PAULO HENRIQUE DE SOUZA PEREIRA PRAZERES

PREDIÇÃO DE READMISSÃO HOSPITALAR DE PACIENTES COM DIABETES MELLITUS

Monografia apresentada ao Programa de Educação Continuada da Escola Politécnica da Universidade de São Paulo, para obtenção do título de Especialista, pelo Programa de Pós-Graduação em Big Data - Inteligência na Gestão dos Dados.

SÃO PAULO

2023

PAULO HENRIQUE DE SOUZA PEREIRA PRAZERES

PREDIÇÃO DE READMISSÃO HOSPITALAR DE PACIENTES COM DIABETES MELLITUS

Monografia apresentada ao Programa de Educação Continuada da Escola Politécnica da Universidade de São Paulo, para obtenção do título de Especialista, pelo Programa de Pós-Graduação em Engenharia de Dados e Big Data.

Área de concentração: Tecnologia da Informação – Engenharia/ Tecnologia/ Gestão

Orientador: Luiz Sergio de Souza

SÃO PAULO

2023

FICHA CATALOGRÁFICA

Prazeres, Paulo

PREDIÇÃO DE READMISSÃO HOSPITALAR DE PACIENTES COM DIABETES MELLITUS / P. Prazeres -- São Paulo, 2023. 105 p.

Monografia (Especialização em Engenharia de Dados & Big Data) - Escola Politécnica da Universidade de São Paulo. PECE – Programa de Educação Continuada em Engenharia.

1.Big Data 2.Mineração de dados 3.Machine learning I.Universidade de São Paulo. Escola Politécnica. PECE – Programa de Educação Continuada em Engenharia II.t

AGRADECIMENTOS

Agradeço ao professor Dr. Luiz Sergio pela paciência e por toda a ajuda durante o curso.

Agradeço a coordenadora Dra. Solange Nilce por me apoiar a concluir o curso, mesmo em tempos difíceis.

Agradeço também aos meus colegas de classe Tulio, Erik Figueiredo e William Barbosa por todo o trabalho em equipe, aprendizado e troca de experiências durante 2 anos incríveis estudando juntos.

CURSO ENGENHARIA DE BIG DATA

Coord.: Prof. Solange N. Alves de Souza
Vice-Coord.: Prof. Anarosa Alves Franco Brandão

Perspectivas profissionais alcançadas com o curso:

O curso trouxe muitos benefícios na minha carreira profissional como analista de dados, desde o entendimento do funcionamento básico de um Big Data até o tratamento adequado de dados e sugestões de melhorias de processos nos setores da saúde e bancário.

RESUMO

A hospitalização é o maior peso nos gastos hospitalares, e muitas internações têm raízes em condições crônicas. O número de pacientes diabéticos hospitalizados, em geral é considerável, e antecipar a possibilidade de readmissão hospitalar é de vital importância para todos os atores envolvidos em uma hospitalização: hospitais públicos e privados, seguradoras de saúde, o SUS (Sistema Único de Saúde) e a parte interessada mais importante neste contexto, o próprio paciente.

O foco principal desta pesquisa é comparar os resultados de diferentes modelos de aprendizado de máquina, na função de realizar a predição de readmissão hospitalar, aplicando técnicas e conceitos de ciência de dados que, por sua vez, podem ser adaptados para bases de dados provenientes de hospitais e seguradas brasileiras.

A base de dados utilizada é proveniente de prontuários médicos eletrônicos desidentificados, com uma década de histórico que contempla o período de 1999 a 2008, originados de atendimentos clínicos em 130 hospitais nos EUA, cortesia esta, do Centro de Pesquisa Clínica e Translacional da Virginia Commonwealth. Para tal, faz-se uso de cinco algoritmos computacionais - Regressão Logística, Máquina de Vetores de Suporte, Floresta Aleatória, Árvore de decisão e Redes Neurais Artificiais -, confrontando os resultados e os avaliando-os através de indicadores de performance utilizados em estudos correlatos, tais como acurácia, sensibilidade, especificidade e a curva ROC. Dentre os algoritmos estudados, observou-se que o modelo de Máquina de Vetores de Suporte resultou na melhor classificação de readmissão hospitalar, conquistando uma acurácia de 61%.

Os resultados indicam que a aplicação de modelos de aprendizado pode desempenhar um papel significativo em colaborar na predição de readmissão hospitalar de pacientes diagnosticados com diabetes, figurando como uma ferramenta valiosa para contribuir para o aprimoramento de sistemas de apoio à decisão clínica, otimizando o processo de diagnóstico e tratamento da diabetes.

Palavras-chave: Readmissão hospitalar de pacientes diabéticos, Diabetes mellitus

ABSTRACT

Hospitalization represents the major cost factor in healthcare expenditures, and numerous hospital admissions stem from chronic conditions. The number of hospitalized diabetic patients is generally substantial, and anticipating the possibility of hospital readmission is of vital importance for all stakeholders involved in a hospitalization: public and private hospitals, health insurers, the Unified Health System (SUS), and the most important stakeholder in this context, the patient themselves.

The focus of this research is to compare the outcomes of different machine learning models in predicting hospital readmission, applying data science techniques and concepts that can be adapted to databases from Brazilian hospitals and insurance companies.

Leveraging data from de-identified electronic medical records spanning a decade from 1999 to 2008, originating from clinical encounters in 130 hospitals in the USA, courtesy of the Virginia Commonwealth University's Center for Clinical and Translational Research. This study employs five computational algorithms - Logistic Regression, Support Vector Machine, Random Forest, Decision Tree, and Artificial Neural Networks - contrasting and evaluating the outcomes using performance indicators typically utilized in related studies, such as accuracy, sensitivity, specificity, and the ROC curve (Y. ZHAO et al., 2019).

Among the studied algorithms, it was observed that the Support Vector Machine model resulted in the best classification of hospital readmission, achieving an accuracy of 61%.

The findings suggest that employing machine learning models can significantly contribute to predicting hospital readmission in patients diagnosed with diabetes, serving as a valuable tool to enhance clinical decision support systems and optimize the diagnosis and treatment processes for diabetes.

Keywords: Hospital readmission of diabetic patients, medical readmission of diabetic patients, Diabetes mellitus

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

SUMARIO

1. INTRODUÇÃO

O aumento da prevalência da diabetes ao longo dos anos é notável. Em comparação com 108 milhões de adultos que viviam com a doença em 1980 em todo o mundo, esse número subiu para cerca de 422 milhões em 2014 (ORGANIZAÇÃO MUNDIAL DA SAÚDE, 2016). Em 2021 esse número subiu para 537 milhões e segundo a Federação Internacional de Diabetes, as projeções apontam para um crescimento ainda mais acentuado nos próximos anos, alcançando cerca de 643 milhões em 2030 e 783 milhões em 2045. Neste cenário preocupante, o Brasil ocupa o sexto lugar do ranking de países com o maior número de habitantes convivendo com a doença, com aproximadamente 16 milhões de pessoas com diabetes. Além disso, as projeções indicam um aumento ainda maior para o Brasil em 2045, com a estimativa de que esse número alcance a marca de 23 milhões (INTERNATIONAL DIABETES FEDERATION, 2021).

Devido ao alto custo de internação hospitalar, é de suma importância para as seguradoras de saúde, hospitais e clínicas, identificar o risco de readmissão hospitalar da sua carteira de participantes, principalmente tratando-se de participantes com diabetes, que por sua vez, possuem um alto índice de readmissões hospitalares. Esta diligência não apenas impacta diretamente nos custos, pois a hospitalização é o item em que as instituições mais gastam (ANS,2022), conforme pode ser observado na tabela 1 a seguir, mas fundamentalmente aprimora a qualidade do atendimento prestado, fomentando uma abordagem mais abrangente e eficaz para os pacientes.

Ao realizar essas previsões, os envolvidos podem utilizar tais informações para tomar medidas estratégicas, aprimorando a gestão de recursos ou até mesmo implementando programas de estímulo à saúde do paciente para prevenir essas readmissões. Essa abordagem proativa não apenas reduz os custos, mas também fortalece a saúde global do indivíduo, refletindo diretamente na eficácia do sistema de saúde.

Tabela 1 - Despesas por item em 2022

Item	**Despesa (R$)**
Internações	95.136.242.684,83

Exames	43.953.429.133,38
Consultas Médicas	29.835.353.523,23
Outros Atendimentos Ambulatoriais	22.356.961.643,64
Consultas Médicas Ambulatoriais	21.886.696.574,18
Terapias	18.403.585.953,42
Demais despesas médico-hospitalares	14.293.244.799,63
Consultas Médicas em Pronto Socorro	7.902.427.955,48
Procedimentos Odontológicos	3.329.990.109,38
Procedimentos preventivos	541.425.616,69
Consultas odontológicas iniciais	195.964.053,92
Próteses odontológicas unitárias (coroa total e restauração metálica fundida)	154.965.963,02
Próteses odontológicas	154.329.427,53
Exodontias simples de permanentes (12 anos ou mais)	47.067.670,26

Fonte: Adaptado do Mapa Assistencial da Saúde Suplementar (AGÊNCIA NACIONAL DE SAÚDE, 2022)

A tarefa de predição de readmissão hospitalar para pacientes que vivem com diabetes é uma em que hospitais e seguradoras de saúde trabalham constantemente, processando dados coletados no dia a dia do serviço de saúde. Como forma de auxiliar e automatizar esta classificação, o aprendizado computacional tem sido amplamente analisado em estudos acadêmicos (ASLAM et al., 2021; Y. ZHAO et al., 2019; DUGGAL, et al., 2016), utilizando algoritmos juntamente com a mineração de dados para gerar modelos de previsão, com intuito de permitir ações preventivas direcionadas aos pacientes diabéticos que possuem maior risco de readmissão.

Nesta pesquisa, utiliza-se dados provenientes de registros médicos com histórico de 10 anos, contemplando o período de 1999 até 2008, de atendimentos clínicos em 130 hospitais dos EUA, fornecidos pelo repositório de aprendizado de máquina da Universidade da Califórnia, Irvine (UCI) e doados pelo Centro de Pesquisa Clínica e Translacional da Universidade da Comunidade da Virgínia.

O objetivo é comparar resultados de diferentes modelos de classificação e analisar seus desempenhos. Ao identificar quais modelos de classificação apresentam melhores resultados, pretende-se contribuir para o aprimoramento de sistemas de apoio à decisão clínica, otimizando o processo de diagnóstico e tratamento da diabetes. No âmbito acadêmico, a pesquisa contribui para o avanço da aplicação de

técnicas de aprendizado de máquina na área da saúde, fornecendo uma base para estudos futuros.

Também se espera que os resultados possam servir como referência para estudos comparativos em saúde, especialmente aqueles que buscam avaliar a eficácia de modelos de classificação em contextos clínicos.

É importante ressaltar que esta pesquisa utiliza dados de hospitais e pacientes dos Estados Unidos, o que significa que os dados e consequentemente resultados, não refletem diretamente a realidade do contexto brasileiro. Não obstante, os conceitos e técnicas empregadas neste trabalho, podem ser adaptadas para bases de dados de hospitais brasileiros. Para este, faz-se uso de 5 algoritmos: Regressão Logística, Árvore de decisão, Floresta aleatória, Redes Neurais Artificiais e Máquinas de Vetores de Suporte. Comparando os resultados e os avaliando através de indicadores de performance constantemente utilizados em estudos semelhantes, como acurácia, sensibilidade, especificidade e a curva ROC (ASLAM et al., 2021).

1.1 Objetivo

Esta pesquisa tem como objetivo analisar e comparar os resultados de cinco algoritmos de classificação – Regressão Logística, Árvore de decisão, Floresta aleatória, Classificador de Redes Neurais Artificiais e Máquina de Vetores de Suporte – na predição de readmissão hospitalar de pacientes com diabetes. A escolha desses algoritmos baseou-se em sua eficácia comprovada em pesquisas semelhantes. Uma introdução das pesquisas pode ser observada no capitulo 2.

1.2 Justificativa

A Federação Internacional de Diabetes, menciona que, com o passar dos anos, a proporção de pessoas diabéticas tem aumentado e um dos principais impulsionadores do custo da diabetes são as internações hospitalares. Além disto, dados da Agência Nacional de Saúde (ANS) revelam que no Brasil, o custo de internações médicas são os maiores dentre todos os gastos da produção assistencial do setor.

Estudos ressaltam que existe uma redução significativa nas readmissões de pacientes que mantém consultas médicas após a alta hospitalar (GREGOR et al, 2022). Neste sentido, a utilização de modelos computacionais para predição de readmissões hospitalares, torna-se fundamental para promover a melhoria da gestão de recursos e apoiar análises clínicas, possibilitando hospitais e seguradoras de saúde e até mesmo os próprios pacientes, agirem de forma proativa, seja criando programas de incentivo a saúde ou acompanhamentos remotos e presenciais com os pacientes.

1.3 Contribuição

Do ponto de vista prático, os resultados podem impulsionar melhorias nos modelos de classificação utilizados na prática clínica, refinando o processo de diagnóstico e tratamento. No contexto acadêmico, a pesquisa contribui para o avanço da aplicação de técnicas de aprendizado de máquina na área da saúde, validando métodos de análise de dados e servindo como referência para estudos comparativos em saúde. Essas contribuições coletivas destacam a relevância e o potencial impacto dessa investigação no campo da saúde e ciência de dados.

1.4 Metodologia

A pesquisa foi realizada nas seguintes etapas principais: revisão da literatura, visando coletar pesquisas que apresentem métodos técnicas, processos, experiências, fracassos e sucessos de diferentes algoritmos de aprendizado computacional frequentemente aplicados à classificação de risco de readmissão hospitalar, seguido pelo levantamento e análise dos dados utilizados na prova de conceito do estudo e por fim, a apresentação e comparação dos resultados obtidos por diferentes técnicas de aprendizado computacional.

2. LITERATURA RELACIONADA

Nos últimos anos, diversos algoritmos têm sido utilizados para predição de readmissão de pacientes com diabetes mellitus. Essas iniciativas práticas têm gerado uma significativa produção acadêmica, demonstrando a viabilidade dessas aplicações. Neste tópico, analisa-se alguns destes esforços para que, identificados os algoritmos e métodos usualmente utilizados em classificações semelhantes, seja possível justificar a escolha dos métodos durante a execução deste estudo e analisar os seus resultados.

Em um estudo de 2013 desenvolvido por TRACK et al., buscou-se analisar o impacto das medições de hemoglobina glicada em pacientes diagnosticados com diabetes mellitus, mantendo-se a hipótese que a medição deste indicador estaria associada com uma redução nas taxas de readmissão de indivíduos internados. Também conhecido como HbA1c, este indicador se forma da interação entre a glicose e a hemoglobina, sendo muito utilizado para avaliar o controle do açúcar no sangue de diabéticos (LITTLE; SACKS, 2009). O referido estudo, se embasando em 70.000 encontros com pacientes portadores de diabetes mellitus, fez uso da regressão logística multivariável para “encaixar a relação entre a medição de HbA1c e a readmissão precoce”, em conjunto com variáveis presentes no banco de dados - como o tipo da doença e dados demográficos. Com esta aplicação, os autores concluíram que o simples fato de ser medida a hemoglobina glicada está associado a menores taxas de readmissões.

Em 2019, Y. ZHAO et al., produziram o estudo intitulado “Predicting 30-Day Hospital Readmissions for Patients with Diabetes” (Previsão de readmissões hospitalares de 30 dias para pacientes com diabetes, em tradução nossa), buscando identificar pacientes com diabetes mellitus que tem maior probabilidade de serem readmitidos em até 30 dias, foi explorado a partir de dados reais de 124.678 registros de admissão extraídos do banco de dados de internação do estado da Flórida, o desempenho de quatro algoritmos computacionais normalmente utilizados em classificações – Matriz de Vetores de Suporte; Redes Neurais; Regressão Logística e Floresta Aleatória – e também os resultados de dois métodos de aprendizado por conjunto, sendo eles Ensemble Learning e XGBoost. Além disso, os autores

analisaram os principais fatores de risco associados às readmissões hospitalares de 30 dias de pacientes diabéticos. Os resultados produzidos apresentaram que informações clínicas e demográficas dos pacientes são importantes para que as técnicas de aprendizado computacional apresentem resultados eficazes na previsão de readmissões hospitalares precoces, todos os algoritmos resultaram em previsões satisfatórias com a curva AUC acima de 0,75, mas a técnica que gerou o melhor resultado foi o XGBoost com a curva AUC de 0,79.

TAMIN e ISWARI, em um estudo realizado em 2017 que visou implementar o algoritmo C4.5 para determinar a taxa de readmissão hospitalar de pacientes com diabetes, utilizou-se da mesma base de dados empregada nesta pesquisa. Essa base contém informações de pacientes diabéticos de 130 hospitais nos Estados Unidos, abrangendo um período de 10 anos (1999-2008). No experimento realizado, foram criadas quatro amostras de dados, cada uma com um tipo de tratamento e conjuntos de variáveis, a fim de avaliar o desempenho do algoritmo para previsão da readmissão, em diferentes cenários. Concluiu-se que o experimento precisa ser reanalisado, visto que o algoritmo da árvore de decisão é muito vulnerável a ocorrência de condição de sobre ajuste. Embora a precisão produzida de 76% seja alta, o artigo ressalta que se os dados de teste forem diferentes dos dados de treinamento, o resultado do modelo pode variar significativamente.

A pesquisa conduzida por SHANG et al em 2021, buscou comparar métodos de aprendizado de máquina parar realizar a predição de readmissão hospitalar em até 30 dias após a alta do paciente diagnosticado com diabetes. Este estudo foi realizado utilizando a mesma base de dados proposta em nossa pesquisa e na pesquisa de TAMIN e ISWARI, mencionado acima. Seus resultados mostraram que o método de Árvore Aleatória obteve o melhor resultado dentre os modelos estudados, com esta base de dados para este objetivo, além disso, os autores destacam que a precisão pode ser melhorada ainda mais se os dados clínicos relacionados as hospitalizações dos pacientes puderem ser estendidas para amostras maiores e com mais atributos.

Com embasamento nas experiências acadêmicas prévias mencionadas, percebe-se que diversas técnicas, métodos e algoritmos são explorados para prever a readmissão hospitalar de pacientes diagnosticados com diabetes. Passa-se, então, à análise e ao

detalhamento das técnicas e métodos que foram escolhidos para serem utilizadas na construção de modelos preditivos proposta neste trabalho.

3. FUNDAMENTAÇÃO TEÓRICA E CONCEITUAL

Neste capítulo, são contextualizados os conceitos teóricos, apresentando um breve resumo sobre a diabetes mellitus e as readmissões médicas hospitalares. Além disso, explora-se o uso de modelos de aprendizado computacional para a predição de readmissões médicas de pacientes diagnosticados com diabetes.

3.1 Diabetes Mellitus no Brasil e no mundo

O ministério da saúde descreve o diabetes como uma doença causada pela produção insuficiente ou má absorção de insulina, hormônio este, que tem a função de quebrar as moléculas de glicose para transformar em energia, para manutenção das células do organismo. O diabetes pode causar o aumento da glicemia e quando apresentam taxas altas, podem levar a complicações no coração, artérias, olhos, rins e nos nervos, podendo levar até mesmo à morte (MINISTÉRIO DA SAÚDE, 2023).

A Sociedade Brasileira de Diabetes destaca que o Diabetes Mellitus pode se apresentar de diversas formas e tipos diferentes e existem 2 tipos principais, são eles:

- **Diabetes Mellitus Tipo 1**
  Condição em que o sistema imunológico erroneamente ataca as células do pâncreas, afetando o nível ideal de geração de insulina, levando a um acúmulo de glicose no sangue, em vez de ser usada como fonte de energia. Esse tipo afeta de 5 a 10% das pessoas com diabetes e geralmente se desenvolve na infância ou adolescência, não obstante, também pode ocorrer em adultos. Seu tratamento envolve o uso de insulina, medicamentos e planejamento alimentar e de atividades físicas, com o objetivo de controlar os níveis de glicose no sangue.

- **Diabetes Mellitus Tipo 2**
  Esse é o tipo mais comum, afetando cerca de 90% das pessoas com diabetes. Neste tipo, a insulina é produzida corretamente, mas o corpo encontra-se em uma condição que não consegue utilizar essa insulina adequadamente, ou até

> mesmo, não produz insulina em quantidades suficientes para regular os níveis de glicose no sangue. Em casos menos graves, seu tratamento é frequentemente realizado com atividade física e planejamento alimentar, enquanto em situações mais severas, pode ser necessário o uso de insulina e/ou outros medicamentos para regular os níveis de glicose.

Infelizmente, até o momento, a causa do tipo de diabetes ainda é desconhecida e ainda não há cura, a melhor forma de preveni-la é com práticas de vida saudável, mantendo boa alimentação e atividades físicas regularmente.

3.2 Readmissões hospitalares

A readmissão hospitalar é o retorno de um paciente em um determinado período ao hospital após receber alta, geralmente com períodos de até 30 dias, até 60 dias ou até 90 dias. A readmissão é um tópico importante na área da saúde, pois tem implicações significativas na qualidade do atendimento, nos custos de saúde e no bem-estar dos pacientes.

Dado a importância de acompanhar os cuidados da saúde da população, alguns países adotaram medidas para analisar e comparar os cuidados da saúde de diferentes hospitais, através de indicadores de qualidade, um desses indicadores é justamente a readmissão hospitalar. Uma das medidas adotadas pelo governo dos Estados Unidos é o Programa de Redução de Readmissões Hospitalares, em tradução nossa (HRRP), esta medida, tem como objetivo incentivar hospitais a reduzir as taxas de readmissão hospitalar, naturalmente, isso é feito por meio de um sistema de pagamento baseado em desempenho (CENTERS FOR MEDICARE & MEDICAID SERVICES, 2023).

O Brasil também adotou formas de avaliar a qualidade dos prestadores de serviço na saúde suplementar, através de indicadores que têm validade, comparabilidade e capacidade de discriminação dos resultados. Conforme a Resolução Normativa nº 546 da ANS, foi criado o programa chamado Programa de Qualificação dos Prestadores de Serviços de Saúde (QUALISS) e se trata de uma iniciativa da Agência Nacional de

Saúde Suplementar (ANS) e seu objetivo é incentivar a melhoria da qualidade dos serviços prestados pelos hospitais brasileiros.

3.3 Aprendizado de máquina

O aprendizado de máquina é a uma abordagem que envolve o uso de algoritmos e modelos de computador para realizar tarefas aprendendo com dados e experiencias anteriores (ALPAYDIN, 2020). Em outras palavras, os computadores são treinados por especialistas de forma que aprendam padrões através de dados históricos e então realizam tarefas de acordo com critérios específicos. Os modelos podem ser tanto preditivos para fazerem predições do futuro quanto descritivos a fim de obter conhecimento de dados ou ambos.

Aprendizado de máquina trabalha com predições analíticas dentro de conjuntos de dados pré-existentes, contribuindo assim, com uma assistência de análise muito mais eficiente e rápida que um ser humano pode realizar (MUELLER e MASSARON, 2022). Essa capacidade de predição é viabilizada por meio de modelos de aprendizado de máquina, que fazem uso de algoritmos e modelos estatísticos para processar dados e reconhecer padrões, gerando previsões em novos conjuntos de dados. (NVIDIA, 2021). Alguns modelos de aprendizado computacional incluem a Regressão Logística, Naive Bayes, Máquinas de Vetores de Suporte e as Árvores de Decisão.

O aprendizado de máquina pode ser dividido em duas categorias: Aprendizagem supervisionada e não supervisionada, que são detalhadas a seguir e para mais informações, é possível consultar o livro “Introdução a mineração de dados” de (CASTRO; FERRARI, 2017).

- **Aprendizagem supervisionada:** Os rótulos das classes dos dados de treinamento são conhecidos previamente e então, usados para ajustar o modelo de predição. Tradicionalmente utilizado em tarefas de classificação de objetos, atribuição de crédito e detecção de fraudes.
  Em outras palavras, os dados são como uma bússola apontando para o norte, pois eles já possuem os rótulos associados.

- **Aprendizagem não supervisionada:** O algoritmo é treinado em um conjunto de dados que não contém rótulos ou saídas previamente conhecidas. O objetivo deste aprendizado, é descobrir padrões, estruturas ou agrupamento nos dados. Muito utilizado em tarefas de recomendação de produtos e classificação de imagens.

  Em termos mais simples, os dados de treinamento são como uma floresta obscura sem trilhas pré-determinadas e sem uma bússola para apontar uma direção. Neste caso, a máquina precisa achar o caminho correto, com a sua própria perspicácia, descobrindo padrões e estruturas da floresta.

Nesta era tecnológica e científica, essas duas abordagens desempenham papéis de destaque na ciência de dados e tem aplicações em muitos campos, de análise de dados de redes sociais à predição de readmissões hospitalares.

3.4 Árvore de decisão

A árvore de decisão é um dos métodos mais populares entre os de classificação. São produzidas por algoritmos que discernem formas de segmentar conjuntos de dados em segmentos semelhantes, resultando em uma estrutura de ramificação. Suas ramificações formam uma árvore invertida que se origina com um nó raiz no topo da árvore, que por sua vez, é conectado a outros nós através de novos ramos, culminando nas folhas da árvore, que correspondem a regras de classificação.

Uma vantagem dos métodos de árvore de decisão é a sua capacidade de analisar múltiplas variáveis. Isso possibilita explicar, descrever, prever e classificar resultados, indo além dos relacionamentos simplificados de causa e efeito. Essa facilidade de compreensão e representação gráfica permite uma explicação intuitiva do funcionamento do algoritmo ao seguir a árvore a partir do nó raiz, para explicar seus desfechos.

Ademais, além da interpretação simplificada, a comunidade encontra atrativos na facilidade de implementação, pois, ao contrário de alguns algoritmos, as árvores de

decisão aceitar variáveis categóricas. Outro fator atrativo é que as árvores de decisão têm a habilidade de lidar com valores ausentes. (HASTIE; TIBSHIRANI; FRIEDMAN, 2008).

A imagem 1 a seguir, exemplifica um modelo de árvore de decisão binária, no qual a árvore começa com uma informação raiz, e a partir dela geram novos testes, ou “nós”. As ramificações, são equivalentes aos resultados dos testes, podendo ou não gerar novas ramificações. Por fim, na extremidade inferior da árvore, temos as folhas, que encerram as subcategorias criadas pela árvore, representando as decisões a serem tomadas.

Figura 1 - Exemplo de Árvore de Decisão

Fonte: Adaptado de Decision Tree and Ensemble Learning Based on Ant Colony Optimization (KOZAK, 2019, p. 4)

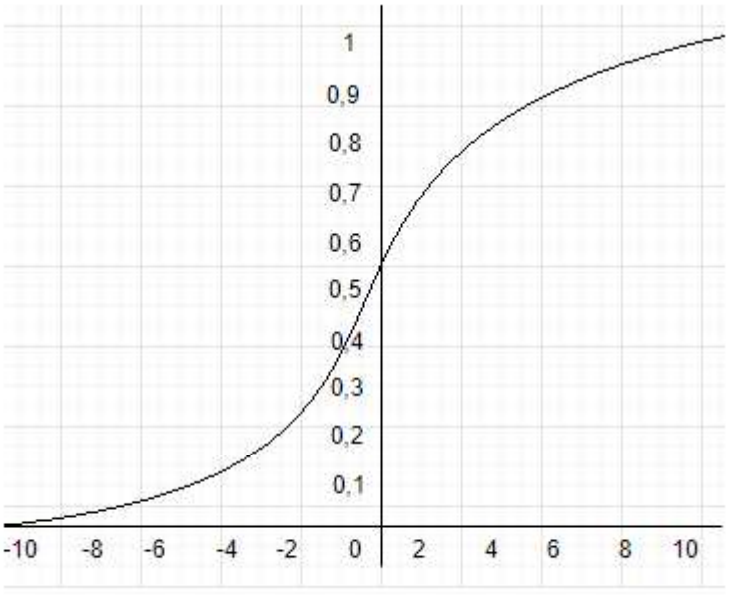
3.5 Regressão logística

Os objetivos da regressão logística são resumidos, no âmbito das categorias possíveis, em classificar observações ao estimar a probabilidade de ela estar inserida em uma categoria particular, modelando, estimando e prevendo a probabilidade deste encaixe. Quando as variáveis categóricas possuírem apenas dois estados possíveis,

têm-se a regressão logística binária, ao passo que, se o número de valores possíveis for maior, é caracterizada como regressão logística multiclasse. Neste trabalho, o foco recai sobre a binária.

Diferente da regressão linear, que tenta ajustar uma linha reta aos dados, a regressão logística funciona calculando uma soma ponderada das características de entrada, semelhante a um modelo de regressão linear, mas em vez de gerar um resultado contínuo, a regressão logística aplica uma função logística (sigmoide) em forma de S, para modelar a probabilidade de uma instancia pertencer a uma classe em problemas de classificação binária (GÉRON, 2019), (GEETHA; SENDHILKUMAR, 2023).

Figura 2 - Exemplo de função logística



Fonte: Adaptado de Mãos à Obra: Aprendizado de Máquina com Scikit-Learn & TensorFlow (GÉRON, 2019)

A equação da regressão logística é dada da seguinte forma (GARETH et al., 2023):

$$p(X) = \frac{e\beta_0 + \beta_1 X}{1 + e\beta_0 + \beta_1 X}$$

Onde:

- $\rho(X)$ - Trata-se da probabilidade de pertencer à categoria 1

- $e -$ - É a base do logaritmo natural
- $e\beta_0 + \beta_1 X$ - São os coeficientes associados a cada variável preditora - $x_1, x_{2\ldots} x_n$

3.6 Redes Neurais Artificiais

Redes neurais artificiais são modelos computacionais criados, usando como base, a estrutura e funcionamento do sistema nervoso humano, essas redes são compostas por unidades interconectadas chamadas de neurônios artificiais.

Redes neurais artificiais são um conceito muito antigo, com estudos datados de 1943, ambos descrevem o modelo como podendo ter várias camadas ocultas e um número diferente de neurônios em cada camada.

Em geral, um modelo de redes neurais é constituído por três tipos de camadas (IZBICKI, R.; SANTOS, T. M., 2020), são elas:

- Camada de Entrada: Primeira camada do processo, no qual os dados são recebidos e passados adiante.
- Camadas Ocultas: Camadas intermediárias que contém neurônios que realizam operações matemáticas para aprender com os dados.
- Camada de saída: Resultado, que pode ser probabilidades de classificação, regressão ou outro tipo.

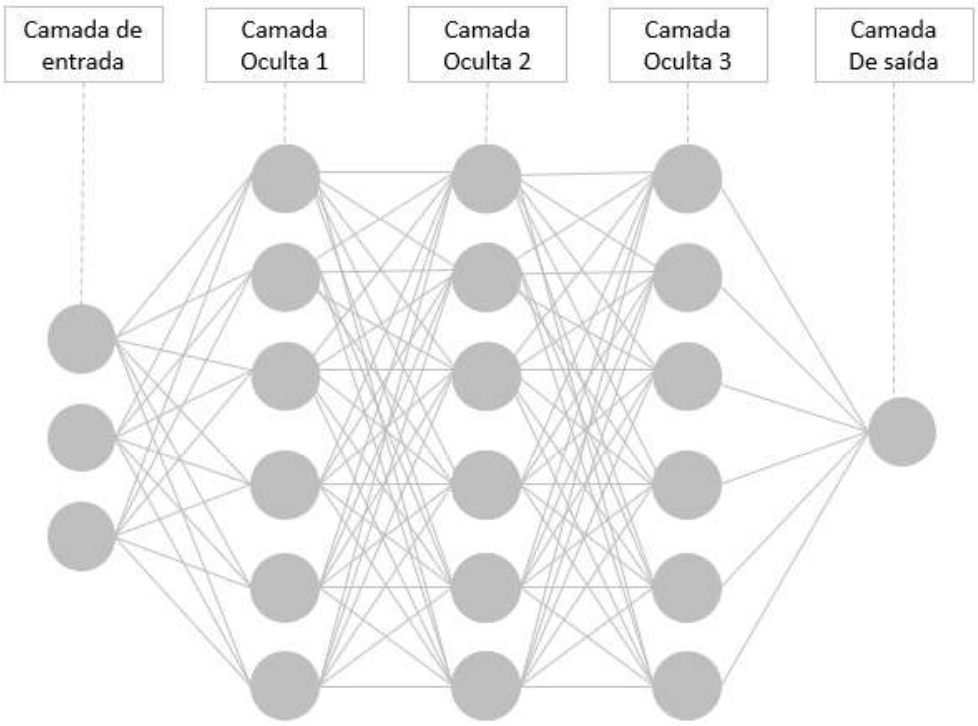
Cada neurônio artificial recebe um ou mais sinais de entrada, realiza uma combinação linear dos sinais ponderados por peso e em seguida aplica funções de ativação (como a sigmoide e tangente hiperbólica) para produzir um sinal de saída.

Os tipos mais amplamente reconhecidos de redes neurais incluem a Rede Neural Feedforward, também conhecida como Multilayer Perceptron, que se trata de uma rede em que as informações fluem em uma única direção, da camada de entrada, para as camadas ocultas e finalmente, para a camada de saída. Essa arquitetura é particularmente útil para resolver problemas de classificação. Além da Rede Neural Feedforward, outro tipo é a Rede Neural Recorrente, que se caracteriza pelas conexões retroativas, que permitem transferência de informações entre camadas do

mesmo tipo ou anteriores. Essa abordagem é muito usada para resolver tarefas que envolvem sequência de dados, como processamento de linguagem natural.

O foco deste estudo recai sobre a Rede Neural Artificial Feedforward, representada graficamente pela figura 3 a seguir:

Figura 3 - Exemplo de uma rede neural Feedforward com 3 camadas ocultas



Fonte: Adaptado de Aprendizado de máquina: uma abordagem estatística (IZBICKI; SANTOS, 2020)

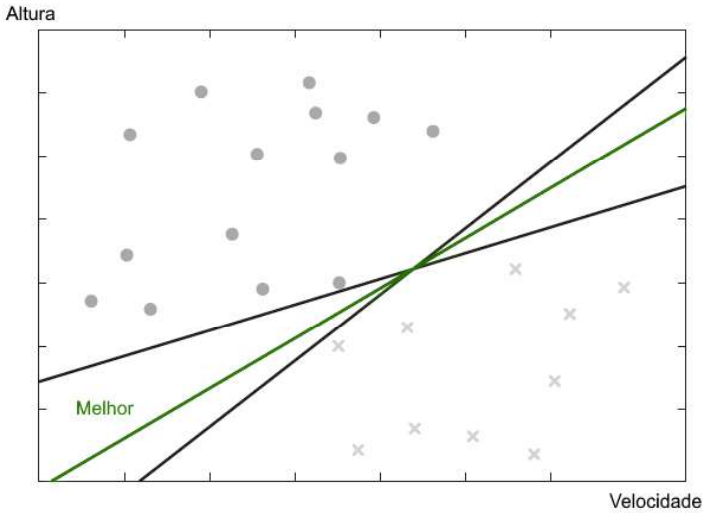
3.7 Máquinas de Vetores de Suporte (SVM)

O método de Máquinas de Vetores de Suporte, também é conhecido pelo seu termo em inglês “Support Vector Machine” (SVM), utiliza funções lineares em um espaço de características de alta dimensão, são treinadas com um algoritmo de aprendizado proveniente de duas grandes teorias, são elas a teoria da generalização, cujo o principal objetivo é fornecer técnicas matemáticas necessárias para encontrar hiperplanos que otimizem as medidas de margem, evitando overfitting (sobreajuste, em tradução nossa), e a teoria da otimização, o qual fornece técnicas matemáticas

para encontrar hiperplanos que otimizem essas medidas. Esse método de aprendizado computacional é conhecido devido a sua eficiência em treinamento e teste e sua capacidade de lidar de maneira eficaz com o overfittinq. Também é utilizado por muitos estudos e resoluções de problemas complexos, como estratificação de risco e sistemas de recomendação, citando como exemplo o estudo “Content-based Recommender System using Social Networks for Cold-start Users” dos autores Prando; Contratres; Solange Souza; Luiz Souza.

SEGARAN (2007), em sua introdução ao assunto, explica que seu objetivo é pegar bancos de dados com inputs numéricos e tentar prever em qual categoria, entre duas possíveis, eles são classificados, construindo um modelo preditivo que funciona com base na procura da linha que divide exatamente as duas categorias. Para exemplificar, o autor considera a situação de uma equipe de basquete: considerando a dinâmica do jogo, posições na frente requerem jogadores mais altos, ao passo que no fundo da quadra é mais vantajoso o time contar com jogadores mais ágeis. Classificar os jogadores com relação a suas alturas e velocidades possibilita, então, categorizar quais jogariam de forma mais eficiente em cada uma das posições. A forma como o modelo funciona pode ser melhor visualizada plotando os dados em um gráfico como o seguinte:

Figura 4 – Exemplo do modelo Máquinas de Vetores de Suporte



Fonte: Adaptado de Programming Collective Intelligence: Building Smart Web 2.0 Applications (SEGARAN, 2007, p. 289)

No exemplo, plotando as alturas e velocidades dos jogadores, é possível categorizá-los de várias formas possíveis, dividindo o gráfico com linhas. Há, contudo, uma linha que representa uma divisão mais limpa, com os dados dos pontos estando o mais longe possível dela; esta é a marcada em verde e denominada de “Melhor”. O objetivo desse modelo é justamente encontrar qual é essa linha, determinando-a partir dos pontos que devem estar mais próximos dela - denominados de vetores de suporte. Delimitadas as áreas de cada categoria com a linha, plotar novos dados no gráfico é só o que é necessário para classificá-los, de forma que o processo é bem ágil.

3.8 Análise de Componentes Principais

A Análise de Componentes Principais, conhecida também pela sua sigla em inglês “PCA” (Principal Component Analysis), é uma técnica estatística para reduzir a dimensionalidade de dados. Segundo (CASTRO; FERRARI, 2017), transformando um conjunto de atributos possivelmente correlacionados em um conjunto de atributos linearmente descorrelacionados chamados de componentes principais, que capturam a maior variabilidade dos dados. Em outras palavras, essa técnica projeta os dados em um espaço de dimensão menor, mantendo a informação mais relevante e reduzindo a complexidade do conjunto de dados.

3.9 Medidas de desempenho de modelos de classificação

A partir dos resultados obtidos pelos modelos de classificação, as organizações podem decidir quais decisões estratégicas tomarão. Desta forma, é de extrema importância analisar medidas estatísticas que garantam a confiabilidade do modelo. As medidas tipicamente utilizadas para avaliar modelos de classificação são acurácia, precisão, sensibilidade e especificidade. Detalhes sobre as medidas são apresentados a seguir (FACELI, 2011)

- Acurácia: A acurácia é uma métrica facilmente interpretável, varia de 0 a 1, sendo 0 o pior valor e 1 o melhor valor. É calculada pela soma dos valores da diagonal principal da matriz e dividida pela soma dos valores dos elementos da matriz.

$$ac(\hat{f}) = \frac{VP + VN}{n}$$

- Precisão: Variando de 0 a 1, sendo 0 o pior valor e 1 o melhor valor. Trata-se da proporção de exemplos positivos classificados corretamente entre todos aqueles que foram preditos como positivo por $\hat{f}$.

$$\rho rec(\hat{f}) = \frac{VP}{VP + FP}$$

- Sensibilidade: Variando de 0 a 1, sendo 0 o pior valor e 1 o melhor valor. Representa a taxa de acerto na classe positiva (TVP)

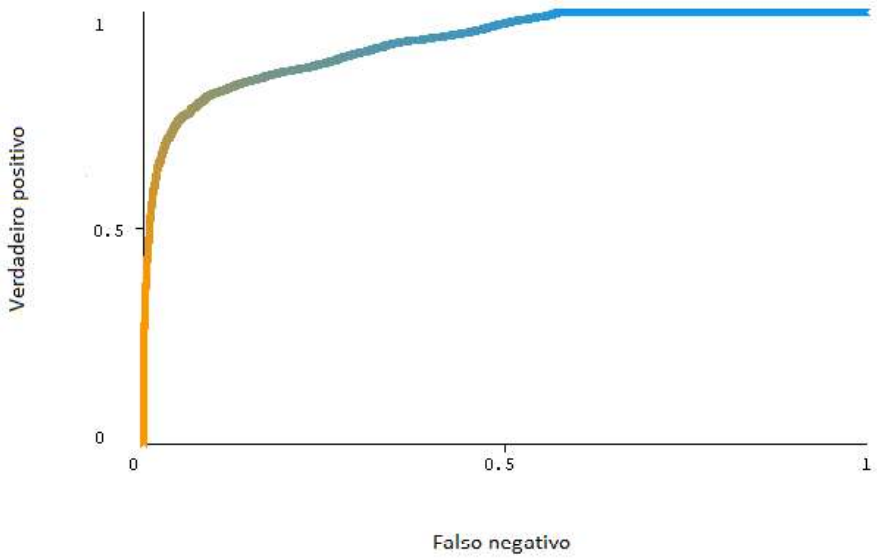
$$sens(\hat{f}) = rev(\hat{f}) = TVP(\hat{f}) = \frac{VP}{VP + FN}$$

- Especificidade: Variando de 0 a 1, sendo 0 o pior valor e 1 o melhor valor. Representa a taxa de acerto na classe negativa (TFP)

$$esp(\hat{f}) = \frac{VN}{VN + FP} = 1 - TFP(\hat{f})$$

Assim como a acurácia, precisão, sensibilidade e especificidade, o parâmetro Receiver Operating Characteristic (ROC), ou Característica de Operação do Receptor, em tradução nossa, é uma medida de desempenho para problemas de classificação, que apresenta o quanto o modelo é capaz de distinguir entre as classes. Quanto mais próximo de 1 a curva ROC do modelo se aproxima, melhor será a capacidade de classificação do algoritmo, em contrapartida, quanto mais próximo de 0, pior é a sua capacidade de classificação. No entanto, quando a área sob a curva está muito próxima de 0,5 significa que o modelo não é preciso. A figura 5 a seguir exemplifica uma curva ROC.

Figura 5 - Exemplo de uma curva ROC



Fonte: Adaptado de Inteligência artificial: uma abordagem de aprendizado de máquina (FACELI, 2011)

4. DESENVOLVIMENTO DA PROVA DE CONCEITO

A aplicação trata-se do uso de algoritmos de aprendizado computacional para predição de readmissão hospitalar de pacientes diagnosticados com diabetes mellitus. Para isto, optou-se pela utilização da linguagem Python que é muito usado para o desenvolvimento de soluções de aprendizado de máquina.

4.1 Contexto de negócio

A gestão de recursos no ambiente clínico hospitalar e seguradoras de saúde é de extrema importância para garantir a continuidade do negócio. Dado este fato, a preocupação com o risco à saúde é grande, devido ao potencial que representa para os sinistros excederem os recursos disponíveis.

Um dos principais contribuintes para o total de despesas médicas e, inclusive, um indicador de suma importância para a qualidade do atendimento oferecido, é a readmissão hospitalar. Estudos concluem que grande parte dessas readmissões médicas são provenientes de pacientes com condições crônicas (DUNGAN, 2012).

O propósito deste estudo, como apresentado em capítulos anteriores, não é, por si só, uma contribuição para saúde do país, uma vez que a base de dados utilizada pode não refletir exatamente a realidade brasileira. O foco principal deste estudo reside na análise e comparação dos resultados de quatro modelos computacionais - Regressão Logística, Máquina de Vetores de Suporte, Floresta Aleatória, Árvore de decisão e Redes Neurais Artificiais -, aplicados à predição de readmissão hospitalar de pacientes diagnosticados com diabetes mellitus. Essas técnicas e conceitos tem o potencial de servir como base para futuras análises em bases de dados brasileiras. Esta prova de conceito utiliza dados de 130 hospitais nos EUA, fornecidos pelo Centro de Pesquisa Clínica e Translacional da Virginia Commonwealth.

4.2 Linguagem de Programação utilizada

Para o desenvolvimento da aplicação, optou-se pela utilização do Python por ser uma linguagem de programação de código aberto e vasto ecossistema de bibliotecas para os mais diversos fins, incluindo pré-processamento de dados e algoritmos de aprendizado.

- Python 3.9.13 – Linguagem de programação de código aberto de alto nível, de propósito geral. Utilizada neste estudo para a preparação dos dados que serão utilizados pelas aplicações de aprendizado de máquina, passando pelas etapas de ingestão e tratamento dos dados, análise exploratória e pré-processamento. Uma vantagem adicional do Python é que, para realizar análises de dados de maneira produtiva, não é necessário possuir proficiência avançada na linguagem (MCKINNEY, 2018).

4.2.1 Bibliotecas Python

Para utilizar as bibliotecas mencionadas a seguir, basta efetuar sua instalação e importá-las no código Python.

- **Pandas** – Fornece estruturas de dados e ferramentas para manipulação e análise de dados.
- **Numpy** – Abreviatura de Numerical Python – Essa biblioteca fornece suporte para arrays e funções matemáticas de alto desempenho.
- **Matplotlib** – Biblioteca de visualização de dados, permitindo criar gráficos estáticos ou interativos, figuras 2D ou 3D.

  Scikit-Learn – Biblioteca de aprendizado computacional, fornece diversos algoritmos e ferramentas para pré-processamento de dados, tarefas de classificação, regressão e avaliação dos algoritmos.
- **SMOTE** – Abreviatura de Synthetic Minority Over-sampling Technique (Técnica de Superamostragem da Minoria Sintética, em tradução nossa). É uma técnica de processamento de dados para lidar com desequilíbrios de classe, utilizada no campo de aprendizado de máquina para tarefas de classificação.

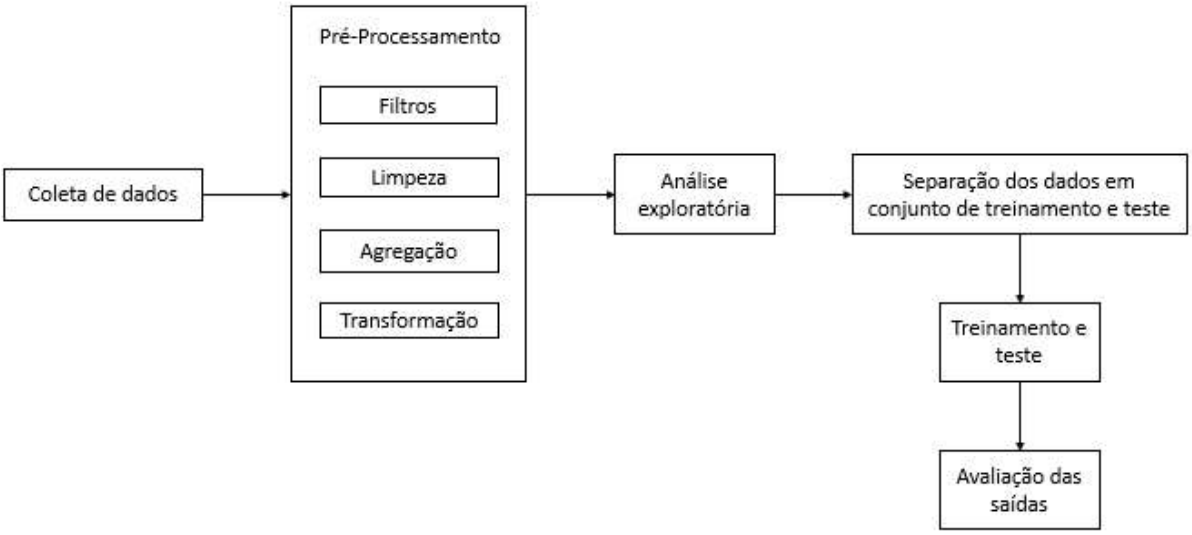
- **Keras –** Técnica muito popular para uso de redes neurais.

4.3 Fluxo do processo

Para alcançar o objetivo deste estudo, que é a predição de readmissão hospitalar de pacientes diagnosticados com diabetes, foram adotadas como referência, embora não restritas a elas, as etapas convencionais para a criação de um processo preditivo, essas etapas são representadas através da figura 6 a seguir. Essas etapas são detalhadas no livro de (CASTRO; FERRARI, 2017), que destaca quatro etapas que desempenham papeis cruciais no processo de criação e avaliação de modelos preditivos:

- **Pré-processamento de dados:** preparação da base de dados, podendo envolver todas as etapas típicas de pré-processamento de dados, como limpeza, integração, redução e transformação.
- **Separação dos dados em conjuntos de treinamento e teste:** Uma parte dos dados é usada para treinar o modelo, e a outra parte é usada para avaliar a qualidade do modelo gerado.
- **Treinamento e teste:** O treinamento é realizado usando, naturalmente, os dados de treinamento para então, ajustar os parâmetros livres do modelo, de tal forma que o seu desempenho atinja determinado nível de qualidade. Por sua vez, o desempenho é avaliado considerando os dados de teste.
- **Avaliação da saída:** Os modelos preditivos geram suas saídas, que são avaliadas quanto ao desempenho, com ênfase na capacidade de generalização. As medidas de avaliação se baseiam no cálculo de erros entre a saída do modelo e a saída desejada, uma vez que os rótulos das classes do treinamento já são previamente conhecidos.

Figura 6 - Fluxo do processo



Fonte: Adaptado de (CASTRO; FERRARI, 2017) p 318.

4.4 Base de dados

Foram utilizados dados provenientes de registros médicos eletrônicos anonimizados, com histórico de 10 anos, contemplando o período de 1999 até 2008, de atendimentos clínicos em 130 hospitais dos Estados Unidos, fornecidos pelo Centro de Pesquisa Clínica e Translacional da Virginia Commonwealth e distribuídos pelo repositório de aprendizado de máquina da Universidade da Califórnia, Irvine (UCI).

A base de dados inicial satisfaz os seguintes critérios:

1. É uma admissão médica hospitalar.
2. Trata-se de um paciente que possui diagnóstico de diabetes.
3. Tempo de internação mínimo de 1 dia.
4. Exames laboratoriais foram realizados durante o encontro.
5. Medicamentos foram administrados durante o encontro.

Possuindo um total de 101.766 registros, a base contém tanto dados demográficos como gênero, raça, idade, entre outros. Como também contempla dados clínicos hospitalares, como diagnósticos, resultados de exames e medicações. O quadro no

apêndice A.1 trata-se de uma adaptação da lista de características do conjunto de dados inicial utilizada no estudo de (STRACK et al, 2014), estudo este, que serve de referência para o conjunto de dados em questão.

4.5 Pré-processamento

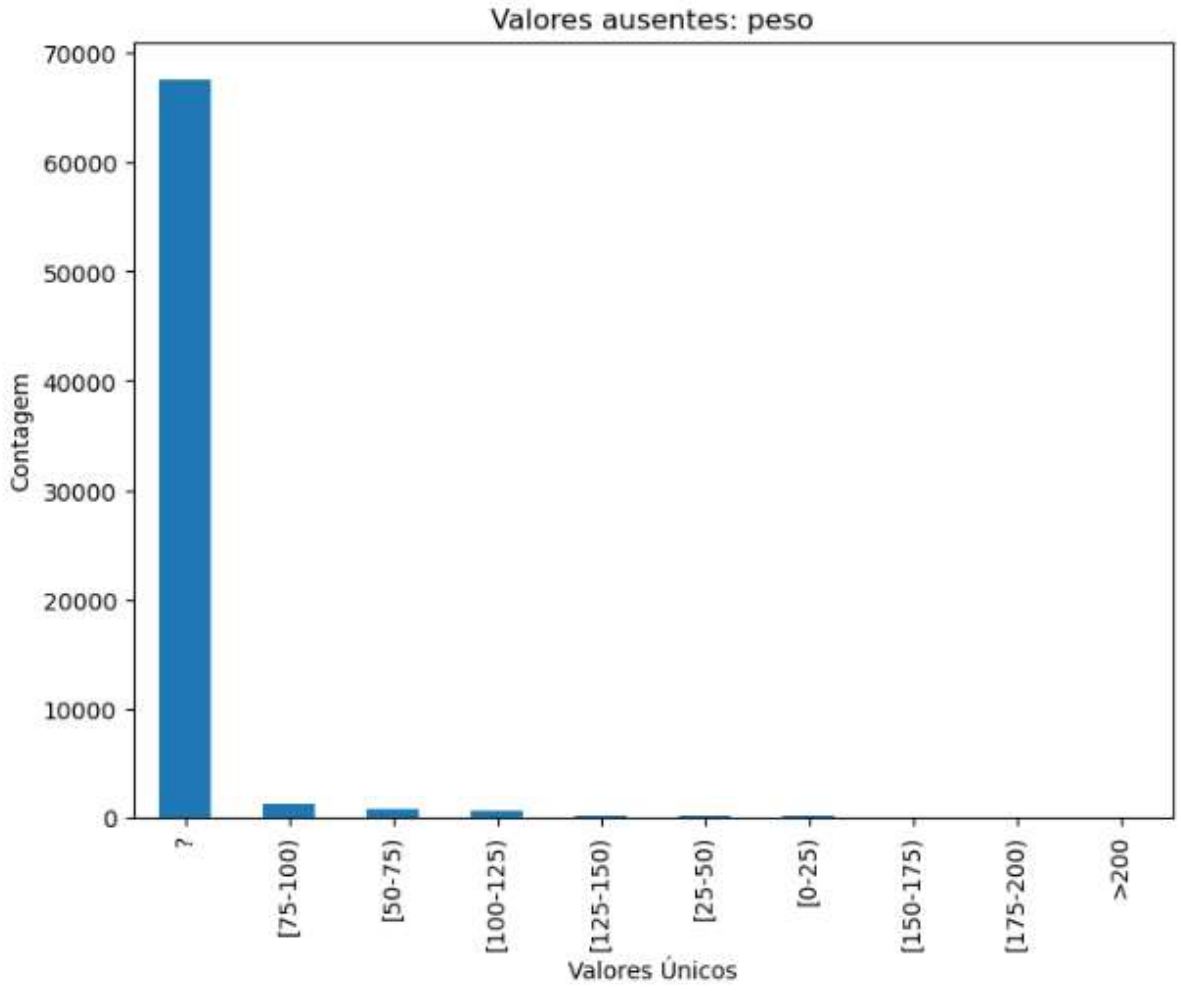
Diversos livros e pesquisas de referência ressaltam a importância do pré-processamento de dados, ao exemplo de GARCÍA; LUENGO; HERRERA, que destaca, no livro intitulado “Data Preprocessing in Data Mining”, publicado em 2015, que grande quantidade dos dados no mundo real são altamente influenciados por fatores negativos, como a presença de ruído, dados inconsistentes e supérfluos, de tamanhos enormes em ambas as dimensões, exemplos e recursos.

O pré-processamento de dados é um passo primordial no fluxo de trabalho de análise dos dados, envolvendo a preparação dos dados brutos, melhorando a qualidade dos dados, corrigindo erros, lidando com valores ausentes e aplicando técnicas de transformação para, ao final do processo, transformar a base em um formato que os algoritmos de aprendizado computacional possam compreendê-la de forma eficaz, afinal, dados de baixa qualidade levarão, irremediavelmente, a resultados de baixa qualidade.

4.6 Aplicação do pré-processamento de dados

Ao realizar uma análise preliminar da base de dados que pode ser observada na análise exploratória inicial de dados no apêndice A.3, optou-se pela remoção dos atributos: peso e código do pagador, pois possuem 97% e 40% de valores ausentes, respectivamente, conforme os gráficos 7 e 8 a seguir. (STRACK et al, 2014), analisando o mesmo conjunto de dados, explica que os valores ausentes do atributo de peso podem ser pelo fato de que hospitais e clínicas não eram obrigados a capturar essa informação em um formato estruturado antes da legislação HITECH da lei americana de reinvestimento e recuperação em 2009.

Figura 7 - Valores ausentes do atributo peso



Fonte: Próprio autor

Figura 8 - Valores ausentes do atributo código do pagador

Fonte: Próprio autor

Dado que o conjunto inicial possui múltiplas internações para alguns pacientes, foi adotado o mesmo tratamento realizado no estudo de STRACK, que por sua vez, foi utilizado em outros estudos semelhantes, mantendo apenas um encontro por paciente e removendo encontros que resultaram em óbito ou liberação e transferência para outra unidade de internação para evitar o viés da análise e dos algoritmos, tratamento realizado no apêndice A.3.

Antes de efetuar os tratamentos dos dados de baixa porcentagem, foi realizado testes com diferentes técnicas, preenchendo os valores ausentes com o dado da linha anterior, com o valor mais frequente e agrupando. Dado os resultados, optou-se pelo tratamento a seguir pois representaram melhores resultados. Sobre os atributos que possuem uma baixa porcentagem de valores ausentes ou que representam potencial técnico para os algoritmos, os registros foram preenchidos com o valor mais frequente

na base, esses atributos são: raça, diagnóstico primário e especialidade médica, exemplificados na tabela 2 a seguir e no apêndice A.7:

Tabela 2 - Valores ausentes por atributo

Atributo	Valores ausentes	Porcentagem em relação ao total
raça	2273	2,00%
especialidade médica	49949	49%
diagnóstico primário	21	0,02%

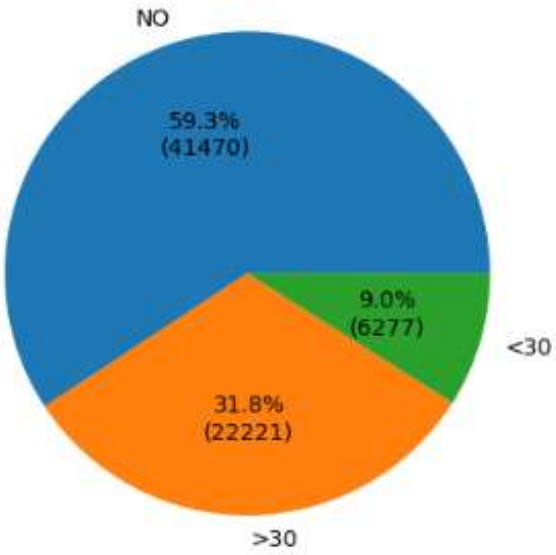
Fonte: Próprio autor

Optou-se por remover os atributos examide, citoglipton e glimeripide-pioglitazone, pois possuem apenas um valor distinto, o que, por consequência, é irrelevante para o aprendizado (Apêndice A.9). Além disso, foram removidos o número do paciente e o número do encontro, pois ambos também não possuem relevância para o aprendizado de máquina.

A variável alvo “readmitido” inicialmente possuía 3 valores: “NO”, representando que não se trata de uma readmissão, “>30” quando a readmissão foi após 30 dias em relação a última internação, e “<30”, quando a readmissão foi inferior a 30 dias em relação a última internação.

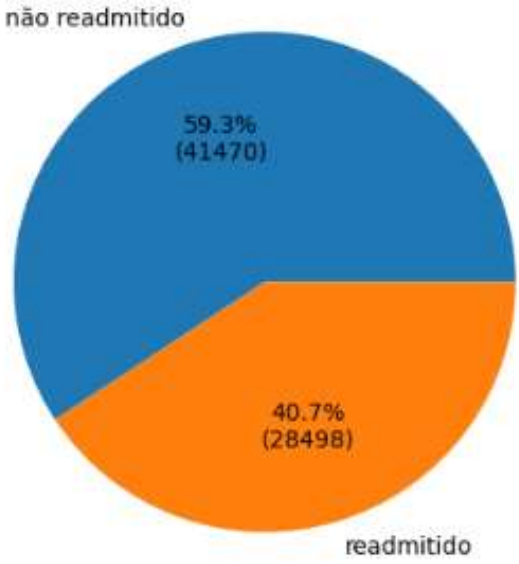
O objetivo é fazer a predição da readmissão, neste caso, optou-se por ajustar a variável alvo para apenas 2 valores: 1 que representa readmissões médicas, naturalmente contemplando “>30” e “<30” e 0 para não readmissões. Este tratamento pode ser consultado no apêndice A.10. A seguir, é ilustrado, através das figuras 9 e 10, o antes e depois da recategorização, respectivamente:

Figura 9 - Variável alvo antes da transformação



Fonte: Próprio autor

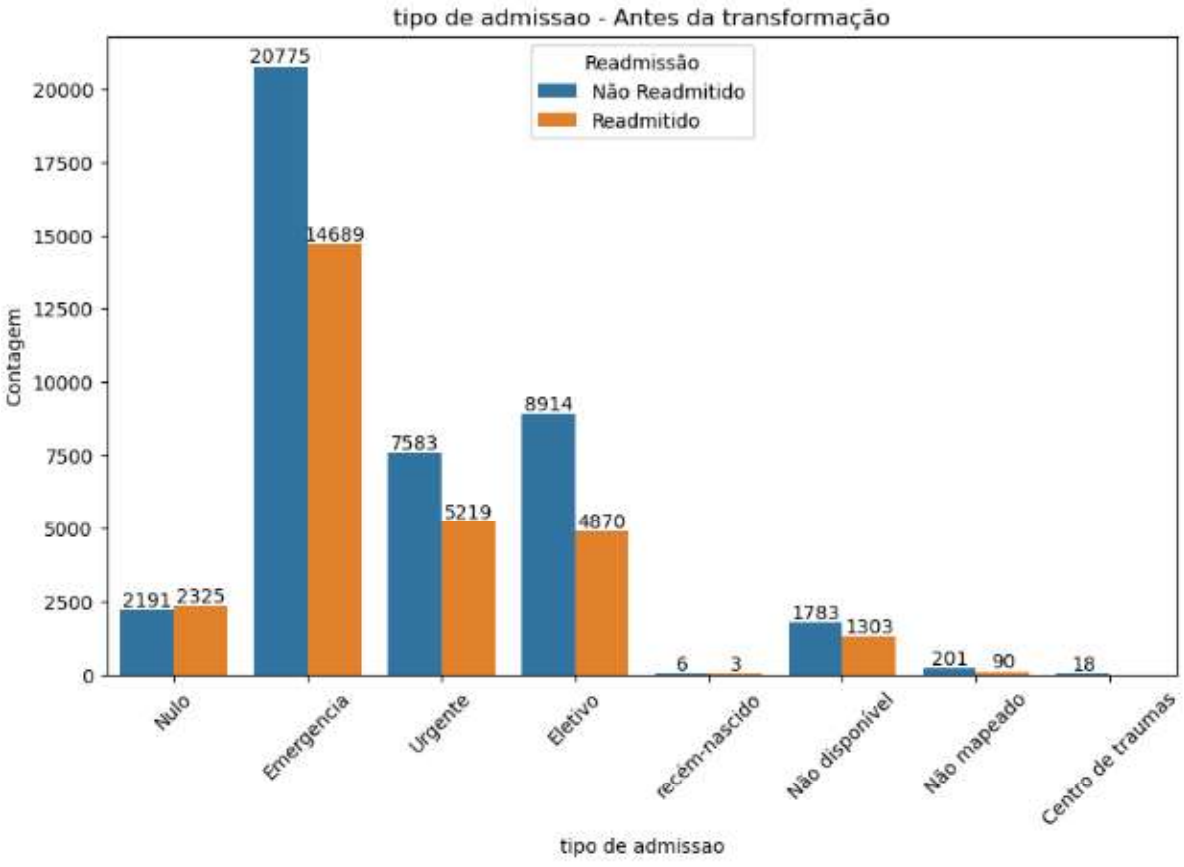
Figura 10 - Variável alvo depois da transformação



Fonte: Próprio autor

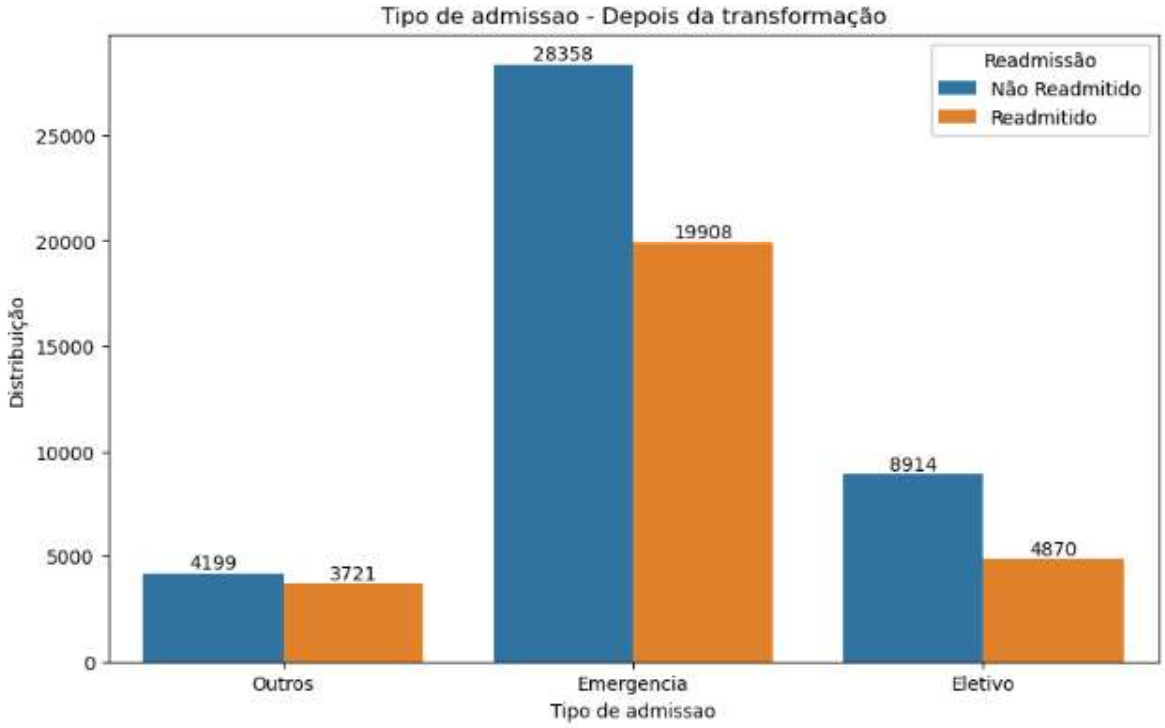
O atributo “tipo de admissão” foi recategorizado, distribuindo os pacientes por faixas menores, pois, conforme a figura 11 a seguir, pode-se observar que possui 3 grandes valores significativos e muitos valores inválidos e pouco significativos para o aprendizado computacional, nestes casos, foram agrupados em somente um valor apresentado como “outros”. A figura 12 ilustra o pós-tratamento desse atributo.

Figura 11 - Tipo de admissão antes da transformação



Fonte: Próprio autor

Figura 12 - Tipo de admissão depois da transformação



Fonte: Próprio autor

O atributo de disposição de alta, que possui 23 valores distintos, foi recategorizado com os valores que representam a alta do paciente e agregando em um valor chamado “Alta para casa”, e os demais foram recategorizados para “Outros” (apêndice A.10). O atributo “fonte de admissão” seguiu o mesmo conceito. As técnicas de redução de dimensionalidade realizadas para os atributos “fonte admissão” e “tipo de admissão”, foram utilizadas em trabalhos semelhantes como o de (STRACK et al, 2014). Estes agrupamentos podem ser consultados no apêndice A.11.

Os atributos de diagnóstico 1, diagnóstico 2 e diagnóstico 3, que representam o CID-9 e possuíam inicialmente 848, 923 e 954 valores distintos, também foram recategorizados para faixas menores, de acordo com seus respectivos CIDS, a fim de evitar uma forte correlação entre as variáveis, o que poderia impactar de forma negativa o aprendizado dos algoritmos (Apêndice A.12).

As variáveis “teste AC1” e “teste de soro de glicose” também sofreram alterações, onde os valores que representam um resultado acima do normal, foram alterados para

o valor numérico 1, os resultados normais foram alterados para o 0 e por fim, o valor -99 foi incluído para os exames que não foram realizados.

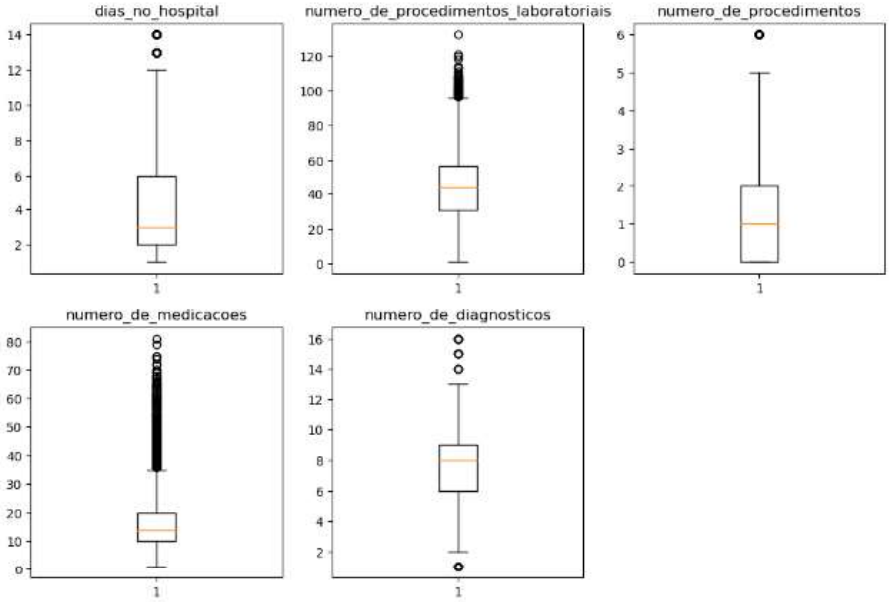
Referente os atributos de medicações que representam se o medicamento foi prescrito para o paciente ou se houve alteração na dosagem (apêndice A.13). Esse atributo foi alterado para o tipo binário, aplicando 1 para quando o medicamento foi administrado e 0, quando não foi. Tecnica esta pode ser consultada no apêndice A.14, também empregada pelo estudo de (ASLAM et al, 2021).

Foram criadas duas medidas que representam o uso de serviços no ano anterior (essa métrica trata-se da soma de consultas ambulatoriais, emergenciais e internações). E o total de medicamentos usados (trata-se da soma dos medicamentos prescritos). (Apêndice A15).

Optou-se por remover os atributos de medicações que possuíam baixa relevância quantitativa com valores diferentes de NO. As colunas afetadas foram: acetohexamide, nateglinide, repaglinide, tolbutamide, acarbose, miglitol, troglitazone, tolazamide, glyburide-metformin, glipizide-metformin, metformin-pioglitazone, chlorpropamide (Apêndice A.16).

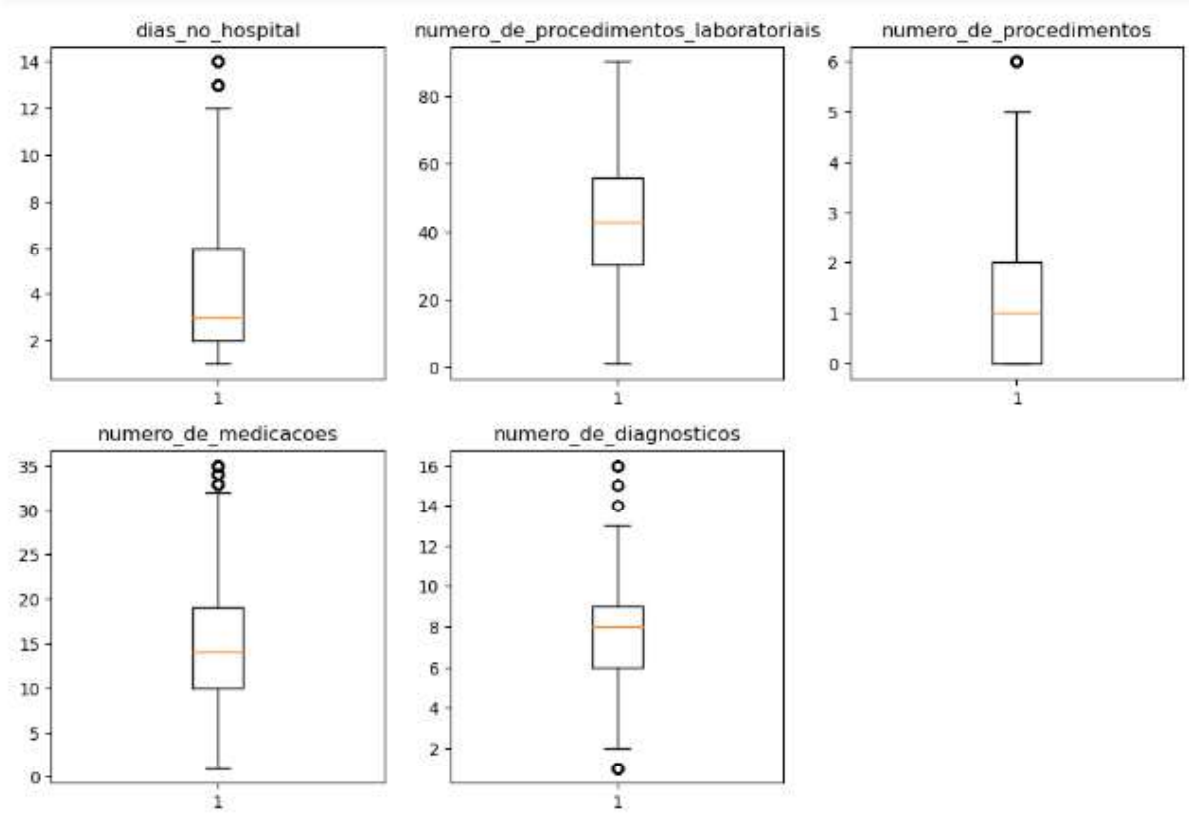
Foi realizado a identificação e tratamento de outliers para as colunas: Número de procedimentos laboratoriais e número de medicações, conforme os gráficos 13 e 14 a seguir e apêndice A.17:

Figura 13 - Outliers de dados numéricos



Fonte: Próprio autor

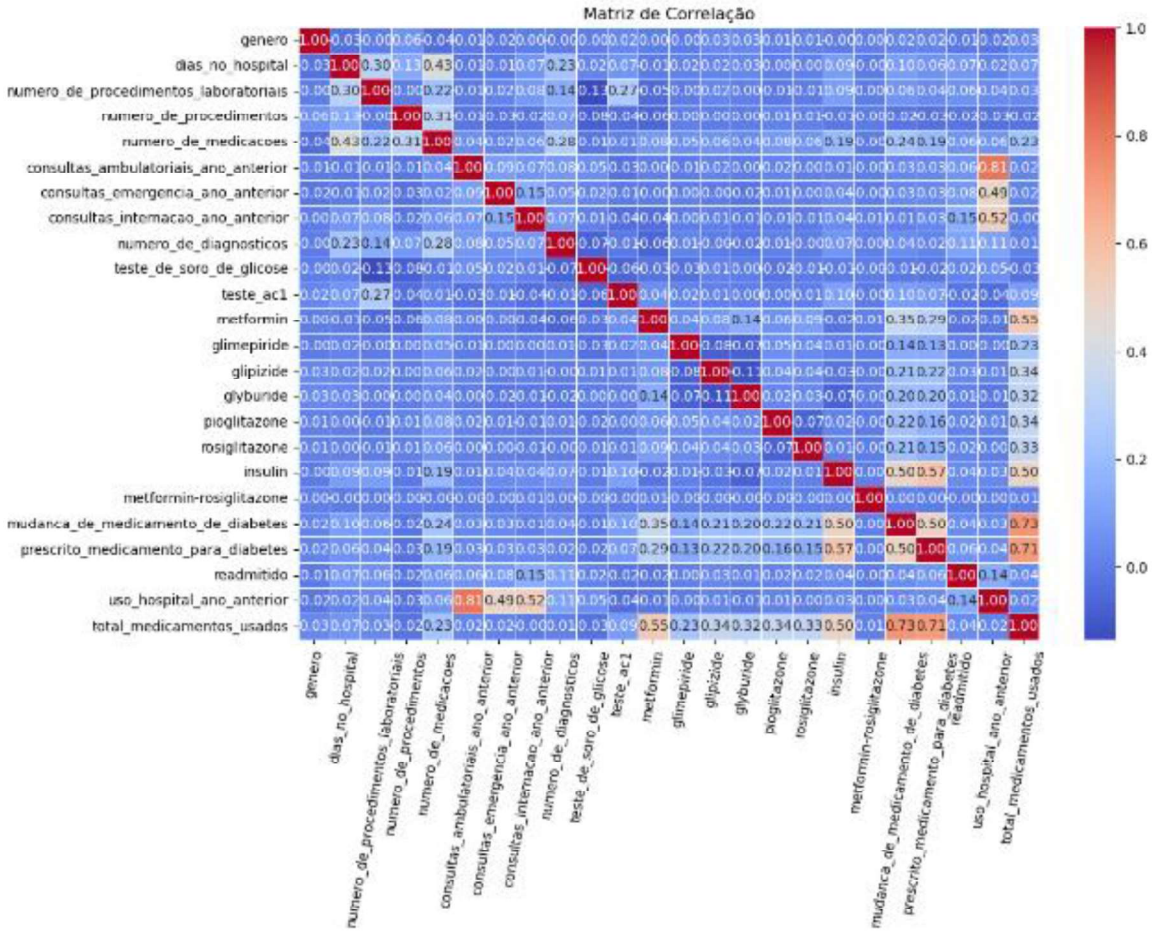
Figura 14 - Após o ajuste de outliers



Fonte: Próprio autor

Observou-se que os atributos número de medicações e uso do hospital no ano anterior, possuem uma correlação entre si. Além do uso do hospital no ano anterior e consultas ambulatoriais também, já que a sua representatividade no conjunto de dados é grande.

Figura 15 - Matriz de correlação



Fonte: Próprio autor

A matriz de correlação, exibida na figura 15 acima, apresenta poucos atributos fortemente correlacionados, não obstante, sabe-se que correlações entre variáveis pode levar a resultados imprecisos, uma vez que os modelos podem atribuir pesos maiores para componentes que apareçam mais de uma vez no conjunto de dados (apêndice A.18). Segundo (FACELI, 2011), a redução de dimensionalidade pode melhorar o desempenho do modelo induzido, reduzir o custo computacional e tornar os resultados obtidos mais compreensíveis. Desta forma, para obter uma tratativa

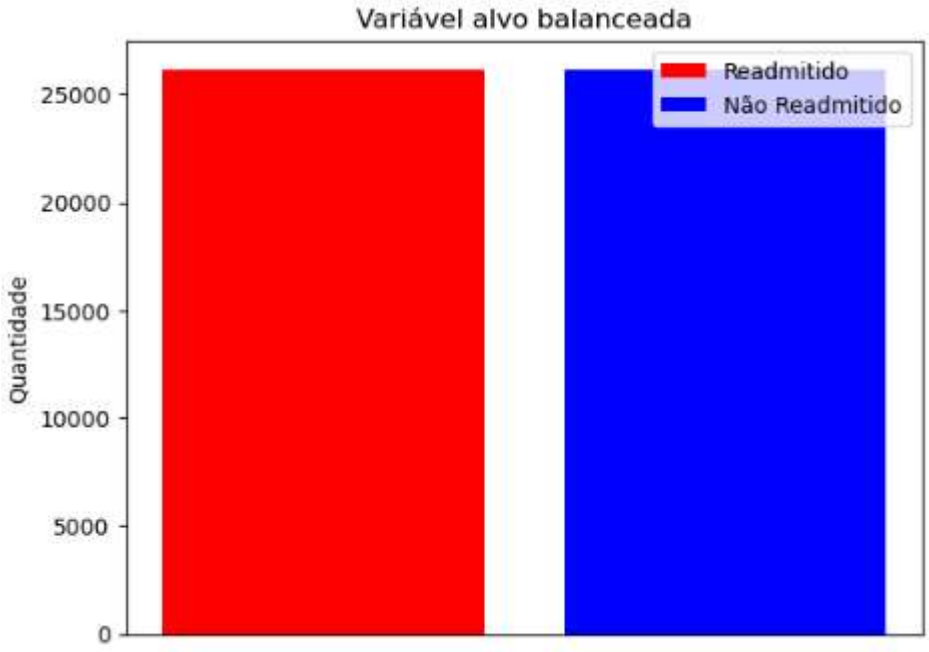
eficaz nestes casos, optou-se por utilizar o método de análise de componentes principais, mais conhecido pelo seu termo em inglês “PCA”.

Após a divisão de treino e teste (Apêndice A.19), verificou-se que a variável alvo “readmitido” não está balanceada, conforme a imagem 16. Neste caso, foi realizado o uso da técnica Oversampling para o balanceamento das classes no conjunto de dados de treino. A imagem 17 ilustra a variável alvo balanceada (Apêndice A.20). Por fim, realizado a limpeza e tratamento de dados, a base final possui um total de 67.845 registros e 31 colunas, conforme o quadro que pode ser consultado no apêndice A.21, que representa o conjunto de dados final.

Figura 16 - Variável alvo antes do balanceamento de classe

Fonte: Próprio autor

Figura 17 - Variável alvo após o balanceamento de classe



Fonte: Próprio autor

4.7 Aplicação e resultados dos modelos

Os testes foram realizados com uma técnica usada em estudos relacionados, com uma divisão aleatória de 70% dos dados do conjunto para treino e 30% para teste, de forma com que cada grupo tenha uma quantidade igual ou semelhante de dados rotulados. Além disto, esta proporção apresentou os melhores resultados dentre os testes realizados em nosso conjunto de dados.

As métricas escolhidas para avaliação da performance entre os modelos, são métricas amplamente utilizadas em diversos estudos de classificação, como apresentado no capítulo 2, de pesquisas referencias. Essas métricas são: Acurácia, Precisão, Sensibilidade, F1-Score e a Curva ROC.

Após o pré-processamento e normalização de dados, um total de 67.845 registros foram utilizados como entrada nos modelos, com um total de 31 atributos, incluindo idade, raça, sexo, tempo de permanência no hospital e medicações administradas ao paciente.

A aplicação dos métodos de classificação podem ser consultados no apêndice A.22. A comparação do desempenho dos cinco algoritmos, relevou que o modelo de Máquina de Vetores de Suporte obteve o melhor desempenho, alcançando a marca de 61% de acurácia, 60% de precisão e 60% de sensibilidade, seguido do modelo de Regressão Logística com 60% de acurácia, 65% de precisão e 60% de sensibilidade, Floresta Aleatória alcançou 59% de acurácia, precisão e sensibilidade, e por fim, os modelos de Redes Neurais Artificiais e Árvore de decisão alcançaram uma acurácia de 59% e 55%, respectivamente. A tabela 3 a seguir, apresenta um resumo com as principais métricas de comparação entre os algoritmos.

Tabela 3 - Resultados dos modelos

Modelo	Acurácia	Precisão	Sensibilidade	F1-Score	Curva AUC
Árvore de decisão	55%	0,54	0,54	0,54	0,55
Regressão Logística	60%	0,59	0,60	0,59	0.64
Máquina de Vetores de Suporte	61%	0,60	0,60	0,58	0,64
Floresta Aleatória	59%	0,59	0,59	0,59	0.63
Redes Neurais Artificiais	59%	0,57	0,57	0,57	0.60

Fonte: Próprio autor. Adaptado de (STRACK, 2014)

No contexto das redes neurais artificiais, os hiper parâmetros adotados foram selecionados com base nos resultados mais promissores durante os testes. O hiper parâmetros de Função de perda com valor “binary_crossentropy” que é aplicado em problemas de classificação binária. Quanto ao otimizador, componente que regula o tamanho dos passos durante a otimização do algoritmo, optou-se pelo método Adam com uma taxa de aprendizado estabelecida em 0.001.

No cenário da Regressão Logística, o hiper parâmetro hiper parâmetro de tolerância para critério de parada (tol) foi ajustado para 0.0001, indicando uma convergência mais rigorosa. O parâmetro de regularização Inversa (c) recebeu o valor 0.001, sendo que valores maiores que “C” correspondem a uma regularização mais suave, permitindo uma adaptação mais flexível aos dados de treinamento. O algoritmo de otimização (solver) escolhido foi o “liblinear”, conhecido pela sua eficácia em problemas binários.

Referente ao modelo de Árvore de Decisão, o hiper parâmetro que controla o critério de seleção de recursos (criterion), foi configurado como “entropy”, que se refere ao critério para medir a impureza de um nó da árvore. O número mínimo de amostras em folhas foi estipulado em 10, enquanto a estratégia de divisão foi definida como “best”, indicando que o algoritmo deve escolher a divisão que estatisticamente resulta no melhor desempenho.

Sobre o algoritmo de Floresta Aleatória, os hiper parâmetros foram especificados com um número mínimo de amostras em folhas de 50 e o número de estimadores que é o total de árvores na floresta foi de 300.

Finalmente, para o algoritmo de Máquinas de Vetores de Suporte, o parâmetro de regularização que controla a penalidade por erros de classificação foi fixado em “1”, e o tipo de kernel definido como “linear”.

Referente os resultados, o algoritmo de Máquinas de Vetores de Suporte pode ter tido melhores resultados devido a sua capacidade de lidar com dados complexos, não lineares e com uma alta dimensionalidade, pois apesar do tratamento realizado na etapa de pré-processamento de dados, o conjunto de dados final ainda assim possui muitas características.

A pesquisa conduzida por Y. ZHAO et al em 2019 buscou prever readmissões hospitalares de pacientes diagnosticados com diabetes em até 30 dias após a alta hospitalar. Seu conjunto de dados, composto por 124.678 registros de admissões, abrangeu o período de 2012 a 2014 e foi extraído do banco de dados de internações do estado da Flórida. Embora o estudo de Y. ZHAO et al tenha alcançado um desempenho de 72%, sua pesquisa diverge do nosso foco, que se concentra na previsão de readmissão hospitalar sem especificar o período de até 30 dias.

Já a pesquisa de TAMIN e ISWARI, realizada em 2017, alcançou uma acurácia de 76% ao prever readmissões em pacientes com diabetes, usando o algoritmo C4.5. Apesar desta pesquisa ter utilizado a mesma base de dados que a nossa, apresenta diferenças nos tratamentos dos dados. Para alcançar a acurácia de 76% TAMIN e ISWARI optaram por criar um conjunto de dados separado para teste, com a remoção diagnósticos CID "E" até "V" para substituir os valores por dados numéricos e remover

as linhas com valores "não" para a variável "readmitido". Já um de seus conjuntos de dados que não faz essa remoção, obteve uma acurácia de 57%.

Diferentemente deste estudo, a nossa abordagem envolveu a redução de dimensionalidade proposta por (STRACK, 2014), levando em consideração a existência de aproximadamente 950 códigos CID distintos em nosso conjunto de dados.

Adicionalmente, o estudo de SHANG et al, realizado em 2021, buscou prever readmissões para pacientes com diabetes em até 30 dias após a alta, obtendo uma curva AUC de 0,68, com tratamentos de dados semelhantes aos nossos. Embora utilizassem a mesma base de dados e alguns tratamentos de dados em comum, como a redução de dimensionalidade dos diagnósticos de acordo com a codificação da CID-9 e substituição modal de dados ausentes em atributos como "raça" e "especialidade médica", as pesquisas divergiram-se também na técnica de balanceamento de dados, adotando o Undersampling enquanto nós optamos pelo Oversampling.

Os resultados obtidos em nossa pesquisa, com uma acurácia de 61% e curva AUC de 0,64, convergem bem com os alcançados nas pesquisas relacionadas. Por outro lado, as particularidades em nosso estudo se concentram não apenas na previsão de readmissões dentro do período de 30 dias, mas também além desse intervalo temporal. É importante destacar que, mesmo nos estudos relacionados que se utiliza da mesma base de dados que o nosso, divergem-se nos métodos de pré-processamento. Essas variações nos conjuntos de dados, características e métodos de pré-processamento explicam a discrepância nos resultados entre os estudos mencionados. Vale ressaltar que se torna desafiador realizar uma comparação minuciosa em relação aos hiper parâmetros dos algoritmos, uma vez que detalhes específicos sobre tais configurações não foram mencionados nos estudos relacionados.

5. CONCLUSÃO

Diante do crescente aumento da incidência de diabetes tanto no Brasil quanto globalmente, antecipar as readmissões hospitalares de pacientes diagnosticados com esta condição torna-se fundamental. Essa medida não somente aprimora a qualidade dos cuidados de saúde prestados aos pacientes, como também implica em benefícios significativos para os hospitais e seguradoras de saúde. Ao prever e mitigar readmissões, os hospitais conseguem gerenciar recursos de forma mais eficiente, reduzir custos operacionais e melhorar a alocação de pessoal e leitos, tudo isso contribuindo para a otimização do Sistema Único de Saúde.

Além disso, esse gerenciamento proativo abre espaço para cuidado pós-alta e gestão do cuidado contínuo aos pacientes e programas de incentivo à saúde dos pacientes. Ao identificar antecipadamente as necessidades dos indivíduos com diabetes, é possível implementar estratégias de intervenção precoce e personalizada, promovendo um acompanhamento mais atento e eficaz. Isso não apenas reduz as readmissões, mas também melhora a qualidade de vida dos pacientes, possibilitando a gestão mais eficiente e bem-sucedida da condição.

Essa abordagem, centrada na prevenção de readmissões e na gestão proativa da saúde dos pacientes, é fundamental para um sistema de saúde mais eficiente e centrado no paciente.

Neste estudo, o objetivo foi a previsão de readmissões hospitalares de pacientes diabéticos, empregando técnicas de pré-processamento de dados e a aplicação de modelos computacionais. Ao analisar os resultados dos modelos, podemos constatar de maneira quantitativa os benefícios de aplicá-los.

Os resultados apontam que o algoritmo Máquina de Vetores de Suporte obteve o desempenho mais consistente, alcançando uma acurácia de 62% na predição das readmissões. Esse modelo destacou-se como a escolha mais eficaz dentre as opções analisadas neste estudo, no contexto da base de dados utilizada. Além disso, os resultados evidenciaram a relevância de variáveis como o número de hospitalizações anteriores, tempo de permanência no hospital, quantidade de medicações e procedimentos para a realização das predições.

É importante ressaltar, no entanto, que as conclusões deste estudo devem ser interpretadas considerando as limitações da base de dados utilizada, pois abrange o período de 1999 a 2008 e pacientes diabéticos dos Estados Unidos. Os resultados podem variar de acordo com as características da base de dados, o período, o país de origem das informações e inclusive, as particularidades culturais da população estudada.

5.1 Contribuições do trabalho

Embora as internações hospitalares representem uma das maiores parcelas de custos na área da saúde, e a incidência de pessoas com diabetes no Brasil esteja aumentando ao longo dos anos, a quantidade de estudos relacionados à predição de readmissões hospitalares em pacientes diagnosticados com diabetes, através de algoritmos de aprendizado de máquina, ainda é limitada.

A principal contribuição deste trabalho reside na aplicação de técnicas e conceitos de métodos de classificação, tipicamente empregados em estudos acadêmicos no exterior, para avaliação e comparação dos resultados de diferentes modelos com o propósito de prever readmissões hospitalares de pacientes com diabetes. O destaque da contribuição, está na sua adaptabilidade para bases de dados brasileiras, permitindo o uso das técnicas e conceitos explorados neste estudo, em contextos nacionais.

5.2 Trabalhos futuros

Empregar o uso dos conceitos e técnicas utilizados neste estudo, com uma base de dados nacional, pode relevar resultados interessantes em predições de pacientes diagnosticados com diabetes mellitus, devido as particularidades do país, como o diferente estilo de vida, por exemplo. Vale ressaltar que ao empregar a abordagem proposta em outros conjuntos de dados, é importante realizar uma análise detalhada das características individuais de cada conjunto. Isso inclui a identificação de padrões específicos, seleção de modelos adequados ao contexto, o seu devido pré-

processamento de dados e ajustes de hiper parâmetros para garantir a melhor performance possível.

Outro passo adiante é a inclusão de dados de triagem, pois acrescentar mais dados relacionados a saúde do paciente, pode gerar resultados significativos nas predições, aumentando a assertividade das classificações.

Por último e não menos importante, pode ser realizado a colaboração com profissionais de saúde para compreender as nuances específicas do cenário de saúde do conjunto de dados estudado.

REFERÊNCIAS BIBLIOGRÁFICA

ALPAYDIN. **Introduction to machine learning.** Cambridge, Massachusetts. 4. Ed. 712 p. 2020

ASLAM et el. **Predicting Diabetic Patient Hospital Readmission Using Optimized Random Forest and Firefly Evolutionary Algorithm**. 11. Ed, 2021. Disponível em: https://www.researchgate.net/publication/355778740_Predicting_Diabetic_Patient_Hospital_Readmission_Using_Optimized_Random_Forest_and_Firefly_Evolutionary_Algorithm. Acesso em: 03 de mar. de 2023

BÉLGICA. INTERNATIONAL DIABETES FEDERATION. **IDF Diabetes Atlas**, 10. ed. Brussels, Belgium: 2021. Disponível em: https://www.diabetesatlas.org. Acesso em: 30 out. 2023.

BRASIL. MINISTÉRIO DA SAÚDE (MS**). Saúde de A a Z. Diabetes (diabetes mellitus)**. Brasília, 2023 Disponível em: <https://www.gov.br/saude/pt-br/assuntos/saude-de-a-a-z/d/diabetes>. Acesso em: 28 out. 2023.

BRASIL. SOCIEDADE BRASILEIRA DE DIABETES (SBD). **Diagnóstico e Tratamento**. São Paulo, 2023. Disponível em: <https://diabetes.org.br/diagnostico-e-tratamento/>. Acesso em: 28 out. 2023.

BRASIL. AGÊNCIA NACIONAL DE SAÚDE SUPLEMENTAR (ANS). **Mapa Assistencial da Saúde Suplementar**, 2022. Disponível em: https://app.powerbi.com/view?r=eyJrIjoiMTE4YzM2MDUtOTcyMS00ZTg0LWIyZDYtN2QzY2Y1MzAxYWI2IiwidCI6IjlkYmE0ODBjLTRmYTctNDJmNC1iYmEzLTBmYjEzNzVmYmU1ZiJ9. Acesso em: 25 de out. 2023.

BRASIL. AGÊNCIA NACIONAL DE SAÚDE SUPLEMENTAR (ANS). Resolução Normativa nº 546, de 04 de julho de 2018. **Dispõe sobre a concessão de isenção de débitos pelos Conselhos Regionais de Administração, e dá outras providências**. Conselho Federal de Administração, Brasília, 04 de julho de 2018. Disponivel em:https://documentos.cfa.org.br/?c=documento&a=show&id=700. Acesso em: 30 de out. 2023.

CASTRO; FERRARI**. Introdução a mineração de dados**, 2017. 673 p.

DUGGAL, et al. **Predictive risk modelling for early hospital readmission of patients with diabetes in India. International Journal of Diabetes in Developing Countries.** 36. Ed. 519–528 p., 2016 Disponível em: https://link.springer.com/article/10.1007/s13410-016-0511-8. Acesso em: 30 out. 2023.

DUNGAN. **The effect of diabetes on hospital readmissions. J Diabetes Sci Technol**. 6,5. Ed. 2012. Journal of diabetes science and technology, 2012. Disponível em: https://pubmed.ncbi.nlm.nih.gov/23063030/. Acesso em: 30 out. 2023.

FACELI. **Inteligência artificial: uma abordagem de aprendizado de máquina**, 2011. Rio de Janeiro, RJ. 396 p.

F. TAMIN AND N. M. S. ISWARI**. Implementation of C4.5 algorithm to determine hospital readmission rate of diabetes patient**. 4. Ed, 2017. International Conference on New Media Studies (CONMEDIA), Yogyakarta, Indonésia, 2017, 15-18 p. Disponível em: https://ieeexplore.ieee.org/abstract/document/8266024. Acesso em: 30 de out. de 2023.

GARETH, et al. **An Introduction to Statistical Learning with Applications in Python**, 2023. 607 p.

GEETHA; SENDHILKUMAR. **Machine Learning: Concepts, Techniques and Applications**, 2023. 478 p.

GÉRON. **Mãos à Obra: Aprendizado de Máquina com Scikit-Learn & TensorFlow**. (Rafael Contatori, Trad.). Rio de Janeiro, RJ: Alta Books, 2019. 576 p.

Gregor John, Loïc Payrard, Jacques Donzé. **Associations between post-discharge medical consultations and 30-day unplanned hospital readmission: A prospective observational cohort study**. European Journal of Internal Medicine, 2022. Volume 99, 57-62 p. Disponível em: https://www.sciencedirect.com/science/article/pii/S0953620522000140. Acesso em: 01 dez. 2023.

HASTIE; TIBSHIRANI; FRIEDMAN. **The Elements of Statistical Learning: Data Mining, Inference, and Prediction**, 2008. 2. Ed. 311 p.

IZBICKI, R.; SANTOS, T. M. **Aprendizado de máquina: uma abordagem estatística**. São Carlos, SP, 2020. 252 p.

KOZAK**. Decision Tree and Ensemble Learning Based on Ant Colony Optimization**. Katowice, Poland, 2019. 159 P.

LITTLE; SACKS. **"HbA1c: how do we measure it and what does it mean?". Current opinion in endocrinology, diabetes, and obesity**. 16,2. Ed. 113-118 p., 2009. Disponível em: https://pubmed.ncbi.nlm.nih.gov/19300091/. Acesso em: 30. out. 2023

MCKINNEY. **Python para Análise de Dados**. 2. ed. (Lúcia A. Kinoshita, Trad.). São Paulo: Novatec, 2018. 616 p.

MUELLER; MASSARON. **Artificial Intelligence for Dummies**. Hoboken, New Jersey, 2022. 2. Ed. 327 p.

NVIDIA. **Aprendizado de máquina**. 2023, Disponível em: https://www.nvidia.com/en-us/glossary/data-science/machine-learning/. Acesso em: 31 out. 2023.

Quirino, Guilherne & Mals, Nadav & Groterhorst, Vitor & Souza, Solange N A & DeSouza, L.. (2015). Meneduca **— Social school network to support the educational environment**. 1-6. Acesso em: 06 nov. 2023. Disponível em: https://www.researchgate.net/publication/307784059_Meneduca_-_Social_school_network_to_support_the_educational_environment.

SEGARAN. **Programming Collective Intelligence: Building Smart Web 2.0 Applications**, 2007. 360 p.

Shang, Y., Jiang, K., Wang, L. et al**. The 30-days hospital readmission risk in diabetic patients: predictive modeling with machine learning classifiers**. BMC Med Inform Decis Mak 21 (Suppl 2), 57 (2021). https://doi.org/10.1186/s12911-021-01423-y. Disponível em: https://doi.org/10.1186/s12911-021-01423-y. Acesso em: 06 dez. 2023.

STRACK et al. **Impact of HbA1c measurement on hospital readmission rates: analysis of 70,000 clinical database patient records**. BioMed research international vol. 2014. Disponível em: https://pubmed.ncbi.nlm.nih.gov/24804245/. Acesso em: 01 nov. 2023.

UNITED STATES. WORLD HEALTH ORGANIZATION (WHO). **Global Status Report on diabetes**. 2016. 83 p. Disponível em: https://www.who.int/publications/i/item/9789241565257. Acesso em: 30 out. 2023.

UNITED STATES. Centers for Medicare & Medicaid Services (CMS). **Hospital Readmissions Reduction Program (HRRP)**. 2023. Disponivel em: https://www.cms.gov/medicare/payment/prospective-payment-systems/acute-inpatient-pps/hospital-readmissions-reduction-program-hrrp Acesso em: 29 de out. 2023.

Y. ZHAO et al. **Predicting 30-Day Hospital Readmissions for Patients with Diabetes. International Conference on Health Informatics (HIMS)**, 2019. 7 p. Disponível em: https://storm.cis.fordham.edu/~yzhao/tp/Publications/C14_HIMS_2019_readmission.pdf. Acesso em: 30 out. 2023.

APÊNDICE A – CÓDIGO UTILIZADO

A seguir, se encontra o código Python desenvolvido para o estudo.

A.1. Quadro com lista de características do conjunto de dados inicial

Atributo	**Coluna**	**Tipo**	**Descrição e valores**	**% ausentes**
ID do encontro	encounter_id	Numérico	Identificador único de um encontro	0%
Número do paciente	patient_nbr	Numérico	Identificador único de um paciente	0%
Raça	race	Nominal	Valores: Caucasian, Asian, African American, Hispanic, and other	2%
Gênero	gender	Nominal	Valores: male, female, and unknown/invalid	0%
Idade	age	Nominal	Agrupado em intervalos de 10 anos: [0,10),[10,20),...,[90,100)	0%
Peso	weight	Numérico	Peso em libras	97%
Tipo de admissão	admission_type_id	Nominal	Identificador inteiro que corresponde a 9 valores distintos, por exemplo, emergência, urgência, eletivo, recém-nascido e não disponível	0%
Disposição de alta	discharge_disposition_id	Nominal	Identificador inteiro correspondente a 29 valores distintos, por exemplo, alta para casa, expirado e não disponível	0%
Fonte de admissão	admission_source_id	Nominal	Identificador inteiro correspondente a 21 valores distintos, por exemplo, encaminhamento médico, pronto-socorro e transferência de hospital	0%
Tempo no hospital	time_in_hospital	Numérico	Número inteiro de dias entre a admissão e a alta	0%
Código do pagador	payer_code	Nominal	Identificador inteiro correspondente a 23 valores distintos, por exemplo, Blue	52%

			Cross\Blue Shield, Medicare e autopagamento	
Especialidade médica	medical_specialty	Nominal	Identificador inteiro de uma especialidade do médico internante, correspondendo a 84 valores distintos, por exemplo, cardiologia, medicina interna, família\clínica geral e cirurgião	53%
Número de procedimentos laboratoriais	num_lab_procedures	Numérico	Número de exames laboratoriais realizados durante o encontro	0%
Número de procedimentos	num_procedures	Numérico	Número de procedimentos (exceto exames laboratoriais) realizados durante o encontro	0%
Número de medicações	num_medications	Numérico	Número de nomes genéricos distintos administrados durante o encontro	0%
Número de consultas ambulatoriais	number_outpatient	Numérico	Número de consultas ambulatoriais do paciente no ano anterior ao encontro	0%
Número de visitas de emergência	number_emergency	Numérico	Número de visitas de emergência do paciente no ano anterior ao encontro	0%
Número de consultas de internação	number_inpatient	Numérico	Número de visitas de internação do paciente no ano anterior ao encontro	0%
Diagnóstico 1	diag_1	Nominal	O diagnóstico primário (codificado como os três primeiros dígitos do CID9); 848 valores distintos	0%
Diagnóstico 2	diag_2	Nominal	Diagnóstico secundário (codificado como os três primeiros dígitos do CID9); 923 valores distintos	0%
Diagnóstico 3	diag_3	Nominal	Diagnóstico secundário adicional (codificado como os três primeiros dígitos do CID9); 954 valores distintos	1%

Número de diagnósticos	number_diagnoses	Numérico	Número de diagnósticos inseridos no sistema	0%
Resultado do teste de soro de glicose	max_glu_serum	Nominal	Indica o intervalo do resultado ou se o teste não foi realizado. Valores: “>200,” “>300,” “normal,” e “nenhum” se não medido	0%
Resultado do teste A1c	A1Cresult	Nominal	“>8” se o resultado foi maior que 8%, “>7” se o resultado foi maior que 7%, mas menor que 8%, “normal” se o resultado foi menor que 7% e “nenhum” se não foi medido.	0%
Mudança de medicamentos	change	Nominal	Indica se houve alteração nos medicamentos para diabéticos (seja posologia ou nome genérico). Valores: “mudar” e “sem mudança”	0%
Medicamentos para diabetes	diabetesMed	Nominal	Indica se foi prescrito algum medicamento para diabéticos. Valores: “sim” e “não”	0%
24 atributos para medicamentos		Nominal	Para os nomes genéricos: metformina, repaglinida, nateglinida, clorpropamida, glimepirida, acetohexamida, glipizida, gliburida, tolbutamida, pioglitazona, rosiglitazona, acarbose, miglitol, troglitazona, tolazamida, examida, sitagliptina, insulina, gliburida-metformina, glipizida-metformina, glimepirida -pioglitazona, metformina-rosiglitazona e metformina-pioglitazona, a característica indica se o medicamento foi prescrito ou houve alteração na posologia. Valores: “up” se a dosagem foi aumentada durante o encontro, “down” se a dosagem foi diminuída, “steady” se a dosagem não mudou e “no” se o medicamento não foi prescrito	0%

Readmitido	readmitted	Nominal	Dias para reinternação do paciente. Valores: “<30” se o paciente foi reinternado em menos de 30 dias, “>30” se o paciente foi reinternado em mais de 30 dias e “Não” para nenhum registro de reinternação.	0%

Fonte: Adaptado de Impact of HbA1c measurement on hospital readmission rates: analysis of 70,000 clinical database patient records (STRACK et al, 2014)

A.2. Importação das bibliotecas

In [1]:
```
import numpy as np
import pandas as pd

# Data Viz
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns

# Metricas
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, roc_curve, auc, confusion_matrix

# Métodos
# RL
from sklearn.linear_model import LogisticRegression

# RNA
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import Adam
from keras.metrics import AUC
```

In [2]:
```
diabetes = pd.read_csv("C:/Users/crash/Desktop/USP/MONOGRAFIA/BASE DE DADOS/diabetes_br.csv",encoding='latin-1',sep=';')
```

A.3. Análise exploratória inicial

Análise inicial exploratória

```
In [3]: diabetes.shape
Out[3]: (101766, 50)
```

```
In [4]: diabetes.head()
```

Out[4]:

	id do encontro	numero do paciente	raça	genero	idade	peso	tipo de admissao	disposicao de alta	fonte de admissao	dias no hospital	...	citoglipton	insulin	glyburide-metformin	glipizide-metformin	glimepiride pioglitazon
0	2278392	8222157	Caucasian	Female	[0-10)	?	6	25	1	1	...	No	No	No	No	N
1	149190	55629189	Caucasian	Female	[10-20)	?	1	1	7	3	...	No	Up	No	No	N
2	64410	86047875	AfricanAmerican	Female	[20-30)	?	1	1	7	2	...	No	No	No	No	N
3	500364	82442376	Caucasian	Male	[30-40)	?	1	1	7	2	...	No	Up	No	No	N
4	16680	42519267	Caucasian	Male	[40-50)	?	1	1	7	1	...	No	Steady	No	No	N

5 rows × 50 columns

```
In [5]: diabetes.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 101766 entries, 0 to 101765
Data columns (total 50 columns):
 #   Column                        Non-Null Count   Dtype
---  ------                        --------------   -----
 0   id_do_encontro                101766 non-null  int64
 1   numero_do_paciente            101766 non-null  int64
 2   raça                          101766 non-null  object
 3   genero                        101766 non-null  object
 4   idade                         101766 non-null  object
 5   peso                          101766 non-null  object
 6   tipo_de_admissao              101766 non-null  int64
 7   disposicao_de_alta            101766 non-null  int64
 8   fonte_de_admissao             101766 non-null  int64
 9   dias_no_hospital              101766 non-null  int64
 10  codigo_do_pagador             101766 non-null  object
 11  especialidade_medica          101766 non-null  object
 12  numero_de_procedimentos_laboratoriais  101766 non-null  int64
 13  numero_de_procedimentos       101766 non-null  int64
 14  numero_de_medicacoes          101766 non-null  int64
 15  consultas_ambulatoriais_ano_anterior  101766 non-null  int64
 16  consultas_emergencia_ano_anterior  101766 non-null  int64
 17  consultas_internacao_ano_anterior  101766 non-null  int64
 18  diagnostico_1                 101766 non-null  object
 19  diagnostico_2                 101766 non-null  object
 20  diagnostico_3                 101766 non-null  object
 21  numero_de_diagnosticos        101766 non-null  int64
 22  teste_de_soro_de_glicose      101766 non-null  object
 23  teste_ac1                     101766 non-null  object
 24  metformin                     101766 non-null  object
 25  repaglinide                   101766 non-null  object
 26  nateglinide                   101766 non-null  object
 27  chlorpropamide                101766 non-null  object
 28  glimepiride                   101766 non-null  object
 29  acetohexamide                 101766 non-null  object
 30  glipizide                     101766 non-null  object
 31  glyburide                     101766 non-null  object
 32  tolbutamide                   101766 non-null  object
 33  pioglitazone                  101766 non-null  object
 34  rosiglitazone                 101766 non-null  object
 35  acarbose                      101766 non-null  object
 36  miglitol                      101766 non-null  object
 37  troglitazone                  101766 non-null  object
 38  tolazamide                    101766 non-null  object
 39  examide                       101766 non-null  object
 40  citoglipton                   101766 non-null  object
 41  insulin                       101766 non-null  object
 42  glyburide-metformin           101766 non-null  object
 43  glipizide-metformin           101766 non-null  object
 44  glimepiride-pioglitazone      101766 non-null  object
 45  metformin-rosiglitazone       101766 non-null  object
 46  metformin-pioglitazone        101766 non-null  object
 47  mudanca_de_medicamento_de_diabetes  101766 non-null  object
 48  prescrito_medicamento_para_diabetes  101766 non-null  object
 49  readmitido                    101766 non-null  object
dtypes: int64(13), object(37)
memory usage: 38.8+ MB
```

In [6]:
```
diabetes.describe()
```

Out[6]:

	id do encontro	numero do paciente	tipo de admissao	disposicao de alta	fonte de admissao	dias no hospital	numero de procedimentos laboratoriais	numero de procedimentos	numero de medicacoes	cc
count	1.017660e+05	1.017660e+05	101766.000000	101766.000000	101766.000000	101766.000000	101766.000000	101766.000000	101766.000000	
mean	1.652016e+08	5.433040e+07	2.024006	3.715642	5.754437	4.395987	43.095641	1.339730	16.021844	
std	1.026403e+08	3.869636e+07	1.445403	5.280166	4.064081	2.985108	19.674362	1.705807	8.127566	
min	1.252200e+04	1.350000e+02	1.000000	1.000000	1.000000	1.000000	1.000000	0.000000	1.000000	
25%	8.496119e+07	2.341322e+07	1.000000	1.000000	1.000000	2.000000	31.000000	0.000000	10.000000	
50%	1.523890e+08	4.550514e+07	1.000000	1.000000	7.000000	4.000000	44.000000	1.000000	15.000000	
75%	2.302709e+08	8.754595e+07	3.000000	4.000000	7.000000	6.000000	57.000000	2.000000	20.000000	
max	4.438672e+08	1.895026e+08	8.000000	28.000000	25.000000	14.000000	132.000000	6.000000	81.000000	

In [7]:
```
# Confirmando as colunas que possuem valores '?'

resultado = diabetes.isin(['?']).any()

valores_true = resultado.loc[resultado]

total_com_interrogacao = 0

for coluna in diabetes.columns:
    if coluna in valores_true.index:

        quantidade_interrogacao = diabetes[coluna][diabetes[coluna] == '?'].count()

        total_com_interrogacao += quantidade_interrogacao

        porcentagem = (quantidade_interrogacao / len(diabetes)) * 100

        print(f"Coluna: {coluna}")
        print(f"Quantidade de '?' na coluna: {quantidade_interrogacao}")
        print(f"Porcentagem em relação ao total: {porcentagem:.2f}%")
        print(" ")
# Exiba o total de valores com '?'
print(f"Total de '?' em todo o DataFrame: {total_com_interrogacao}")
```

```
Coluna: raca
Quantidade de '?' na coluna: 2273
Porcentagem em relação ao total: 2.23%

Coluna: peso
Quantidade de '?' na coluna: 98569
Porcentagem em relação ao total: 96.86%

Coluna: codigo_do_pagador
Quantidade de '?' na coluna: 40256
Porcentagem em relação ao total: 39.56%

Coluna: especialidade_medica
Quantidade de '?' na coluna: 49949
Porcentagem em relação ao total: 49.08%

Coluna: diagnostico_1
Quantidade de '?' na coluna: 21
Porcentagem em relação ao total: 0.02%

Coluna: diagnostico_2
Quantidade de '?' na coluna: 358
Porcentagem em relação ao total: 0.35%

Coluna: diagnostico_3
Quantidade de '?' na coluna: 1423
Porcentagem em relação ao total: 1.40%

Total de '?' em todo o DataFrame: 192849
```

In [8]:
```
# Confirmando as colunas que possuem valores 'Unknown/Invalid'

resultado = diabetes.isin(['Unknown/Invalid']).any()

valores_true = resultado.loc[resultado]

total_com_interrogacao = 0

for coluna in diabetes.columns:
    if coluna in valores_true.index:

        quantidade_interrogacao = diabetes[coluna][diabetes[coluna] == 'Unknown/Invalid'].count()

        total_com_interrogacao += quantidade_interrogacao

        porcentagem = (quantidade_interrogacao / len(diabetes)) * 100

        print(f"Coluna: {coluna}")
        print(f"Quantidade de 'Unknown/Invalid' na coluna: {quantidade_interrogacao}")
        print(f"Porcentagem em relação ao total: {porcentagem:.2f}%")
        print(" ")
# Exiba o total de valores com 'Unknown/Invalid'
print(f"Total de '?' em todo o DataFrame: {total_com_interrogacao}")
```

```
Coluna: genero
Quantidade de 'Unknown/Invalid' na coluna: 3
Porcentagem em relação ao total: 0.00%

Total de '?' em todo o DataFrame: 3
```

In [9]:
```
#colunas que possuem apenas um valor

for column in diabetes.columns:
    if diabetes[column].nunique() == 1:
        print(column)
```

```
examide
citoglipton
```

In [10]:
```
# Distribuição de disposição de alta

diabetes_2 = diabetes.copy()

mapeamento = {
    1: 'Alta para casa',
    2: 'Alta/transf. para outro hospital de curto prazo',
    3: 'Alta/transf. para SNF',
    4: 'Alta/transf. para ICF',
    5: 'Alta/transf. para outra instituição de cuidados internos',
    6: 'Alta/transf. para casa com serviço de saúde domiciliar',
    7: 'Saída por conta própria (AMA)',
    8: 'Alta/transf. para casa com provedor de IV domiciliar',
    9: 'Admissão como paciente internado neste hospital',
    10: 'Recém-nascido transferido para outro hospital para cuidados neonatais',
    11: 'Óbito',
    12: 'Ainda paciente ou espera-se retorno para serviços ambulatoriais',
    13: 'Hospício / casa',
    14: 'Hospício / instituição médica',
    15: 'Alta/transf. dentro desta instituição para leito swing aprovado pelo Medicare',
    16: 'Alta/transf./encaminhamento para outra instituição para serviços ambulatoriais',
    17: 'Alta/transf./encaminhamento para esta instituição para serviços ambulatoriais',
    18: 'Nulo',
    19: 'Óbito em casa. Apenas Medicaid, hospício.',
    20: 'Óbito em uma instituição médica. Apenas Medicaid, hospício.',
    21: 'Óbito, local desconhecido. Apenas Medicaid, hospício.',
    22: 'Alta/transf. para outra instituição de reabilitação, incluindo unidades de reabilitação de hospital.',
    23: 'Alta/transf. para hospital de longa permanência.',
    24: 'Alta/transf. para uma instituição de enfermagem certificada pelo Medicaid, mas não certificada pelo Medicare.',
    25: 'Não mapeado',
    26: 'Desconhecido/Inválido',
    30: 'Alta/transf. para outro tipo de instituição de cuidados de saúde não definida em outro lugar',
    27: 'Alta/transf. para uma instalação de saúde federal.',
    28: 'Alta/transf./encaminhamento para um hospital psiquiátrico ou unidade distinta de hospital psiquiátrico',
    29: 'Alta/transf. para um Hospital de Acesso Crítico (CAH).'
}

contagem_valores = diabetes_2['disposicao_de_alta'].value_counts()
contagem_valores = contagem_valores.sort_index()

contagem_valores.index = contagem_valores.index.map(mapeamento)

plt.figure(figsize=(10, 6))

contagem_valores.plot(kind='bar')

# Adiciona valor em cima de cada barra
for i, v in enumerate(contagem_valores):
    plt.text(i, v, str(v), ha='center', va='bottom')

plt.title('Gráfico de Barras')
plt.xlabel('disposicao_de_alta')
plt.ylabel('Contagem')

plt.xticks(rotation=88)
plt.show()
```

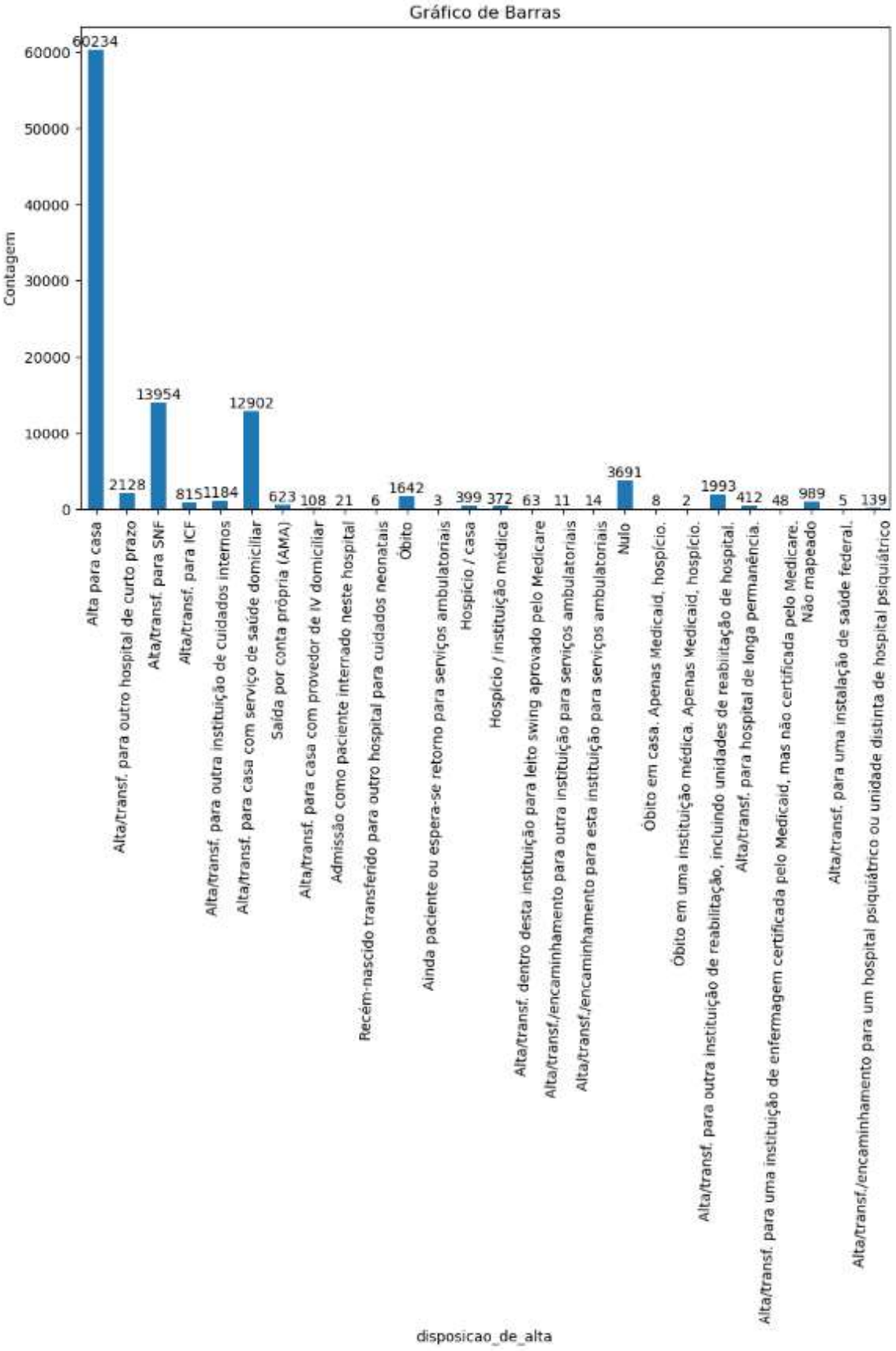
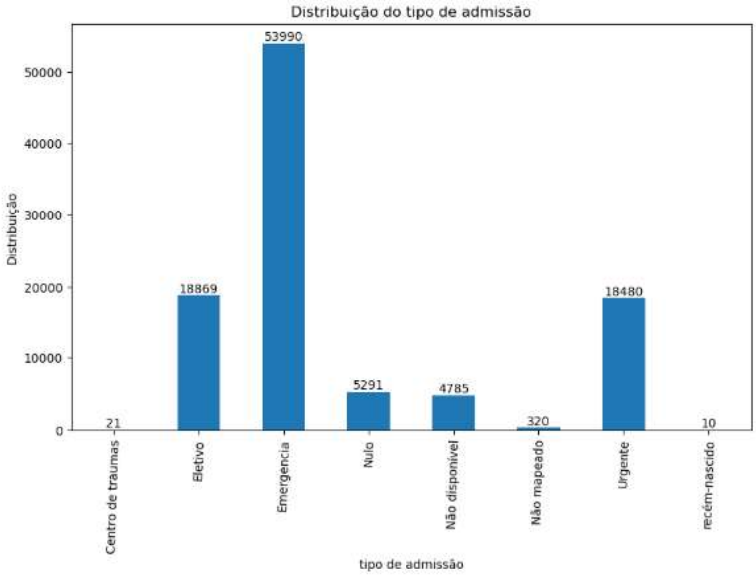
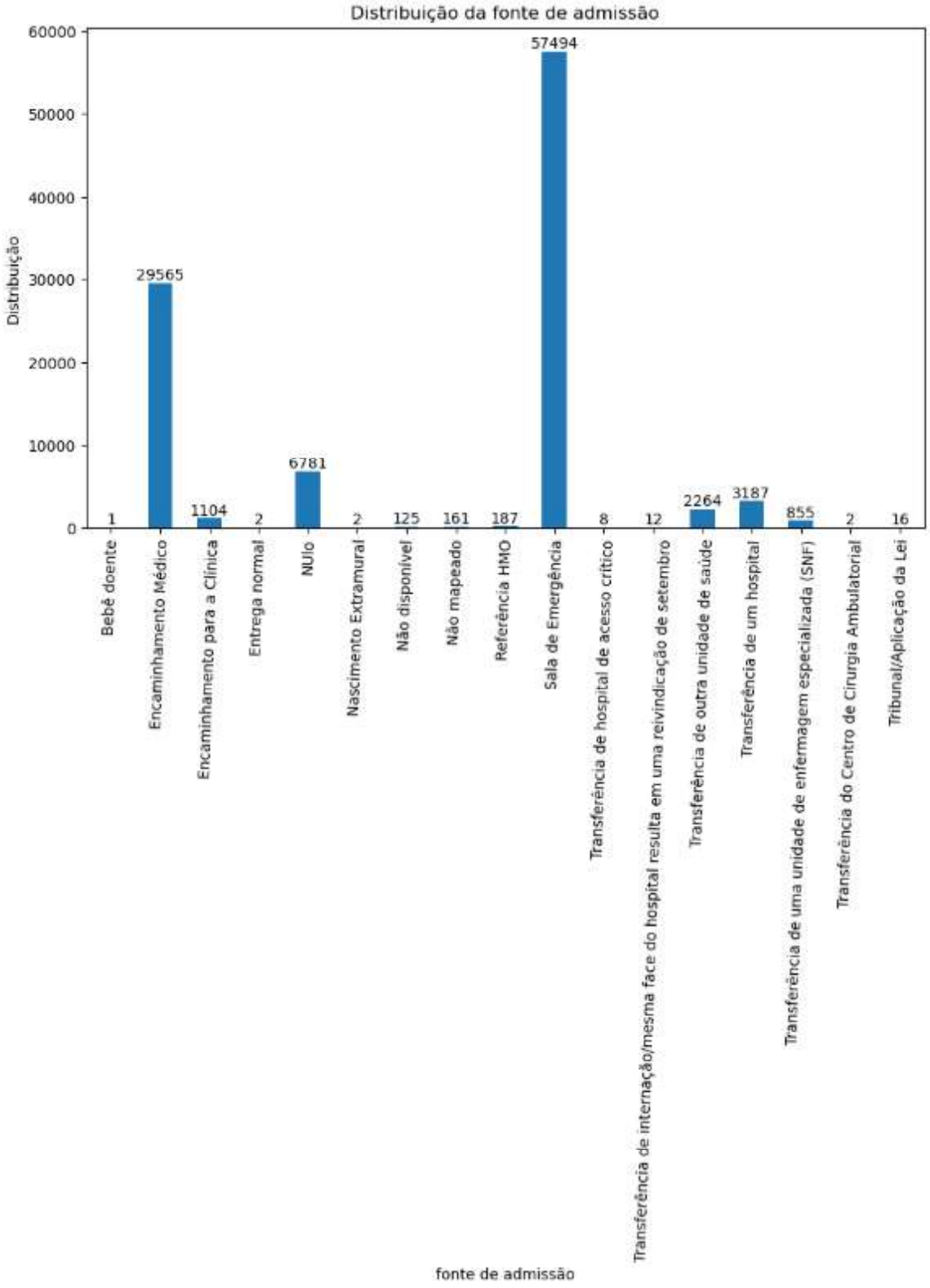


Gráfico de Barras
Contagem
disposicao_de_alta
60234
2128
13954
815
1184
12902
623
108
21
6
1642
3
399
372
63
11
14
3691
8
2
1993
412
48
989
5
139
Alta para casa
Alta/transf. para outro hospital de curto prazo
Alta/transf. para SNF
Alta/transf. para ICF
Alta/transf. para outra instituição de cuidados internos
Alta/transf. para casa com serviço de saúde domiciliar
Saída por conta própria (AMA)
Alta/transf. para casa com provedor de iV domiciliar
Admissão como paciente interrado neste hospital
Recém-nascido transferido para outro hospital para cuidados neonatais
Óbito
Ainda paciente ou espera-se retorno para serviços ambulatoriais
Hospício / casa
Hospício / instituição médica
Alta/transf. dentro desta instituição para leito swing aprovado pelo Medicare
Alta/transf./encaminhamento para outra instituição para serviços ambulatoriais
Alta/transf./encaminhamento para esta instituição para serviços ambulatoriais
Nulo
Óbito em casa. Apenas Medicaid, hospício.
Óbito em uma instituição médica. Apenas Medicaid, hospício.
Alta/transf. para outra instituição de reabilitação, incluindo unidades de reabilitação de hospital.
Alta/transf. para hospital de longa permanência.
Alta/transf. para uma instituição de enfermagem certificada pelo Medicaid, mas não certificada pelo Medicare.
Não mapeado
Alta/transf. para uma instalação de saúde federal.
Alta/transf./encaminhamento para um hospital psiquiátrico ou unidade distinta de hospital psiquiátrico

In [11]:
```
# Distribuição de tipo de admissão


diabetes_2 = diabetes.copy()

mapeamento = {
    1: 'Emergencia',
    2: 'Urgente',
    3: 'Eletivo',
    4: 'recém-nascido',
    5: 'Não disponivel',
    6: 'Nulo',
    7: 'Centro de traumas',
    8: 'Não mapeado'
}

def substituir_valor(valor):
    if valor in mapeamento:
        return mapeamento[valor]
    else:
        return valor

diabetes_2['tipo_de_admissao'] = diabetes_2['tipo_de_admissao'].apply(substituir_valor)
diabetes_2['tipo_de_admissao']

contagem_valores = diabetes_2['tipo_de_admissao'].value_counts()
contagem_valores = contagem_valores.sort_index()

plt.figure(figsize=(10, 6))

contagem_valores.plot(kind='bar')

# Adiciona valor em cima de cada barra
for i, v in enumerate(contagem_valores):
    plt.text(i, v, str(v), ha='center', va='bottom')

plt.title('Distribuição do tipo de admissão')
plt.xlabel('tipo de admissão')
plt.ylabel('Distribuição')

plt.xticks(rotation=88)

plt.show()
```



In [12]:
```
# Distribuição de fonte de admissão

diabetes_2 = diabetes.copy()

mapeamento = {
    1: 'Encaminhamento Médico',
    2: 'Encaminhamento para a Clínica',
    3: 'Referência HMO',
    4: 'Transferência de um hospital',
    5: 'Transferência de uma unidade de enfermagem especializada (SNF)',
    6: 'Transferência de outra unidade de saúde',
    7: 'Sala de Emergência',
    8: 'Tribunal/Aplicação da Lei',
    9: 'Não disponível',
    10: 'Transferência de hospital de acesso crítico',
    11: 'Entrega normal',
    12: 'Parto prematuro',
    13: 'Bebê doente',
    14: 'Nascimento Extramural',
    15: 'NUlo',
    17: 'NUlo',
    18: 'Transferência de outra agência de saúde domiciliar',
    19: 'Readmissão na mesma agência de saúde domiciliar',
    20: 'Não mapeado',
    21: 'Desconhecido/Inválido',
    22: 'Transferência de internação/mesma face do hospital resulta em uma reivindicação de setembro',
    23: 'Nascido dentro deste hospital',
    24: 'Nascido fora deste hospital',
    25: 'Transferência do Centro de Cirurgia Ambulatorial',
    26: 'Transferência do hospício'
}

def substituir_valor(valor):
    if valor in mapeamento:
        return mapeamento[valor]
    else:
        return valor

diabetes_2['fonte_de_admissao'] = diabetes_2['fonte_de_admissao'].apply(substituir_valor)
diabetes_2['fonte_de_admissao']

contagem_valores = diabetes_2['fonte_de_admissao'].value_counts()
contagem_valores = contagem_valores.sort_index()

plt.figure(figsize=(10, 6))

contagem_valores.plot(kind='bar')

# Adiciona valor em cima de cada barra
for i, v in enumerate(contagem_valores):
    plt.text(i, v, str(v), ha='center', va='bottom')

plt.title('Distribuição da fonte de admissão')
plt.xlabel('fonte de admissão')
plt.ylabel('Distribuição')

plt.xticks(rotation=88)

plt.show()
```


Distribuição da fonte de admissão
Distribuição
fonte de admissão
60000
50000
40000
30000
20000
10000
0
1
29565
1104
2
6781
2
125
161
187
57494
8
12
2264
3187
855
2
16
Bebê doente
Encaminhamento Médico
Encaminhamento para a Clínica
Entrega normal
NUlo
Nascimento Extramural
Não disponível
Não mapeado
Referência HMO
Sala de Emergência
Transferência de hospital de acesso crítico
Transferência de internação/mesma face do hospital resulta em uma reivindicação de setembro
Transferência de outra unidade de saúde
Transferência de um hospital
Transferência de uma unidade de enfermagem especializada (SNF)
Transferência do Centro de Cirurgia Ambulatorial
Tribunal/Aplicação da Lei

A.4. Deduplicação do número do paciente

LIMPEZA INICIAL DE DADOS

```
In [13]: # Deduplicação de dados
         # Um paciente (patient_nbr) pode ter tido mais de um encontro (encounter_id)
         # Neste caso, optou-se por realizar a deduplicação de dados, filtrando o primeiro encontro de cada paciente,
         # simulando tratamento realizado no artigo de (Beata Strack, 2014)

         diabetes_passo_1 = diabetes.drop_duplicates(subset = ['numero_do_paciente'], keep = 'first')
         diabetes_passo_1.shape

Out[13]: (71518, 50)
```

A.5. Remoção de linhas referente óbito

```
In [15]: # Remoção de linhas referente óbito, para evitar viés do algormito

         # 11: 'Óbito',
         # 13: 'Hospício / casa'
         # 14: 'Hospício / instituição médica'
         # 19: 'Óbito em casa. Apenas Medicaid, hospício.'
         # 20: 'Óbito em uma instituição médica. Apenas Medicaid, hospício.'
         # 21: 'Óbito, local desconhecido. Apenas Medicaid, hospício.'

         valores_filtro = [11,13,14,19,20,21]

         diabetes_passo_2 = diabetes_passo_1[~diabetes_passo_1['disposicao_de_alta'].isin(valores_filtro)]
         diabetes_passo_2.shape

Out[15]: (69973, 50)
```

A.6. Remoção das variáveis que não são úteis para classificação

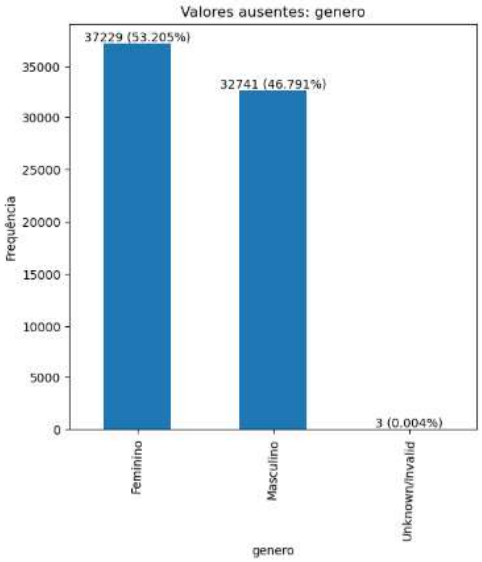
```
In [16]: # As variaveis de numero do cliente e encontro, não são uteis para classificação e foram removidas
         # colunas deletadas devido grande quantidade de valores invalidos:
             # peso: quantidade: 98569;  porcentagem em relação ao total: 96.8%
             # codigo_do_pagador: quantidade: 40256;  porcentagem em relação ao total: 39.5%

         diabetes_passo_3 = diabetes_passo_2.drop(['numero_do_paciente','id_do_encontro','peso','codigo_do_pagador'], axis=1)
         diabetes_passo_3
```

Out[16]:

	raça	genero	idade	tipo de admissao	disposicao de alta	fonte de admissao	dias no hospital	especialidade medica	numero de procedimentos laboratoriais	numero de procedimentos	...	cito
0	Caucasian	Female	[0-10)	6	25	1	1	Pediatrics-Endocrinology	41	0	...	
1	Caucasian	Female	[10-20)	1	1	7	3	?	59	0	...	
2	AfricanAmerican	Female	[20-30)	1	1	7	2	?	11	5	...	
3	Caucasian	Male	[30-40)	1	1	7	2	?	44	1	...	
4	Caucasian	Male	[40-50)	1	1	7	1	?	51	0	...	
...	...	...	...	...	...	...	...	...	...	...	...	...
101754	Caucasian	Female	[70-80)	1	1	7	9	?	50	2	...	
101755	Other	Female	[40-50)	1	1	7	14	?	73	6	...	
101756	Other	Female	[60-70)	1	1	7	2	?	46	6	...	
101758	Caucasian	Female	[80-90)	1	1	7	5	?	76	1	...	
101765	Caucasian	Male	[70-80)	1	1	7	6	?	13	3	...	

69973 rows × 46 columns

In [17]:
```
# Distribuição de genero

# Mapeia as legendas no eixo X
legenda_genero = {
    'Male': 'Masculino',
    'Female': 'Feminino'
}
valores = diabetes_passo_3['genero'].replace(legenda_genero)


plt.figure(figsize=(6, 6))
ax = valores.value_counts().plot(kind='bar')
plt.title('Valores ausentes: genero')
plt.xlabel('genero')
plt.ylabel('Frequência')

# Adiciona a porcentagem e o número em relação ao total
total = valores.count()
for i, (value, count) in enumerate(valores.value_counts().items()):
    porcentagem = count / total * 100
    ax.annotate(f'{count} ({porcentagem:.3f}%)', xy=(i, count), ha='center', va='bottom')


plt.xticks(rotation=88)
plt.show()
```



A.7. Preenche valores nulos com a moda

In [18]:
```
# Remoção de linhas com a porcentagem de quantidade baixa em relação ao total de linhas, referente à valores invalidos

linhas_deletadas = {}

diabetes_passo_4 = diabetes_passo_3.copy()

colunas_verificadas = ['genero']

for coluna in colunas_verificadas:
    linhas_anteriores = len(diabetes_passo_4)
    diabetes_passo_4 = diabetes_passo_4.loc[diabetes_passo_4[coluna] != 'Unknown/Invalid']
    linhas_deletadas[coluna] = linhas_anteriores - len(diabetes_passo_4)

# quantidade de linhas deletadas por coluna
for coluna, linhas_deletadas in linhas_deletadas.items():
    print(f"Quantidade de linhas deletadas para a coluna '{coluna}': {linhas_deletadas}")
```

```
Quantidade de linhas deletadas para a coluna 'genero': 3
```

In [19]:
```
# Preenche os valores nulos na coluna 'raca' com a moda
# tratamento para valores ausentes, replicando o processo utilizado no trabalho de (Félix Tamin1, Ni Made Satvika Iswari2, 2017)

diabetes_passo_5 = diabetes_passo_4.copy()

diabetes_passo_5['raca'] = diabetes_passo_5['raca'].replace('?', pd.NA)
moda_raca = diabetes_passo_5['raca'].mode().iloc[0]
diabetes_passo_5['raca'].fillna(moda_raca, inplace=True)
```

In [20]:
```
# Preenche os valores nulos na coluna 'especialidade_medica' com a moda

diabetes_passo_5['especialidade_medica'] = diabetes_passo_5['especialidade_medica'].replace('?', pd.NA)
moda_raca = diabetes_passo_5['especialidade_medica'].mode().iloc[0]
diabetes_passo_5['especialidade_medica'].fillna(moda_raca, inplace=True)
```

In [21]:
```
# Preenche os valores nulos na coluna 'diagnóstico 1 (primária)' com a moda

diabetes_passo_5['diagnostico_1'] = diabetes_passo_5['diagnostico_1'].replace('?', pd.NA)
moda_raca = diabetes_passo_5['diagnostico_1'].mode().iloc[0]
diabetes_passo_5['diagnostico_1'].fillna(moda_raca, inplace=True)
```

A.8. Análise de especialidade médica

In [22]:
```
# Agrupamento de especialidade_medica

# Count da coluna "especialidade_medica"
contagem_valores = diabetes_passo_5['especialidade_medica'].value_counts()

df_contagem = pd.DataFrame({'Especialidade médica': contagem_valores.index, 'Distribuição': contagem_valores.values})

df_contagem = df_contagem.sort_values(by='Distribuição', ascending=False) # Ordena

with pd.option_context('display.max_rows', None, 'display.max_columns', None):
    print(df_contagem)
```

	Especialidade médica	Distribuição
0	InternalMedicine	44278
1	Family/GeneralPractice	4978
2	Emergency/Trauma	4393
3	Cardiology	4206
4	Surgery-General	2205
5	Orthopedics	1128
6	Orthopedics-Reconstructive	1041
7	Radiologist	821
8	Nephrology	797
9	Pulmonology	637
10	Psychiatry	613
11	ObstetricsandGynecology	593
12	Urology	530
13	Surgery-Cardiovascular/Thoracic	488
14	Surgery-Neuro	404
15	Gastroenterology	383
16	Surgery-Vascular	359
17	Oncology	205
18	Pediatrics	195
19	PhysicalMedicineandRehabilitation	194
20	Neurology	167
21	Pediatrics-Endocrinology	147
22	Otolaryngology	110
23	Hematology/Oncology	109
24	Endocrinology	97
25	Surgery-Thoracic	91
26	Surgery-Cardiovascular	85
27	Pediatrics-CriticalCare	73
28	Podiatry	63
29	Gynecology	54
30	Psychology	53
31	Surgeon	40
32	Radiology	38
33	Osteopath	37
34	Hospitalist	36
35	Ophthalmology	35
36	Hematology	31
38	Surgery-Plastic	29
37	InfectiousDiseases	29
39	SurgicalSpecialty	26
40	Obsterics&Gynecology-GynecologicOnco	18
41	Obstetrics	17
42	Anesthesiology-Pediatric	13
43	Rheumatology	10
44	OutreachServices	9
45	Surgery-Colon&Rectal	9
46	Surgery-Maxillofacial	8
47	Pediatrics-Neurology	7
48	Cardiology-Pediatric	7
49	Anesthesiology	7
50	PhysicianNotFound	7
51	Endocrinology-Metabolism	7
55	AllergyandImmunology	6
56	Psychiatry-Child/Adolescent	6
53	Pathology	6
54	Surgery-Pediatric	6
52	Pediatrics-Pulmonology	6
57	Dentistry	4
58	DCPTEAM	4
59	Pediatrics-EmergencyMedicine	3
60	Pediatrics-Hematology-Oncology	3
61	Neurophysiology	1
62	Perinatology	1
63	Surgery-PlasticwithinHeadandNeck	1
64	Speech	1
65	SportsMedicine	1
66	Dermatology	1
67	Proctology	1
68	Psychiatry-Addictive	1
69	Resident	1

A.9. Remoção de linhas com valores únicos

In [24]:
```
# Remoção de variaveis com valor unico
for column in diabetes_passo_5.columns:
    if diabetes_passo_5[column].nunique() == 1:
        print('Atributo deletado, pois possui apenas um valor: ', column)

diabetes_passo_6 = diabetes_passo_5.loc[:, diabetes_passo_5.nunique() != 1]
diabetes_passo_6.shape
```

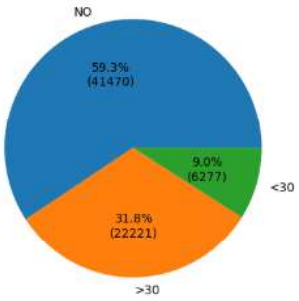
```
Atributo deletado, pois possui apenas um valor:  examide
Atributo deletado, pois possui apenas um valor:  citoglipton
Atributo deletado, pois possui apenas um valor:  glimepiride-pioglitazone
```

Out[24]:
```
(69970, 43)
```

A.10. Distribuição de readmissão

In [25]:
```
# Distribuição de readmissão

dados = diabetes_passo_6['readmitido'].value_counts()

plt.pie(dados, labels=dados.index, autopct=lambda pct: f'{pct:.1f}%\n({int(pct/100 * len(diabetes_passo_6))})')
```

Out[25]:
```
([<matplotlib.patches.Wedge at 0x1f03d2debe0>,
  <matplotlib.patches.Wedge at 0x1f03d3ed340>,
  <matplotlib.patches.Wedge at 0x1f03d3eda60>],
 [Text(-0.3158286761339514, 1.053685079707424, 'NO'),
  Text(0.01032196441610371, -1.099951570320527, '>30'),
  Text(1.0566022717877108, -0.3059275065322641, '<30')],
 [Text(-0.17227018698215527, 0.5747373162385867, '59.3%\n(41470)'),
  Text(0.005630162408783841, -0.5999735838111964, '31.8%\n(22221)'),
  Text(0.5763285118842059, -0.16686954901759862, '9.0%\n(6277)')])
```



In [26]:
```
# Optou-se por converter para variavel binaria
# Os valores >30 e <30 (pacientes readmitidos) foram convertidos para 1, e não readmitidos, para 0

diabetes_passo_7 = diabetes_passo_6.copy()
diabetes_passo_7['readmitido'] = diabetes_passo_7['readmitido'].apply(lambda x: 0 if x == "NO" else 1)
diabetes_passo_7
```

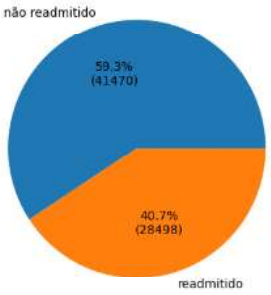
Out[26]:

	raça	genero	idade	tipo de admissao	disposicao de alta	fonte de admissao	dias no hospital	especialidade medica	numero de procedimentos laboratoriais	numero de procedimentos	...	trog
0	Caucasian	Female	[0-10)	6	25	1	1	Outros	41	0	...	
1	Caucasian	Female	[10-20)	1	1	7	3	Medicina interna	59	0	...	
2	AfricanAmerican	Female	[20-30)	1	1	7	2	Medicina interna	11	5	...	
3	Caucasian	Male	[30-40)	1	1	7	2	Medicina interna	44	1	...	
4	Caucasian	Male	[40-50)	1	1	7	1	Medicina interna	51	0	...	
...	...	...	...	...	...	...	...	...	...	...	...	...
101754	Caucasian	Female	[70-80)	1	1	7	9	Medicina interna	50	2	...	
101755	Other	Female	[40-50)	1	1	7	14	Medicina interna	73	6	...	
101756	Other	Female	[60-70)	1	1	7	2	Medicina interna	46	6	...	
101758	Caucasian	Female	[80-90)	1	1	7	5	Medicina interna	76	1	...	
101765	Caucasian	Male	[70-80)	1	1	7	6	Medicina interna	13	3	...	

69970 rows × 43 columns

In [27]:
```
# Distribuição de readmissão
dados = diabetes_passo_7['readmitido'].value_counts()

dados_labels = dados.index.map({0: 'não readmitido', 1: 'readmitido'})

plt.pie(dados, labels=dados_labels, autopct=lambda pct: f'{pct:.1f}%\n({int(pct/100 * len(diabetes_passo_7))})')
```

Out[27]:
```
([<matplotlib.patches.Wedge at 0x1f03d43da00>,
  <matplotlib.patches.Wedge at 0x1f03d44c190>],
 [Text(-0.3158286761339514, 1.0536850797707424, 'não readmitido'),
  Text(0.3158285774808325, -1.0536851093407527, 'readmitido')],
 [Text(-0.17227018698215527, 0.5747373162385867, '59.3%\n(41470)'),
  Text(0.17227013317136314, -0.5747373323676832, '40.7%\n(28498)')])
```



A.11. Agrupamento de dados

In [23]:
```
# Agrupamento de especialidade_medica

conditions = [
    diabetes_passo_5['especialidade_medica'].str.contains("Cardiology", case=False, na=False),
    diabetes_passo_5['especialidade_medica'].str.contains("GeneralPractice", case=False, na=False),
    diabetes_passo_5['especialidade_medica'].str.contains("InternalMedicine", case=False, na=False),
    diabetes_passo_5['especialidade_medica'].str.contains("Surgery", case=False, na=False),
    diabetes_passo_5['especialidade_medica'].str.contains("Emergency", case=False, na=False)
]

choices = [
    "Cardiologia",
    "Prática geral",
    "Medicina interna",
    "Cirurgia",
    "Emergência"
]

default_choice = "Outros"

diabetes_passo_5['especialidade_medica'] = np.select(conditions, choices, default=default_choice)

contagem = diabetes_passo_5['especialidade_medica'].value_counts()

for valor, quantidade in contagem.items():
    print(f'Valor: {valor}, Contagem: {quantidade}')
```

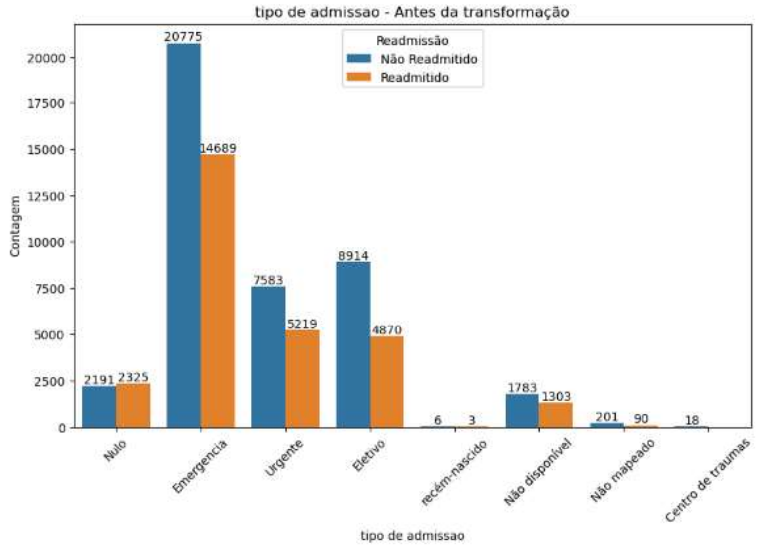
```
Valor: Medicina interna, Contagem: 44278
Valor: Outros, Contagem: 8420
Valor: Prática geral, Contagem: 4978
Valor: Emergência, Contagem: 4396
Valor: Cardiologia, Contagem: 4213
Valor: Cirurgia, Contagem: 3685
```

In [24]:
```
# Remoção de variáveis com valor único
for column in diabetes_passo_5.columns:
    if diabetes_passo_5[column].nunique() == 1:
        print('Atributo deletado, pois possui apenas um valor: ', column)

diabetes_passo_6 = diabetes_passo_5.loc[:, diabetes_passo_5.nunique() != 1]
diabetes_passo_6.shape
```

```
Atributo deletado, pois possui apenas um valor:  examide
Atributo deletado, pois possui apenas um valor:  citoglipton
Atributo deletado, pois possui apenas um valor:  glimepiride-pioglitazone
```

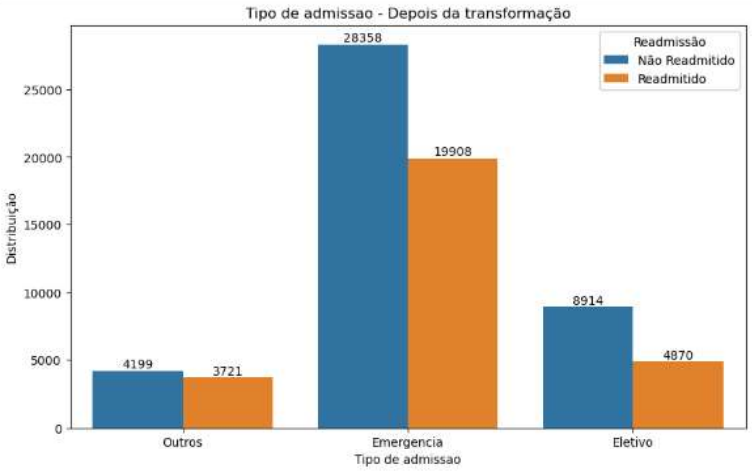
Out[24]:
```
(69978, 43)
```

In [28]:
```
# Distribuição do tipo de admissão categorizado por readmissão

# Lista de substituição dos valores

substituicoes = {
    1: 'Emergencia',
    2: 'Urgente',
    3: 'Eletivo',
    4: 'recém-nascido',
    5: 'Não disponivel',
    6: 'Nulo',
    7: 'Centro de traumas',
    8: 'Não mapeado'
}


diabetes_passo_7_temp = diabetes_passo_7.copy()

diabetes_passo_7_temp['tipo_de_admissao'] = diabetes_passo_7_temp['tipo_de_admissao'].replace(substituicoes)

plt.figure(figsize=(10, 6))
ax = sns.countplot(data=diabetes_passo_7_temp, x='tipo_de_admissao', hue='readmitido')

for p in ax.patches:
    ax.annotate(format(p.get_height(), '.0f'), (p.get_x() + p.get_width() / 2., p.get_height()), ha = 'center', va = 'center', xytext = (0, 5), textcoords = 'offset points')

plt.title('tipo de admissao - Antes da transformação')
plt.xlabel('tipo de admissao')
plt.ylabel('Contagem')

legenda = plt.legend(title='Readmissão', labels=['Não Readmitido', 'Readmitido'])
plt.gca().add_artist(legenda)

plt.xticks(rotation=45)
plt.show()
```



In [29]:
```
# Agrupamento do tipo de admissão

diabetes_passo_8 = diabetes_passo_7.copy()
diabetes_passo_8

def substituir_valor(valor):
    if valor == 1:
        return 'Emergencia'
    elif valor == 2:
        return 'Emergencia'
    elif valor == 3:
        return 'Eletivo'
    elif valor == 4:
        return 'Outros'
    elif valor == 5:
        return 'Outros'
    elif valor == 6:
        return 'Outros'
    elif valor == 7:
        return 'Outros'
    elif valor == 8:
        return 'Outros'
    else:
        return valor

diabetes_passo_8['tipo_de_admissao'] = diabetes_passo_8['tipo_de_admissao'].apply(substituir_valor)
```

In [30]:
```
# Tipo de admissão

plt.figure(figsize=(10, 6))
ax = sns.countplot(data=diabetes_passo_8, x='tipo_de_admissao', hue='readmitido')

for p in ax.patches:
    ax.annotate(format(p.get_height(), '.0f'), (p.get_x() + p.get_width() / 2., p.get_height()), ha = 'center', va = 'center', xytext = (0, 5), textcoords = 'offset points')

plt.title('Tipo de admissao - Depois da transformação')
plt.xlabel('Tipo de admissao')
plt.ylabel('Distribuição')

legenda = plt.legend(title='Readmissão', labels=['Não Readmitido', 'Readmitido'])
plt.gca().add_artist(legenda)

plt.show()
```

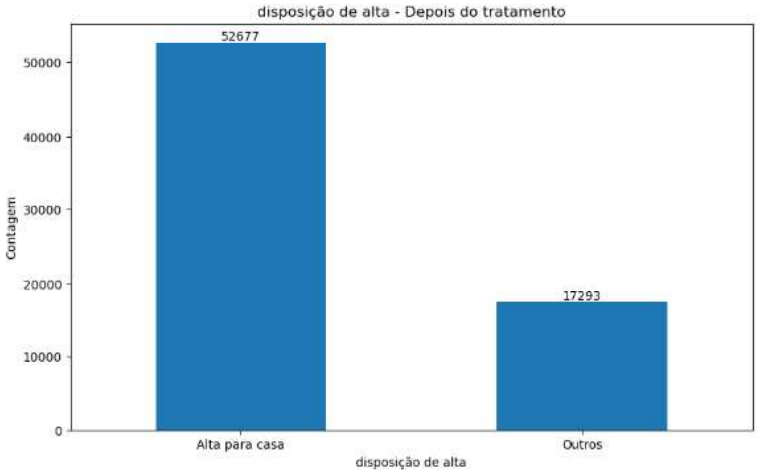


In [31]:
```
import pandas as pd

# Mapeamento de valores
mapeamento = {
    1: 'Alta para casa',
    2: 'Alta/transf. para outro hospital de curto prazo',
    3: 'Alta/transf. para SNF',
    4: 'Alta/transf. para ICF',
    5: 'Alta/transf. para outra instituição de cuidados internos',
    6: 'Alta/transf. para casa com serviço de saúde domiciliar',
    7: 'Saída por conta própria (AMA)',
    8: 'Alta/transf. para casa com provedor de IV domiciliar',
    9: 'Admissão como paciente internado neste hospital',
    10: 'Recém-nascido transferido para outro hospital para cuidados neonatais',
    11: 'Óbito',
    12: 'Ainda paciente ou espera-se retorno para serviços ambulatoriais',
    13: 'Hospício / casa',
    14: 'Hospício / instituição médica',
    15: 'Alta/transf. dentro desta instituição para leito swing aprovado pelo Medicare',
    16: 'Alta/transf./encaminhamento para outra instituição para serviços ambulatoriais',
    17: 'Alta/transf./encaminhamento para esta instituição para serviços ambulatoriais',
    18: 'Nulo',
    19: 'Óbito em casa. Apenas Medicaid, hospício.',
    20: 'Óbito em uma instituição médica. Apenas Medicaid, hospício.',
    21: 'Óbito, local desconhecido. Apenas Medicaid, hospício.',
    22: 'Alta/transf. para outra instituição de reabilitação, incluindo unidades de reabilitação de hospital.',
    23: 'Alta/transf. para hospital de longa permanência.',
    24: 'Alta/transf. para uma instituição de enfermagem certificada pelo Medicaid, mas não certificada pelo Medicare.',
    25: 'Nulo',
    26: 'Nulo',
    30: 'Alta/transf. para outro tipo de instituição de cuidados de saúde não definida em outro lugar',
    27: 'Alta/transf. para uma instalação de saúde federal.',
    28: 'Alta/transf./encaminhamento para um hospital psiquiátrico ou unidade distinta de hospital psiquiátrico',
    29: 'Alta/transf. para um Hospital de Acesso Crítico (CAH).'
}

# Contagem dos valores da coluna 'disposicao_de_alta'
contagem_valores = diabetes_passo_8['disposicao_de_alta'].value_counts().sort_index()

# Mapeie os índices dos valores
contagem_valores.index = contagem_valores.index.map(mapeamento)

# Crie um DataFrame com os valores e suas contagens
df_contagem = pd.DataFrame({'Disposição de Alta': contagem_valores.index, 'Contagem': contagem_valores.values})

# Exiba o DataFrame completo sem o índice
with pd.option_context('display.max_rows', None):
    print(df_contagem)
```

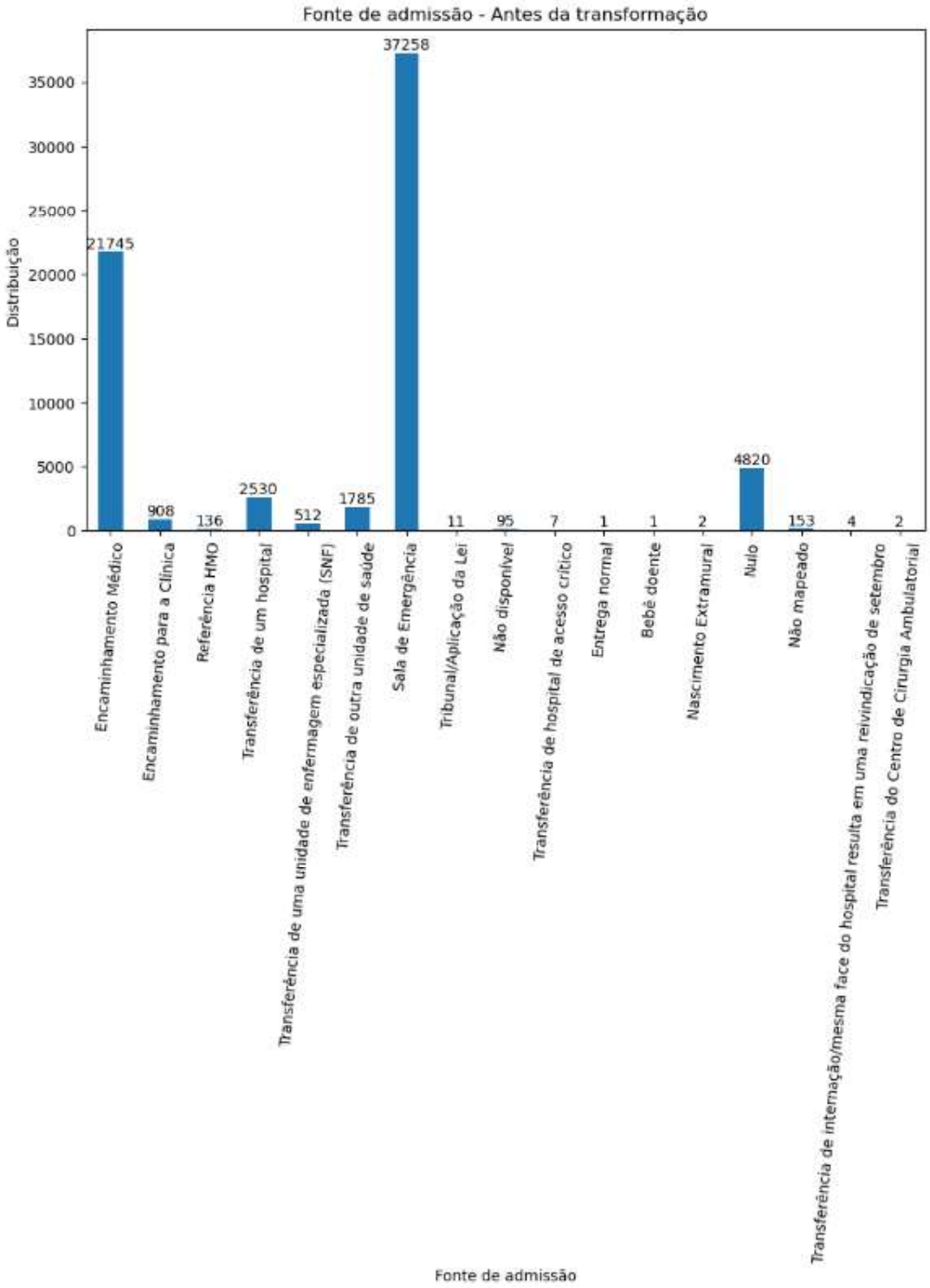
```
                            Disposição de Alta  Contagem
0                                  Alta para casa     44315
1     Alta/transf. para outro hospital de curto prazo      1539
2                               Alta/transf. para SNF      8784
3                               Alta/transf. para ICF       541
4   Alta/transf. para outra instituição de cuidado...       913
5   Alta/transf. para casa com serviço de saúde do...      8289
6                     Saída por conta própria (AMA)       409
7   Alta/transf. para casa com provedor de IV domi...        73
8     Admissão como paciente internado neste hospital         9
9   Recém-nascido transferido para outro hospital ...         6
10  Ainda paciente ou espera-se retorno para servi...         2
11  Alta/transf. dentro desta instituição para lei...        40
12  Alta/transf./encaminhamento para outra institu...         3
13  Alta/transf./encaminhamento para esta instituí...         8
14                                             Nulo      2474
15  Alta/transf. para outra instituição de reabili...      1409
16   Alta/transf. para hospital de longa permanência.       260
17  Alta/transf. para uma instituição de enfermage...        25
18                                             Nulo       778
19  Alta/transf. para uma instalação de saúde fede...         3
20  Alta/transf./encaminhamento para um hospital p...        90
```

In [32]:
```
# Agrupamento de disposição de alta

diabetes_passo_9 = diabetes_passo_8.copy()

def substituir_valor(valor):
    if valor == 1:
        return 'Alta para casa'
    elif valor == 6:
        return 'Alta para casa'
    elif valor == 8:
        return 'Alta para casa'
    elif valor == 13:
        return 'Alta para casa'
    elif valor == 19:
        return 'Alta para casa'
    else:
        return 'Outros'

diabetes_passo_9['disposicao_de_alta'] = diabetes_passo_9['disposicao_de_alta'].apply(substituir_valor)
```

In [33]:
```
# disposicao de alta

contagem_valores = diabetes_passo_9['disposicao_de_alta'].value_counts()
contagem_valores = contagem_valores.sort_index()

plt.figure(figsize=(10, 6))

contagem_valores.plot(kind='bar')

for i, v in enumerate(contagem_valores):
    plt.text(i, v, str(v), ha='center', va='bottom')

plt.title('disposição de alta - Depois do tratamento')
plt.xlabel('disposição de alta')
plt.ylabel('Contagem')

plt.xticks(rotation=0)
plt.show()
```



In [34]:
```
# Agrupamento da Fonte de admissão

mapeamento = {
    1: 'Encaminhamento Médico',
    2: 'Encaminhamento para a Clínica',
    3: 'Referência HMO',
    4: 'Transferência de um hospital',
    5: 'Transferência de uma unidade de enfermagem especializada (SNF)',
    6: 'Transferência de outra unidade de saúde',
    7: 'Sala de Emergência',
    8: 'Tribunal/Aplicação da Lei',
    9: 'Não disponível',
    10: 'Transferência de hospital de acesso crítico',
    11: 'Entrega normal',
    12: 'Parto prematuro',
    13: 'Bebê doente',
    14: 'Nascimento Extramural',
    15: 'Nulo',
    17: 'Nulo',
    18: 'Transferência de outra agência de saúde domiciliar',
    19: 'Readmissão na mesma agência de saúde domiciliar',
    20: 'Não mapeado',
    21: 'Nulo',
    22: 'Transferência de internação/mesma face do hospital resulta em uma reivindicação de setembro',
    23: 'Nascido dentro deste hospital',
    24: 'Nascido fora deste hospital',
    25: 'Transferência do Centro de Cirurgia Ambulatorial',
    26: 'Transferência do hospício'
}

contagem_valores = diabetes_passo_9['fonte_de_admissao'].value_counts()
contagem_valores = contagem_valores.sort_index()

contagem_valores.index = contagem_valores.index.map(mapeamento)

plt.figure(figsize=(10, 6))

contagem_valores.plot(kind='bar')

for i, v in enumerate(contagem_valores):
    plt.text(i, v, str(v), ha='center', va='bottom')

plt.title('Fonte de admissão - Antes da transformação')
plt.xlabel('Fonte de admissão')
plt.ylabel('Distribuição')

plt.xticks(rotation=85)
plt.show()
```


Fonte de admissão - Antes da transformação
Distribuição
0
5000
10000
15000
20000
25000
30000
35000
21745
908
136
2530
512
1785
37258
11
95
7
1
1
2
4820
153
4
2
Encaminhamento Médico
Encaminhamento para a Clínica
Referência HMO
Transferência de um hospital
Transferência de uma unidade de enfermagem especializada (SNF)
Transferência de outra unidade de saúde
Sala de Emergência
Tribunal/Aplicação da Lei
Não disponível
Transferência de hospital de acesso crítico
Entrega normal
Bebê doente
Nascimento Extramural
Nulo
Não mapeado
Transferência de internação/mesma face do hospital resulta em uma reivindicação de setembro
Transferência do Centro de Cirurgia Ambulatorial
Fonte de admissão

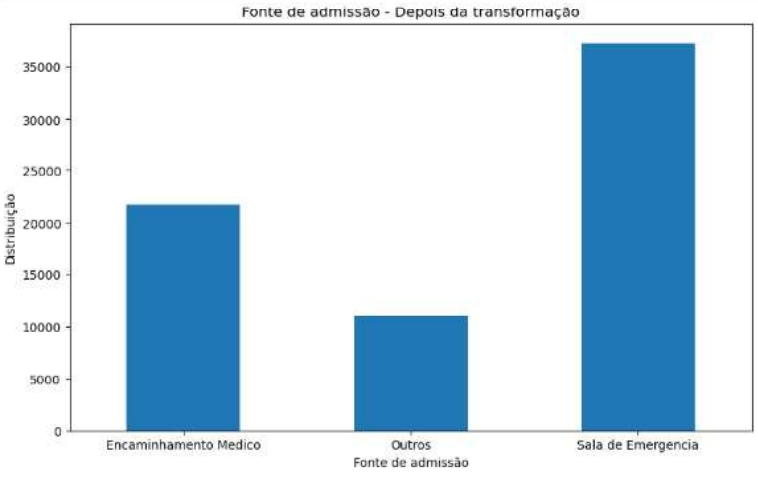
In [35]:
```
# Agrupamento da Fonte de admissão

diabetes_passo_10 = diabetes_passo_9.copy()

diabetes_passo_10.loc[diabetes_passo_10['fonte_de_admissao'] == 1, 'fonte_de_admissao'] = 'Encaminhamento Medico'

# Verifica se o valor da coluna é igual a 2 e atribui o novo valor 'teste_2'
diabetes_passo_10.loc[diabetes_passo_10['fonte_de_admissao'] == 7, 'fonte_de_admissao'] = 'Sala de Emergencia'

# Atribui o novo valor 'testinho' para os demais valores da coluna
diabetes_passo_10.loc[(diabetes_passo_10['fonte_de_admissao'] != 'Encaminhamento Medico') & (diabetes_passo_10['fonte_de_admissao'] != 'Sala de Emergencia'), 'fonte_de_admissao'] =
diabetes_passo_10
```

Out[35]:

	raça	genero	idade	tipo de admissao	disposicao de alta	fonte de admissao	dias no hospital	especialidade medica	numero de procedimentos laboratoriais	numero de procedimentos	...	trog
0	Caucasian	Female	[0-10)	Outros	Outros	Encaminhamento Medico	1	Outros	41	0	...	
1	Caucasian	Female	[10-20)	Emergencia	Alta para casa	Sala de Emergencia	3	Medicina interna	59	0	...	
2	AfricanAmerican	Female	[20-30)	Emergencia	Alta para casa	Sala de Emergencia	2	Medicina interna	11	5	...	
3	Caucasian	Male	[30-40)	Emergencia	Alta para casa	Sala de Emergencia	2	Medicina interna	44	1	...	
4	Caucasian	Male	[40-50)	Emergencia	Alta para casa	Sala de Emergencia	1	Medicina interna	51	0	...	
...	...	...	...	...	...	...	...	...	...	...	...	...
101754	Caucasian	Female	[70-80)	Emergencia	Alta para casa	Sala de Emergencia	9	Medicina interna	50	2	...	
101755	Other	Female	[40-50)	Emergencia	Alta para casa	Sala de Emergencia	14	Medicina interna	73	6	...	
101756	Other	Female	[60-70)	Emergencia	Alta para casa	Sala de Emergencia	2	Medicina interna	46	6	...	
101758	Caucasian	Female	[80-90)	Emergencia	Alta para casa	Sala de Emergencia	5	Medicina interna	76	1	...	
101765	Caucasian	Male	[70-80)	Emergencia	Alta para casa	Sala de Emergencia	6	Medicina interna	13	3	...	

69970 rows × 43 columns

In [36]:
```
# Distribuição da Fonte de admissão

contagem_valores = diabetes_passo_10['fonte_de_admissao'].value_counts()
contagem_valores = contagem_valores.sort_index()

plt.figure(figsize=(10, 6))

contagem_valores.plot(kind='bar')

plt.title('Fonte de admissão - Depois da transformação')
plt.xlabel('Fonte de admissão')
plt.ylabel('Distribuição')

plt.xticks(rotation=0)
plt.show()
```



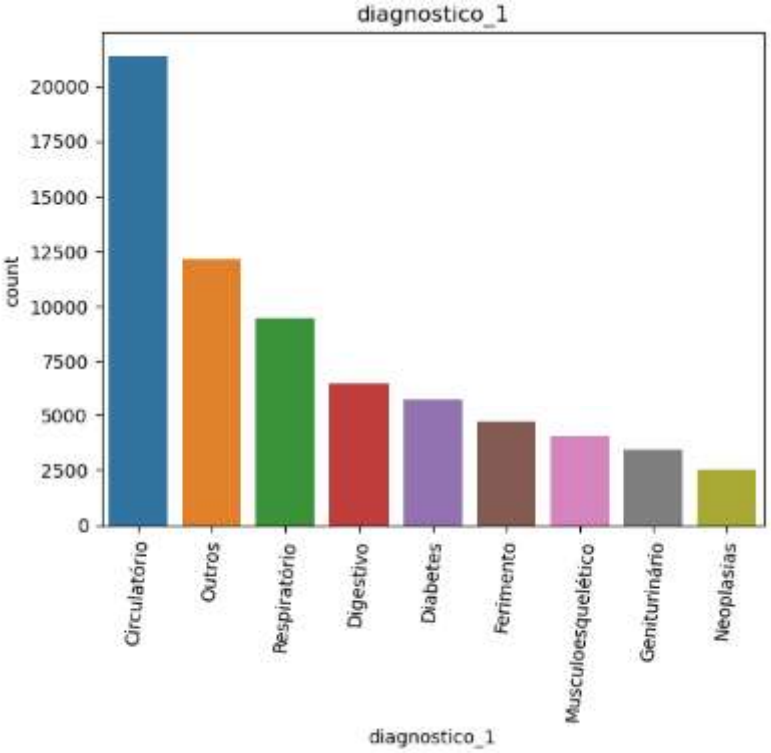
A.12. Agrupamento dos atributos de diagnosticos

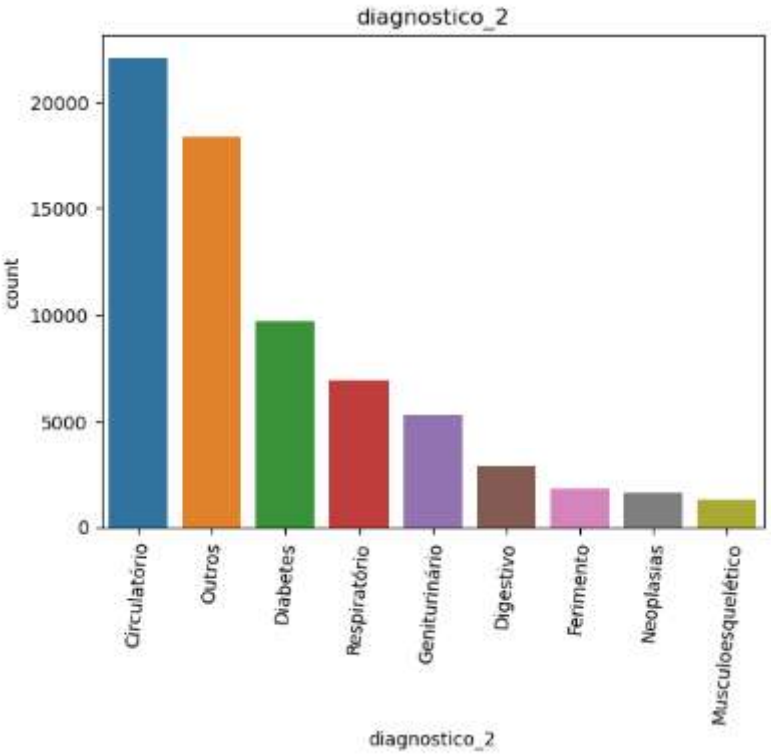
In [37]:
```
# Realizado agrupamento de diagnóstico
# Replicando o procedimento utilizado em pesquisas relacionadas (Beata Strack,2014)

def map_primary_diagnosis(row, column_name):
    value = row[column_name]
    if any(char in value for char in ['V', 'E']):
        return "Outros"
    elif value == "250" or value.startswith("250."):
        return "Diabetes"
    elif (value.isdigit() and (390 <= int(float(value)) <= 459)) or value == "785":
        return "Circulatório"
    elif (value.isdigit() and (460 <= int(float(value)) <= 519)) or value == "786":
        return "Respiratório"
    elif (value.isdigit() and (520 <= int(float(value)) <= 579)) or value == "787":
        return "Digestivo"
    elif (value.isdigit() and (580 <= int(float(value)) <= 629)) or value == "788":
        return "Geniturinário"
    elif (value.isdigit() and (140 <= int(float(value)) <= 239)):
        return "Neoplasias"
    elif (value.isdigit() and (710 <= int(float(value)) <= 739)):
        return "Musculoesquelético"
    elif (value.isdigit() and (800 <= int(float(value)) <= 999)):
        return "Ferimento"
    else:
        return "Outros"

diabetes_passo_10['diagnostico_1'] = diabetes_passo_10.apply(map_primary_diagnosis, column_name='diagnostico_1', axis=1)
diabetes_passo_10['diagnostico_2'] = diabetes_passo_10.apply(map_primary_diagnosis, column_name='diagnostico_2', axis=1)
diabetes_passo_10['diagnostico_3'] = diabetes_passo_10.apply(map_primary_diagnosis, column_name='diagnostico_3', axis=1)

diabetes_passo_11 = diabetes_passo_10.copy()

pd.set_option('display.max_columns', None)
diabetes_passo_11.head()
```

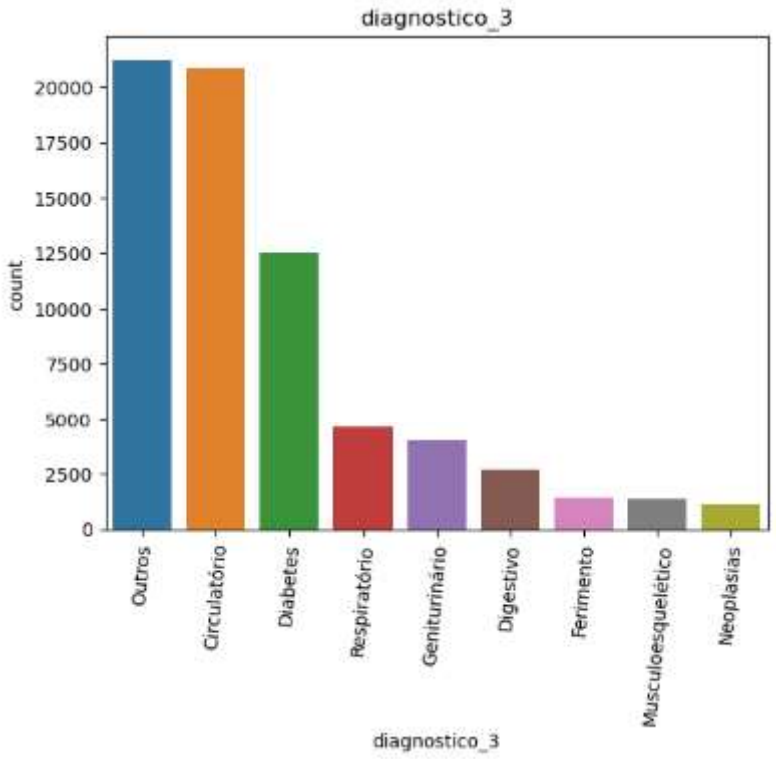
Out[37]:

	raça	genero	idade	tipo de admissao	disposicao de alta	fonte de admissao	dias no hospital	especialidade medica	numero de procedimentos laboratoriais	numero de procedimentos	numero de m
0	Caucasian	Female	[0-10)	Outros	Outros	Encaminhamento Medico	1	Outros	41	0	
1	Caucasian	Female	[10-20)	Emergencia	Alta para casa	Sala de Emergencia	3	Medicina interna	59	0	
2	AfricanAmerican	Female	[20-30)	Emergencia	Alta para casa	Sala de Emergencia	2	Medicina interna	11	5	
3	Caucasian	Male	[30-40)	Emergencia	Alta para casa	Sala de Emergencia	2	Medicina interna	44	1	
4	Caucasian	Male	[40-50)	Emergencia	Alta para casa	Sala de Emergencia	1	Medicina interna	51	0	

In [38]:
```
# Distribuiçao dos diagnósticos

def plot_diags(col,data):
    sns.countplot(x = col, data = data,
                  order = data[f"{col}"].value_counts().index)
    plt.xticks(rotation = 85)
    plt.title(col)
    plt.show()

diag_cols = ["diagnostico_1","diagnostico_2","diagnostico_3"]

for d in diag_cols:
    plot_diags(d,diabetes_passo_11)
```


diagnostico_1
count
20000
17500
15000
12500
10000
7500
5000
2500
0
Circulatório
Outros
Respiratório
Digestivo
Diabetes
Ferimento
Musculoesquelético
Geniturinário
Neoplasias
diagnostico_1

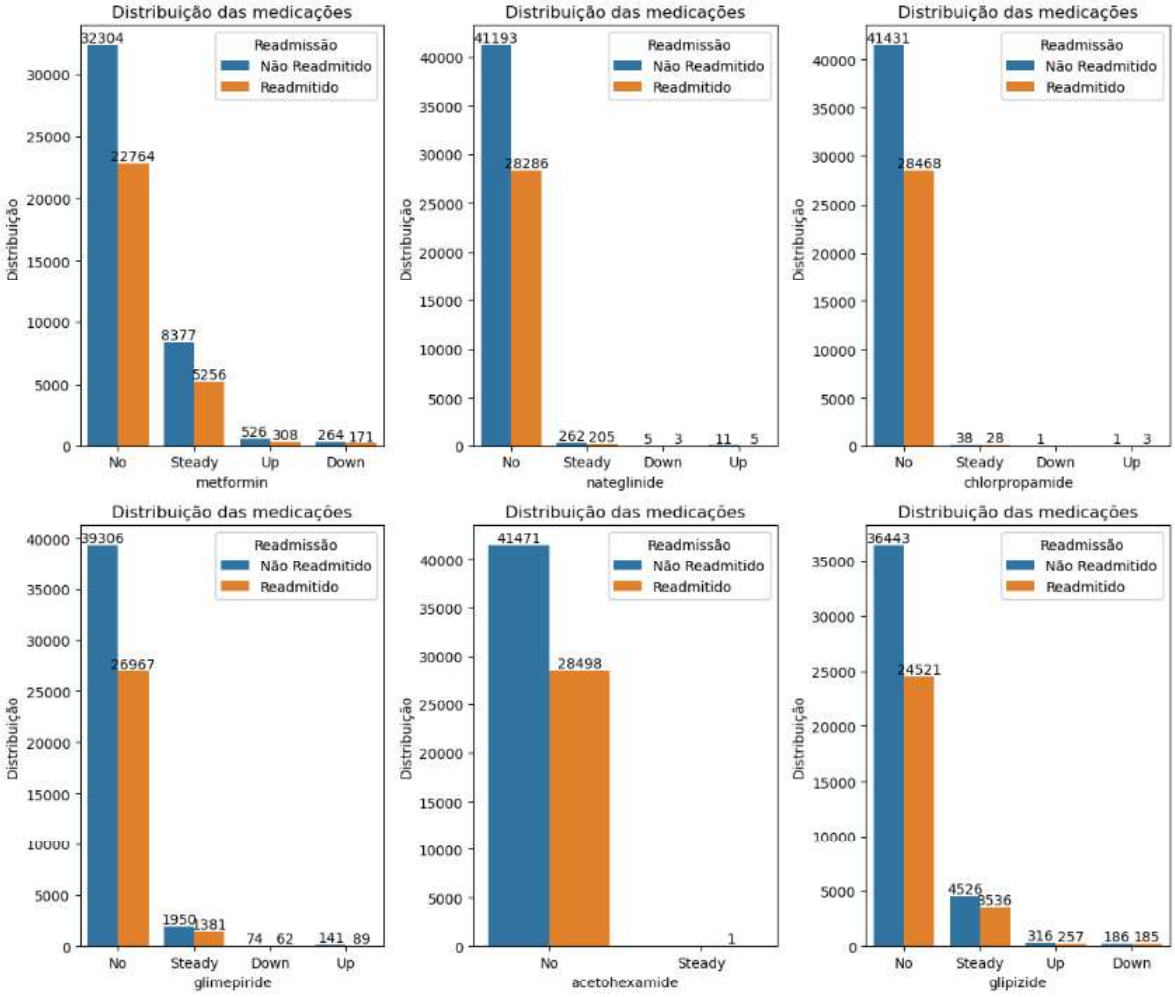

diagnostico_2
count
20000
15000
10000
5000
0
Circulatório
Outros
Diabetes
Respiratório
Geniturinário
Digestivo
Ferimento
Neoplasias
Musculoesquelético
diagnostico_2

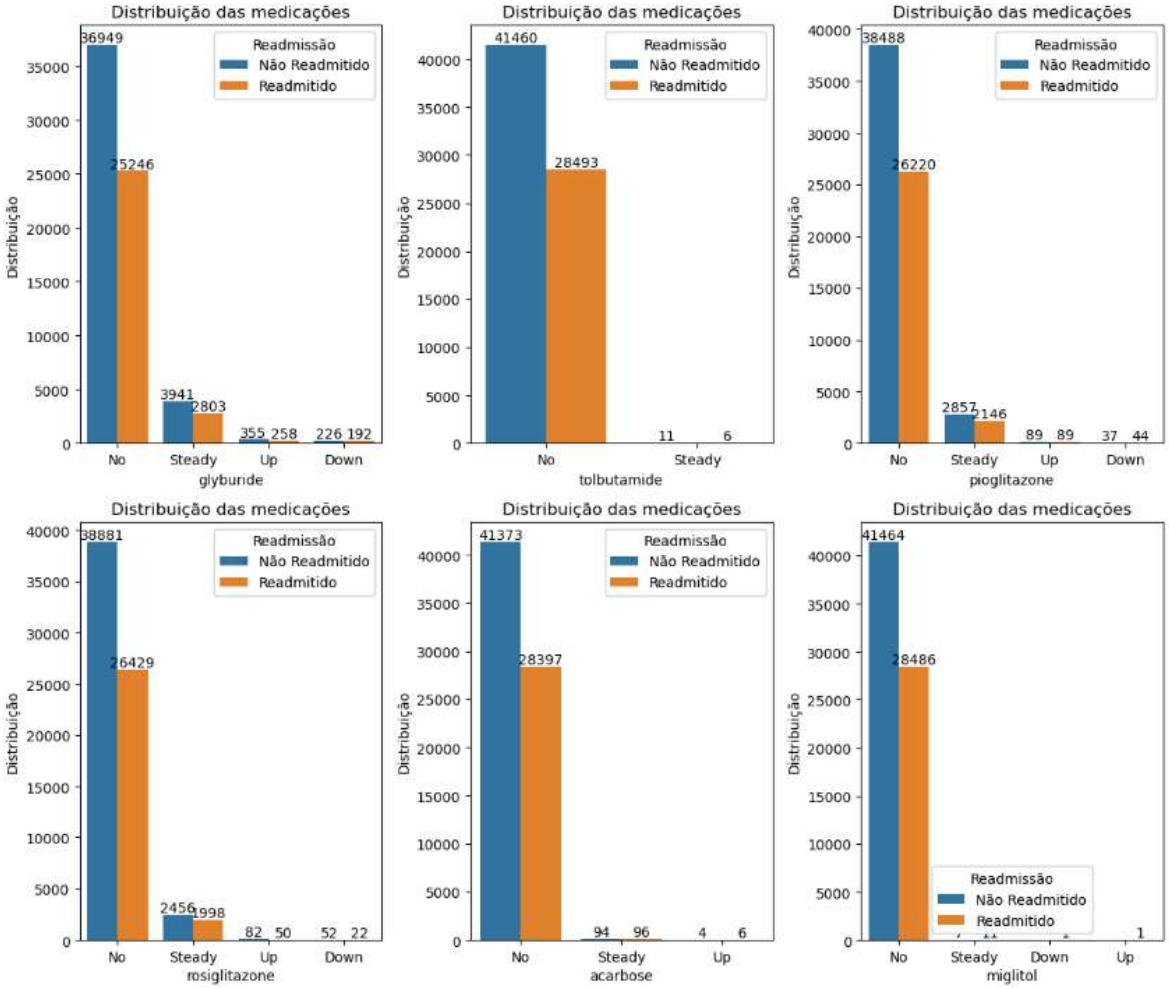


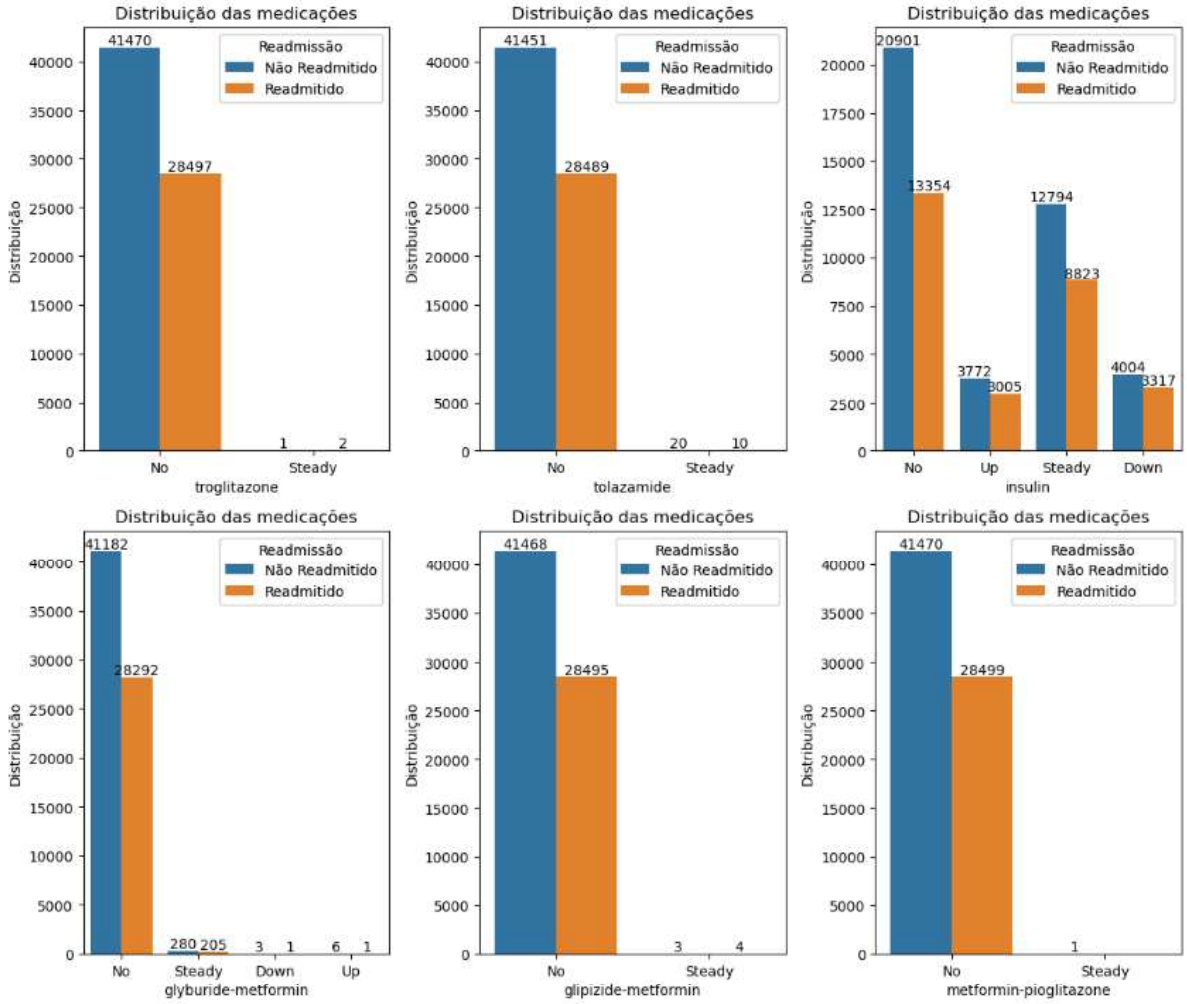
A.13. Análise distribuição das medicações

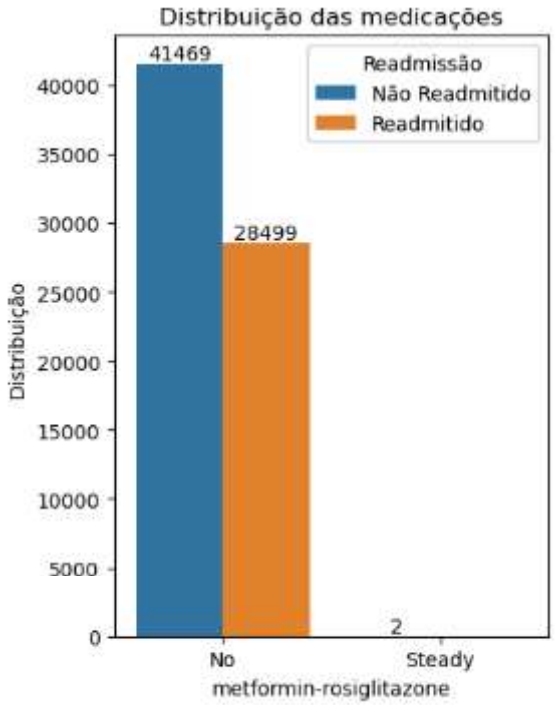
In [87]:

```
# Distribuição das medicações

diabetes_passo_12 = diabetes_passo_11.copy()


diabetes_passo_12 = diabetes_passo_11.copy()

cols = ['metformin', 'nateglinide', 'chlorpropamide',
        'glimepiride', 'acetohexamide', 'glipizide', 'glyburide', 'tolbutamide',
        'pioglitazone', 'rosiglitazone', 'acarbose', 'miglitol', 'troglitazone',
        'tolazamide', 'insulin', 'glyburide-metformin', 'glipizide-metformin',
        'metformin-pioglitazone', 'metformin-rosiglitazone']

def explore_drug(remedios):
    num_plots = len(remedios)
    num_per_row = 3
    num_rows = -(-num_plots // num_per_row)  # Ceil division

    fig, axes = plt.subplots(num_rows, num_per_row, figsize=(12, num_rows * 5))

    for i, remedio in enumerate(remedios):
        row = i // num_per_row
        col = i % num_per_row

        ax = axes[row, col] if num_rows > 1 else axes[col]

        sns.countplot(data=diabetes_passo_11, x=remedio, hue='readmitido', ax=ax)

        for p in ax.patches:
            ax.annotate(format(p.get_height(), '.0f'), (p.get_x() + p.get_width() / 2., p.get_height()),
                        ha='center', va='center', xytext=(0, 5), textcoords='offset points')

        ax.set_title('Distribuição das medicações')
        ax.set_xlabel(remedio)
        ax.set_ylabel('Distribuição')

        legenda = ax.legend(title='Readmissão', labels=['Não Readmitido', 'Readmitido'])
        ax.add_artist(legenda)

    plt.tight_layout()
    plt.show()

explore_drug(cols)
```


Distribuição das medicações
Readmissão
Não Readmitido
Readmitido
Distribuição
metformin
32304
22764
8377
5256
526
308
264
171
nateglinide
41193
28286
262
205
5
3
11
5
chlorpropamide
41431
28468
38
28
1
1
3
glimepiride
39306
26967
1950
1381
74
62
141
89
acetohexamide
41471
28498
1
glipizide
36443
24521
4526
3536
316
257
186
185
No
Steady
Up
Down


Distribuição das medicações
Readmissão
Não Readmitido
Readmitido
Distribuição
glyburide
No 36949
Steady 3941
Up 355
Down 226
Readmitido: 25246, 2803, 258, 192
tolbutamide
No 41460
Steady 11
Readmitido: 28493, 6
pioglitazone
No 38488
Steady 2857
Up 89
Down 37
Readmitido: 26220, 2146, 89, 44
rosiglitazone
No 38881
Steady 2456
Up 82
Down 52
Readmitido: 26429, 1998, 50, 22
acarbose
No 41373
Steady 94
Up 4
Readmitido: 28397, 96, 6
miglitol
No 41464
Readmitido: 28486
Up 1


Distribuição das medicações
Readmissão
Não Readmitido
Readmitido
Distribuição
41470
28497
1
2
No
Steady
troglitazone
Distribuição das medicações
41451
28489
20
10
No
Steady
tolazamide
Distribuição das medicações
20901
13354
3772
3005
12794
8823
4004
3317
No
Up
Steady
Down
insulin
Distribuição das medicações
41182
28292
280
205
3
1
6
1
No
Steady
Down
Up
glyburide-metformin
Distribuição das medicações
41468
28495
3
4
No
Steady
glipizide-metformin
Distribuição das medicações
41470
28499
1
No
Steady
metformin-pioglitazone


Distribuição das medicações
Readmissão
Não Readmitido
Readmitido
Distribuição
41469
28499
2
No
Steady
metformin-rosiglitazone

A.14. Agrupamento de teste ac1 e teste de soro de glicose e mudança de medicamento

In [41]:
```
# teste_ac1 e teste_de_soro_de_glicose
# transformação de dados de exames médicos, replicando tecnica utilizada no trabalho de (Nida Aslam, 2021)

diabetes_passo_12['teste_ac1'] = diabetes_passo_12['teste_ac1'].replace('>7', 1)
diabetes_passo_12['teste_ac1'] = diabetes_passo_12['teste_ac1'].replace('>8', 1)
diabetes_passo_12['teste_ac1'] = diabetes_passo_12['teste_ac1'].replace('Norm', 0)
diabetes_passo_12['teste_ac1'] = diabetes_passo_12['teste_ac1'].replace('None', -99)
diabetes_passo_12['teste_de_soro_de_glicose'] = diabetes_passo_12['teste_de_soro_de_glicose'].replace('>200', 1)
diabetes_passo_12['teste_de_soro_de_glicose'] = diabetes_passo_12['teste_de_soro_de_glicose'].replace('>300', 1)
diabetes_passo_12['teste_de_soro_de_glicose'] = diabetes_passo_12['teste_de_soro_de_glicose'].replace('Norm', 0)
diabetes_passo_12['teste_de_soro_de_glicose'] = diabetes_passo_12['teste_de_soro_de_glicose'].replace('None', -99)
diabetes_passo_12
```

Out[41]:

	raça	genero	idade	tipo_de_admissao	disposicao_de_alta	fonte_de_admissao	dias_no_hospital	especialidade_medica	numero_de_procedimentos_laboratoriais	numero_d
0	Caucasian	Female	[0-10)	Outros	Outros	Encaminhamento Medico	1	Outros	41	
1	Caucasian	Female	[10-20)	Emergencia	Alta para casa	Sala de Emergencia	3	Medicina interna	59	
2	AfricanAmerican	Female	[20-30)	Emergencia	Alta para casa	Sala de Emergencia	2	Medicina interna	11	
3	Caucasian	Male	[30-40)	Emergencia	Alta para casa	Sala de Emergencia	2	Medicina interna	44	
4	Caucasian	Male	[40-50)	Emergencia	Alta para casa	Sala de Emergencia	1	Medicina interna	51	
...	...	...	...	...	...	...	...	...	...	
101754	Caucasian	Female	[70-80)	Emergencia	Alta para casa	Sala de Emergencia	9	Medicina interna	50	
101755	Other	Female	[40-50)	Emergencia	Alta para casa	Sala de Emergencia	14	Medicina interna	73	
101756	Other	Female	[60-70)	Emergencia	Alta para casa	Sala de Emergencia	2	Medicina interna	46	
101758	Caucasian	Female	[80-90)	Emergencia	Alta para casa	Sala de Emergencia	5	Medicina interna	76	
101765	Caucasian	Male	[70-80)	Emergencia	Alta para casa	Sala de Emergencia	6	Medicina interna	13	

69970 rows × 43 columns

```
In [42]: diabetes_passo_12['mudanca_de_medicamento_de_diabetes'] = diabetes_passo_12['mudanca_de_medicamento_de_diabetes'].replace('Ch', 1)
         diabetes_passo_12['mudanca_de_medicamento_de_diabetes'] = diabetes_passo_12['mudanca_de_medicamento_de_diabetes'].replace('No', 0)
         diabetes_passo_12['genero'] = diabetes_passo_12['genero'].replace('Male', 1)
         diabetes_passo_12['genero'] = diabetes_passo_12['genero'].replace('Female', 0)
         diabetes_passo_12['prescrito_medicamento_para_diabetes'] = diabetes_passo_12['prescrito_medicamento_para_diabetes'].replace('Yes', 1)
         diabetes_passo_12['prescrito_medicamento_para_diabetes'] = diabetes_passo_12['prescrito_medicamento_para_diabetes'].replace('No', 0)

         for col in cols:
             diabetes_passo_12[col] = diabetes_passo_12[col].replace('No', 0)
             diabetes_passo_12[col] = diabetes_passo_12[col].replace('Steady', 1)
             diabetes_passo_12[col] = diabetes_passo_12[col].replace('Up', 1)
             diabetes_passo_12[col] = diabetes_passo_12[col].replace('Down', 1)

In [43]: diabetes_passo_12.head()
```

Out[43]:

	raça	genero	idade	tipo_de_admissao	disposicao_de_alta	fonte_de_admissao	dias_no_hospital	especialidade_medica	numero_de_procedimentos_laboratoriais	numero_de_proc
0	Caucasian	0	[0-10)	Outros	Outros	Encaminhamento Medico	1	Outros	41	
1	Caucasian	0	[10-20)	Emergencia	Alta para casa	Sala de Emergencia	3	Medicina interna	59	
2	AfricanAmerican	0	[20-30)	Emergencia	Alta para casa	Sala de Emergencia	2	Medicina interna	11	
3	Caucasian	1	[30-40)	Emergencia	Alta para casa	Sala de Emergencia	2	Medicina interna	44	
4	Caucasian	1	[40-50)	Emergencia	Alta para casa	Sala de Emergencia	1	Medicina interna	51	

A.15. Criação de duas novas medidas

```
In [44]: # Soma as três colunas e cria uma nova coluna 'total_consultas_ano_anterior'
         diabetes_passo_12['uso_hospital_ano_anterior'] = diabetes_passo_12['consultas_ambulatoriais_ano_anterior'] + diabetes_passo_12['consultas_internacao_ano_anter

In [45]: diabetes_passo_12['total_medicamentos_usados'] = diabetes_passo_12[cols].sum(axis=1)
         print(diabetes_passo_12['total_medicamentos_usados'].unique())

[0 1 2 3 4 5 6]
```

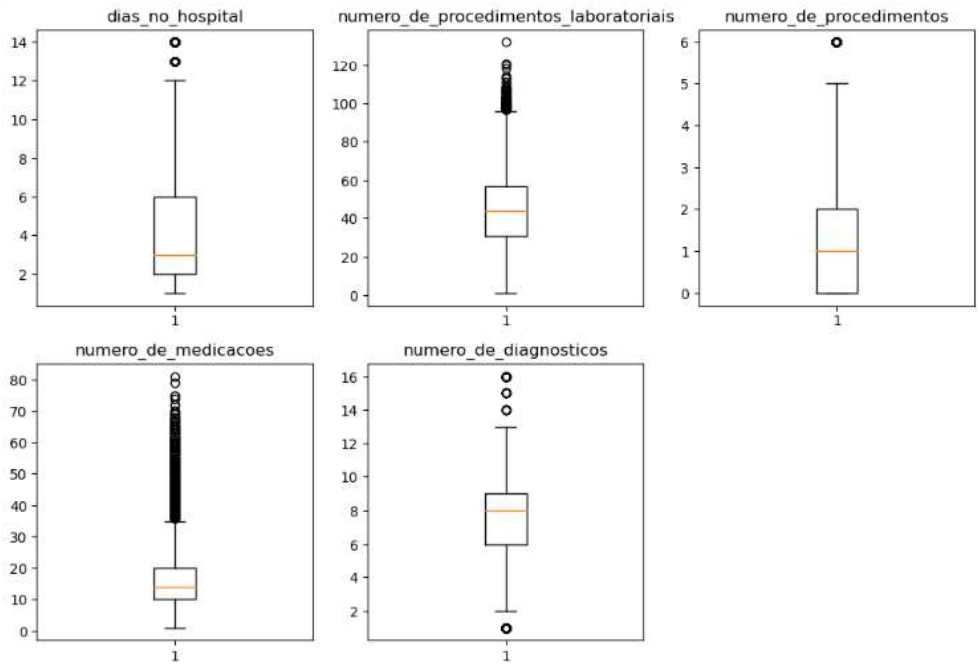
A.16. Remoção de atributos de medicação com baixa relevância quantitativa

```
In [46]: # medicação
         # Deletado medicações que possuiam baixa relevância quantitativa com valores diferentes de NO

         diabetes_passo_12 = diabetes_passo_12.drop(['acetohexamide','nateglinide','repaglinide', 'tolbutamide', 'acarbose','miglitol','t
         diabetes_passo_12.head()
```

A.17. Tratamento de outliers

Tratamento de outliers

In [47]:
```
plt.figure(figsize=(10, 10))

# Gráfico 1
plt.subplot(3, 3, 1)
plt.boxplot(diabetes_passo_12['dias_no_hospital'])
plt.title('dias_no_hospital')

# Gráfico 2
plt.subplot(3, 3, 2)
plt.boxplot(diabetes_passo_12['numero_de_procedimentos_laboratoriais'])
plt.title('numero_de_procedimentos_laboratoriais')

# Gráfico 3
plt.subplot(3, 3, 3)
plt.boxplot(diabetes_passo_12['numero_de_procedimentos'])
plt.title('numero_de_procedimentos')

# Gráfico 4
plt.subplot(3, 3, 4)
plt.boxplot(diabetes_passo_12['numero_de_medicacoes'])
plt.title('numero_de_medicacoes')

# Gráfico 5
plt.subplot(3, 3, 5)
plt.boxplot(diabetes_passo_12['numero_de_diagnosticos'])
plt.title('numero_de_diagnosticos')

# Ajuste o layout dos gráficos
plt.tight_layout()

plt.show()
```

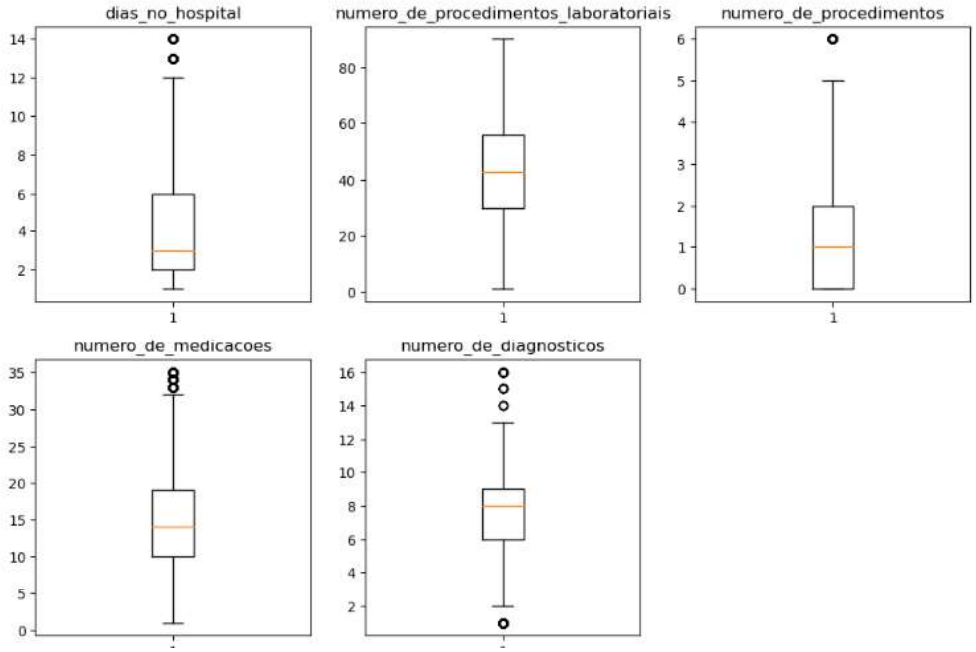


Os outliers foram facilmente identificáveis, desta torma, optou-se por excluí-los manualmente

In [48]:
```
# tratamento de outlier

diabetes_passo_14 = diabetes_passo_12.copy()

linhas_deletadas = diabetes_passo_14.loc[diabetes_passo_14['numero_de_procedimentos_laboratoriais'] >= 90]
print('linhas deletadas:', linhas_deletadas.shape)

diabetes_passo_14 = diabetes_passo_14.loc[diabetes_passo_14['numero_de_procedimentos_laboratoriais'] <= 90]

linhas_deletadas = diabetes_passo_14.loc[diabetes_passo_14['numero_de_medicacoes'] >= 35]
print('linhas deletadas:', linhas_deletadas.shape)

diabetes_passo_14 = diabetes_passo_14.loc[diabetes_passo_14['numero_de_medicacoes'] <= 35]
diabetes_passo_14.shape
```

Out[48]:
```
linhas deletadas: (353, 33)
linhas deletadas: (2069, 33)
(67845, 33)
```

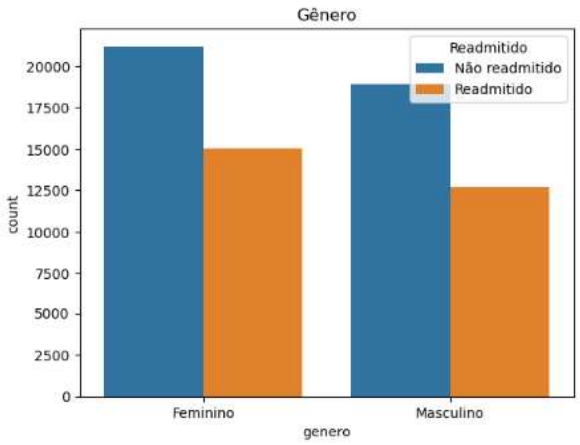
In [49]:
```
plt.figure(figsize=(10, 10))

# Gráfico 1
plt.subplot(3, 3, 1)
plt.boxplot(diabetes_passo_14['dias_no_hospital'])
plt.title('dias_no_hospital')

# Gráfico 2
plt.subplot(3, 3, 2)
plt.boxplot(diabetes_passo_14['numero_de_procedimentos_laboratoriais'])
plt.title('numero_de_procedimentos_laboratoriais')

# Gráfico 3
plt.subplot(3, 3, 3)
plt.boxplot(diabetes_passo_14['numero_de_procedimentos'])
plt.title('numero_de_procedimentos')

# Gráfico 4
plt.subplot(3, 3, 4)
plt.boxplot(diabetes_passo_14['numero_de_medicacoes'])
plt.title('numero_de_medicacoes')

# Gráfico 5
plt.subplot(3, 3, 5)
plt.boxplot(diabetes_passo_14['numero_de_diagnosticos'])
plt.title('numero_de_diagnosticos')

# Ajuste o layout dos gráficos
plt.tight_layout()

plt.show()
```



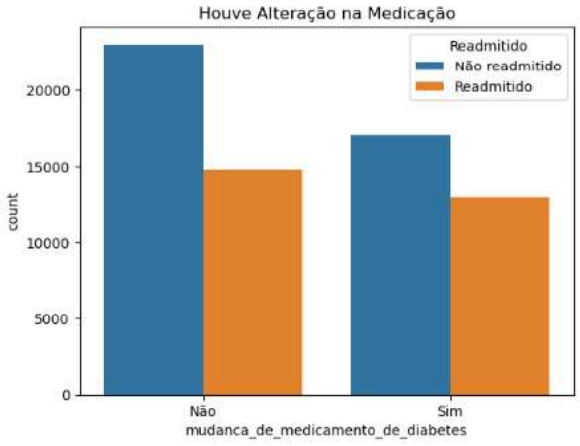
A.18. Análise pós tratamento de dados

analise exploratória

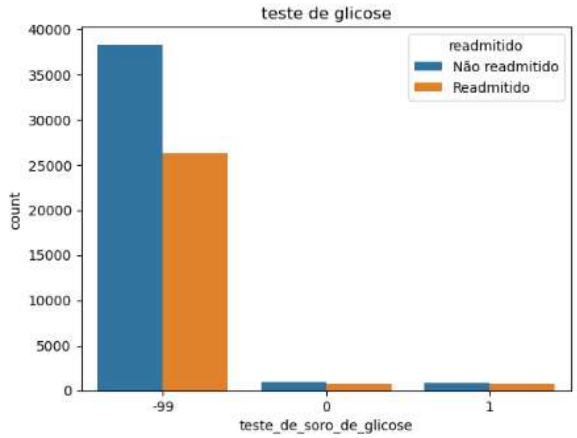
In [50]:
```
sns.countplot(x="genero", hue="readmitido", data=diabetes_passo_14)

plt.title("Gênero")
plt.legend(title='Readmitido', labels=['Não readmitido', 'Readmitido'])

plt.xticks([0, 1], ['Feminino', 'Masculino'])

plt.show()
```



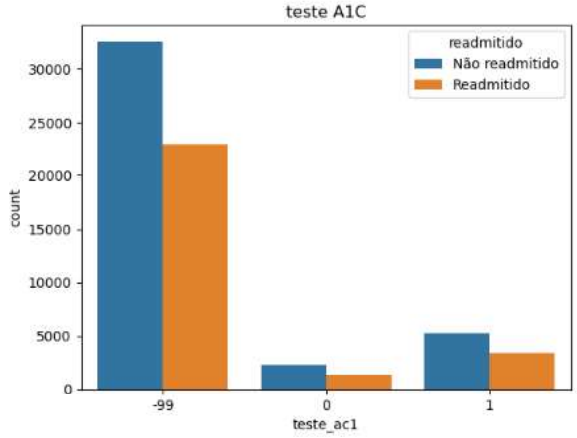
In [51]:
```
sns.countplot(x="mudanca_de_medicamento_de_diabetes", hue="readmitido", data=diabetes_passo_14)

plt.title("Houve Alteração na Medicação")
plt.legend(title='Readmitido', labels=['Não readmitido', 'Readmitido'])

plt.xticks([0, 1], ['Não', 'Sim'])

plt.show()
```



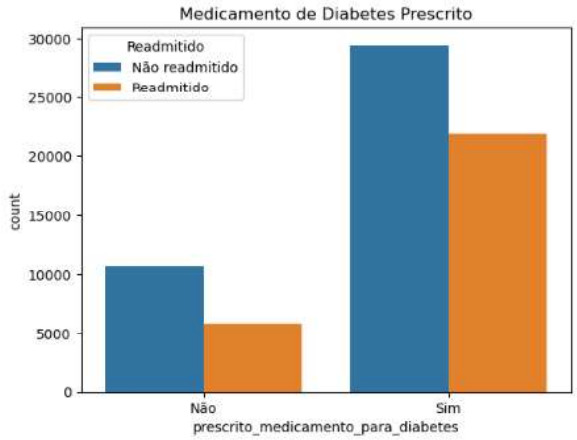
In [52]:
```
# Distribuição de teste de soro de glicose, por readmissão
sns.countplot(x="teste_de_soro_de_glicose", hue="readmitido", data=diabetes_passo_14)
plt.title("teste de glicose")
# Adiciona a Legenda
plt.legend(title='readmitido', labels=['Não readmitido', 'Readmitido'])
plt.show()
```



In [53]:
```
# Criar o gráfico de barras
sns.countplot(x="teste_ac1", hue="readmitido", data=diabetes_passo_14)
plt.title("teste A1C")
# Adiciona a Legenda
plt.legend(title='readmitido', labels=['Não readmitido', 'Readmitido'])
plt.show()
```



In [54]:
```
sns.countplot(x="prescrito_medicamento_para_diabetes", hue="readmitido", data=diabetes_passo_14)

plt.title("Medicamento de Diabetes Prescrito")
plt.legend(title='Readmitido', labels=['Não readmitido', 'Readmitido'])

plt.xticks([0, 1], ['Não', 'Sim'])

plt.show()
```



proporção de classe

In [55]:
```
diabetes_passo_15 = diabetes_passo_14.copy()

# proporção de classe
diabetes_passo_15['readmitido'].value_counts()
```

Out[55]:
```
0    40101
1    27744
Name: readmitido, dtype: int64
```

In [56]:
```
# 0 representa não readmitido e 1 readmmitido
nao_readmitido = diabetes_passo_15[diabetes_passo_15['readmitido']==0].shape[0]
readmitido = diabetes_passo_15[diabetes_passo_15['readmitido']==1].shape[0]
```

In [57]:
```
cores = ['red', 'blue']

rotulos = ['Readmitido', 'Não Readmitido']

# Barras separadas
plt.bar(x=[1], height=[readmitido], color=['red'], label='Readmitido')
plt.bar(x=[2], height=[nao_readmitido], color=['blue'], label='Não Readmitido')

plt.legend()

plt.xlabel("")
plt.ylabel("Quantidade")

plt.title("Variável alvo não balanceada")

plt.xticks([]) # Remove rótulos do eixo x

plt.show()
```

pre processamento de dados

In [58]:
```
diabetes_passo_15.head()
```

Out[58]:

	raça	genero	idade	tipo_de_admissao	disposicao_de_alta	fonte_de_admissao	dias_no_hospital	especialidade_medica	numero_de_procedimentos_laboratoriais	numero_de_proc
0	Caucasian	0	[0-10)	Outros	Outros	Encaminhamento Medico	1	Outros	41	
1	Caucasian	0	[10-20)	Emergencia	Alta para casa	Sala de Emergencia	3	Medicina interna	59	
2	AfricanAmerican	0	[20-30)	Emergencia	Alta para casa	Sala de Emergencia	2	Medicina interna	11	
3	Caucasian	1	[30-40)	Emergencia	Alta para casa	Sala de Emergencia	2	Medicina interna	44	
4	Caucasian	1	[40-50)	Emergencia	Alta para casa	Sala de Emergencia	1	Medicina interna	51	

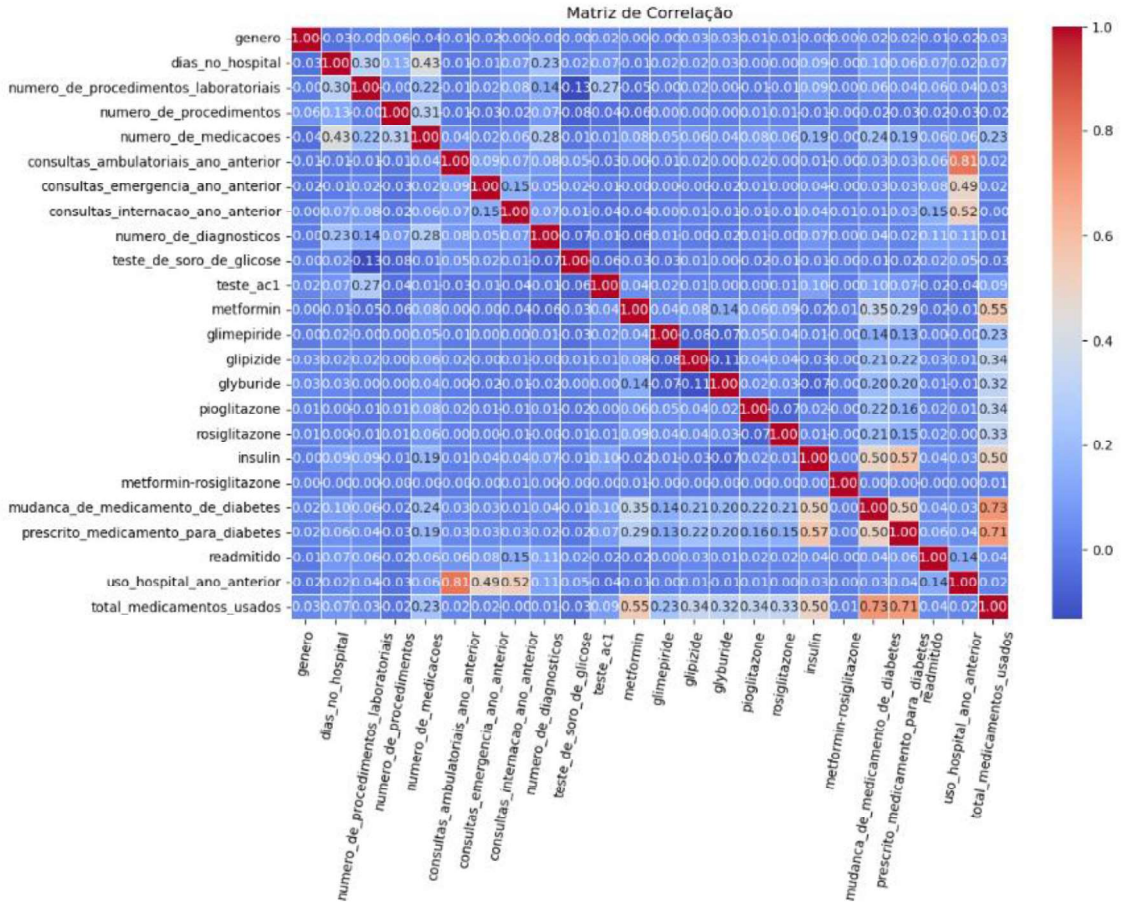
In [59]:
```
diabetes_passo_15.shape
```

Out[59]:
```
(67845, 33)
```

In [60]:
```
diabetes_passo_15.columns
```

Out[60]:
```
Index(['raça', 'genero', 'idade', 'tipo_de_admissao', 'disposicao_de_alta',
       'fonte_de_admissao', 'dias_no_hospital', 'especialidade_medica',
       'numero_de_procedimentos_laboratoriais', 'numero_de_procedimentos',
       'numero_de_medicacoes', 'consultas_ambulatoriais_ano_anterior',
       'consultas_emergencia_ano_anterior',
       'consultas_internacao_ano_anterior', 'diagnostico_1', 'diagnostico_2',
       'diagnostico_3', 'numero_de_diagnosticos', 'teste_de_soro_de_glicose',
       'teste_ac1', 'metformin', 'glimepiride', 'glipizide', 'glyburide',
       'pioglitazone', 'rosiglitazone', 'insulin', 'metformin-rosiglitazone',
       'mudanca_de_medicamento_de_diabetes',
       'prescrito_medicamento_para_diabetes', 'readmitido',
       'uso_hospital_ano_anterior', 'total_medicamentos_usados'],
      dtype='object')
```

In [61]:
```
correlation_matrix = diabetes_passo_15.corr()

plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title("Matriz de Correlação")
plt.xticks(rotation=80)
plt.show()
```



A.19. Divide treino e teste

Divide treino e teste

In [62]:
```
label = diabetes_passo_15['readmitido']
diabetes = diabetes_passo_15.drop(['readmitido'], axis = 1)
```

In [63]:
```
# dummy para variáveis categóricas
diabetes = pd.get_dummies(diabetes)
```

In [64]:
```
diabetes.head()
```

Out[64]:

	genero	dias_no_hospital	numero_de_procedimentos_laboratoriais	numero_de_procedimentos	numero_de_medicacoes	consultas_ambulatoriais_ano_anterior	consultas_emergencia_ano_ante
0	0	1	41	0	1	0	
1	0	3	59	0	18	0	
2	0	2	11	5	13	2	
3	1	2	44	1	16	0	
4	1	1	51	0	8	0	

In [65]:
```
# Split dos dados
X_treino, X_teste, Y_treino, Y_teste = train_test_split(diabetes,
                                                        label,
                                                        test_size = .3,
                                                        random_state = 0,
                                                        stratify = label )
```

In [66]:
```
# Shape
print('X_treino : ', X_treino.shape)
print('Y_treino : ', Y_treino.shape)
print('X_teste : ', X_teste.shape)
print('Y_teste : ', Y_teste.shape)
```

```
X_treino :  (47491, 79)
Y_treino :  (47491,)
X_teste :  (20354, 79)
Y_teste :  (20354,)
```

In [67]:
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# Cria o padronizador
scaler = StandardScaler()

# Treina o padronizador com o método fit() e aplica com o método transform() nos dados de treino.
X_treino_scaled = scaler.fit_transform(X_treino)
print('X_treino_scaled : ', X_treino_scaled.shape)

# Aplica PCA nos dados de treino padronizados
# pca = PCA(n_components=NUM_COMPONENTS)  # Substitua NUM_COMPONENTS pelo número desejado de componentes
pca = PCA()
X_treino_scaled = pca.fit_transform(X_treino_scaled)
print('X_treino_pca : ', X_treino_scaled.shape)

# Aplica o mesmo PCA nos dados de teste (apenas o transform)
X_teste_scaled = scaler.transform(X_teste)
X_teste_scaled = pca.transform(X_teste_scaled)
print('X_teste_pca : ', X_teste_scaled.shape)
```

```
X_treino_scaled :  (47491, 79)
X_treino_pca :  (47491, 79)
X_teste_pca :  (20354, 79)
```

In [68]:
```
X_teste_scaled.shape
```

Out[68]:
```
(20354, 79)
```

In [69]:
```
Y_teste.shape
```

Out[69]:
```
(20354,)
```

In [70]:
```
X_teste_scaled.shape
```

Out[70]:
```
(20354, 79)
```

In [71]:
```
# Caso a reprodução do estudo apresente erro no balanceamento de classes, execute o codigo desta célula
# pip install threadpoolctl==3.1.0
```

balanceamento de classes

In [72]:
```
from imblearn.over_sampling import SMOTE

# dados de treino em X e Y
X = X_treino_scaled
y = Y_treino

# objeto SMOTE
balanceador = SMOTE()

# treina e balancea o conjunto de dados
X_treino_scaled_bal, Y_treino_bal = balanceador.fit_resample(X, y)
```

In [73]:
```
# Concatena x e y no mesmo df
dados_pos_balanceamento = pd.DataFrame(X_treino_scaled_bal)
dados_pos_balanceamento['readmitido'] = pd.DataFrame(Y_treino_bal)
dados_pos_balanceamento.head()
```

Out[73]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	4.660649	-0.010975	-2.417983	-0.630444	-1.130941	0.135296	1.893467	2.039814	-2.145460	0.669179	1.414430	-1.612932	1.496987	-0.110556	-0.516227	0.254893	-0.523609	-0
1	-2.116749	0.272607	0.650451	-0.221999	-0.360695	1.206273	-0.231071	-2.040309	0.424302	-0.494133	0.354632	-0.044268	2.171169	0.986355	0.819881	-1.145465	1.336121	-0
2	-0.186160	-1.619718	2.131846	-0.650975	-1.819402	1.499496	0.612725	-0.263638	2.251832	1.230529	-0.280919	-0.211714	1.047914	1.284746	2.060327	0.460304	-0.173085	-1
3	-1.296007	0.390560	0.487923	-1.216639	0.030290	-1.020496	-0.729387	0.504748	0.624167	1.272895	-2.049975	1.960997	-0.474695	1.116256	-0.836083	-0.713222	0.314258	-0
4	2.706590	0.945715	1.344796	-1.847964	-0.452737	0.917356	-0.604020	-0.240176	-0.081319	-1.841206	-1.039018	-0.632100	-1.503432	-0.062158	1.558872	2.079078	1.505036	0

In [74]:
```
dados_pos_balanceamento.shape
```

Out[74]:
```
(56140, 80)
```

In [75]:
```
# proporção de classe
diabetes_passo_15['readmitido'].value_counts()
```

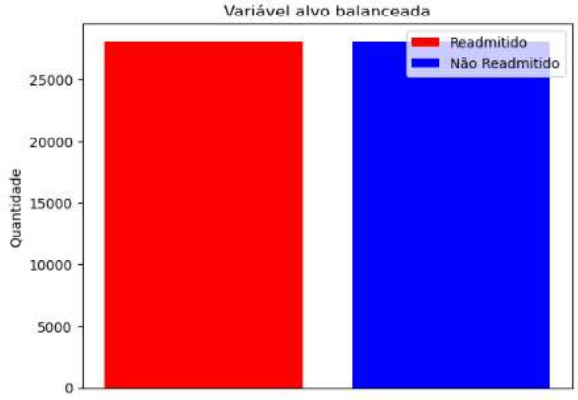
Out[75]:
```
0    40101
1    27744
Name: readmitido, dtype: int64
```

In [76]:
```
# proporção de classe pós balanceamento
dados_pos_balanceamento['readmitido'].value_counts()
```

Out[76]:
```
1    28070
0    28070
Name: readmitido, dtype: int64
```

A.20. Balanceamento de classe

```
In [77]: nao_readmitido = dados_pos_balanceamento[dados_pos_balanceamento['readmitido']==0].shape[0]
         readmitido = dados_pos_balanceamento[dados_pos_balanceamento['readmitido']==1].shape[0]

In [78]: cores = ['red', 'blue']

         rotulos = ['Readmitido', 'Não Readmitido']

         plt.bar(x=[1], height=[readmitido], color=['red'], label='Readmitido')
         plt.bar(x=[2], height=[nao_readmitido], color=['blue'], label='Não Readmitido')

         plt.legend()

         plt.xlabel("")
         plt.ylabel("Quantidade")

         plt.title("Variável alvo balanceada")

         plt.xticks([]) # Remove rótulos do eixo x

         plt.show()
```



A.21. Quadro com lista de características do conjunto de dados final

Atributo	Coluna	Tipo	Descrição e valores
Raça	raça	Nominal	Valores: Caucasian, Asian, African American, Hispanic, and other
Gênero	genero	Nominal	Valores: 0, 1
Idade	idade	Nominal	Agrupado em 5 intervalos: [0,50), [50,60), [60,70), [70,80), [80,100)
Tipo de admissão	tipo_de_admissao	Nominal	Identificador inteiro que corresponde a 3 valores distintos: Emergência, Eletivo e Outros
Disposição de alta	disposicao_de_alta	Nominal	Identificador inteiro correspondente a 29 valores distintos, por exemplo, alta para casa, expirado e não disponível
Fonte de admissão	fonte_de_admissao	Nominal	Identificador inteiro correspondente a 21 valores distintos. Por exemplo:

			encaminhamento médico, pronto-socorro e transferência de hospital
Tempo no hospital	dias_no_hospital	Numérico	Número inteiro de dias entre a admissão e a alta
Número de procedimentos laboratoriais	numero_de_procedimentos_laboratoriais	Numérico	Número de exames laboratoriais realizados durante o encontro
Número de procedimentos	numero_de_procedimentos	Numérico	Número de procedimentos (exceto exames laboratoriais) realizados durante o encontro
Número de medicações	numero_de_medicacoes	Numérico	Número de nomes genéricos distintos administrados durante o encontro
Número de consultas ambulatoriais	consultas_ambulatoriais_ano_anterior	Numérico	Número de consultas ambulatoriais do paciente no ano anterior ao encontro
Número de visitas de emergência	consultas_emergencia_ano_anterior	Numérico	Número de visitas de emergência do paciente no ano anterior ao encontro
Número de consultas de internação	consultas_internacao_ano_anterior	Numérico	Número de visitas de internação do paciente no ano anterior ao encontro
Diagnóstico 1	diagnostico_1	Nominal	O diagnóstico primário, reclassificado em 9 atributos de seus respectivos CID's. Valores: Diabetes, Circulatório, Respiratório, Digestivo, Geniturinário, Neoplasias, Musculoesquelético, Ferimento e Outros.
Diagnóstico 2	diagnostico_2	Nominal	O diagnóstico primário, reclassificado em 9 atributos de seus respectivos CID's. Valores: Diabetes, Circulatório, Respiratório, Digestivo, Geniturinário, Neoplasias, Musculoesquelético, Ferimento e Outros.
Diagnóstico 3	diagnostico_3	Nominal	O diagnóstico primário, reclassificado em 9 atributos de seus respectivos CID's. Valores: Diabetes, Circulatório,

			Respiratório, Digestivo, Geniturinário, Neoplasias, xMusculoesquelético, Ferimento e Outros.
Número de diagnósticos	numero_de_diagnosticos	Numérico	Número de diagnósticos inseridos no sistema
Resultado do teste de soro de glicose	teste_de_soro_de_glicose	Nominal	“0” se o resultado foi normal, “1” se o resultado foi fora do normal, “-99” se não foi medido.
Resultado do teste A1c	teste_ac1	Nominal	“0” se o resultado foi normal, “1” se o resultado foi fora do normal, “-99” se não foi medido.
Especialidade médica	especialidade_medica	Nominal	Valores: "Cardiologia", "Prática geral", "Medicina interna", "Cirurgia”, "Emergência".
Uso do hospital no ano anterior	uso_hospital_ano_anterior	Numérico	Total de internações no ano anterior
Total de medicamentos usados	total_medicamentos_usados	Numérico	Total de medicamentos administrados no encontro
Mudança de medicamentos	mudanca_de_medicamento_de_diabetes	Nominal	Indica se houve alteração nos medicamentos para diabéticos (seja posologia ou nome genérico). Valores: “1” quando houve alteração e “0” se não houve
Medicamentos para diabetes	prescrito_medicamento_para_diabetes	Nominal	Indica se foi administrado algum medicamento para diabéticos. Valores: “1” se foi prescrito e “0” se não foi administrado
7 atributos para medicamentos		Nominal	Para os nomes genéricos: metformina, glimepirida, glipizida, gliburida, pioglitazona, rosiglitazona, insulina. A característica indica se o medicamento

			foi prescrito ou não. “1” em casos afirmativos e “0” para casos negativos.
Readmitido	readmitido	Nominal	Valores: “1”, caso seja readmissão e “0”, caso não seja.

Fonte: Adaptado de Impact of HbA1c measurement on hospital readmission rates: analysis of 70,000 clinical database patient records (STRACK et al, 2014)

A.22. Aplicação dos algoritmos

A.22.1. Redes Neurais Artificiais

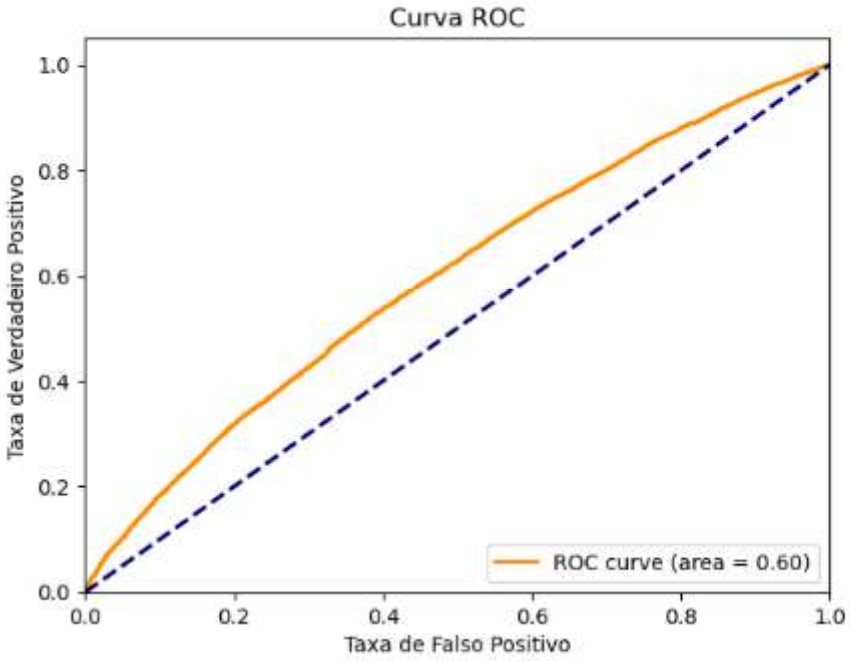
Modelagem Preditiva

Redes Neurais Artificiais

In [79]:

```
X, Y = X_treino_scaled_bal, Y_treino_bal

# Cria modelo de rede neural
model = Sequential()
model.add(Dense(64, input_dim=X_treino.shape[1], activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

# Compila o modelo
model.compile(loss='binary_crossentropy', optimizer=Adam(learning_rate=0.001), metrics=['accuracy', AUC()])

# Treina o modelo
history = model.fit(X_treino_scaled_bal, Y_treino_bal, epochs=50, batch_size=32, validation_split=0.1)

# Faz as previsões
nn_Y_pred = (model.predict(X_teste_scaled) > 0.5).astype(int)

# Calcula a acurácia com dados de teste
nn_accuracy = model.evaluate(X_teste_scaled, Y_teste, verbose=0)[1] * 100

print("Acurácia do modelo de Rede Neural (%):", nn_accuracy)
print("\n")


# Calcula a matriz de confusão
conf_matrix = confusion_matrix(Y_teste, nn_Y_pred)

print("Matriz de Confusão:")
print(conf_matrix)
print(" ")

print('Relatório de Classificação')
print(classification_report(Y_teste, nn_Y_pred, target_names=['Não', 'Sim']))

# Calcula as probabilidades para a classe positiva
y_prob = model.predict(X_teste_scaled).ravel()

# Calcula a curva ROC
fpr, tpr, _ = roc_curve(Y_teste, y_prob)
roc_auc = auc(fpr, tpr)

# Plota a curva ROC
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Taxa de Falso Positivo')
plt.ylabel('Taxa de Verdadeiro Positivo')
plt.title('Curva ROC')
plt.legend(loc="lower right")
plt.show()
```

```
Matriz de Confusão:
[[8086 3945]
 [4497 3826]]

Relatório de Classificação
              precision    recall  f1-score   support

         Não       0.64      0.67      0.66     12031
         Sim       0.49      0.46      0.48      8323

    accuracy                           0.59     20354
   macro avg       0.57      0.57      0.57     20354
weighted avg       0.58      0.59      0.58     20354

637/637 [==============================] - 0s 645us/step
```



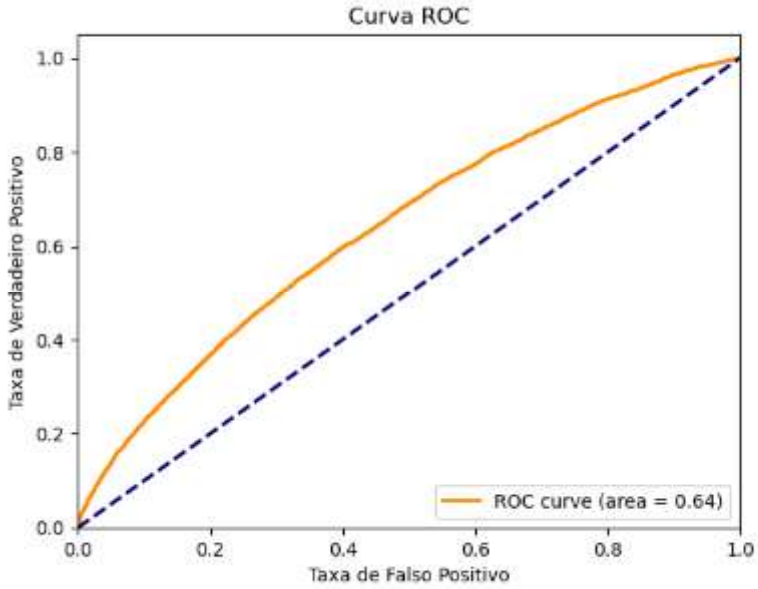
A.22.2. Algoritmo de regressão logística

Regressão Logística

In [80]:

```
# Cria modelo LR
modelo_lr = LogisticRegression()

# Hiperparâmetros para busca em grade
param_grid = {
     'tol': [1e-4, 1e-5, 1e-6],
     'C': [0.001, 0.01, 0.1, 1.0],
     'solver': ['liblinear', 'lbfgs', 'sag', 'saga']
}

params = {'C': 0.001, 'solver': 'liblinear', 'tol': 0.0001}
modelo_lr = LogisticRegression(**params)

# GridSearchCV
grid_search = GridSearchCV(modelo_lr, param_grid, scoring='accuracy')

# Treina o modelo com o GridSearchCV
grid_search.fit(X_treino_scaled_bal, Y_treino_bal)

# Melhor modelo encontrado pelo GridSearchCV
melhor_modelo = grid_search.best_estimator_

# Usa o melhor modelo
lr_Y_pred = melhor_modelo.predict(X_teste_scaled)

# Calcula a acurácia com dados de teste
lr_accuracy = melhor_modelo.score(X_teste_scaled, Y_teste) * 100

# Print
print("Melhores hiperparâmetros encontrados:", grid_search.best_params_)
print("\nAcurácia do melhor modelo de Regressão Logística (%):", lr_accuracy)
print("\n")

# Calcula a matriz de confusão
conf_matrix = confusion_matrix(Y_teste, lr_Y_pred)

print("Matriz de Confusão:")
print(conf_matrix)
print(" ")


# Calcula a precisão, sensibilidade (razão) e F1-score
report = classification_report(Y_teste, lr_Y_pred, target_names=['Não', 'Sim'])
print("Relatório de Classificação:")
print(report)

# Calcula as probabilidades para a classe positiva
y_prob = melhor_modelo.predict_proba(X_teste_scaled)[:, 1]

# Calcula a curva ROC
fpr, tpr, _ = roc_curve(Y_teste, y_prob)
roc_auc = auc(fpr, tpr)

# Plota a curva ROC
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Taxa de Falso Positivo')
plt.ylabel('Taxa de Verdadeiro Positivo')
plt.title('Curva ROC')
plt.legend(loc="lower right")
plt.show()

# Obtém os coeficientes das variáveis
coeficientes = melhor_modelo.coef_[0]

# Associa os coeficientes às colunas do DataFrame original
coeficientes_df = pd.DataFrame({'Variável': X_treino.columns, 'Coeficiente': coeficientes})

# Ordena as variáveis por importância
coeficientes_df = coeficientes_df.sort_values(by='Coeficiente', ascending=False)

# 10 variáveis mais importantes
print("As 10 variáveis mais importantes:")
print(coeficientes_df.iloc[:10])
```

```
Melhores hiperparâmetros encontrados: {}

Acurácia do melhor modelo de Regressão Logística (%): 60.17981723494154


Matriz de Confusão:
[[7403 4628]
 [3477 4846]]

Relatório de Classificação:
              precision    recall  f1-score   support

         Não       0.68      0.62      0.65     12031
         Sim       0.51      0.58      0.54      8323

    accuracy                           0.60     20354
   macro avg       0.60      0.60      0.60     20354
weighted avg       0.61      0.60      0.60     20354
```



As 10 variáveis mais importantes:

	Variável	Coeficiente
4	numero_de_medicacoes	0.152864
2	numero_de_procedimentos_laboratoriais	0.149109
63	diagnostico_2_Digestivo	0.118467
64	diagnostico_2_Ferimento	0.096615
5	consultas_ambulatoriais_ano_anterior	0.090337
34	idade_[60-70)	0.086513
59	diagnostico_1_Outros	0.069274
43	fonte_de_admissao_Encaminhamento Medico	0.060972
20	prescrito_medicamento_para_diabetes	0.059452
61	diagnostico_2_Circulatório	0.059364

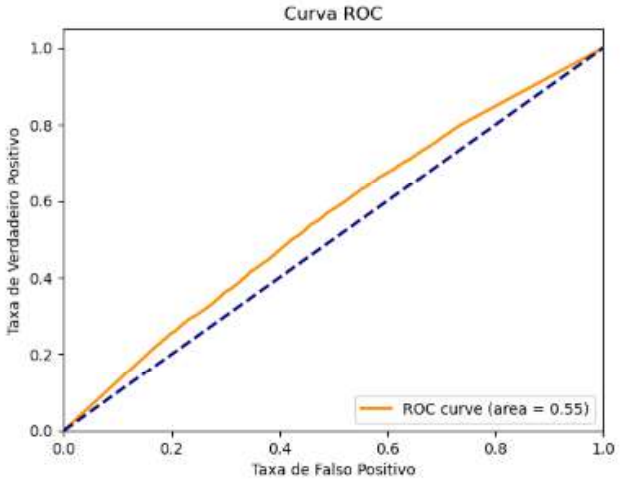
A.22.3. Algoritmo de árvore de decisão

Arvore de decisão

In [81]:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix, classification_report, roc_curve, auc

# Cria o modelo Árvore de Decisão
modelo_tree = DecisionTreeClassifier(random_state=1)

# Hiperparâmetros para busca em grade
param_grid = {
    'criterion': ['gini', 'entropy'],  # Critério de divisão
    'splitter': ['best', 'random'],  # Estratégia de divisão
    'max_depth': [None, 10, 20, 30],  # Profundidade máxima da árvore
    'min_samples_split': [2, 5, 10],  # Mínimo de amostras para dividir um nó
    'min_samples_leaf': [10, 20, 30]  # Mínimo de amostras em uma folha
}

# Crie o objeto GridSearchCV
grid_search = GridSearchCV(modelo_tree, param_grid, scoring='accuracy')

# Treine o modelo com a busca em grade
grid_search.fit(X_treino_scaled_bal, Y_treino_bal)

# Obtenha o melhor modelo encontrado pela busca em grade
melhor_modelo = grid_search.best_estimator_

# Faz as previsões com o melhor modelo
tree_Y_pred = melhor_modelo.predict(X_teste_scaled)

# Calcula a acurácia com dados de teste
tree_accuracy = melhor_modelo.score(X_teste_scaled, Y_teste) * 100

# Print
print("Melhores hiperparâmetros encontrados:", grid_search.best_params_)
print("\nAcurácia do melhor modelo Decision Tree (%):", tree_accuracy)
print("\n")

# Calcula a matriz de confusão
conf_matrix = confusion_matrix(Y_teste, tree_Y_pred)

# Exibe a matriz de confusão
print("Matriz de Confusão:")
print(conf_matrix)
print(" ")

# Exibe o relatório de classificação
print('Relatório de Classificação:')
print(classification_report(Y_teste, tree_Y_pred, target_names=['Não', 'Sim']))
print("\n")

# Calcula as probabilidades para a classe positiva
tree_y_prob = melhor_modelo.predict_proba(X_teste_scaled)[:, 1]

# Calcula a curva ROC
fpr, tpr, thresholds = roc_curve(Y_teste, tree_y_prob)
roc_auc = auc(fpr, tpr)

# Plota a curva ROC
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Taxa de Falso Positivo')
plt.ylabel('Taxa de Verdadeiro Positivo')
plt.title('Curva ROC')
plt.legend(loc="lower right")
plt.show()
```

```
Melhores hiperparâmetros encontrados: {'criterion': 'entropy', 'max_depth': 30, 'min_samples_leaf': 10, 'min_samples_split': 2, 'splitter': 'best'}

Acurácia do melhor modelo Decision Tree (%): 54.77547410828338


Matriz de Confusão:
[[7241 4790]
 [4415 3908]]

Relatório de Classificação:
              precision    recall  f1-score   support

         Não       0.62      0.60      0.61     12031
         Sim       0.45      0.47      0.46      8323

    accuracy                           0.55     20354
   macro avg       0.54      0.54      0.54     20354
weighted avg       0.55      0.55      0.55     20354
```

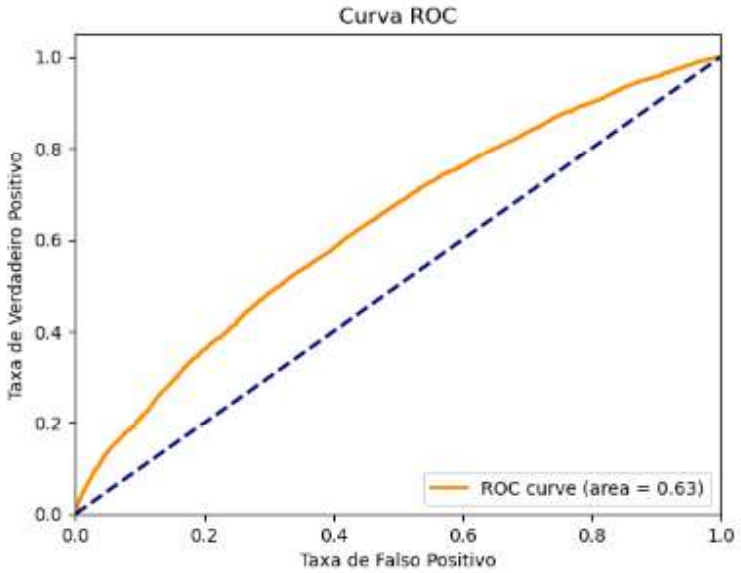


A.22.4. Algoritmo de floresta aleatória

In [82]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.metrics import classification_report, roc_curve, auc

# Crie o modelo Random Forest
rf = RandomForestClassifier(random_state=1)


# Defina um espaço de hiperparâmetros para busca em grade
param_grid = {
    'n_estimators': [100, 300, 600],
    'max_depth': range(5, 15, 30),
    'min_samples_leaf': range(50, 150, 300),
    'min_samples_split': range(50, 150, 300),
}

# GridSearchCV
grid_search = GridSearchCV(rf, param_grid, scoring='accuracy')

# Treina o modelo com a busca em grade
grid_search.fit(X_treino_scaled_bal, Y_treino_bal)

# Melhor modelo encontrado pelo grid_search
melhor_modelo = grid_search.best_estimator_

# Melhores hiperparâmetros encontrados
print("Melhores hiperparâmetros encontrados:", grid_search.best_params_)

# Usa o melhor modelo Random Forest
rf_Y_pred = melhor_modelo.predict(X_teste_scaled)

# Calcula a acurácia com dados de teste
rf_accuracy = melhor_modelo.score(X_teste_scaled, Y_teste) * 100

# Print
print("Acurácia do melhor modelo Random Forest:", rf_accuracy)
print("\n")


# Calcula a matriz de confusão
conf_matrix = confusion_matrix(Y_teste, rf_Y_pred)

# Exibe a matriz de confusão
print("Matriz de Confusão:")
print(conf_matrix)
print(" ")

print('Relatório de Classificação')
print(classification_report(Y_teste, rf_Y_pred, target_names=['Não', 'Sim']))


# Calcular as probabilidades para a classe positiva
y_prob = melhor_modelo.predict_proba(X_teste_scaled)[:, 1]

# Calcula a curva ROC
fpr, tpr, _ = roc_curve(Y_teste, y_prob)
roc_auc = auc(fpr, tpr)

# Plota a curva ROC
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Taxa de Falso Positivo')
plt.ylabel('Taxa de Verdadeiro Positivo')
plt.title('Curva ROC')
plt.legend(loc="lower right")
plt.show()

# Importância das features
importances = melhor_modelo.feature_importances_
feature_names = X_treino.columns

# DataFrame para mostrar as importâncias das features
importances_df = pd.DataFrame({'Feature': feature_names, 'Importance': importances})

# Ordena em ordem decrescente
importances_df = importances_df.sort_values(by='Importance', ascending=False)

# Exibe as features mais importantes
print("As 10 features mais importantes:")
print(importances_df.head(10))
```

```
Melhores hiperparâmetros encontrados: {'max_depth': 5, 'min_samples_leaf': 50, 'min_samples_split': 50, 'n_estimators': 300}
Acurácia do melhor modelo Random Forest: 59.221774589761225


Matriz de Confusão:
[[7252 4779]
 [3521 4802]]

Relatório de Classificação
              precision    recall  f1-score   support

         Não       0.67      0.60      0.64     12031
         Sim       0.50      0.58      0.54      8323

    accuracy                           0.59     20354
   macro avg       0.59      0.59      0.59     20354
weighted avg       0.60      0.59      0.60     20354
```




```
As 10 features mais importantes:
                                Feature  Importance
2   numero_de_procedimentos_laboratoriais    0.151373
4                    numero_de_medicacoes    0.130174
0                                  genero    0.074518
1                     dias_no_hospital     0.065368
5    consultas_ambulatoriais_ano_anterior    0.048694
52             diagnostico_1_Circulatório    0.041210
44               fonte_de_admissao_Outros    0.036123
63                diagnostico_2_Digestivo    0.027664
67               diagnostico_2_Neoplasias    0.024491
64                diagnostico_2_Ferimento    0.022351
```

A.22.5. Algoritmo de máquinas de vetores de suporte

In [83]:
```
### Support Vector Machinne
```

In [84]:
```
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.metrics import classification_report, roc_curve, auc, confusion_matrix


svm = SVC(random_state=1, probability=True)
# svm = SVC(random_state=1, probability=False)

# hiperparâmetros para busca em grade
param_grid = {
#     'C': [0.1, 1, 10],  # Parâmetro de regularização
#     'kernel': ['linear', 'rbf'],  # Tipo de kernel
#     'gamma': ['scale', 'auto', 0.1, 1],  # Parâmetro gamma para kernels 'rbf'
}

# GridSearchCV
grid_search = GridSearchCV(svm, param_grid, scoring='accuracy')

# Treina SVM com a busca em grade
grid_search.fit(X_treino_scaled_bal, Y_treino_bal)

# Melhor modelo encontrado pela busca em grade
melhor_modelo = grid_search.best_estimator_

# Melhores hiperparâmetros encontrados
print("Melhores hiperparâmetros encontrados:", grid_search.best_params_)

# Usa o melhor modelo SVM
svm_Y_pred = melhor_modelo.predict(X_teste_scaled)

# Calcula a acurácia com dados de teste
svm_accuracy = melhor_modelo.score(X_teste_scaled, Y_teste) * 100

# Print
print("Acurácia do melhor modelo SVM:", svm_accuracy)
print("\n")

# Calcula a matriz de confusão
conf_matrix = confusion_matrix(Y_teste, svm_Y_pred)

# Exibe
print("Matriz de Confusão:")
print(conf_matrix)
print(" ")

print('Relatório de Classificação')
print(classification_report(Y_teste, svm_Y_pred, target_names=['Não', 'Sim']))

# Calcula as probabilidades para a classe positiva
y_prob = melhor_modelo.predict_proba(X_teste_scaled)[:, 1]

# Calcula a curva ROC
fpr, tpr, _ = roc_curve(Y_teste, y_prob)
roc_auc = auc(fpr, tpr)

# Plota a curva ROC
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Taxa de Falso Positivo')
plt.ylabel('Taxa de Verdadeiro Positivo')
plt.title('Curva ROC')
plt.legend(loc="lower right")
plt.show()
```

```
Melhores hiperparâmetros encontrados: {}
Acurácia do melhor modelo SVM: 61.295077134715534


Matriz de Confusão:
[[8154 3877]
 [4001 4322]]

Relatório de Classificação
              precision    recall  f1-score   support

         Não       0.67      0.68      0.67     12031
         Sim       0.53      0.52      0.52      8323

    accuracy                           0.61     20354
   macro avg       0.60      0.60      0.60     20354
weighted avg       0.61      0.61      0.61     20354
```