, na ausência dos outros fatores considerados. Assume-se como premissa que a realização de exercício de maior intensidade causa maior redução. Por inspeção dos valores reais obtidos, foram considerados os valores de 30, para baixa intensidade e 50 para média intensidade;
j) a adição do ruído $\mathbb{N}$, com média zero e desvio padrão 10 mg/dL, acrescenta uma imprecisão ao modelo, com intuito de modelar variáveis não consideradas no modelo, dadas as complexidades do sistema biológico que se pretende modelar;
k) por fim, os valores foram limitados inferiormente em 60 mg/dL. Essa restrição foi imposta para que o modelo não apresentasse resultados inverídicos para a glicemia, como valores negativos.

APÊNDICE B – Conjunto completo de amostras coletadas

A Tabela 15 apresenta o conjunto completo das amostras reais coletadas.

Tabela 15 – Conjunto completo de dados coletados

Amostra	G_C	I	IOB	CHO	CR	CF	ROC	Hipo
1	175	Baixa	0,7	0	7	15	0,5	Não
2	138	Média	0	15	7	15	0,5	Não
3	102	Média	0	15	7	15	0	Sim
4	193	Média	2	30	7	15	0,7	Sim
5	267	Baixa	4	0	7	15	0,1	Não
6	299	Baixa	4	0	7	15	1	Não
7	140	Baixa	0	0	7	15	-0,1	Não
8	216	Média	4	26	7	15	0,6	Não
9	177	Baixa	1	0	7	15	0	Não
10	240	Baixa	6	0	7	15	1,7	Não
11	280	Média	3	0	7	15	0	Sim
12	283	Média	2	0	7	15	0,7	Sim
13	227	Baixa	5	50	7	15	0,7	Sim
14	156	Média	0	0	7	15	2	Não
15	159	Média	0,7	0	7	15	0	Sim
16	156	Baixa	1,3	10	7	15	0	Não
17	246	Média	6,5	20	7	15	-1,1	Não
18	220	Baixa	0	0	7	15	2,1	Não
19	153	Média	0	0	7	15	-0,1	Sim
20	180	Baixa	1,7	0	7	15	1,7	Não
21	185	Baixa	2	0	7	15	1,5	Sim
22	143	Baixa	0,6	0	7	15	-0,4	Não
23	193	Baixa	0	0	7	15	0,2	Não
24	167	Baixa	3,3	0	7	15	-0,5	Não
25	250	Média	0	0	7	15	0,3	Não
26	303	Média	2	0	7	15	-0,4	Não
27	138	Média	0	0	7	15	-1,6	Sim
28	257	Baixa	10	40	7	15	-1	Sim
29	249	Média	1,3	0	7	15	-0,3	Sim
30	212	Baixa	2	0	7	15	0,2	Não
31	124	Baixa	1,5	11	7	15	-0,4	Sim
32	265	Baixa	4	0	7	15	1,4	Não

Fonte: elaborado pelo autor.

APÊNDICE C – Código do protótipo implementado

O trecho de código na Figura 27 mostra o código do pré-processador de dados, implementado com suas funções.

A função preprocess_dataset lê um arquivo csv que contém os dados reais coletados e separa os dados em um conjunto de treinamento de 22 amostras e um conjunto de testes de 10 amostras, com train_test_split. Em seguida, utiliza o OrdinalEncoder para o pré-processamento das características categóricas e o MinMaxScaler para a normalização das outras características, de acordo com o que foi definido na seção 4.2. A coluna Hipo, que contém o rótulo das amostras foi transformada manualmente em zeros, representando não hipoglicemia, e uns, representando hipoglicemia.

Já a função preprocess_virtual_dataset realiza atividade similar para um arquivo csv que contém as amostras virtuais geradas. Nesse caso, 10% das amostras virtuais são selecionadas para teste com o train_test_split.

Figura 27 – Pré-processador de dados

```
from sklearn.compose import ColumnTransformer
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OrdinalEncoder, MinMaxScaler
import numpy as np
import pandas as pd


def preprocess_dataset():
    glicemia = pd.read_csv("./datasets/glicemia/glicemia.csv", sep=";", thousands=".", decimal=",")
    train_set, test_set = train_test_split(glicemia, test_size=10, 63andom_state=10)

    glicemia_num = glicemia.select_dtypes(include=[np.number]).drop("G", axis=1).drop("Hipo", axis=1)
    num_attribs = list(glicemia_num)
    cat_attribs = ["I"]

    num_pipeline = Pipeline([
                ('scaler', MinMaxScaler())
            ])

    full_pipeline = ColumnTransformer([
            ("num", num_pipeline, num_attribs),
            ("cat", OrdinalEncoder(), cat_attribs),
        ])
```

```
    X_train = full_pipeline.fit_transform(train_set)
    y_train = train_set[“Hipo”].copy()

    X_test = full_pipeline.transform(test_set)
    y_test = test_set[“Hipo”].copy()

    return X_train, y_train, X_test, y_test, glicemia


def preprocess_virtual_dataset():
    glicemia = pd.read_csv(“./datasets/glicemia/virtual_dataset.csv”, sep=”;”, thousands=”.”,
decimal=”,”)
    train_set, test_set = train_test_split(glicemia, test_size=0.1, 64andom_state=10)

    glicemia_num = glicemia.select_dtypes(include=[np.number]).drop(“G”, axis=1).drop(“Hipo”, axis=1)
    num_attribs = list(glicemia_num)
    cat_attribs = [“I”]

    num_pipeline = Pipeline([
                ('scaler', MinMaxScaler())
            ])

    full_pipeline = ColumnTransformer([
            (“num”, num_pipeline, num_attribs),
            (“cat”, OrdinalEncoder(), cat_attribs),
        ])

    X_train = full_pipeline.fit_transform(train_set)
    y_train = train_set[“Hipo”].copy()

    X_test = full_pipeline.transform(test_set)
    y_test = test_set[“Hipo”].copy()

    return X_train, y_train, X_test, y_test, glicemia
```

Fonte: elaborado pelo autor.

Já o trecho de código apresentado na Figura 28 apresenta o código do gerador de amostras virtuais, seguindo as especificações apresentadas na seção 5.2, complementadas pelo Apêndice A. Por padrão, o método generate_dataset gera 50.000 amostras virtuais. Neste trabalho, foi utilizado tanto com esse valor padrão, quanto para N = 1.000.000. O conjunto de dados gerados é armazenado em um arquivo csv.

Figura 28 – Gerador de amostras virtuais

```
import numpy as np
import pandas as pd


def generate_dataset(file_name='virtual_dataset.csv', N=50000):
    Gc, I, IOB, CHO, CF, CR, ROC, g, G = generate_virtual_samples(N)

    df = pd.DataFrame({
        'Gc': Gc,
        'I': I,
        'IOB': IOB,
        'CHO': CHO,
        'CR': CR,
        'CF': CF,
        'ROC': ROC,
        'G': g,
        'Hipo': G
    })

    df.to_csv(file_name, index=False, sep=";", decimal=",")


def generate_virtual_samples(N=50000):
    Gc = generate_Gc(N)
    I = generate_I(N)
    IOB = generate_IOB(N)
    CHO = generate_CHO(N)
    CF = generate_CF(N)
    CR = generate_CR(N)
    ROC = generate_ROC(N)

    g = np.zeros_like(I)

    g[I==0] = Gc[I==0] - 10*IOB[I==0] + 3*CHO[I==0] + 30*ROC[I==0] - 30
    g[I==1] = Gc[I==1] - 20*IOB[I==1] + 1.5*CHO[I==1] + 30*ROC[I==1] - 50

    n_g = (10 * np.random.randn(N)).round(0)

    g = g + n_g

    g[g<60] = 60

    G = np.zeros_like(I)
    G[g<70] = 1

    return Gc, I, IOB, CHO, CF, CR, ROC, g, G
```

```
def generate_Gc(N):
    Gc = (150 + 30*np.random.randn(N)).round(0)

    Gc[Gc<90] = 90

    return Gc

def generate_I(N):
    I = np.random.choice([0,1], N)
    return I

def generate_IOB(N):
    IOB = np.abs(2 * np.random.randn(N)).round(1)
    return IOB

def generate_CHO(N):
    CHO = np.abs((10 * np.random.randn(N).round(0)))
    return CHO

def generate_CF(N):
    CF = np.full(N, 15)
    return CF

def generate_CR(N):
    CR = np.full(N, 7)
    return CR

def generate_ROC(N):
    ROC = np.random.randn(N).round(1)
    return ROC
```

Fonte: elaborado pelo autor

Por fim, a Figura 29 mostra como foram treinados e testados os modelos k-vizinhos mais próximos, árvore de decisão e floresta aleatória.

Figura 29 – Treinamento e teste dos modelos

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score, cross_val_predict
from sklearn.metrics import confusion_matrix, precision_score, recall_score, f1_score,
precision_recall_curve, roc_curve, roc_auc_score
import matplotlib.pyplot as plt
from MachineLearning.preprocessing import preprocess_dataset

X_train, y_train, X_test, y_test, glicemia = preprocess_dataset()
```

```
knn_clf = KneighborsClassifier(n_neighbors=3)
knn_clf.fit(X_train, y_train)

tree_clf = DecisionTreeClassifier(max_depth=8)
tree_clf.fit(X_train, y_train)

forest_clf = RandomForestClassifier(67andom_state=10)
forest_clf.fit(X_train, y_train)

def model_accuracy(model, cv=2):
    if (model == 'knn'):
        classifier = knn_clf
    elif (model == 'tree'):
        classifier = tree_clf
    elif (model == 'forest'):
        classifier = forest_clf
    else:
        error('No such model')

    return cross_val_score(classifier, X_train, y_train, scoring='accuracy', cv=cv)

def model_confusion_matrix(model, cv=2):
    if (model == 'knn'):
        classifier = knn_clf
    elif (model == 'tree'):
        classifier = tree_clf
    elif (model == 'forest'):
        classifier = forest_clf
    else:
        error('No such model')

    y_real_pred = cross_val_predict(classifier, X_real, y_real, cv=cv)

    return confusion_matrix(y_real, y_real_pred)

def model_precision_recall_f1(model, cv=2):
    if (model == 'knn'):
        classifier = knn_clf
    elif (model == 'tree'):
        classifier = tree_clf
    elif (model == 'forest'):
        classifier = forest_clf
    else:
        error('No such model')

    y_train_pred = cross_val_predict(classifier, X_train, y_train, cv=cv)

    precision = precision_score(y_real, y_real_pred)
```

```
    recall = recall_score(y_real, y_real_pred)
    f1 = f1_score(y_real, y_real_pred)

    return precision, recall, f1

def plot_precision_recall_curve_1(model, cv=2):
    if (model == 'knn'):
        classifier = knn_clf
    elif (model == 'tree'):
        classifier = tree_clf
    elif (model == 'forest'):
        classifier = forest_clf
    else:
        error('No such model')

    y_proba = cross_val_predict(classifier, X_train, y_train, cv=cv, method="predict_proba")
    y_scores = y_proba[:,1]
    precisions, recalls, thresholds = precision_recall_curve(y_train, y_scores)

    plt.plot(thresholds, precisions[:-1], "b—", label="Precision")
    plt.plot(thresholds, recalls[:-1], "g-", label="Recall")
    plt.xlabel('Threshold')
    plt.grid(linestyle='—', linewidth=1)
    plt.xlim(0, 1)
    plt.ylim(0, 1.1)
    plt.legend()
    plt.show()

def plot_precision_recall_curve_2(model, cv=2):
    if (model == 'knn'):
        classifier = knn_clf
    elif (model == 'tree'):
        classifier = tree_clf
    elif (model == 'forest'):
        classifier = forest_clf
    else:
        error('No such model')

    y_proba = cross_val_predict(classifier, X_train, y_train, cv=cv, method="predict_proba")
    y_scores = y_proba[:,1]
    precisions, recalls, thresholds = precision_recall_curve(y_train, y_scores)

    plt.plot(recalls, precisions, "g-")
    plt.xlabel('Recall')
    plt.ylabel('Precision')
    plt.grid(linestyle='—', linewidth=1)
    plt.xlim(0, 1)
    plt.ylim(0, 1.1)
    plt.show()
```

```
def plot_roc_curve(model, cv=2):
    if (model == ‘knn’):
        classifier = knn_clf
    elif (model == ‘tree’):
        classifier = tree_clf
    elif (model == ‘forest’):
        classifier = forest_clf
    else:
        error(‘No such model’)

    y_proba = cross_val_predict(classifier, X_train, y_train, cv=cv, method=”predict_proba”)
    y_scores = y_proba[:,1]
    fpr, tpr, thresholds = roc_curve(y_train, y_scores)

    plt.plot(fpr, tpr, linewidth=2, label=”ROC curve”)
    plt.plot([0,1],[0,1], ‘k–')
    plt.xlabel(‘False Positive Rate’)
    plt.ylabel(‘True Positive Rate (Recall)’)
    plt.grid(linestyle=’–', linewidth=1)
    plt.xlim(0, 1)
    plt.ylim(0, 1.1)
    plt.legend()
    plt.show()

def model_roc_score(model, cv=2):
    if (model == ‘knn’):
        classifier = knn_clf
    elif (model == ‘tree’):
        classifier = tree_clf
    elif (model == ‘forest’):
        classifier = forest_clf
    else:
        error(‘No such model’)

    y_proba = cross_val_predict(classifier, X_train, y_train, cv=cv, method=”predict_proba”)
    y_scores = y_proba[:,1]
    return roc_auc_score(y_train, y_scores)
```

Fonte: elaborado pelo autor.