MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE CIÊNCIAS DA SAÚDE DE PORTO ALEGRE
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM TECNOLOGIAS DA INFORMAÇÃO E GESTÃO EM SAÚDE

Gabriel Constantin da Silva

Aprendizado de Máquina Supervisionado para Classificação Automática de Prontuários Médicos da Cardiologia

Porto Alegre
2024

Gabriel Constantin da Silva

Aprendizado de Máquina Supervisionado para Classificação Automática de Prontuários Médicos da Cardiologia

Dissertação no Programa de Mestrado Acadêmico em Tecnologias da Informação e Gestão em Saúde da Universidade Federal de Ciências da Saúde de Porto Alegre.

Orientador: Prof. Dr. Silvio César Cazella

Coorientador: Prof. Dr. Fernando Lucchese

Porto Alegre

2024

Catalogação na Publicação

Constantin da Silva, Gabriel
Aprendizado de Máquina Supervisionado para Classificação Automática de Prontuários Médicos da Cardiologia / Gabriel Constantin da Silva. -- 2024.
144 p. : 30 cm.

Dissertação (mestrado) -- Universidade Federal de Ciências da Saúde de Porto Alegre, Programa de Pós-Graduação em Tecnologias da Informação e Gestão em Saúde, 2024.

Orientador(a): Silvio César Cazella ; coorientador(a): Fernando Lucchese.

1. Inteligência Artificial. 2. Aprendizado de Máquina. 3. Prontuário Médico Eletrônico. I. Título.

Sistema de Geração de Ficha Catalográfica da UFCSPA com os dados fornecidos pelo(a) autor(a).

Gabriel Constantin da Silva

Aprendizado de Máquina Supervisionado para Classificação Automática de Prontuários Médicos da Cardiologia

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Tecnologias da Informação e Gestão em Saúde da Universidade Federal de Ciências da Saúde de Porto Alegre como requisito parcial para a obtenção do título de Mestre em Tecnologias da Informação e Gestão em Saúde.

Orientador: Prof. Dr. Silvio César Cazella

Co-orientador: Prof. Dr. Fernando Lucchese

Aprovada em: _____ de _____ de _____.

BANCA EXAMINADORA:

_____

Profª. Dra. Aline Aver Vanin
Universidade Federal de Ciências da Saúde de Porto Alegre

_____

Profª. Dra. Cássia Trojahn dos Santos
Universidade de Toulouse 2 - Jean Jaurès

_____

Prof. Dr. Marcelo Haertel Miglioranza
Universidade Federal de Ciências da Saúde de Porto Alegre

AGRADECIMENTOS

Agradeço primeiramente ao Doutor Fernando Antônio Lucchese por ter me dado a chance de trabalhar com ele, além de todo o apoio desde o primeiro dia que entrei no Hospital São Francisco, lá no ano de 2012.

Agradeço muito ao Professor Silvio César Cazella pela oportunidade de entrar no Programa de Pós-Graduação em Tecnologias da Informação e Gestão em Saúde e me orientar durante todo o desenvolvimento do meu trabalho. Sou muito honrado aos mestres por me concederem a oportunidade de trabalhar com eles e tentar absorver um pouco dos seus ensinamentos.

Por fim, agradeço também por terem me orientado durante todo o trabalho, com paciência, dedicação e sabedoria.Agradeço muito ao colega e amigo Álvaro Roesler por sua amizade e por ter me apoiado durante todos esses anos de vida profissional, desde o início até o momento. Ele é um grande exemplo para mim, e sua orientação sempre foi fundamental para o meu crescimento pessoal e profissional.

Agradeço aos meus pais e irmãos pelo amor, apoio e incentivo incondicional. Eles sempre estiveram ao meu lado, mesmo nos momentos mais difíceis.

Agradeço a todos os professores e colegas do Programa de Pós-Graduação em Tecnologias da Informação e Gestão em Saúde, pela troca de conhecimento e experiências.

Agradeço aos meus colegas de trabalho Jonathan, Pedro, Ingrid, Vinícius e Gustavo, pela parceria e companheirismo. Além de agradecer a todos os queridos colegas do Hospital São Francisco, como a Doutora Marcela, Fabiana, Marcinha, Fernandez e Rosenara.

Agradeço a todos que contribuíram, de alguma forma, para a realização deste trabalho. Sem o apoio e a colaboração de todos vocês, esta dissertação não seria possível.

Com carinho e gratidão,

Gabriel

RESUMO

Introdução: A gestão dos documentos eletrônicos gerados no atendimento em saúde, como por exemplo, os Prontuários Médicos Eletrônicos (PME), é um desafio devido ao significativo volume de informações relacionadas, como doenças e condições clínicas. Nesse contexto, a classificação automática de textos de PME torna-se viável para processar eficientemente a crescente quantidade desses documentos eletrônicos sem necessitar de intervenção humana. **Objetivo:** Esta dissertação propõe desenvolver modelos de aprendizado de máquina supervisionado para a classificação automática de textos de prontuários eletrônicos de pacientes atendidos em um hospital de cardiologia. **Método:** A pesquisa seguiu as seguintes etapas: 1) Coletar amostras de PME do mundo real; 2) Gerar dados sintéticos de PME com uso Inteligência Artificial Generativa 3) Pré-processar as amostras de textos; 4) Treinar os modelos preditivos utilizando os textos reais e sintéticos pré-processados; 5) Criar amostras sintéticas. 6) Avaliar o desempenho dos modelos com base nas métricas de acurácia, precisão, abrangência e F1–Score; 7) Verificar o modelo com melhor desempenho para classificar os PME. **Resultados:** O modelo obtido através do uso do algoritmo de Regressão Logística e treinado com amostras do mundo real obteve maior valor de acurácia, enquanto o gerado com Support Vector Machine (SVM) obteve maior valor de precisão; Após uso de amostras sintéticas de prontuários criadas com uso de ChatGPT para treinamento, o modelo gerado com uso em Long Short-Term Memory (LSTM) obteve valor de acurácia de 0,88, valor de precisão de 0,97, valor de abrangência de 0,68 e valor de Escore-F1 de 0,80. **Conclusões:** Em termos de desempenho para classificação automática de PME, o LSTM se sobressaiu sem uso de amostras sintéticas, enquanto o SVM apresentou melhor resultado com elas. As técnicas de aprendizado de máquina supervisionado possibilitam a criação de modelos robustos a partir de conjuntos de textos significativos em volume, podendo após uma validação com médicos cardiologistas serem utilizados como apoio no processo de tomada de decisão.

Palavras-chave: Processamento de Linguagem Natural; Aprendizado de Máquina Supervisionado; Prontuários Médicos Eletrônicos.

ABSTRACT

Introduction: The management of electronic documents generated in healthcare, such as Electronic Medical Records (PME), is a challenge due to the significant volume of related information, such as diseases and clinical conditions. In this context, the automatic classification of SME texts becomes viable to efficiently process the growing number of these electronic documents without requiring human intervention. **Objective:** This dissertation proposes to develop supervised machine learning models for the automatic classification of texts from electronic medical records of patients treated in a cardiology hospital. **Method:** The research followed the following steps: 1) Collect samples from real-world SMEs; 2) Generate synthetic SME data using Generative Artificial Intelligence 3) Pre-process text samples; 4) Train predictive models using pre-processed real and synthetic texts; 5) Create synthetic samples. 6) Evaluate the performance of the models based on accuracy, precision, coverage and F1–Score metrics; 7) Check the model with the best performance to classify SMEs. **Results:** The model obtained using the Logistic Regression algorithm and trained with real-world samples obtained a higher accuracy value, while the one generated with Support Vector Machine (SVM) obtained a higher precision value; After using synthetic records samples created using ChatGPT for training, the model generated using Long Short-Term Memory (LSTM) obtained an accuracy value of 0.88, a precision value of 0.97, a coverage value of 0.68 and F-Score value of 0.80. **Conclusions:** In terms of performance for automatic PME classification, LSTM stood out without using synthetic samples, while SVM showed better results with them. Supervised machine learning techniques enable the creation of robust models from sets of significant texts in volume, which can be used after validation with cardiologists to support the decision-making process.

Keywords: Natural Language Processing; Supervised Machine Learning; Electronic Health Records.

LISTA DE FIGURAS

Figura 1 - Exemplo de Prontuário Médico hipotético……16
Figura 2 - Exemplo do resultado de treinamento de KNN……..25
Figura 3 - Exemplo de construção de uma Árvore de Decisão……..28
Figura 4 - Exemplo de representação dos resultados SVM……29
Figura 5 - Arquitetura de MLP……..31
Figura 6 - Representação da unidade de LSTM..……33
Figura 7 - Arquitetura BERT……34
Figura 8 - Etapas de pré-processamento……...36
Figura 9 - Checklist PRISMA……48
Figura 10 - Etapas metodológicas da pesquisa……54
Figura 11 - Etapas seguidas para aquisição das amostras..……..59
Figura 12 - Imagem da tela do sistema Tasy®……..60
Figura 13 - Histograma com a distribuição de diagnósticos…..……..61
Figura 14 - Etapas para obtenção do modelo de PME sintético……....62
Figura 15 - Etapas para criação de PME sintético…….……..65
Figura 16 - Etapas para criação do modelo com Naive Bayes…….……70
Figura 17 - Etapas para criação do modelo com Regressão Logística..……...71
Figura 18 - Etapas para criação do modelo com Árvores de Decisão.……73
Figura 19 - Etapas para criação do modelo com Floresta Aleatória……..74
Figura 20 - Etapas para criação do modelo com KNN…….……...76
Figura 21 - Etapas para criação do modelo com SVM…….……...77
Figura 22 - Etapas para criação do modelo com MLP…….……...79
Figura 23 - Etapas para criação do modelo com LSTM…….……....80
Figura 24 - Criação do modelo com LSTM com amostras sintéticas……....82

LISTA DE QUADROS

Quadro 1 - Critérios de Inclusão de Artigos……49
Quadro 2 - Critérios de Exclusão de Artigos……..49
Quadro 3 - Sumarização dos estudos relacionados…….51
Quadro 4 - Sumarização dos comandos e resultados……..62

LISTA DE TABELAS

Tabela 1 - Proporção dos diagnósticos dos prontuários médicos…….....68
Tabela 2 - Comparação entre modelos por métricas de avaliação…….....84

LISTA DE ABREVIATURAS E SIGLAS

BOW: *Bag-of-words*

CNN: *Convolutional Neural Network*

IA: Inteligência Artificial

KNN: *K-Nearest Neighbors*

LR: *Logistic Regression*

LSTM: *Long Short-Term Memory*

ML: *Machine Learning*

MLP: *Multilayer Perceptron*

PLN: Processamento de Linguagem Natural

PME: Prontuários Médicos Eletrônicos

RNN: *Recurrent Neural Network*

SVM: *Support Vector Machine*

SUMÁRIO

1. INTRODUÇÃO.....15
1.1 Questão de pesquisa.....17
1.2 Justificativa da pesquisa.....17
1.3 Contribuições esperadas.....17
1.4 Estrutura adotada nesta Dissertação.....18
2. OBJETIVOS 19
2.1 Objetivo Geral 19
2.2 Objetivos Específicos 19
3. REFERENCIAL TEÓRICO.....20
3.1 Prontuários Médicos Eletrônicos.....20
3.2 Aprendizado de Máquina Supervisionado 22
3.2.1 Naive Bayes 23
3.2.2 Regressão Logística.....23
3.2.3 *K-Nearest Neighbors* 24
3.2.4 Árvores de Decisão.....26
3.2.5 Floresta Aleatória.....28
3.2.6 *Support Vector Machines* 29
3.2.7 Perceptron de Multicamadas.....31
3.2.8 Long Short Term Memory 32
3.2.9 Bidirectional Encoder Representations for Transformers.....34
3.3 Processamento de Linguagem Natural.....35
3.3.1 Definição de *Corpus* 35
3.3.2 Pré-processamento de texto.....36
3.3.2.1 Stopwords.....36
3.3.2.2 Tokenização.....37
3.3.2.3 Stemização e Lematização.....37
3.3.2.4 Bag-of-Words.....37
3.3.2.5 Label Encoder.....38
3.4 Classificação de textos dos prontuários.....38
3.4.1 Classificação usando Naive Bayes 38
3.4.2 Classificação usando Regressão Logística.....39
3.4.3 Classificação usando KNN.....40

3.4.4 Classificação usando Árvores de Decisão……40
3.4.5 Classificação usando Floresta Aleatória……41
3.4.6 Classificação usando SVM……....42
3.4.7 Classificação usando MLP……....42
3.4.8 Classificação usando LSTM……43
3.4.9 Classificação usando BERT……43
3.5 Métricas de Avaliação dos Modelos de Classificação……44
3.5.1 Acurácia……...44
3.5.2 Precisão……...44
3.5.3 Abrangência (Recall)……..45
3.5.4 Escore F1..……...45
4. REVISÃO RÁPIDA DA LITERATURA……..47
4.1 Protocolo da Revisão……47
4.2 Questão de Pesquisa……47
4.3 Consulta das bases de dados……..48
4.4 Análise dos estudos obtidos……49
4.5 Estudos selecionados……50
5. MATERIAIS E MÉTODOS……...53
5.1 Caracterização da Pesquisa……53
5.2 Delineamento da Pesquisa…… ……..53
5.3 Linguagem Python……..55
5.4 Scikit-Learn……..55
5.5 Natural Language Tool Kit……55
5.6 Dados Sintéticos……56
5.7 ChatGPT……..56
5.8 Dados Sintéticos para Prontuários Médicos……..56
5.8.1 Aumento de Dados……57
5.8.2 Melhoria da Generalização……...57
5.8.3 Mitigação de *Overfitting*……57
5.8.4 Redução da Dependência do Volume de Dados Reais……57
5.9 Comitês de Ética……57
5.10 Material de Pesquisa……58
5.10.1 Aquisição das Amostras Reais de PMEs……..58

5.10.2 Análise Estatística das Amostras de PME……61
5.11 Geração do dataset sintético de PME……...61
5.11.1 Interação com o ChatGPT para criação do modelo PME……...62
5.11.2 Interação com o ChatGPT para geração de PMEs……..65
5.12 Pré-Processamento das Amostras……66
5.12.1 Anonimização dos PME Reais……...66
5.12.2 Limpeza das amostras de PME……..66
5.12.3 Transformação dos PME para vetores……..67
6. ANÁLISE E DISCUSSÃO DOS RESULTADOS……..70
6.1 Análise estatística dos diagnósticos dos registros eletrônicos……70
6.2 Criação dos modelos classificadores de prontuários……71
6.2.1 Criação do modelo utilizando Naive Bayes……71
6.2.2 Criação do modelo utilizando Regressão Logística……...73
6.2.3 Criação do modelo utilizando Árvore de Decisão...……...74
6.2.4 Criação do modelo utilizando Floresta Aleatória……...76
6.2.5 Criação do modelo utilizando KNN…..……..77
6.2.6 Criação do modelo utilizando SVM……..79
6.2.7 Criação do modelo utilizando MLP……...79
6.2.8 Criação do modelo utilizando LSTM……82
6.2.9 Criação do modelo utilizando LSTM com amostras sintéticas…….....83
6.3 Comparação de desempenho entre modelos……83
7. CONCLUSÃO……...85
7.1 Limitações da Pesquisa……85
7.2 Produções da Pesquisa……86
7.3 Trabalhos Futuros……...87
8. REFERÊNCIAS BIBLIOGRÁFICAS……88
APÊNDICE A - Anonimização dos Prontuários Médicos Eletrônicos.....99
APÊNDICE B - Limpeza das amostras de Prontuários Médicos Eletrônicos.....100
APÊNDICE C - Script utilizando Naive Bayes……...101
APÊNDICE D - Script com utilização de Regressão Logística.....103
APÊNDICE E - Script com utilização de Árvore de Decisão.....105
APÊNDICE F - Script com utilização de Floresta Aleatória.....107
APÊNDICE G - Script com utilização de KNN.....109
APÊNDICE H - Script com utilização de SVM.....111

APÊNDICE I - Script com utilização de MLP.....113
APÊNDICE J - Script com utilização de LSTM.....115
APÊNDICE L - Utilização de LSTM em conjunto com amostras sintéticas.....118
APÊNDICE M - Artigo Científico para submissão…….....130
ANEXO I - Aprovação do Projeto pelo Comitê de Ética em Pesquisa UFCSPA.....121
ANEXO II - Aprovação do Projeto pelo Comitê de Ética em Pesquisa ISCMPA.....122
ANEXO III - Pôster Apresentado no 2º Congresso CI-IA saúde.....123
ANEXO IV - Pôster Apresentado no 3º Congresso UFCSPA……..124
ANEXO V - Conjunto de dados sintéticos publicado no site Zenodo.....125
ANEXO VI - Instruções Para Submissão à Revista Cadernos De Saúde Pública...126

1. Introdução

A classificação automática de textos sempre foi objeto de pesquisa, ganhando cada vez mais importância devido ao crescimento da quantidade de documentos eletrônicos sendo gerada diariamente (ALBERTO et al., 2009). Para isso, faz-se necessário a utilização de tecnologias como aprendizado de máquina (do inglês, *machine learning*), capazes de lidar com esse volume e com a complexidade dos textos (SINGRU et al., 2020). Esse crescimento significa também que as instituições de saúde de todos os portes precisam estar preparadas para lidar com essa quantidade significativa de dados e evitar perder informações importantes acerca dos pacientes atendidos por elas (YANG et al., 2018).

Um Prontuário Médico Eletrônico (PME) é um registro digital sistemático das informações de saúde de um indivíduo, mantido por provedores de cuidados de saúde. Esses registros podem incluir uma gama de dados, incluindo características demográficas do paciente, histórico médico, resultados de exames, medicamentos, histórico de vacinação, alergias, dados de imagem radiológica e resultados de laboratório.

Os textos contidos em PMEs apresentam um grande desafio para serem avaliados por algoritmos de aprendizado de máquina, pois a natureza dos dados é muito diversa, incluindo dados numéricos como índice de massa corpórea, data de nascimento ou dia da admissão em formatos diversos, valores categóricos como gênero, ou raça do paciente, além de texto livre para descrever as evoluções de saúde ou notas de alta (SHICKEL et al., 2018).

No contexto da cardiologia, por exemplo, um PME pode conter informações como resultados de eletrocardiogramas, imagens de ultrassom do coração (ecocardiogramas), descrição de cirurgias, etc. Essas informações são frequentemente semi-estruturadas; elas podem incluir dados numéricos (como leituras de pressão arterial ou níveis de colesterol), mas também notas de texto dos médicos, imagens médicas e outros tipos de dados que não se encaixam facilmente em uma tabela estruturada (RAHMAN et al., 2020).

Os PMEs geram uma quantidade enorme de dados, conforme ilustrado no PME hipotético na Figura 1. Como pode-se observar neste exemplo hipotético, o PME é composto por um texto no qual as informações sobre o paciente estão organizadas em seções.

Cada paciente pode ter centenas ou até milhares de entradas de dados, e um hospital pode ter milhões de pacientes. Analisar manualmente todos esses dados seria um processo demorado e praticamente impossível. Os dados médicos podem ser complexos e requerem conhecimento especializado para interpretá-los corretamente, e erros de interpretação podem levar a conclusões erradas sobre o estado de saúde de um paciente.

Paciente: João da Silva
Idade: 65 anos
Sexo: Masculino
Data da última consulta: 15 de abril de 2024

Histórico médico:
- Hipertensão diagnosticada em 2005
- Diabetes tipo 2 diagnosticada em 2010
- Infarto do miocárdio em 2018

Medicamentos atuais:
- Lisinopril 10mg diariamente para hipertensão
- Metformina 500mg duas vezes ao dia para diabetes

Alergias:
- Penicilina

Notas do médico:
- O paciente apresenta dor no peito ocasional durante o exercício. Recomendado para teste de estresse.
- O controle da glicose no sangue tem sido inconsistente. Considerar ajuste da medicação.

Resultados de exames:
- Eletrocardiograma (12 de abril de 2024): Ritmo sinusal regular, sem evidência de isquemia
- Hemoglobina A1c (12 de abril de 2024): 8,2% (objetivo: <7,0%)

Imagens médicas:
- Ecocardiograma (15 de abril de 2024): Função ventricular esquerda ligeiramente reduzida, fração de ejeção estimada em 50%.

Próximos passos:
- Teste de estresse agendado para 22 de abril de 2024
- Consulta de acompanhamento em 29 de abril de 2024 para discutir os resultados e possíveis ajustes na medicação.

Figura 1 - Exemplo de PME hipotético.

Fonte: Próprio Autor.

A classificação automática pode ser feita usando inteligência artificial (IA), através do aprendizado e identificação de padrões nos prontuários, classificando-os de forma rápida e precisa. Por exemplo, um modelo pode ser treinado em um

conjunto de PEs no qual o atributo objetivo é se o paciente tem ou não uma doença cardíaca, e qual seria ela. O modelo aprenderia a identificar padrões nos dados do PME que estão associados à presença de doença cardíaca. Depois de treinado, o modelo poderia ser usado para classificar automaticamente novos PMEs, prevendo a probabilidade de cada paciente ter uma doença cardíaca.

1.1 Questão de pesquisa

A questão de pesquisa a ser respondida por este estudo é a seguinte: Qual modelo desenvolvido com base em diferentes algoritmos de aprendizado de máquina supervisionado apresenta, com base em métricas, desempenho superior para classificação automática do diagnóstico de doença, a partir do uso de técnicas de Processamento de Linguagem Natural (PLN) do conteúdo de PME?

1.2 Justificativa da pesquisa

A partir desse trabalho espera-se colaborar para o avanço tecnológico na área da saúde ao permitir a classificação automatizada de PME, facilitando o processamento de dados dos pacientes, melhorando a qualidade do atendimento e reduzindo os custos médicos.

Primeiramente, a grande quantidade de dados em formato de texto exige grande esforço dos profissionais de saúde que os processam manualmente. Além disso, a informação contida nos textos pode ser complexa e ambígua, exigindo interpretação cuidadosa. A falta de padronização nos registros médicos também pode dificultar a compreensão e a análise dos dados. Portanto, soluções tecnológicas como a classificação automatizada de PME podem auxiliar na superação desses desafios.

1.3 Contribuições esperadas

Espera-se fornecer as seguintes contribuições a partir da realização deste estudo: 1) Melhoria na eficiência do cuidado ao paciente através do uso de Machine Learning para classificar PME, ao identificar rapidamente pacientes em risco e priorizar seu atendimento. Isso pode levar a intervenções mais oportunas e melhorar

os resultados de saúde. 2) Apoio à tomada de decisão clínica pois os sistemas inteligentes podem fornecer insights valiosos que apoiam a tomada de decisão clínica ao identificar padrões sutis nos dados que podem não ser facilmente perceptíveis para os humanos.

Por ser um programa de pós-graduação interdisciplinar, este trabalho promove a colaboração entre os campos da tecnologia da informação e da saúde. Isso pode levar a soluções mais integradas e holísticas para os desafios na prestação de cuidados de saúde.

1.4 Estrutura adotada nesta Dissertação

Essa pesquisa está dividida em oito capítulos, incluindo Introdução e Conclusão. O capítulo 2 apresenta os objetivos desta pesquisa, onde está descrito o Objetivo Principal e os Objetivos Específicos. No capítulo 3 estão descritos e explicados os conceitos abordados durante a pesquisa. No capítulo 4 é apresentada a revisão da literatura realizada durante a pesquisa.

No capítulo 5 estão descritos os materiais utilizados na pesquisa, que foram os PME reais e os PME sintéticos, além das técnicas utilizadas para preparar as amostras, treinar os modelos e verificar seus desempenhos. No capítulo 6 são apresentados os resultados obtidos a partir da execução dos códigos. No capítulo 7 é realizada a discussão sobre a importância destes resultados para atender o objetivo principal da pesquisa. No capítulo 8 é apresentada a conclusão e fechamento da pesquisa.

2. Objetivos

2.1 Objetivo Geral

Esta pesquisa visa desenvolver modelos de aprendizado de máquina para a classificação automática de textos de prontuários eletrônicos de pacientes atendidos em um hospital de cardiologia, com base no seu diagnóstico. Para isso, serão utilizados algoritmos de aprendizado de máquina supervisionado.

2.2 Objetivos Específicos

Os objetivos específicos definidos para alcançar o objetivo geral desta pesquisa são os que seguem:

1. Estudar a literatura sobre algoritmos de Aprendizado de Máquina Supervisionado.
2. Selecionar os algoritmos de Aprendizado de Máquina Supervisionado a serem utilizados no estudo.
3. Coletar amostras de Prontuários Eletrônicos de pacientes.
4. Pré-processar com base em PLN os conteúdos dos Prontuários Médicos Eletrônicos de pacientes da Cardiologia coletados.
5. Desenvolver um dataset/conjunto de dados para treinamento.
6. Treinar modelos para classificação automática com base nos algoritmos selecionados.
7. Avaliar o desempenho dos modelos obtidos com base nas métricas clássicas da área de Aprendizado de Máquina.

3. Referencial Teórico

Este capítulo apresenta o referencial teórico utilizado para a escrita desta dissertação. A seguir são apresentados e explicados os principais conceitos abordados durante o trabalho.

3.1. Prontuários Médicos Eletrônicos

O Prontuário Médico Eletrônico (PME) do paciente é o conjunto de documentos médicos padronizados e ordenados, destinados ao registro dos cuidados profissionais prestados ao paciente pelos serviços de saúde. Deve ser preenchido sempre que houver necessidade de inserir algum dado e relatar informações pertinentes, em qualquer setor em que o paciente se encontrar, por qualquer profissional que o estiver atendendo. O Código de Ética Médica, aprovado pela resolução n.º1246/88, estabelece, no artigo 69, a obrigatoriedade de elaboração de prontuário para cada paciente, bem como trata dos princípios éticos da profissão envolvidos nesse registro (Conselho Federal de Medicina, 1988).

De acordo com o Conselho Federal de Medicina, a partir da Resolução 1638/2002 - Artigo 1, o prontuário médico, ou prontuário do paciente, é definido como: “documento único constituído de um conjunto de informações, sinais e imagens registradas, geradas a partir de fatos, acontecimentos e situações sobre a saúde do paciente e a assistência a ele prestada, de caráter legal, sigiloso e científico, que possibilita a comunicação entre membros da equipe multiprofissional e a continuidade da assistência prestada ao indivíduo.”

Para Possari (2005) o prontuário eletrônico é um registro online que consiste no conjunto dos documentos referentes ao estado de saúde de um paciente, com a finalidade de facilitar a assistência médica e documentar a história de exames, tratamentos e diagnósticos. Os PME se encaixam na categoria de dados não estruturados, pois possuem campo para inserção de dados não estruturados, como é o caso de texto livre preenchido pelos profissionais de saúde que atendem o paciente enquanto está sendo atendido pela instituição de saúde (CAWSEY; WEBBER; JONES, 1997).

A adoção dos PME teve um crescimento exponencial tanto em hospitais quanto consultórios particulares. Apesar de terem sido criados para melhorar a

eficiência operacional das atividades relacionadas ao cuidado com a saúde, os PME abrem possibilidades de aproveitamento para a área de informática médica como extração de conceitos médicos, suporte de tomada de decisões, entre outras aplicações voltadas para a saúde (SHICKEL et al., 2018).

Muitas informações como sumários de admissão e notas médicas sobre procedimentos permanecem na forma textual, e para que esses dados possam ser analisados, precisam ser convertidos de texto para uma forma mais estruturada. O potencial de uso destes dados é muito grande, principalmente para avaliação de qualidade, inteligência de negócio, e pesquisas médicas (WEINTRAUB, 2019).

Levando em consideração que o uso do meio eletrônico diminui a necessidade da comunicação face a face, as funções do PME deveriam aperfeiçoar a troca de informações em relação ao sistema que não utiliza tecnologias da informação. No entanto, o registro que deveria condensar e otimizar informações acaba gerando informações de difícil acesso e interpretação, dificultando o estabelecimento do trabalho em equipe multidisciplinar e interprofissional (KOSSMAN; BONNEY, 2013).

Para a utilização apropriada do PME é necessário que os profissionais de saúde estejam comprometidos com a qualidade e atenção no preenchimento destes registros eletrônicos de saúde. Para profissionais da Enfermagem, por exemplo, existe um guia de recomendações publicado pelo Conselho Federal de Enfermagem (Conselho Federal de Enfermagem, 2016), que serve de guia para os registros, buscando garantir a qualidade das informações. Este guia traz a fundamentação legal para os registros de enfermagem, regras e exemplos do que deve ser anotado, considerando diversos cenários de atendimento ao paciente.

O Conselho Federal de Medicina (CFM) possui resoluções legais (CFM 1605/2000; CFM 1638/2002; CFM 1821/2007) sobre o papel do médico no preenchimento dos prontuários, sobre o que pode ou não constar, quando deve-se anotar algo e como algumas informações devem aparecer, mas não existe um documento único que guie essa prática. As resoluções e o Código de Ética Médica orientam sobre o que deve/pode constar nos prontuários e questões legais, como devem ser preenchidos, e armazenados os prontuários.

O valor dos PME foi claramente demonstrado durante a pandemia de coronavírus, quando sistemas de saúde em todo o mundo utilizaram dados dos pacientes para compreender e responder à crise. Uma das principais lições

aprendidas foi a importância da integração de dados entre diferentes sistemas de saúde para obter maior poder estatístico e identificar padrões clínicos e associações que sejam mais gerais em diferentes populações de pacientes. No entanto, os PME também apresentam muitos desafios, como o uso desses sistemas de registro por médicos, ou pesquisadores, para navegar pelas diferentes funcionalidades, encontrar e analisar informações relevantes, tudo isso através de procedimentos redundantes para cumprir seu objetivo (HOLMES et al., 2021).

O sistema de registro eletrônico utilizado no Hospital São Francisco da Irmandade Santa Casa de Misericórdia de Porto Alegre é o TASY® (Koninklijke Philips N.V., 2004 – 2017). No capítulo de Materiais e Métodos serão abordados detalhes sobre o sistema TASY®, a fim de tornar claras a sua organização e a função de registro das notas clínicas referentes aos pacientes.

3.2 Aprendizado de Máquina Supervisionado

O aprendizado de máquina supervisionado (do inglês, *Supervised Machine Learning*) depende de dados rotulados previamente por humanos (supervisão humana) para que a máquina possa aprender através de exemplos, e depois consiga realizar previsões utilizando amostras inéditas. Esse aprendizado engloba em duas categorias diferentes: Regressão e Classificação. A regressão trabalha com dados numéricos contínuos, definindo relacionamento entre variáveis dependentes e independentes. Por sua vez, a classificação envolve dados binários ou de múltiplas classes, para ensinar a máquina a fazer estas distinções (GUPTA et al., 2022).

No aprendizado de máquina supervisionado voltado para análise textual, precisamos de um conjunto de dados com rótulos atribuídos por humanos com conhecimento especializado no assunto do texto. Isso é necessário para ser usado no treino correto do algoritmo, sendo que a classificação de texto pode ser realizada com base no tópico ou no gênero textual (BHANOT, 2019). Alguns exemplos de algoritmos de aprendizado supervisionado como Máquinas de Vetores Suporte, ou *Support Vector Machines* (SVM) em inglês, Regressão Logística, K-Vizinhos mais Próximos, ou *K-Nearest Neighbors* (KNN) em inglês, e *Naive Bayes* têm sido usados para categorização de texto (MADAHIAN, 2014).

3.2.1 *Naive Bayes*

O *Naive Bayes* (NB) é um classificador probabilístico que pressupõe uma independência entre os atributos dos dados. É popular para aplicações de classificação de textos, por exemplo, para identificação de spams. Ele se baseia no Teorema de Bayes e assume independência condicional entre os recursos, ou seja, uma palavra não teria influência sobre outras. Apesar dessa suposição simplificadora (daí o "ingênuo" ou "Naive"), o Naive Bayes muitas vezes funciona bem na prática, especialmente para problemas de classificação de texto (ERTEL, 2017).

Uma vantagem desse modelo é que, por assumir independência entre os atributos, ele é capaz de fazer o treinamento de um modelo com um número pequeno de amostras, porém o modelo não consegue capturar as interações entre os atributos. Esse modelo simples também pode trabalhar com dados que tenham vários atributos, servindo como um bom modelo de base (HARRISON, 2020).

O modelo NB funciona bem em muitos cenários do mundo real, principalmente considerando amostras de dados “sujas”, sem nenhum pré-processamento. O Teorema de Bayes permite atualizar a estimativa de probabilidade para um evento à medida que mais evidências ou informações se tornam disponíveis (BIAN, 2023).

A equação 1 apresenta a fórmula de cálculo da probabilidade de um dado pertencer a uma determinada classe (ERTEL, 2017):

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)} \quad (1)$$

Onde P(A|B) é a probabilidade do evento A ocorrer dado que B é verdadeiro; P(B|A) é a probabilidade do evento B ocorrer dado que A é verdadeiro; P(A) é a probabilidade do evento A; P(B) é a probabilidade do evento B.

3.2.2 Regressão Logística

A Regressão Logística (RL) é um modelo estatístico utilizado para modelar a probabilidade de um evento ocorrer como uma função de variáveis independentes.

Apesar do nome, a regressão logística é comumente aplicada a problemas de classificação binária, na qual o objetivo é predizer se uma instância pertence a uma das duas classes (NUSINOVICI, 2020).

A RL utiliza uma função de perda, como a *log loss*, para avaliar a diferença entre as probabilidades preditas e as classes reais, de forma a ajustar os pesos e minimizar essa perda. A equação 2 apresenta a fórmula de cálculo da probabilidade de um dado pertencer a uma determinada classe (AGGARWAL, 2018).

$$\hat{y}_i = P(y_i = 1) = \frac{1}{1 + \exp(-\overline{W} \cdot \overline{X_i})} \quad (2)$$

Onde $P(y_1=1)$ é a probabilidade de pertencer à classe 1; $W = (w_1,w_2,\ldots,w_{n)}$ são os pesos associados às características $X = (x_1,x_2,\ldots,x_n)$.

3.2.3 *K-Nearest Neighbors*

O algoritmo KNN (K-Nearest Neighbor, ou K Vizinhos Mais Próximos, em português) faz classificações com base na distância de uma amostra até outra, em função do número de categorias (k) determinadas no treinamento. É chamado de aprendizado baseado em instâncias (*instance-based learning*), pois não há parâmetros para aprender. O modelo pressupõe que a distância entre uma nova amostra e outra representante da categoria é suficiente para fazer uma inferência sobre a classe da nova amostra (HARRISON, 2020).

Na figura 2 é apresentado o resultado do treinamento onde "*training class*" nos valores de "0" ou "1" correspondem às amostras de treino das respectivas classes. "test pred" 0 e 1 correspondem às previsões feitas pelo modelo, O eixo X representa classes de amostras, e o eixo Y representa classe alvo (MULLER; GUIDO, 2017).

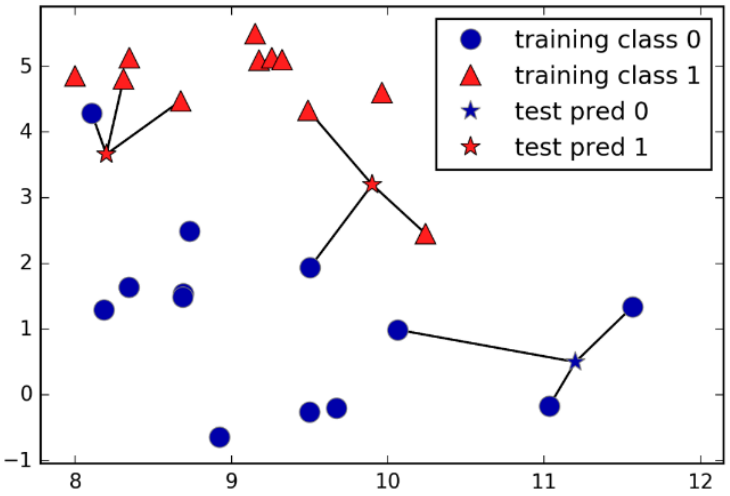


Figura 2 - Resultado da classificação por KNN.

Fonte: (MULLER; GUIDO, 2017)

O KNN é um dos modelos mais populares e extensivamente usados, porém possui algumas limitações como falta de capacidade de atribuir diferentes pesos entre as amostras, além da alta dependência sobre o conjunto de dados, levando a necessidade de treinar novamente um modelo mesmo com uma mudança pequena nos dados de treino (YONG; YOUWEN; SHIXIONG, 2009).

O valor de k determina o número de grupos a ser usados para dividir as amostras e por isso possui uma influência crucial no desempenho do algoritmo. Existem muitas abordagens para tentar encontrar o número adequado para cada caso de aplicação do algoritmo (PATRICK; FICHER, 1970),

Na parte central do funcionamento do algoritmo KNN está a classificação de uma amostra não rotulada de acordo com o agrupamento mais próximo considerando características semelhantes entre as amostras. A distância entre o vetor do texto a ser classificado e os vetores de todos os exemplos de treinamento é calculada. A distância euclidiana é uma escolha comum, mas outras métricas de distância também podem ser usadas. As classes dos KNN são consideradas, e o exemplo de teste é classificado com base na classe mais frequente entre esses vizinhos. Pode-se utilizar uma votação simples ou ponderada, onde vizinhos mais próximos têm mais influência (HARRISON, 2020). Uma limitação encontrada ao se utilizar tal técnica para análise de textos é que em amostras desbalanceadas o KNN tende a classificar a amostras de acordo com a categoria mais presente (QIAN LI et al., 2020).

3.2.4 Árvores de Decisão

Classificador de Árvore de Decisão (do inglês, *Decision Tree Classifier*) é um algoritmo de aprendizado de máquina que pode ser usado para classificação e regressão. Em classificação, uma Árvore de Decisão prediz a classe de um novo dado com base em uma série de regras de decisão. Em regressão, uma Árvore de Decisão estima o valor de um novo dado com base em uma série de regras de decisão. As vantagens deste modelo incluem suporte para dados não numéricos, pouca preparação dos dados, suporte para lidar com relacionamentos não lineares, e importância dos atributos fácil de explicar (HARRISON, 2020).

As Árvores de Decisão são fáceis de entender e implementar, o que as torna uma escolha popular para muitos problemas de aprendizado de máquina. Elas também podem ser muito eficazes, especialmente quando os dados são grandes e complexos (PRIYANKA; KUMAR, 2020). Um componente muito importante nesse modelo de classificador é o chamado algoritmo, que pode variar de acordo com o contexto do problema que estamos lidando. A seguir estão algumas considerações sobre algoritmos utilizados, segundo PRIYAM et. al, 2013:

- ID3: O algoritmo *Iterative Dichotomiser* 3 foi desenvolvido em 1986 por Ross Quinlan. O algoritmo cria uma árvore de múltiplos caminhos, encontrando para cada nó (o que é conhecido como "*greedy*", ou ganancioso em português) a característica categórica que resultará no maior ganho de informação para alvos categóricos. As árvores são construídas até seu tamanho máximo e, em seguida, geralmente é aplicada uma etapa de poda da árvore para limitar seu tamanho e assim melhorar a capacidade da árvore de generalizar para dados não vistos.
- C4.5: O algoritmo C4.5 é o algoritmo sucessor do ID3 e removeu a restrição de que as características devem ser categóricas, definindo dinamicamente um atributo discreto (com base em variáveis numéricas) que divide o valor do atributo contínuo em um conjunto discreto de intervalos. O C4.5 converte as árvores treinadas (ou seja, a saída do algoritmo ID3) em conjuntos de regras do tipo "se-então". A precisão de cada regra é então avaliada para determinar a ordem em que devem ser aplicadas. A poda é feita removendo a condição prévia de uma regra se a precisão da regra melhorar sem ela.

- CART: O algoritmo Classification and Regression Trees é muito semelhante ao C4.5, mas difere no sentido de que suporta variáveis-alvo numéricas (regressão) e não calcula conjuntos de regras. O CART constrói árvores binárias usando a característica e o limiar que resultam no maior ganho de informação em cada nó.

Algumas vantagens da utilização da técnica de Árvores de Decisão são destacadas a seguir (KOTSIANTIS, 2013):

- São simples de entender e interpretar, pois permitem visualização das regras usadas para tomada de decisão de “folha” da árvore.
- Requerem pouca preparação dos dados, enquanto outras técnicas exigem pré-processamento das amostras para lidar com valores em branco, por exemplo.
- Têm capacidade de lidar com problemas de saída múltipla (não binário).

Apesar de todas as vantagens descritas anteriormente, o modelo Árvores de Decisão pode criar modelos muito complexos, o que pode causar dificuldade de o modelo generalizar sua tomada de decisão para dados inéditos, fato que é conhecido como *overfitting*. Isso é geralmente observado em conjunto de dados com muitas características, e poucas amostras, por isso deve-se considerar o uso de técnicas de redução de dimensionalidade a fim de obter melhores resultados (BREIMAN et al., 1984).

A figura 3 apresenta um exemplo de construção de uma Árvore de Decisão com seis níveis de decisão. Esta imagem permite compreender como o algoritmo utiliza regras específicas para a tomada de decisão e, ao final, chegar à classificação final (PEDREGOSA et al., 2011).

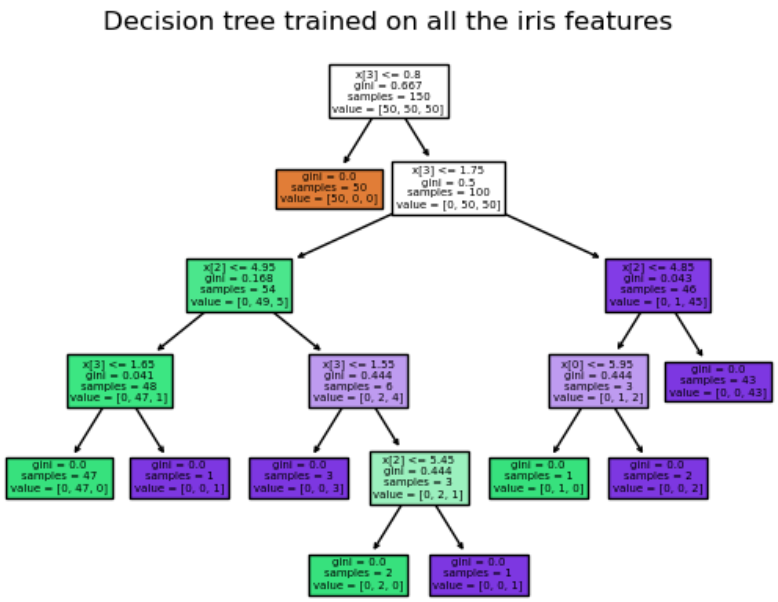


Figura 3 - Exemplo de construção de uma Árvore de Decisão.

Fonte: (PEDREGOSA et al., 2011)

3.2.5 Floresta Aleatória

Uma Floresta Aleatória (*Random Forest*) é um classificador formado por uma coleção de classificadores de Árvore de Decisão, vistos no tópico anterior. Esse tipo de modelo usa a técnica de *bagging*, que envolve a geração de várias amostras de treinamento a partir do conjunto de dados original (BREIMAN, 2001).

Para fazer previsões no final, as previsões de cada árvore individual são combinadas por meio de votação (no caso de problemas de classificação) ou média (no caso de problemas de regressão). A classe ou valor que recebe mais votos ou a média é escolhida como a previsão final da Floresta Aleatória (SEGAL; XIAO, 2011).

A ideia das florestas aleatórias é criar uma “floresta” de árvores de decisão treinadas em diferentes colunas dos dados de treinamento. Se cada árvore tiver uma chance melhor que 50% de fazer uma classificação correta, terá incorporada sua predição. A floresta aleatória tem sido uma excelente ferramenta tanto para classificação como para regressão, embora tenha cedido espaço para as árvores com gradiente de *boosting* recentemente (HARRISON, 2020).

Comparando o uso de Florestas Aleatórias ao uso de Árvores de Decisão de

forma individual, encontra-se maior robustez e eficiência para variadas tarefas de aprendizado de máquina. Além disso, essa técnica é menos suscetível a super adequação (*overfitting)* em comparação com uma única árvore de decisão devido à combinação de várias árvores treinadas de forma independente (HASTIE et. al, 2009).

O ponto mais forte desse modelo é a habilidade de descobrir interações significativas e efeitos não-lineares dos preditores. Por sua vez, por terem funcionamento mais complexo, torna-se mais difícil interpretar quais variáveis estão influenciando mais fortemente na predição (RIGATTI, 2017).

3.2.6 *Support Vector Machines*

Uma *Support Vector Machine* (SVM) ou Máquina de Vetores de Suporte, em português, é um algoritmo que faz a adequação de uma linha (ou plano) entre as diferentes classes de modo a maximizar a distância da linha até os pontos das classes. Dessa forma, ela tenta encontrar uma separação robusta entre as classes. Os vetores de suporte (*support vectors*) são os pontos da fronteira do hiperplano divisor (HARRISON, 2020). A figura 4 mostra um exemplo de SVM para um problema de classificação binário. A classe determinada pelos vetores de suporte aparecem circulados na figura (ERTEL, 2017).

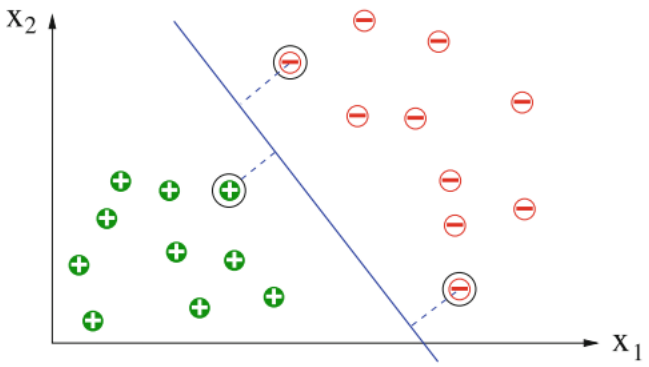


Figura 4 - Exemplo do resultado do uso de SVM para classificação.
Fonte: (ERTEL, 2017)

Essa técnica foi introduzida em 1995 como uma nova forma de lidar com problemas de classificação binária. O modelo combina três ideias: hiperplanos

otimizados; convolução de produtos escalares; margens não-rígidas, permitindo erros no conjunto de treino (CORTES, VAPNIK, 1995).

O algoritmo SVM possui a habilidade de aprender independentemente da dimensionalidade das características, já que calculam a complexidade tendo como base a margem que separa os dados e não com base no número de características. Isso significa que ele consegue generalizar mesmo quando há muitos parâmetros na amostra (JOACHIMS, 1998). Esse modelo tem se mostrado uma ferramenta adequada para muitos problemas de processamento de dados, com ênfase em Classificação e Regressão (VAPNIK, 1998).

Citam-se a seguir algumas vantagens deste modelo (XUE et. al, 2021):

- Sua solução é expressa em termos de um subconjunto das observações, os chamados vetores de suporte, sendo, portanto, intrinsecamente uma solução esparsa;
- São regularizados de uma maneira inteligente e altamente eficiente, pois seguem o princípio de minimização de risco estrutural;
- Eles apresentam um desempenho muito bom em problemas mal colocados (recursos de entrada de alta dimensão, baixo número de amostras de treinamento);

De acordo com (JOACHIMS, 1998), SVM se comporta como um aprendiz universal, ou seja, pode aprender qualquer tarefa ou conceito sem ser explicitamente programado. Tal algoritmo seria capaz de generalizar a partir de uma pequena quantidade de dados para resolver novos problemas, e seria capaz de aprender de múltiplas fontes de informação, incluindo texto, imagens e áudio.

O modelo SVM busca encontrar o hiperplano de decisão que melhor separa as classes no espaço de características. Esse hiperplano é escolhido de forma a maximizar a margem entre as instâncias de treinamento e o hiperplano. Em alguns casos, é utilizado um *kernel* para mapear o espaço de características original em um espaço de características de maior dimensão, diferentes *kernels* podem ajudar a lidar com relações não lineares entre as características (HARRISON, 2020).

Outra propriedade notável desse modelo é ser independente da dimensionalidade do número de características avaliadas. Isso significa que é possível generalizar sua capacidade de previsão, mesmo com um conjunto de dados

muito grande. Além disso, permite o ajuste automático de parâmetros sem a necessidade de validação cruzada, que é um processo de alto custo computacional (JOACHIMS, 1998).

3.2.7 Perceptron de Multicamadas

O Multilayer Perceptron (MLP), ou Perceptron de Multicamadas em português, é uma arquitetura de rede neural artificial que faz parte da família de modelos de redes neurais *feedforward*. É composto por várias camadas de neurônios, incluindo uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída (AGGARWAL, 2018). A figura 5 mostra ilustra a arquitetura de MLP; onde INPUT LAYER (Camada de Entrada); HIDDEN LAYER (Camada Oculta); OUTPUT LAYER (Camada de Saída) (AGGARWAL, 2018).

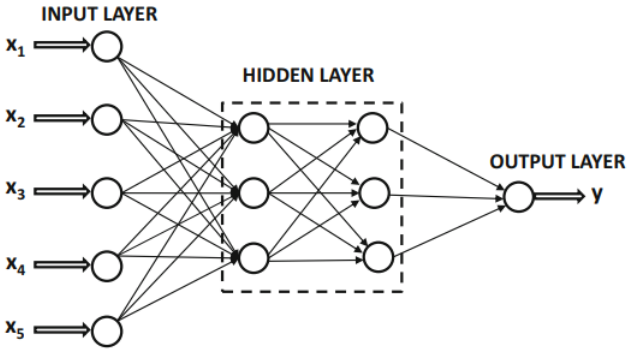


Figura 5 - Desenho da arquitetura MLP.
Fonte: (AGGARWAL, 2018)

Cada neurônio em uma camada utiliza uma função de ativação para produzir sua saída. Funções de ativação incluem a função sigmóide, a tangente hiperbólica (tanh) e a função de ativação ReLU (*Rectified Linear Unit*). Os dados de treinamento são alimentados pela rede a partir da camada de entrada, passando pelas camadas ocultas até alcançar a camada de saída. Cada camada realiza cálculos usando os pesos associados aos neurônios.

A saída da rede é comparada com as classes reais do conjunto de treinamento e uma função de perda é calculada para medir a discrepância entre as previsões e os rótulos reais. O algoritmo de retropropagação é usado para ajustar os

pesos da rede de forma a minimizar a função de perda. Os gradientes são calculados em relação aos pesos, e os pesos são atualizados usando um otimizador, como o Gradiente Descendente. O processo de propagação e retropropagação é repetido por várias ciclos, chamados de épocas, ajustando gradualmente os pesos para melhorar a precisão da rede (AGGARWAL, 2018).

Modelos baseados em MLP são estruturas simples de rede neural de aprendizado que consistem em uma camada de entrada, camadas ocultas e uma camada de saída, sendo cada elemento da rede associado com um certo valor de peso aleatório (ALSMADI et al., 2009). Uma das limitações para o uso dessa técnica é ser difícil de implementá-la e interpretá-la, porém têm a vantagem de uso em problemas representados por sistemas não-lineares (GARDNER; DORLING, 1998).

O MLP é o tipo mais conhecido e usado de rede neural, onde os sinais são transmitidos em uma só direção dentro da rede, da entrada para a saída, portanto não havendo retorno do sinal para os neurônios da mesma rede, por isso esta arquitetura é chamada *feedforward* (POPESCU, 2009).

3.2.8 Long Short-Term Memory

As redes neurais recorrentes (RNN) são capazes de utilizar suas conexões de retroalimentação para armazenar representações de entradas recentes na forma de ativação de neurônios, o que pode ser chamado de memória de curta duração. Essa abordagem funciona para muitos cenários como processamento de discurso ou composição musical, porém seu desempenho é ruim se houver uma lacuna muito grande entre os eventos na sequência (como palavras em uma frase) (ARUMUGAM; SHANMUGAMANI, 2018).

Isso ocorre pois os modelos de redes neurais possuem uma etapa de retropropagação (“*backpropagation*”) em que ocorre o ajuste dos pesos por gradientes pela multiplicação contínua de derivadas. No caso de haver derivadas muito pequenas, pode ocorrer o efeito de desaparecimento de gradiente. Para contornar este problema, criou-se a técnica de Long Short-Term Memory (LSTM), que captura melhor a conexão de contexto entre palavras com maior valor de classificação, “esquecendo” palavras sem valor para o modelo (HOCHREITER; SCHMIDHUBER, 1997).

A LSTM apresenta uma abordagem inovadora, pois consegue lembrar de

informações inicialmente inseridas na rede usando um "portão de esquecimento". A vantagem da LSTM é que ela pode reter informações por longos períodos. Na LSTM, existem vários portões, como o "portão de esquecimento", o "portão de saída" e o "portão de entrada" (ARUMUGAM; SHANMUGAMANI, 2018).

Cada unidade de LSTM possui uma porta de entrada, uma porta de saída e uma porta de esquecimento, que controlam o fluxo de informações na célula. Após processar toda a sequência, a última saída da LSTM ou a representação final da sequência é usada para fazer a previsão. Geralmente, essa saída é alimentada em uma camada densa (totalmente conectada) com ativação *softmax* para obter as probabilidades de cada classe (GERS, 2000).

A figura 6 ilustra a arquitetura da LSTM, onde o estado da célula anterior (Ct-1) representa a memória de longo prazo da rede, capturando informações relevantes de etapas de tempo anteriores. A entrada atual (*Xt*) contém as informações novas a serem processadas no momento atual.

Os parâmetros da rede (pesos e vieses) determinam como a rede combina o estado da célula anterior e a entrada atual para produzir a saída e atualizar o estado da célula para a próxima etapa de tempo. A figura 6 mostra a representação da unidade de LSTM; Onde $\mathbf{X_t}$: tempo de entrada; $\mathbf{h_t}$: saída; $\mathbf{C_t}$: estado da célula; $\mathbf{f_t}$: portão de esquecimento; $\mathbf{i_t}$: portão de entrada; $\mathbf{O_t}$: portão de saída; $\mathbf{\hat{C}_t}$: estado interno da célula; **tanh**: função tangente hiperbólica (HOCHREITER; SCHMIDHUBER, 1997).

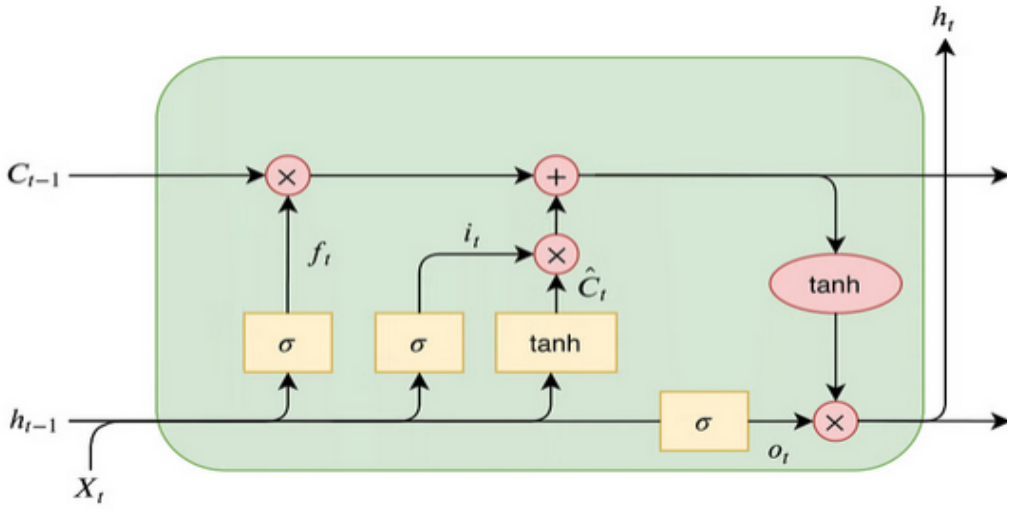


Figura 6 - Representação da unidade de LSTM.
Fonte: (HOCHREITER; SCHMIDHUBER, 1997)

O impacto do uso de LSTM tem sido notável para campos como processamento de linguagem, classificação de texto, tradução e transcrição de discurso para texto. (SHERSTINSKY, 2020).

3.2.9 Bidirectional Encoder Representations for Transformers

BERT (*Bidirectional Encoder Representations from Transformers*)[1] é um modelo de linguagem baseado em *transformers* pré-treinados, que se destacou em diversas tarefas de processamento de linguagem natural, incluindo a classificação de texto. O Transformer é um modelo de aprendizado profundo introduzido em 2017, e que utiliza o mecanismo de autoatenção para capturar relacionamentos entre palavras (VASWANI et al., 2017).

A grande inovação do BERT está em sua capacidade de entender o contexto bidirecional das palavras em uma sentença, o que o torna especialmente eficaz para tarefas que envolvem compreensão de contexto e ambiguidade (TENSORFLOW, 2023). A figura 7 apresenta a representação gráfica da arquitetura BERT (VASWANI et al., 2017).

[1] https://research.google/pubs/bert-pre-training-of-deep-bidirectional-transformers-for-language-understanding/

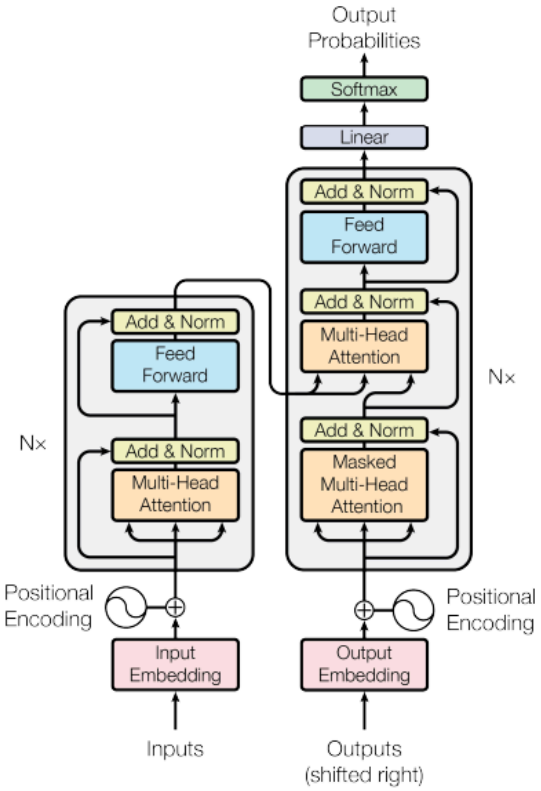


Figura 7 - Arquitetura BERT.
Fonte: (VASWANI et al., 2017).

O modelo BERT foi desenvolvido pela empresa Google, e é um grande avanço na área de PLN em uma ampla variedade de tarefas. A inovação desta técnica está na aplicação bidirecional do modelo *Transformer*. Pesquisas realizadas com esse modelo bidirecional mostram que é possível entendimento mais profundo do contexto quando comparado com modelos unidirecionais.

Esse modelo é capaz de realizar a análise do contexto das palavras, considerando o termo à esquerda e à direita em todas as camadas do texto. Essa tecnologia serviu de base para o desenvolvimento de tecnologias inovadoras como os Large Language Models (LLM) (DEVLIN et al., 2018).

3.3 Processamento de Linguagem Natural

O processamento de linguagem natural (PLN) começou em 1950 como uma intersecção entre os campos de inteligência artificial e linguística. Do período de 1960 até 2010 os métodos supervisionados de aprendizado para análise de textos

predominaram. A partir de 2010 houve ascensão de aprendizado profundo, com redes neurais convolucionais e recorrentes que revolucionaram o PLN, permitindo a extração de características complexas de grandes volumes de dados (QIAN LI et al., 2020).

Uma particularidade dos textos médicos é que o algoritmo precisa lidar com textos muito telegráficos, com frases curtas e concisas, com foco em informações relevantes, que são escritos de forma bastante direta, sem muitas conjunções, preposições e artigos. Além disso, possuem termos técnicos e jargões específicos da área médica, vocabulário extenso em constante mudança, exigindo atualização constante dos modelos de PLN. (PRAKASH, 2011).

Os primeiros experimentos com o uso de aprendizado supervisionado aplicado em um grande conjunto de dados médicos anotados (*corpus*) foram feitos com a utilização de SVM. Esse modelo precisa lidar com ambiguidade na interpretação, necessitando de conhecimento específico para desambiguação (SIMON et al. 2016).

3.3.1 Definição de *Corpus*

O termo *corpus* é definido como uma coleção de textos possível de ser lida por máquina, e que é escrita em alguma linguagem natural, incluindo linguagem falada. O conjunto de *corpus* específico pela área de conhecimento, chamado de *corpora*, têm um papel essencial na pesquisa sobre processamento de linguagem natural (MCENERY et al., 2006).

Existem centenas de *corpora* diferentes, porém a maioria é criada para um projeto específico e não são publicadas e acessíveis. O processo de criação do *corpus* leva tempo e exige investimentos financeiros, por isso, seria desejável que os pesquisadores tivessem acesso a *corpus* prontos e consolidados na sua área de atuação para desenvolverem seus trabalhos (XIAO, 2008).

Neste estudo, o *corpus* é todo o conjunto de registros de evolução escritas pelos profissionais de saúde, e que foram armazenadas no sistema eletrônico para cada paciente. Os textos contém a descrição dos sintomas sentidos pelos pacientes, histórico de eventos relacionados a atendimentos anteriores, medicações utilizadas e tratamentos.

3.3.2 Pré-processamento de texto

Para serem utilizados em treinamento de modelos de IA os textos precisam passar por etapas de preparo dos dados para adequá-los ao modelo de aprendizado de máquina. Por exemplo, retirada de palavras como verbos auxiliares, conjunções e artigos, pois não tem valor para análise textual; ou transformação das palavras em uma forma menor, como o radical (RANDLES, 2017). A figura 8 apresenta a sequência de etapas seguida para pré-processamento.

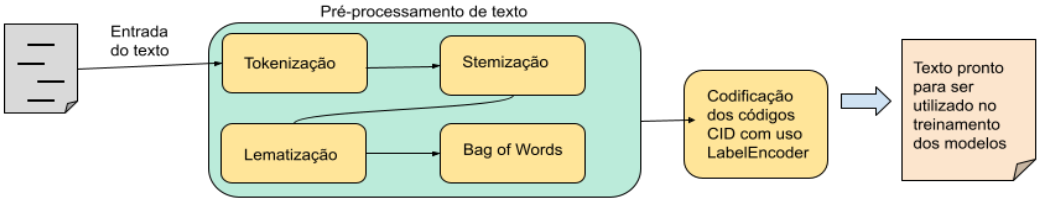


Figura 8 - Etapas de pré-processamento do texto de PME.
Fonte: Próprio autor

3.3.2.1 Stopwords

A remoção de *stopwords* é outra etapa comum de pré-processamento em aplicações de PLN. Nesse passo, removemos palavras que não influenciam no entendimento do texto, como artigos e pronomes indefinidos. Alguns exemplos desses termos são 'um', 'uma', 'ele' e 'ela'. Sozinhas, essas palavras podem não impactar tarefas de PLN, como categorização de texto ou busca, já que são usadas com frequência no texto. Porém, em contexto, podem provocar mudanças semânticas na interpretação. Exemplo disso são os artigos definidos e indefinidos. (ARUMUGAM; SHANMUGAMANI, 2018).

3.3.2.2 Tokenização

Como dito anteriormente, a etapa de pré-processamento consiste em uma série de alterações e adaptações das amostras, sendo o mais comum a transformação de cada elemento do texto para a forma de “*token*”, pelo processo

conhecido como “*tokenização*”. Esse processo busca fazer a segmentação do texto em elementos menores e que carregam o significado da palavra original (WEBSTER; KIT; CHUNYU, 1992).

Os tokens de palavras são as unidades básicas de texto envolvidas em qualquer tarefa de PLN. O primeiro passo ao processar texto é dividi-lo em tokens, como por exemplo, a palavra “gato” seria transformada em “gat”. A biblioteca NLTK fornece diferentes tipos de tokenizadores para realizar essa tarefa através de diferentes técnicas (ARUMUGAM; SHANMUGAMANI, 2018).

3.3.2.3 Stemização e Lematização

A derivação ou Stemização (do inglês, stemming) é uma etapa de pré-processamento de texto que transforma palavras relacionadas ou similares (como 'caminhando') em sua forma base (como 'caminhar'), já que possuem significados análogos. Uma das transformações é reduzir palavras no plural para o singular: 'maçãs' se torna 'maçã', por exemplo. Embora essa seja uma transformação simples, existem outras mais complexas (ARUMUGAM; SHANMUGAMANI, 2018).

Há também o processo de “*lematização*”, que baseia-se na transformação de palavras para que ela possa ser reconhecida por apenas um simples elemento, conhecido como lema. Por exemplo, “corredor” e “corrida” se transformam em “correr” (KHYANI; DIVYA; SIDDHARTHA, 2021).

3.3.2.4 Bag-of-Words

A técnica chamada *Bag-of-Words* (BOW), ou Bolsa de Palavras, que representa as palavras através de vetores de dicionários, no qual o valor individual do vetor denota a frequência da palavra no texto, criando uma matriz de ocorrência para a sentença ou documento sem considerar a gramática e ordem das palavras, que é então usada para o treinamento do modelo classificador (HACOHEN, 2020).

3.3.2.5 Label Encoder

Finalmente, alguns modelos classificadores exigem transformações

específicas da amostra textual. Por exemplo, a transformação das classificações em códigos numéricos a partir do uso de *LabelEncoder* para representar as classes, é uma etapa de pré-processamento essencial de modelos como SVM e Regressão Logística. *LabelEncoder* é uma técnica usada em aprendizado de máquina para converter variáveis categóricas (textuais) em valores numéricos. (IKONOMAKIS; KOTSIANTIS; TAMPAKAS, 2005).

Segundo QIAN LI et al., 2020, muitos modelos têm sido propostos para classificação de texto, identificação de tópicos, classificação de notícias e resposta automáticas de perguntas. O algoritmo Naive Bayes foi o primeiro modelo de aprendizado supervisionado usado para a classificação de texto, sendo usado posteriormente classificadores como KNN, SVM e Florestas Aleatórias (KIM, 2014).

3.4 Classificação de textos dos Prontuários

A classificação de PE foi realizada usando diferentes algoritmos de aprendizado de máquina. Será apresentado a seguir como os algoritmos foram usados para este fim, baseando-se na literatura consultada durante a revisão da literatura.

3.4.1 Classificação usando Naive Bayes

O modelo Naive Bayes (NB) é um algoritmo de aprendizado de máquina supervisionado usado para classificação de texto. O modelo é treinado em um conjunto de dados rotulados que consiste em exemplos de textos já categorizados. Para cada classe, o modelo calcula a probabilidade de ocorrência de cada palavra no conjunto de treinamento. Para cada palavra, calcula a probabilidade condicional de ela ocorrer em cada classe. Isso é feito dividindo o número de vezes que a palavra aparece em exemplos da classe pela contagem total de palavras naquela classe. O modelo calcula a probabilidade de ser uma classe, e a probabilidade de não sê-la para cada palavra no texto, multiplicando-as pelas probabilidades *a priori* de pertencer e de não pertencer à classe, respectivamente. A classe atribuída seria aquela com a maior probabilidade calculada (BIRD; KLEIN; LOPER, 2009).

Além da filtragem de spam, existem muitas outras aplicações para a classificação automática de texto com NB. Exemplos importantes incluem filtrar

entradas indesejadas em fóruns de discussão online e rastrear sites com conteúdo criminoso, como atividades terroristas ou racismo. Ela também pode ser usada para personalizar motores de busca de acordo com as preferências do usuário a fim de classificar melhor os resultados da pesquisa. Apesar da sua simplicidade, o NB entrega excelentes resultados para classificação de texto, e quando bem treinado, pode alcançar até 99% de acerto (ERTEL, 2017).

3.4.2 Classificação usando Regressão Logística

A Regressão Logística é um modelo estatístico utilizado para resolver problemas de classificação binária, e pode ser adaptada para lidar com problemas de classificação de texto. O algoritmo da Regressão Logística classifica textos estruturados, como artigos de notícias, por exemplo, ao aprender os padrões que diferenciam as classes de texto usando uma função logística (BIRD; KLEIN; LOPER, 2009). A função logística é uma função matemática que estima a probabilidade de um texto pertencer a uma classe específica, sendo que valores mais próximos de 1 indicam probabilidade maior do texto pertencer à determinada classe.

Assim como em outros modelos de aprendizado supervisionado, a Regressão Logística é treinada em um conjunto de dados rotulados que consiste em exemplos de texto associados a suas respectivas classes (ERTEL, 2017). O texto é convertido em uma representação numérica, geralmente usando técnicas como a contagem de palavras (Bag of Words). Isso transforma o texto em um vetor de características no formato exigido pelo modelo. O modelo aprende os parâmetros dos textos da amostra de treino durante esse processo. Esses parâmetros incluem os pesos associados a cada característica (palavra) e um termo de viés (bias). Um limiar de decisão é escolhido (tipicamente 0.5). Se a probabilidade calculada for maior que o limiar, o exemplo é classificado como pertencente à classe. Os pesos aprendidos durante o treinamento são aplicados, e a função logística é usada para calcular a probabilidade de pertencer à classe (AGGARWAL, 2018). ALOTAIBI, 2019 usou o modelo de Regressão Logística para classificar uma série de textos em cinco categorias diferentes (Alegria, Vergonha, Medo, Tristeza e Culpa), expressando o sentimento expresso no texto.

3.4.3 Classificação usando KNN

O KNN é um algoritmo de aprendizado de máquina que pode ser utilizado para classificação de texto, embora seja mais comumente associado a problemas de classificação em geral. Ele é especialmente útil para classificar texto que não é estruturado, como e-mails, comentários de mídia social ou postagens em fóruns. O modelo KNN é treinado em um conjunto de dados rotulados que consiste em exemplos de texto associados às suas respectivas classes. O texto a ser classificado é vetorizado da mesma maneira que os textos durante o treinamento, convertendo-o em uma representação numérica com uso do *Bag-of-Words* (BIRD; KLEIN; LOPER, 2009).

Como explicado na seção **3.3.3** o número k de exemplos de treinamento são identificados com base nas distâncias calculadas. As classes dos “k” elementos vizinhos mais próximos são avaliadas, e o texto é classificado com base na classe mais frequente. Um valor de k maior leva em consideração mais textos vizinhos, o que pode levar a uma classificação mais precisa. No entanto, um valor de k muito alto pode levar a um *overfitting* (super adequação), em que o modelo se adapta demais aos dados de treinamento e não é capaz de generalizar para novos dados (ARUMUGAM; SHANMUGAMANI, 2018).

A aplicação inicial de KNN para categorização textual por BIJALWAN et al., 2014 determinou a categoria de documentos com grande precisão, porém os autores concluíram que o modelo leva mais tempo em comparação com NB. XING et al., 2019 utilizaram KNN para classificação de textos médicos.

3.4.4 Classificação usando Árvores de Decisão

As Árvores de Decisão são modelos de aprendizado de máquina que podem ser utilizados para a classificação de texto, convertendo sequências de palavras em decisões binárias. As Árvores de Decisão são frequentemente usadas para classificar texto porque são capazes de aprender padrões complexos nos dados. A ideia principal por trás do algoritmo é dividir recursivamente os dados em subconjuntos homogêneos com base em características específicas, formando uma estrutura de árvore (BIRD; KLEIN; LOPER, 2009).

O modelo é treinado em um conjunto de dados rotulados que consiste em exemplos de texto associados às suas respectivas classes. O texto é convertido em

uma representação numérica usando técnicas como *Bag-of-Words*. A Árvores de Decisão procura pelo melhor atributo (característica) para dividir os dados. Para texto, isso pode ser uma palavra específica ou uma contagem de palavras (ARUMUGAM; SHANMUGAMANI, 2018).

Um nó na árvore é criado com base no melhor atributo, e representa uma decisão, como "Se a palavra X está presente no texto: Sim ou Não?". Os dados são divididos com base no valor do atributo escolhido. Esse processo é repetido recursivamente para cada subconjunto, criando novos nós na árvore. O processo de divisão continua até que um critério de parada seja atendido, como profundidade máxima da árvore, número mínimo de exemplos por folha, ou impureza mínima (AGGARWAL, 2018).

O texto de teste é passado pela árvore, seguindo os ramos correspondentes às decisões baseadas nas características do texto. Quando o texto atinge uma folha da árvore, a classe associada a essa folha é atribuída ao texto como a predição.

3.4.5 Classificação usando Floresta Aleatória

O modelo de Floresta Aleatória é uma extensão do modelo de Árvore de Decisão que utiliza múltiplas árvores para realizar a classificação de texto. Cada corpus do conjunto de dados é associado a uma classe, como "notícia", "artigo científico" ou "ficção", por exemplo. O modelo cria um conjunto de árvores de decisão, cada uma com um conjunto aleatório de dados de treinamento. Para classificar um novo texto, o modelo vota nas classes mais prováveis de cada árvore de decisão (BIRD; KLEIN; LOPER, 2009).

Para cada árvore na floresta, é criado um conjunto de treinamento por amostragem aleatória com substituição (*bootstrap*) a partir do conjunto de dados original. Isso significa que alguns exemplos podem aparecer várias vezes, enquanto outros podem não aparecer. Cada árvore da floresta é treinada independentemente do seu conjunto de dados de inicialização. O processo de treinamento é semelhante ao treinamento de uma única árvore de decisão. A Floresta Aleatória lida bem com uma grande quantidade de características (palavras) e é capaz de avaliar a importância relativa de diferentes características no processo de classificação, além de ser robusta a *outliers* e ruídos nos dados (ARUMUGAM; SHANMUGAMANI, 2018). CHE et al., 2023 usaram o modelo Floresta Aleatória para classificar registros

em PME de pacientes de oncologia de acordo com classes diferentes.

3.4.6 Classificação usando SVM

As Máquinas de Vetores de Suporte (do inglês, Support Vector Machines - SVM) são modelos de aprendizado de máquina utilizados para classificação de texto. As SVM são eficazes em problemas de classificação binária e multiclasse, sendo particularmente úteis em amostras de alta dimensionalidade, como o espaço de características gerado a partir da representação de texto após transformação por Bag of Words (ARUMUGAM; SHANMUGAMANI, 2018).

O conjunto de dados é representado como pontos em um espaço de características, no qual cada dimensão corresponde a uma característica (por exemplo, a presença ou frequência de uma palavra). O texto a ser classificado é vetorizado da mesma forma que modelos anteriormente citados, convertendo-o em um vetor de características. Se um kernel for utilizado, o texto de teste é projetado no espaço de características correspondente. O texto é classificado com base na posição em relação ao hiperplano de decisão.

3.4.7 Classificação usando MLP

Redes Neurais Multilayer Perceptron (MLP) são modelos de aprendizado profundo que podem ser utilizados para a classificação de texto. O modelo MLP é treinado em um conjunto de dados rotulados que consiste em exemplos de texto associados às suas respectivas classes. O texto é convertido em uma representação numérica, geralmente usando Bag of Words. Isso transforma o texto em um vetor de características. O vetor de características do texto de teste é então usado pela rede neural, passando pela camada de entrada, camadas ocultas e produzindo uma previsão na camada de saída (ARUMUGAM; SHANMUGAMANI, 2018).

A classificação de textos com uso de MLP consegue obter valores de 99% de acurácia quando combinado com a técnica *Bag-of-Words* (LUBIS et al., 2023).

3.4.8 Classificação usando LSTM

Modelos de aprendizado de máquina baseados em Redes Neurais

Recorrentes (RNN), como exemplo da LSTM interpretam um texto palavra por palavra e guardam o significado semântico de todo texto anterior à palavra em uma camada oculta de tamanho fixo (HOCHREITER; SCHMIDHUBER, 1997).

O texto é convertido em uma representação numérica, geralmente usando técnicas como *word embeddings*. Cada palavra ou token no texto é mapeado para um vetor. Ao contrário de modelos que tratam cada entrada independentemente, um LSTM processa a sequência de palavras no texto uma de cada vez, mantendo uma memória interna que é atualizada a cada passo. A previsão da classe é obtida a partir da saída final da rede, geralmente após uma camada densa com ativação softmax (ARUMUGAM; SHANMUGAMANI, 2018).

3.4.9 Classificação usando BERT

O algoritmo BERT é baseado na arquitetura de *Transformer*, que utiliza camadas de atenção e autoatenção para capturar relacionamentos entre palavras em sequências de entrada. Após o pré-treinamento, camadas adicionais são adicionadas ao BERT para adaptá-lo a tarefas específicas, como a classificação de texto. Uma camada de classificação é adicionada no topo do BERT (TENSORFLOW, 2023).

O texto de entrada é tokenizado e vetorizado de acordo com o vocabulário do BERT. Cada palavra é convertida em um vetor de incorporação (*embedding*) contextualizado. A saída da representação é alimentada na camada de classificação, onde a tarefa específica, como classificação binária ou multiclasse, é realizada (DEVLIN et al., 2018).

‘ O BERT tem se destacado em tarefas de classificação de texto devido à sua capacidade de entender o contexto bidirecional das palavras em uma sequência, capturando relações entre elas. O ajuste fino permite que o modelo seja adaptado a tarefas específicas com um conjunto relativamente pequeno de dados rotulados (TENSORFLOW, 2023).

3.5 Métricas de Avaliação dos Modelos de Classificação

Neste trabalho serão utilizadas algumas métricas de avaliação comumente vistas em artigos científicos (NANCY, 1992); (NAEEM; TAGHI; JASON, 2009);

(VAKILI et al., 2020) que tratam sobre a avaliação de métodos classificadores em relação à sua performance.

O objetivo destas métricas é permitir que sejam feitas comparações de desempenho entre os diferentes modelos de aprendizado de máquina e assim discriminar e selecionar a solução otimizada gerada durante o treinamento dos modelos.

3.5.1 Acurácia

A acurácia (*Accuracy*) é uma métrica para avaliar modelos de classificação, e representa a fração das predições que o modelo obteve de forma correta (HARRISON, 2020). Devemos notar que esta medida não é apropriada quando temos um conjunto de dados desbalanceados (HOSSIN; SULAIMAN, 2015). A equação 3 apresenta o cálculo da métrica (HARRISON, 2020):

$$Acurácia = (TP + TN) / (TP + TN + FP + FN) \quad (3)$$

Onde:
- TP = Verdadeiro Positivo;
- TN = Verdadeiro Negativo;
- FP = Falso Positivo;
- FN = Falso Negativo

3.5.2 Precisão

A medida de precisão responde qual proporção dos valores positivos foram realmente positivos, considerando o número total determinado como positivo (HARRISON, 2020). Também é conhecido como Valor Preditivo Positivo (HOSSIN; SULAIMAN, 2015). A equação 4 apresenta o cálculo da métrica (HARRISON, 2020):

$$Precisão = TP / (TP + FP) \quad (4)$$

Onde:
- TP = Verdadeiro Positivo;
- FP = Falso Positivo;

3.5.3 Abrangência (*Recall*)

A medida de Abrangência ou “Recall” (Revocação ou Sensibilidade) mostra a proporção, ou porcentagem, dos verdadeiros positivos que foram identificados corretamente (HOSSIN; SULAIMAN, 2015) - Quantos resultados relevantes são devolvidos? A equação 5 apresenta o cálculo da métrica (HARRISON, 2020).

$$Abrangência\ (Recall)\ =\ TP\ /\ (TP\ +\ FN) \tag{5}$$

Onde:
- TP = Verdadeiro Positivo;
- FN = Falso Negativo;

3.5.4 Medida F-Score

A medida Escore F1 é uma métrica comumente utilizada para avaliar o desempenho de modelos de classificação, especialmente em situações em que as classes são desequilibradas. Ela combina as métricas de precisão (precision) e recall em uma única pontuação, oferecendo uma visão equilibrada do desempenho do modelo (SOKOLOVA; JAPKOWICZ; SZPAKOWICZ, 2006). A equação 4 apresenta o cálculo da métrica (HARRISON, 2020).

$$EscoreF1\ =\ 2\ *\ (TP/(TP\ +\ FP)\ *\ TP/(\ TP\ +\ FN))\ /\ (TP/(TP\ +\ FP)\ +\ TP/(\ TP\ +\ FN)) \tag{6}$$

Onde:
- TP = Verdadeiro Positivo;
- TN = Verdadeiro Negativo;
- FP = Falso Positivo;
- FN = Falso Negativo;

O F-Score é a média harmônica da precisão e do recall, e varia de 0 a 1. Ele atinge o valor máximo de 1 quando a precisão e o recall são ambos 1 (indicando uma classificação perfeita) e atinge o valor mínimo de 0 quando a precisão ou o

recall é 0 (indicando uma classificação muito ruim).

3.6 Inteligência Artificial Generativa

4. Revisão da Literatura

Esse capítulo descreve uma Revisão Rápida da Literatura (do inglês, *Rapid Review*) que foi realizada visando responder a questão de pesquisa. Este tipo de revisão constitui-se em uma forma de sintetizar o conjunto de evidências científicas de forma rápida, ajudando no embasamento para tomada de decisão (SCHÜNKE et al., 2022).

Esse tipo de revisão deve seguir os seguintes pontos: elaboração da pergunta de investigação, busca estruturada da literatura, seleção e análise crítica dos estudos, revisão por pares, disponibilização do material em site eletrônico (SETSUKO, 2016). Deve focar em resultados relevantes na literatura para produzir conclusões válidas (KONNYU et al., 2012) (WATT et al., 2008).

4.1 Protocolo da Revisão

A seguir será descrito em detalhes o protocolo de pesquisa elaborado para a realização da Revisão rápida da literatura.

4.2 Questão de Pesquisa

Considerando os pontos citados anteriormente, elaborou-se a pergunta de pesquisa utilizada para busca em cinco bases de artigos científicos. Para seleção e análise crítica dos artigos, realizadas por duas pessoas diferentes, utilizou-se a plataforma Rayyan, que é uma ferramenta web projetada para ajudar pesquisadores que trabalham em revisões sistemáticas, revisões de escopo e outros projetos de síntese de conhecimento, acelerando drasticamente o processo de triagem e seleção de estudos[2].

A pergunta de pesquisa foi elaborada utilizando a estratégia PICO (*Population, Intervention, Comparison, Outcome*) ou em português; População, Intervenção, Comparação e Desfecho.

P: Pacientes da cardiologia.
I: Uso de modelos de aprendizado de máquina na análise dos PME.
C: Entre modelos de aprendizado, com base em métricas de avaliação.

[2] https://www.rayyan.ai/

O: Determinação do diagnóstico relacionado ao PME.

A questão formulada com base na estratégia PICO foi: - *Qual o modelo de aprendizado de máquina obteve melhor desempenho para classificar o diagnóstico a partir de prontuários eletrônicos de pacientes do setor de cardiologia internados em um hospital?*

4.3 Consulta das bases de dados

A consulta das bases foi realizada no mês de Julho de 2022 e seguiu as orientações do checklist PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) (Figura 9) para seleção dos artigos. Cinco fontes foram utilizadas: Biblioteca Virtual em Saúde (BVS), PubMed, Science Direct, BioMed Central (BMC) e Institute of Electrical and Electronics Engineers (IEEE); os artigos foram publicados no período de 2016 a 2021, e escritos na língua inglesa.

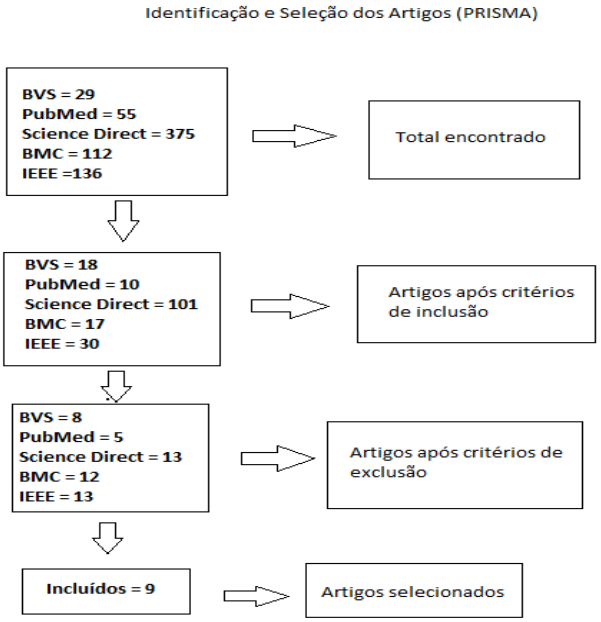


Figura 9 - Etapas de seleção dos artigos de revisão.

Fonte: O próprio autor.

Após exclusão dos artigos duplicados realizou-se a seleção detalhada através da leitura do título e do resumo de cada artigo. Os artigos restantes passaram por uma avaliação criteriosa, através da leitura na íntegra em pares, com o objetivo de

avaliar qual algoritmo de aprendizado de máquina foi utilizado e qual resultado cada estudo obteve.

Quadro 1 - Critérios de Inclusão dos Artigos

Critério 1	Artigos completos que empregam algum modelo de aprendizado de máquina análise de informações de prontuários eletrônicos
Critério 2	Artigos publicados entre 2016 e 2021
Critério 3	Artigos escritos na língua inglesa

Quadro 2 - Critérios de Exclusão de Artigos

Critério 1	Materiais publicados em livros, editoriais, cartas ou dissertações
Critério 2	Artigos que não estejam relacionados à aprendizado de máquina e área da saúde
Critério 3	O artigo não se encontra disponível em sua totalidade
Critério 4	O artigo não possui acesso aberto
Critério 5	O artigo se repete nas pesquisas

4.4 Análise dos estudos obtidos

Os estudos podem ser divididos em três grandes objetivos primários a partir da análise de PME: selecionar tratamentos adequados aos pacientes com arritmia cardíaca (CHENG et al., 2021); determinar desfechos e possíveis complicações durante o tratamento (CHEN et al., 2019), (ZHAO et al., 2019), (WENG et al., 2017) e (CHU et al., 2020); prever ocorrência de eventos adversos após o paciente ter passado pela terapia médica (CHEN et al., 2020), (ZHAO et al., 2019), (AHN et al., 2021) e (ASHFAQ et al., 2019).

Os trabalhos que utilizaram Redes Neurais foram mais frequentes, citadas em pelo menos seis dos trabalhos, com diferentes configurações de arquitetura (CHEN et al., 2020), (ZHAO et al., 2019), (KRITTANAWONG et al., 2021), (WENG et al., 2017), (ASHFAQ et al., 2019) e (CHENG et al., 2021).

O algoritmo de Florestas Aleatórias (Random Forest) é citada em quatro dos estudos (ZHAO et al., 2019), (KRITTANAWONG et al., 2021), (AHN et al., 2021) e (CHENG et al., 2021). O modelo de Árvore de Decisão, que apesar de ser mais facilmente interpretável em relação aos resultados, possui menor poder de predição, e foi usado em um artigo apenas (CHENG et al., 2021).

O algoritmo de KNN foi utilizado com bons resultados de precisão na predição dos desfechos de tratamento em pacientes com insuficiência cardíaca (CHEN et al., 2019). O modelo de Regressão Logística foi aplicado em um contexto longitudinal, trazendo grande melhoria na capacidade preditiva (ZHAO et al., 2019).

Duas técnicas de aprendizado de máquina foram propostas por (PRABHAKAR, 2021) para analisar textos anotados com informações médicas para treinamento. Os autores usaram um modelo baseado em LSTM híbrido, contendo 4 canais para obter uma acurácia na classificação em torno de 76% em um conjunto de dados, e 96,72% em outro. Outro modelo utilizado neste trabalho foi baseado em RNN bidirecional obtendo acurácia semelhante quando testado para classificação de textos médicos.

No trabalho de (SHUANG et al. 2019) foi proposto uma variação do modelo de SVM com o objetivo de construir um classificador confiável. Os autores compararam este modelo com outros classificadores como Naive Bayes, KNN, MLP, e Árvores de Decisão contra 12 conjuntos de dados. Foram comparados o desempenho dos modelos utilizando métricas de avaliação como Acurácia, Precisão, Recall, F-Score e Área sobre a Curva ROC (do inglês, area under the receiver operating characteristic - AUROC).

Finalmente, no trabalho realizado por Pakhomov, 2007 ficou demonstrado que a detecção de diagnóstico médico de dor no peito (angina) por análise de prontuários médicos com modelos de NLP obteve maior precisão em comparação com diagnóstico atribuído por especialistas humanos utilizando texto livre contido nos PME.

4.5 Estudos selecionados

Com base na revisão rápida da literatura foi possível responder às questões definidas e identificar pesquisas relacionadas ao proposto nessa pesquisa. Os estudos selecionados buscaram prever com mais precisão a evolução de pacientes em diferentes cenários médicos, levando a tomada de decisões clínicas mais assertivas, otimizando o tratamento e prevenindo complicações.

As pesquisas também utilizaram técnicas de Machine Learning para analisar automaticamente grandes volumes de dados contidos em prontuários eletrônicos. Isso permitiu identificar informações relevantes de forma mais rápida e eficiente,

como histórico de doenças, medicamentos utilizados, diagnósticos e outros dados importantes para o acompanhamento do paciente.

Quadro 3 - Sumarização dos estudos relacionados

Citação	Objetivo	Técnica Utilizada	Resultados Principais
(CHEN et al., 2019)	Estimar efeitos de tratamento em pacientes	K Vizinhos Mais Próximos (KNN)	O modelo alcançou precisão de 1,7
(CHEN et al., 2020)	Predizer o risco de readmissão por insuficiência cardíaca	Modelo de Rede Neural	O modelo alcançou acurácia de 66,7
(ZHAO et al., 2019)	Predição de eventos cardiovasculares com dados longitudinais	Regressão Logística, Florestas Aleatórias, Redes Neurais Convolucionais	Os modelos foram melhorados ao aplicar dados longitudinais
(KRITTANAWONG et al., 2021)	Predição de mortalidade hospitalar em pacientes com dissecção arterial coronariana	Floresta Aleatória, AdaBoost e Rede Neural Profunda	A Rede Neural Profunda obteve maior acurácia preditiva e poder discriminatório em pacientes com maior risco
(AHN et al., 2021)	Modelo preditivo de probabilidade de alta em pacientes cardiovasculares	Extreme Gradient Booster, Regressão Logística, Floresta Aleatória, SVM e Perceptron Multicamadas	Extreme Gradient Booster obteve valor AUROC de 0,865
(WENG et al., 2017)	Classificar um domínio médico através da análise dos prontuários eletrônicos	Modelo de Rede Neural Convolucional	O modelo obteve valor de AUROC de 0,975
(ASHFAQ et al., 2019)	Prever readmissão não planejada em 30 dias	Rede Neural do tipo LSTM	O modelo obteve valor AUROC de 0,77
(CHU et al., 2020)	Desenvolver um modelo para prever efeitos de tratamento	Modelo criado pelos autores	O modelo criado teve melhor desempenho quando comparado às técnicas tradicionais
(CHENG et al., 2021)	Identificar padrões de pacientes com melhor resposta ao tratamento de arritmias cardíacas	Árvores de decisão, Junção de Modelos (Ensemble), Máquinas de Gradiente	A técnica nova de Junção (Ensemble) obteve valor AUROC de 0,76

		Melhorado, Floresta Aleatória e CNN	

Os trabalhos relacionados descritos na tabela serviram como subsídio para o planejamento e execução das ações desta dissertação, principalmente em relação aos modelos de aprendizado de máquina escolhidos, etapas de processamento das amostras e métricas de avaliação de desempenho dos modelos. A pesquisa apresentada nesta dissertação é voltada para a classificação automatizada de prontuários na área de cardiologia se diferencia de outros trabalhos semelhantes em vários aspectos.

A cardiologia é uma área médica complexa que lida com várias condições cardíacas, cada uma com suas próprias características e tratamentos. Um sistema de classificação automatizada que é especificamente projetado para esta área pode ser capaz de identificar nuances e padrões específicos que podem não ser evidentes em um sistema de classificação mais genérico.

5. Materiais e Métodos

A pesquisa visa resolver um problema específico encontrado em contextos reais, neste caso, propor um modelo para a classificação automática de textos de prontuários médicos eletrônicos em um hospital de cardiologia. Além disso, o estudo é experimental porque envolve a criação de condições controladas e diferentes modelos de aprendizado de máquina que são desenvolvidos e testados para avaliar sua eficácia na classificação de textos médicos.

Neste capítulo serão apresentados os principais conceitos e ferramentas utilizadas para a execução da pesquisa.

5.1 Caracterização da Pesquisa

A pesquisa está focada na área de saúde, especificamente na gestão de documentos eletrônicos gerados no atendimento em saúde, como os Prontuários Médicos Eletrônicos (PME). O desafio é lidar com o volume significativo de informações relacionadas, como doenças e condições clínicas, contidas nos PME. O objetivo da pesquisa é desenvolver modelos de aprendizado de máquina supervisionado para a classificação automática de textos de prontuários eletrônicos de pacientes atendidos em um hospital de cardiologia, que são usados como ilustração da viabilidade desta abordagem.

A pesquisa é aplicada e experimental. As etapas seguidas foram: coleta de amostras de PME, pré-processamento das amostras de textos, treinamento dos modelos preditivos utilizando textos pré-processados, criação de amostras sintéticas com uso de Inteligência Artificial Generativa, avaliação do desempenho dos modelos com base nas métricas de acurácia, precisão, abrangência e F-Score.

5.2 Delineamento da Pesquisa

Para realização deste estudo seguiu-se uma série de passos, ou etapas, de forma a permitir obter sucesso ao final, tendo em vista que necessariamente deve seguir esta ordem.

O capítulo 4 apresenta o **Passo 1**, Mapeamento sistemático da literatura, servindo como o referencial teórico que serviu de embasamento neste trabalho,

Utilizou-se aqui a abordagem ***Rapid Review***. Foi encontrada uma série de trabalhos que seguiram proposta semelhante a presente dissertação, qual seja, utilizar algum modelo de Aprendizado de Máquina Supervisionado para processar informações de registros eletrônicos de pacientes a fim de extrair alguma informação clínica relevante, ou predizer algum desfecho clínico, de forma automatizada.

Essa revisão da literatura serviu como ponto de partida para elaboração da estratégia de pré-processamento das amostras e escolha de modelos classificatórios para serem utilizados neste trabalho.

Os capítulos 5.2 a 5.5 representam o **Passo 2**, no qual detalha-se o material utilizado e o preparo do mesmo para uso nos modelos propostos. A seção 5.5.4 aborda o **Passo 3**, com a geração dos scripts de código Python com os modelos, e treino dos mesmos com as amostras separadas para treino, a partir do conjunto de dados inicial.

O capítulo 6 aborda a avaliação de desempenho dos modelos **(Passo 4)**. Finalmente, no capítulo 7 **(Passo 5)**, discute-se os resultados obtidos com os experimentos.

Além disso, analisa-se os resultados obtidos por trabalhos relacionados com os obtidos nos experimentos desta pesquisa. A figura 10 ilustra os passos metodológicos adotados:

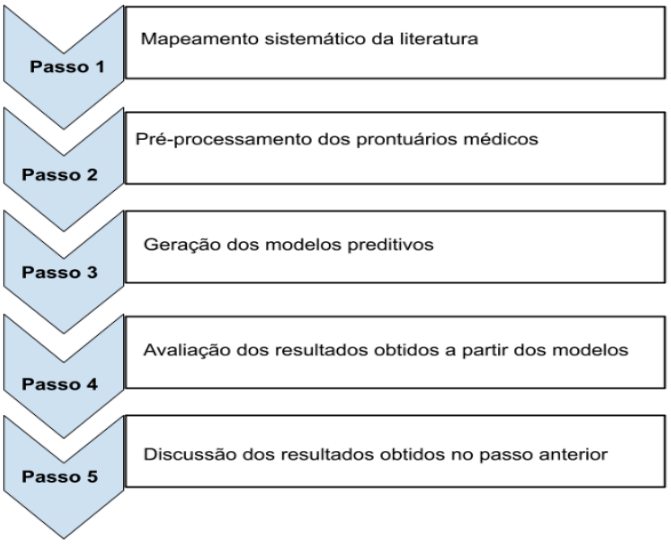


Figura 10 - Etapas metodológicas da pesquisa.

Fonte: O próprio autor.

5.3 Linguagem Python

Python[3] é uma linguagem de programação de alto nível, interpretada, orientada a objetos e de propósito geral. Criada por Guido van Rossum, Python é conhecida por sua sintaxe clara e legível, o que a torna uma excelente escolha para desenvolvedores de todos os níveis de experiência.

Python oferece bibliotecas robustas e eficientes para desenvolvimento de IA, como TensorFlow e PyTorch, que facilitam a implementação de redes neurais, algoritmos de aprendizado de máquina e outros modelos de IA (BIRD; KLEIN; LOPER, 2009). Por esta razão, a linguagem Python foi utilizada, e não outras linguagens,

5.4 Scikit-Learn

A scikit-learn[4] é uma biblioteca de aprendizado de máquina de código aberto para a linguagem de programação Python. Essa biblioteca reúne uma série de funções que permitem desenvolver modelos de classificação, regressão e pré-processamento, por exemplo (PEDREGOSA *et al.*, 2011).

5.5 Natural Language Toolkit

A Ferramenta de Linguagem Natural (NLTK) é uma biblioteca Python para lidar com tarefas de processamento de linguagem natural (PLN), abrangendo desde a segmentação de palavras ou frases até a realização de tarefas avançadas, como análise sintática e classificação de texto. O NLTK fornece diversos módulos e interfaces para trabalhar com linguagem natural, úteis para tarefas como identificação de tópicos de documentos, etiquetagem de partes do discurso (POS), análise de sentimento e assim por diante. Para experimentação com várias tarefas de PLN, o NLTK também inclui módulos para uma ampla variedade de corpora de texto, desde coleções básicas de texto até textos marcados e estruturados, como o

[3] https://www.python.org/
[4] https://scikit-learn.org/stable/

WordNet (ARUMUGAM; SHANMUGAMANI, 2018).

NLTK[5] É uma plataforma para a construção de programas Python para trabalhar com dados de linguagem humana. Ela fornece interfaces fáceis de usar para mais de 50 *corpora* e recursos lexicais juntamente com um conjunto de bibliotecas de processamento de texto para classificação, tokenização, stemming, tagging, parsing, além de wrappers para bibliotecas de PLN.

5.6 Dados Sintéticos

O aprendizado de máquina depende de dados, mas enfrenta problemas como baixa qualidade, insuficiência de dados e dificuldades de acesso devido a questões de privacidade e regulamentações. A geração de dados sintéticos surge como uma alternativa promissora, permitindo o compartilhamento e uso de dados de formas que os dados reais não permitem. Um estudo de Lu et al. 2023, analisou exemplos de estudos que utilizam modelos de aprendizado de máquina para gerar dados sintéticos, abordando aplicações em visão computacional, fala, processamento de linguagem natural, saúde e negócios. Explora métodos de aprendizado de máquina, com foco em redes neurais e modelos generativos profundos, e discute preocupações com privacidade e equidade. O estudo identifica desafios e oportunidades na área, destacando caminhos para pesquisas futuras.

A geração de dados sintéticos, segundo Clarke (1976), consiste em um processo multifacetado que se divide em duas etapas cruciais. A primeira etapa concentra-se na criação de dados realistas e consistentes internamente, utilizando técnicas como modelos probabilísticos e algoritmos de geração adversarial. Na segunda etapa, a validação lógica garante a coerência das características geradas, utilizando regras predefinidas e modelos de inferência lógica.

5.7 ChatGPT

O principal componente que permitiu ao ChatGPT suas respostas coerentes e envolventes é atribuído ao uso de aprendizado por reforço. Nesse contexto, avaliadores humanos classificaram as respostas do ChatGPT, o que permitiu otimizar alguns de seus parâmetros para essencialmente se tornar um melhor

[5] https://www.nltk.org/

interlocutor. O ChatGPT foi desenvolvido com base no GPT-3, um modelo de linguagem generativo autoregressivo com 175 bilhões de parâmetros (BROWN et al., 2020).

ChatGPT[6] é um modelo de linguagem de última geração que utiliza técnicas de aprendizagem profunda para gerar texto semelhante ao humano, utilizando uma arquitetura de rede neural profunda baseada no modelo *Transformer*. O modelo é pré-treinado em grandes quantidades de dados de texto, permitindo-lhe aprender os padrões e estruturas subjacentes da linguagem.

Uma das principais vantagens do ChatGPT é a sua capacidade de gerar dados coerentes e contextualizados, tornando-o uma ferramenta popular para aplicações como chatbots, tradução de idiomas e geração de texto. Foi desenvolvido pela OpenAI[7], uma empresa dedicada ao desenvolvimento de tecnologias de aprendizado de máquina (HASSANI; SILVA, 2023).

No entanto, o Chat GPT utiliza textos de uso cotidiano, extraídos de redes sociais, Wikipedia e periódicos. Porém, em alguns casos, utiliza textos sem o consentimento dos autores.

5.8 Dados Sintéticos para Prontuários Médicos

A geração de dados sintéticos por meio de modelos de IA generativa pode ser uma estratégia eficaz para melhorar o treinamento de algoritmos de classificação de texto, especialmente quando a pesquisa envolve dados sensíveis. A seguir estão listadas algumas formas de uso dessa abordagem, segundo Eigenschink et al. (2023):

1. Aumento de Dados

Muitas vezes, conjuntos de dados disponíveis para treinamento são limitados em tamanho. A geração de dados sintéticos permite expandir o conjunto de dados original, proporcionando mais exemplos para o modelo aprender.

O aumento de dados é especialmente útil quando o conjunto de dados real é pequeno ou desbalanceado, o que pode levar a modelos de classificação

[6] https://chatgpt.com/
[7] https://openai.com/

enviesados ou com desempenho limitado em certas classes.

2. Melhoria da Generalização

A introdução de dados sintéticos pode ajudar a melhorar a capacidade do modelo de generalizar para novos exemplos. Isso é crucial para garantir que o modelo funcione bem em dados de teste ou em situações do mundo real que podem não ter sido totalmente representadas no conjunto de treinamento original.

3. Mitigação de *Overfitting*

Gerar dados sintéticos pode reduzir o risco de *overfitting*, um problema comum em modelos de aprendizado de máquina nos quais o modelo se ajusta em excesso aos dados de treinamento e tem dificuldade em generalizar para novos dados.

Introduzir variação por meio de dados sintéticos pode ajudar a evitar que o modelo se torne muito específico para os exemplos do conjunto de treinamento.

4. Redução da Dependência do Volume de Dados Reais

Em algumas situações, como na área da saúde, pode ser difícil obter grandes volumes de dados rotulados. A geração de dados sintéticos permite treinar modelos eficazes mesmo em cenários em que a disponibilidade de dados reais é limitada.

5.9 Comitês de Ética

A coleta dos dados teve anuência do responsável pelo serviço de cirurgia cardiovascular do Hospital São Francisco, e a pesquisa foi aprovada pelos Comitês de Ética em Pesquisa da Universidade Federal de Ciências da Saúde de Porto Alegre e da Irmandade Santa Casa de Misericórdia de Porto Alegre, sob os CAAE 67548123.6.0000.5345 e 73285323.2.0000.5335 respectivamente (Anexos I e II).

5.10 Material de Pesquisa

Foram utilizados prontuários sintéticos e também prontuários reais durante a

realização deste estudo. Os textos sintéticos foram usados para aumentar a quantidade de dados disponíveis para treinamento dos modelos de aprendizado de máquina, após avaliação pelo autor deste estudo, que possui experiência na área de cardiologia. Isso é especialmente útil quando os dados reais são limitados ou insuficientes. Os textos reais de PME foram essenciais para garantir que os modelos de aprendizado de máquina fossem treinados em dados que refletiam as condições e cenários do mundo real.

Os textos reais permitiram avaliar o desempenho dos modelos em condições reais, fornecendo uma medida mais precisa de como o modelo provavelmente sairá na prática. Os dados utilizados foram obtidos de prontuários eletrônicos de pacientes e acessados apenas após a aprovação dos respectivos comitês éticos das instituições envolvidas. As amostras passaram por pré-processamento para adequá-las para uso nas etapas posteriores. Nesta etapa foram utilizadas bibliotecas da linguagem Python específicas para este fim.

5.10.1 Aquisição das Amostras de PMEs

O material utilizado na pesquisa foi obtido a partir dos PME de pacientes atendidos no Hospital São Francisco da Santa Casa de Porto Alegre, registrados no sistema de prontuários eletrônicos Tasy® (PISSAI, 2021), ou seja, são dados do mundo real. As etapas seguidas para obtenção dos PME estão resumidas na figura 11.

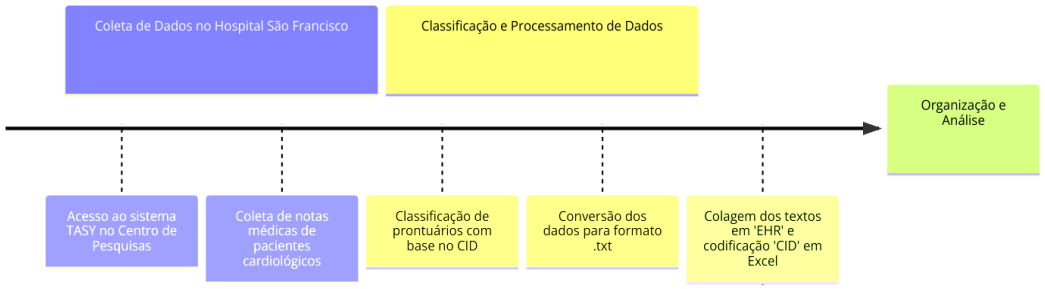


Figura 11 - Etapas para obtenção de PME do mundo real.

Fonte: O próprio autor.

Esse tipo de prontuário apresenta os dados na forma estruturada, em que existem campos para Código Internacional de Doenças (CID), que é atribuído pelo

médico responsável pelo paciente. Os prontuários também contêm dados na forma não-estruturada com um campo para descrições de notas clínicas escritas na forma de texto livre.

Nesta pesquisa, o *corpus* será classificado com base no código CID, que deve refletir o diagnóstico predominante acerca da condição de saúde do paciente internado na instituição. Essa classificação é definida pelos médicos assistentes, que possuem contato direto com o paciente, e por isso podem ser considerados suficientemente qualificados na atribuição do rótulo correto ao prontuário.

A forma de registro principal dentro do TASY® mais caracterizada como instrumento de comunicação é o campo da Evolução (Figura 13), que funciona como um registro escrito em que os profissionais devem descrever a avaliação clínica. Comumente, trazem relatos das condutas estabelecidas do tratamento, trazendo as informações que indiquem a manutenção ou alteração do plano terapêutico.

As evoluções podem ser divididas pelas categorias profissionais de quem a registrou e pelo horário e data em que foi inserida, pois a nota clínica deve ser tratada como documento único. Cores diferentes indicam o tipo de profissional que preencheu aquela evolução – e.g., verde claro para os enfermeiros, amarelo para os médicos. Ao mesmo tempo em que esta divisão organiza e facilita o acesso à informação de interesse, pode trazer uma unidirecionalidade que prejudica o compartilhamento de informações entre os diferentes profissionais.

As amostras de PME obtidas para a realização deste trabalho foram coletadas através do acesso ao sistema, realizado através de um computador presente no Centro de Pesquisas em Cirurgia Cardiovascular, localizado no mezanino do Hospital São Francisco.

A estratégia de seleção adotada nesta pesquisa consistiu na coleta de notas médicas de pacientes com alguma condição cardiológica, e que estavam internados em uma duas UTIs do hospital, sendo uma destinada para atender pacientes em tratamento clínico, e outra para pacientes que passaram por procedimento cirúrgico. Os pacientes estiveram internados entre Setembro de 2021 e Janeiro de 2022.

A seleção de PME de pacientes internados nas unidades de tratamento intensivo foi uma decisão tomada pelos pesquisadores , sendo assumido que estes pacientes recebem atenção de forma mais contínua, e sendo assim, suas notas clínicas refletiam melhor suas condições de saúde. A conversão dos arquivos em formato .txt se deu de forma manual, via TASY®, copiando os textos do sistema e

colando para um arquivo .txt, em que cada arquivo recebeu uma identificação sobre o CID. Esses arquivos de texto foram colados em uma coluna chamada “EHR” de uma planilha no software Excel. Em outra coluna, chamada “CID” foi escrito o código atribuindo o diagnóstico do prontuário do paciente correspondente.

A Figura 12 ilustra um exemplo de como as notas clínicas geralmente se encontram registradas no sistema TASY®.

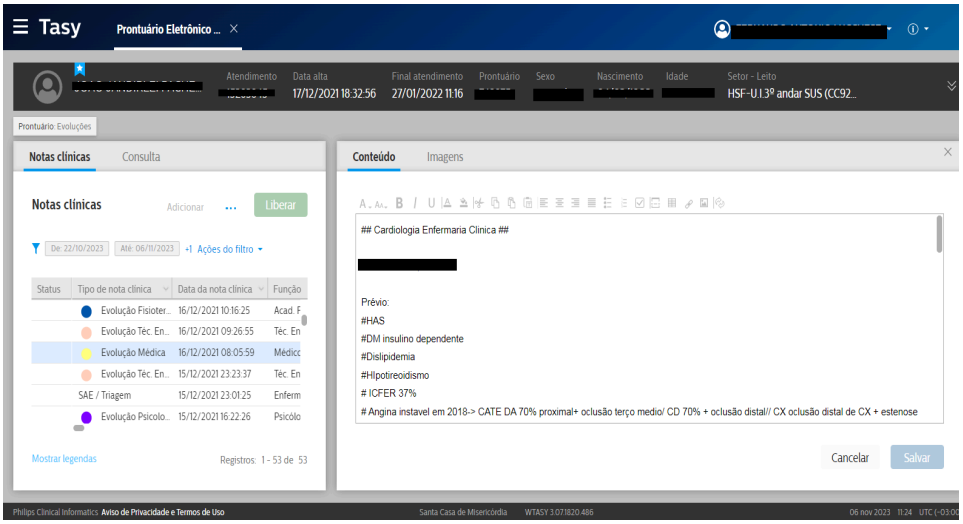


Figura 12 - Captura de tela de um PME registrado no TASY®.
Fonte: Sistema TASY.

5.10.2 Análise Estatística das Amostras de PME

Realizou-se uma análise de ocorrência de amostras obtidas a partir da coleta do material. Os diagnósticos mais frequentes foram visualizados através da construção de uma gráfico do tipo histograma, em que o eixo X mostra o código CID informado no PME, o eixo Y mostra a quantidade, conforme Figura 13.

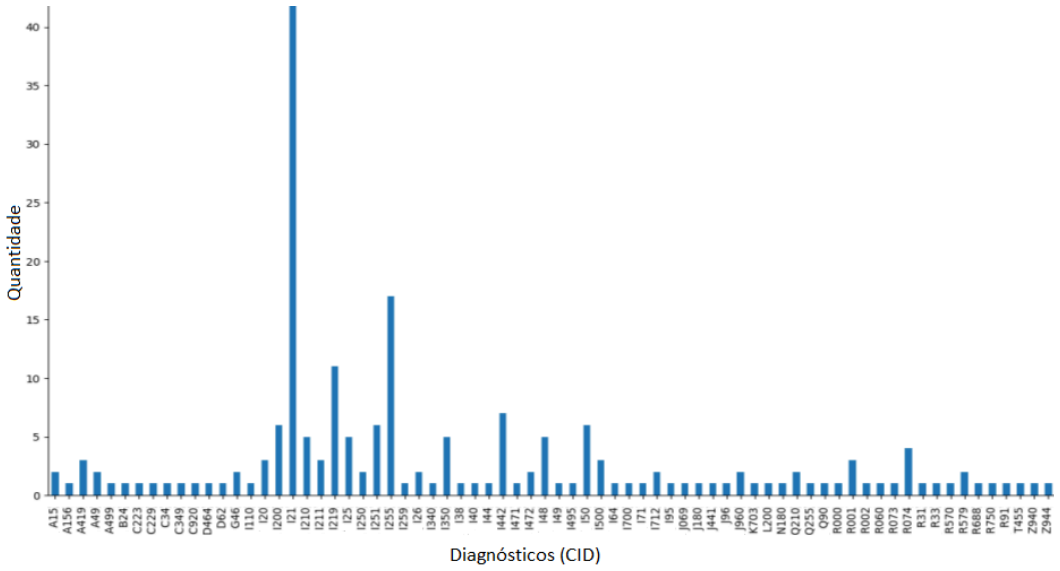


Figura 13 - Distribuição de frequências de diagnósticos por CID.
Fonte: Próprio Autor.

5.11 Geração do dataset sintético de PME

Para criação dos prontuários utilizou-se ChatGPT versão 3.5 que é treinado com uma grande variedade de textos disponíveis na internet, permitindo-lhe gerar texto, responder perguntas, resumir informações, traduzir línguas e executar muitas outras tarefas linguísticas[8].

O processo foi dividido em duas etapas principais: uma de criação do modelo de prontuários; e outra de criação dos prontuários com diferentes diagnósticos.

5.11.1 Interação com o ChatGPT para criação do modelo PME

As etapas sequenciais até a obtenção de um modelo de prontuário simulado adequado, de forma a refletir um PME real de um paciente atendido em um hospital de cardiologia estão representadas na figura 14.

[8] https://openai.com/index/chatgpt/

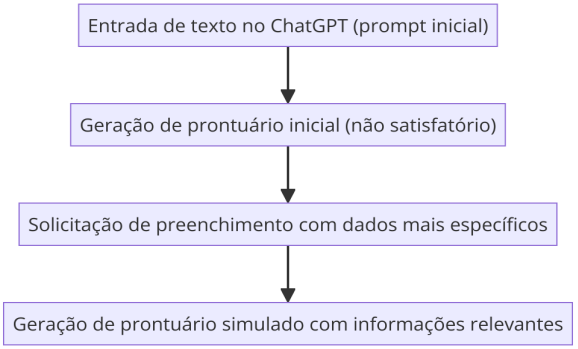


Figura 14 - Etapas para geração do modelo de PME sintético.
Fonte: Próprio Autor.

5.11.2 Resultados obtidos a partir da interação para criação do modelo PME

Através do campo de entrada de texto, chamado *prompt,* solicitou-se ao ChatGPT a geração de paciente cardiológico que apresentou fibrilação arterial, escrito por enfermeiro. O resultado gerado a partir desse primeiro comando não foi satisfatório, pois gerou um prontuário incompleto, com campos de dados sem todas as informações relevantes e com alguns dados faltantes. Os comandos inseridos no prompt e as respostas obtidas estão no Quadro 4.

Quadro 4 - Sumarização dos comandos e resultados obtidos.

Prompt	**Resposta**
Simule um prontuário médico de um paciente cardiológico que apresentou fibrilação arterial, escrito por enfermeiro	Data: [data de atendimento] Nome do Paciente: [Nome Completo] Idade: [Idade] Sexo: [Masculino/Feminino] Data de Nascimento: [Data de Nascimento] Profissão: [Profissão] Queixa Principal: O paciente apresentou-se na unidade de saúde com queixa de palpitações intensas e falta de ar. Histórico Médico: Antecedentes Pessoais: [Lista de doenças pré-existentes, como hipertensão arterial, diabetes, etc.] Cirurgias prévias, se aplicável. Histórico Familiar: [Mencionar doenças cardíacas ou outras condições relevantes na família] Hábitos de Vida: Tabagismo, se aplicável Consumo de álcool e outras substâncias, se aplicável Atividades físicas e dieta habitual

	Exame Físico: Sinais Vitais: Pressão arterial: [Valor mmHg] Frequência cardíaca: [Valor bpm] Frequência respiratória: [Valor rpm] Temperatura: [Valor ºC] Estado Geral: [Descrever a aparência geral do paciente] Exame Cardiovascular: Bulhas cardíacas rítmicas Sopros cardíacos, se presentes Pulso irregular, auscultado e palpado Edema, se presente e em que região Outros Achados Relevantes: [Mencionar quaisquer outros achados físicos relevantes] Exames Complementares: Eletrocardiograma (ECG): [Descrição dos resultados do ECG] Outros Exames Cardiológicos, se aplicável: [Resultado de exames de sangue, ecocardiograma, teste ergométrico, etc.] Diagnóstico: Com base nos sintomas apresentados e nos resultados dos exames, o paciente foi diagnosticado com fibrilação arterial (ou atrial). Conduta: Medicamentos Prescritos: [Lista dos medicamentos, doses e frequência de administração] Orientações ao Paciente: Repouso relativo Evitar tabagismo e consumo excessivo de álcool Dieta equilibrada e baixa em sal Monitorar frequência cardíaca e pressão arterial regularmente Importância de comparecer a consultas de acompanhamento Encaminhamentos: O paciente foi encaminhado para consulta com o cardiologista para avaliação detalhada e acompanhamento do caso. Assinatura do Enfermeiro: [Nome do Enfermeiro] CRN: [Número do Conselho Regional de Enfermagem]
Preencha os campos com informações aleatórias, mas condizentes com a realidade	Data: 27 de julho de 2023 Nome do Paciente: João da Silva Idade: 58 anos Sexo: Masculino Data de Nascimento: 12 de março de 1965 Profissão: Aposentado Queixa Principal: O paciente apresentou-se na unidade de saúde com queixa de palpitações intensas e falta de ar. Histórico Médico: Antecedentes Pessoais: Hipertensão arterial controlada com medicamentos. Diabetes tipo 2 em tratamento com dieta e medicação. Cirurgia de hérnia inguinal realizada em 2010. Histórico Familiar: Pai faleceu em decorrência de um infarto aos 67 anos. Mãe possui hipertensão e diabetes. Hábitos de Vida: Ex-fumante (parou há 10 anos). Consumo moderado de álcool. Caminhadas diárias e alimentação balanceada.

	Exame Físico: Sinais Vitais: Pressão arterial: 140/90 mmHg Frequência cardíaca: 110 bpm Frequência respiratória: 20 rpm Temperatura: 36,8 ºC Estado Geral: Paciente consciente, orientado e colaborativo. Exame Cardiovascular: Bulhas cardíacas rítmicas. Ausência de sopros cardíacos. Pulso irregular, característico da fibrilação arterial. Sem edema periférico. Outros Achados Relevantes: Pele com boa coloração e elasticidade. Ausculta pulmonar sem alterações significativas. Exames Complementares: Eletrocardiograma (ECG): Apresenta ritmo de fibrilação arterial. Outros Exames Cardiológicos, se aplicável: Hemograma, colesterol e triglicerídeos dentro dos valores de referência. Ecocardiograma agendado para avaliar a função cardíaca. Diagnóstico: Com base nos sintomas apresentados e nos resultados dos exames, o paciente foi diagnosticado com fibrilação arterial. Conduta: Medicamentos Prescritos: Digoxina 0,25 mg, 1 comprimido via oral, 1 vez ao dia. Varfarina 5 mg, 1 comprimido via oral, 1 vez ao dia. Atenolol 50 mg, 1 comprimido via oral, 2 vezes ao dia. Orientações ao Paciente: Evitar esforços físicos intensos. Manter a dieta equilibrada e pobre em sal. Monitorar a pressão arterial e frequência cardíaca diariamente. Comparecer à consulta com o cardiologista agendada para a próxima semana. Encaminhamentos: O paciente foi encaminhado para consulta com o cardiologista para avaliação detalhada e acompanhamento do caso. Assinatura do Enfermeiro: Maria Santos CRN: 12345

Em seguida, solicitou-se o preenchimento dos campos com informações aleatórias, mas condizentes com a realidade. Como resultado da segunda solicitação, gerou-se um prontuário simulado, com informações condizentes com o diagnóstico de fibrilação atrial, incluindo recomendações de medicamentos.

5.11.3 Interação com o ChatGPT para geração de PMEs

A próxima etapa consistiu na solicitação ao ChatGPT para que gerasse 20 prontuários sintéticos com o diagnóstico de Infarto Agudo do Miocárdio; 20

prontuários sintéticos com o diagnóstico de Aneurisma de Aorta; 20 prontuários sintéticos com o diagnóstico de Bradicardia; 20 prontuários sintéticos com o diagnóstico de angina; 20 prontuários sintéticos com o diagnóstico de Mixoma Cardíaco; 20 prontuários sintéticos com o diagnóstico de Insuficiência Cardíaca e 20 prontuários sintéticos com o diagnóstico de Insuficiência Mitral. Isso totalizou 140 prontuários sintéticos de pacientes com condições cardiológicas.

O número de 20 prontuários gerados para cada prontuário, entre os mais frequentes, objetivou balancear a quantidade de prontuários reais. Os passos seguidos para isso estão ilustrados na figura 15.

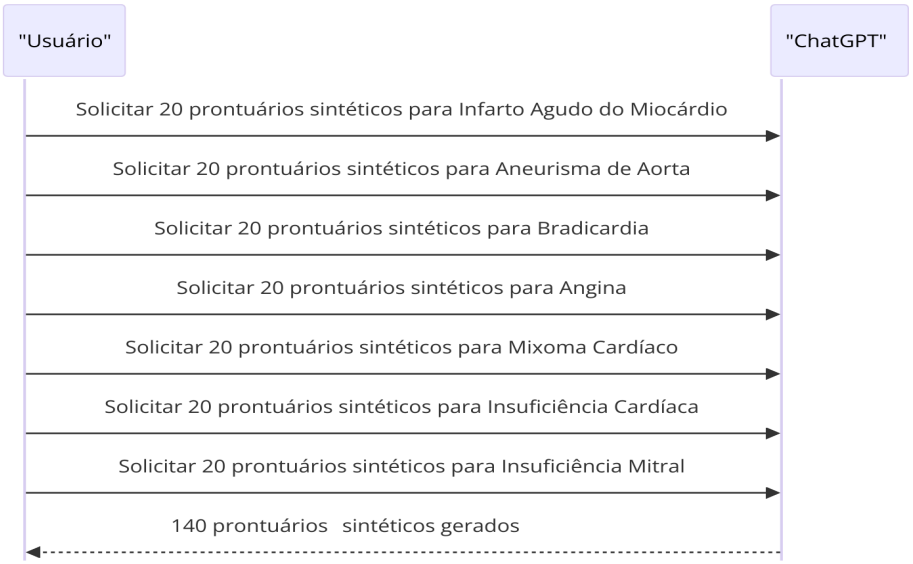


Figura 15 - Etapas para geração das amostras de PME sintéticos.
Fonte: O próprio Autor.

5.12 Pré-Processamento das Amostras

5.12.1 Anonimização dos PME Reais

Para preservar a privacidade e anonimato dos dados contidos nos Prontuários Médicos Eletrônicos (PME), foi desenvolvida uma função específica (ver Apêndice A) para realizar a anonimização. Essa função foi projetada para ler as amostras de texto, identificar e remover os nomes masculinos e femininos mais comuns na língua portuguesa, conforme listas fornecidas pelo Instituto Brasileiro de Geografia e Estatística (IBGE).

Dessa forma, qualquer menção aos nomes de pacientes e médicos, que porventura tenham sido incluídos inadvertidamente nos trechos de texto extraídos dos PME, foi completamente eliminada das amostras finais. Esse processo assegura que os dados analisados se mantenham em conformidade com as regulamentações de proteção à privacidade e confidencialidade, garantindo a segurança e integridade das informações.

5.12.2 Limpeza das amostras de PME

O processamento descrito a seguir foi implementado em linguagem Python e pode ser encontrado no Apêndice B. Inicialmente, são definidas expressões regulares para remoção de padrões específicos de texto. A expressão "pattern" é utilizada para identificar e remover sequências no formato '(números)' da coluna 'CID', enquanto a expressão "pattern_evol" é empregada para detectar e eliminar padrões que contenham caracteres como ,, ;, ., :, #, -, +, /, (,), e números na coluna 'EHR'. Em seguida, são aplicadas as operações de substituição de padrões definidos pelas expressões regulares nas colunas 'CID' e 'EHR', respectivamente.

Além disso, é realizada a conversão de todos os caracteres da coluna 'EHR' para minúsculas e a remoção de caracteres não alfanuméricos, substituindo-os por espaços em branco. Para a remoção de palavras irrelevantes, é feito o download da lista de stopwords da língua portuguesa disponíveis na biblioteca Natural Language Toolkit (NLTK)[9], que fornece um conjunto abrangente de ferramentas e recursos para lidar com texto e linguagem, permitindo que desenvolvedores e pesquisadores implementem soluções avançadas de análise e processamento de texto.. A função "remove_stop_words()" é então definida para eliminar essas stopwords da coluna 'EHR' do DataFrame, resultando em uma nova série de dados onde essas palavras são removidas.

[9] https://www.nltk.org/

5.12.3 Transformação dos PME em vetores

No contexto do uso de Long Short-Term Memory (LSTM), foi necessário converter as amostras de texto para o formato de vetores. Um vetor representa uma sequência de valores numéricos que codificam informações relevantes para a tarefa em questão. Essa sequência pode ter diferentes comprimentos e cada valor no vetor pode representar um tipo distinto de informação. Essa transformação está relacionada à criação de uma camada de inserção a ser empregada em uma rede neural recorrente, como é o caso da rede LSTM.

Os valores específicos para os parâmetros input_dim, output_dim e input_length variam de acordo com as características do conjunto de dados utilizado. O objetivo fundamental dessa camada é aprender representações densas e significativas para as palavras, o que tem o potencial de aprimorar o desempenho da rede em tarefas de Processamento de Linguagem Natural (NLP), como classificação de texto, tradução automática, entre outras.

6. ANÁLISE E DISCUSSÃO DE RESULTADOS

Neste capítulo será realizada uma exploração dos resultados obtidos através dos modelos de classificação aplicados aos prontuários médicos. Inicialmente, será apresentada uma análise estatística abrangente da distribuição dos diagnósticos nos dados analisados. Posteriormente, serão detalhados os processos envolvidos na criação dos modelos de classificação de textos médicos.

6.1 Análise estatística dos diagnósticos dos registros eletrônicos

As ocorrências mais frequentes foram do diagnóstico CID I21, correspondente ao Infarto agudo do miocárdio, com 50 amostras; CID I255, referente à Miocardia Isquêmica, com 30 amostras; CID I219, referente ao Infarto Agudo do Miocárdio Não Especificado, com 11 amostras. Os resultados podem ser observados na Tabela 1.

Tabela 1 - Proporção dos diagnósticos dos prontuários médicos.

Diagnóstico	Quantidade	Proporção das amostras
Infarto Agudo do Miocardio (I21)	50	25%
Miocardia Isquêmica (I255)	30	15%
Infarto Agudo do Miocárdio Não Especificado (I219)	11	5,5%
Angina (I200)	9	4,5%
Insuficiência Cardíaca (I50)	9	4,5%
Bloqueio Atrioventricular Total (I442)	7	3,5%
Flutter e fibrilação atrial (I48)	5	2,5%
Estenose da valva aórtica (I350)	5	2,5%
Dor Torácica (R07)	4	2,0%
Bradicardia não especificada (R001)	3	1,5%
Septicemia (I419)	3	1,5%
Outros (<=2 casos)	64	32%

Fonte: Próprio Autor

Como observado, o Infarto agudo é diagnóstico mais prevalente, e Miocardiopatia Isquêmica, que é seu diagnóstico clinicamente relacionado. As doenças cardíacas, como o infarto do miocárdio, representam um problema de saúde pública global, com alta prevalência e impacto significativo na mortalidade. Segundo a Sociedade Brasileira de Cardiologia (SBC), as doenças

cardiovasculares (DCV) são a principal causa de morte no Brasil, respondendo por cerca de 30% dos óbitos no país.

Estima-se que 12 milhões de pessoas sofram infarto agudo do miocárdio (IAM) a cada ano, e cerca de 7 milhões morram em decorrência da doença. A SBC estima que 300 mil pessoas sofram IAM anualmente, com cerca de 200 mil mortes. A prevalência de IAM aumenta com a idade, sendo mais comum em homens do que em mulheres.

Da mesma forma observou-se o diagnóstico de Bloqueio atrioventricular total (I442). Essa é uma condição médica na qual há um bloqueio completo da condução dos impulsos elétricos do átrio para o ventrículo. Isso significa que os átrios e os ventrículos do coração batem em ritmos diferentes, o que pode levar a uma série de problemas cardíacos, incluindo bradicardia (frequência cardíaca lenta), síncope (desmaio) e insuficiência cardíaca. A prevalência do BAVT na população geral é estimada em torno de 0,05% a 0,2%.

6.2 Criação dos modelos classificadores de prontuários

6.2.1 Criação do modelo utilizando Naive Bayes

Os passos seguidos para criação do modelo utilizando o algoritmo Naive Bayes estão ilustrados na figura 16.

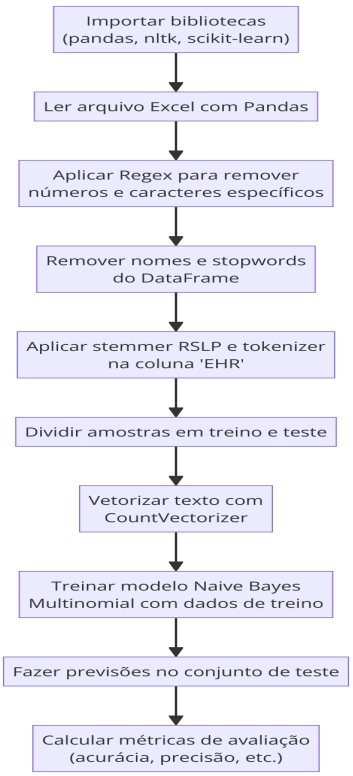


Figura 16 - Etapas para criação do modelo com Naive Bayes.
Fonte: O próprio autor.

O código começa importando as bibliotecas necessárias, como pandas, nltk e módulos do scikit-learn para machine learning. Em seguida, é lido um arquivo Excel contendo amostras utilizando o Pandas (pd.read_excel). Expressões regulares são aplicadas para remover números e parênteses da coluna 'CID', além de outros caracteres específicos da coluna 'EHR'. A função remove_names_from_dataframe é definida para eliminar nomes masculinos e femininos do conjunto de dados CSV.

Uma lista de stopwords em português é carregada do NLTK, e a função remove_stop_words é criada para excluir essas stopwords da coluna 'EHR' do DataFrame. Funções como aplicaStemmer e aplicaTokenizer são definidas para aplicar o stemmer RSLP e um tokenizer do NLTK, respectivamente, na coluna 'EHR' do DataFrame.

As amostras são divididas em conjuntos de treinamento e teste (30% das amostras) utilizando a função train_test_split do sklearn. O CountVectorizer do sklearn é utilizado para transformar os textos em vetores de contagem, tanto para os conjuntos de treinamento quanto para os de teste.

O modelo Naive Bayes Multinomial é treinado usando o MultinomialNB do

sklearn, que é um algoritmo de classificação probabilística robusto e eficiente, ideal para textos e dados categóricos, e se baseia no modelo Naive Bayes e na distribuição multinomial para calcular as probabilidades de um dado pertencer a cada classe. Posteriormente, o modelo é utilizado para fazer previsões no conjunto de testes. Diversas métricas de avaliação, como acurácia, precisão, revocação (recall), F-Score e matriz de confusão, são calculadas e impressas no console. O script Python pode ser encontrado no Apêndice C.

6.2.2 Criação do modelo utilizando Regressão Logística

Os passos seguidos para criação do modelo utilizando o algoritmo Regressão Logística estão ilustrados na figura 17.

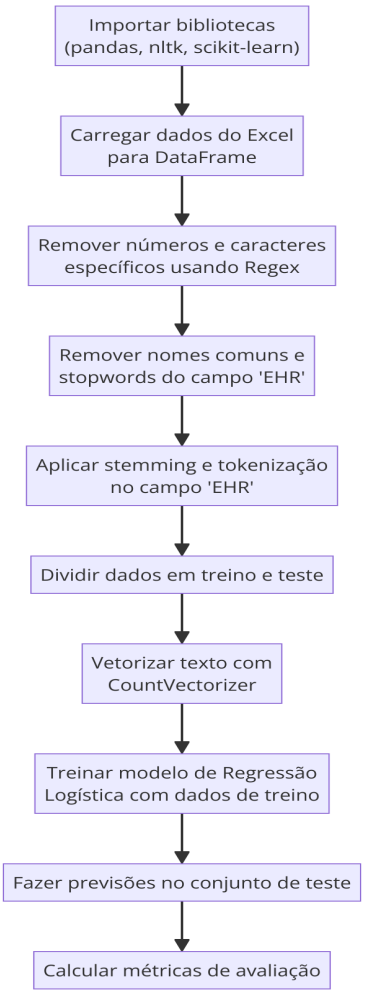


Figura 17 - Etapas para criação do modelo com Regressão Logística.

Fonte: O próprio autor.

O código começa com a importação das bibliotecas necessárias para manipulação de dados, processamento de linguagem natural e modelagem: pandas, nltk e vários módulos do scikit-learn para machine learning (model_selection, feature_extraction, linear_model e metrics). Carrega os dados a partir de um arquivo Excel chamado “Planilha_EHR.xlsx” para um DataFrame usando o pandas. Remove números e caracteres específicos dos campos 'CID' e 'EHR' no DataFrame com a utilização de Expressões Regulares.

Segue-se a remoção de nomes masculinos e femininos comuns do campo 'EHR' usando a função ***remove_names_from_dataframe*** e remoção de *stopwords* do campo 'EHR' utilizando a biblioteca NLTK. Aplicou-se *stemming* ao campo 'EHR' para reduzir palavras às suas raízes, e tokenização do campo 'EHR', dividindo o texto em palavras individuais. O próximo passo foi dividir os dados em conjuntos de treino e teste (30% das amostras) e vetorizar o texto usando o CountVectorizer.

Segue-se o treinamento do modelo onde cria-se e treina-se o modelo de Regressão Logística com os dados de treino. Depois disso, faz-se previsões no conjunto de teste e calcula métricas de avaliação, incluindo acurácia, precisão, revocação (recall) e F-Score. O script Python pode ser encontrado no Apêndice D.

6.2.3 Criação do modelo utilizando Árvore de Decisão

Os passos seguidos para criação do modelo utilizando o algoritmo Árvore de Decisão estão ilustrados na figura 18.

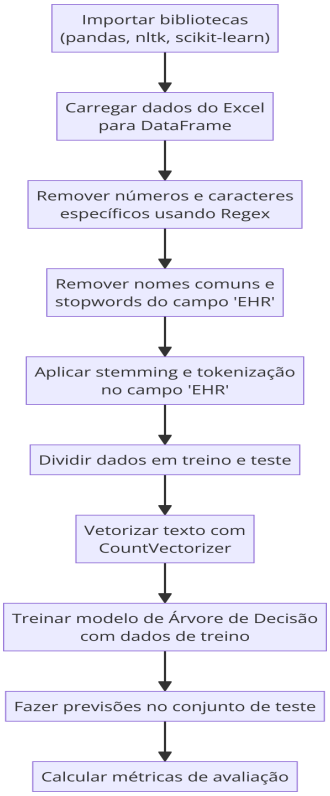


Figura 18 - Etapas para criação do modelo com Árvores de Decisão.

Fonte: O próprio autor.

Importar as bibliotecas necessárias para manipulação de dados, processamento de linguagem natural e modelagem: pandas, nltk e vários módulos do scikit-learn para machine learning (model_selection, feature_extraction , linear_model e metrics).

Carrega os dados a partir de um arquivo Excel chamado “Planilha_EHR.xlsx” para um DataFrame usando o pandas. Remove números e caracteres específicos dos campos 'CID' e 'EHR' no DataFrame com a utilização de Expressões Regulares.

Segue-se a remoção de nomes masculinos e femininos comuns do campo 'EHR' usando a função remove_names_from_dataframe e remoção de stopwords do campo 'EHR' utilizando a biblioteca NLTK. Aplicou-se stemming ao campo 'EHR' para reduzir palavras às suas raízes, e tokenização do campo 'EHR', dividindo o texto em palavras individuais.

O próximo passo foi dividir os dados em conjuntos de treino e teste (30% das amostras) e vetorizar o texto usando o CountVectorizer. Segue-se o treinamento do modelo onde cria-se e treina-se o modelo de Regressão Logística com os dados de

treino. Depois disso, faz-se previsões no conjunto de teste e calcula métricas de avaliação, incluindo acurácia, precisão, revocação (recall) e F-Score. O script Python pode ser encontrado no Apêndice E.

6.2.4 Criação do modelo utilizando Floresta Aleatória

Os passos seguidos para criação do modelo utilizando o algoritmo Floresta Aleatória estão ilustrados na figura 19.

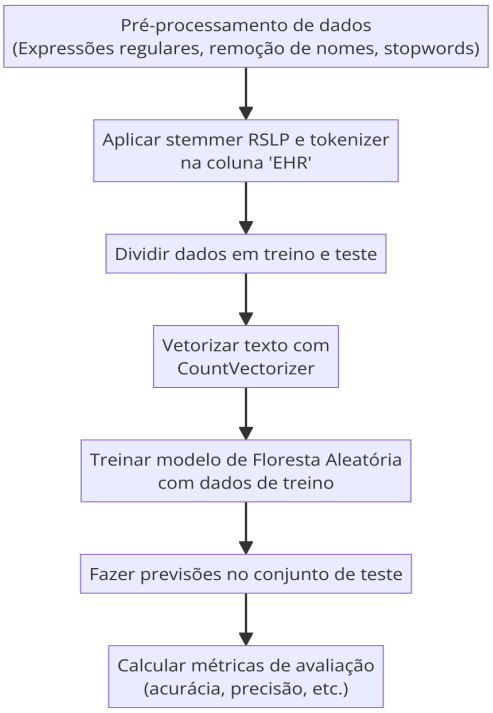


Figura 19 - Etapas para criação do modelo com Floresta Aleatória.

Fonte: O próprio autor.

O processo de utilização de Floresta Aleatória inicia-se com o pré-processamento dos dados. Isso envolve a aplicação de expressões regulares para remover números e caracteres específicos das colunas 'CID' e 'EHR', visando limpar o texto e eliminar elementos irrelevantes para a análise.

Além disso, a função remove_names_from_dataframe é empregada para eliminar nomes masculinos e femininos das amostras presentes na coluna 'EHR', garantindo a privacidade dos pacientes e médicos. Em seguida, as stopwords, palavras comuns que não possuem significado relevante para a análise, são removidas da coluna 'EHR' utilizando a função remove_stop_words, com o objetivo

de reduzir o ruído nos dados.

O stemmer RSLP do NLTK é então aplicado à coluna 'EHR' através da função aplicaStemmer, reduzindo as palavras às suas raízes e simplificando a análise. Por fim, um tokenizer do NLTK é utilizado para dividir o texto em palavras individuais com a função aplicaTokenizer, preparando os dados para a próxima etapa.

Após o pré-processamento, os dados são divididos em conjuntos de treinamento e teste (30% das amostras) utilizando a função train_test_split do scikit-learn. Essa divisão é fundamental para avaliar a capacidade de generalização do modelo em dados não vistos durante o treinamento.

Em seguida, o texto é vetorizado utilizando o CountVectorizer do scikit-learn, transformando-o em vetores de contagem. Cada palavra única no texto é representada por um número inteiro, e a contagem de ocorrências de cada palavra é armazenada no vetor. Isso é feito separadamente para os conjuntos de treinamento e teste.

O modelo de Floresta Aleatória é então criado e treinado com os dados de treinamento utilizando o RandomForestClassifier do scikit-learn. A Floresta Aleatória é um algoritmo de aprendizado supervisionado que combina múltiplas árvores de decisão para realizar a classificação ou regressão.

Finalmente, métricas de avaliação, como acurácia, precisão, revocação (recall) e Escore F1, são calculadas com base nas previsões do modelo no conjunto de teste. Essas métricas são essenciais para avaliar o desempenho do modelo e determinar sua eficácia na classificação dos dados. Os resultados das métricas são então impressos no console para análise e interpretação. O script Python pode ser encontrado no Apêndice F.

6.2.5 Criação do modelo utilizando KNN

Os passos seguidos para criação do modelo utilizando o algoritmo KNN estão ilustrados na figura 20.

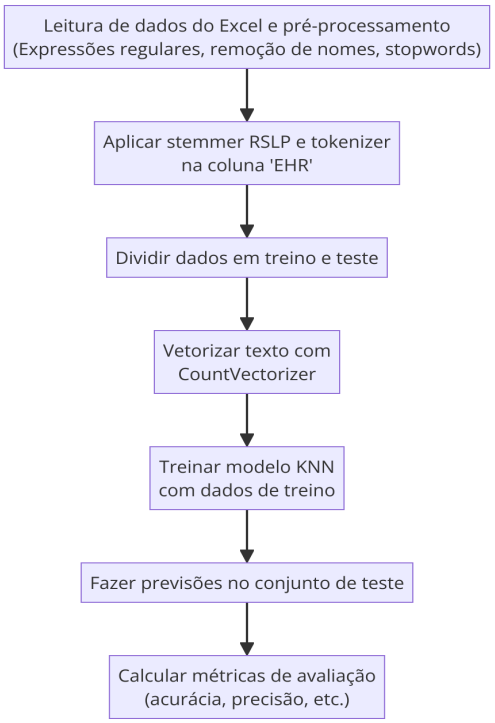


Figura 20 - Etapas para criação do modelo com KNN.

Fonte: O próprio autor.

Primeiramente, os dados são lidos de um arquivo Excel para um DataFrame. Em seguida, são aplicadas expressões regulares para remover números e outros caracteres específicos das colunas 'CID' e 'EHR', com o objetivo de limpar o texto e eliminar elementos irrelevantes para a análise. Além disso, a função remove_names_from_dataframe é utilizada para remover nomes masculinos e femininos das amostras presentes nas colunas 'EHR', visando proteger a privacidade dos pacientes e médicos.

Após o pré-processamento, as stopwords são removidas da coluna 'EHR' utilizando a função remove_stop_words, reduzindo o ruído nos dados. Em seguida, um stemmer RSLP do NLTK é aplicado à coluna 'EHR' utilizando a função aplicaStemmer, reduzindo as palavras às suas raízes e simplificando a análise. Adicionalmente, um tokenizer do NLTK é aplicado à coluna 'EHR' utilizando a função aplicaTokenizer, dividindo o texto em palavras individuais e preparando os dados para a próxima etapa.

As amostras são então divididas em conjuntos de treinamento e teste (30% das amostras) utilizando a função train_test_split do scikit-learn. Em seguida, o CountVectorizer do sklearn é utilizado para transformar o texto em vetores de

contagem. Cada palavra única no texto é representada por um número inteiro, e a contagem de ocorrências de cada palavra é armazenada no vetor. Essa vetorização é realizada separadamente para os conjuntos de treinamento e teste.

Por fim, o modelo KNeighborsClassifier do sklearn é utilizado para criar e treinar um modelo k-NN com os dados de treinamento. Após o treinamento, métricas de avaliação como acurácia, precisão, revocação (recall) e Escore F1 são calculadas com base nas previsões do modelo no conjunto de teste, e os resultados são impressos no console para análise. Essas métricas são essenciais para avaliar o desempenho do modelo e determinar sua eficácia na classificação dos dados. O script Python pode ser encontrado no Apêndice G.

6.2.6 Criação do modelo utilizando SVM

Os passos seguidos para criação do modelo utilizando o algoritmo SVM estão ilustrados na figura 21.

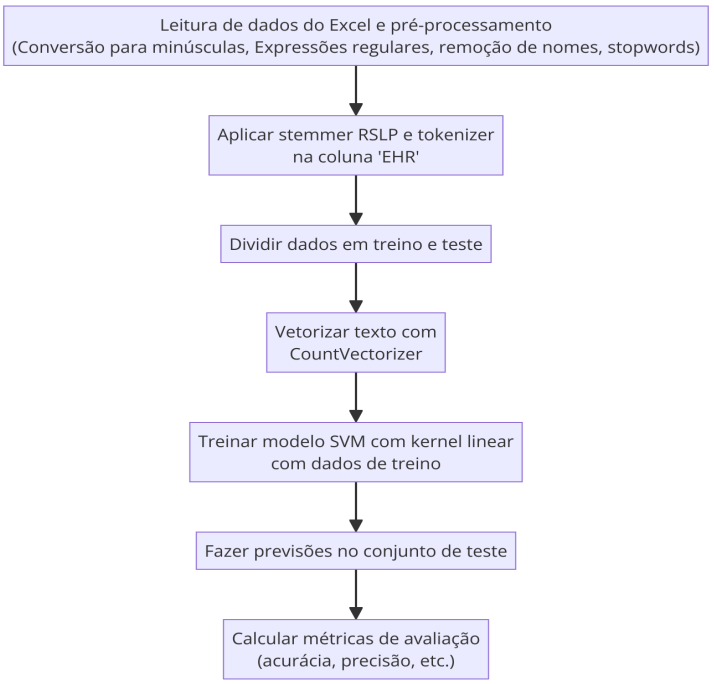


Figura 21 - Etapas para criação do modelo com SVM.

Fonte: O próprio autor.

Primeiramente, os dados são lidos de um arquivo Excel. A coluna 'EHR' é convertida para minúsculas e, em seguida, expressões regulares são aplicadas para

remover números e outros caracteres específicos das colunas 'CID' e 'EHR', a fim de limpar o texto e eliminar elementos irrelevantes para a análise. Em seguida, a função remove_names_from_dataframe é utilizada para remover nomes masculinos e femininos das amostras presentes nas colunas 'EHR', visando proteger a privacidade dos pacientes e médicos.

Após o pré-processamento, as stopwords são removidas da coluna 'EHR' utilizando a função remove_stop_words, reduzindo o ruído nos dados. Um stemmer RSLP do NLTK é então aplicado à coluna 'EHR' usando a função aplicaStemmer, reduzindo as palavras às suas raízes e simplificando a análise. Além disso, um tokenizer do NLTK é aplicado à coluna 'EHR' usando a função aplicaTokenizer, dividindo o texto em palavras individuais e preparando os dados para a próxima etapa. As amostras são divididas em conjuntos de treinamento e teste (30% das amostras) utilizando a função train_test_split do scikit-learn. Em seguida, o CountVectorizer do scikit-learn é utilizado para transformar o texto em vetores de contagem. Isso é feito separadamente para os conjuntos de treinamento e teste.

Por fim, o SVC (Support Vector Classifier) do scikit-learn é utilizado para criar e treinar um modelo SVM com kernel linear usando os dados de treinamento. Após o treinamento, métricas de avaliação como acurácia, precisão, revocação (recall) e Escore F1 são calculadas com base nas previsões do modelo no conjunto de teste, e os resultados são impressos no console para análise. Essas métricas são essenciais para avaliar o desempenho do modelo e determinar sua eficácia na classificação dos dados. O script Python pode ser encontrado no Apêndice H.

6.2.7 Criação do modelo utilizando MLP

Os passos seguidos para criação do modelo com MLP estão ilustrados na figura 22.

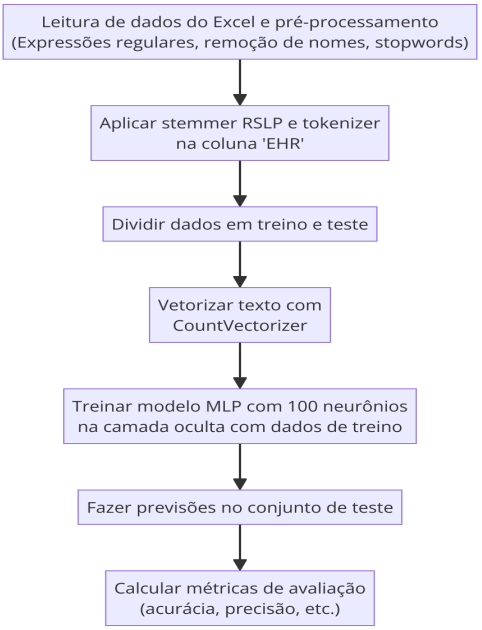


Figura 22 - Etapas para criação do modelo com MLP.

Fonte: O próprio autor.

Primeiramente, os dados são lidos de um arquivo Excel. Em seguida, expressões regulares são aplicadas para remover números e outros caracteres específicos das colunas 'CID' e 'EHR', visando limpar o texto e eliminar elementos irrelevantes para a análise. Além disso, a função remove_names_from_dataframe é utilizada para remover nomes masculinos e femininos da coluna 'EHR', contribuindo para proteger a privacidade dos pacientes.

Após o pré-processamento, as stopwords são removidas da coluna 'EHR' utilizando a função remove_stop_words, visando reduzir o ruído nos dados. Um stemmer RSLP do NLTK é então aplicado à coluna 'EHR' usando a função aplicaStemmer, simplificando as palavras para suas raízes e facilitando a análise. Além disso, um tokenizer do NLTK é aplicado à coluna 'EHR' utilizando a função aplicaTokenizer, dividindo o texto em palavras individuais para preparar os dados para a próxima etapa.

As amostras são divididas em conjuntos de treinamento e teste (30% das amostras) utilizando a função train_test_split. Em seguida, o CountVectorizer do sklearn é utilizado para transformar o texto em vetores de contagem, separadamente para os conjuntos de treinamento e teste. Por fim, o MLPClassifier do sklearn é

utilizado para criar e treinar um modelo MLP com uma camada oculta contendo 100 neurônios, utilizando os dados de treinamento. Após o treinamento, métricas de avaliação como acurácia, precisão, revocação (recall) e Escore F1 são calculadas e impressas no console, fornecendo uma avaliação detalhada do desempenho do modelo. O script Python pode ser encontrado no Apêndice I.

6.2.8 Criação do modelo utilizando LSTM

Os passos seguidos para criação do modelo utilizando o algoritmo LSTM estão ilustrados na figura 23.

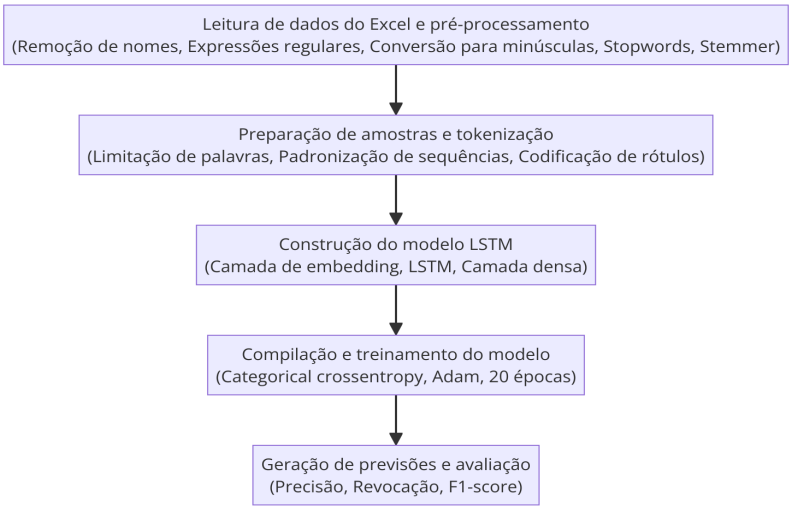


Figura 23 - Etapas para criação do modelo com LSTM.

Fonte: O próprio autor.

O processo de utilização de uma rede neural recorrente LSTM (Long Short-Term Memory) por meio de um script segue os seguintes passos. Inicialmente, os dados são lidos de um arquivo Excel, seguido pela remoção dos nomes masculinos e femininos da coluna 'EHR', realizada para preservar a privacidade dos pacientes. Em seguida, expressões regulares são aplicadas para eliminar números e outros caracteres específicos das colunas 'CID' e 'EHR', enquanto a coluna 'EHR' é convertida para minúsculas e caracteres não alfanuméricos são removidos. Adicionalmente, stopwords são eliminadas da coluna 'EHR' para reduzir a interferência de palavras comuns. Um stemmer RSLP do NLTK é então aplicado à coluna 'EHR' para simplificar as palavras para suas raízes.

Após o pré-processamento, as amostras são preparadas para alimentar o modelo, mantendo amostras com 6 ou mais ocorrências e em seguida, as amostras são tokenizadas utilizando o Tokenizer do Keras, limitando a 10.000 palavras. As sequências de tokens são padronizadas para ter o mesmo comprimento máximo (16516 caracteres) para alimentação do modelo. Além disso, os rótulos de classe são codificados utilizando o LabelEncoder e convertidos para formato one-hot usando a função to_categorical, uma prática comum em problemas de classificação multiclasse.

Na construção do modelo, é criado um modelo sequencial do Keras composto por uma camada de embedding, uma camada LSTM e uma camada densa com função de ativação softmax. Métricas personalizadas de precisão, revocação (recall) e F-Score são definidas para monitorar o desempenho do modelo durante o treinamento.

Posteriormente, o modelo é compilado utilizando a função de perda 'categorical_crossentropy' e o otimizador 'adam', configurados para problemas de classificação. O modelo é então treinado utilizando os dados de treinamento com 20 épocas, significando que o conjunto de dados é percorrido 20 vezes durante o treinamento. Ao final do treinamento, as previsões são geradas e impressas no console, possivelmente para avaliar o desempenho do modelo e sua capacidade de classificação. O script Python pode ser encontrado no Apêndice J.

6.2.9 Criação do modelo utilizando LSTM com amostras sintéticas

Os passos seguidos para criação do modelo utilizando o algoritmo LSTM em conjunto com PME sintéticos estão ilustrados na figura 24.

Figura 24 - Etapas para criação do modelo com LSTM e amostras sintéticas.

Fonte: O próprio autor.

Inicialmente, os dados reais são carregados de um arquivo Excel, enquanto as amostras sintéticas são obtidas de outro arquivo Excel. Os dois conjuntos de dados são então concatenados verticalmente usando a função pd.concat. Em seguida, é realizado o pré-processamento dos dados, onde nomes masculinos e femininos são removidos da coluna 'EHR', e expressões regulares são empregadas para eliminar números e outros caracteres específicos das colunas 'CID' e 'EHR'.

A coluna 'EHR' é convertida para minúsculas e caracteres não alfanuméricos são removidos. Uma frase específica, "queixa principal: o paciente apresentou-se na unidade de saúde", é eliminada da coluna 'EHR', e stopwords são removidas para limpar o texto. Ademais, um stemmer RSLP do NLTK é aplicado à coluna 'EHR', e apenas as amostras com 6 ou mais ocorrências são mantidas para garantir a qualidade dos dados. As amostras são então tokenizadas utilizando o Tokenizer do Keras, limitando-se a 10.000 palavras, e as sequências de tokens são padronizadas para terem o mesmo comprimento máximo de 16516 (caracteres). Os rótulos de classe são codificados utilizando o LabelEncoder e convertidos para o formato one-hot usando a função to_categorical.

Em seguida, é construído um modelo sequencial do Keras composto por uma

camada de embedding, uma camada LSTM e uma camada densa com ativação softmax. O modelo é compilado utilizando a função de perda 'categorical_crossentropy' e o otimizador 'adam'. Finalmente, o modelo é treinado com os dados de treinamento durante 20 épocas, e as previsões são impressas no console ao final do processo. O script Python pode ser encontrado no Apêndice L.

6.3 Comparação de desempenho entre modelos

O foco era identificar a eficácia de diferentes algoritmos no contexto específico da cardiologia, utilizando técnicas de PLN para tratar e preparar os dados de texto. A resposta à questão de pesquisa encontrada indica que diferentes algoritmos demonstraram variados graus de eficácia, com destaque para o modelo LSTM, que apresentou excelente desempenho, principalmente quando combinado com amostras sintéticas criadas via ChatGPT. O modelo de Regressão Logística também se destacou em termos de F-Score.

A acurácia do Naive Bayes é relativamente alta, mas a precisão e o recall são baixos. Isso indica que o modelo está classificando muitos exemplos negativos como positivos. O baixo desempenho do Naive Bayes pode ser devido à sua suposição de independência entre as características. Essa suposição nem sempre é verdadeira na prática, o que pode levar a erros na classificação.

A acurácia da Árvore de Decisão é similar à do Naive Bayes, mas a precisão e o recall são melhores. Isso indica que o modelo está classificando os exemplos com mais precisão. O bom desempenho da Árvore de Decisão pode ser devido à sua capacidade de capturar regras complexas nas características. Isso permite que o modelo classifique os exemplos com mais precisão.

A acurácia do KNN é a mais alta entre os modelos comparados. A precisão e o recall também são altos, mas o Escore F1 é um pouco menor que o da Árvore de Decisão. Isso indica que o KNN está classificando os exemplos com mais precisão, mas está perdendo alguns exemplos positivos. O alto desempenho do KNN pode ser devido à sua capacidade de se adaptar a diferentes tipos de dados. Isso permite que o modelo generalize bem para novos exemplos.

A acurácia do MLP é a mais baixa entre os modelos comparados. A precisão e o recall também são baixos. Isso indica que o modelo está tendo

dificuldade em classificar os exemplos corretamente. A acurácia do LSTM é similar à do MLP, mas a precisão e o recall são melhores. O modelo LSTM destaca-se quanto à métrica de Escore F1, obtendo valor bem maior em relação aos outros modelos. Quando utilizou-se o conjunto de dados sintéticos para treinamento do modelo, aumentou-se o valor do Escore F1 de 0,671 para 0,835 (Tabela 2).

Tabela 2 - Comparação entre modelos por métricas de avaliação.

Classificador	Acurácia	Precisão	Recall	F-Score
Naive_Bayes	0,129	0,051	0,129	0,066
Regressão_Log	0,250	0,176	0,250	0,176
Árvore de Decisão	0,166	0,060	0,073	0,065
Floresta Aleatória	0,150	0,166	0,692	0,050
KNN	0,150	0,158	0,391	0,069
SVM	0,233	0,478	0,424	0,167
MLP	0,200	0,219	0,462	0,115
LSTM	0,833	0,834	0,467	0,671
LSTM + Amostras Sintéticas	0,934	0,976	0,714	0,835

Fonte: Próprio Autor

7. Conclusão

O desenvolvimento desta pesquisa foi motivado pela necessidade de gerenciar eficientemente o crescente volume de Prontuários Médicos Eletrônicos (PME) em um ambiente de saúde, como um hospital de cardiologia. A lacuna identificada reside na falta de automatização eficaz na classificação dos textos dos prontuários, o que é crucial para processar a vasta quantidade de dados gerados diariamente sem intervenção manual, melhorando assim a gestão e o acesso às informações críticas para a tomada de decisão clínica.

A pesquisa buscou desenvolver e avaliar modelos de aprendizado de máquina supervisionado que pudessem classificar automaticamente textos de prontuários eletrônicos de pacientes.

Além disso, a pesquisa destacou a importância e o potencial das tecnologias de PLN e IA generativa na criação de conjuntos de dados de treinamento robustos e na melhoria da generalização dos modelos. A capacidade de processar e analisar grandes volumes de dados pode facilitar diagnósticos mais rápidos e precisos, melhorar os resultados de saúde e personalizar tratamentos, evidenciando um avanço tecnológico importante para a gestão de saúde.

7.1 Limitações da Pesquisa

Apesar das limitações técnicas e de escopo que restringiram a quantidade de modelos avaliados para a classificação de Prontuários Médicos Eletrônicos (PME) neste estudo, foi demonstrada com sucesso a capacidade de utilizar modelos de inteligência artificial para essa finalidade. A seleção dos modelos utilizados foi cuidadosamente orientada pela proposta do trabalho, a qual visa investigar e avaliar técnicas e recursos que possam ser aplicados em um contexto real de classificação automatizada de PME no futuro.

Dessa forma, mesmo com as restrições impostas, os resultados obtidos fornecem uma base sólida para a continuidade dessa linha de pesquisa, destacando o potencial das abordagens de inteligência artificial na interpretação e análise de textos complexos como os encontrados nos prontuários médicos eletrônicos.

7.2 Produções da Pesquisa

Destacam-se como produções resultantes da pesquisa realizada:

- Apresentação de pôster no II Encontro Anual do CI-IA Saúde, 2023, Belo Horizonte (CONSTANTIN, G.; CAZELLA, S. C. “Aprendizado supervisionado de máquina para classificação automática de Prontuários Médicos da Cardiologia” Anais do I Simpósio CI-IA Saúde da UFMG. Belo Horizonte: UFMG, 2023. v. 1. p. 16-17. http://dx.doi.org/10.5281/zenodo.8403347) (Anexo III).
- Apresentação do pôster no 3º Congresso da Universidade Federal de Ciências da Saúde de Porto Alegre (CONSTANTIN, G.; CAZELLA, S. C. “Aprendizado De Máquina Supervisionado Para Classificação Automática De Prontuários Médicos Da Cardiologia". Congresso da Universidade Federal de Ciências da Saúde de Porto Alegre, 3, 2023, Porto Alegre, Pôster) (Anexo IV).
- Curso de curta duração na categoria de Pós-Graduação e Pesquisa na modalidade EAD intitulado "Conceitos básicos de ciência de dados aplicados à saúde utilizando a plataforma KNIME". (https://moodle.ufcspa.edu.br/course/view.php?id=2570).
- Base de Dados contendo PME sintéticos criados com uso de ChatGPT (CONSTANTIN, G.; CAZELLA, S. C. Amostras Sintéticas de Prontuários Eletrônicos em Cardiologia [Dataset] (https://doi.org/10.5281/zenodo.10472522).
- Submissão do artigo científico intitulado “Utilizando Large Language Models para sintetizar dados de Prontuários Eletrônicos” ao Simpósio brasileiro em Informática em Saúde (SBIS/2024).
- Escrita do artigo científico intitulado “Aprendizado de Máquina Supervisionado para Classificação Automática de Prontuários Médicos da Cardiologia” para submissão ao periódico Revista Cadernos De Saúde Pública[10], e encontra-se no Apêndice M.

O estudo segue os princípios FAIR (Findable, Accessible, Interoperable, and Reusable), garantindo a disponibilidade e acessibilidade dos dados utilizados por meio do repositório Zenodo. Dessa forma, os dados subjacentes ao estudo estão acessíveis para consulta e verificação por outros pesquisadores interessados. Zenodo é uma plataforma criada para compartilhar e preservar resultados de pesquisa de forma aberta e acessível. Criada pela Organização Europeia para a Pesquisa Nuclear (CERN) e pelo projeto OpenAIRE, ela oferece um conjunto de recursos valiosos para pesquisadores de todas as áreas. O conjunto de dados está

[10] https://cadernos.ensp.fiocruz.br/ojs/index.php/csp

hospedado no repositório Zenodo (https://zenodo.org/records/10472) (Anexo IV).

Além disso, os códigos desenvolvidos durante a análise e processamento dos dados estão disponíveis no GitHub[11], promovendo a transparência e possibilitando a replicação dos resultados obtidos. Essa abordagem alinhada aos princípios FAIR contribui para a colaboração e o avanço da pesquisa científica na área de informática em saúde.

7.3 Trabalhos Futuros

Quanto a trabalhos futuros que advém dessa pesquisa é possível destacar , a exploração de bases de dados mais robustas em quantidade de conteúdo, bem como a aplicação de modelos de inteligência artificial que demandam maior capacidade de processamento, tais como redes neurais profundas e outros algoritmos de aprendizado profundo.

A utilização dessas bases de dados maiores permitirá uma análise mais abrangente e detalhada, possibilitando a identificação de padrões e correlações mais sutis nos dados de saúde.

[11] https://github.com/gcbiotec/Mestrado_UFCSPA

8. REFERÊNCIAS BIBLIOGRÁFICAS

AGGARWAL, C. (2018). Neural Networks and Deep Learning. ed.1. Springer Cham.

AHN I., GWON H., KANG H., KIM Y., SEO H., CHOI H., CHO H., KIM M., JUN T., KIM Y. (2021). Machine Learning-Based Hospital Discharge Prediction for Patients With Cardiovascular Diseases: Development and Usability Study. JMIR medical informatics, 9(11), e32662. https://doi.org/10.2196/32662

ALBERTO, K. G., ABELLA, C. M., SICAT, M. G. C. E., NIGUIDULA, J. D., & CABALLERO, J. M. (2009). Compiling Remote Files: Redefining Electronic Document Management System Infrastructure (CReED). In 2009 International Conference on Information and Multimedia Technology (pp. 347-350). IEEE.

ALOTAIBI, F. M. (2019). Classifying text-based emotions using logistic regression.

ALSMADI M., OMAR K., NOAH S., ALMARASHDAH I (2009). “Performance comparison of multi-layer perceptron (back propagation, delta rule and perceptron) algorithms in neural networks,” in 2009 IEEE International Advance Computing Conference, pp. 296–299.

ARUMUGAM, R.; SHANMUGAMANI, R. (2018). Hands-On Natural Language Processing with Python: A practical guide to applying deep learning architectures to your NLP applications. 1ºed. Reino Unido.

ASHFAQ A, SANT'ANNA A, LINGMAN M, NOWACZYK S. Readmission prediction using deep learning on electronic health records. J Biomed Inform. 2019;97:103256. doi:10.1016/j.jbi.2019.103256

BENKE, K., BENKE, G. (2018). Artificial Intelligence and Big Data in Public Health. International journal of environmental research and public health, 15(12), 2796. https://doi.org/10.3390/ijerph15122796

BHANOT, N., SINGH, H., SHARMA, D., JAIN, H., JAIN, S. (2019). Python vs. R: A Text Mining Approach for analyzing the Research Trends in Scopus Database.

BIAN, S., OUYANG, X., FAN, Z., & KOUTRIS, P. (2023). Certifiable Robustness for Naive Bayes Classifiers. *arXiv preprint arXiv:2303.04811*.

BROWN, T., MANN, B., RYDER, N., SUBBIAH, M., KAPLAN, J. D., DHARIWAL, P., & AMODEI, D. (2020). Language models are few-shot learners. Advances in neural information processing systems, 33, 1877-1901.

XUE, B., ZHANG, X., XU, Y., LI, Y., ZHANG, H., & YU, C. (2021). Sparse signal recovery based on adaptive algorithms for debris detector. AIP Advances, 11(6), 065131. https://doi.org/10.1063/5.0050715

BIJALWAN, V., KUMAR, V., KUMARI, P., & PASCUAL, J. (2014). KNN based machine learning approach for text and document mining. International Journal of Database Theory and Application, 7(1), 61-70.

BIRD, S.; KLEIN, E.; LOPER, E. (2009). Natural Language Processing with Python, 1ºed. O'Reilly Media. Estados Unidos da América.

BREIMAN, L. (2001). Random forests. *Machine learning*, *45*, 5-32.

BREIMAN, L.; FRIEDMAN, J.; OLSHEN, R.; STONE, C. Classification and Regression Trees. Belmont, CA: Wadsworth, 1984.

CAWSEY A. J., WEBBER B.L., JONES R.B. Natural language generation in health care. J Am Med Inform Assoc. 1997;4(6):473-482. doi:10.1136/jamia.1997.0040473

CHE, C., HU, H., ZHAO, X., LI, S., & LIN, Q. (2023). Advancing Cancer Document Classification with R andom Forest. Academic Journal of Science and Technology, 8(1), 278-280.

CHEN P, DONG W, LU X, KAYMAK U, HE K, HUANG Z. Deep representation learning for individualized treatment effect estimation using electronic health records. J Biomed Inform. 2019;100:103303. doi:10.1016/j.jbi.2019.103303

CHEN P, DONG W, WANG J, LU X, KAYMAK U, HUANG Z. Interpretable clinical prediction via attention-based neural network. BMC Med Inform Decis Mak. 2020;20(Suppl 3):131. Published 2020 Jul 9. doi:10.1186/s12911-020-1110-7

CHU J, DONG W, WANG J, HE K, HUANG Z. Treatment effect prediction with adversarial deep learning using electronic health records. BMC Med Inform Decis Mak. 2020;20(Suppl 4):139. Published 2020 Dec 14. doi:10.1186/s12911-020-01151-9

CLARKE, L.A. A system to generate test data and symbolically execute programs. IEEE Trans Softw Eng. 1976; SE-2(3):215-222. [author translation: Clarke, LA. A system to generate test data and symbolically execute programs. IEEE Trans Softw Eng. 1976; SE-2(3):215-222.]

CONSELHO FEDERAL DE ENFERMAGEM. Guia de Recomendações Para Registro de Enfermagem no Prontuário do Paciente e Outros Documentos de Enfermagem. Aprovado pela Resolução COFEN nº 0514/2016

CONSELHO FEDERAL DE MEDICINA. Código de Ética Médica. Resolução CFM n° 1246/88. Rio de Janeiro, Idéia & Produções, 1988.

CONSELHO FEDERAL DE MEDICINA. Define prontuário médico e torna obrigatória a criação da Comissão de Revisão de Prontuários nas instituições de Saúde. Resolução CFM 1.638, de 10 de julho de 2002. Brasília: CFM, 2002.

CONSELHO FEDERAL DE MEDICINA. Resolução CFM 1605, publicada no D.O.U. 29 SET 2000, Seção I, pg. 30

CONSELHO FEDERAL DE MEDICINA. Resolução CFM 1821, publicada no D.O.U. de 23 nov. 2007, Seção I, pg. 252

CORTES, C., VAPNIK, V. (1995). Support-vector networks. *Machine learning*, *20*, 273-297.

DEVLIN, J., CHANG, M. W., LEE, K., & TOUTANOVA, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

EIGENSCHINK, P., REUTTERER, T., VAMOSI, S., VAMOSI, R., SUN, C., & KALCHER, K. (2023). Deep Generative Models for Synthetic Sequential Data: A Survey. IEEE Access.

ERTEL, W. Introduction to Artificial Intelligence. 2º ed. Alemanha, Springer, 2017

GARDNER, M. W., & DORLING, S. R. (1998). Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment*, *32*(14-15), 2627-2636.

GERS, F. A., SCHMIDHUBER, J., & CUMMINS, F. (2000). Learning to forget: Continual prediction with LSTM. *Neural computation*, *12*(10), 2451-2471.

GUPTA V., MISHRA V. K., SINGHAL P., KUMAR A. "An Overview of Supervised Machine Learning Algorithm," 2022 11th International Conference on System.

HACOHEN, Y., MILLER, D., YIGAL, Y. (2020). The influence of preprocessing on text classification using a bag-of-words representation. PloS one, 15(5), e0232525

HARRISON, M. Machine Learning: Guia de Referência Rápida. São Paulo. Novatec Editora, 2020.

HASSANI, H. SILVA, E.S. The Role of ChatGPT in Data Science: How AI-Assisted Conversational Interfaces Are Revolutionizing the Field. Big Data Cogn Comput. 2023; v. 7, n. 2, p. 62. 10.3390/bdcc7020062.

HASTIE, T., TIBSHIRANI, R., FRIEDMAN, J., HASTIE, T., TIBSHIRANI, R., & FRIEDMAN, J. (2009). Random forests. *The elements of statistical learning: Data mining, inference, and prediction*, 587-604.

HOCHREITER, S., & SCHMIDHUBER, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735-1780.

HOLMES, J. H., BEINLICH, J., BOLAND, M. R., BOWLES, K. H., CHEN, Y., COOK, T. S., DEMIRIS, G., DRAUGELIS, M., FLUHARTY, L., GABRIEL, P. E., GRUNDMEIER, R., HANSON, C. W., HERMAN, D. S., HIMES, B. E., HUBBARD, R. A., KAHN, C. E., JR, KIM, D., KOPPEL, R., LONG, Q., MIRKOVIC, N., MOORE, J. H. (2021). Why is the Electronic HEALTH RECORD SO CHALLENGING FOR RESEARCH AND CLINICAL CARE? *METHODS OF INFORMATION IN MEDICINE*, *60*(1-02), 32−48. https://doi.org/10.1055/s-0041-1731784.

IKONOMAKIS M., KOTSIANTIS S., TAMPAKAS V. Text Classification Using Machine Learning Techniques. WSEAS TRANSACTIONS on COMPUTERS, Issue 8, Volume 4, August 2005, pp. 966-974

JOACHIMS, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In European Conference on Machine Learning (ECML).

KHYANI, D., SIDDHARTHA B. S. (2021). An Interpretation of Lemmatization and Stemming in Natural Language Processing. Shanghai Ligong Daxue Xuebao/Journal of University of Shanghai for Science and Technology. 22. 350-357.

KIM, Y. (2014) Convolutional Neural Networks for Sentence Classification. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMPLN), Doha, 1746-1751

KOTSIANTIS, S. B. (2013). Decision trees: a recent overview. *Artificial Intelligence Review*, *39*, 261-283.

KONNYU K., KWOK E., SKIDMORE B., MOHER D. The effectiveness and safety of emergency department short stay units: a rapid review. Open Med. 2012;6:e10–e16.

KOSSMAN, S.P.; BONNEY, L.A.; KIM, M.J.; Electronic Health Record Tools' Support of Nurses' Clinical Judgment and Team Communication. Computers, Informatics & Nursing. 2013 Vol. 31, No. 11, 539–544.

KRITTANAWONG C, VIRK H., KUMAR A, AYDAR M., W ZHEN, P. MATTHEW, L. JONATHAN. Machine learning and deep learning to predict mortality in patients with spontaneous coronary artery dissection. Sci Rep. 2021;11(1):8992. Published 2021 Apr 26. doi:10.1038/s41598-021-88172-0

LU, Y., SHEN, M., WANG, H., WANG, X., VAN RECHEM, C., & WEI, W. (2023). Machine learning for synthetic data generation: a review. arXiv preprint arXiv:2302.04062.

LUBIS, D. H.; SAWALUDDIN, S.; CANDRA, A. Machine Learning Model for Language Classification: Bag-of-words and Multilayer Perceptron. JOURNAL OF INFORMATICS AND TELECOMMUNICATION ENGINEERING, v. 7, n. 1, p. 356-365, 2023.

MADAHIAN, B., FAGHIHI, U. (2014) A Fully Bayesian Sparse Probit Model for Text Categorization. Open Journal of Statistics, 4, 611-619

MCENERY, A. AND XIAO, R. (2005) Character encoding in corpus construction. In M. Wynne (ed.), Developing Linguistic Corpora: A Guide to Good Practice, pp. 47–58. Oxford, U.K.: AHDS.

MODELING & ADVANCEMENT IN RESEARCH TRENDS (SMART), Moradabad, India, 2022, pp. 87-92, doi: 10.1109/SMART55829.2022.10047618.

MULLER, A.; GUIDO, S. Introduction to Machine Learning with Python: A Guide for Data Scientists. 1º ed. O'Reilly. EUA, 2017.

PAKHOMOV S, HEMINGWAY H, WESTON S, JACOBSEN J, RODEHEFFER R, ROGER V. Epidemiology of angina pectoris: role of natural language processing of the medical record. Am Heart J. 2007 Apr;153(4):666-73. doi: 10.1016/j.ahj.2006.12.022. PMID: 17383310; PMCID: PMC1929015.

PATRICK, E. A., & FISCHER III, F. P. (1970). A generalized k-nearest neighbor rule. *Information and control*, *16*(2), 128-152.

PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., MICHEL, V., THIRION, B., GRISEL, O., DUCHESNAY, E. (2011). Scikit-learn: Machine learning in Python. the Journal of machine Learning research, 12, 2825-2830.

PISSAIA, LUÍS FELIPE, HEPP REHFELDT, M. J., KUNZ DA COSTA, A. E., MORESCHI, C. Tasy e o ensino em saúde. Revista Sustinere, [S.l.], v. 9, n. 2, p. 629-640, jul. 2022. ISSN 2359-0424. Disponível em: <https://www.e-publicacoes.uerj.br/index.php/sustinere/article/view/51556/41085>. Acesso em: 04 dez. 2022. doi:https://doi.org/10.12957/sustinere.2021.51556.

POPESCU, M. C., BALAS, V. E., PERESCU-POPESCU, L., & MASTORAKIS, N. (2009). Multilayer perceptron and neural networks. *WSEAS Transactions on Circuits and Systems*, *8*(7), 579-588.

POSSARI, J. Prontuário do paciente e os registros de enfermagem. São Paulo: Iátria; 2005.

PRABHAKAR, S. K., & WON, D. O. (2021). Medical text classification using hybrid deep learning models with multihead attention. Computational intelligence and neuroscience, 2021.

PRIYAM, A., ABHIJEETA, G. R., RATHEE, A., & SRIVASTAVA, S. (2013). Comparative analysis of decision tree classification algorithms. *International Journal of current engineering and technology*, *3*(2), 334-337.

PRIYANKA, & KUMAR, D. (2020). Decision tree classifier: a detailed survey.

International Journal of Information and Decision Sciences, *12*(3), 246-269.

QIAN L., HAO P., JIANXIN L., CONGYIN X., RENYU Y., LICHAO S., PHILIP S. YU, LIFANG H. 2020. A Survey on Text Classification: From Shallow to Deep Learning. ACM Comput. Surv. 37, 4, Article 35 (July 2020).

RAHMAN, R., KODESH, A., LEVINE, S. Z., SANDIN, S., REICHENBERG, A., & SCHLESSINGER, A. (2020). Identification of newborns at risk for autism using electronic medical records and machine learning. European psychiatry : the journal of the Association of European Psychiatrists, 63(1), e22. https://doi.org/10.1192/j.eurpsy.2020.17

RANDLES, B., PASQUETTO, I., GOLSHAN M., BORGMAN C. (2017) "Using the Jupyter Notebook as a Tool for Open Science: An Empirical Study," ACM/IEEE Joint Conference on Digital Libraries (JCDL), 2017, pp. 1-2

RIGATTI, S. J. (2017). Random forest. *Journal of Insurance Medicine*, *47*(1), 31-39.

SCARSELLI, FRANCO; GORI, MARCO; TSOI, AH CHUNG; HAGENBUCHNER, MARKUS; MONFARDINI, GABRIELE (2009). "The Graph Neural Network Model". IEEE Transactions on Neural Networks. 20 (1): 61–80. doi:10.1109/TNN.2008.2005605

SCHÜNKE, L. C., MELLO, B., DA COSTA, C. A., ANTUNES, R. S., RIGO, S. J., DE OLIVEIRA RAMOS, G., DONIDA, B. (2022). A rapid review of machine learning approaches for telemedicine in the scope of COVID-19. Artificial Intelligence in Medicine, 129, 102312.

SEGAL, M., & XIAO, Y. (2011). Multivariate random forests. *Wiley interdisciplinary reviews: Data mining and knowledge discovery*, *1*(1), 80-87

SETSUKO TOMA, T.; SOARES, . A. C. Uma revisão rápida sobre revisões rápidas. BIS. Boletim do Instituto de Saúde, São Paulo, v. 17, n. 1, p. 142–151, 2016. DOI:

10.52753/bis.2016.v17.35388. Disponível em: https://periodicos.saude.sp.gov.br/bis/article/view/35388. Acesso em: 4 dez. 2022.

SIMON B., ILONA S., YUFAN G., IMRAN A., JOHAN H., ULLA S., ANNA K. Automatic semantic classification of scientific literature according to the hallmarks of cancer. Bioinformatics, 32(3):432–440. 2016

SHERSTINSKY, A. (2020). Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *Physica D: Nonlinear Phenomena*, *404*, 132306.

SHICKEL B, TIGHE P, BIHORAC A, RASHIDI P. (2018). Deep EHR: a survey of recent advances in deep learning techniques for electronic health record (EHR) analysis. IEEE J Biomed Health Inform; 22(5):1589–604.

SHUANG Y., XIONGFEI L., XIAOLI Z., HANCHENG W. (2019). "The OCS-SVM: An Objective-Cost-Sensitive SVM With Sample-Based Misclassification Cost Invariance," in IEEE Access, vol. 7, pp. 118931-118942, doi: 10.1109/ACCESS.2019.2933437.

SINGRU, R., BHANDARI, P., PATEL, K., MANE, P., & GULHANE, C. (2020). Efficient Electronic Document Access Control Management using Natural Language Processing. In 2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC) (pp. 714-719). IEEE.

SOKOLOVA, M.; JAPKOWICZ, N.; SZPAKOWICZ, S. Beyond accuracy, f-score and roc: A family of discriminant measures for performance evaluation. In: SATTAR, A.; KANG, B.-h. (Ed.). AI 2006: Advances in Artificial Intelligence. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. p. 1015–1021. ISBN 978-3-540-49788-2.

TENSORFLOW. TensorFlow Tutorial. 2023. Disponível em: https://www.tensorflow.org/text/tutorials/classify_text_with_bert?hl=pt-br. Acesso em 07 jan. 2024.

OFFICE OF THE NATIONAL COORDINATOR FOR HEALTH INFORMATION TECHNOLOGY. (2020). Strategy on Reducing Regulatory and Administrative Burden Relating to the Use of Health IT and EHRs Final Report.

VAPNIK V. Statistical Learning Theory, John Wiley & Sons, New York, NY, USA, 1998.

VASWANI, A., SHAZEER, N., PARMAR, N., USZKOREIT, J., JONES, L., GOMEZ, A. N., POLOSUKHIN, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.

WATT A., CAMERON A., STURM L., LATHLEAN T. Rapid reviews versus full systematic reviews: an inventory of current methods and practice in health technology assessment. Int J Technol Assess Health Care. 2008;24:133–139. doi: 10.1017/S0266462308080185.

WEBSTER, JONATHAN & KIT, CHUNYU. (1992). Tokenization as the initial phase in NLP. 1106-1110. 10.3115/992424.992434.

WEINTRAUB, WILLIAM S. “Role of Big Data in Cardiovascular Research.” Journal of the American Heart Association vol. 8,14 (2019): e012791. doi:10.1161/JAHA.119.012791

WENG W., WAGHOLIKAR K., MCCRAY A., SZOLOVITS P., CHUEH H. Medical subdomain classification of clinical notes using a machine learning-based natural language processing approach. BMC Med Inform Decis Mak. 2017;17(1):155. Published 2017 Dec 1. doi:10.1186/s12911-017-0556-8

WU, LINGFEI; CUI, PENG; PEI, JIAN; ZHAO, LIANG (2022). "Graph Neural Networks: Foundations, Frontiers, and Applications". Springer Singapore: 725.

XIAO, R. (2008) Well-known and influential corpora. In A. Lüdeling and M. Kyto (eds.), Corpus Linguistics: An International Handbook (Vol. 1), pp. 383–457. Berlin, Germany: Mouton de Gruyter.

XING, W., & BEI, Y. (2019). Medical health big data classification based on KNN classification algorithm. IEEE Access, 8, 28808-28819.

YANG, Y., ZHOU, X., GAO, S., LIN, H., XIE, Y., FENG, Y., HUANG, K., ZHAN, S. (2018). Evaluation of Electronic Healthcare Databases for Post-Marketing Drug Safety Surveillance and Pharmacoepidemiology in China. Drug safety, 41(1), 125–137. https://doi.org/10.1007/s40264-017-0589-z.

YONG, Z., YOUWEN, L., & SHIXIONG, X. (2009). An improved KNN text classification algorithm based on clustering. Journal of computers, 4(3), 230-237

ZHAO J., FENG Q., WU P., LUPU R., WILKE R., WELLS Q. , DENNY J., WEI W. Learning from Longitudinal Data in Electronic Health Record and Genetic Data to Improve Cardiovascular Event Prediction. Scientific reports, 9(1), 717. (2019) https://doi.org/10.1038/s41598-018-36745-x

APÊNDICE A

Anonimização dos Prontuários Médicos Eletrônicos

A seguir é apresentado o código em Python para a realização da Anomização dos PME:

```
# Retirando nomes mais comuns da língua Portuguesa:

def remove_names_from_dataframe(csv_file_path, dataframe, column_name):
    try:
            with open(csv_file_path, 'r') as file:
            names_to_remove = [line.strip() for line in file]

        if column_name not in dataframe.columns:
            raise ValueError(f"Column '{column_name}' not found in the DataFrame.")

        dataframe = dataframe[~dataframe[column_name].isin(names_to_remove)]

        return dataframe
    except Exception as e:
        print(f"An error occurred: {str(e)}")
        return None
```

Código 1 - Anonimização dos PME

APÊNDICE B

Limpeza das amostras de Prontuários Médicos Eletrônicos

A seguir é apresentado o código em Python para a retirada de Stopwords e elementos gráficos de pontuação:

```
import nltk
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

amostras_excel = r"Planilha_EHR.xlsx"
amostras_df = pd.read_excel(amostras_excel)

'''Retirando os números e parenteses de CID'''
pattern = r'\(\d+\)'
pattern_evol = r"[,;.:#-+/)(\d+>]"
amostras_df['CID'] = amostras_df['CID'].str.replace(pattern, '')
amostras_df['EHR'] = amostras_df['EHR'].str.replace(pattern_evol, '')

stopWordNLTK = nltk.corpus.stopwords.words('portuguese')
def remove_stop_words(df):
    evolucao = df['EHR']
    cid = df['CID']

    if pd.isna(evolucao) or evolucao.strip() == '':
        return pd.Series([None, cid], index=['EHR', 'CID'])

    semStopWord = ' '.join([p for p in evolucao.split() if p not in stopWordNLTK])
    return pd.Series([semStopWord, cid], index=['EHR', 'CID'])
```

Código 2 - Retirada de Stopwords dos PME

APÊNDICE C

Script utilizando Naive Bayes

A seguir é apresentado o código Python com o modelo classificador utilizando o algoritmo Naive Bayes:

```
import pandas as pd
import nltk
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

amostras_excel = r"Planilha_EHR.xlsx"
amostras_df = pd.read_excel(amostras_excel)

'''Retirando os números e parenteses de CID'''
pattern = r'\(\d+\)'
pattern_evol = r"[,;.:#-+/)(\d+>]"
amostras_df['CID'] = amostras_df['CID'].str.replace(pattern, '')
amostras_df['EHR'] = amostras_df['EHR'].str.replace(pattern_evol, '')

print('Classes direrentes')
print(amostras_df['CID'].value_counts())

# Retirando nomes mais comuns da língua Portuguesa:

def remove_names_from_dataframe(csv_file_path, dataframe, column_name):
    try:
        # Read the CSV file into a list of names
        with open(csv_file_path, 'r') as file:
            names_to_remove = [line.strip() for line in file]

        # Check if the specified column exists in the DataFrame
        if column_name not in dataframe.columns:
            raise ValueError(f"Column '{column_name}' not found in the DataFrame.")

        # Remove rows containing names from the DataFrame
        dataframe = dataframe[~dataframe[column_name].isin(names_to_remove)]

        return dataframe
    except Exception as e:
        print(f"An error occurred: {str(e)}")
        return None

# Carregamento das bases de nomes comuns, em formato .csv:
nomes_masc = r'C:\Users\gabriel.silva\Downloads\ibge-mas-10000.csv'
nomes_fem = r'C:\Users\gabriel.silva\Downloads\ibge-fem-10000.csv'

# Retirada de nomes Masculinos:
amostras_df = remove_names_from_dataframe(nomes_masc,amostras_df,'EHR')

# Retirando os nomes Femininos:
amostras_df = remove_names_from_dataframe(nomes_fem,amostras_df,'EHR')

stopWordNLTK = nltk.corpus.stopwords.words('portuguese')
def remove_stop_words(df):
    evolucao = df['EHR']
    cid = df['CID']
```

```
    if pd.isna(evolucao) or evolucao.strip() == '':
        return pd.Series([None, cid], index=['EHR', 'CID'])

    semStopWord = ' '.join([p for p in evolucao.split() if p not in stopWordNLTK])
    return pd.Series([semStopWord, cid], index=['EHR', 'CID'])

amostrasSemStopWords = amostras_df.apply(remove_stop_words, axis=1)

def aplicaStemmer(df):
    stemmer = nltk.stem.RSLPStemmer()
    evolucao = df['EHR']
    cid = df['CID']

    evolComStem = ' '.join([str(stemmer.stem(p)) for p in evolucao.split()])
    return pd.Series([evolComStem, cid], index=['EHR', 'CID'])

amostrasComStemmer = amostrasSemStopWords.apply(aplicaStemmer, axis=1)
# print(amostrasComStemmer)

def aplicaTokenizer(df):
    tokenizer = nltk.word_tokenize
    evolucao = df['EHR']
    cid = df['CID']

evolComTok = ' '.join([str(tokenizer(p,language='portuguese')) for p in evolucao.split()])
    return pd.Series([evolComTok, cid], index=['EHR', 'CID'])

amostrasComTok = amostrasComStemmer.apply(aplicaTokenizer, axis=1)
#print(amostrasComTok)

# Divisão das amostras:
X = amostrasComTok['EHR']
y = amostrasComTok['CID']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.35, random_state=42)

vectorizer = CountVectorizer()
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)

naive_bayes = MultinomialNB()
naive_bayes.fit(X_train, y_train)

y_pred = naive_bayes.predict(X_test)

print("Avaliando o modelo:")
from sklearn.metrics import precision_score, recall_score, f1_score
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')
accuracy = accuracy_score(y_test, y_pred)
confusion = confusion_matrix(y_test, y_pred)
report = classification_report(y_test, y_pred)

print(f'Acurácia: {accuracy}')
print(f'Precisão (Precision): {precision}')
print(f'Revocação (Recall): {recall}')
print(f'F-Score: {f1}')
```

Código 3 - Modelo utilizando o algoritmo Naive Bayes.

APÊNDICE D

Script com utilização de Regressão Logística

A seguir é apresentado o código em Python com o modelo classificador utilizando o algoritmo de Regressão Logística:

```
import pandas as pd
import nltk
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

amostras_excel = r"Planilha_EHR.xlsx"
amostras_df = pd.read_excel(amostras_excel)

'''Retirando os números e parenteses de CID'''
pattern = r'\(\d+\)'
pattern_evol = r"[,;.:#-+/)(\d+>]"
amostras_df['CID'] = amostras_df['CID'].str.replace(pattern, '')
amostras_df['EHR'] = amostras_df['EHR'].str.replace(pattern_evol, '')

# Retirando nomes mais comuns da língua Portuguesa:

def remove_names_from_dataframe(csv_file_path, dataframe, column_name):
    try:
        # Read the CSV file into a list of names
        with open(csv_file_path, 'r') as file:
            names_to_remove = [line.strip() for line in file]

        # Check if the specified column exists in the DataFrame
        if column_name not in dataframe.columns:
            raise ValueError(f"Column '{column_name}' not found in the DataFrame.")

        # Remove rows containing names from the DataFrame
        dataframe = dataframe[~dataframe[column_name].isin(names_to_remove)]

        return dataframe
    except Exception as e:
        print(f"An error occurred: {str(e)}")
        return None

# Carregamento das bases de nomes comuns, em formato .csv:
nomes_masc = r'C:\Users\gabriel.silva\Downloads\ibge-mas-10000.csv'
nomes_fem = r'C:\Users\gabriel.silva\Downloads\ibge-fem-10000.csv'

# Retirada de nomes Masculinos:
amostras_df = remove_names_from_dataframe(nomes_masc,amostras_df,'EHR')

# Retirando os nomes Femininos:
amostras_df = remove_names_from_dataframe(nomes_fem,amostras_df,'EHR')

stopWordNLTK = nltk.corpus.stopwords.words('portuguese')
def remove_stop_words(df):
    evolucao = df['EHR']
    cid = df['CID']

    if pd.isna(evolucao) or evolucao.strip() == '':
```

```
        return pd.Series([None, cid], index=['EHR', 'CID'])

    semStopWord = ' '.join([p for p in evolucao.split() if p not in stopWordNLTK])
    return pd.Series([semStopWord, cid], index=['EHR', 'CID'])

amostrasSemStopWords = amostras_df.apply(remove_stop_words, axis=1)

def aplicaStemmer(df):
    stemmer = nltk.stem.RSLPStemmer()
    evolucao = df['EHR']
    cid = df['CID']

    evolComStem = ' '.join([str(stemmer.stem(p)) for p in evolucao.split()])
    return pd.Series([evolComStem, cid], index=['EHR', 'CID'])

amostrasComStemmer = amostrasSemStopWords.apply(aplicaStemmer, axis=1)
# print(amostrasComStemmer)

def aplicaTokenizer(df):
    tokenizer = nltk.word_tokenize
    evolucao = df['EHR']
    cid = df['CID']

    evolComTok = ' '.join([str(tokenizer(p,language='portuguese')) for p in evolucao.split()])
    return pd.Series([evolComTok, cid], index=['EHR', 'CID'])

amostrasComTok = amostrasComStemmer.apply(aplicaTokenizer, axis=1)
#print(amostrasComTok)

# Divisão das amostras:
X = amostrasComTok['EHR']
y = amostrasComTok['CID']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=42)

vectorizer = CountVectorizer()
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)

logistic_regression = LogisticRegression(max_iter=1000)  # Aumentamos max_iter para garantir
convergência.
logistic_regression.fit(X_train, y_train)

y_pred = logistic_regression.predict(X_test)

precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')
accuracy = accuracy_score(y_test, y_pred)

print(f'Acurácia: {accuracy}')
print(f'Precisão (Precision): {precision}')
print(f'Revocação (Recall): {recall}')
print(f'F-Score: {f1}')
```

Código 4 - Modelo utilizando o algoritmo de Regressão Logística.

APÊNDICE E

Script com utilização de Árvore de Decisão

A seguir é apresentado o código Python mostrando o código do modelo classificador utilizando o algoritmo Árvores de Decisão:

```
import nltk
import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

amostras_excel = r"Planilha_EHR.xlsx"
amostras_df = pd.read_excel(amostras_excel)

'''Retirando os números e parenteses de CID'''
pattern = r'\(\d+\)'
pattern_evol = r"[,;.:#-+/)(\d+>]"
amostras_df['CID'] = amostras_df['CID'].str.replace(pattern, '')
amostras_df['EHR'] = amostras_df['EHR'].str.replace(pattern_evol, '')

# Retirando nomes mais comuns da língua Portuguesa:

def remove_names_from_dataframe(csv_file_path, dataframe, column_name):
    try:
        # Read the CSV file into a list of names
        with open(csv_file_path, 'r') as file:
            names_to_remove = [line.strip() for line in file]

        # Check if the specified column exists in the DataFrame
        if column_name not in dataframe.columns:
            raise ValueError(f"Column '{column_name}' not found in the DataFrame.")

        # Remove rows containing names from the DataFrame
        dataframe = dataframe[~dataframe[column_name].isin(names_to_remove)]

        return dataframe
    except Exception as e:
        print(f"An error occurred: {str(e)}")
        return None

# Carregamento das bases de nomes comuns, em formato .csv:
nomes_masc = r'C:\Users\gabriel.silva\Downloads\ibge-mas-10000.csv'
nomes_fem = r'C:\Users\gabriel.silva\Downloads\ibge-fem-10000.csv'

# Retirada de nomes Masculinos:
amostras_df = remove_names_from_dataframe(nomes_masc,amostras_df,'EHR')

# Retirando os nomes Femininos:
amostras_df = remove_names_from_dataframe(nomes_fem,amostras_df,'EHR')

stopWordNLTK = nltk.corpus.stopwords.words('portuguese')
def remove_stop_words(df):
    evolucao = df['EHR']
    cid = df['CID']
```

```
    if pd.isna(evolucao) or evolucao.strip() == '':
        return pd.Series([None, cid], index=['EHR', 'CID'])

    semStopWord = ' '.join([p for p in evolucao.split() if p not in stopWordNLTK])
    return pd.Series([semStopWord, cid], index=['EHR', 'CID'])

amostrasSemStopWords = amostras_df.apply(remove_stop_words, axis=1)

def aplicaStemmer(df):
    stemmer = nltk.stem.RSLPStemmer()
    evolucao = df['EHR']
    cid = df['CID']

    evolComStem = ' '.join([str(stemmer.stem(p)) for p in evolucao.split()])
    return pd.Series([evolComStem, cid], index=['EHR', 'CID'])

amostrasComStemmer = amostrasSemStopWords.apply(aplicaStemmer, axis=1)
# print(amostrasComStemmer)

def aplicaTokenizer(df):
    tokenizer = nltk.word_tokenize
    evolucao = df['EHR']
    cid = df['CID']

    evolComTok = ' '.join([str(tokenizer(p,language='portuguese')) for p in evolucao.split()])
    return pd.Series([evolComTok, cid], index=['EHR', 'CID'])

amostrasComTok = amostrasComStemmer.apply(aplicaTokenizer, axis=1)
#print(amostrasComTok)

# Divisão das amostras:
X = amostrasComTok['EHR']
y = amostrasComTok['CID']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=42)

vectorizer = CountVectorizer()
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)

decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, y_train)

y_pred = decision_tree.predict(X_test)

precision = precision_score(y_test, y_pred, average='macro')
recall = recall_score(y_test, y_pred, average='macro')
f1 = f1_score(y_test, y_pred, average='macro')
accuracy = accuracy_score(y_test, y_pred)

print(f'Acurácia: {accuracy}')
print(f'Precisão (Precision): {precision}')
print(f'Revocação (Recall): {recall}')
print(f'F-Score: {f1}')
```

Código 5 - Modelo utilizando o algoritmo Árvores de Decisão.

APÊNDICE F

Script com utilização de Floresta Aleatória

A seguir é apresentado o código Python do modelo classificador utilizando Floresta Aleatória.

```
import nltk
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

amostras_excel = r"C:\Users\gabriel.silva\Documents\Mestrado\Planilha_EHR.xlsx"
amostras_df = pd.read_excel(amostras_excel)
pattern = r'\(\d+\)'
pattern_evol = r"[,;.:#-+/)(\d+>]"
amostras_df['CID'] = amostras_df['CID'].str.replace(pattern, '')
amostras_df['EHR'] = amostras_df['EHR'].str.replace(pattern_evol, '')

def remove_names_from_dataframe(csv_file_path, dataframe, column_name):
    try:
        # Read the CSV file into a list of names
        with open(csv_file_path, 'r') as file:
            names_to_remove = [line.strip() for line in file]

        # Check if the specified column exists in the DataFrame
        if column_name not in dataframe.columns:
            raise ValueError(f"Column '{column_name}' not found in the DataFrame.")

        # Remove rows containing names from the DataFrame
        dataframe = dataframe[~dataframe[column_name].isin(names_to_remove)]

        return dataframe
    except Exception as e:
        print(f"An error occurred: {str(e)}")
        return None

# Carregamento das bases de nomes comuns, em formato .csv:
nomes_masc = r'C:\Users\gabriel.silva\Downloads\ibge-mas-10000.csv'
nomes_fem = r'C:\Users\gabriel.silva\Downloads\ibge-fem-10000.csv'

# Retirada de nomes Masculinos:
amostras_df = remove_names_from_dataframe(nomes_masc,amostras_df,'EHR')

# Retirando os nomes Femininos:
amostras_df = remove_names_from_dataframe(nomes_fem,amostras_df,'EHR')

stopWordNLTK = nltk.corpus.stopwords.words('portuguese')
def remove_stop_words(df):
    evolucao = df['EHR']
    cid = df['CID']

    if pd.isna(evolucao) or evolucao.strip() == '':
        return pd.Series([None, cid], index=['EHR', 'CID'])
```

```
    semStopWord = ' '.join([p for p in evolucao.split() if p not in stopWordNLTK])
    return pd.Series([semStopWord, cid], index=['EHR', 'CID'])

amostrasSemStopWords = amostras_df.apply(remove_stop_words, axis=1)

def aplicaStemmer(df):
    stemmer = nltk.stem.RSLPStemmer()
    evolucao = df['EHR']
    cid = df['CID']

    evolComStem = ' '.join([str(stemmer.stem(p)) for p in evolucao.split()])
    return pd.Series([evolComStem, cid], index=['EHR', 'CID'])

amostrasComStemmer = amostrasSemStopWords.apply(aplicaStemmer, axis=1)
# print(amostrasComStemmer)

def aplicaTokenizer(df):
    tokenizer = nltk.word_tokenize
    evolucao = df['EHR']
    cid = df['CID']

    evolComTok = ' '.join([str(tokenizer(p,language='portuguese')) for p in evolucao.split()])
    return pd.Series([evolComTok, cid], index=['EHR', 'CID'])

amostrasComTok = amostrasComStemmer.apply(aplicaTokenizer, axis=1)
#print(amostrasComTok)

# Divisão das amostras:
X = amostrasComTok['EHR']
y = amostrasComTok['CID']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=42)

vectorizer = CountVectorizer()
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)

random_forest = RandomForestClassifier()
random_forest.fit(X_train, y_train)

y_pred = random_forest.predict(X_test)

precision = precision_score(y_test, y_pred, average='weighted', labels=np.unique(y_pred))
recall = recall_score(y_test, y_pred, average='weighted', labels=np.unique(y_pred))
f1 = f1_score(y_test, y_pred, average='weighted')
accuracy = accuracy_score(y_test, y_pred)

print(f'Acurácia: {accuracy}')
print(f'Precisão (Precision): {precision}')
print(f'Revocação (Recall): {recall}')
print(f'F-Score: {f1}')
```

Código 6 - Modelo de classificador utilizando o algoritmo Random Florest.

APÊNDICE G

Script com utilização de KNN

A seguir é apresentado o código Python do modelo classificador utilizando o algoritmo KNN.

```
import nltk
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

amostras_excel = r"Planilha_EHR.xlsx"
amostras_df = pd.read_excel(amostras_excel)

'''Retirando os números e parenteses de CID'''
pattern = r'\(\d+\)'
pattern_evol = r"[,;.:#-+/)(\d+>]"
amostras_df['CID'] = amostras_df['CID'].str.replace(pattern, '')
amostras_df['EHR'] = amostras_df['EHR'].str.replace(pattern_evol, '')

# Retirando nomes mais comuns da língua Portuguesa:

def remove_names_from_dataframe(csv_file_path, dataframe, column_name):
    try:
        # Read the CSV file into a list of names
        with open(csv_file_path, 'r') as file:
            names_to_remove = [line.strip() for line in file]

        # Check if the specified column exists in the DataFrame
        if column_name not in dataframe.columns:
            raise ValueError(f"Column '{column_name}' not found in the DataFrame.")

        # Remove rows containing names from the DataFrame
        dataframe = dataframe[~dataframe[column_name].isin(names_to_remove)]

        return dataframe
    except Exception as e:
        print(f"An error occurred: {str(e)}")
        return None

# Carregamento das bases de nomes comuns, em formato .csv:
nomes_masc = r'C:\Users\gabriel.silva\Downloads\ibge-mas-10000.csv'
nomes_fem = r'C:\Users\gabriel.silva\Downloads\ibge-fem-10000.csv'

# Retirada de nomes Masculinos:
amostras_df = remove_names_from_dataframe(nomes_masc,amostras_df,'EHR')

# Retirando os nomes Femininos:
amostras_df = remove_names_from_dataframe(nomes_fem,amostras_df,'EHR')

stopWordNLTK = nltk.corpus.stopwords.words('portuguese')
def remove_stop_words(df):
    evolucao = df['EHR']
    cid = df['CID']
```

```
    if pd.isna(evolucao) or evolucao.strip() == '':
        return pd.Series([None, cid], index=['EHR', 'CID'])

    semStopWord = ' '.join([p for p in evolucao.split() if p not in stopWordNLTK])
    return pd.Series([semStopWord, cid], index=['EHR', 'CID'])

amostrasSemStopWords = amostras_df.apply(remove_stop_words, axis=1)

def aplicaStemmer(df):
    stemmer = nltk.stem.RSLPStemmer()
    evolucao = df['EHR']
    cid = df['CID']

    evolComStem = ' '.join([str(stemmer.stem(p)) for p in evolucao.split()])
    return pd.Series([evolComStem, cid], index=['EHR', 'CID'])

amostrasComStemmer = amostrasSemStopWords.apply(aplicaStemmer, axis=1)
# print(amostrasComStemmer)

def aplicaTokenizer(df):
    tokenizer = nltk.word_tokenize
    evolucao = df['EHR']
    cid = df['CID']

    evolComTok = ' '.join([str(tokenizer(p,language='portuguese')) for p in evolucao.split()])
    return pd.Series([evolComTok, cid], index=['EHR', 'CID'])

amostrasComTok = amostrasComStemmer.apply(aplicaTokenizer, axis=1)
#print(amostrasComTok)

# Divisão das amostras:
X = amostrasComTok['EHR']
y = amostrasComTok['CID']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=42)

vectorizer = CountVectorizer()
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)

knn = KNeighborsClassifier(n_neighbors=5)  # Você pode ajustar o número de vizinhos
(n_neighbors) conforme necessário.
knn.fit(X_train, y_train)

y_pred = knn.predict(X_test)

precision = precision_score(y_test, y_pred, average='weighted', labels=np.unique(y_pred))
recall = recall_score(y_test, y_pred, average='weighted', labels=np.unique(y_pred))
f1 = f1_score(y_test, y_pred, average='weighted')
accuracy = accuracy_score(y_test, y_pred)

print(f'Acurácia: {accuracy}')
print(f'Precisão (Precision): {precision}')
print(f'Revocação (Recall): {recall}')
print(f'F-Score: {f1}')
```

Código 7 - Classificação utilizando o algoritmo KNN.

APÊNDICE H

Script com utilização de SVM

A seguir é apresentado o código Python com o modelo classificador utilizando o algoritmo SVM:

```
import nltk
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

amostras_excel = r"Planilha_EHR.xlsx"
amostras_df = pd.read_excel(amostras_excel)

amostras_df['EHR'] = amostras_df['EHR'].str.lower()

'''Retirando os números e parenteses de CID'''
pattern = r'\(\d+\)'
pattern_evol = r"[,;.:#-+/)(\d+>]"
amostras_df['CID'] = amostras_df['CID'].str.replace(pattern, '')
amostras_df['EHR'] = amostras_df['EHR'].str.replace(pattern_evol, '')

def remove_names_from_dataframe(csv_file_path, dataframe, column_name):
    try:
        # Read the CSV file into a list of names
        with open(csv_file_path, 'r') as file:
            names_to_remove = [line.strip() for line in file]

        # Check if the specified column exists in the DataFrame
        if column_name not in dataframe.columns:
            raise ValueError(f"Column '{column_name}' not found in the DataFrame.")

        # Remove rows containing names from the DataFrame
        dataframe = dataframe[~dataframe[column_name].isin(names_to_remove)]

        return dataframe
    except Exception as e:
        print(f"An error occurred: {str(e)}")
        return None

# Carregamento das bases de nomes comuns, em formato .csv:
nomes_masc = r'C:\Users\gabriel.silva\Downloads\ibge-mas-10000.csv'
nomes_fem = r'C:\Users\gabriel.silva\Downloads\ibge-fem-10000.csv'

# Retirada de nomes Masculinos:
amostras_df = remove_names_from_dataframe(nomes_masc,amostras_df,'EHR')

# Retirando os nomes Femininos:
amostras_df = remove_names_from_dataframe(nomes_fem,amostras_df,'EHR')

# Retirada de Stopwords:
stopWordNLTK = nltk.corpus.stopwords.words('portuguese')

def remove_stop_words(df):
    evolucao = df['EHR']
```

```
    cid = df['CID']

    if pd.isna(evolucao) or evolucao.strip() == '':
        return pd.Series([None, cid], index=['EHR', 'CID'])

    semStopWord = ' '.join([p for p in evolucao.split() if p not in stopWordNLTK])
    return pd.Series([semStopWord, cid], index=['EHR', 'CID'])

amostrasSemStopWords = amostras_df.apply(remove_stop_words, axis=1)

def aplicaStemmer(df):
    stemmer = nltk.stem.RSLPStemmer()
    evolucao = df['EHR']
    cid = df['CID']

    evolComStem = ' '.join([str(stemmer.stem(p)) for p in evolucao.split()])
    return pd.Series([evolComStem, cid], index=['EHR', 'CID'])

amostrasComStemmer = amostrasSemStopWords.apply(aplicaStemmer, axis=1)

def aplicaTokenizer(df):
    tokenizer = nltk.word_tokenize
    evolucao = df['EHR']
    cid = df['CID']

    evolComTok = ' '.join([str(tokenizer(p,language='portuguese')) for p in evolucao.split()])
    return pd.Series([evolComTok, cid], index=['EHR', 'CID'])

amostrasComTok = amostrasComStemmer.apply(aplicaTokenizer, axis=1)

# Divisão das amostras:
X = amostrasComTok['EHR']
y = amostrasComTok['CID']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=42)

vectorizer = CountVectorizer()
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)

#Treinamento do modelo SVM
svm_classifier = SVC(kernel='linear')
svm_classifier.fit(X_train, y_train)

# Avaliação do modelo
y_pred = svm_classifier.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted', labels=np.unique(y_pred))
recall = recall_score(y_test, y_pred, average='weighted', labels=np.unique(y_pred))
f1 = f1_score(y_test, y_pred, average='weighted')

print(f'Acurácia: {accuracy}')
print(f'Precisão (Precision): {precision}')
print(f'Revocação (Recall): {recall}')
print(f'F-Score: {f1}')
```

Código 8 - Classificação dos PME com utilização de SVM.

APÊNDICE I

Script com utilização de MLP

A seguir é apresentado o código em Python com o modelo classificador utilizando o algoritmo MLP:

```
import nltk
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

amostras_excel = r".\Planilha_EHR.xlsx"
amostras_df = pd.read_excel(amostras_excel)

pattern = r'\(\d+\)'
pattern_evol = r"[,;.:#-+/)(\d+>]"
amostras_df['CID'] = amostras_df['CID'].str.replace(pattern, '')
amostras_df['EHR'] = amostras_df['EHR'].str.replace(pattern_evol, '')

def remove_names_from_dataframe(csv_file_path, dataframe, column_name):
    try:
        with open(csv_file_path, 'r') as file:
            names_to_remove = [line.strip() for line in file]

        if column_name not in dataframe.columns:
            raise ValueError(f"Column '{column_name}' not found in the DataFrame.")

        dataframe = dataframe[~dataframe[column_name].isin(names_to_remove)]

        return dataframe
    except Exception as e:
        print(f"An error occurred: {str(e)}")
        return None

nomes_masc = r'C:\Users\gabriel.silva\Downloads\ibge-mas-10000.csv'
nomes_fem = r'C:\Users\gabriel.silva\Downloads\ibge-fem-10000.csv'

amostras_df = remove_names_from_dataframe(nomes_masc,amostras_df,'EHR')
amostras_df = remove_names_from_dataframe(nomes_fem,amostras_df,'EHR')

stopWordNLTK = nltk.corpus.stopwords.words('portuguese')
def remove_stop_words(df):
    evolucao = df['EHR']
    cid = df['CID']

    if pd.isna(evolucao) or evolucao.strip() == '':
        return pd.Series([None, cid], index=['EHR', 'CID'])

    semStopWord = ' '.join([p for p in evolucao.split() if p not in stopWordNLTK])
    return pd.Series([semStopWord, cid], index=['EHR', 'CID'])

amostrasSemStopWords = amostras_df.apply(remove_stop_words, axis=1)

def aplicaStemmer(df):
    stemmer = nltk.stem.RSLPStemmer()
    evolucao = df['EHR']
```

```
    cid = df['CID']

    evolComStem = ' '.join([str(stemmer.stem(p)) for p in evolucao.split()])
    return pd.Series([evolComStem, cid], index=['EHR', 'CID'])

amostrasComStemmer = amostrasSemStopWords.apply(aplicaStemmer, axis=1)

def aplicaTokenizer(df):
    tokenizer = nltk.word_tokenize
    evolucao = df['EHR']
    cid = df['CID']

    evolComTok = ' '.join([str(tokenizer(p,language='portuguese')) for p in evolucao.split()])
    return pd.Series([evolComTok, cid], index=['EHR', 'CID'])

amostrasComTok = amostrasComStemmer.apply(aplicaTokenizer, axis=1)

X = amostrasComTok['EHR']
y = amostrasComTok['CID']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=42)

vectorizer = CountVectorizer()
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)

mlp = MLPClassifier(hidden_layer_sizes=(100,), max_iter=300, random_state=42)
mlp.fit(X_train, y_train)

y_pred = mlp.predict(X_test)

precision = precision_score(y_test, y_pred, average='weighted', labels=np.unique(y_pred))
recall = recall_score(y_test, y_pred, average='weighted', labels=np.unique(y_pred))
f1 = f1_score(y_test, y_pred, average='weighted')
accuracy = accuracy_score(y_test, y_pred)

print(f'Acurácia: {accuracy}')
print(f'Precisão (Precision): {precision}')
print(f'Revocação (Recall): {recall}')
print(f'F-Score: {f1}')
```

Código 9 - Modelo classificador utilizando o algoritmo MLP.

APÊNDICE J

Script com utilização de LSTM

A seguir é apresentado o código em Python do modelo classificador utilizando o algoritmo LSTM treinado com amostras de mundo real:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Embedding, LSTM, Dense
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences

amostras_excel = r"/content/Planilha_EHR.xlsx"
amostras_df = pd.read_excel(amostras_excel)
print(amostras_df)

def remove_names_from_dataframe(csv_file_path, dataframe, column_name):
    try:
        with open(csv_file_path, 'r') as file:
            names_to_remove = [line.strip() for line in file]

        if column_name not in dataframe.columns:
            raise ValueError(f"Column '{column_name}' not found in the DataFrame.")

        dataframe = dataframe[~dataframe[column_name].isin(names_to_remove)]

        return dataframe
    except Exception as e:
        print(f"An error occurred: {str(e)}")
        return None

nomes_masc = '/content/ibge-mas-10000.csv'
nomes_fem = '/content/ibge-fem-10000.csv'

Check_nomesMasc = remove_names_from_dataframe(nomes_masc,amostras_df,'EHR')
Check_nomesFem = remove_names_from_dataframe(nomes_fem,amostras_df,'EHR')

pattern = r'\(\d+\)'
pattern_evol = r"[,;.:#-+/)(\d+>]"
amostras_df['CID'] = amostras_df['CID'].str.replace(pattern, '')
amostras_df['EHR'] = amostras_df['EHR'].str.replace(pattern_evol, '')

amostras_df['EHR'] = amostras_df['EHR'].str.lower()
amostras_df['EHR'] = amostras_df['EHR'].str.replace('[^a-zA-Z0-9]', ' ')

!pip install nltk

nltk.download('stopwords')

stopWordNLTK = nltk.corpus.stopwords.words('portuguese')

def remove_stop_words(df):
    evolucao = df['EHR']
    cid = df['CID']
```

```
    if pd.isna(evolucao) or evolucao.strip() == '':
        return pd.Series([None, cid], index=['EHR', 'CID'])

    semStopWord = ' '.join([p for p in evolucao.split() if p not in stopWordNLTK])
    return pd.Series([semStopWord, cid], index=['EHR', 'CID'])

amostras_df = amostras_df.apply(remove_stop_words, axis=1)

nltk.download('stopwords')

from nltk.stem import RSLPStemmer
nltk.download('rslp')

def aplicaStemmer(df):
    stemmer = nltk.stem.RSLPStemmer()
    evolucao = df['EHR']
    cid = df['CID']

    evolComStem = ' '.join([str(stemmer.stem(p)) for p in evolucao.split()])
    return pd.Series([evolComStem, cid], index=['EHR', 'CID'])

amostras_df = amostras_df.apply(aplicaStemmer, axis=1)

cid_counts = amostras_df['CID'].value_counts()

df_filtered = amostras_df[amostras_df['CID'].isin(cid_counts[cid_counts >= 6].index)]
df_filtered['CID'].value_counts()

tokenizer = Tokenizer(num_words=10000)
tokenizer.fit_on_texts(df_filtered['EHR'])
text_sequences = tokenizer.texts_to_sequences(df_filtered['EHR'])

max_len = 16516
padded_sequences = pad_sequences(text_sequences, maxlen=max_len)

print(padded_sequences)

le = LabelEncoder()
class_labels = le.fit_transform(df_filtered['CID'])

print(class_labels)

ehr_lengths = df_filtered['EHR'].str.len()
longest_ehr_length = ehr_lengths.max()
print(longest_ehr_length)

unique_tokens = set()

for text in df_filtered['EHR']:
    tokens = text.split()
    unique_tokens.update(tokens)

X_train, X_test, y_train, y_test = train_test_split(padded_sequences, class_labels, test_size=0.3)

y_train_encoded = to_categorical(y_train, num_classes=7)

embedding_layer = Embedding(input_dim=5562, output_dim=128, input_length=16516)
```

```
lstm_layer = LSTM(units=128, dropout=0.3)
dense_layer = Dense(7, activation='softmax')

model = Sequential()
model.add(embedding_layer)
model.add(lstm_layer)
model.add(dense_layer)

import tensorflow as tf

def precision(y_true, y_pred):
    true_positives = tf.keras.backend.sum(tf.keras.backend.round(tf.keras.backend.clip(y_true *
y_pred, 0, 1)))
    predicted_positives = tf.keras.backend.sum(tf.keras.backend.round(tf.keras.backend.clip(y_pred,
0, 1)))
    return true_positives / (predicted_positives + tf.keras.backend.epsilon())

def recall(y_true, y_pred):
    true_positives = tf.keras.backend.sum(tf.keras.backend.round(tf.keras.backend.clip(y_true *
y_pred, 0, 1)))
    actual_positives = tf.keras.backend.sum(tf.keras.backend.round(tf.keras.backend.clip(y_true, 0,
1)))
    return true_positives / (actual_positives + tf.keras.backend.epsilon())

def f1_score(y_true, y_pred):
    p = precision(y_true, y_pred)
    r = recall(y_true, y_pred)
    return 2 * ((p * r) / (p + r + tf.keras.backend.epsilon()))

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy', precision,
recall, f1_score])

model.fit(X_train, y_train_encoded, epochs=20)

y_pred = model.predict(X_test)

import numpy as np
for prev in y_pred:
    idx = np.argmax(prev)
    print("Index of maximum value in array:", prev, "is", idx)
```

Código 10 - Classificador utilizando o algoritmo LSTM treinado com amostras de mundo real.

APÊNDICE L

Script com utilização de LSTM em conjunto com amostras sintéticas

A seguir é apresentado o código Python do modelo classificador utilizando o algoritmo LSTM treinado com dados de mundo real e dados sintéticos:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Embedding, LSTM, Dense
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences

amostras_excel = r"/content/Planilha_EHR.xlsx"
amostras_df = pd.read_excel(amostras_excel)
print(amostras_df)

amostras_sinteticas = r"/content/Amostras__Sinteticas.xlsx"
amostras_sint = pd.read_excel(amostras_sinteticas)
amostras_df = pd.concat([amostras_df, amostras_sint], axis=0)
amostras_df.reset_index(drop=True, inplace=True)

def remove_names_from_dataframe(csv_file_path, dataframe, column_name):
    try:
        with open(csv_file_path, 'r') as file:
            names_to_remove = [line.strip() for line in file]

        if column_name not in dataframe.columns:
            raise ValueError(f"Column '{column_name}' not found in the DataFrame.")
        dataframe = dataframe[~dataframe[column_name].isin(names_to_remove)]
        return dataframe
    except Exception as e:
        print(f"An error occurred: {str(e)}")
        return None

nomes_masc = '/content/ibge-mas-10000.csv'
nomes_fem = '/content/ibge-fem-10000.csv'

Check_nomesMasc = remove_names_from_dataframe(nomes_masc,amostras_df,'EHR')
print(Check_nomesMasc)

Check_nomesFem = remove_names_from_dataframe(nomes_fem,amostras_df,'EHR')
print(Check_nomesFem)

pattern = r'\(\d+\)'
pattern_evol = r"[,;.:#-+/)(\d+>]"
amostras_df['CID'] = amostras_df['CID'].str.replace(pattern, '')
amostras_df['EHR'] = amostras_df['EHR'].str.replace(pattern_evol, '')

amostras_df['EHR'] = amostras_df['EHR'].str.lower()
amostras_df['EHR'] = amostras_df['EHR'].str.replace('[^a-zA-Z0-9]', ' ')
amostras_df['EHR'] = amostras_df['EHR'].str.replace('queixa principal: o paciente apresentou-se na
unidade de saúde', ' ')

!pip install nltk
nltk.download('stopwords')
```

```
stopWordNLTK = nltk.corpus.stopwords.words('portuguese')

def remove_stop_words(df):
    evolucao = df['EHR']
    cid = df['CID']

    if pd.isna(evolucao) or evolucao.strip() == '':
        return pd.Series([None, cid], index=['EHR', 'CID'])

    semStopWord = ' '.join([p for p in evolucao.split() if p not in stopWordNLTK])
    return pd.Series([semStopWord, cid], index=['EHR', 'CID'])

amostras_df = amostras_df.apply(remove_stop_words, axis=1)

from nltk.stem import RSLPStemmer
nltk.download('rslp')

def aplicaStemmer(df):
    stemmer = nltk.stem.RSLPStemmer()
    evolucao = df['EHR']
    cid = df['CID']

    evolComStem = ' '.join([str(stemmer.stem(p)) for p in evolucao.split()])
    return pd.Series([evolComStem, cid], index=['EHR', 'CID'])

amostras_df = amostras_df.apply(aplicaStemmer, axis=1)

cid_counts = amostras_df['CID'].value_counts()
df_filtered = amostras_df[amostras_df['CID'].isin(cid_counts[cid_counts >= 6].index)]
df_filtered['CID'].value_counts()

tokenizer = Tokenizer(num_words=10000)
tokenizer.fit_on_texts(df_filtered['EHR'])
text_sequences = tokenizer.texts_to_sequences(df_filtered['EHR'])
max_len = 16516
padded_sequences = pad_sequences(text_sequences, maxlen=max_len)

le = LabelEncoder()
class_labels = le.fit_transform(df_filtered['CID'])
ehr_lengths = df_filtered['EHR'].str.len()
longest_ehr_length = ehr_lengths.max()
unique_tokens = set()

for text in df_filtered['EHR']:
    tokens = text.split()
    unique_tokens.update(tokens)

X_train, X_test, y_train, y_test = train_test_split(padded_sequences, class_labels, test_size=0.2)
y_train_encoded = to_categorical(y_train, num_classes=14)

embedding_layer = Embedding(input_dim=6000, output_dim=128, input_length=16516)

# LSTM layer
lstm_layer = LSTM(units=128, dropout=0.2)

# Dense layer
dense_layer = Dense(14, activation='softmax')

# Sequential model
```

```
model = Sequential()
model.add(embedding_layer)
model.add(lstm_layer)
model.add(dense_layer)

import tensorflow as tf

def precision(y_true, y_pred):
    true_positives = tf.keras.backend.sum(tf.keras.backend.round(tf.keras.backend.clip(y_true *
y_pred, 0, 1)))
    predicted_positives = tf.keras.backend.sum(tf.keras.backend.round(tf.keras.backend.clip(y_pred,
0, 1)))
    return true_positives / (predicted_positives + tf.keras.backend.epsilon())

def recall(y_true, y_pred):
    true_positives = tf.keras.backend.sum(tf.keras.backend.round(tf.keras.backend.clip(y_true *
y_pred, 0, 1)))
    actual_positives = tf.keras.backend.sum(tf.keras.backend.round(tf.keras.backend.clip(y_true, 0,
1)))
    return true_positives / (actual_positives + tf.keras.backend.epsilon())

def f1_score(y_true, y_pred):
    p = precision(y_true, y_pred)
    r = recall(y_true, y_pred)
    return 2 * ((p * r) / (p + r + tf.keras.backend.epsilon()))

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy', precision,
recall, f1_score])

model.fit(X_train, y_train_encoded, epochs=20)

y_pred = model.predict(X_test)

import numpy as np
for prev in y_pred:
    idx = np.argmax(prev)
    print("Index of maximum value in array:", prev, "is", idx)
```

Código 11 - Classificador criado com o algoritmo LSTM treinado com amostras de mundo real e sintéticas.

ANEXO I

Aprovação do Projeto pelo Comitê de Ética em Pesquisa UFCSPA

UNIVERSIDADE FEDERAL DE CIÊNCIAS DA SAÚDE DE PORTO ALEGRE

Continuação do Parecer: 6.056.370

Considerações Finais a critério do CEP:

De acordo com o parecer do Relator.

Este parecer foi elaborado baseado nos documentos abaixo relacionados:

Tipo Documento	Arquivo	Postagem	Autor	Situação
Informações Básicas do Projeto	PB_INFORMAÇÕES_BÁSICAS_DO_PROJETO_2087039.pdf	31/03/2023 17:15:20		Aceito
Outros	Carta_CEP.pdf	31/03/2023 17:14:49	Gabriel Constantin da Silva	Aceito
Cronograma	Cronograma.pdf	28/02/2023 00:05:48	Gabriel Constantin da Silva	Aceito
Projeto Detalhado / Brochura Investigador	Projeto_CEP.pdf	17/02/2023 13:18:07	Gabriel Constantin da Silva	Aceito
Outros	Entrega_Relat_Preenchido_assinado.pdf	17/02/2023 13:16:41	Gabriel Constantin da Silva	Aceito
Declaração de Instituição e Infraestrutura	Termo_Dados_Preenchido_assinado.pdf	17/02/2023 13:15:36	Gabriel Constantin da Silva	Aceito
Folha de Rosto	Folha_de_Rosto_Silvio_assinado_data.pdf	17/02/2023 13:12:16	Gabriel Constantin da Silva	Aceito

Situação do Parecer:
Aprovado

Necessita Apreciação da CONEP:
Não

PORTO ALEGRE, 12 de Maio de 2023

Assinado por:
Fernanda Bordignon Nunes
(Coordenador(a))

ANEXO II

Aprovação do Projeto pelo Comitê de Ética em Pesquisa ISCMPA

IRMANDADE DA SANTA CASA DE MISERICORDIA DE PORTO ALEGRE - ISCMPA

Plataforma Brasil

Continuação do Parecer: 6.327.803

Este parecer foi elaborado baseado nos documentos abaixo relacionados:

Tipo Documento	Arquivo	Postagem	Autor	Situação
Informações Básicas do Projeto	PB_INFORMAÇOES_BASICAS_DO_PROJETO_2186141.pdf	25/09/2023 15:43:59		Aceito
Projeto Detalhado / Brochura Investigador	Proposta_Projeto_CEP_ISCMPA_02_02.pdf	25/09/2023 15:43:18	Gabriel Constantin da Silva	Aceito
Outros	Carta_Pesquisador_CEP_assinado.pdf	25/09/2023 15:41:23	Gabriel Constantin da Silva	Aceito
Projeto Detalhado / Brochura Investigador	Proposta_Projeto_CEP_ISCMPA.docx	18/08/2023 23:15:50	Gabriel Constantin da Silva	Aceito
TCLE / Termos de Assentimento / Justificativa de Ausência	TERMO_DE_SOLICITACAO_DE_DISPENSA_DO_TCLE_assinado.pdf	18/08/2023 23:13:30	Gabriel Constantin da Silva	Aceito
Outros	FORMULARIO_DE_INSCRICAO_DE_PROJETOS_DE_PESQUISA_COMITE_DE_ETICA_assinado.pdf	18/08/2023 23:13:18	Gabriel Constantin da Silva	Aceito
Outros	DECLARACAO_DE_CONFIDENCIALIDADE_DO_PARTICIPANTE_assinado.pdf	18/08/2023 23:12:30	Gabriel Constantin da Silva	Aceito
Outros	DECLARACAO_DE_UTILIZACAO_DE_DADOS_DE_PRONTUARIOS_assinado.pdf	18/08/2023 23:11:57	Gabriel Constantin da Silva	Aceito
Declaração de Instituição e Infraestrutura	DECLARACAO_DE_ISENCAO_DE_ONUS_A_INSTITUICAO_assinado.pdf	18/08/2023 23:10:56	Gabriel Constantin da Silva	Aceito
Brochura Pesquisa	Proposta_Projeto_CEP_ISCMPA.pdf	18/08/2023 23:07:26	Gabriel Constantin da Silva	Aceito
Folha de Rosto	folhaDeRosto_ISCMPA_assinado.pdf	18/08/2023 23:06:51	Gabriel Constantin da Silva	Aceito

Situação do Parecer:
Aprovado

Necessita Apreciação da CONEP:
Não

PORTO ALEGRE, 27 de Setembro de 2023

Assinado por:
RENATA NETO PIRES
(Coordenador(a))

Endereço: Avenida Osvaldo Aranha, nº 80, sala 17, Centro Administrativo da Santa Casa, 2º andar.
Bairro: Cidade Baixa **CEP:** 90.035-190
UF: RS **Município:** PORTO ALEGRE
Telefone: (51)3214-8571 **Fax:** (51)3214-8571 **E-mail:** cep@santacasa.tche.br

ANEXO III

Pôster Apresentado no 2º Congresso CI-IA saúde

CONSTANTIN, G.; CAZELLA, S. C. Aprendizado supervisionado de máquina para classificação automática de Prontuários Médicosda Cardiologia. In: II Encontro Anual do CI-IA Saúde, 2023, Belo Horizonte. Anais do I Simpósio CI-IA Saúde da UFMG. Belo Horizonte: UFMG, 2023. v. 1. p. 16-17.

Aprendizado supervisionado de máquina para classificação automática de Prontuários Médicos da Cardiologia

Gabriel Constantin da Silva[1]

1 Pós-graduando emTecnologia da Informação e Gestão em Saúde pela UFCSPA.

Orientadores:
Silvio Cesar Cazella - silvioc@ufcspa.edu.br

E-mail dos autores:
1 gabriel.constantin@ufcspa.edu.br

Introdução

O prontuário médico eletrônico (PME), preenchido por médicos, enfermeiros e técnicos, desempenha um papel fundamental na obtenção de dados organizados sobre a evolução clínica, contendo dados valiosos para pesquisas retrospectivas (1). Porém, os dados não são estruturados e possuem repetições e redundâncias. Neste contexto, o uso das técnicas de processamento de linguagem natural (PLN) permite o processamento e análise de documentos seguido da extração de informações relevantes (2). Algumas técnicas modernas de PLN têm se destacado, como a extração de palavras-chave, reconhecimento de entidades nomeadas, análise semântica e segmentação de tópicos (3).

Método

Pretende-se utilizar prontuários médicos reais assim que receber aprovação no Comitê de Ética em Pesquisa da Santa Casa de Porto Alegre. Realizaremos o pré-processamento dos dados sintéticos para formatação dos textos para análise. Com os dados pré-processados, serão treinados modelos para classificar os prontuários médicos com base em seu conteúdo. Utilizaremos bibliotecas NLTK (Natural Language Toolkit), spaCy e scikit-learn porque são amplamente utilizadas nesta tarefa e possuem uma grande comunidade de usuários.

Resultados

Como parte da realização deste trabalho, foi realizada uma revisão da literatura com o objetivo de encotrar trabalhos acadêmicos que tivessem a intenção de utilizar técnicas de aprendizado de máquina para análise de PME. De acordo com a Tabela 1 podemos observar que os modelos de aprendizado de máquino utilizados nos trabalhos variam de acordo com a técnica em Support Vector Machine (SVM), Recurrent Neural Network (RNN), k-Nearest Neighbors (KNN), e Long Short Term Memory (LSTM). O desempenho dos modelos será avaliado por acurácia, precisão e sensibilidade (4). O desafio enfrentado foi a criação do dataset sintético para simular informações reais, além de configurar o modelo para aprender padrões e realizar classificações precisas. O aprendizado obtido com este trabalho envolveu em especial as etapas de preparação do conjunto de PME utilizados nos modelos de classificação, segundo o diagnóstico atribuído pelo profissional médico.

Trabalhos Relacionados	Uso PME?	Aprendizado Supervisionado?	Modelo Utilizado?	Cardiologia?	Detecção de Diagnóstico
(PRABHAKAR, 2021)	Sim	Não	LSTM	Não	Não
(CHEN et al., 2019)	Sim	Sim	KNN	Sim	Não
(KRITTANAWONG et al., 2021)	Sim	Não	RNN	Sim	Não
(SIMON et al., 2016)	Sim	Sim	SVM	Sim	Não
Proposta de Pesquisa	Sim	Sim	SVM, RNN	Sim	Sim

Tabela 1. Trabalhos relacionados à proposta desta pesquisa.

Discussão

Existem uma série de métodos para determinar a eficácia de um modelo; no entanto, Precisão, Recall, Acurácia e AUC são mais amplamente utilizadas. A precisão dos modelos determinará a probabilidade condicional de que um documento aleatório seja classificado como pertencente à classificação correta, ou seja, representa a habilidade do classificador em alocar um documento como pertencente à categoria correta, em oposição a todos documentos alocados nesta referida categoria, de forma correta ou incorreta. A métrica de Recall representa a probabilidade de um documento aleatório seja classificado em uma dada categoria (5). As medidas citadas anteriormente serão utilizadas para comparar os modelos entre si, e posteriormente servirão como referência para melhoria de novos modelos com esse objetivo. A aplicação de inteligência artificial, na forma de aprendizado supervisionado de máquina, apresenta muitas possibilidades para melhoria de processos na área da saúde.

Conclusão

Este projeto terá um caráter experimental, na medida em que fará uma comparação entre diferentes modelos de aprendizado de máquina aplicados para a análise de textos sintéticos, e que posteriomente serão utilizados para processamento de prontuários reais, pretendendo gerar uma prova de conceito (PoC). O projeto só será iniciado após aprovação pelo Comitê de Ética em Pesquisa da UFCSPA e pelo Comitê de Ética da entidade co-participante, Santa Casa de Misericórdia de Porto Alegre.

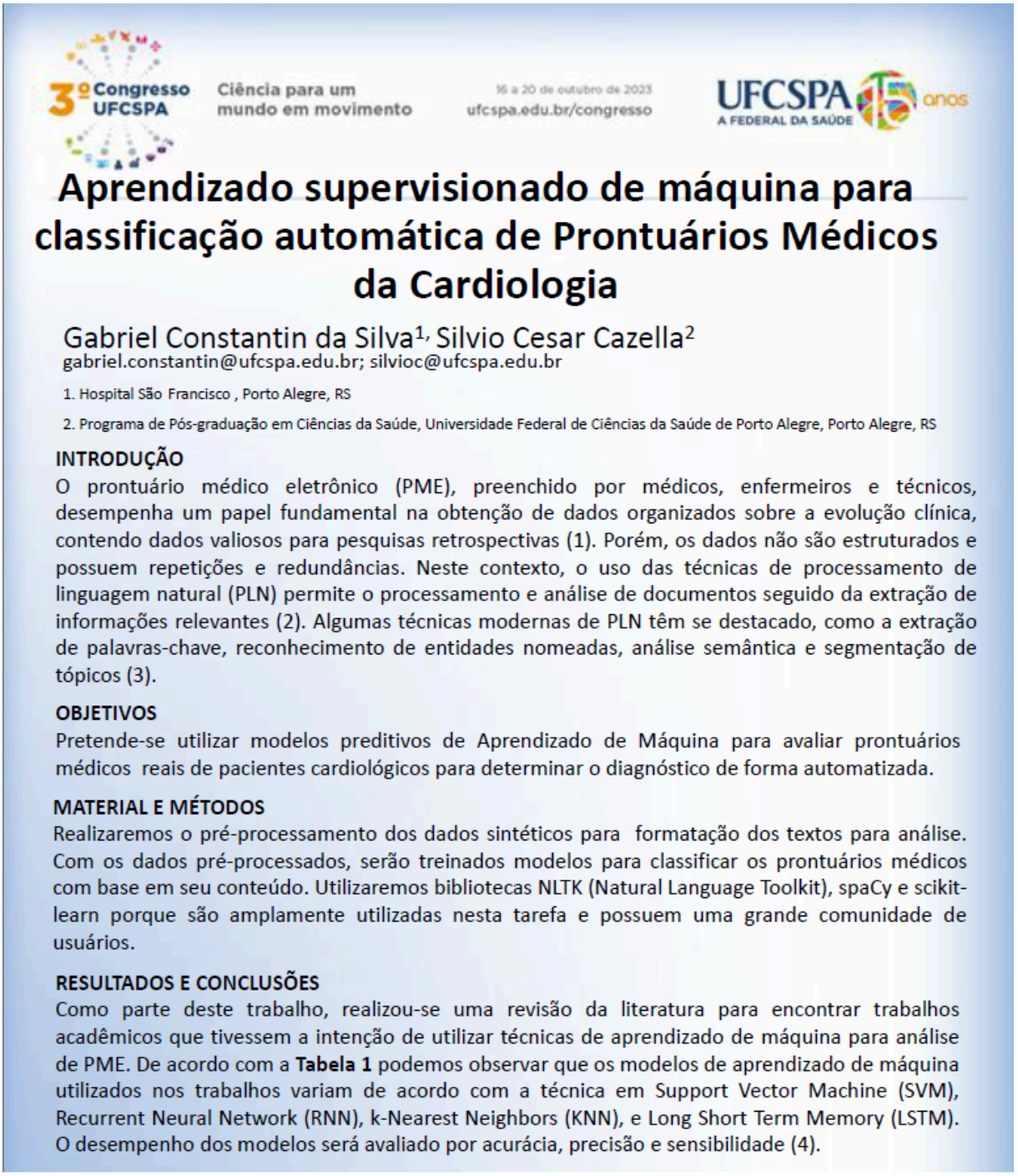
Referências

1. Peek N, Holmes JH, Sun J. Technical challenges for big data in biomedicine and health: data sources, infrastructure, and analytics. Yearb Med Inform. 2014 Aug 15;9(1):42-7. doi: 10.15265/IY-2014-0018. PMID: 25123720; PMCID: PMC4287098.
2. Meystre SM, Savova GK, Kipper-Schuler KC, Hurdle JF. Extracting information from textual documents in the electronic health record: a review of recent research. Yearb Med Inform. 2008:128-44. PMID: 18660887.
3. Beliga S, Meštrović A, Martinčić-Ipšić S. An Overview of Graph-Based Keyword Extraction Methods and Approaches. J Inform Organ Sci. 2015;39:1-20.
4. Hossin M, Sulaiman MN. A review on evaluation metrics for data classification evaluations. Int J Data Min Knowledge Manage Process. 2015;5(2):1.
5. Ikonomakis M.; Kotsiantis S.; Tampakas V. Text Classification Using Machine Learning Techniques. WSEAS TRANSACTIONS on COMPUTERS, v.4, p. 966-974, August 2005,

ANEXO IV

Pôster Apresentado no 3º Congresso UFCSPA

(CONSTANTIN, G.; CAZELLA, S. C. “Aprendizado De Máquina Supervisionado Para Classificação Automática De Prontuários Médicos Da Cardiologia". Congresso da Universidade Federal de Ciências da Saúde de Porto Alegre, 3, 2023, Porto Alegre, Pôster)



3º Congresso UFCSPA — Ciência para um mundo em movimento — 16 a 20 de outubro de 2023 — ufcspa.edu.br/congresso — UFCSPA A FEDERAL DA SAÚDE 15 anos

Aprendizado supervisionado de máquina para classificação automática de Prontuários Médicos da Cardiologia

Gabriel Constantin da Silva[1], Silvio Cesar Cazella[2]
gabriel.constantin@ufcspa.edu.br; silvioc@ufcspa.edu.br

1. Hospital São Francisco , Porto Alegre, RS

2. Programa de Pós-graduação em Ciências da Saúde, Universidade Federal de Ciências da Saúde de Porto Alegre, Porto Alegre, RS

INTRODUÇÃO

O prontuário médico eletrônico (PME), preenchido por médicos, enfermeiros e técnicos, desempenha um papel fundamental na obtenção de dados organizados sobre a evolução clínica, contendo dados valiosos para pesquisas retrospectivas (1). Porém, os dados não são estruturados e possuem repetições e redundâncias. Neste contexto, o uso das técnicas de processamento de linguagem natural (PLN) permite o processamento e análise de documentos seguido da extração de informações relevantes (2). Algumas técnicas modernas de PLN têm se destacado, como a extração de palavras-chave, reconhecimento de entidades nomeadas, análise semântica e segmentação de tópicos (3).

OBJETIVOS

Pretende-se utilizar modelos preditivos de Aprendizado de Máquina para avaliar prontuários médicos reais de pacientes cardiológicos para determinar o diagnóstico de forma automatizada.

MATERIAL E MÉTODOS

Realizaremos o pré-processamento dos dados sintéticos para formatação dos textos para análise. Com os dados pré-processados, serão treinados modelos para classificar os prontuários médicos com base em seu conteúdo. Utilizaremos bibliotecas NLTK (Natural Language Toolkit), spaCy e scikit-learn porque são amplamente utilizadas nesta tarefa e possuem uma grande comunidade de usuários.

RESULTADOS E CONCLUSÕES

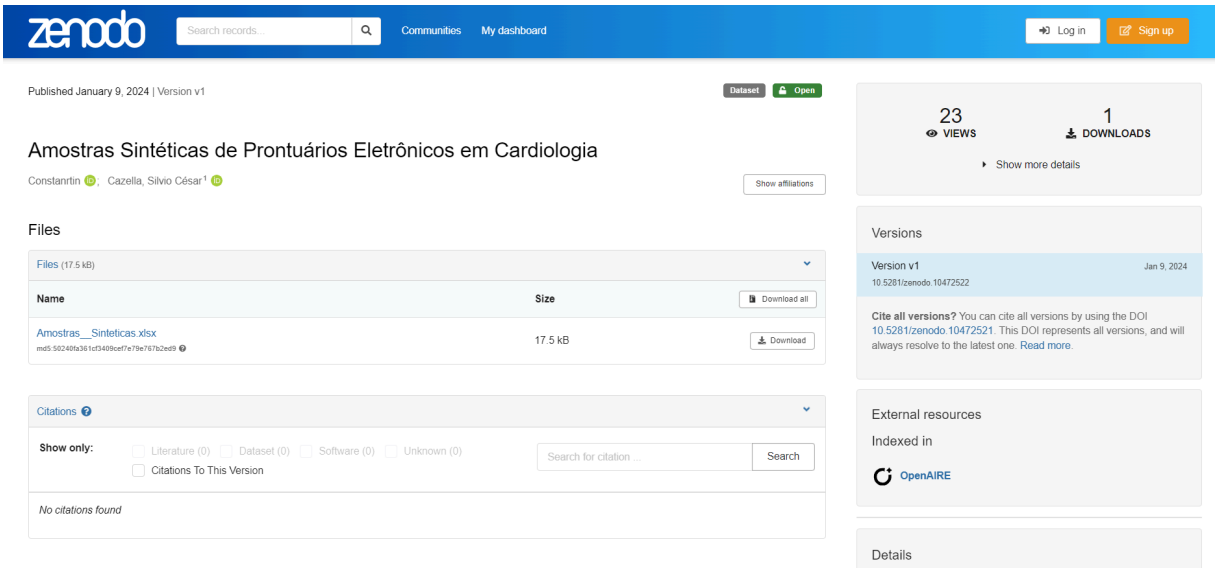
Como parte deste trabalho, realizou-se uma revisão da literatura para encontrar trabalhos acadêmicos que tivessem a intenção de utilizar técnicas de aprendizado de máquina para análise de PME. De acordo com a **Tabela 1** podemos observar que os modelos de aprendizado de máquina utilizados nos trabalhos variam de acordo com a técnica em Support Vector Machine (SVM), Recurrent Neural Network (RNN), k-Nearest Neighbors (KNN), e Long Short Term Memory (LSTM). O desempenho dos modelos será avaliado por acurácia, precisão e sensibilidade (4).

ANEXO V

Conjunto de dados sintéticos publicado no site Zenodo

Zenodo é um repositório de pesquisa de propósito geral onde qualquer resultado de pesquisa pode ser compartilhado por seus usuários.

CONSTANTIN, G. ; CAZELLA, S. C. . Amostras Sintéticas de Prontuários Eletrônicos em Cardiologia [Dataset]. (https://doi.org/10.5281/zenodo.10472522). 2024. (Base de dados técnico-científica) 2024.



Anexo VI
Instruções para submissão à Revista Cadernos de Saúde Pública

1 Envio do Artigo

1.1 – A submissão online é feita na área restrita de gerenciamento de artigos. O autor deve acessar a seção “Submeta seu texto”.

1.2 – A primeira etapa do processo de submissão consiste na verificação das normas de publicação de CSP. O artigo somente será avaliado pela Secretaria Editorial de CSP se cumprir todas essas normas.

1.3 – Na segunda etapa são inseridos os dados referentes ao artigo: título, título resumido, área de concentração, palavras-chave, informações sobre financiamento e conflito de interesses, resumo e agradecimentos, quando necessário. Se desejar, o autor pode sugerir potenciais consultores (nome, e-mail e instituição) que ele julgue capaz de avaliar o artigo.

1.4 – Na terceira etapa são incluídos o(s) nome(s) do(s) autor(es), respectiva(s) instituição(ões) por extenso, com endereço completo, telefone e e-mail, bem como a colaboração de cada um e o respectivo número de registro no ORCID (https://orcid.org/). Não serão aceitos autores sem registro. O autor que cadastrar o artigo, automaticamente será incluído como autor do artigo e designado autor de correspondência. A ordem dos nomes dos autores deverá ser estabelecida no momento da submissão.

1.5 – Na quarta etapa é feita a transferência do arquivo com o corpo do texto e as referências.

1.6 – O arquivo com o texto do artigo deve estar nos formatos DOC (Microsoft Word), RTF (Rich Text Format) ou ODT (Open Document Text) e não deve ultrapassar 1MB.

1.7 – O texto deve ser apresentado em espaço 1,5cm, fonte Times New Roman, tamanho 12.

1.8 – O arquivo com o texto deve conter somente o corpo do artigo e as referências bibliográficas. Os seguintes itens deverão ser inseridos em campos à parte durante o processo de submissão: resumos; nome(s) do(s) autor(es), afiliação ou qualquer outra informação que identifique o(s) autor(es); agradecimentos e colaborações; ilustrações (fotografias, fluxogramas, mapas, gráficos e tabelas).

1.9 – Na quinta etapa são transferidos os arquivos das ilustrações do artigo

(fotografias, fluxogramas, mapas, gráficos e tabelas), quando necessário. Cada ilustração deve ser enviada em arquivo separado clicando em “Transferir”.

1.10 – Os autores devem obter autorização, por escrito, dos detentores dos direitos de reprodução de ilustrações que já tenham sido publicadas anteriormente.

1.11 – Finalização da submissão. Ao concluir o processo de transferência de todos os arquivos, clique em “Finalizar Submissão”.

1.12 – Confirmação da submissão. Após a finalização da submissão, o autor receberá uma mensagem por e-mail confirmando o recebimento do artigo pelos CSP. Caso não receba o e-mail de confirmação dentro de 24 horas, entre em contato com a Secretaria Editorial de CSP no endereço: cadernos@ensp.fiocruz.br ou cadernos@fiocruz.br.

2 Preparação Do Manuscrito

2.1 – O título completo (no idioma original do artigo) deve ser conciso e informativo, e conter, no máximo, 150 caracteres com espaços.

2.2 – O título corrido poderá ter o máximo de 70 caracteres com espaços.

2.3 – As palavras-chave (mínimo de 3 e máximo de 5 no idioma original do artigo) devem constar na base do DeCS (Descritores em Ciências da Saúde) da Biblioteca Virtual em Saúde BVS.

2.4 – Resumo. Com exceção das contribuições enviadas às seções Resenhas, Cartas, Comentários ou Perspectivas, todos os artigos submetidos deverão ter resumo no idioma original do artigo, podendo ter no máximo 1.700 caracteres com espaços. Visando a ampliar o alcance dos artigos publicados, CSP publica os resumos nos idiomas português, inglês e espanhol. No intuito de garantir um padrão de qualidade do trabalho, oferecemos gratuitamente a tradução do Resumo para os idiomas a serem publicados. Não são aceitos equações e caracteres especiais (por exemplo: letras gregas, símbolos) no Resumo.

2.4.1 – Como o Resumo do artigo alcança maior visibilidade e distribuição do que o artigo em si, indicamos a leitura atenta da recomendação específica para sua elaboração (Leia mais).

2.5 – Equações e Fórmulas: as equações e fórmulas matemáticas devem ser desenvolvidas diretamente nos editores (Math, Equation, Mathtype ou outros que sejam equivalentes). Não serão aceitas equações e fórmulas em forma de imagem.

2.6 – Agradecimentos. Possíveis agradecimentos às instituições e/ou pessoas

poderão ter no máximo 500 caracteres com espaços.

2.7 – Quadros. Destina-se a apresentar as informações de conteúdo qualitativo, textual do artigo, dispostas em linhas e/ou colunas. Os quadros podem ter até 17cm de largura, com fonte de tamanho 9. Devem ser submetidos em arquivo text: DOC (Microsoft Word), RTF (Rich Text Format) ou ODT (Open Document TEXT). Cada dado do quadro deve ser inserido em uma célula separadamente, ou seja, não incluir mais de uma informação dentro da mesma célula.

2.8 – Tabelas. Destina-se a apresentar as informações quantitativas do artigo. As tabelas podem ter até 17cm de largura, com fonte de tamanho 9. Devem ser submetidas em arquivo de texto: DOC (Microsoft Word), RTF (Rich Text Format) ou ODT (Open Document Text). As tabelas devem ser numeradas (algarismos arábicos) de acordo com a ordem em que aparecem no texto, e citadas no corpo do mesmo. Cada dado na tabela deve ser inserido em uma célula separadamente, e dividida em linhas e colunas. Ou seja, não incluir mais de uma informação dentro da mesma célula.

2.9 – Figuras. Os seguintes tipos de figuras serão aceitos por CSP: mapas, gráficos, imagens de satélite, fotografias, organogramas, e fluxogramas. As Figuras podem ter até 17cm de largura. O arquivo de cada figura deve ter o tamanho máximo de 10Mb para ser submetido, devem ser desenvolvidas e salvas/exportadas em formato vetorial/editável. As figuras devem ser numeradas (algarismos arábicos) de acordo com a ordem em que aparecem no texto, e devem ser citadas no corpo do mesmo.

2.9.1 – Os mapas devem ser submetidos em formato vetorial e são aceitos nos seguintes tipos de arquivo: WMF (Windows MetaFile), EPS (Encapsuled PostScript) ou SVG (Scalable Vectorial Graphics). Nota: os mapas gerados originalmente em formato de imagem e depois exportados para o formato vetorial não serão aceitos.

2.9.2 – Os gráficos devem ser submetidos em formato vetorial e são aceitos nos seguintes tipos de arquivo: XLS (Microsoft Excel), ODS (Open Document Spreadsheet), WMF (Windows MetaFile), EPS (Encapsuled PostScript) ou SVG (Scalable Vectorial Graphics).

2.9.3 – As imagens de satélite e fotografias devem ser submetidas nos seguintes tipos de arquivo: TIFF (Tagged Image File Format) ou BMP (Bitmap). A resolução mínima deve ser de 300dpi (pontos por polegada), com tamanho mínimo de 17,5cm de largura. O tamanho limite do arquivo deve ser de 10Mb.

2.9.4 – Os organogramas e fluxogramas devem ser submetidos em arquivo de texto

ou em formato vetorial e são aceitos nos seguintes tipos de arquivo: DOC (Microsoft Word), RTF (Rich Text Format), ODT (Open Document Text), WMF (Windows MetaFile), EPS (Encapsuled PostScript) ou SVG (Scalable Vectorial Graphics).

2.9.5 – Formato vetorial. O desenho vetorial é originado com base em descrições geométricas de formas e normalmente é composto por curvas, elipses, polígonos, texto, entre outros elementos, isto é, utilizam vetores matemáticos para sua descrição.

2.10 – Títulos e legendas de figuras devem ser apresentados em arquivo de texto separado dos arquivos das figuras.

2.11 – CSP permite a publicação de até cinco ilustrações (Figuras e/ou Quadros e/ou Tabelas) por artigo. Ultrapassando esse limite os autores deverão arcar com os custos extras.

Figuras compostas são contabilizadas separadamente; cada ilustração é considerada uma figura.

APÊNDICE M
Artigo Científico para submissão a Revista Cadernos de Saúde Pública

Aprendizado de Máquina Supervisionado para Classificação Automática de Prontuários Médicos da Cardiologia

Supervised Machine Learning for Automatic Classification of Cardiology Medical Records

Aprendizaje Automático Supervisado para la Clasificación Automática de Historias Clínicas de Cardiología

Resumo: A classificação automática de Prontuários Médicos Eletrônicos (PME) usando aprendizado de máquina supervisionado é uma solução capaz de lidar com volume e complexidade destes documentos. Textos reais de PME de cardiologia, e textos sintéticos criados por ChatGPT foram usados para treinar e avaliar diversos algoritmos de classificação. A regressão logística alcançou a maior precisão geral e escore F1. Os registros sintéticos gerados pelo ChatGPT melhoraram o desempenho do LSTM, alcançando a maior acurácia (0,93) e precisão (0,97). Essas descobertas sugerem o uso de aprendizado de máquina para classificação automatizada confiável, potencialmente auxiliando os cardiologistas na tomada de decisões após validação adicional.

Aprendizado de Máquina Supervisionado; Registros Eletrônicos de Saúde; Cardiologia; Processamento de Linguagem Natural; Tomada de Decisões

Introdução

A classificação automática de textos sempre foi objeto de pesquisa, e sua importância aumenta exponencialmente com a crescente quantidade de documentos eletrônicos gerados diariamente[1]. Para lidar com esse volume e a complexidade dos textos, tecnologias de aprendizado de máquina (Machine Learning) se tornam indispensáveis[2].

O Prontuário Médico Eletrônico (PME) é um documento digital que reúne informações sobre a saúde do paciente e a assistência prestada a ele. É obrigatório por lei e deve ser preenchido por todos os profissionais que atendem o paciente. O PME oferece

vantagens como eficiência na organização e acesso às informações, qualidade na avaliação da assistência prestada e potencial para pesquisas médicas. No entanto, ainda há desafios a serem superados, como o preenchimento correto por parte dos profissionais de saúde, a integração entre diferentes sistemas de PME e a análise automatizada dos dados em formato de texto livre[3,4].

O aprendizado de máquina supervisionado, utilizando dados rotulados por humanos, permite que a máquina aprenda e faça previsões em novos exemplos. Essa técnica se divide em regressão (para dados numéricos) e classificação (para dados binários ou multiclasse). Na análise textual supervisionada, um conjunto de dados com rótulos humanos é usado para treinar algoritmos como SVM, Regressão Logística, KNN e Naive Bayes, possibilitando a categorização de textos por tópico ou gênero[5].

O objetivo deste artigo é desenvolver modelos de aprendizado de máquina supervisionado para a classificação automática de textos de prontuários eletrônicos de pacientes atendidos em um hospital de cardiologia de Porto Alegre, Rio Grande do Sul.

Métodos

Desenho do estudo

A pesquisa está focada na área de saúde, especificamente na gestão de documentos eletrônicos gerados no atendimento em saúde, como os PME. O desafio é lidar com o volume significativo de informações relacionadas, como doenças e condições clínicas, contidas nos PME. O objetivo da pesquisa é desenvolver modelos de aprendizado de máquina supervisionado para a classificação automática de textos de prontuários eletrônicos de pacientes atendidos em um hospital de cardiologia.

A pesquisa é aplicada e experimental. As etapas seguidas foram: coleta de amostras de PME, pré-processamento das amostras de textos, treinamento dos modelos preditivos utilizando textos pré-processados, criação de amostras sintéticas com uso de Inteligência Artificial Generativa, avaliação do desempenho dos modelos com base nas métricas de acurácia, precisão, abrangência e F-Score.

População do estudo

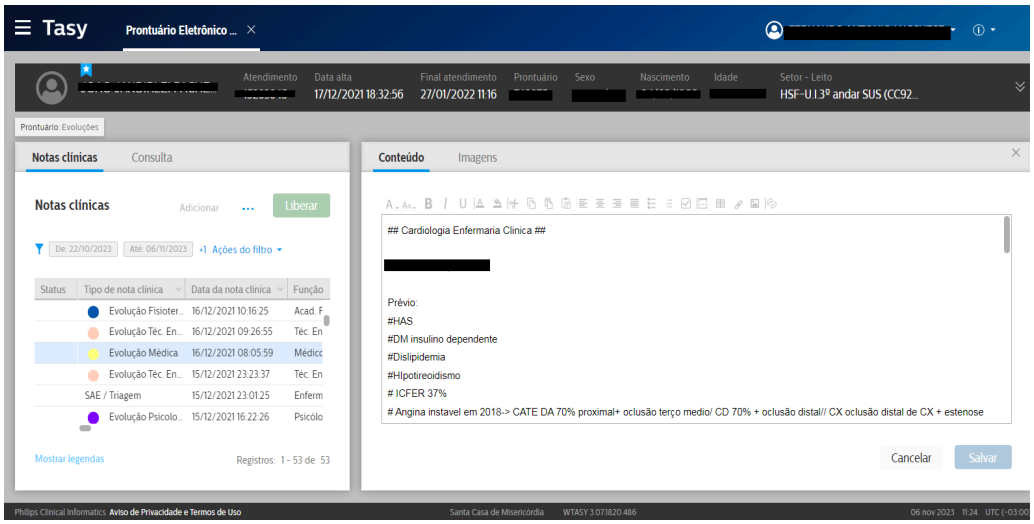
Foram selecionados prontuários de pacientes com condições cardiológicas, internados em Unidades de Terapia Intensiva (clínica ou cirúrgica). A coleta dos dados foi realizada manualmente, copiando os textos do sistema e colando-os em arquivos .txt, com identificação da Classificação Internacional de Doenças (CID).

Base de Dados

O estudo utilizou prontuários eletrônicos do sistema Tasy® do Hospital São Francisco da Santa Casa de Porto Alegre, entre setembro de 2021 e janeiro de 2022. Foram selecionados prontuários de pacientes com condições cardiológicas internados em UTIs (clínica ou cirúrgica).

Os arquivos foram organizados em uma planilha no Excel com as colunas "EHR" (texto da nota clínica) e "CID" (código do diagnóstico). As notas clínicas refletem as condições de saúde dos pacientes de forma mais contínua. A divisão das evoluções por categorias profissionais pode prejudicar o compartilhamento de informações. O estudo foi aprovado pelos Comitês de Ética em Pesquisa da UFCSPA e da Irmandade Santa Casa de Misericórdia de Porto Alegre. A **Figura 1** ilustra um exemplo de como as notas clínicas se encontram registradas no sistema Tasy®.

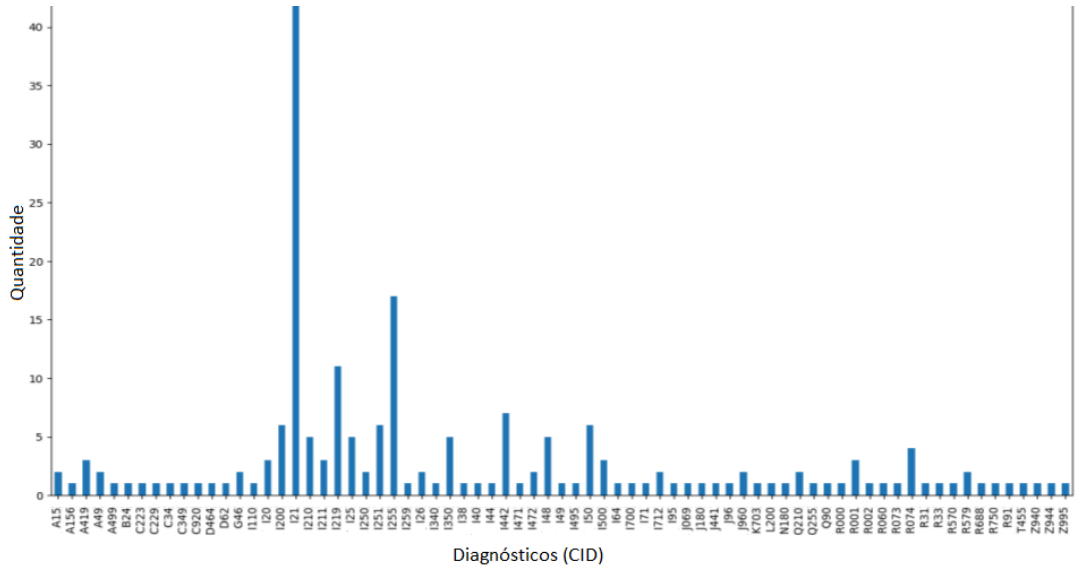
Figura 1 - Imagem da tela de visualização de um prontuário de paciente



Fonte: Próprio autor

Realizou-se uma análise de ocorrência de amostras obtidas a partir da coleta do material. Os diagnósticos mais frequentes foram visualizados através da construção de uma gráfico do tipo histograma utilizando-se a biblioteca Matplotlib **Figura 2**.

Figura 2 - Histograma com a distribuição de diagnósticos



Fonte: Próprio autor

Anonimização dos PME Reais

Para preservar a privacidade e anonimato dos dados contidos nos PMEs, foi desenvolvida uma função específica para realizar a anonimização. Essa função identificou e removeu os nomes masculinos e femininos mais comuns na língua portuguesa, conforme listas fornecidas pelo Instituto Brasileiro de Geografia e Estatística (IBGE).

Limpeza das Amostras de PME

Realizou-se a limpeza das amostras para a retirada de palavras que não possuem valor para a análise do texto. Nessa etapa são definidas expressões regulares para remoção de padrões específicos de texto. A função "remove_stop_words()" foi definida para eliminar stopwords da coluna "EHR" do DataFrame, resultando em uma nova série de dados onde essas palavras são removidas.

Tokenização

Esse processo busca fazer a segmentação do texto em elementos menores e que carregam o significado da palavra original[6].

Os tokens de palavras são as unidades básicas de texto envolvidas em qualquer tarefa de PLN. O primeiro passo ao processar texto é dividi-lo em tokens. A biblioteca NLTK fornece diferentes tipos de tokenizadores para realizar essa tarefa através de diferentes técnicas[7].

Stemização e Lematização

A derivação ou Stemização (do inglês, stemming) é uma etapa de pré-processamento de texto que transforma palavras relacionadas ou similares (como 'caminhando') em sua forma base (como 'caminhar'), já que possuem o mesmo significado. Uma das transformações básicas é reduzir palavras no plural para o singular: 'maçãs' se torna 'maçã', por exemplo. Embora essa seja uma transformação simples, existem outras mais complexas[7].

Há também o processo de “lematização”, que baseia-se na transformação de palavras para que ela possa ser reconhecida por apenas um simples elemento, conhecido como lema[8].

Bag-of-Words

A técnica chamada Bag-of-Words (BOW), ou Bolsa de Palavras, que representa as palavras através de vetores de dicionários, onde o valor individual do vetor denota a frequência da palavra no texto, criando uma matriz de ocorrência para a sentença ou documento sem considerar a gramática e ordem das palavras, que é então usada para o treinamento do modelo classificador[9].

Label Encoder

Alguns modelos classificadores exigem transformações específicas da amostra textual. Por exemplo, a transformação das classificações em códigos numéricos a partir do uso de LabelEncoder para representar as classes, é uma etapa de pré-processamento essencial de modelos como SVM e Regressão Logística. LabelEncoder é uma técnica usada em aprendizado de máquina para converter variáveis categóricas (textuais) em valores numéricos[10].

Transformação dos PME em Vetores

No contexto do uso de Long Short-Term Memory (LSTM), foi necessário converter as amostras de texto para o formato de vetores. Esta transformação está relacionada à criação de uma camada de inserção a ser empregada em uma rede neural recorrente, como é o caso da rede LSTM.

Criação dos Prontuários Sintéticos Utilizando ChatGPT

Para a criação dos prontuários médicos sintéticos, utilizou-se a ferramenta ChatGPT (https://openai.com/chatgpt/), desenvolvida pela OpenAI. O processo foi dividido em duas etapas principais: a criação do modelo de prontuários e a geração dos prontuários com diferentes diagnósticos.

- **Etapa 1: Criação do Modelo de Prontuários**

Na primeira etapa, foi necessário desenvolver um modelo base de prontuário médico. Através do campo de entrada de texto (*prompt*) do ChatGPT, solicitou-se a geração de um prontuário médico que incluísse informações detalhadas e pertinentes ao contexto hospitalar. Por exemplo, foi solicitado ao ChatGPT que gerasse um prontuário de um paciente cardiológico que apresentasse fibrilação atrial, escrito por um enfermeiro.

Durante este processo inicial, o modelo produziu prontuários que incluíam campos como nome do paciente, idade, sexo, histórico médico, exame físico, exames complementares, diagnóstico e conduta. Este primeiro esboço foi revisado e ajustado conforme necessário para garantir que todas as informações relevantes e realistas estivessem presentes. Com base nas revisões, foram feitas novas solicitações para refinar e detalhar ainda mais o prontuário gerado.

- **Etapa 2: Geração dos Prontuários com Diferentes Diagnósticos**

Após a criação do modelo base, a segunda etapa consistiu na geração de prontuários sintéticos para uma variedade de diagnósticos específicos. Utilizando novamente o ChatGPT, foram gerados múltiplos prontuários para cada diagnóstico de interesse. Por exemplo, foram gerados 20 prontuários sintéticos para os seguintes diagnósticos: Infarto Agudo do Miocárdio,

Aneurisma de Aorta, Bradicardia, Angina, Mixoma Cardíaco, Insuficiência Cardíaca e Insuficiência Mitral. Este procedimento resultou em um total de 140 prontuários sintéticos.

O número de 20 prontuários gerados para cada condição foi definido para balancear a quantidade de prontuários reais disponíveis. Durante a geração, o ChatGPT utilizou informações contextuais para preencher os campos do prontuário de maneira coerente e detalhada, incluindo dados demográficos, sintomas, histórico médico, resultados de exames e planos de tratamento.

Criação dos Modelos de Classificação de Prontuários

Foram criados vários modelos de classificação utilizando diferentes algoritmos de aprendizado de máquina, incluindo Naive Bayes, Regressão Logística, Árvore de Decisão, Floresta Aleatória, K-Nearest Neighbors (KNN), Máquinas de Vetores de Suporte (SVM), Multilayer Perceptron (MLP) e Long Short-Term Memory (LSTM). Cada modelo foi treinado e avaliado utilizando as amostras reais e sintéticas de PME.

Resultados

Análise Estatística dos Diagnósticos dos Registros Eletrônicos

As ocorrências mais frequentes nos prontuários eletrônicos foram diagnosticadas com o CID I21 (Infarto Agudo do Miocárdio), com 50 amostras; CID I255 (Miocardiopatia Isquêmica), com 30 amostras; e CID I219 (Infarto Agudo do Miocárdio Não Especificado), com 11 amostras. A tabela a seguir resume a proporção dos diagnósticos:

Tabela 1 - Proporção dos Diagnósticos dos Prontuários Médicos

Diagnóstico	Quantidade	Proporção das amostras
Infarto Agudo do Miocardio (I21)	50	25%
Miocardia Isquêmica (I255)	30	15%
Infarto Agudo do Miocárdio Não Especificado (I219)	11	5,5%
Angina (I200)	9	4,5%
Insuficiência Cardíaca (I50)	9	4,5%
Bloqueio Atrioventricular Total (I442)	7	3,5%
Flutter e fibrilação atrial (I48)	5	2,5%
Estenose da valva aórtica (I350)	5	2,5%
Dor Torácica (R07)	4	2,0%
Bradicardia não especificada (R001)	3	1,5%
Septicemia (I419)	3	1,5%
Outros (<=2 casos)	64	32%

Fonte: Próprio Autor

Como observado, o Infarto Agudo do Miocárdio é o diagnóstico mais prevalente, seguido por Miocardiopatia Isquêmica. Esse resultado pode ser explicado pelo fato de as doenças cardíacas, como o infarto do miocárdio, representarem um problema de saúde pública global, com alta prevalência e impacto significativo na mortalidade.

Segundo a Sociedade Brasileira de Cardiologia (SBC), as doenças cardiovasculares são a principal causa de morte no Brasil, respondendo por cerca de 30% dos óbitos no país. Estima-se que 12 milhões de pessoas sofram infarto agudo do miocárdio a cada ano, resultando em cerca de 7 milhões de mortes. No Brasil, aproximadamente 300 mil pessoas sofrem infarto anualmente, com cerca de 200 mil mortes. Portanto, a classificação rápida e correta de pacientes com esse diagnóstico pode direcionar o atendimento da equipe médica para o paciente.

A prevalência de diagnósticos como Bloqueio Atrioventricular Total (I442) também foi observada. Esta condição médica envolve um bloqueio completo da condução dos impulsos elétricos do átrio para o ventrículo, resultando em ritmos cardíacos descoordenados, bradicardia, síncope e insuficiência cardíaca. Por isso é importante que essa condição seja rapidamente identificada e o paciente receba o atendimento de forma rápida.

A comparação dos desempenhos entre os modelos utilizados no estudo está detalhadamente apresentada na tabela 2. Esta tabela sintetiza os resultados obtidos de cada modelo em termos de acurácia, precisão, recall e F-Score, permitindo uma avaliação direta e quantitativa da eficácia de cada método na classificação de prontuários médicos eletrônicos.

Tabela 2 - Comparação dos desempenhos entre os modelos utilizados no estudo

Classificador	Acurácia	Precisão	Recall	F1-Score
Naive_Bayes	0,129	0,051	0,129	0,066
Regressão_Log	0,250	0,176	0,250	0,176
Árvore de Decisão	0,166	0,060	0,073	0,065
Floresta Aleatória	0,150	0,166	0,692	0,050
KNN	0,150	0,158	0,391	0,069
SVM	0,233	0,478	0,424	0,167
MLP	0,200	0,219	0,462	0,115
LSTM	0,833	0,834	0,467	0,671
LSTM + Amostras Sintéticas	0,934	0,976	0,714	0,835

Fonte: Próprio Autor

Discussão

Ao comparar as métricas dos modelos para avaliar seu desempenho na tarefa de classificar os prontuários médicos eletrônicos, diversos pontos importantes foram observados. O modelo Naive Bayes apresentou uma acurácia relativamente baixa e precisões ainda menores, indicando que ele está classificando muitos exemplos negativos como positivos. Esse baixo desempenho pode ser atribuído à suposição de independência entre as características, uma premissa que nem sempre é verdadeira na prática e que pode levar a erros na classificação. Em contraste, a Regressão Logística apresentou uma melhora significativa em relação ao Naive Bayes, com melhores valores de precisão e abrangência. Esse modelo conseguiu classificar os exemplos com mais precisão, devido à sua capacidade de modelar a relação entre as variáveis preditoras e a variável de resposta de maneira mais flexível e robusta.

A Árvore de Decisão teve uma melhor precisão e abrangência do que o Naive Bayes, embora ainda apresentasse desempenho inferior em comparação a outros modelos. A capacidade da Árvore de Decisão de capturar regras complexas nas características permitiu uma classificação mais precisa, mas o modelo pode sofrer de overfitting, especialmente com um conjunto de dados pequeno. O modelo KNN apresentou acurácia e precisão relativamente altas, mas com um F-Score menor do que o da Árvore de Decisão. Isso indica que, embora o KNN seja bom em identificar os exemplos corretamente, ele perde alguns exemplos positivos, possivelmente devido à sua sensibilidade a ruídos nos dados e à necessidade de um número adequado de vizinhos (k) para otimização.

O modelo SVM mostrou-se robusto, com alta precisão e abrangência, sendo eficaz na classificação de textos médicos. O desempenho superior do SVM pode ser atribuído à sua capacidade de encontrar um hiperplano ótimo que separa as classes, mesmo em espaços de alta dimensão, o que é crucial para a classificação de dados textuais complexos. Por outro lado, o MLP apresentou acurácia e precisão baixas, indicando dificuldades em classificar corretamente os exemplos. O baixo desempenho do MLP pode ser devido à complexidade do modelo e à necessidade de um conjunto de dados maior e mais diversificado para treinamento eficaz.

O modelo LSTM destacou-se por obter a melhor acurácia, precisão e F-Score, tornando-se o modelo mais eficaz para a tarefa proposta. A arquitetura LSTM é especialmente adequada para a análise de sequências de texto, pois pode capturar dependências de longo prazo e padrões complexos nos dados. Quando combinada com amostras sintéticas, a LSTM apresentou um desempenho ainda melhor, com um aumento significativo no F-Score. A introdução de dados sintéticos foi fundamental para mitigar o problema de overfitting e melhorar a capacidade do modelo de generalizar para novos exemplos. Este resultado destaca a importância de utilizar dados sintéticos para complementar conjuntos de dados reais, especialmente em contextos onde a disponibilidade de dados é limitada.

Os prontuários sintéticos gerados através do ChatGPT desempenharam um papel crucial na melhoria do desempenho dos modelos de classificação. A criação de dados adicionais permitiu expandir o conjunto de dados original, proporcionando mais exemplos para o treinamento dos modelos. Isso foi especialmente útil para aumentar a robustez dos modelos e melhorar sua capacidade de generalização. A utilização de dados sintéticos demonstrou ser uma estratégia eficaz para lidar com a escassez de dados reais e melhorar a performance dos modelos de aprendizado de máquina em tarefas complexas, como a classificação de textos médicos.

Este estudo reforça a relevância de técnicas de inteligência artificial generativa na área de saúde, contribuindo para avanços significativos no processamento e análise de grandes volumes de dados clínicos.

Limitações da Pesquisa

Este estudo enfrentou algumas limitações técnicas e de escopo que restringiram a quantidade de modelos avaliados para a classificação de Prontuários Médicos Eletrônicos (PME).

Embora tenha sido possível demonstrar a capacidade de utilizar modelos de inteligência artificial para essa finalidade, a seleção dos modelos foi limitada.

Os modelos escolhidos foram cuidadosamente orientados pela proposta do trabalho, que visa investigar e avaliar técnicas aplicáveis em um contexto real de classificação automatizada de PME no futuro. Essas limitações, no entanto, não comprometem a relevância dos resultados obtidos, que fornecem uma base sólida para a continuidade dessa linha de pesquisa, destacando o potencial das abordagens de inteligência artificial na interpretação e análise de textos complexos encontrados nos PME.

Estudos Adicionais

Para futuras pesquisas, recomenda-se expandir a avaliação para incluir uma maior variedade de algoritmos de aprendizado de máquina e técnicas de Processamento de Linguagem Natural (PLN). Estudos adicionais poderiam explorar o impacto de diferentes métodos de pré-processamento de texto, a utilização de modelos mais avançados como transformers, e a integração de dados multimodais que incluam imagens e sinais vitais dos pacientes.

Além disso, seria benéfico aumentar o tamanho e a diversidade das amostras de PME para melhorar a generalização dos modelos. A implementação de estudos longitudinais para avaliar a eficácia dos modelos em cenários clínicos reais ao longo do tempo também é crucial.

Conclusão

Este estudo alcançou todos os objetivos propostos, desenvolvendo modelos de aprendizado de máquina supervisionado para a classificação automática de textos de prontuários eletrônicos de pacientes cardiológicos. A revisão da literatura identificou algoritmos adequados, e uma amostra representativa de Prontuários Médicos Eletrônicos (PME) foi coletada e pré-processada usando técnicas de Processamento de Linguagem Natural (PLN).

Os modelos treinados demonstraram boa performance na classificação de diagnósticos. A pesquisa demonstrou que algoritmos de aprendizado de máquina são eficazes para a classificação automática de dados de PME, ressaltando o papel crucial das tecnologias da informação na melhoria da identificação de condições de saúde, previsão de resultados e personalização de tratamentos, contribuindo para avanços na gestão de dados de saúde e qualidade do atendimento ao paciente.

Referências

1. Alberto KG, Abella CM, Sicat MGCE, Niguidula JD, Caballero JM. Compiling Remote Files: Redefining Electronic Document Management System Infrastructure (CReED). In 2009 International Conference on Information and Multimedia Technology. 2009. p. 347-350.
2. Singru R, Bhandari P, Patel K, Mane P, Gulhane C. Efficient Electronic Document Access Control Management using Natural Language Processing. In 2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC). 2020. p. 714-719.
3. Rahman, R., Kodesh, A., Levine, S. Z., Sandin, S., Reichenberg, A., & Schlessinger, A. (2020). Identification of newborns at risk for autism using electronic medical records and machine learning. European Psychiatry: The Journal of the Association of European Psychiatrists, 63(1), e22. https://doi.org/10.1192/j.eurpsy.2020.17
4. Shickel, B., Tighe, P., Bihorac, A., & Rashidi, P. (2018). Deep EHR: A survey of recent advances in deep learning techniques for electronic health record (EHR) analysis. IEEE Journal of Biomedical and Health Informatics, 22(5), 1589-1604.
5. Madahian, B., & Faghihi, U. (2014). A fully Bayesian sparse probit model for text categorization. Open Journal of Statistics, 4, 611-619.
6. Webster, J., & Kit, C. (1992). Tokenization as the initial phase in NLP. In Proceedings of the 14th conference on computational linguistics, 1106-1110. [https://doi.org/10.3115/992424.992434] (notice the inclusion of DOI and conference title)
7. Arumugam, R., & Shanmugamani, R. (2018). Hands-On Natural Language Processing with Python: A practical guide to applying deep learning architectures to your NLP applications (1ª ed.). Reino Unido.
8. Khyani, D., & Siddhartha, B. S. (2021). An Interpretation of Lemmatization and Stemming in Natural Language Processing. Shanghai Ligong Daxue Xuebao/Journal of University of Shanghai for Science and Technology, 22, 350-357.
9. Hacohen, Y., Miller, D., & Yigal, Y. (2020). The influence of preprocessing on text classification using a bag-of-words representation. PloS one, 15(5), e0232525.
10. Ikonomakis, M., Kotsiantis, S., & Tampakas, V. (2005). Text Classification Using Machine Learning Techniques. WSEAS TRANSACTIONS on COMPUTERS, 4(8), 966-974.

Summary: The automatic classification of Electronic Medical Records (EMR) using supervised machine learning is a solution capable of dealing with the volume and complexity of these documents. Real texts from cardiology PMEs and synthetic texts created by ChatGPT were used to train and evaluate several classification algorithms. Logistic regression achieved the highest overall accuracy and F1 score. The synthetic records generated by ChatGPT improved the performance of LSTM, achieving the highest accuracy (0.93) and precision (0.97). These findings suggest the use of machine learning for reliable automated classification, potentially assisting cardiologists in decision-making after further validation.

Supervised Machine Learning; Electronic Health Records; Cardiology; Natural Language Processing; Decision-making

Resumen: La clasificación automática de Historias Médicas Electrónicas (HCE) mediante aprendizaje automático supervisado es una solución capaz de hacer frente al volumen y complejidad de estos documentos. Se utilizaron textos reales de PME de cardiología y textos sintéticos creados por ChatGPT para entrenar y evaluar varios algoritmos de clasificación. La regresión logística logró la precisión general y la puntuación F1 más altas. Los registros sintéticos generados por ChatGPT mejoraron el rendimiento de LSTM, logrando la mayor exactitud (0,93) y precisión (0,97). Estos hallazgos sugieren el uso del aprendizaje automático para una clasificación automatizada confiable, lo que podría ayudar a los cardiólogos en la toma de decisiones después de una validación adicional.

Aprendizaje automático supervisado; Registros Médicos Electrónicos; Cardiología; Procesamiento natural del lenguaje; Toma de decisiones