UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

PITER OLIVEIRA VERGARA

Desenvolvimento e Avaliação de Algoritmos de Classificação e Decisão na Regulação de Pacientes Encaminhados para a Atenção Especializada Ambulatorial

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação

Orientador: Prof. Dr. Leandro Krug Wives

Porto Alegre
2020

CIP — CATALOGAÇÃO NA PUBLICAÇÃO

Vergara, Piter Oliveira

Desenvolvimento e Avaliação de Algoritmos de Classificação e Decisão na Regulação de Pacientes Encaminhados para a Atenção Especializada Ambulatorial / Piter Oliveira Vergara. – Porto Alegre: PPGC da UFRGS, 2020.

125 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2020. Orientador: Leandro Krug Wives.

1. Classificação de Textos. 2. Aprendizado de Máquina. 3. Processamento de Linguagem Natural. 4. Regulação de Pacientes. 5. TelessaúdeRS. I. Krug Wives, Leandro. II. Título.

AGRADECIMENTOS

A conclusão deste trabalho é uma grande conquista pessoal, mas não seria atingida sem a participação e o apoio de muitos além de mim.

Gostaria de agradecer à minha família, meus pais, irmãos e sobrinhos pelo incentivo e pela tolerância das repetidas ausências cuja justificativa foi este trabalho. De igual forma, agradeço aos meus amigos próximos e à minha namorada pelo mesmo apoio e compreensão.

Agradeço também aos professores do Instituto de Informática da UFRGS cujas as aulas tive a oportunidade de assistir e tenho a satisfação de lembrar que sempre acertei a escolha das cadeiras que cursei, pois todas me ajudaram no desenvolvimento deste trabalho. De igual forma, agradeço aos colegas do INF com quem interagi, que sempre mostraram um notável espírito de colaboração, do qual tantas vezes me beneficiei, em especial aos colegas do meu grupo de pesquisa que não só contribuíram com ideias como também com a revisaão de meus textos.

Minha participação neste programa de mestrado também não teria sido possível sem a manifestação de interesse da administração do Ministério Público do RS, instituição na qual trabalhava quando ingressei no programa, ou sem o incentivo de meu coordenador à época, que facilitou de diversas formas meu comparecimento às aulas. Pelos mesmos motivos, agradeço também meu atual empregador, a Justiça Federal do RS, por viabilizar a continuidade minha participação no programa.

Agradeço ao TelessaúdeRS pela oportunidade de trabalhar num projeto de tamanha relevância social e por me permitir contribuir com a procura de uma solução prática para um problema tão importante. Em especial, agradeço ao Dr. Dimitris Rucks Varvaki Rados, principal parceiro e especialista médico do projeto, que conduziu todos os aspectos deste junto ao Telessaúde e às outras instituições parceiras, bem como ao Dr. Rudi Roman que junto com o Dr. Dimitris auxiliou enormemente na condução e direcionamento dos experimentos do projeto.

Agradeço muito ao meu orientador pelo voto de confiança ao me aceitar como orientando e por ter sido meu fiador junto ao TelessaúdeRS, ao me apresentar ao projeto, bem como por me conduzir, com grande paciência, neste caminho.

Por fim, agradeço ao Instituto de Informática da UFRGS e à CAPES por terem oportunizando minha participação neste programa de pós-graduação de reconhecida excelência.

RESUMO

O desequilíbrio entre oferta e demanda por serviços especializados no Sistema Único de Saúde brasileiro implica que nem todas as solicitações podem ser atendidas na medida em que chegam ao sistema. No estado do Rio Grande do Sul, uma abordagem usada para lidar com as filas de solicitações que se formam é a regulação realizada pelo projeto TelessaúdeRS, em que profissionais chamados Reguladores avaliam as solicitações e autorizam ou não a consulta. Essa atividade de Regulação é bastante onerosa, fazendo com que haja sempre um grande número de solicitações pendentes de avaliação. Nesse contexto, percebe-se a necessidade de melhorar a eficiência do processo regulatório dos encaminhamentos, auxiliando os médicos Reguladores através da ferramentas que otimizem o tempo dedicado ao processo. Com vistas a atender essa necessidade, neste trabalho são avaliadas diferentes alternativas para o desenvolvimento de um Classificador de Textos baseado em Aprendizado de Máquina (ML) capaz de auxiliar na tarefa de Regulação. Trabalha-se com os textos dos encaminhamentos regulados pelo TelessaúdeRS no período de jun/2016 até abr/2019 para as especialidades de Urologia, Reumatologia, Gastro-enterologia, Endocrinologia e Proctologia. Diferentes combinações de tarefas de pré-processamento e engenharia de *features* foram realizadas, diferentes representações do *corpus* foram criadas e cinco diferentes algoritmos de ML foram experimentados para a classificação, reportando os resultados em função da ROC AUC. O melhor resultado foi obtido com uma Rede Neural Recorrente e textos representados via *word embeddings*. Com esse algoritmo, obteve-se uma ROC AUC de 0.83 para o todo o conjunto de *holdout* (sendo ela de 0.81 para Endocrinologia, 0.77 para Gastro-enterologia, $0,81$ para Proctologia, 0.85 para Reumatologia e 0.84 para Urologia). O algoritmo proposto foi também comparado ao método atual de Regulação, usando um novo conjunto de dados, e verificou-se resultados próximos aos obtidos pelo método atual de regulação, o que é evidenciado pela Medida F1 de $0,57$ do método proposto, ligeiramente superior aos $0,54$ do método atual. Por fim, um componente funcional de software foi desenvolvido para encapsular o algoritmo em um *web service* permitindo o reúso da solução e a continuidade da pesquisa.

Palavras-chave: Classificação de Textos. Aprendizado de Máquina. Processamento de Linguagem Natural. Regulação de Pacientes. TelessaúdeRS.

Development and evaluation of classification and decision algorithms in the regulatory process of patients referrals to ambulatory specialized attention

ABSTRACT

The imbalance between supply and demand for specialized services in the Brazilian Unified Health System implies that not all requests can be met as they reach the system. In the state of Rio Grande do Sul, an approach used to deal with the request queues is the regulation carried out by the TelessaúdeRS project, in which professionals called Regulators evaluate the requests and authorize or not the consultation. This regulation activity is very costly, resulting in a permanent number of requests to be evaluated. In this context, there is a need to improve the efficiency of the referrals regulatory process, assisting Regulatory Physicians through tools that optimize the time dedicated to the process. In order to meet this need, in this work, different alternatives for the development of a Text Classifier based on Machine Learning (ML) capable of assisting in the Regulation task are evaluated. The work utilizes texts from the referrals regulated by TelessaúdeRS from Jun/2016 to Apr/2019 for the specialties of Urology, Rheumatology, Gastroenterology, Endocrinology, and Proctology. Different combinations of preprocessing and feature engineering tasks are performed, different representations of the corpus are created, and five different ML algorithms are experimented, reporting the results in terms of ROC AUC. The best result is obtained with a recurrent neural network and texts represented via word embeddings. This algorithm obtains a ROC AUC of 0.83 for the entire holdout set (0.81 for endocrinology, 0.77 for gastroenterology, 0.81 for proctology, 0.85 for rheumatology and 0.84 for urology). A comparison between the presented algorithm and the current Regulation method was carried out using a new data set and found results close to those obtained by the current regulation method, which is evidenced by the F1-Measure of 0.57 of the propoused method, slightly higher than 0.54 of the current method. Finally, a software component that encapsulates the propused algorithm in a web service is developed, aiming the reused of the solution and the continuity of the research.

Keywords: Text Classification, Machine Learning, Natural Language Processing, Patient Referrals Regulation, TelessaúdeRS.

LISTA DE FIGURAS

Figura 1.1 Como funciona o RegulaSUS 15
Figura 1.2 Contribuição geral deste trabalho 17

Figura 2.1 Classificação de aplicações de tecnologia em Medicina 19
Figura 2.2 Classificação de Textos usando Aprendizado de Máquina Supervisionado .. 24
Figura 2.3 Arquiteturas CBOW e SkipGram 32
Figura 2.4 Exemplo de RNA multicamadas típica 45
Figura 2.5 Célula de redes LSTM e GRU 48
Figura 2.6 Exemplo de Curvas ROC 53

Figura 4.1 As três fases desenvolvidas para realização deste trabalho 65

Figura 5.1 Visão geral das etapas dos experimentos 66
Figura 5.2 Quantidade de casos, por especialidade agrupada 68
Figura 5.3 Quantidade de casos aprovados ou não, por especialidade agrupada 69
Figura 5.4 Quantidade de casos aprovados ou não, por especialidade agrupada, nos *datasets* de Teste e Treino 70
Figura 5.5 Uso do CTAKES, via CAS Visual Debugger (CVD), para identificar conceitos presentes no texto de exemplo 73
Figura 5.6 Exemplo de metadatados obtidos do UMLS com o nosso método 76
Figura 5.7 Técnicas de vetorização, representações vetoriais e as respectivas variações exploradas 79
Figura 5.8 Técnicas de Seleção e Transformação de *Features* e as respectivas variações exploradas 80
Figura 5.9 Técnicas de Resampling utilizadas nos experimentos 80
Figura 5.10 ROC AUC média para um mesmo conjunto de dados e classificador, com diferentes técnicas de *resampling* 81
Figura 5.11 Modelos de classificação testados 82
Figura 5.12 Uma visão simplificada de um dos possíveis fluxos de experimento 83
Figura 5.13 Fluxo com apenas pré-processamento e vetorização dos dados 84
Figura 5.14 Fluxo 1 - Avaliado pela ROC AUC média (validação cruzada com 10 folds) 84
Figura 5.15 Melhores resultados para o Fluxo 1 85
Figura 5.16 Visão simplificada do fluxo do experimento incluindo *resampling* dos dados 86
Figura 5.17 Fluxo com pré-processamento, vetorização e resampling dos dados 87
Figura 5.18 Melhores resultados para o Fluxo 2 87
Figura 5.19 Fluxo de experimento que incluiu transformação e seleção de *features* 88
Figura 5.20 Resultados da classificação combinando diferentes pré-processamentos e técnicas de seleção de *features*) 88
Figura 5.21 Melhores resultados para o Fluxo 3 90
Figura 5.22 Melhor configuração do Fluxo 3, que usa SVM Linear 91
Figura 5.23 Resultados para o melhor classificador do Fluxo 3 quando aplicado aos dados de *Holdout* 91
Figura 5.24 Resultados do melhor classificador do Fluxo 3 por especialidade (*Holdout*) 92
Figura 5.25 Fluxo de experimento utilizando Tipos Semânticos extraídos do UMLS em um classificador *Decision Tree* 93
Figura 5.26 Detalhes da geração do vetor utilizado no Fluxo 4 93

Figura 5.27 Curvas ROC obtidas pela classificação usando a contagem de ocorrência dos tipos semânticos em diferentes configurações de DecisionTrees 94
Figura 5.28 Curvas ROC obtidas pela classificação usando a contagem de ocorrência dos tipos semânticos em diferentes configurações de AdaBoost 94
Figura 5.29 Fluxo de experimento utilizando AdaBoost, sem e com resampling 95
Figura 5.30 Resultados de experimento utilizando AdaBoost sem resampling.....95
Figura 5.31 Resultados de experimento utilizando AdaBoost com resampling 95
Figura 5.32 Fluxo de experimento utilizando *Random Forests* 96
Figura 5.33 Resultados de experimento utilizando *Random Forests* 97
Figura 5.34 Resultados do classificador do Fluxo 6 aplicado aos dados de *Holdout* 97
Figura 5.35 Fluxo de experimento utilizando *Word Enbeddings* e Rede Neural 98
Figura 5.36 Topologia da rede usada na avaliação dos *Word embeddings*.....99
Figura 5.37 Resultados da aplicação dos *embeddings* de DOS SANTOS et al. (2018) .99
Figura 5.38 Resultados da aplicação dos *embeddings* de DOS SANTOS et al. (2018) após estendê-lo com dados deste trabalho 100
Figura 5.39 Melhores resultados para os *embeddings* criados apenas com textos deste trabalho 101
Figura 5.40 Resultados da variação dos meta-parâmetros da rede 103
Figura 5.41 Resultados da variação da Topologia da Rede.....105
Figura 5.42 Resultados do novo treinamento da melhor topologia (Topologia A).....105
Figura 5.43 Resultados do melhor classificador do Fluxo 7 aplicado aos dados de *Holdout* 105
Figura 5.44 Resultados do melhor classificador do Fluxo 7, por especialidade (Holdout) 106

Figura 6.1 Medidas de comparação entre classificações do método Experimental e Atual para as três filas na Fase 3 111
Figura 6.2 Exemplo de cliente para uso do software enviando dados em uma planilha113
Figura 6.3 Exemplo de cliente para uso do *software* enviando dados como *JSON* 114

Figura A.1 Matrizes de confusão comparando o melhor algoritmo de regulação experimental ao método atual de regulação para os dados da Fase 3 122

LISTA DE TABELAS

Tabela 1.1 Novos casos para regulação pelo TelessaudeRS entre jan/2019 e set/2019, nas especialidades consideradas neste trabalho 15

Tabela 2.1 Comparação entre os processos de *stemming* e lematização 27
Tabela 2.2 Exemplos de textos representados nas tabelas seguintes 29
Tabela 2.3 Representação *Bag-of-Words* usando a contagem de termos 29
Tabela 2.4 Representação *Bag-of-Words* usando TFiDf 29
Tabela 2.5 Bag-of-ngrams, com n-gramas de 1 e 2 termos, usando contagem de termos 29

Tabela 3.1 Comparativo dos trabalhos relacionados 61

Tabela 5.1 Especialidades agrupadas e as suas quantidades de encaminhamentos 68
Tabela 5.2 Os pré-processadores e suas respectivas tarefas 70
Tabela 5.3 Exemplos do resultado de cada pré-processador 71
Tabela 5.4 Categorias gramaticais identificadas no texto de exemplo pela aplicação do *POS-Tagger* utilizado neste trabalho 74
Tabela 5.5 Textos retornados na saída, pelo script que obtém os meta-dados do UMLS 75
Tabela 5.6 Apresenta os tipos semânticos do UMLS identificados no texto de exemplo através do uso do Apache CTAKES e do método próprio 77
Tabela 5.7 Matriz com a contagem de ocorrência dos tipos semânticos identificados no texto de exemplo, considerando-se identificações com 1+ tokens 78
Tabela 5.8 Comparativo dos melhores resultados para o Fluxo 1 85
Tabela 5.9 Comparativo dos melhores resultados para o Fluxo 2 86
Tabela 5.10 Comparativo dos melhores resultados para o Fluxo 3 90
Tabela 5.11 Detalha as diferentes topologias testadas na rede neural 104

Tabela 6.1 Medidas de comparação entre classificações tendo como métodos o Melhor Algoritmo Experimental (Exp.) e a classificação por reguladores (Atual) para as filas de Procto, Reumato e Uro na Fase 3 111

Tabela B.1 Métricas para a classificação dada pelo algoritmo para os 1049 casos utilizados na fase 3 (Capítulo 6) ao configurar o algoritmo com diferentes limiares 123

LISTA DE ABREVIATURAS E SIGLAS

ADR	*Adverse Drug Reaction*
APS	Atenção Primária à Saúde
AUC	*Area Under ROC Curve*
CBOW	*Continuous Bag-of-Words*
CCI	*Charlson Comorbidity Index*
CDS	*Clinical Decision Support*
CID	Classificação Internacional de Doenças
CUI	*Concept Unique Identifier*
CVD	*CAS Visual Debugger*
GRU	*Gated Reccurent Unit*
HCPA	Hospital de Clínicas de Porto Alegre
HNSC	Hospital Nossa Senhora da Conceição
IA	Inteligência Artificial
IE	*Information Extraction*
IM	Informação Mútua (IM)
JSON	*Javascript Object Notation*
KNN	*K-Nearest Neighbors*
LSA	*Latent Semantic Analysis*
LSTM	*Long Short-Term Memory*
MeSH	*Medical Subject Headings*
ML	*Machine Learning*
NLM	*National Library of Medicine*
NTLK	*Natural Language Toolkit*
PCA	*Principal Component Analysis*

PLN	Processamento de Linguagem Natural
RAS	Redes de Atenção à Saúde
RBF	*Radial Basis Function*
RNA	Redes Neurais Artificiais
RNN	*Reccurent Neural Networks*
ROC	*Receiving Operating Characteristics*
RSLP	Removedor de Sufixos da Língua Portuguesa
SMOTE	*Synthetic Minority Oversampling Technique*
SNOMED	*Systematized Nomenclature of Human and Veterinary Medicine*
SQL	*Structured Query Language*
SUS	Sistema Único de Saúde brasileiro
SVD	*Singular Value Decomposition*
SVM	*Support Vector Machines*
TFiDF	*Term Frequency-Inverse Document Frequency*
TFP	Taxa de Falsos Positivos
TVP	Taxa deVerdadeiros Positivos
UMLS	*Unified Medical Language System*
VSM	*Vector Space Model*

SUMÁRIO

1 INTRODUÇÃO 13
2 FUNDAMENTAÇÃO TEÓRICA 18
2.1 Informática Médica 18
2.1.1 Sistema Unificado de Linguagem Médica 19
2.1.2 Apache *clinical Text Analysis and Knowledge Extraction System* 20
2.2 Aprendizado de Máquina 21
2.3 Classificação de Textos 23
2.4 Pré-processamento de Textos 25
2.4.1 Remoção de *stop-words* 25
2.4.2 *Stemming* e Lematização 26
2.5 Representação de Textos 27
2.5.1 *Bag-of-Words* e *Bag-of-ngrams* 27
2.5.2 *Word Embeddigns* 29
2.5.2.1 Word2vec e FastText 30
2.5.2.2 CBOW e SkipGram 31
2.6 Extração de características 32
2.6.1 Ganho de Informação 35
2.6.2 Informação Mútua 36
2.6.3 *Chi-square* 36
2.6.4 Análise Semântica Latente 36
2.6.5 *Quantile Transformer* 37
2.6.6 Select K-best 38
2.7 Balanceamento de Classes 39
2.7.1 *Synthetic Minority Over-sampling Technique* 39
2.7.2 *Borderline-SMOTE* 40
2.7.3 *K*-means SMOTE 40
2.8 Algoritmos de Aprendizado de Máquina 41
2.8.1 Máquinas de Vetor de Suporte 41
2.8.2 Árvores de Decisão 42
2.8.3 Modelos Múltiplos Preditivos 43
2.8.3.1 *Adaptative Boosting - AdaBoost* 44
2.8.3.2 Florestas Aleatórias 44
2.8.4 Redes Neurais Artificiais 45
2.8.5 Redes Neurais Recorrentes 47
2.8.6 Gated Reccurent Unit 47
2.9 Otimização de hiperparâmetros 49
2.10 Avaliação de Modelos de Aprendizado de Máquina 50
2.10.1 Matriz de Confusão 51
2.10.2 Curvas ROC 52
2.11 Resumo do Capítulo 54
3 TRABALHOS RELACIONADOS 56
3.1 Resumo do Capítulo 60
4 ABORDAGEM PROPOSTA 62
4.1 Fase 1 - Desenvolvimento incremental de algoritmos de Classificação 62
4.2 Fase 2 - Teste de desempenho dos algoritmos de Classificação 63
4.3 Fase 3 - Avaliação da eficácia do algoritmo de Classificação de Textos na regulação ambulatorial 64
4.4 Resumo do Capítulo 65

5 VALIDAÇÃO EXPERIMENTAL....66
5.1 Conjunto de Dados....67
5.2 Holdout....69
5.3 Pré-processamento....69
5.4 Enriquecimento dos textos....72
5.5 Vetorização e representação dos vetores....77
5.6 Seleção e Transformação de Features....79
5.7 Resampling....80
5.8 Treinamento e Avaliação dos Modelos....81
5.8.1 Diferentes Fluxos....83
5.8.2 Fluxo 1....84
5.8.3 Fluxo 2....86
5.8.4 Fluxo 3....88
5.8.5 Fluxo 4....93
5.8.6 Fluxo 5....95
5.8.7 Fluxo 6....96
5.8.8 Fluxo 7....98
5.9 Resumo do Capítulo....107
6 AVALIAÇÃO DA EFICÁCIA DO ALGORITMO....110
6.1 Resultados....110
6.2 Entregável....112
6.3 Resumo do Capítulo....113
7 CONCLUSÃO....115
REFERÊNCIAS....117
APÊNDICEA FASE 3 MÉTODOS EXPERIMENTAL *VS* MÉTODO ATUAL....122
APÊNDICEB MÉTRICAS DO ALGORITMO EM DIFERENTES LIMIARES 123

1 INTRODUÇÃO

O Sistema Único de Saúde brasileiro (SUS) é um sistema orientado para a Atenção Primária à Saúde (APS). Nos últimos anos houve um crescimento da oferta de atendimentos na APS, alavancado por diferentes estímulos para o aumento do número de profissionais e de unidades de atendimento. Essa expansão foi benéfica por ampliar o acesso à saúde básica, mas acentuou o problema da baixa resolutividade existente nesse nível de atendimento. No mesmo período, os serviços ambulatoriais especializados - aos quais são encaminhados os casos não resolvidos na APS - cresceram pouco, gerando desequilíbrio entre demanda e oferta desses serviços.

Esse desequilíbrio entre oferta e demanda por serviços especializados no SUS causa problemas tanto para os pacientes que não deveriam ser encaminhados a especialistas quanto para aqueles corretamente encaminhados. Durante o período de espera, o quadro clínico daqueles que poderiam ter sido tratados pela APS tende a agravar-se forçando-os a procurar serviços de urgência e emergência, contribuindo inclusive para a superlotação das unidades de pronto atendimento. Enquanto isso, aqueles que de fato precisam de encaminhamento têm seu atendimento postergado devido à alocação inadequada dos recursos especializados e hospitalares, usados desnecessariamente para o atendimento de demandas que deveriam ser tratadas pela Atenção Primária.

Nesse contexto está inserida a Telessaúde, que segundo SUS (2020) é um componente da Estratégia e-Saúde (Saúde Digital) para o Brasil, a qual tem como finalidade a expansão e melhoria da rede de serviços de saúde, sobretudo da Atenção Primária à Saúde (APS), e sua interação com os demais níveis de atenção fortalecendo as Redes de Atenção à Saúde (RAS) do SUS.

No Rio Grande do Sul, segundo TelessaudeRS (2020), o TelessaúdeRS conta com uma equipe de profissionais qualificados e com reconhecida produção científica em suas áreas de atuação e seus serviços auxiliam e qualificam o atendimento que os profissionais da saúde prestam à população, evitando, na maior parte dos casos, o deslocamento dos pacientes para fora dos seus municípios.

Ainda conforme TelessaudeRS (2020), o projeto atua em três frentes: Telediagnósitico, Teleeducação e Teleconsultoria. O Telediagnóstico é um serviço que utiliza tecnologia da informação e da comunicação para apoio a diagnósticos a distância, por exemplo, através de fotos enviadas via plataforma online, na qual os exames são avaliados por uma equipe especializada. A Teleducação fornece diversas ações a distância

de educação continuada para profissionais de saúde da APS e estudantes de graduação das áreas da saúde, tais como cursos, aplicativos, webpalestras e materiais educativos. A Teleconsultoria oferece aos profissionais da APS um serviço de 0800 para discussão de casos e dúvidas sobre a sua atuação no SUS. Outra maneira de obter consultoria é utilizando a Plataforma de Telessaúde do Ministério da Saúde. Nesse caso, um profissional da APS, dito "solicitante", pode registrar um questionamento que é então classificado, utilizando terminologia médica específica, por um "teleregulador" que em seguida encaminha o questionamento ao profissional mais adequado a respondê-lo, o "teleconsultor". Esses mecanismos de auxílio permitem definir, com base nas melhores evidências científicas, a conduta para os casos em discussão, evitando assim encaminhamentos desnecessários, o que melhora a experiência de quem usa o SUS e otimiza o uso de recurso públicos.

Além do 0800, a Teleconsultoria inclui o RegulaSUS, projeto desenvolvido em conjunto com a Secretaria Estadual de Saúde do RS, que regula a fila de pacientes da APS para consultas com médicos de 14 diferentes especialidades em Porto Alegre. O objetivo é diminuir o tempo de espera para consulta com um especialista, priorizar o atendimento para os casos mais graves e resolver boa parte dos problemas de saúde das pessoas em seu próprio município de residência.

No processo de Regulação, os casos encaminhados da Atenção Primária para a Atenção Especializada são avaliados pelos profissionais do TelessaudeRS, chamados Reguladores. Com base em Protocolos de Regulação, os médicos Reguladores podem solicitar mais informações ou recomendar um teleconsultoria via 0800, nesse caso Não Aprovando o encaminhamento ao especialista, ou autorizar a consulta, Aprovando o encaminhamento. A Figura 1.1 ilustra o processo completo da Regulação.

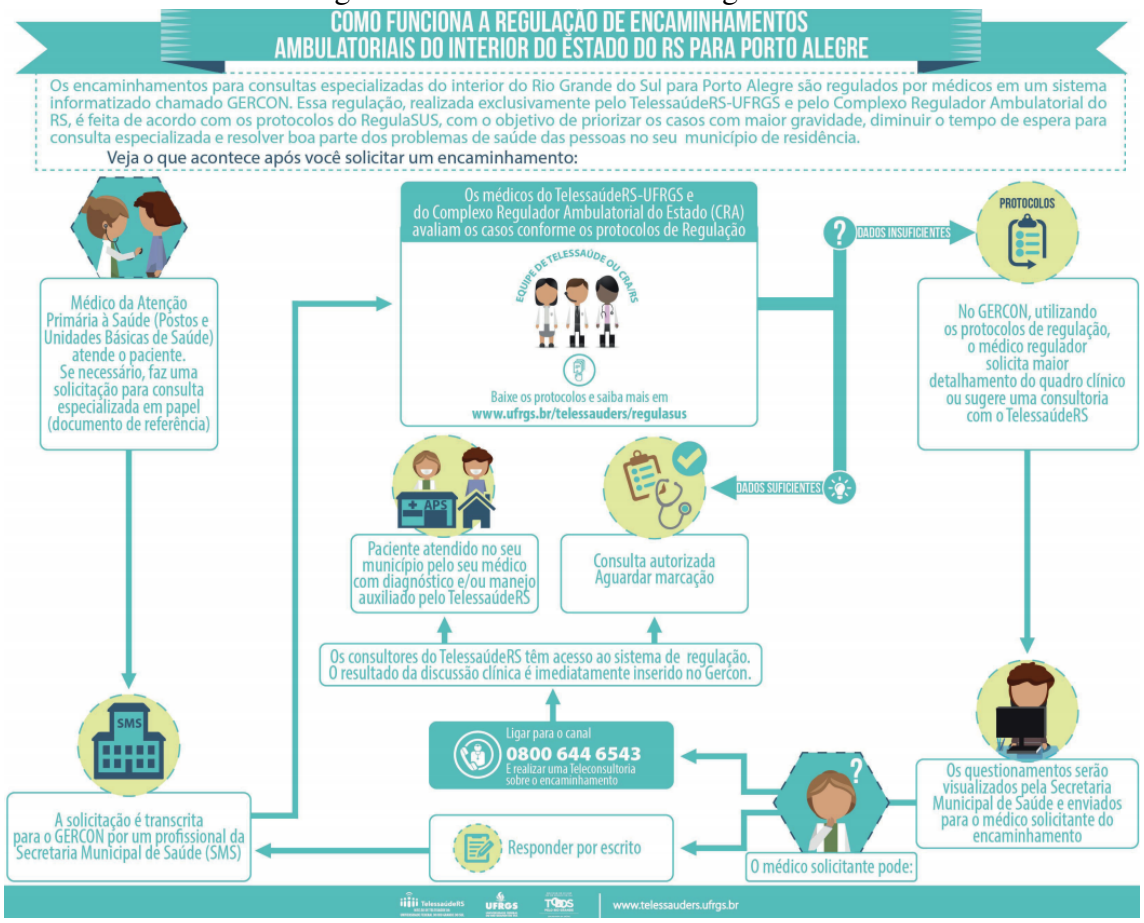
As atividades de regulação, contudo, são onerosas em termos de recursos humanos e a demanda supera a capacidade de resposta da equipe de telerregulação.

A Tabela 1.1 mostra a quantidade de novos casos entre janeiro e setembro de 2019 paras as filas consideradas neste trabalho e nela é possível notar a grande demanda de regulação.

Nota-se, portanto, a necessidade de melhorar a eficiência do processo regulatório dos encaminhamentos, auxiliando os médicos reguladores através da ferramentas que otimizem o tempo dedicado ao processo. Assim, considera-se que aqui se apresenta uma oportunidade para automação de tarefas do processo de regulação no TelessaúdeRS.

Um dos principais desafios para essa automação é a forma desestruturada das informações usadas no julgamento dos encaminhamentos, as quais são, em grande parte,

Figura 1.1: Como funciona o RegulaSUS



Fonte: TelessaudeRS (2019)

Tabela 1.1: Novos casos para regulação pelo TelessaudeRS entre jan/2019 e set/2019, nas especialidades consideradas neste trabalho

Origem	Especialidade	jan/19	fev/19	mar/19	abr/19	mai/19	jun/19	jul/19	ago/19	set/19
Interior	ENDOCRINOLOGIA ADULTO	231	287	263	344	374	262	446	538	355
	ENDOCRINOLOGIA TIREOIDE	66	83	86	90	90	51	124	79	75
	GASTROENTEROLOGIA ADULTO	452	457	468	512	549	489	478	541	415
	GASTROENTEROLOGIA HEPATITES VIRAIS ADULTO	25	38	22	41	44	32	37	46	28
	GASTROENTEROLOGIA-DOENÇA INFLAMATÓRIA INTESTINAL	9	16	13	9	7	7	6	6	6
	PROCTO PROCTOLOGIA ADULTO	281	266	260	350	331	294	304	293	282
	REUMATO REUMATOLOGIA ADULTO	288	275	248	304	287	289	353	348	331
	UROLOGIA ADULTO	535	568	586	669	636	536	669	610	605
	UROLOGIA LITÍASE RENAL	0	0	0	6	0	0	0	0	0
	UROLOGIA VASECTOMIA	0	0	0	0	0	0	0	0	0
	Total Interior	**1887**	**1990**	**1946**	**2325**	**2318**	**1960**	**2417**	**2461**	**2097**
Capital	ENDOCRINOLOGIA ADULTO	257	225	224	217	234	188	227	203	187
	ENDOCRINOLOGIA TIREOIDE	53	43	10	46	45	42	25	57	54
	GASTROENTEROLOGIA ADULTO	309	301	251	309	264	272	281	293	275
	GASTROENTEROLOGIA HEPATITES VIRAIS ADULTO	80	79	82	94	90	71	110	100	90
	GASTROENTEROLOGIA-DOENÇA INFLAMATÓRIA INTESTINAL	8	8	4	14	8	8	7	8	15
	PROCTO PROCTOLOGIA ADULTO	227	226	185	262	225	200	232	233	219
	REUMATO REUMATOLOGIA ADULTO	200	206	156	201	203	159	151	180	204
	UROLOGIA ADULTO	557	456	392	516	499	411	491	463	389
	UROLOGIA LITÍASE RENAL	0	0	0	4	2	0	0	0	1
	UROLOGIA VASECTOMIA	103	97	76	91	112	89	109	95	91
	Total Capital	**1794**	**1641**	**1380**	**1754**	**1682**	**1440**	**1633**	**1632**	**1525**

expressas na forma de texto livre, redigido pelos profissionais da Atenção Básica. Esse problema requer uma solução capaz de analisar estes textos e extrair deles características capazes de, segundo algum modelo, indicar a adequação ou não do encaminhamento, isto é, aprová-lo ou não. Conforme a definição de Manning, Raghavan e Schütze (2008), uma das possíveis abordagens para esse tipo de problema é a Classificação de Textos baseada em aprendizado de máquina.

Para identificar e representar as características textuais que são usadas para gerar um modelo de classificação com base em Aprendizado de Máquina, podem ser usadas abordagens estatísticas ou semânticas. Ao discutir essas abordagens no escopo da Mineração de Dados e Textos, Faceli (2011) descreve que as primeiras normalmente geram uma análise estatística baseada na ocorrência de palavras isoladas ou compostas presentes no texto, que são identificadas pela sua frequência de coocorrência ou pela presença em dicionários ou ontologias. Já as últimas fazem uma análise sintática e semântica do texto, dando importância à função de cada palavra e não apenas à sua presença no texto. Este segundo tipo de abordagem está associado ao conceito de Processamento de Linguagem Natural (PLN).

Conforme será demonstrado no Capítulo 3, se observa que abordagens estatísticas apresentam resultados competitivos para tarefas de classificação de textos e que o PLN tem servido como um forte aliado na análise de textos clínicos, apesar da área ainda apresentar importantes desafios. Considerando-se os resultados que têm sido apresentados por soluções baseadas em Classificação de Textos, a hipótese deste trabalho é que uma solução com base nessas técnicas de representação de textos, que inclua uma cuidadosa seleção e configuração de modelos de Aprendizado de Máquina, e seja devidamente mentorada por especialistas do domínio, seria capaz de atender à demanda por automação do processo de regulação no projeto TelessaúdeRS.

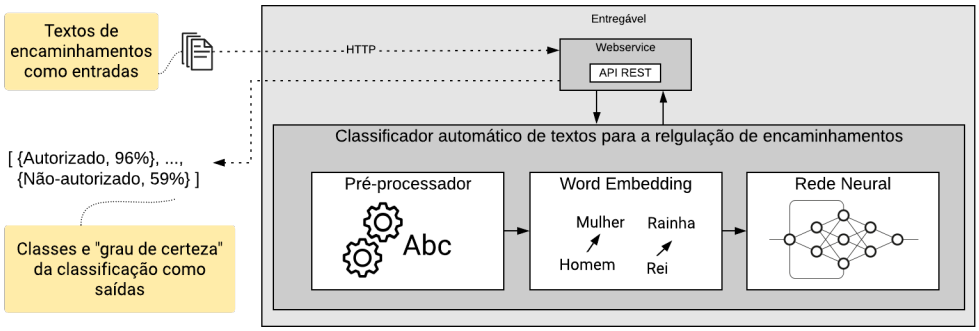
Assim, o objetivo geral deste trabalho é desenvolver, testar e avaliar uma ferramenta de Classificação de Textos baseada em Aprendizado de Máquina que otimize o processo de regulação de encaminhamentos ambulatoriais da Atenção Básica à Atenção Especializada, no âmbito do projeto TelessaúdeRS.

Para isso, são exploradas tanto abordagens estatísticas quando semânticas para seleção de características (*feature engineering*) textuais úteis à classificação, as quais são submetidas a diferentes algoritmos de Aprendizado de Máquina para induzir os classificadores. Tendo-se como objetivo identificar, dentre os conjuntos de tarefas avaliados, qual a melhor combinação de tarefas para criar um fluxo de extração de *features* dos tex-

tos de encaminhamentos e selecionar o Modelo de Classificação mais adequado à tarefa, considerando essas *features*.

O resultado é atingido através de diversos experimentos reportados ao longo deste documento, culminando com a publicação do componente de software representado pela Figura 1.2.

Figura 1.2: Contribuição geral deste trabalho



O presente trabalho é desenvolvido em parceria entre o Instituto de Informática e o TelessaúdeRS, ambos da UFRGS, com médicos reguladores participando direta e ativamente das discussões que conduzem elaboração dos algoritmos aqui desenvolvidos. A Secretaria Estadual de Saúde do RS é centro coparticipante, como campo de pesquisa, ao fornecer os dados dos encaminhamentos para avaliação.

O restante deste documento está estruturado da seguinte forma: O Capítulo 2 introduz brevemente os conceitos necessários à compreensão dos experimentos realizados. O Capítulo 3 descreve alguns dos trabalhos relacionados a esta pesquisa. No Capítulo 4 a proposta é formalmente apresentada e no Capítulo 5 são reportados os experimentos conduzidos. O Capítulo 6 faz um comparativo de resultados do método proposto e do método atual de Regulação e, por fim, no Capítulo 7 são apresentadas a conclusão, as limitações e os possíveis trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo trata dos principais conceitos teóricos envolvidos neste trabalho, visando instrumentar o leitor para a compreensão dos capítulos seguintes, em que serão apresentadas as aplicações práticas desses mesmos princípios.

Os temas principais deste trabalho são introduzidos na seção 2.1 que apresenta o conceito da Informática Médica, na seção 2.2 que aborda o conceito de Aprendizado de Máquina e na seção 2.3 onde são apresentados conceitos relativos à Classificação de Textos.

Alguns elementos destes últimos dois temas são detalhados nas seções seguintes. As seções 2.4, 2.5 e 2.6 apresentam, respectivamente, técnicas de pré-processamento, modelos computacionais de representação de textos e técnicas de extração de características em *corpus* textuais. A seção 2.7 aborda a questão do balanceamento dos dados e na seção 2.8 são apresentados alguns algoritmos de Aprendizado de Máquina. Por fim, na 2.10 são abordadas formas de avaliar a aplicação destes algoritmos.

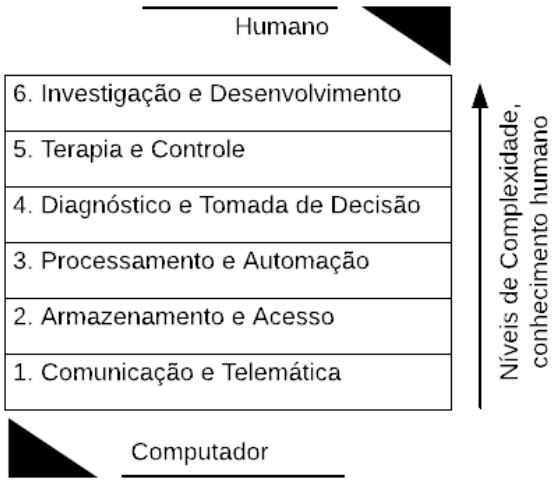
2.1 Informática Médica

É possível encontrar na literatura diversas definições do que vem a ser informática médica. Em Hoyt (2009) são apresentadas três diferentes definições e, a despeito das variações em cada uma, todas as têm em comum a informação como ponto central da disciplina. Em um artigo sobre a história da disciplina Hogarth (1998) apresenta a definição de que a Informática Médica "é o campo científico que trata do armazenamento, recuperação, e uso otimizado da informação biomédica, dados, e conhecimento para a resolução rápida de problemas e tomada de decisões".

Partindo de uma definição ampla como a acima apresentada, há que se supor que a área tende a abarcar uma ampla gama de aplicações práticas. Uma sistematização da aplicação da informática na área médica é apresentada em Pinho (2002) e reproduzida com adaptações na Figura 2.1. Nas palavras do autor, são seis os níveis considerados, cada um tratando de diversos tipos de aplicações. No primeiro nível, estão aplicações ligadas com aquisição de dados e comunicação. O segundo nível tem a ver com o armazenamento de dados, tal como foram adquiridos ou quando muito com algum tratamento, não existindo ainda a sua interpretação. O terceiro nível está associado à fase de processamento de dados. No quarto os dados processados são interpretados. É uma fase em que

a intervenção humana é preponderante. Os resultados interpretados são usados no quinto nível para gerar uma terapia ou para controlar uma dada variável. Finalmente o sexto nível relaciona-se com novas pesquisas e desenvolvimentos e é outro dos níveis em que a presença humana é essencial.

Figura 2.1: Classificação de aplicações de tecnologia em Medicina



Fonte: Adaptado de Pinho (2002)

Uma questão útil para facilitar o desenvolvimento de aplicações em todos os níveis, e que retoma a importância da organização da informação como tema central na informática médica é a definição clara e compartilhada dos conceitos envolvidos na área. Uma das iniciativas para endereçar esta questão é o Sistema Unificado de Linguagem Médica, apresentado na seção a seguir.

2.1.1 Sistema Unificado de Linguagem Médica

Ao tratar das origens da disciplina da Informática Médica, Hogarth (1998) menciona que, partindo da ideia de um cirurgião escocês chamado Roget, que inventou um método de representar o conhecimento baseado no princípio que todas as coisas "são somente conceitos", a Biblioteca Nacional de Medicina *(National Library of Medicine)* (NLM) dos EUA em 1986 começou o desenvolvimento do metatesauro do *Unified Medical Language System* (UMLS).

O sistema, em desenvolvimento ainda hoje, é descrito em sua página de internet[1] como "um conjunto de arquivos e software que reúne muitos vocabulários e padrões de

[1] https://www.nlm.nih.gov/research/umls/index.html

saúde e biomédicos para permitir a interoperabilidade entre sistemas de computadores". O UMLS pode ser aplicado para facilitar o mapeamento entre diferentes terminologias, desenvolver sistemas de recuperação de informação, processar textos para extrair conceitos e relacionamentos, etc. As três fontes de conhecimento do UMLS são o metatesauro, a "Rede Semântica" e as ferramentas SPECIALIST.

O conjunto de ferramentas *SPECIALIST Lexicon and Lexical Tools*[2] oferece ferramentas de Processamento de Linguagem Natural para ajudar desenvolvedores com variações léxicas de termos no domínio da biomedicina.

A Rede Semântica[3] provê uma categorização consistente de todos os conceitos representados no metatesauro do UMLS e um conjunto de relacionamentos semânticos importantes entre os elementos dentro dessas categorias.

O metatesauro inclui termos e códigos de vários vocabulários distintos como ICD-10, que trata da classificação internacional de doenças (CID); o *Medical Subject Headings* (MeSH)[4], usado para indexar e pesquisar informações relacionadas à saúde, criado com base em artigos que aparecem nas bases da NLM como o MEDLINE/PubMed; e o *Systematized Nomenclature of Human and Veterinary Medicine* - SNOMED[5], que é uma terminologia clínica global cujo escopo cobre uma ampla gama de especialidades clínicas.

Os conceitos nesse metatessauro são identificados unicamente através de Identificadores Únicos de Conceitos (*Concept Unique Identifiers* - CUI), permitindo encontrar e relacionar as ocorrências de um mesmo conceito nas várias fontes de dados, ainda que nomeados de maneira diferente.

2.1.2 Apache *clinical Text Analysis and Knowledge Extraction System*

Uma das maneiras de se valer das bases do conhecimento do UMLS é utilizando-o como um dicionário integrado ao *clinical Text Analysis and Knowledge Extraction System*, comumente referido como Apache cTAKES, que é um sistema para Processamento de Linguagem Natural dedicado a extrair informações de registros médicos eletrônicos, Savova et al. (2010).

O Apache cTAKES foi desenvolvido com base no Apache UIMA. UIMA, que é

[2]https://lexsrv3.nlm.nih.gov/Specialist/Home/index.html
[3]https://semanticnetwork.nlm.nih.gov/
[4]https://www.nlm.nih.gov/mesh/meshhome.html
[5]http://www.snomed.org/

sigla para *Unstructured Information Management applications*, é uma especificação para construção de softwares que, conforme descrito no site do projeto[6], analisam grandes volumes de informação não estruturada com vistas a descobrir nela conhecimento que seja relevante ao usuário final.

A ferramenta cTAKES é uma das mais completas opções de código aberto voltada ao processamento de textos clínicos. Os componentes de que o sistema é composto são treinados especificamente para esse domínio e, segundo Savova et al. (2010), são capazes de criar anotações linguísticas e semânticas em textos que podem então ser utilizados no Suporte à Decisão Clínica.

As análises no cTAKES são feitas com a ingestão do texto na ferramenta que então faz com que o texto flua entre os diversos componentes disponíveis e que formam o chamado *pipeline* de análise. Cada componente executa uma avaliação particular do texto (e das anotações já disponíveis) e adiciona novas metainformações em forma de anotações no texto. Os usuários podem configurar *pipelines* próprios, mas a instalação padrão do cTAKES já oferece um padrão chamado *Default Clinical Pipeline*[7].

Outra ferramenta desenvolvida dentro do projeto Apache UIMA e usada em conjunto com o cTAKES é o *CAS Visual Debuger - CVD*. O CVD é um software que funciona como interface gráfica para execução de ferramentas de análises de texto implementadas conforme a especificação UIMA. Conforme explicado em seu site[8], é uma ferramenta para desenvolvedores e não para usuários finais e tem como principal objetivo permitir que se navegue pelos metadados criados pelo processo de análises dos textos a fim de que se possa avaliar os resultados obtidos.

2.2 Aprendizado de Máquina

Na computação, problemas são geralmente resolvidos por um algoritmo que especifica um sequência de passos para a solução. Entretanto, é difícil definir um algoritmo que resolva alguns problemas que humanos realizam com facilidade diariamente, por exemplo, reconhecer um rosto ou compreender um texto. É difícil saber exatamente quais características do rosto devem ser observadas para que possamos reconhecê-lo mesmo quando aparece com um óculos, uma barba, ou de um ângulo totalmente diferente. Com

[6] https://uima.apache.org/
[7] https://cwiki.apache.org/confluence/display/CTAKES/Default+Clinical+Pipeline
[8] https://uima.apache.org/d/uimaj-current/tools.htmlugr.tools.cvd

textos, da mesma forma, não é fácil tornar explícito nosso raciocínio acerca dos conceitos subjacentes em uma palavra ou as associações complexas existentes entre os seus sentidos porque, em geral, dependem do contexto.

Apesar da dificuldade de programar computadores para realizar essas tarefas complexas, elas hoje são frequente e constantemente realizadas usando a computação. Casos típicos que estão presentes no dia a dia da maioria das pessoas atualmente incluem sistemas de anti-spam como o do Gmail, que afirma ter precisão de 99,9% para distinguir mensagens legítimas de spam e phishing, conforme Wen (2020), e o reconhecimento facial do Facebook, que usa fotos do usuário para criar um número único chamado *gabarito* e depois o compara a fotos, vídeos e lives para encontrar os outros conteúdos nos quais o usuário aparece, Facebook (2020).

Esses dois exemplos, assim como inúmeras outras soluções semelhantes, usam técnicas de Inteligência Artificial (IA), especialmente o Aprendizado de Máquinas (do inglês *Machine Learning* ML), para instruir computadores a realizar tarefas altamente complexas cuja solução dificilmente poderia ser codificada de maneira explícita através de um algoritmo e que também não poderiam ser realizadas com a mesma eficiência por humanos, dada a imensa quantidade de informações que precisam ser consideradas para sua execução.

Um problema para o Aprendizado de Máquinas, segundo Mitchell (1997), pode ser definido precisamente como melhorar alguma medida de desempenho P quando executando uma tarefa T através de algum tipo de experiência de treinamento E. Por exemplo, aprender a filtrar spam é aprender uma função (hipótese) que mapeia qualquer e-mail de entrada para um rótulo de spam ou não-spam. Quando esses três componentes (T, P, E) estão especificados, tem-se um problema de aprendizado bem definido.

Essa ideia é resumida por Faceli (2011) como "A capacidade de melhorar o desempenho na realização de alguma tarefa por meio da experiência". Essa melhoria acontece quando os algoritmos conseguem aproximar uma função capaz de resolver um problema usando dados que representam instâncias desse problema. Para aproximar essa função, usam o princípio da indução, no qual partindo-se de características de casos particulares se formula uma hipótese geral capaz de descrevê-los. A ideia, segundo Mitchell (1997), é que qualquer hipótese que aproximar a função alvo bem o bastante num grande conjunto de exemplos também a irá aproximar bem o bastante para um exemplo nunca antes observado, permitindo assim a generalização dessa hipótese.

As tarefas em que os algoritmos de Aprendizado de Máquinas são empregados

podem ser Descritivas ou Preditivas. Segundo Faceli (2011), as tarefas descritivas têm como meta explorar ou descrever um conjunto de dados. Encontrar grupos de objetos semelhantes ou regras de associação que relacionam objetos semelhantes em um conjunto são exemplos típicos de tarefas descritivas.

Já as tarefas preditivas visam encontrar uma função a partir dos dados de treinamento que possa prever um rótulo ou um valor que caracterize um exemplo nunca antes visto. Os algoritmos preditivos seguem o paradigma de Aprendizado Supervisionado, em que se simula a presença de um supervisor que conhece o rótulo ou valor correto para cada exemplo, permitindo assim avaliar a capacidade preditiva da hipótese induzida.

Tarefas preditivas são empregadas em problemas de Regressão e de Classificação. Problemas de regressão são aqueles em que o resultado buscado pela função hipótese é um número, isto é, o retorno da função pertence a um espaço contínuo de valores. Exemplos típicos de tarefa de regressão incluem a predição do preço de ações e a detecção de fraudes por operadoras de cartão de crédito.

Tarefas de classificação têm como retorno um valor de um espaço discreto, comumente chamado atributo-alvo ou classe-alvo. O tema da classificação é bastante estudado por comunidades de áreas como banco de dados, mineração de dados e recuperação de informações. A noção geral do problema de classificação, para Manning, Raghavan e Schütze (2008), é a de que: dado um conjunto de classes deve-se determinar a qual delas um determinado objeto pertence.

2.3 Classificação de Textos

A Classificação de Textos é um tipo particular do problema de Classificação e tem aplicação em múltiplas áreas com domínios bastante diversos, incluindo diagnósticos médicos, filtragem de notícias, roteamento de demandas em sistemas de *help-desk*, identificação de spam e detecção de plágio. O tema também é referido na literatura por outros nomes como *Categorização de Textos*, *Classificação de tópicos* e *Detecção de tópicos*.

Normalmente, para a Classificação de Textos usando Aprendizado de Máquina Supervisionado, um conjunto de documentos, dito *corpus*, está disponível junto com uma identificação da classe a que cada um dos documentos pertence. A classe, também dita *target* ou *label*, é a informação segundo a qual pretendemos classificar os documentos, por exemplo: Spam / Não Spam; Sentimento Negativo / Sentimento Positivo; Problema Técnico / Problema Financeiro etc. A classificação pode se dar entre apenas duas classes,

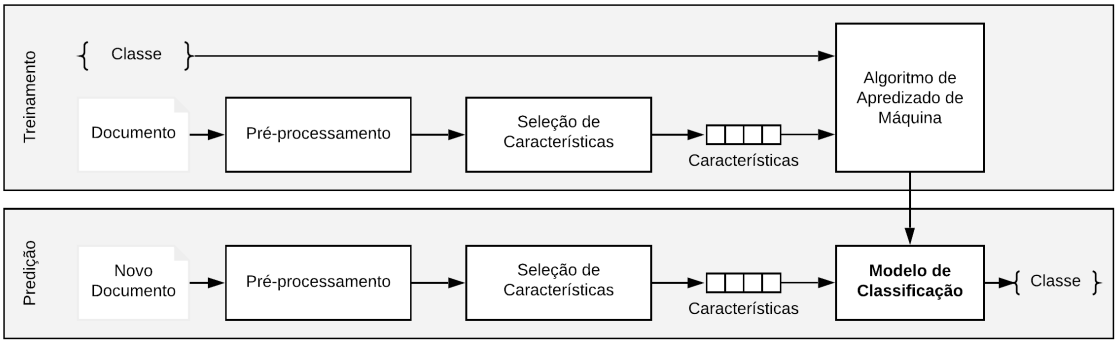
como nos exemplos anteriores, dita *binary classification*, ou entre múltiplas classes distintas, dita *multi-class classification*[9]. Além disso a classificação pode, como nos exemplos acima, implicar em atribuir apenas uma classe ao documento ou múltiplas classes, por exemplo, atribuir a uma notícia às classes Esportes e Futebol, simultaneamente.

De posse dos documentos e suas respectivas classes, realiza-se um processo conhecido como Engenharia de Caraterísticas, que inclui a Seleção de Características. Nesse processo, normalmente são realizadas algumas etapas de pré-processamento no corpo de texto de cada documento e depois algumas características do texto resultante são selecionadas diretamente ou a partir de alguma transformação. Skiena (2017), em seu capítulo sobre Engenharia de *features* a resume como "a fina arte de aplicar conhecimento de domínio para tornar mais fácil para os algoritmos de Aprendizado de Máquina fazerem seu trabalho".

As características e a respectiva classe do documento são submetidas a um algoritmo de Aprendizado de Máquina que busca encontrar um modelo de classificação, dito classificador, capaz de prever a classe do documento a partir das características. Uma vez determinado este modelo, novos documentos podem ser classificados sendo submetidos ao mesmo pré-processamento, engenharia de *features* e análise pelo classificador.

A Figura 2.2, representa o fluxo envolvido nas etapas de treinamento, quando o modelo é gerado, e na etapa de predição, quando o modelo é usado para predizer a classe de novos documentos.

Figura 2.2: Classificação de Textos usando Aprendizado de Máquina Supervisionado



Fonte: Acervo do projeto, adaptada de Bird, Klein e Loper (2009)

[9]https://developers.google.com/machine-learning/glossary

2.4 Pré-processamento de Textos

Apesar de algoritmos de Aprendizado de Máquina serem frequentemente adotados para extrair conhecimento de conjuntos de dados, Faceli (2011) explica que o seu desempenho é geralmente afetado pelo estado dos dados. Os conjuntos, segundo a autora, podem conter ruídos e imperfeições, podem ter um número muito elevado de atributos, etc. Técnicas de pré-processamento são frequentemente empregadas para melhorar a qualidade dos dados minimizando esses problemas.

No contexto da Classificação de Textos, a etapa de pré-cessamento visa tornar os documentos de entrada mais consistentes para facilitar a representação vetorial do texto. Esta é uma etapa que pode consumir mais de 60% do tempo, segundo WÄhner e Penchikala (2017), em projetos de Aprendizado de Máquina.

Nas seções seguintes, algumas das técnicas de pré-processamento comuns na Classificação de Textos serão introduzidas.

2.4.1 Remoção de *stop-words*

Uma das transformações comumente realizadas no pré-processamento é a remoção de *Stop-words*, a qual elimina palavras que, por serem muito comuns, em regra, não ajudam a distinguir entre classes. A respeito disso, o trabalho de Srividhya e Anitha (2010) que se dedica a avaliar o impacto das tarefas de pré-processamento sobre os resultados da classificação de textos, conclui que a remoção das *stop-words* se mostra muito importante como tarefa de pre-processamento. Já em Sarker e Gonzalez (2015), onde são analisados textos coletados em redes sociais para avaliar efeitos colaterais de medicamentos, a conclusão é que as *stop-words* têm papel importante e efeito positivo no desempenho dos classificadores quando são usados n-gramas[10] com dois e três termos. Depreende-se disso que a aplicação ou não da técnica precisa ser avaliada conforme o contexto da tarefa e o vocabulário envolvido.

[10]https://developers.google.com/machine-learning/glossary#N-gram

Quadro 2.1: Exemplo de *Part Of Speech Tagging* com a biblioteca textacy

ENTRADA: *Classificação de textos*
SAÍDA: *[[('Classificação', 'NOUN'), ('de', 'ADP'), ('textos', 'NOUN')]]*

2.4.2 *Stemming* e Lematização

Por questões gramaticais, documentos de texto usam diferentes formas de uma mesma palavra, com variações de gênero, número e grau, além de outras como flexões verbais, etc. Para lidar com essas variações, na etapa de pré-processamento também é comum a aplicação das técnicas de *stemming* e de lematização (*lemmatization*), que têm como objetivo reduzir as formas flexionadas e derivadas dos termos para uma base comum.

No caso do *stemming*, normalmente o processo usa heurísticas para cortar as terminações das palavras, removendo sufixos típicos de derivações e buscando com isso acertar na maioria das vezes na redução de termos ao seu radical. Naturalmente, essas heurísticas são dependentes de idioma. O *RSLP Stemmer* (Removedor de Sufixos da Língua Portuguesa), apresentado em Orengo e Huyck (2001), é um *stemmer* especializado na língua portuguesa e pode ser utilizado diretamente em bibliotecas como a NTLK[11].

O processo de lematização demanda análises morfológicas dos termos a fim de reduzir a palavra ao formato no qual ela seria encontrada em um dicionário, conhecido como "lema". Para essas análises, o processo requer que os tokens estejam identificados com uma etiqueta que indique função sintática do termo (POS Tag - *Part of speech tag*). Algumas bibliotecas de PLN permitem adicionar essas etiquetas automaticamente, a textacy, apresentada em labs (2016), é uma delas e tem suporte nativo a diversos idiomas, inclusive ao português. Um exemplo do processo de *POS Tagging* pode ser observado no Quadro 2.1.

Como se observa na Tabela 2.1, os processos de lematização e de *stemming* produzem resultados diferentes, sendo que a lematização tem uma tendência de produzir menos tokens, já que reduz as várias flexões de verbos para sua forma infinitiva, o que pode ser considerado positivo pela redução de dimensões do vetor resultante. Contudo, assim como no caso de *stop-words* esse resultado mais eficiente na redução das dimensões pode não ser o desejado já que, por exemplo, perde-se referências à questão do tempo (passado, presente, futuro) no discurso. Segundo Allahyari et al. (2017), na prática, o processo de

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

Tabela 2.1: Comparação entre os processos de *stemming* e lematização

Entrada[a]	**Saída**	
	Stemming (RSLP)	**Lematização (textacy)**
organizar organizando organizado	{"organiz":3}	{"organizar": 3}
tiver tenho tinha tem	{"tiv":1, "tenh":1, "tinh":1, "tem":1}	{"ter": 4}

[a] Termos da coluna Entrada são submetidos a cada um dos dois processos. A coluna Saída mostra os *tokens* gerados e a sua respectiva contagem.

stemming ainda é o mais usado, contudo, com o crescimento e o aperfeiçoamento das funcionalidades de anotação automática das bibliotecas de PLN é possível que isso mude no futuro.

2.5 Representação de Textos

Uma vez que os textos tenham sido tratados pela aplicação das técnicas mencionadas na seção anterior eles precisam ser convertidos em uma representação que possa servir como entrada para os algoritmos de ML. A maneira mais comum de representar documentos é usando o *Vector Space Model* - VSM. No VSM, cada documento é um vetor numérico cujas dimensões são dadas pelos termos presentes no documento e o valor de cada dimensão corresponde ao "peso" (importância) deste termo. Algumas das técnicas para criação dessas representações são discutida a seguir.

2.5.1 *Bag-of-Words* e *Bag-of-ngrams*

Um das maneiras mais intuitivas, segundo SCIKIT-LEARN developers (2017), de representar um *corpus* no VSM é usando a representação chamada *Bag-of-Words*, que funciona da seguinte forma: Cria-se uma matriz M x N, onde M é a quantidade de documentos e N é a quantidade de termos distintos do *corpus*. Cada linha da matriz refere-se a um documento, cada coluna refere-se a um termo. Para cada documento do *corpus*, conta-se a quantidade de vezes que cada termo ocorre e armazena-se o valor dessa contagem na respectiva linha/coluna. Essa representação de frequências baseada na contagem

de ocorrência dos termos é chamada de *Word Count*.

Um problema da representação *Bag-of-Words* - e que inclusive justifica o seu nome - é que ela não é capaz de capturar relações de coocorrência ou mesmo posição entre os termos. Como ela apenas registra a quantidade de vezes que um termo ocorre no documento, as representações de um documento contendo "João matou Maria" e de outro contendo "Maria matou João" são exatamente iguais.

Uma alternativa para endereçar este problema é utilizar n-gramas de dois ou mais termos como colunas da referida matriz. Assim, em vez de cada coluna representar apenas um termo (ex.: 'João'), ela representaria 2 ou mais (ex.: 'João matou'). Essa representação é chamada *Bag of n-grams*.

Um problema da representação *Word Count* é o fato de que documentos mais longos terão uma contagem maior de termos do que documentos mais curtos, criando vetores bastante diferentes, ainda que discorram do mesmo assunto. Para evitar essas discrepâncias entre os vetores, basta normalizar as contagens de cada termo dividindo-as pela quantidade total de termos do documento. Essa nova representação é chamada de Frequência de Termo (do inglês *Term Frequency*).

A representação baseada em *Term Frequency* entretanto não resolve um outro problema importante, o fato de que palavras muito comuns no *corpus* podem dominar a composição dos vetores. Para lidar com isso, um outro refinamento utilizado é diminuir os pesos dos termos que são comuns a muitos documentos. Essa nova representação é chamada *"Term Frequency X Inverse Document Frequency"* ou TFiDF. O TFiDF calcula o peso de um termo considerando não só o documento em questão mas também os outros documentos do *corpus*. Uma representação da função peso do TFiDF é dada por:

$$\text{TFiDF}(w) = \text{tf}(w) \cdot \log \frac{N}{\text{df}(w)} \tag{2.1}$$

Onde *tf(w)* representa a frequência do termo no documento em questão, *N* é a quantidade de documentos no *corpus* e *df(w)* é a quantidade de documentos que contém o termo. Cabe mencionar que existem variações da fórmula onde o primeiro termo é computado como $1+\log \text{tf}(w)$ para evitar que múltiplas ocorrências do termo num mesmo documento elevem de mais seu peso.

Segundo Allahyari et al. (2017) o TFiDF é o mais popular cômputo de peso para a definição do VSM, contudo, é importante observar que há modelos de aprendizado de máquina tais como o *Multinomial Naive Bayes* que funcionam melhor com a simples contagem de termos por trabalharem com números inteiros, segundo SCIKIT-LEARN

developers (2017). Assim, mais uma vez, percebe-se que a forma de representação dos documentos precisa ser avaliada conforme o contexto da tarefa.

Exemplos das diferentes representações mencionadas nesta seção são apresentados nas tabelas a seguir.

Tabela 2.2: Exemplos de textos representados nas tabelas seguintes

doc1	Classificação de textos baseada em aprendizado de máquina
doc2	Modelo de classificação com base em aprendizado de máquina
doc3	Mineração de dados e textos

Tabela 2.3: Representação *Bag-of-Words* usando a contagem de termos

	aprendizado	**base**	**baseada**	**classificação**	**com**	**dados**	**de**	**em**	**mineração**	**modelo**	**máquina**	**textos**
doc1	1	0	1	1	0	0	2	1	0	0	1	1
doc2	1	1	0	1	1	0	2	1	0	1	1	0
doc3	0	0	0	0	0	1	1	0	1	0	0	1

Tabela 2.4: Representação *Bag-of-Words* usando TFiDf

	aprendizado	**base**	**baseada**	**classificação**	**com**	**dados**	**de**	**em**	**mineração**	**modelo**	**máquina**	**textos**
doc1	0,3307	0	0,4349	0,3307	0	0	0,5137	0,3307	0	0	0,3307	0,3307
doc2	0,2936	0,3861	0	0,2936	0,3861	0	0,4560	0,2936	0	0,3861	0,2936	0
doc3	0	0	0	0	0	0,5845	0,3452	0	0,5845	0	0	0,4445

Tabela 2.5: Bag-of-ngrams, com n-gramas de 1 e 2 termos, usando contagem de termos

	aprendizado	**aprendizado de**	**base**	**base em**	**baseada**	**baseada em**	**classificação**	**classificação com**	**classificação de**	**...**	**em aprendizado**	**mineração**	**mineração de**	**modelo**	**modelo de**	**máquina**	**textos**	**textos baseada**
doc1	1	1	0	0	1	1	1	0	1	...	1	0	0	0	0	1	1	1
doc2	1	1	1	1	0	0	1	1	0	...	1	0	0	1	1	1	0	0
doc3	0	0	0	0	0	0	0	0	0	...	0	1	1	0	0	0	1	0

2.5.2 *Word Embeddigns*

As representações *Bag-of-Words* e suas variações não capturam bem a semântica e a similaridade das palavras, além de terem uma alta dimensionalidade, pois as matrizes têm a quantidade de colunas definidas pelo número de termos distintos do *corpus*, ou alguma variação disso.

Uma alternativa àquelas representações são as "representações distribuídas de palavras" (do inglês *Word Embeddings*), cuja ideia base foi proposta por Bengio et al. (2003). Naquele artigo os autores mencionam o problema da "maldição da dimensionalidade" explicando que para modelar a distribuição conjunta entre variáveis categóricas (ex.: a coocorrência de palavras) é necessário modelar todas as possíveis combinações. Por outro lado, quando se está modelando variáveis contínuas é possível se obter mais facilmente uma função de generalização, visto ser esperado que se possa encontrar suaviza-

ções dos valores contínuos que equivalham ao rearranjo de todas as variáveis, necessário quando se trata de dados categóricos.

Com base nessa ideia, os autores propõem uma representação de textos em que cada palavra do vocabulário tem seu próprio vetor de valores reais, um *word feature vector*, representando características daquele termo no contexto do *corpus*[12], assim a função da distribuição conjunta de sequências de palavras pode ser expressa em função desses vetores.

Esses vetores de representação são criados com base no contexto das palavras no *corpus*, fazendo com que termos usados de maneira semelhante tenham representações próximas no espaço vetorial. Isso permite capturar, de certa forma, a semântica desses termos e, por serem representados por valores reais, é possível executar sobre suas representações operações de álgebra linear capazes de simular operações semânticas.

Um exemplo comum disso encontrado na literatura a respeito de *Word Embeddings* é o do artigo de Mikolov et al. (2013) em que se demonstrou que realizar a operação *King + Woman - Man* (com os respectivos vetores dessas palavras) resultava em um *word feature vector* próximo àquele da palavra *Queen*. O que esse exemplo demonstra é que tais modelos são capazes de capturar atributos de palavras que representam conceitos como "gênero" e outros bem mais abstratos como "realeza".

Quando uma rede neural demonstra bom desempenho em uma tarefa, por exemplo, na predição do sentimento relacionado a uma palavra, pode-se assumir que as camadas ocultas dessa rede representam características que são úteis para relacionar as entradas (a palavra) com a saída da rede (o sentimento relacionado). Representações *Word Embeddings* são geradas com base nessa ideia, i.e., quando o modelo aprende a realizar com qualidade essa tarefa, suas camadas ocultas podem ser consideradas uma representação distribuída da palavra fornecida como entrada. A ideia geral, portanto, para a criação de *Word Embeddings* é treinar uma rede neural e capturar sua camada intermediária. Nas seções seguintes algumas das variações desse processo são discutidas.

2.5.2.1 Word2vec e FastText

O processo de construção de modelos de *Word Embeddings*, segundo Santos Joaquim; Vieira (2020), envolve os seguintes passos: Construir o vocabulário do *corpus*; Gerar o *input* para a arquitetura; Construir a arquitetura do modelo; Treinar o modelo e

[12] Um exemplo introdutório bastante didático dos conceitos de *word embeddings* pode ser encontrado em https://gist.github.com/aparrish/2f562e3737544cf29aaf1af30362f469

Obter os *embeddings*.

O modelo de treinamento (i.e. Word2vec ou FastText) relaciona-se com a etapa de geração do *input* para a arquitetura. O Word2vec, proposto no já referido artigo de Mikolov et al. (2013), utiliza as palavras como a menor unidade para o treinamento da rede neural que gera os *embeddings*, i.e., as entradas para a rede são as palavras do vocabulário e a rede gera diretamente um vetor que representa a palavra em questão.

No FastText, proposto por Bojanowski et al. (2016), as entradas são *subwords* ou *"bag of character n-grams"*, isto é, uma palavra é segmentada num certo número de caracteres (o tamanho dos n-gramas é um parâmetro do modelo) e essas pequenas partes das palavras servem como entradas para a rede. A saída final do algoritmo FastText é equivalente, em termos de formato, à saída produzida pelo Word2vec, contudo os vetores aqui não são computados diretamente para cada palavra, mas sim produzidos pela combinação dos vetores produzidos para cada *subword*.

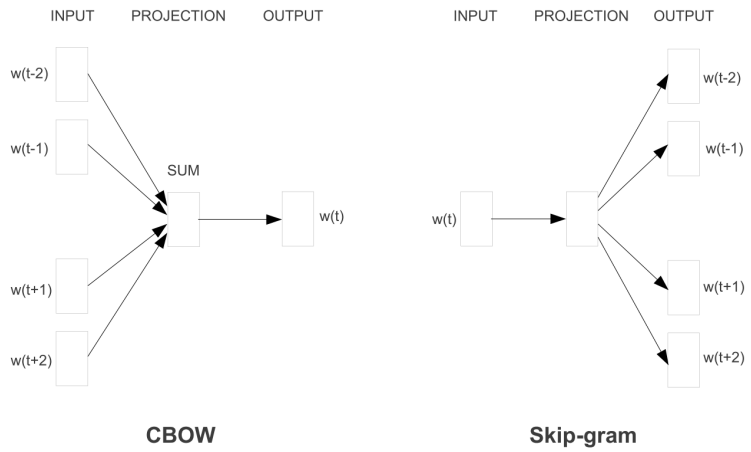
Uma das principais vantagens do FastText sobre o Word2vec é ele consegue generalizar melhor, já que consegue resolver um dos grandes problemas do Word2vec que é lidar com palavras fora do vocabulário. No caso do Word2vec, se uma palavra não foi vista em tempo de treinamento, mas é encontrada durante uso do *embedding*, ela será representada como um vetor de zeros, pois o *embedding* não é capaz de produzir uma representação dela. No caso do FastText, uma representação para uma palavra nunca antes vista pode ser produzida pela combinação dos vetores das suas partes.

Além dessa vantagem na generalização, o FastText também extrai mais informação dos dados ao segmentar as palavras em vez de usá-las como *tokens* únicos, isso permite que modelos FastText sejam treinados em *corpus* menores do que aqueles necessários para bons resultados com Word2vec.

2.5.2.2 CBOW e SkipGram

O modo de treinamento usado na construção de *Word Embeddings* relaciona-se com a etapa de construção da arquitetura do modelo. O modo *Continuous Bag-of-Words - CBOW* tem como tarefa predizer a enésima palavra, dado que se conhece as demais palavras de uma sentença. Dito de outra forma, o modo busca predizer uma palavra a partir do seu contexto. Dessa forma, um modelo é treinado recebendo N palavras como entrada e tendo uma palavra como saída. O valor de N é um parâmetro do treinamento, chamado "janela" (*window*), e determina a distância máxima entre a palavra-alvo (*t*) e as que são usadas como entrada, por exemplo, com $N = 2$, duas palavras anteriores (*t-1* e

Figura 2.3: Arquiteturas CBOW e SkipGram



O CBOW prevê a palavra corrente a partir de seu contexto e o SkipGram prevê as palavras do entorno a partir da palavra corrente.
Fonte: Mikolov et al. (2013)

t-2) e duas palavras posteriores (*t+1* e *t+2*) à palavra-alvo seriam usadas como entrada para o treinamento.

O modo de treinamento SkipGram funciona de maneira inversa ao CBOW. Aqui, o contexto é identificado a partir de uma palavra de entrada. Um parâmetro *window* também determina quantas são as palavras do contexto que serão descobertas. A rede para treinamento do modo SkipGram tem, naturalmente, uma arquitetura diferente, e na camada de saída prevê diferentes probabilidades para as várias possíveis palavras que formam o contexto previsto a partir da entrada. A Figura 2.3 ilustra as duas arquiteturas.

2.6 Extração de características

Guyon e Elisseeff (2006) argumentam que encontrar uma boa representação para os dados é uma tarefa muito dependente do domínio e muito relacionada com as medidas disponíveis. Os autores explicam que entre os aspectos da extração de características (ou *features*) estão: Construção de *features*; Geração de subconjuntos de *features* (aqui chamada Seleção de características); Definição de critérios de avaliação (e.g., índice de relevância ou poder preditivo) e Método de avaliação.

A definição de critérios de avaliação tem a ver com a forma como se escolhe as *features* que serão utilizadas. Aqui podem ser usadas técnicas de filtragem ou métodos de empacotamento. Guyon e Elisseeff (2006) ensinam que as técnicas de filtragem estão relacionadas a métodos de ranqueamento que provém uma ordenação das *features* com

base na sua relevância. Incluem, por exemplo, os coeficientes de correlação (e.g., Informação Mútua) que medem a dependência de *features* individuais com a saída desejada (ex.: classe-alvo da classificação). Já os métodos de empacotamento (*wrapper methods*) e métodos de conjunto (*ensemble methods*), por outro lado, usam algum modelo de aprendizado de máquina como uma "caixa preta" para pontuar subconjuntos de *features* de acordo com seu poder preditivo.

O Método de Avaliação trata do problema de que o método escolhido precisa ser avaliado por um conjunto limitado de dados. Há basicamente duas estratégias para lidar com isso, segundo Guyon e Elisseeff (2006): *in-sample* e *out-of-sample*. A primeira usa todos os dados disponíveis para computar empiricamente uma estatística e depois aplica um teste estatístico clássico (ex.: *Chi-square*) para validar a significância do valor calculado. A segunda, conhecida como *Holdout*, separa os dados disponíveis em um conjunto de treinamento e outro de teste e usa o primeiro para estimar os parâmetros do modelo preditivo e o segundo para avaliar o desempenho do modelo. A técnica de validação cruzada (em inglês, *cross-validation*), que consiste em realizar várias dessas divisões de *holdout*, separando a cada vez porções diferentes dos dados para treino e teste e computando a média do desempenho do modelo é comumente empregada para diminuir a variância nos resultados do modelo.

Com relação à construção de *features*, o mesmo autor explica que ela pode estar embutida no processo de geração de um modelo, por exemplo os pesos dados a cada atributo em uma rede neural, ou ser um pré-processamento. Nesse segundo caso estão incluídas as técnicas de uniformização (*Standarization*) que convertem diferentes unidades de medida para uma base comum; Normalização que transformam os dados alterando sua distribuição (ex.: *Quantile Transformer*) e Métodos de incorporação (*embedding*) de espaço, que projetam o espaço de *features* em um conjunto menor de dimensões, mantendo parte relevante da informação (e.g., *Latent Semantic Analysis - LSA*).

A seleção de características é o processo de delimitar um sub-conjunto de características a serem utilizadas no processo de aprendizado de máquina para derivar o modelo de classificação. Conforme ensinam Manning, Raghavan e Schütze (2008), esse processo serve a dois propósitos: Melhorar o desempenho do treinamento e da classificação de documentos, pela redução de características a serem consideradas; e aumentar a acurácia dos classificadores pela redução de *ruído*.

Enquanto se pode considerar operações como a remoção de *stop-words*, o *stemming* e a lematização como seleção de características, já que na prática seus efeitos podem

corresponder justamente a esses recém mencionados, neste trabalho elas foram alocadas na seção de pré-processamento, pois é onde mais comumente foram encontradas na revisão bibliográfica realizada. Reservou-se, assim, a seção atual para os procedimentos em que se realizam avaliações de quais características do vetor resultante do pré-processamento melhor atendem àqueles dois critérios e como esse vetor pode ser transformado ainda mais para melhorar os resultados.

Segundo Bird, Klein e Loper (2009), apesar de ser frequentemente possível conseguir bons resultados usando características óbvias dos textos, normalmente há melhorias significativas em usar um conjunto de características cuidadosamente selecionados com base no conhecimento dos dados e da tarefa a ser desenvolvida. Alguns exemplos práticos de como realizar essa seleção usando a biblioteca *Natural Language ToolKit* - NLTK[13], uma das principais plataformas para construção de programas para manipulação de linguagem natural, são apresentados naquele trabalho juntamente com o detalhamento do raciocínio utilizado na tarefa.

Especificamente sobre a redução de ruídos, tem-se que uma característica é considerada ruído se ela aumenta o erro de um classificador. Por exemplo, se uma palavra rara aparece no conjunto de treinamento acidentalmente sempre associada a uma dada classe, mas não carrega nenhuma relação real com aquela classe, essa palavra tende a aumentar o erro do classificador, pois ele irá associar a ocorrência do termo àquela classe, mas isso não irá generalizar para fora do conjunto de treinamento. Essa associação entre uma característica e uma classe que funciona bem no conjunto de treinamentos, mas não se prova verdadeira num conjunto de dados maior, é chamada de *overfitting*.

O processo de seleção de características funciona basicamente analisando cada uma das características e computando uma *medida de utilidade*, conforme a definição de critérios de avaliação escolhida, aplicando-se um método de avaliação e tomando-se as características com o maior valor de utilidade. O limite de quantas serão as características escolhidas pode ser dado por um valor fixo ou tomando-se aquelas dentro de um certo limiar. Entre as possíveis medidas de utilidade de uma característica estão: *Ganho de informação* e o já citado *Informação Mútua*.

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

2.6.1 Ganho de Informação

Em Ravi e Srinivasan (2018) os autores explicam que a Entropia é uma medida de incerteza a cerca de uma questão qualquer e que o Ganho de informação é a medida da diferença da entropia antes e depois de uma certa informação ser conhecida. Os autores exemplificam o conceito com o seguinte problema: Suponha que se tenha que classificar um animal como hipopótamo ou girafa. Se nos é declarado que "o animal em questão tem 4 patas", nossa incerteza continua a mesma de antes de sabermos dessa característica - a entropia não muda, pois as respostas possíveis ainda são as mesmas de antes, já que ambos os animais têm 4 patas. Diz-se que a declaração não carrega informação. Por outro lado, se nos for declarado que "o animal em questão tem aproximadamente 3m de altura", então seremos capazes de adivinhar que o animal é a girafa, pois essa é uma característica das girafas, mas não dos hipopótamos.

O exemplo acima demonstra que algumas caraterísticas carregam mais informação do que outras para responder a certas questões. No contexto da classificação de textos, o ganho de informação refere-se ao quão útil é o conhecimento de que uma determinada caraterística ocorre nos documentos de uma certa classe. Semelhante ao conhecimento sobre a quantidade de patas ou a altura de um animal, uma determinada característica pode ser necessária e suficiente para identificar uma certa classe, ou pode ser totalmente irrelevante.

De maneira mais formal, no contexto de uma classificação binária, se considerarmos num conjunto de S elementos, onde p_+ representa a proporção de exemplos positivos e p_- a de exemplos negativos desse conjunto, a entropia do conjunto pode ser definida pela equação 2.2:

$$Entropia(S) = -p_+ \log_2 p_+ - p_- \log_2 p_- \tag{2.2}$$

Nesse mesmo contexto, o Ganho de Informação, que como descrito representa uma diminuição da entropia, pode ser formalmente representado pela equação 2.3 em que A é um atributo escolhido para separar os elementos do grupo e v é cada um dos possíveis valores que este atributo assume no conjunto:

$$Ganho(S, A) = Entropia(S) - \sum_{V \in valores(A)} \frac{\|S_v\|}{\|s\|} Entropia(S) \tag{2.3}$$

2.6.2 Informação Mútua

A Informação Mútua (IM) é uma outra medida que busca determinar o quanto de informação a presença ou a ausência de um termo carrega a respeito do pertencimento de um documento a uma classe. Um resumo é dado por Manning, Raghavan e Schütze (2008) ao informar que, se a distribuição do termo *t* é a mesma tanto na coleção como um todo como numa classe específica *c*, então a medida da informação mútua entre o termo e a classe é 0 ($A(t, c) = 0$). Essa medida atingirá seu maior valor quando o termo *t* ocorrer em um documento se e somente se o documento pertencer à classe c. Portanto, pode-se interpretar o valor da Informação Mútua para ordenar as *features* pela utilidade que têm para diferenciar as classes, com valores maiores de IM representando maior utilidade.

2.6.3 *Chi-square*

Semelhante a informação mútua, num problema de classificação, a métrica *Chi-Square* computa a coocorrência observada entre cada feature e a classe do documento e a coocorrência esperada deste mesmo par, assumindo que elas fossem independentes. A partir da comparação desses dois valores, a métrica pode ser interpretada para saber se não há ou não relevância da feature para separar a classe.

2.6.4 Análise Semântica Latente

A Análise Semântica Latente (do inglês, *Latent Semantic Analysis* - LSA) é uma técnica que busca criar representações significativas de conjuntos de textos de maneira não supervisionada. A teoria do LSA endereça o problema de como o significado de palavras e de passagens de textos é construído a partir da experiência com a linguagem. Segundo Landauer et al. (2007), o LSA demonstra um método computacional através do qual uma parte significativa do aprendizado e uso da linguagem pode ser atingido. Conforme o autor, essa teoria permite o desenvolvimento de sistemas computacionais que realizam, frequentemente bem, uma série de tarefas cognitivas desenvolvidas pelos humanos.

O LSA se baseia numa premissa que Landauer et al. (2007) chama de "composicionalidade" (*compositionality*) que supõe que a representação de qualquer passagem texto com algum sentido deve ser composta como uma função da soma da representação das

palavras que ela contém. Para exemplificar essa ideia de indução dos sentidos na composição dos textos, o autor apresenta as seguintes equações algébricas *A + 2B = 8* e *A + B = 5* e revela que, assim como sabemos que nenhuma dessas duas equações individualmente revela o valor de *A* ou de *B*, quando analisamos as duas juntas, obtemos o valor de ambas as variáveis (respectivamente, 2 e 3). Portanto, o LSA assume que há relações semânticas entre palavras que não são inicialmente explícitas, mas que podem ser descobertas quando se analisa uma amostra suficientemente grande de documentos.

No contexto de PLN o LSA é realizado sobre uma matriz *m x n*, onde *m* são os documentos de um *corpus* e *n* os termos do vocabulário, como descrito seção 2.5.1. Sobre essa matriz aplica-se a técnica de álgebra linear *Singular value decomposition* - SVD, que segundo Manning, Raghavan e Schütze (2008) fatora uma matriz M qualquer em um produto de 3 matrizes individuais: $M = U * S * V$, onde S é a matriz diagonal dos valores singulares de M e representa os tópicos descobertos na matriz original. Tomando-se os K maiores valores da matriz S, é possível identificar os K tópicos mais frequentes e, a partir da análise das outras duas matrizes decompostas, identificar os documentos em que estão presentes, bem como os termos que mais se relacionam a eles.

Sendo K um valor menor do que a quantidade de linhas originais da matriz M, diz-se que o resultado é uma versão *truncada* dos valores singulares, motivo pelo qual a técnica é referida como *Truncated Singular Value Decomposition* e considerada também uma técnica de redução de dimensionalidade.

2.6.5 *Quantile Transformer*

Transformações de *features* realizam conversões nos valores das *features* alterando o espaço vetorial em que elas estão representadas. Uma das possíveis transformações é a de escala dos valores apresentados. Segundo Roy (2020), essa transformação de escala pode afetar significativamente os resultados de alguns algoritmos de Aprendizado de Máquina e pode não ter efeito algum em outros.

Os algoritmos de ML operam apenas com números e se houver uma grande diferença de escala entre eles alguns dos algoritmos - basicamente aqueles baseados no cálculo de distâncias no espaço vetorial - tendem a assumir que valores maiores têm alguma superioridade. Considera-se nesse caso que uma variável com valores maiores “domina” outras com valores menores.

Além do problema de domínio de algumas variáveis sobre outras, quando usa-

das redes neurais com a função de ativação sigmoide, há também um problema chamado "saturação" Roy (2020). Nesse caso, quando os valores de entrada são muito grandes, a saída da função sigmoide tende a ser muito próxima de zero ou de um, fazendo com que o gradiente a ser aplicado na rede seja muito pequeno, impedindo ajustes relevantes do pesos. A reescala de *features* tende a fazer com o que o algoritmo de gradiente descendente, usado na redes neurais, convirja mais rapidamente ao ponto ótimo, segundo Ng (2020).

No caso do *Quantile Transformer*, o valor das *features* é transformado para seguir uma distribuição normal ou uniforme. Assim, para uma determinada *feature*, essa transformação tende a espalhar os valores mais frequentes. Isso também ajuda a reduzir o impacto dos chamados *outliers*, que são valores muito distantes da maioria dos outros valores.

2.6.6 Select K-best

Quando se tem um conjunto de dados com amostras de diferentes classes e busca-se encontrar as características dos dados que melhor separem essas classes, podem ser feitas avaliações que usam cada variável individualmente, comparando-as com a classe-alvo para verificar se há algum relacionamento estatisticamente significativo entre elas ou analisar as amostras como um todo em comparação com a classe-alvo, possivelmente projetando os atributos num espaço vetorial menor usando técnicas como o PCA[14]. No primeiro caso, temos a chamada análise "univariada", assim chamada porque usa uma variável por vez, e no segundo temos uma análise "multivariada".

Uma seleção de *features* univariada funciona aplicando a análise estatística univariada aos dados e ranqueando as *features* pelo valor do teste aplicado. A técnica *Select K-best* é um exemplo desse tipo, onde o procedimento é executado e as K melhores *features* são selecionadas. Dois parâmetros, portanto, são relevantes aqui: o valor de K, que determina quantas serão as *features* resultantes da seleção, e o teste estatístico aplicado, que indica como as variáveis serão avaliadas quanto à sua importância para a separação da classes. Exemplos de tipos de testes usados aqui são os já mencionados, Informação mútua e o *Chi-squared*.

[14] A Análise de Componentes Principais ou PCA (do inglês *Principal Component Analysis*) serve para sumarizar a informação de múltiplas variáveis originais num conjunto menor de componentes, sem que isso implique em perda significativa de informação. Uma explicação detalhada pode ser encontrada em Vasconcelos Simone; Conci (2020)

2.7 Balanceamento de Classes

Um problema para o Aprendizado de Máquinas Supervisionado é que em vários conjuntos de dados reais o número de objetos para cada classe apresenta frequências diferentes, fazendo com que haja mais exemplos de uma classe do que de outra. Nesses conjuntos ditos desbalanceados, em se tratando de problemas de duas classes, a classe com mais exemplos é chamada de *majoritária* enquanto a outra é chamada *minoritária*.

Segundo Monard e Batista (2002), frequentemente classificadores têm boa acurácia para a classe majoritária, mas sua acurácia para a classe minoritária é inaceitável. Isso se torna um problema quando o custo de uma classificação errada na classe minoritária é muito maior do que o custo de uma má classificação na classe majoritária. Sobre isso, Faceli (2011) menciona que, para ser aceitável, a acurácia preditiva de um classificador para um conjunto de dados desbalanceados deve ser maior do que a acurácia obtida atribuindo-se todo novo objeto à classe majoritária.

Para lidar com o problema de conjuntos de dados desbalanceados, normalmente são adotadas estratégias de balanceamento (em inglês, *resampling*), que tentam balancear artificialmente os dados. Uma das técnicas comuns de *resampling* é redefinir o tamanho do conjunto de dados, o que pode ser feito tanto com o acréscimo de novos exemplos na classe minoritária, dito *over-sampling*, quanto com a eliminação de exemplos da classe majoritária, dito *under-sampling*.

Um exemplo de aplicação de *over-sampling* é o *Random Over-sample*, que, conforme sugere o nome, escolhe aleatoriamente exemplos da classe minoritária e os replica no conjunto. A técnica é, basicamente, um exemplo de amostragem aleatória com reposição. Outras técnicas mais complexas podem ser usadas e as seções a seguir discutem algumas delas.

2.7.1 *Synthetic Minority Over-sampling Technique*

A técnica *Synthetic Minority Over-sampling Technique* - (SMOTE), Chawla et al. (2002), realiza o *over-sampling* criando novos exemplos através de variações nos atributos dos exemplos existentes.

Para fazer o *over-sampling*, o SMOTE seleciona um exemplo aleatório *P* da classe minoritária e seus K vizinhos mais próximos. Para cada novo ponto que deseja gerar tendo como referência o exemplo *P*, escolhe um dos vizinhos k_{i}, computa a diferença

entre os vetores de atributos de *P* e k_{i} e multiplica essa diferença por um valor aleatório entre 0 e 1, gerando um novo exemplo em algum lugar entre *P* e k_{i}.

2.7.2 *Borderline-SMOTE*

Existem inúmeras variações do algoritmo SMOTE, uma delas é o *Borderline-SMOTE*, Han, Wang e Mao (2005). Essa variação faz *over-sampling* apenas de exemplos da classe minoritária localizados na área em torno dos limites da classe, onde as classes se sobrepõem, chamada de *borderline*. Encontrar esses exemplos limítrofes (*borderline examples*) é o primeiro passo desse algoritmo. Em seguida, tomando individualmente cada um deles, o algoritmo os avalia de forma que, se todos os seus vizinhos forem da classe majoritária, ele é tido como um exemplo com muito ruído e não é usado no *over-sampling*. De igual forma, são descartados exemplos cuja maioria dos vizinhos é da classe minoritária, pois são tidos como exemplos fáceis de classificar, logo, duplic á-los ajudaria pouco. Se mais da metade dos vizinhos de um *borderline example* for da classe majoritária, ele é tido como exemplo "perigoso" (com grande chance de ser mal classificado) e é portanto escolhido como base para o *over-sample*. Depois de definidos os candidatos, o SMOTE é aplicado para realizar o *over-sampling*.

2.7.3 *K*-means SMOTE

Outra variação possível do SMOTE é o algoritmo, de K*-means SMOTE* Douzas, Bacao e Last (2018). Ele opera em três passos: Clusterização, Filtragem e *over-sampling*. No primeiro passo, K grupos são criados usando o algoritmo K-means, que agrupa exemplos pela proximidade no espaço de características. O passo da filtragem seleciona os *clusters* em que ocorrerá *over-sampling*. Essa seleção é feita com base na taxa de desequilíbrio entre as classes (*imbalance ratio*) dos exemplos no *cluster*. Por fim, o último passo aplica o algoritmo SMOTE nos *clusters* cujo *imbalance ratio* indica que há mais exemplos da classe majoritária que da minoritária.

As duas primeiras etapas, que diferenciam o *K-means SMOTE* do SMOTE *regular*, fazem com que os novos exemplos sejam criados em regiões "seguras" evitando, por exemplo, que um elemento da classe minoritária, que é um *outlier*, cercado de exemplos da classe majoritária, seja escolhido como base para criar novos exemplos. Da mesma

forma, evita duplicar exemplos da classe minoritária em *cluster* onde ela já predomina e, portanto, novos exemplos contribuiriam pouco para separar as classes.

2.8 Algoritmos de Aprendizado de Máquina

Em seu capítulo sobre Modelos Preditivos, Faceli (2011) explica que um algoritmo de Aprendizado de Máquina é uma função que, dado um conjunto de exemplos rotulados, constrói um estimador. No caso de um conjunto de valores nominais, tem-se um problema de classificação e o estimador gerado é um classificador (esta é a ideia que foi apresentada na Figura 2.2).

Segundo Faceli (2011), cada algoritmo de ML possui dois vieses, um de representação e um de busca. Cada algoritmo utiliza uma forma de representação para descrever a hipótese induzida, por exemplo, valores reais, associados aos pesos das conexões em uma rede neural artificial ou a estrutura de árvore com nós externos associados às classes, nas árvores de decisão. Essa forma de representação é o *viés de representação*. O *viés de busca* de um algoritmo é a forma como ele procura a hipótese que melhor se ajusta aos dados de treinamento, por exemplo, algoritmo ID3[15], utilizado para a indução de árvores de decisão, tem como viés de busca a preferência por árvores de decisão com poucos nós.

A hipótese encontrada por um algoritmo para separar os dados estabelece uma *Fronteira de Decisão* que separa os exemplos de uma classe da outra. Em função de seus vieses, cada algoritmo tende a encontrar diferentes fronteiras de decisão. Além disso, diferenças no conjunto de treinamento, variações na ordem de apresentação dos dados durante o treinamento e processos internos podem fazer com que um mesmo algoritmo de ML encontre fronteiras diferentes a cada novo treinamento Faceli (2011).

Nas seções seguintes, os algoritmos de Aprendizado de Máquina utilizados nesse trabalho são brevemente apresentados.

2.8.1 Máquinas de Vetor de Suporte

Máquinas de Vetor de Suporte (ou *Support Vector Machines* – SVM) são algoritmos que tentam encontrar uma boa função linear para separar elementos de duas classes. O algoritmo procura encontrar uma fronteira, dita hiper-plano, que maximize a margem

[15] https://en.wikipedia.org/wiki/ID3_algorithm

entre ela e os elementos que pertencem à classe e os que não pertencem. Também é possível trabalhar com dados cuja separação linear não seja possível, mas para isso é necessário realizar uma certa transformação no espaço das características, tipicamente criando uma nova dimensão, facilitando assim a separação dos dados em classes distintas. Essa manipulação é feita via "funções de *kernel*" ("truque do *kernel*") e uma das opções comuns é a *Radial Basis Function* (RBF). Segundo SCIKIT-LEARN developers (2017), entre as vantagens do SVM estão o fato de que ele é muito efetivo em espaços multidimensionais, que como já visto é o caso do problema da classificação de textos; ele mantém sua efetividade mesmo quando o número de amostras é menor que o número de características e é bastante versátil, graças as possibilidades de funções de *kernel* que podem ser variadas.

Vale destacar que o SVM é um tipo de classificador que aceita hiper-parâmetros, tais como as mencionadas funções de *kernel*, que são parâmetros que não podem ser diretamente derivados dos dados de treinamento. Esses hiper-parâmetros normalmente são determinados através de exaustivos testes em que seus valores são variados, o modelo é treinado usado um subconjunto do conjunto de dados de treinamento e sua precisão é avaliada usando o restante dos dados para teste. Após isso, novos ajustes são feitos nos hiper-parâmetros e os resultados são novamente avaliados, repetindo-se até que se chegue a um valor aceitável. Bibliotecas como a SCIKIT-LEARN developers (2017) oferecem funcionalidades (*Grid Search* e *Random Search*) para explorar a matriz de combinações possíveis desses hiper-parâmetros e selecionar os melhores resultados.

2.8.2 Árvores de Decisão

De maneira geral, as Árvores de Decisão (*Decision trees*) utilizam particionamento hierárquico do espaço de características para determinar o pertencimento de documentos às classes. Algoritmos dessa categoria particionam recursivamente o conjunto de treinamento em subconjuntos menores com base em testes executados em cada nó. Cada nó da árvore corresponde a um teste do valor de algum atributo do exemplo de treinamento e cada ramo descendente desse nó corresponde a um valor possível desse atributo, segundo explica Allahyari et al. (2017).

No caso da Classificação de Textos, em tempo de treinamento, uma árvore pode ser construída, por exemplo, separando-se os documentos entre os que têm e os que não têm uma determinada característica (termo). Depois, para cada subconjunto daí determinado, o mesmo processo de divisão é repetido usando-se outro termo. Um método utilizado para

determinar quais termos devem ser usados para quebrar o conjunto pode ser, por exemplo, o Ganho de Informação, discutido na seção 2.6.1, diminuindo ao máximo a entropia a cada nível da árvore. Uma vez construída a árvore durante a fase de treinamento, um novo documento é classificado aplicando-se cada condição (predicado) desde a raiz da árvore do modelo até que ele seja enquadrado em uma folha da árvore.

2.8.3 Modelos Múltiplos Preditivos

Modelos Múltiplos Preditivos ou *ensembles*, segundo Faceli (2011), são um termo usado para designar um conjunto de preditores distintos (ditos Preditores-base) cujas decisões individuais são combinadas ou agregadas de alguma forma.

A motivação para essas combinações é a ideia de que diferentes algoritmos têm diferentes vieses de busca e de representação e, portanto, induzem hipóteses distintas. Se esses classificadores cometem erros independentes, então uma combinação deles pode levar a um erro menor do que o obtido com qualquer um deles individualmente.

Dois aspectos importantes na criação de *ensembles* são a maneira como os resultados são combinados e a maneira como os preditores base são induzidos. Em relação ao primeiro aspecto, *métodos de votação* são uma solução utilizada quando a saída do classificador base é usada num problema de classificação. As combinações, nesse caso, podem ser de *votação uniforme*, com todos os classificadores base contribuindo igualmente para o resultado ou de *votação com peso*, onde cada classificador base tem um peso associado, pelo qual seu voto é ponderado.

Quando os múltiplos preditores base de um *ensemble* são implementados pelo mesmo algoritmo, Faceli (2011) refere-se a eles como uma combinação de classificadores homogêneos. Se os classificadores são os mesmos, uma maneira de induzir resultados diferentes é manipulando os dados a eles submetidos.

Entre essas possíveis manipulações nos dados está o *Bootstrap Aggregating* (às vezes chamado só de *Bagging* ou *Bootstrap*), apresentado em Breiman (1996). Nessa técnica, várias réplicas do conjunto de treinamento são criadas a partir de uma amostragem aleatória com reposição do conjunto de treinamento original. Todas as réplicas têm a mesma quantidade de exemplos, mas nem todos os exemplos do conjunto original aparecem em cada uma e, para manter o tamanho constante, alguns exemplos são repetidos em cada réplica. Depois da geração dessas réplicas, cada um dos preditores base é treinado com uma delas e o resultado final do *ensemble* é tipicamente uma combinação por votação

uniforme. Florestas Aleatórias, discutidas a seguir, podem se valer desse método.

Outra opção para manipular os dados é a técnica de *Boosting*. A ideia principal por trás do algoritmo de *boosting* é associar um peso a cada exemplo de treinamento, refletindo sua importância. Um ajuste desses pesos de maneira distinta leva a classificadores diferentes, como explica Faceli (2011). Um exemplo de aplicação desse algoritmo é o AdaBoost, apresentado adiante.

2.8.3.1 Adaptative Boosting - AdaBoost

O algoritmo AdaBoost, segundo Faceli (2011), gera uma conjunto de classificadores que participam da classificação de exemplos de teste por votação ponderada. O algoritmo gera os classificadores sequencialmente, fazendo a cada iteração uma mudança no peso dos exemplos, levando em consideração o erro do conjunto de classificadores construído na iteração anterior.

Iniciando com um peso igual para todos os exemplos, o algoritmo diminui o peso dos exemplos que já foram corretamente classificados e aumenta o peso daqueles cuja classificação foi incorreta. O efeito disso é que os classificadores induzidos na iteração seguinte, buscando minimizar o erro, tendem a se concentrar nos exemplos incorretamente classificados anteriormente, já que a penalização por classificá-los mal (dada pelo peso) é maior.

Ao longo de várias iterações, a combinação das fronteiras de decisão que vão sendo desenhadas em cada uma delas acaba por delinear uma fronteira mais complexa, que classifica melhor o conjunto de exemplos como um todo, pois apreende a lidar com os seus diferentes graus de complexidades.

2.8.3.2 Florestas Aleatórias

Florestas Aleatórias (do inglês, *Random Forests*) são basicamente um *ensemble* de Árvores de Decisão, apresentadas na seção 2.8.2. A novidade aqui é que esse algoritmo, para induzir árvores distintas, se vale de estratégias como o já mencionado *Bootstrap Aggregating* combinado com outras que criam variações nos atributos considerados ao analisar um exemplo.

Além de introduzir variabilidade nos dados pela seleção de exemplos com uso do *bootstrap* o algoritmo também faz uma seleção aleatória de atributos a serem considerados em cada nó. A escolha do atributo a ser testado em cada nó ainda pode usar as mesmas

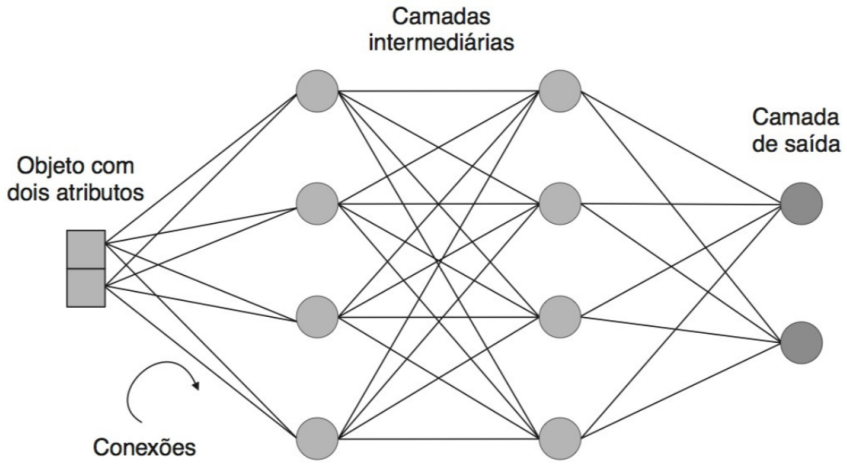
métricas usadas nas árvores de decisão. A diferença é que aqui o conjunto de atributos dentre os quais se irá escolher é menor, sendo determinado entre aqueles aleatoriamente escolhidos para serem considerados nessa árvore me particular.

2.8.4 Redes Neurais Artificiais

Uma Rede Neural Artificial (RNA) é um modelo computacional inspirado na estrutura do sistema nervoso, com o objetivo de simular a capacidade de aprendizado do cérebro humano. Segundo Faceli (2011), elas são caracterizadas por dois aspectos básicos: arquitetura e aprendizado. Enquanto a arquitetura está relacionada ao número de unidades de processamento (neurônios), ao seu tipo e à forma como os eles estão conectados, o aprendizado diz respeito às regras usadas no ajuste dos pesos da rede.

Em uma RNA, um neurônio recebe um valor na entrada, processa-o em uma função matemática (função de ativação) e produz um valor em sua saída. Esse neurônios podem estar dispostos em uma ou mais camadas (rede multicamadas). Em redes multicamadas um neurônio recebe como entrada valores de saída de neurônios da camada anterior e envia sua saída aos neurônios da camada seguinte.

Figura 2.4: Exemplo de RNA multicamadas típica



Fonte: Faceli (2011)

Em qualquer caso, a camada final da rede é chamada camada de saída. Redes em que o processamento se dá de maneira sequencial, com os dados fluindo da camada de entrada, através das camadas intermediárias, para a camada de saída são conhecidas como *feed-forward networks*. A abordagem mais utilizada para definição da arquitetura de uma RNA é a empírica, segundo Faceli (2011). Nessa abordagem diversas arquiteturas

são testadas e comparadas até que se encontre uma RNA cuja acurácia preditiva seja adequada.

Com relação ao aprendizado, há diferentes algoritmos para o ajuste dos parâmetros de uma RNA. No caso de aprendizado supervisionado, geralmente é utilizado um algoritmo de correção de erro. Um exemplo desse tipo de algoritmo é o *back-propagation*, que tem duas fases. Na primeira (*forward*) um objeto de exemplo é apresentado à rede e classificado. O erro é então determinado pela diferença entre valor obtido na classificação e o valor esperado. A partir disso, inicia-se a segunda fase (*backward*) em que o valor do erro é utilizado para ajustar os pesos dos neurônios desde a camada de saída até a primeira camada intermediária.

Na fase de ajuste de pesos, os valores são ponderados em função de uma taxa de aprendizado. Essa taxa tem forte influência no tempo necessário para a convergência da rede. Se ela for muito pequena, podem ser necessárias muitas iterações para induzir um bom modelo, se for muito alta ela pode provocar saltos nos valores dos pesos que dificultam muito a convergência. Para lidar com esse problema, alguns "otimizadores" podem ser aplicados. Um exemplo bastante referido na literatura é o *termo momentum*, que quantifica o grau de importância da variação do peso do ciclo anterior em relação ao peso atual sendo aplicado, evitando grandes oscilações. Outras opções aqui são o algoritmo Adam, visto em Kingma e Ba (2015), o Adadelta, apresentado em Zeiler (2012) e o RMSprop, descrito em Hinton (2012).

Por padrão, o algoritmo de *back-propagation* ajusta os pesos de uma RNA a cada exemplo apresentado. Uma variação possível é o chamado treinamento em lote (*batch*), em que um lote de N exemplos é apresentado à rede para classificação. Após todos os exemplos desse lote terem sido classificados, um erro médio é calculado e esse é o valor usado para ajustar os pesos da rede.

Durante o treinamento, os ciclos de apresentação de exemplos e ajuste de pesos são repetidos indefinidamente, até que algum critério de parada seja atingido. Diferente critérios podem ser adotados, por exemplo, um número máximo de iterações ou uma taxa mínima de erros. Também é comum que o treinamento tenha um conjunto de dados para validação. Esses dados são apresentados a cada N ciclos da rede para avaliar sua taxa de erro para exemplos até então desconhecidos. Se as taxas de erro para esses dados começam a crescer, tem-se um indicativo de *overfitting* e o processo de treinamento deve ser encerrado, isso é conhecido como validação cruzada com *early stop*[16].

[16]https://developers.google.com/machine-learning/glossaryearly-stopping

2.8.5 Redes Neurais Recorrentes

O modelo de RNA *feed-forward* descrito na seção anterior tem em vista um conjunto de dados que é multidimensional, mas em que os atributos são amplamente independentes entre si. Entretanto, textos são um tipo de dado que apresenta dependências sequenciais entre seus atributos e o seu processamento pode, portanto, se beneficiar do conhecimento dessa estrutura. Por conta disso, segundo Aggarwal (2018), é interessante que os modelos construídos para processamento de textos levem em conta a ideia de relação sequencial dos atributos.

Redes neurais recorrentes (RNN) têm uma estrutura particular baseada na ideia de temporalidade capaz de capturar esse tipo de relação. Diferentemente das redes *feed-forward*, onde a ativação dos neurônios se propaga em uma única direção, nas redes recorrentes a saída de um neurônio pode se propagar tando para a entrada de neurônios da camada seguinte quando da própria camada ou de camadas anteriores. Segundo explica Aggarwal (2018), essa arquitetura cria *loops* dentro da rede que funcionam como memória para os neurônios (*memory state*), permitindo que eles possam "lembrar o que aprenderam até então".

2.8.6 Gated Reccurent Unit

A memória dos neurônios nas RNN dá a elas uma vantagem sobre as redes *feed-forward*, pois com esse mecanismo as relações sequenciais dos dados podem ser capturadas. Contudo, segundo explica Aggarwal (2018), as RNN sofrem com o problema do *Vanishing Gradient* - quando valores de ajuste dos pesos tornam-se muito pequenos - e, para lidar com esse problema, diferentes tipos de RNN foram propostos, entre eles as *Long Short-Term Memory* (LSTM) e as *Gated Reccurent Unit* (GRU).

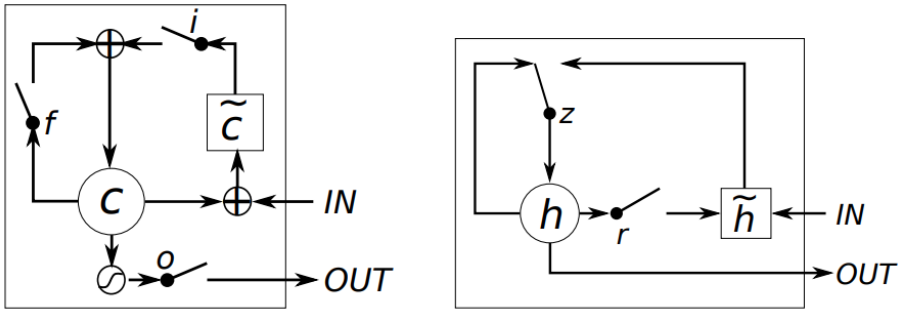
Enquanto as RNN convencionais atualizam sua memória apenas combinando o estado prévio com a entrada corrente e os submetendo a uma função não-linear, as redes LSTM e GRU atualizam sua memória de maneira ponderada através de mecanismos chamados de *gates*, que definem qual informação será lembrada, passando para o estágio seguinte, ou esquecida, deixando de ser propagada a partir do estágio atual.

O primeiro dos dois tipos a ser proposto foi o das LSTM, cuja chave para o funcionamento, conforme Olah (2015), é a célula de memória (*cell state*) usada para a propagação do estado, que flui sequencialmente entre as camadas. As LSTM possuem três *gates*

que realizam individualmente o controle do estado na *cell state*: O *forget gate* decide qual informação do estado atual deve ser descartada. Isso é feito analisando o estado atual e a nova entrada recebida neste estágio do processamento. O *input gate* decide qual informação da entrada atual deve ser adicionada ao estado que será propagado. Aqui também são considerados o estado atual e a entrada corrente. Após a atualização do *cell state* com aquilo que deve ser lembrado e esquecido, o *output gate* computa a saída da célula, com base no estado já atualizado e na entrada corrente. A saída de ativação desse estágio e o *cell state* atualizado são então propagados para o estágio seguinte.

As GRU são bastante similares às LSTM, sendo em certa medida uma simplificação delas, segundo Aggarwal (2018). As GRU também funcionam com base em *gates* para modular o fluxo de informação entre as células, porém, elas não possuem uma *cell state* independente e têm apenas dois *gates*: *reset* e *update*. O *reset gate* é usado para combinar a entrada corrente com a ativação anterior, gerando uma "ativação candidata". O *update gate* atua como uma combinação do *input* e do *forget gate* das LSTM e determina o quanto do estado anterior influencia sua ativação corrente. A ativação candidata e a ativação anterior passam posteriormente por ele gerando a ativação corrente, que é então enviada diretamente para a saída. A Figura 2.5 ilustra as diferenças entre os dois tipos de RNN.

Figura 2.5: Célula de redes LSTM e GRU



(a) Long Short-Term Memory (b) Gated Recurrent Unit

Ilustração de (a) LSTM e (b) GRU. Em (a) i, f e o são o *input, forget* e *output gates*, respectivamente, enquanto c e $\tilde{c}$ denotam o conteúdo atual e o novo conteúdo do *cell state*. Em (b) r e z são o *reset* e o *update gates*, e h e $\tilde{h}$ são a ativação e a ativação candidata. Fonte: Chung et al. (2014)

2.9 Otimização de hiperparâmetros

No contexto do Aprendizado de Máquina, pode-se considerar como *parâmetro* uma variável do modelo que o sistema de ML aprende por conta, conforme Google (2020). Por exemplo, os pesos de uma rede neural são um exemplo de parâmetro que o sistema aprende através de sucessivas iterações de treinamento. Em contraste com isso estão os *hiperparâmetros*, que são configurações que o desenvolvedor do modelo varia em cada iteração buscando encontrar o seu melhor valor. Por exemplo, o *learning rate*[17] de uma rede neural é um exemplo de hiperparâmetro.

Encontrar bons valores de hiperparâmetros é um processo referido como otimização de hiperparâmetros e conforme Bernico (2018) é, em última instância, um problema relacionado ao custo de desenvolvimento do modelo. Segundo o autor, enquanto teoricamente se pode procurar exaustivamente o melhor valor para cada configuração, na prática é preciso considerar o orçamento do projeto, em termos de tempo de computação e tempo dos desenvolvedores do modelo. Isso faz da otimização de hiperparâmetros um campo aberto e ativo de pesquisa, com diferentes estratégias sendo propostas para o tratamento do problema.

Estratégias comuns para a otimização de hiperparâmetros segundo Bernico (2018) incluem: o *Grid Search*, que consiste em testar todas as possíveis combinações entre diferentes valores de hiperparâmetros dentro de um certo espaço de busca; o *Random Search* em que alguns valores aleatórios dentro de um espaço de busca são escolhidos e então testados; métodos de *Bayesian Optimization*, que também funcionam amostrando exemplos dentro de um certo espaço de busca, mas em vez de o fazerem de forma aleatória, utilizam os resultados de testes anteriores para determinar os próximos valores candidatos a serem amostrados; e o *Machine Learning for Hyperparameters*, um campo recente de pesquisa segundo o autor, que consiste no treinamento de redes neurais que podem predizer os melhores parâmetros para uma dada arquitetura de rede.

Entre os hiperparâmetros que comumente precisam ser otimizados em redes neurais pode-se destacar, além do *learning rate*, o tamanho do lote, que termina quantos exemplos considerar no cálculo do erro antes de ajustar os pesos para a próxima iteração.

Para a configuração do *learning rate* é possível utilizar "otimizadores", que são componentes do modelo que implementam algum algoritmo para controlar dinamica-

[17] Um valor que indica quanto do erro da rede (a diferença entre o valor predito e o esperado) deve ser considerado ao ajustar os pesos para a próxima iteração do treinamento.

mente a taxa de aprendizado, acelerando o treinamento e minimizando a chance de uma eventual convergência em um ótimo local. Nesses casos, os hiperparâmetros do otimizador é que precisam ser escolhidos pelo desenvolvedor do modelo, em geral pela determinação de um bom valor inicial de seus parâmetros.

Uma outra técnica usada em redes neurais que requer otimização de hiperparâmetros é a regularização. O objetivo da regularização é minimizar *overfitting*, reduzindo o número de *features* que o modelo precisa considerar e fazendo com que a rede aprenda sem considerar necessariamente uma coocorrência entre *features* que pode não se repetir fora dos dados de treinamento. Um exemplo de implementação de regularização é a técnica de *dropout*, que funciona através do descarte (conversão para zeros) de certos atributos do exemplo sendo processado em cada passo do treinamento. Nesse caso, o percentual de atributos a ser descartado é o hiperparâmetro a ser otimizado pelo desenvolvedor do modelo.

Ao tratar de maneira prática a questão da otimização de hiperparâmetros, Bernico (2018) indica duas possíveis estratégias, complementares entre si, para abordar a questão: Encontrar um problema semelhante que tenha sido resolvido de maneira satisfatória e utilizar as mesmas configurações como ponto de partida para a criação de um novo modelo; e "expandir até o sobreajuste e então regularizar", isto é, tomando como ponto de partida um bom exemplo de configuração de modelo para um problema semelhante, adicionar novas camadas à rede, mais neurônios, etc. Quando a rede atingir um ponto de *overfitting*, parar de expandir a sua complexidade e investir nas técnicas de regularização.

2.10 Avaliação de Modelos de Aprendizado de Máquina

Uma questão a se levar em conta na construção de Modelos de Aprendizado de Máquina é que é preciso testá-los com os dados do caso concreto em que o modelo será aplicado. Segundo Faceli (2011), de maneira geral, pode-se afirmar que não é possível estabelecer *a priori* que uma técnica de ML em particular se sairá melhor que qualquer outra na resolução de um certo tipo de problema. É preciso, portanto, comparar os diferentes modelos para escolher o mais adequado ao problema em questão.

Pode-se comparar modelos sob vários aspectos, por exemplo, explicabilidade, tempo de aprendizado ou desempenho preditivo. Esse último aspecto costuma ser um dos mais importantes na seleção do modelo e, no caso de problemas de classificação, é avaliado através de métricas de erro computadas pelo resultado do classificador ao rotular

exemplos não apresentados ele em tempo de treinamento.

2.10.1 Matriz de Confusão

Uma maneira de visualizar as medidas de desempenho de um classificador é por meio de uma *matriz de confusão*. Nessa matriz uma das dimensões representa a classe verdadeira (já conhecida) e a outra a classe predita pelo classificador. Normalmente coloca-se nas linhas a classe verdadeira e nas colunas a classe predita. Nesse caso, em uma matriz M cada um de seus elementos m_{ij} apresenta o número de exemplos da classe *i* classificados como pertencentes à classe *j*.

Em problemas de duas classes, uma delas é dita classe Positiva e a outra classe Negativa. Da observação da matriz de confusão pode-se extrair diferentes medidas referentes a cada classe, bem como algumas métricas compostas. A lista abaixo apresenta algumas dessas medidas.

Precisão

Proporção de exemplos positivos classificados corretamente entre todos aqueles preditos como positivos.

Sensibilidade ou revocação

Corresponde à taxa de acerto na classe positiva. Também é chamada de Taxa de Verdadeiros Positivos (TVP).

Especificidade

Corresponde à taxa de acerto na classe negativa. Seu complemento corresponde à Taxa de Falsos Positivos.

Valor preditivo negativo

É uma espécie de *precisão na classe negativa*, isto é, corresponde à proporção de exemplos negativos classificados corretamente, entre todos os preditos como negativos.

Taxa de erro na classe positiva

É a proporção de exemplos da classe positiva incorretamente classificados. Também conhecida como Taxa de Falsos Negativos (TFN).

Taxa de erro na classe negativa

Corresponde à proporção de exemplos da classe negativa incorretamente classificados. Também conhecida como Taxa de Falsos Positivos (TFP).

Acurácia

Soma dos valores da diagonal principal da matriz pela soma de todos os elementos da matriz.

Medida F1

Corresponde a uma média harmônica ponderada da precisão e da revocação, na qual é dado o mesmo grau de importância a essas duas medidas.

Taxa de Descobertas Falsas

É o complemento da Precisão. Dito de outra forma: dentre todos os exemplos classificados como positivos, quantos eram Falsos Positivos.

A Taxa de erro na classe negativa, também conhecida como *Taxa de Falsos Positivos* (TFP), e a Taxa de acerto na classe positiva, também conhecida como *Taxa de Verdadeiros Positivos* (TVP), são particularmente importantes para a criação de Curvas ROC, discutidas a seguir.

2.10.2 Curvas ROC

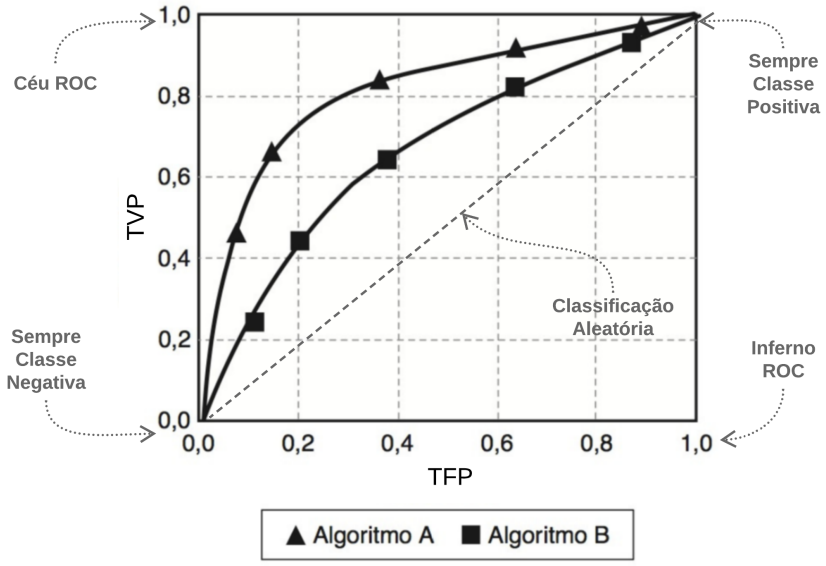
As métricas TVP e TFP, discutidas previamente, podem ser alisadas conjuntamente em um gráfico chamado *Receiving Operating Characteristics* (ROC), cujos princípios são discutidos em Metz (1978). Trata-se de um gráfico bidimensional plotado em um espaço denominado "Espaço ROC", onde o eixo X representa a Taxa de Falsos Positivos e o eixo Y a Taxa de Verdadeiros Positivos. Consideradas essas duas medidas, o desempenho de um classificador pode ser plotado como um ponto nesse espaço.

Embora seja possível comparar dois classificadores simplesmente por seus pontos no espaço ROC, a maneira mais usual é gerar uma *Curva ROC*. Para plotar uma dessas curvas é preciso que o algoritmo possa apresentar como resultado da classificação uma probabilidade de o exemplo pertencer a classe positiva. Tendo como saída do algoritmo esse valor contínuo, pode-se estabelecer diferentes limiares acima dos quais o exemplo seria classificado como positivo. Escolhendo-se diferentes limiares tem-se diferentes valores de TVP e TFP, permitindo que múltiplos pontos sejam representados no

espaço ROC, um para cada limiar. Ao se unir esses pontos têm-se uma Curva ROC para o classificador.

A Figura 2.6, ilustra um exemplo de comparação entre dois algoritmos através de suas Curvas ROC. Na figura também estão representados os principais aspectos do espaço ROC. A linha diagonal representa uma classificação aleatória. O ponto (0,1), chamado de Céu ROC, representa classificações perfeitas, isto é, todos objetos da classe positiva foram corretamente classificados como tal, assim como todos os objetos da classe negativa foram corretamente classificados (não houve Falsos Positivos). O Ponto (1,0), dito Inferno ROC, representa um classificador que erra todas as predições e os pontos (0,0) e (1,1) representam classificadores que aplicam sempre a mesma classe para qualquer exemplo.

Figura 2.6: Exemplo de Curvas ROC



Fonte: Adaptado de Faceli (2011)

Além da avaliação visual obtida na observação das Curvas ROC, uma medida objetiva usada para comparar os classificadores é a "Área Abaixo da Curva" ROC ou AUC (do inglês *Area Under ROC Curve*). A medida da AUC é um valor entre 0 e 1, sendo 1 o valor ideal. Assim, conforme explica Faceli (2011), ao comparar a AUC de dois classificadores, aquele com maior AUC é considerado superior em termos de desempenho preditivo.

2.11 Resumo do Capítulo

Nesse capítulo foi apresentada a Informática Médica como um campo científico que tem a informação biomédica como ponto central da disciplina, lidando com diferentes aspectos desta, indo desde a coleta e armazenamento até a sua aplicação em problemas de alto nível como a tomada de decisões clínicas.

Identificou-se o Aprendizado de Máquina como a capacidade de um algoritmo de melhorar o desempenho na realização de alguma tarefa por meio da experiência e mostrou-se que o treinamento supervisionado é uma das maneiras de atingir esse objetivo. Foi apresentada o noção geral do problema de classificação como sendo a capacidade de determinar a qual classe um determinado objeto pertence, dado um certo conjunto de classes, e descreveu-se a classificação de textos como um tipo particular deste problema. A intersecção destes temas, isto é, a Classificação de Textos usando o Aprendizado de Máquina Supervisionado foi sumarizada através Figura 2.2.

Demonstrou-se que o desempenho de algoritmos de Aprendizado de Máquina é geralmente afetado pelo estado dos dados e tratou-se da etapa de pré-cessamento como uma forma de preparar os dados para obter dos modelos de ML melhores resultados. Referiu-se que a etapa consome a maior parte do tempo do processo de classificação e um dos motivos para isso é que as várias opções de que se dispõe para realizar cada uma das transformações possíveis (eliminação de partes do texto, transformação dos *tokens*, formas de representação dos textos, etc) precisam sempre ser avaliadas no contexto do problema concreto sendo tratado.

Explicou-se que mesmo após o representação dos textos usando o VSM ainda é benéfico selecionar ou transformar cuidadosamente as *features* antes de injeta-las nos algoritmos de ML e que aqui também as opções são inúmeras, sendo necessário que se escolha a medida de utilidade a ser usada, os critérios e o método de avaliação e aplicá-los a cada *feature* individualmente ou a combinações ou projeções destas, requerendo mais uma vez um trabalho particular a cada conjunto de dados.

Verificou-se também que conjuntos de dados desbalanceados são problemáticos para a maioria dos algoritmos de ML e que técnicas de *resampling*, que tentam balancear artificialmente os dados, podem ser usadas para diminuir o impacto desse desequilíbrio sobre a indução dos classificadores.

Foram apresentados alguns algoritmos de aprendizado de máquina como o SVM, que tem entre suas vantagens a efetividade em espaços multidimensionais, como é ti-

picamente o caso do problema da classificação de textos, e as Árvores de Decisão que utilizam o particionamento hierárquico do espaço de características e apresentam entre suas vantagens a explicabilidade do modelo induzido. Também foi apresentada a ideia das combinações de algoritmos, os *ensembles*, e descreveu-se que a motivação para essa abordagem é a ideia de que diferentes algoritmos têm diferentes vieses de busca e de representação e que uma combinação deles pode levar a um erro menor do que o obtido por algoritmos individuais.

Ainda tratando de algoritmos de aprendizado, foram introduzidas as Redes Neurais Artificiais (RNA), como um modelo computacional inspirado na estrutura do sistema nervoso, com o objetivo de simular a capacidade de aprendizado do cérebro humano e mostrou-se que o funcionamento das redes neurais recorrentes têm uma estrutura particular baseada na ideia de temporalidade capaz de capturar relações de dependências sequencias, presentes nos dados textuais.

Por fim, mencionou-se a preocupação acerca do comportamento da solução diante dos dados do caso concreto deve ser considerada também na escolha do algoritmo e por isso faz-se necessário comparar objetivamente os resultados dos classificadores. Para tanto, a ferramenta Matriz de Confusão é de grande utilidade, permitindo facilmente extrair medidas como a Taxa de Verdadeiros Positivos (TVP) e Taxa de Falsos Positivos (TFP).

Concluiu-se mostrando que as métricas TVP e TFP podem ser analisadas conjuntamente em um gráfico bidimensional no qual se pode plotar as chamadas Curvas ROC e a partir do cálculo da área sob elas, a ROC AUC, comparar dois classificadores determinando a superioridade daquele com maior valor desta medida.

3 TRABALHOS RELACIONADOS

O Processamento de Linguagem Natural (PLN) na área da saúde é uma área próspera de pesquisa. O trabalho de Demner-Fushman, Chapman e McDonald (2009) analisa os avanços das pesquisas de PLN associadas a sistemas de Apoio à Decisão Clínica - CDS (*Clinical Decision Support*), cuja definição adotada é:

> Qualquer software projetado para auxiliar diretamente na tomada de decisão clínica, na qual as características de cada paciente são comparadas a uma base de conhecimento computadorizada com o objetivo de gerar avaliações ou recomendações específicas do paciente, que são apresentadas aos médicos para consideração. Hunt et al. (1998).

Partindo dessa definição, ao analisar CDS com componentes de PLN, o trabalho reporta aplicações especializadas em: processamento de eventos clínicos (ex. interação entre drogas), processamento de relatórios de radiologia (ex. codificação automática de diagnósticos), processamento de relatórios de emergências médicas (ex. identificação de problemas respiratórios) e processamento de relatórios de patologias (ex. detecção de indicadores de câncer de mama).

Outro indício da prosperidade das pesquisas na área é a revisão sistemática sobre sistemas de PLN na área médica, produzida por Kreimeyer et al. (2017), que analisou trabalhos publicados em inglês, entre 2006 e 2016, e catalogou 71 sistemas diferentes. No trabalho foi observado que os objetivos desses sistemas cobriam uma ampla gama de tarefas da área clínica e um achado importante da pesquisa foi a observação de que quase todos os sistemas são (ou ao menos nasceram) dedicados a resolver um problema em específico.

Nos dois trabalhos recém mencionados é possível perceber um alinhamento das conclusões em relação do desenvolvimento da área. No primeiro, os autores concluem que mesmo onde há dados bem estruturados para a decisão clínica, há espaço para PLN aprimorar o cuidado dos pacientes e acreditam que os sistemas de PLN integrados a CDS continuarão a se desenvolver, guiados pelas necessidades particulares dos diferentes grupos de usuários (médicos e pesquisadores) envolvidos nas pesquisas. No segundo trabalho, após analisar a especificidade dos diversos sistemas, os autores entendem que a forma como a área pode evoluir é desenvolvendo ferramentas de PLN de qualidade para atender casos de usos específicos.

Na linha do desenvolvimento de componentes de PLN clínicos especializados, o trabalho Uzuner (2009) é uma importante referência, ao reportar diversos sistemas produ-

zidos para um *challenge* cujo o objetivo era identificar obesidade e outras comorbidades associadas a ela a partir de registros de alta de pacientes. Uma característica interessante desse *challenge* era a necessidade de definir se as menções às doenças eram explícitas ou intuitivas, isto é, dependiam de conhecimento do domínio para concluir a partir de outras informações se havia referências não explícitas à doença.

Alguns dos sistemas participantes se valeram de algoritmos baseados em regras, outros em Aprendizado de Máquina. Para a detecção das menções intuitivas, a maioria dos sistemas realizava primeiro a verificação das menções explícitas e se valia dessa informação, porém uma observação interessante dos resultados foi a de que o sistema com o melhor resultado na detecção de referências explícitas não ficou entre o ranking dos 5 melhores na detecção intuitiva, sugerindo que abordagens distintas deveriam ser empregadas para cada caso.

Consonante a isso, ao discutir os resultados do trabalho, os autores registram que as abordagens baseadas em regras tiveram um papel importante nos 10 melhores sistemas para detecção de menções explícitas, enquanto o as abordagens baseadas em ML contribuíram mais para os *top* 10 sistemas na detecção de menções intuitivas. Outra conclusão importante foi a de que, a despeito de a maioria dos participantes ter usado componentes existentes como base, muitos dos sistemas no *challange*, incluindo dois dos melhores, foram desenvolvidos a partir do zero.

Um componente comum a muitos dos sistemas que participaram do *challange* era o *POS-Tagger*. Particularmente em relação a esses componentes, é interessante observar que Oleynik et al. (2010), usando relatórios de alta de pacientes do Hospital de Clínicas de Porto Alegre (HCPA), compararam o desempenho de um *POS-Tagger* treinado com esses relatórios de alta a um outro treinado com base em textos de notícias jornalísticas[1]. Para o treinamento, os autores apresentam um método de "*Active learning*" que iterativamente refina e corrige os erros com base na decisão de um "comitê" de *taggers* e quando há discordância sobre as TAGS de uma sentença eles aplicam uma correção manual. O autores relatam que há divergência na literatura sobre a possibilidade de se usar *POS-Tagger*s treinados em datasets que não sejam da área médica em tarefas de PLN dessa área e as conclusões de seu trabalho confirmam as hipóteses de que isso não é possível, ao menos não sem comprometer significativamente os resultados.

Em um trabalho que adota uma abordagem baseada em regras, Oliveira et al. (2013) analisaram os mesmos relatórios de alta do HCPA com o objetivo de verificar a

[1] http://www.nilc.icmc.usp.br/macmorpho/

existência de referências à necessidade de continuar o tratamento após a alta. Para tanto, usaram um *POS-Tagger* e derivaram da estrutura sintática do texto quatro regras que seriam indicadoras da existência de sentenças que exprimem necessidade de continuidade do cuidado. Concluíram que o método serviria para identificar a informação que estão buscando e que ele poderia ser adaptado para buscar outras informações semelhantes.

Outro trabalho relevante para a discussão sobre o uso de diferentes tipos de textos na construção de componentes de PLN clínicos é o de Sarker e Gonzalez (2015), em que se buscam características textuais úteis ao monitoramento de Reações Adversas a Medicamentos (ADR). Nele os autores misturaram três *corpus* distintos, sendo dois deles coletados em redes sociais onde usuários mencionavam efeitos de medicamentos e um terceiro obtido de relatórios médicos com dupla anotação das sentenças indicando se elas continham ou não menções à ADR. Os autores treinaram então diferentes classificadores baseados em ML para identificar essas referências.

Relatam que misturar os dois *datasets* de redes sociais entre si trouxe bons resultados, mas a adição do terceiro ajudou pouco na classificação. Um achado relevante do trabalho, portanto, é que mesmo textos de um mesmo domínio podem não funcionar bem quando combinados se suas características (nível técnico, corretude ortográfica, etc) forem muito distintas. Cabe ressaltar que os textos usados no trabalho eram do idioma inglês, mas é razoável supor, pelas técnicas adotadas, que o comportamento seja semelhante em outros idiomas.

Voltando aos trabalhos que utilizaram *corpus* em português, citaremos aqui o trabalho de DOS SANTOS et al. (2018) que usou a base de dados de hospitalização do Hospital Nossa Senhora da Conceição (HNSC) para avaliar uma série de métodos de extração de *features* e predizer, através de regressão linear, o "índice CCI" dos casos, sendo CCI a sigla de *Charlson comorbidity index*.

Esse índice é usado para prever a mortalidade de pacientes e também indicar a complexidade do caso de internação. Segundo o trabalho, enquanto uma maneira comum de computá-lo é em duas fases distintas (extrair os CID das narrativas de internação e depois somar os pontos atribuídos a cada um pelo CCI), o trabalho procura predizer o índice em uma única etapa, extraindo as *features* do texto usando técnicas de PLN e aplicando a elas algoritmos de regressão linear baseados em Aprendizado de Máquina.

O trabalho apresenta uma descrição das *features* textuais mais úteis à predição do CCI e também compara e avalia diferentes modelos de ML para a realização da tarefa, sendo o melhor resultado obtido com o uso de Redes Neurais e *Word Embeddings*. Ou-

tra importante contribuição do trabalho foi a publicação dos *embeddings* gerados para condução dos seus experimentos[2] .

Em relação aos dados do TelessaúdeRS, entre os trabalhos que aplicaram métodos de Aprendizado de Máquina, encontramos o de Lenz (2018), que se dedicou a estimar o erro refrativo na visão de pacientes oftalmológicos, medida necessária para a prescrição das lentes corretivas, prescritas por oftalmologistas para tratar pacientes com erros de refração e deficiências no processo de acomodação visual.

Conforme os autores, o diagnóstico necessário à prescrição destas lentes é tipicamente realizado em duas fases: uma estimativa inicial (chamada Refração Objetiva) e um ajuste-fino (chamado Refração Subjetiva). O trabalho concentrou-se nesta segunda fase, buscando aprimorar a qualidade da refração estimada e reduzindo o tempo necessário para determiná-la.

Diferentes modelos de ML foram explorados, tendo como entrada as informações da medida da Refração Subjetiva e outras características do paciente, como idade, sexo, e sintomas apresentados. O melhor desempenho foi obtido pela aplicação de Redes Neurais, tendo os autores - assessorados por médicos especialistas - concluído que o desempenho apresentado pelo modelo estava dentro de limites aceitáveis para ser posto em prática no auxílio aos profissionais da área de oftalmologia.

Enquanto o trabalho anterior lidou com o problema de regressão, Damasceno et al. (2016) realizou um estudo sobre a aplicação de Mineração de Textos sobre dados do TelessaúdeRS. Como mencionado no Capítulo 1, no serviço de teleconsultorias prestado pelo Telessaúde profissionais da APS podem enviar por escrito um questionamento que, após regulação, é encaminhado a um teleconsultor que a responderá.

Para a elaboração das respostas, o primeiro passo do teleconsultor é buscar no sistema informatizado outros questionamentos semelhantes e verificar suas respostas. O trabalho de Damasceno et al. (2016) buscou avaliar o impacto da aplicação da Mineração de Textos nesta etapa de pesquisa na base de conhecimento. Tomados cerca de dois mil processos de teleconsultoria do sistema de teleregulação, foram extraídos grafos para identificação de termos relevantes. Após curadoria dos termos realizada por um especialista, que descartou termos não relevantes, cerca de 500 conceitos pertinentes aos questionamentos e outros 300 pertinentes às respostas foram extraídos e organizados em um sistema experimental de teleconsultoria. Nesse sistema era possível realizar pesquisas na base de conhecimento usando conceitos como termos de pesquisas e obter como

[2]https://www.inf.pucrs.br/linatural/wordpress/recursos-e-ferramentas/word-embeddings-para-saude/

retorno uma lista de questionamentos e respostas anteriores ao conceito pesquisado.

Trinta e sete teleconsultores testaram o sistema e o avaliaram através de um questionário com perguntas abertas e fechadas. Os resultados do trabalho mostraram que a lista de tópicos minerados contribuiu na localização de informações pertinentes para a construção de respostas, demostrando o potencial do emprego de técnicas de mineração de texto para agilizar e qualificar o processo de teleconsultoria.

3.1 Resumo do Capítulo

Neste capítulo mostrou-se que área da saúde é um terreno fértil para o desenvolvimento de pesquisas de PLN e identificou-se que sistemas de apoio a decisão clínica - onde se poderia enquadrar o sistema de Teleregulação - podem se beneficiar do implemento dessas técnicas. Sugeriu-se, pelos primeiros trabalhos analisados, que uma das principais maneiras de alavancar o desenvolvimento da área é com o desenvolvimento de projetos com escopo bem definido, atendendo a necessidades específicas de grupos de usuários particulares.

Essa estratégia de desenvolvimentos especializados está presente nos demais trabalhos discutidos, sendo predominantes as pesquisas utilizando aprendizado de máquinas - em detrimento de sistemas baseados em regras.

Depreendeu-se de algumas das leituras que as peculiaridades dos textos clínicos impõem a necessidade de acesso a textos próprios da área para o desenvolvimento de certos componentes e que a construção de *corpus* a partir de múltiplas fontes exige delicada avaliação da similaridade de características dos textos envolvidos.

No contexto brasileiro e em particular o âmbito do projeto TelessaúdeRS, foram identificados trabalhos realizando processamentos de linguagem natural tanto para tarefas de regressão quanto para classificação e mineração de textos e verificou-se que a aplicação de redes neurais figura frequentemente entres os modelos com melhor resultado prático.

A Tabela 3.1 sumariza os trabalhos mencionados neste capítulo em função de seus objetivos, conjuntos de dados e técnicas utilizadas. Pelo exposto aqui, pode-se supor que o objetivo do presente trabalho têm condições de ser perseguido a partir do implemento de técnicas semelhantes. O principal diferencial em relação aos trabalhos previamente discutidos é a dedicação particular à tarefa de tele-regulação, um trabalho de suma importância para a TelessaúdeRS e cuja possibilidade de otimização utilizando as técnicas mencionadas ainda não foi explorada.

Autor	Objetivo	Dataset	Técnicas/métodos avaliados
Demner-Fushman, Chapman e McDonald (2009)	Analisar os avanços das pesquisas de PLN associadas a Sistema de Apoio à Decisão Clínica.	N/A	Várias. Cada trabalho mencionado utilizava diferentes técnicas e métodos.
Kreimeyer et al. (2017)	Identificar sistemas clínicos que utilizam NLP para gerar informação estruturada a partir de texto livre.	N/A	Várias. Cada trabalho mencionado utilizava diferentes técnicas e métodos.
Uzuner (2009)	Reportar os resultados do challenge cujo foco era detectar automaticamente em relatórios de alta de pacientes informações sobre obesidade e outras 15 comorbidades associadas.	Conjunto de 1.237 relatórios de alta de pacientes, a partir do ano de 2004. O relatórios foram anotados para indicar se continham menções explícitas e implícitas à obesidade e às 15 comorbidades associadas. As anotações foram feitas por dois médicos especialistas no tratamento de obesidade e eventuais discordâncias resolvidas por um terceiro médico (todos do hospital Massachusetts General Hospital Weight Center)	Várias. Cada participante do challange utilizou um conjunto de técnicas. O artigo separa os trabalhos em função dos que adotaram abordagens baseadas em regras e os que adotaram abordagens baseadas em Aprendizado de Máquinas.
Oleynik et al. (2010)	Comparar o desempenho de um POS-Tagger treinado com textos de relatórios de alta de pacientes a um outro treinado com base em textos de um dataset de notícias.	Corpus jornalístico (MAC-Morpho), contendo o texto de 1.167.183 notícias publicadas pelo jornal Folha de São Paulo ao longo do ano de 1994. Corpus médico contendo duas coleções de textos. A primeira composta por 2.453 relatórios de alta de pacientes relativos a diversas especialidades médicas, gerados no mês de junho de 2007. A segunda composta por 5.617 relatórios específicos do departamento de cardiologia gerados entre Junho de 2002 e maio de 2007.	Diferentes estratégias de geração de POS-Tagger (comitê heterogêneo, comitê homogêneo e comitê homogêneo com conjunto inicial otimizado). Impacto de diferentes perfis de texto na criação de POS-Tagger.
Silva E Oliveira et. al. (2013)	Identificar a existência de referências à necessidade de continuar o tratamento do paciente após sua alta.	Dataset composto por 5.617 relatórios de alta do departamento de cardiologia do Hospital de Clínicas de Porto Alegre, gerados entre Junho de 2002 e maio de 2007.	Abordagem baseada em regras para a estruturação de textos a partir da aplicação de POS-Tagging
Sarker e Gonzalez (2015)	Identificar características textuais úteis ao monitoramento de Reações Adversas a Medicamentos (ADR)	Três diferentes conjuntos de dados, sendo dois deles criados durante a condução do estudo e um já existente e publicamente disponível. O dataset já existente continha 23.516 sentenças, extraídas de relatórios de casos médicos, duplamente anotadas indicando a presença ou não de menções à reações adversas a medicamentos. Dos datasets criados para o trabalho, um deles era composto de 10.822 tweets e o outro 10.617 comentários obtidos de uma rede social online focada em saúde chamada DailyStrength. Tanto os tweets quando os comentários foram aleatoriamente selecionados de conjuntos maiores que mencionavam nomes de medicamentos e foram duplamente anotados por profissionais da área médica, sob supervisão de um especialista em farmacovigilância.	Diferentes classificadores (Naive Bayes, SVM e Maximum Entropy); Diferentes técnicas de pré-processamento (tokenização e steaming); Enriquecimento dos textos com sinônimos do Wordnet, dados do UMLS, Topic Modelig, Análise de sentimentos; Diferentes combinações dos três datasets nos treinamentos.
Dos Santos et al. (2018)	Avaliar diferentes métodos de extração de features de textos e métodos para predição do índice "Charlson comorbidity index"(CCI).	Conjunto com 1.551.907 notas clínicas, relativas a 48.907 internações do Hospital Nossa Senhora da Conceição realizadas entre janeiro de 2012 e dezembro de 2017, com a respectiva anotação de valor do índice CCI	Diferentes tipos de representação dos textos (TFiDF com unigramas e n-gramas, LDA e Word Embeddings); Diferentes algoritmos de classificação (Florestas aleatórias, KNN, Linear SVR e Rede Neural Convolucional)
Federal (2018)	Utilizar Aprendizado de Máquina para estimar o erro refrativo na visão de pacientes oftalmológicos, utilizando, além da medida de Refração Objetiva, características do paciente, como idade, sexo, e sintomas apresentados.	Conjunto contendo dados de 3.687 pacientes oftalmológicos, coletado pelo TelessaúdeRS relativos ao período de julho de 2017 a agosto de 2018.	Diferentes algoritmos de classificação (Regressão Linear, Máquinas deVetores de Suporte, e Redes Neurais Multilayer Perceptron)
Damasceno et. al. (2016)	Avaliar como a tecnologia de mineração de texto pode contribuir na construção de respostas a teleconsultorias feitas no contexto da telessaúde.	Conjunto contendo 2.074 solicitações de teleconsultorias do TelessaúdeRS do período de outubro de 2012 até junho de 2014.	Identificação de conceitos a partir do implmento de mineração textual, utilizando a ferramenta existente (Sobek), para correlacionar perguntas e respostas de teleconsultorias

Tabela 3.1: Comparativo dos trabalhos relacionados

4 ABORDAGEM PROPOSTA

O objetivo geral deste projeto é desenvolver, testar e avaliar uma ferramenta de Classificação de Textos baseada em Aprendizado de Máquina que otimize o processo de regulação de encaminhamentos ambulatoriais da Atenção Básica à Atenção Especializada, no âmbito do projeto TelessaúdeRS.

Para o atingimento deste objeto é realizado um extenso comparativo entre diferentes tarefas de pré-processamento e engenharia de *features* para os dados de telerregulação do TelessaúdeRS.

O projeto é desenvolvido em três fases, havendo intersecção entre o andamento das duas primeiras e sendo a terceira realizada após a identificação do melhor resultado obtido nas anteriores. As três fases previstas são listadas abaixo e as seções seguintes as detalham em termos de entradas, procedimentos e desfechos de interesse.

1. Desenvolvimento incremental de algoritmos de Classificação;

2. Teste de desempenho dos algoritmos de Classificação;

3. Avaliação da eficácia dos algoritmos de Classificação de Textos na regulação ambulatorial.

4.1 Fase 1 - Desenvolvimento incremental de algoritmos de Classificação

- **Entradas**: Textos dos encaminhamentos regulados pelo TelessaúdeRS no período de Jun/2016-Abr/2019 para as especialidades de Urologia, Reumatologia, Gastroenterologia, Endocrinologia, Proctologia.

- **Fator em estudo**: Engenharia de características de texto e Modelos de Classificação.

- **Desfecho de interesse**: Identificação, dentre as combinações avaliadas, do melhor conjunto de tarefas para um fluxo de extração de *features*. Identificação do modelo de Classificação testado que melhor se adequa à classificação dos encaminhamentos.

- **Tamanho amostral**: Total de 45.039 encaminhamentos.

- **Procedimentos**: São extraídos da base de dados do sistema atual de regulação os textos de encaminhamos já recebidos e analisados por médicos reguladores. A decisão dada pelo médico regulador associada a cada texto é tomada como a adequada decisão para o caso (classe-alvo). Os textos dos encaminhamentos são submetidos a diferentes tarefas de pré-processamento e diferentes abordagens para extração de características (estatística e semântica). Quando usada abordagem estatística, são avaliadas diferentes métricas de peso para os termos, bem como diferentes formas de representação do vetor de características. Quando utilizada abordagem semântica, são executadas tarefas de Extração de Informações (*Part-of-speech Tagging*, *Named Entity Recognition*, etc), visando estruturar conceitos e informações presentes no texto. Extraídas as características, essas são submetidas a diferentes Modelos de Classificação (Máquinas de Vetor de Suporte (SVM), Árvores de Decisão e Redes Neurais Recorrentes). Combinações de classificadores (*ensembles*) também são avaliados nessa etapa. As tarefas citadas são executadas ora concomitante, ora alternativamente, a fim de avaliar o impacto de cada combinação sobre os resultados, bem como identificar a melhor combinação entre tarefas de extração de características e modelo de classificação.

4.2 Fase 2 - Teste de desempenho dos algoritmos de Classificação

- **Entradas**: Os mesmos textos da etapa da seção 4.1.

- **Fator em estudo**: Algoritmo de Classificação de Textos, incluindo todo o processo desde as tarefas de extração e manipulação de *features* até a configuração do modelo de classificação.

- **Desfecho de interesse**: Sensibilidade do algoritmo (taxa de detecção de encaminhamentos Aprovados).

- **Tamanho amostral**: O mesmo da seção 4.1. Os algoritmos são testados e refinados iterativamente usando técnicas de validação cruzada (*cross-validation*) quando viável, buscando otimizar o uso dos dados disponíveis para treinamento, teste e validação e diminuir a possibilidade de *overfitting*. Quando, em função do custo computacional, não se mostra viável o uso de validação cruzada, os algoritmos são avaliados pela técnica de *holdout*, usando 9009 exemplos que representam 20% dos

dados disponíveis.

- **Procedimentos**: Esta fase iterativamente fornece e recebe *feedback* da fase anterior. Assim, trabalha-se aqui com os mesmos dados extraídos do sistema de regulação. Os encaminhamentos já avaliados previamente por reguladores são submetidos ao algoritmo de classificação de textos. Para fins dessa avaliação, é definida como regulação a primeira decisão tomada por um médico regulador a partir dos dados iniciais. A sensibilidade e especificidade são calculadas considerando-se como padrão-ouro a decisão do médico regulador. O desempenho do algoritmo é sumarizado pela ROC AUC obtida por ele na classificação de exemplos de teste, o que poderá levar a sua escolha como melhor algoritmo experimental encontrado ou a uma alteração nas configurações e tarefas usadas para o seu desenvolvimento, buscando aumentar seu desempenho em termos de ROC AUC.

4.3 Fase 3 - Avaliação da eficácia do algoritmo de Classificação de Textos na regulação ambulatorial

- **Entradas**: Novos encaminhamentos nas filas de encaminhamento para atenção especializada de Porto Alegre.
- **Fator em estudo**: Acurácia diagnóstica do algoritmo experimental com melhor resultado, segundo testes da seção 4.2.
- **Controle**: Regulação realizada por médicos (método atual de regulação).
- **Padrão ouro**: Avaliação do encaminhamento por reguladores independentes com larga experiência.
- **Desfecho de interesse**: Número de encaminhamentos adequadamente autorizados.
- **Procedimentos**: O algoritmo de melhor desempenho segundo a etapa da seção 4.2 é encapsulado em um componente de software capaz de ser integrado ao atual sistema de regulação. Os novos encaminhamentos de cada uma das filas monitoradas são avaliados de forma independente por reguladores externos, além de serem avaliados pelo método experimental (algoritmo) e método atual (regulador único). Os resultados dos métodos (experimental e atual) serão comparados com a do padrão ouro e também entre si.

- **Tamanho amostral**: São avaliados 1050 novos casos, sendo 350 para a fila de Procto, 350 para a fila de Reumato e 350 para a fila de Uro.

4.4 Resumo do Capítulo

O projeto seguirá iterativamente três fases. A primeira delas terá como foco a preparação dos dados, engenharia de *features* dos textos e seleção de um algoritmo de classificação, buscando o estabelecer um *baseline* para o resultado da classificação. A fase seguinte terá como foco o ajuste do fino do algoritmo em consonância com o conjunto de dados, buscando encontrar a configuração destes dois fatores que atinge melhor resultado na classificação, sendo os resultados comparados pela ROC AUC de cada classificador. A fase final aplicará o melhor algoritmo encontrado a uma base totalmente nova, confrontando o seu desempenho com aquele obtido pelo método de classificação atual, isto é, a decisão dada pelo médico regulador no processo normal de regulação e mais uma avaliação de controle, dada por um par de reguladores experientes após revisar individualmente cada um destes casos.

Figura 4.1: As três fases desenvolvidas para realização deste trabalho

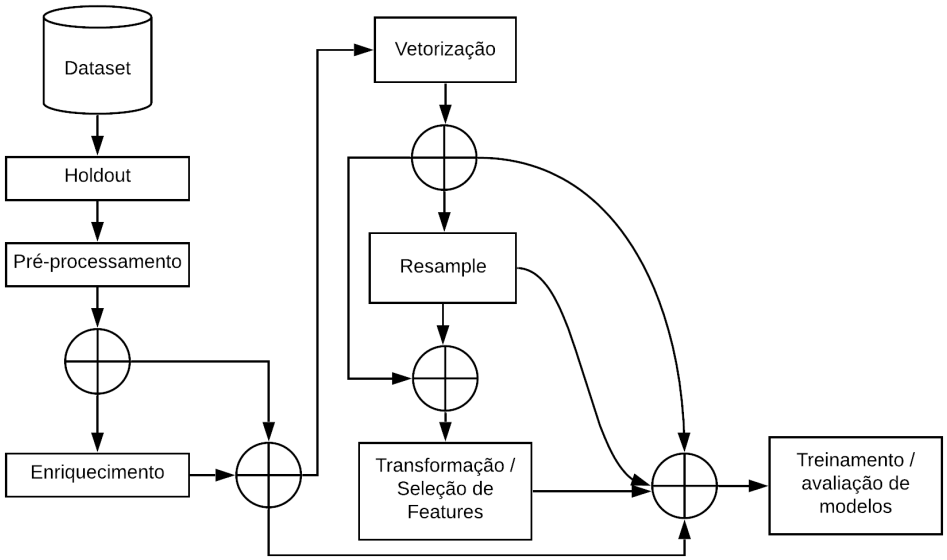
5 VALIDAÇÃO EXPERIMENTAL

Este capítulo apresenta o conjunto de experimentos realizados no desenvolvimento deste trabalho. O objetivo dos experimentos foi encontrar um conjunto ótimo de tarefas para um fluxo de extração de características e selecionar um modelo de classificação de textos com bom desempenho preditivo, i.e., capaz de determinar se uma certa requisição de encaminhamento deveria ser autorizada ou não.

Ao longo dos experimentos, diversas combinações de técnicas foram usadas para tratar os dados e parametrizar modelos, buscando o melhor desempenho na classificação. A Figura 5.1 apresenta as etapas realizadas durante os experimentos. Na Figura, cada retângulo representa uma etapa e as setas indicam o fluxo percorrido pelos textos dos encaminhamentos até serem utilizados como entrada para treinamento de um modelo de classificação. Os círculos da imagem representam pontos de decisão sobre caminhos alternativos que podem ser seguidos após cada etapa.

A execução dessas etapas e suas combinações deu-se de maneira iterativa, com alterações e retomadas do processo em diversos pontos. As escolhas pela condução das trajetórias dos experimentos foram feitas a partir da observação de resultados de classificações em execuções anteriores do processo.

Figura 5.1: Visão geral das etapas dos experimentos



Este capítulo está organizado de forma que a seção 5.1 descreve o conjunto de dados utilizado nos experimentos e a seção 5.2 descreve como ele foi separado para treinamento e teste dos modelos. Após isso, a seção 5.3 descreve os pré-processamentos

básicos realizados e a seção 5.4 aborda em particular uma manipulação realizada para enriquecer semanticamente os dados. Na seção 5.5, são apresentadas as técnicas de vetorização e representação de textos utilizadas. Segue-se a isso, na seção 5.6, a descrição das técnicas de seleção e transformação de *features* avaliadas, e na 5.7 a menção das técnicas de balanceamento de classes utilizadas. O treinamento e a avaliação dos modelos de ML utilizados são então apresentados na seção 5.8.

Dentro desta última seção, após uma rápida explicação em 5.8.1 sobre o padrão adotado na construção das ilustrações dos fluxos, as subseções de 5.8.2 até 5.8.8 apresentam os resultados de sete fluxos alternativos que foram explorados combinando algumas das técnicas apresentadas. Cada uma dessas seções traz os detalhes necessários à compreensão de como foram conduzidos aqueles experimentos em particular, os resultados obtidos e uma breve discussão a seu respeito.

5.1 Conjunto de Dados

Os dados utilizados nos experimentos foram os textos dos encaminhamentos da atenção primária para atenção especializada do interior do Rio Grande do Sul do período de junho de 2016 (início do sistema informatizado) até abril de 2019. Esses textos foram extraídos do sistema informatizado utilizado atualmente na regulação, no qual os encaminhamentos já haviam sido analisados por médicos reguladores. A decisão dada pelo médico regulador naquele sistema e associada a cada texto foi tomada como a decisão adequada para o caso, i.e., a classe-alvo para o treinamento dos classificadores.

Foram obtidos dados de dez diferentes especialidades médicas (elencadas na tabela 5.1) - também chamadas de “filas de encaminhamentos” ou somente filas. Os textos de algumas dessas filas foram agrupados, de acordo com a sugestão dos médicos que acompanharam os experimentos, resultando em cinco especialidades distintas. Inicialmente, um total de 63.958 encaminhamentos foi extraído do sistema informatizado utilizado pela equipe do TelessaúdeRS, chamado GERCON. Esse conjunto inicial continha casos regulados fora do TelessaúdeRS e também por médicos que trabalharam apenas temporariamente no projeto do TelessaúdeRS. Após discussão com os médicos que participaram do trabalho, optou-se por descartar essas solicitações, tendo em vista que elas poderiam não representar os critérios comumente usados pelos reguladores do TelessaúdeRS. O conjunto inicial continha também casos de especialidades infantis que têm critérios distintos daqueles adotados para pacientes adultos. Também por orientação dos

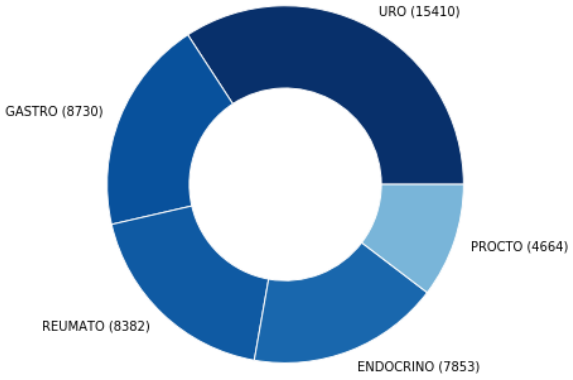
Tabela 5.1: Especialidades agrupadas e as suas quantidades de encaminhamentos

Especialidade Agrupada (Fila)	**Especialidade Original**	**Qtd. Original**	**Qtd. Agrupada**	**Qtd. Autorizados**	**Qtd. Não-autorizados**
ENDOCRINO	Endocrinologia Adulto	6617	7853	3156	4697
	Endocrinologia Tireoide	1236			
GASTRO	Gastroenterologia Adulto	7730	8730	3092	5638
	Gastroenterologia Hepatites Virais Adulto	833			
	Gastroenterologia-Doença Inflamatória Intestinal	167			
PROCTO	Proctologia Adulto	4664	4664	1141	3523
REUMATO	Reumatologia Adulto	8382	8382	2097	6285
URO	Urologia Adulto	15042	15410	7271	8139
	Urologia Litíase Renal	344			
	Urologia Vasectomia	24			
	Total	**45039**	**45039**	**16757 (37%)**	**28282 (63%)**

médicos reguladores, esses casos foram desconsiderados na execução dos experimentos. Após descartar todos esses casos, o número de 63.958 ficou reduzido a 45.039 encaminhamentos, divididos em 5 filas diferentes: Endócrino, Gastro, Procto, Reumato e Uro e esse foi o conjunto utilizado para os experimentos.

As filas e os números do encaminhamentos em cada uma delas constam na Tabela 5.1 e a Figura 5.2 representa a proporção de cada uma das filas agregadas no total dos casos utilizados para os experimentos.

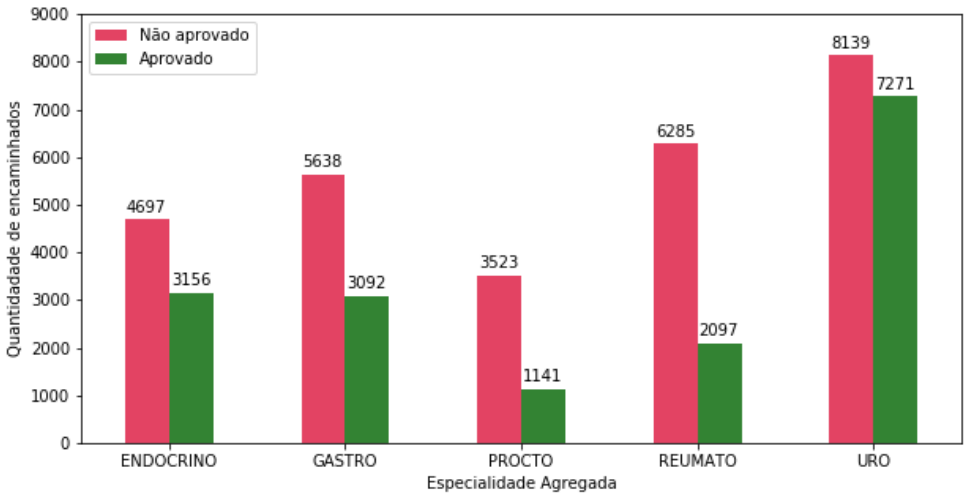
Figura 5.2: Quantidade de casos, por especialidade agrupada



Em relação à decisão adotada, os dados continham diferentes possibilidades para cada um dos casos, por exemplo: Aprovada, Agendada, Cancelada, Pendente, etc. Com orientação dos médicos reguladores, cada um desses valores foi convertido ou para Aprovado ou para Não-aprovado, representando respectivamente os casos em que o encaminhamento foi Aprovado e aqueles em que não foi, independentemente da motivação do

regulador para tal decisão. A distribuição de casos após essa normalização é apresentada na Figura 5.3

Figura 5.3: Quantidade de casos aprovados ou não, por especialidade agrupada



5.2 Holdout

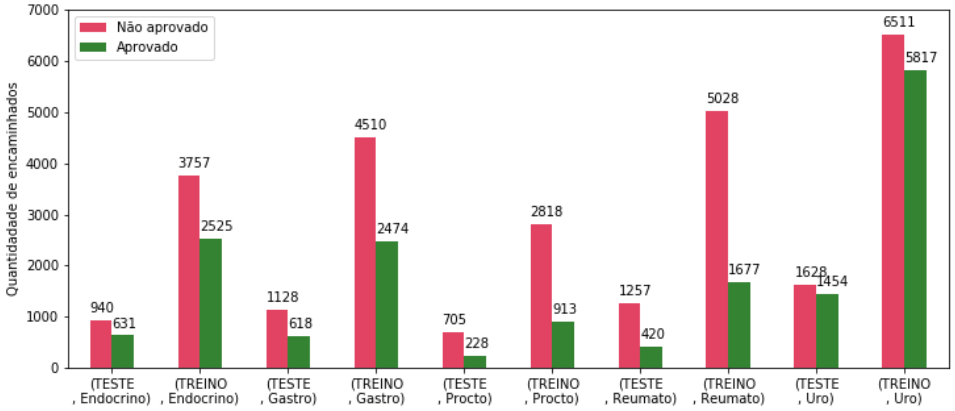
Durante os experimentos, para avaliar o resultado dos classificadores, usou-se tanto o método de validação cruzada quanto o método de *holdout* (ver seção 2.6 para detalhes). A escolha do método deu-se sobretudo conforme o classificador em questão, por exemplo, quando usadas redes neurais, tendo em vista o custo computacional para o uso da validação cruzada, os resultados foram avaliados pelo *holdout*.

A separação de dados realizada na etapa de *holdout* seguiu a seguinte lógica: Para cada uma das especialidades (já agrupadas), tomou-se 20% dos exemplos negativos e 20% dos exemplos positivos para compor a porção de teste. Os 80% restantes compõem a porção de treino. Isso representa uma separação 80%/20% estratificada, garantindo que dentro de cada especialidade esteja preservada a proporcionalidade de aprovados/não-aprovados. Os números após essa separação são apresentados na Figura 5.4.

5.3 Pré-processamento

Ao longo do desenvolvimento dos experimentos foram criados alguns conjuntos de tarefas de pré-processamento diferentes, que foram chamados pré-processadores.

Figura 5.4: Quantidade de casos aprovados ou não, por especialidade agrupada, nos *datasets* de Teste e Treino



Cada conjunto de tarefas foi encapsulado em uma função *python*[1], nomeada como "preprocessor1", "preprocessor2" etc. Inicialmente, ao se trabalhar com a representação *Bag-of-Words*, foram criados três pré-processadores. Posteriormente, visando estender o treinamento de uma representação do tipo *Word Embeddings* foi criado um quarto pré-processador e, por fim, para criar um *Word Embedding* próprio um quinto pré-processador foi elaborado. Na etapa de enriquecimento de textos, a ser discutida na seção 5.4, foram criados os pré-processadores "enriched", especificamente para aqueles dados.

A relação das tarefas executadas por cada um desses pré-processadores é apresentada na Tabela 5.2, sendo os resultados das suas aplicações sobre um exemplo de texto de encaminhamento típico apresentados na Tabela 5.3.

Tabela 5.2: Os pré-processadores e suas respectivas tarefas

	Pré-processador					
	1	2	3	4	5	enriched, enriched-min2
Remoção de acentuação	x				x	x
Lowercase	x	x	x	x	x	x
Remoção de Stopwords (NLTK)	x	x	x		x	x
Stemming (RSLP via NLTK)	x		x		x	
Remoção de números e caracteres especiais		x	x		x	x
Lematização		x				
Remoção de caracteres especiais				x		
Remoção de tokens com somente um caracter					x	
Remoção de tokens com base em POS-Tagging						x
Adição de informações obtidas no UMLS						x

[1]https://www.python.org/

Tabela 5.3: Exemplos do resultado de cada pré-processador

Original	Paciente com relatos de hiperplasia prostática benigna desde 2010, pelo qual já realizou tratamento (não bem definido) com doxazosina 2 mg + finasterida 5 mg 1 cp a noite, agora refere estar tomando apenas doxazosina, assintomático no momento. Teve acompanhamento com urologista até março 2018. Solicita encaminhamento ao especialista pelo SUS. \nExames 14/02/18: PSAL 0,35 / PSAT 2,20 / creatinina 0,8.N
preprocessor1	paci relat hiperplas prosta benign desd 2010 ja realiz trat nao bem defin doxazosin 2 mg finaster 5 mg 1 cp noit agor ref est tom apen doxazosin assintoma moment acompanh urolog ate marc 2018 solicit encaminh especi su exam 14 02 18 psal 0 35 psat 2 20 creatinin 0 8 n
preprocessor2	paciente relato hiperplasia prostatica benigno desde pelar ja realizar tratamento nao bem definir doxazosina mg finasterida cp noite agora referir estar tomar apenas assintomatico momento acompanhamento urologista atar marcar solicita encaminhamento especialista sus exames psal psat creatinina
preprocessor3	paci relat hiperplas prosta benign desd ja realiz trat nao bem defin doxazosin mg finaster mg cp noit agor ref est tom apen doxazosin assintoma moment acompanh urolog ate marc solicit encaminh especi su exam psal psat creatinin n
preprocessor4	paciente com relatos de hiperplasia prostática benigna desde 2010, pelo qual já realizou tratamento (não bem definido) com doxazosina 2 mg + finasterida 5 mg 1 cp a noite, agora refere estar tomando apenas doxazosina, assintomático no momento. teve acompanhamento com urologista até março 2018. solicita encaminhamento ao especialista pelo sus. exames 14/02/18: psal 0,35 / psat 2,20 / creatinina 0,8.n
preprocessor5	paci relat hiperplas prosta benign desd ja realiz trat nao bem defin doxazosin mg finaster mg cp noit agor ref est tom apen doxazosin assintoma moment acompanh urolog ate marc solicit encaminh especi su exam psal psat creatinin
enriched	paciente relatos hiperplasia prostatica benigna realizou tratamento bem definido doxazo sina finasterida noite agora refere estar tomandoapenas doxazosina assintomatico momento acompanha mento urologista marco solicita encaminhamento aoespecialista sus exames psal psat creatinina

5.4 Enriquecimento dos textos

A revisão bibliográfica realizada durante o desenvolvimento deste trabalho mostrou que muitos dos sistemas desenvolvidos na área de PNL clínico procuram extrair conceitos semânticos dos textos para melhorar a compreensão dos mesmos. Enquanto sistemas puramente estatísticos têm bons resultados, como já referido, é comum ver tentativas de estruturação de informações a partir da identificação nos textos de menções a medicamentos, sintomas, exames e outros conceitos da área.

Visando enriquecer os textos deste trabalho com dados semanticamente relevantes, foram realizados experimentos buscando-se acrescentar ao texto informações obtidas a partir do *Unified Medical Language System - UMLS*. Para extrair tais informações, a primeira iniciativa foi utilizar o sistema Apache cTAKES, descrito na seção 2.1.2, já que ele é uma ferramenta bastante completa e voltada exatamente para a tarefa em questão. Após a instalação do sistema e a realização de testes usando o software CAS Visual Debugger com o pipeline padrão, também mencionados na seção 2.1.2, observou-se que alguns termos do texto não eram identificados como elementos semânticos relevantes.

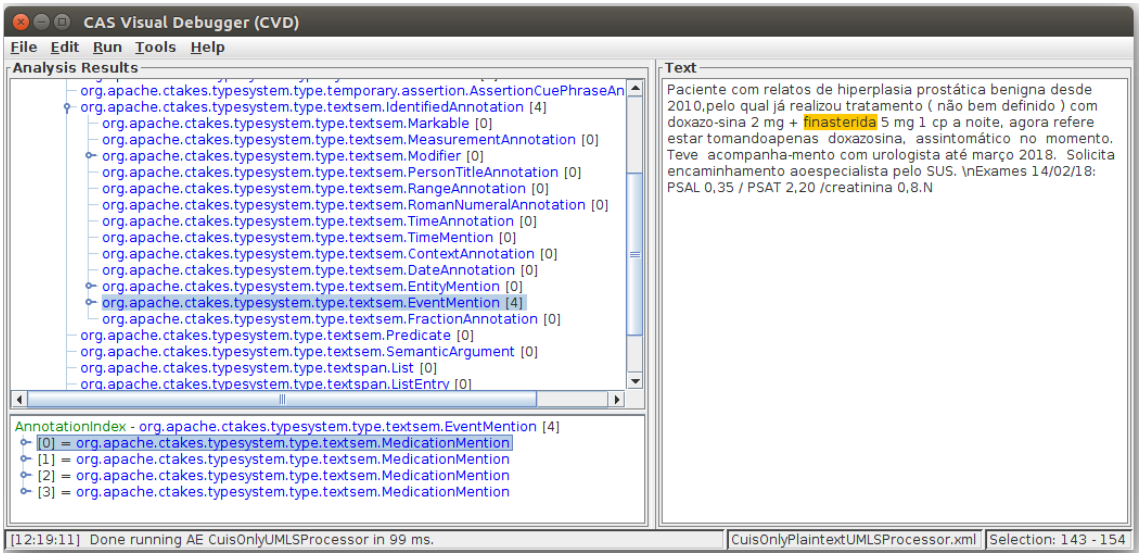
Um exemplo disso pode ser observado na Figura 5.5 em que é apresentado o resultado do processamento do texto original da Tabela 5.3 usando um dos *pipelines* préconfigurados da ferramenta e indicando o idioma do texto como português. Percebe-se que foram identificados apenas 4 Identificadores Únicos de Conceitos - CUI (ver seção 2.1.1) relacionados com medicações, sendo que um deles é na verdade o verbo "estar", que foi erroneamente classificado.

Embora houvessem diversas possibilidade de configuração do sistema CTAKES que poderiam eventualmente melhorar o desempenho obtido, tendo em vista o custo envolvido com o aprendizado do sistema, optou-se por tentar realizar a identificação de elementos semânticos do texto diretamente no UMLS.

Para tanto, uma instalação local do UMLS foi criada usando a ferramenta MetamorphoSys, apresentada em Medicine (), que serve como *wizard* de instalação do UMLS e permite também explorar o conteúdo de suas fontes de dados.

O procedimento realizado para extração das informações da base dessa instalação consistiu em tomar os termos que apareciam nos textos dos encaminhamentos, bem como combinações de dois e três desses termos, e pesquisá-los na base do UMLS. Se encontrados, algumas das meta-informações associadas aos termos eram extraídas para serem posteriormente adicionadas ao corpo de texto. A escolha de pesquisar os termos individu-

Figura 5.5: Uso do CTAKES, via CAS Visual Debugger (CVD), para identificar conceitos presentes no texto de exemplo



ais e combinações e dois e três termos foi arbitrária, balizada por observações dos textos e por testes manuais realizados na base de dados do UMLS usando.

As informações extraídas do UMLS para cada termo - ou combinação de termos - foram: CUI, Termo preferencial no UMLS, Tipo Semântico e Sinônimos. Optou-se por resgatar apenas os três primeiros CUIs retornados nas buscas, isto é, a *query* incluiu uma cláusula *LIMIT=3* para limitar a quantidade de CUIs retornados em cada pesquisa, já que alguns termos podem aparecer em muitos conceitos, por exemplo, "hiperplasia" é listada em 112 conceitos diferentes.

Para selecionar os meta-dados, um *script* Python foi elaborado para pré-processar os textos e realizar as consultas na base de dados. As pesquisas na base do UMLS esperam que os termos usados como critério tenham sido normalizados usando o *NORM*, um dos softwares disponíveis no pacote de programas *SPECIALIST NLP Tools*[2]. Embora se tenha instalado e testado esse pacote, o resultado do *NORM* não era ideal para o teste que se pretendia, pois ele retornava os termos ordenados alfabeticamente (*Bag-of-Words*), perdendo-se a relação de posição entre eles e impedindo que se realizasse o teste pretendido com os n-gramas. Por conta disso, um pré-processamento próprio foi implementado para normalizar os termos, mantendo a posição de cada um no texto.

Além da normalização dos termos, como as consultas SQL estavam retornando muitos resultados para os termos individuais, foram descartados artigos, preposições, pronomes demonstrativos, conjunções (subordinativas e coordenativas) e também pronomes e substantivos próprios com dois caracteres ou menos.

[2]https://lexsrv3.nlm.nih.gov/Specialist/Home/index.html

Tabela 5.4: Categorias gramaticais identificadas no texto de exemplo pela aplicação do *POS-Tagger* utilizado neste trabalho

Token	**POS Tag**
bem, agora	advérbio
estar	infinitivo
mg, mg, cp, sus	nome próprio
paciente, relatos, hiperplasia, prostatica, benigna, tratamento, doxazo, sina, finasterida, noite, tomandoapenas, doxazosina, assintomatico, momento, mento, urologista, marco, encaminhamento, aoespecialista, exames, psal, psat, creatinina, n	nome, substantivo
definido	particípio
desde	preposição
realizou, refere, acompanha, solicita	verbo finito

A identificação dessas categorias gramaticais foi feita com um *POS-Tagger* implementado via biblioteca NLTK, apresentada em Loper e Bird (2002). A escolha das categorias as serem descartadas foi feita com base na observação dos resultados da aplicação do *POS-Tagger* sobre alguns textos de exemplo. A Tabela 5.4 apresenta as classificações fornecidas para o texto de exemplo da Tabela 5.3. O resultado completo do pré-processamento feito antes da fase de pesquisa no UMLS é mostrado na tabela 5.3, na linha do pré-processador *enriched*.

Após a normalização e o descarte de termos nas categorias gramaticais mencionadas, o *script* toma os termos do documento em n-gramas de tamanho 3 e busca na base de dados do UMLS conceitos contendo simultaneamente todos os três termos. Após atingir o fim do texto o *script* retorna ao termo inicial e percorre o texto novamente, agora consultando n-gramas de tamanho 2. Quando algum conceito é identificado, um dicionário Python com os metadados é gerado, conforme ilustra a Figura 5.6. Se um conceito é identificado, o *script* remove do texto os termos utilizados para identificá-lo. Assim, se um conceito mais específico composto por 3 termos como "hiperplasia prostática benigna" é identificado no texto, o conceito mais abrangente "hiperplasia prostática" não será considerado, pois os *tokens* não estarão mais presentes no texto na iteração que busca por conceitos usando dois termos.

Na Figura 5.6 é possível ver um trecho da estrutura de dados retornada pelo *script* elaborado para obter dados do UMLS. Trata-se de um dicionário Python, contendo o CUI o nome preferencial (str) do conceito e o seu tipo semântico, acompanhados de uma lista de sinônimos para o conceito. O atributo *"match"* é adicionado pelo *script* para indicar quais *tokens* do texto foram usados para identificar o conceito em questão.

Ao final da elaboração do *script* foi possível identificar, para o texto de exemplo,

Tabela 5.5: Textos retornados na saída, pelo script que obtém os meta-dados do UMLS

Entrada	paciente relatos hiperplasia prostatica benigna realizou tratamento bem definido doxazo sina finasterida noite agora refere estar tomandoapenas doxazosina assintomatico momento acompanha mento urologista marco solicita encaminhamento aoespecialista sus exames psal psat creatinina
Saída conceitos 1+ tokens	C0030705 Patient or Disabled Group Paciente Pessoas Enfermas Pessoas com Enfermidades Enferma Pessoas Doentes Doente Doentes Paciente Pessoa Doente Pessoas com Doenças Enfermo Pessoa Enferma Pessoa com Enfermidade Pessoa com Doença Pacientes relatos C1704272 Disease or Syndrome Hiperplasia Prostática Benigna Hiperplasia Benigna da Próstata Hiperplasia Prostática Benigna realizou C0087111 Therapeutic or Preventive Procedure Tratamento Procedimento terapêutico NE Procedimento terapêutico Procedimentos Terapêuticos Terapia Procedimentos de Terapia Tratamentos Procedimento de Tratamento Ação Terapêutica Procedimento Curativo Procedimentos Curativos Ações Terapêuticas Medida Terapêutica Tratamento Procedimentos de Tratamento Propriedade Terapêutica Terapias Procedimento de Terapia Medidas Terapêuticas Terapêutica bem definido C0114873 Organic Chemical Doxazosina Doxazossina Doxazosina Doxazosin C0060389 Organic Chemical Finasterida Finasteride Finasterida noite agora refere estar tomando apenas C0114873 Organic Chemical Doxazosina Doxazossina Doxazosina Doxazosin C0560175 Finding Assintomático Portador Sadio Assintomático Portador de Organismo Infeccioso Portador Assintomático Portador Passivo Portador de Agente Etiológico de Doença Infecciosa Portador Eficiente Portador Portador Ativo Portador Ineficiente momento acompanhamento urologista marco solicita C0034927 Health Care Activity Encaminhamento a um Especialista Encaminhamento Encaminhamentos Referência Interconsulta Interconsultas Encaminhamento a um Especialista sus exames psal psat C0201975 Laboratory Procedure Creatinina Creatininemia Creatinina
Saída conceitos 2+ tokens	paciente relatos C1704272 Disease or Syndrome Hiperplasia Prostática Benigna Hiperplasia Benigna da Próstata Hiperplasia Prostática Benigna realizou tratamento bem definido doxazosina finasterida noite agora refere estar tomando apenas doxazosina assintomatico momento acompanhamento urologista marco solicita C0034927 Health Care Activity Encaminhamento a um Especialista Encaminhamento Encaminhamentos Referência Interconsulta Interconsultas Encaminhamento a um Especialista sus exames psal psat creatinina

Figura 5.6: Exemplo de metadatados obtidos do UMLS com o nosso método

```
{'C0034927': {
    'cui': 'C0034927',
    'match': ['encaminhamento', 'especialista'],
    'semantic_type': 'Health Care Activity',
    'str': 'Encaminhamento a um Especialista',
    'synonyms': [
        'Encaminhamento',
        'Encaminhamentos',
        'Referência',
        'Interconsulta',
        'Interconsultas',
        'Encaminhamento a um Especialista'
    ]}
}
```

um número maior de CUIs do que o obtido com o uso do cTAKES em sua configuração padrão. Além do maior número, os conceitos identificados também foram diferentes. A Tabela 5.6 relaciona os CUIs extraídos com cada um dos métodos.

Após os conceitos terem sido obtidos a partir do UMLS, eles foram adicionados ao corpo de texto dos encaminhamentos. A maneira como isso foi feito foi a seguinte: Para cada elemento do dicionário de meta-dados obtidos para um texto, toma-se os *tokens* do atributo *match* e os substitui no texto do encaminhamento por uma nova *string* que contenha os demais atributos (CUI, str, semantic_type e synonyms) concatenados. Duas versões desse processo foram realizadas, gerando dois novos conjuntos de texto chamados, respectivamente, de "enriched" e "enriched-min2". A primeira, substituiu no corpo de texto todos os termos encontrados no UMLs. A segunda substitui no corpo de texto apenas combinações de dois ou mais (2+) *tokens* que tenham sido encontradas no UMLS. O resultado desse processo pode ser observado na Tabela 5.5.

Além dessa inserção direta dos dados do UMLS no corpo de texto dos encaminhamentos no *corpus*, também foram criadas duas matrizes com forma $Dx99$ onde D é a quantidade de exemplos de encaminhamentos e 99 é quantidade de tipos semânticos diferentes encontrados na porção de treinamento.

Em cada linha da matriz, que representa um encaminhamento específico, as 99 colunas foram preenchidas com a contagem de ocorrências daquele tipo semântico no documento em questão. A primeira dessas tabelas inclui a contagem considerando conceitos encontrados com um ou mais *tokens* (ex.: hiperplasia, hiperplasia prostática, hiperplasia

Tabela 5.6: Apresenta os tipos semânticos do UMLS identificados no texto de exemplo através do uso do Apache CTAKES e do método próprio

Método	**CUI**	**Termo preferencial no UMLS**	**Tipo semântico**[a]	**Tokens usados na identificação**
CTAKES	0060389	Finasteride	Medication Mention	finasterida
	C0070942	Phosphoserine aminotransferase	Medication Mention	PSAT
	0114873	Doxazosin	Medication Mention	doxazosina
	C0729252	Estar	Medication Mention	estar
Nosso	C1704272	Hiperplasia Prostática Benigna	Disease or Syndrome	hiperplasia, prostatica, benigna
	C0034927	Encaminhamento a um Especialista	Health Care Activity	encaminhamento, especialista
	C0201975	Creatinina	Laboratory Procedure	creatinina
	C0030705	Paciente	Patient or Disabled Group	paciente
	C0114873	Doxazosina	Organic Chemical	doxazosina
	C0560175	Assintomático	Finding	assintomatico
	C0087111	Tratamento	Therapeutic or Preventive Procedure	tratamento
	C0060389	Finasterida	Organic Chemical	finasterida

[a] Tipo semântico no caso do método CTAKES é o foi apresentado pela interface do software CAS Visual Debugger.

prostática benigna) e a segunda considerou na contagem apenas o conceitos encontrados com dois ou mais *tokens* (ex.: hiperplasia prostática, hiperplasia prostática benigna). A Tabela 5.7 ilustra a primeira matriz para o texto de exemplo.

5.5 Vetorização e representação dos vetores

Em relação à vetorização dos textos para processamento pelos modelos de classificação, foram utilizadas as técnicas de *Word Count* e *TFiDF* e representações *Bag-of-Words* e *Bag-of-ngrams*, sendo elas aplicadas aos textos gerados pelos pré-processadores 1, 2 e 3. Também foram utilizados modelos de *Word Embeddings*, aplicados aos textos dos vetorizadores de 1 a 5. Os textos enriquecidos pelo procedimento descrito na seção 5.4 também foram submetidos a todas essas mesmas vetorizações.

Exceto para os *Word Embeddings*, a vetorização foi realizada com a biblioteca *ScikitLearn*. Os vetorizadores da *ScikitLearn* utilizados foram *CountVectorizer* e *Tfidf-Vectorizer*. Ambos aceitam parâmetros que alteram detalhes do processo de vetorização e alguns desses parâmetros foram explorados, gerando diferentes representações dos textos

Tabela 5.7: Matriz com a contagem de ocorrência dos tipos semânticos identificados no texto de exemplo, considerando-se identificações com 1+ tokens

QUADROCLINICO	**Organic Chemical**	**Patient or Disabled Group**	**Laboratory Procedure**	**Health Care Activity**	**Therapeutic or Preventive Procedure**	**Finding**	**Disease or Syndrome**	**Congenital Abnormality**	**Mental Process**	**Individual Behavior**	...	**Gene or Genome**
paciente com relatos hiperplasia prosta...	2	1	1	1	1	1	1	0	0	0	...	0

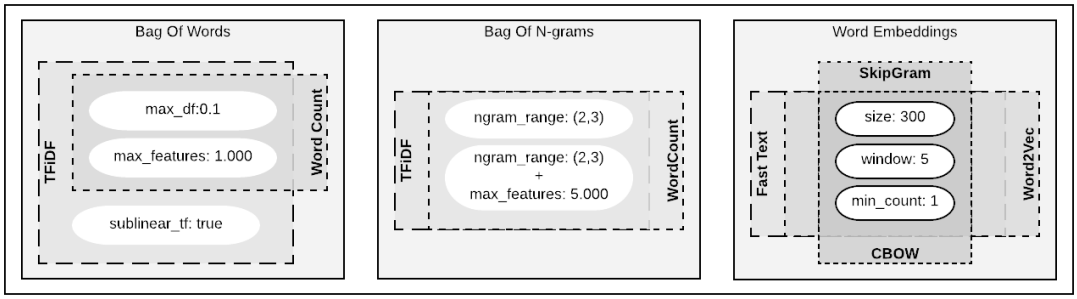
vetorizados. As variações utilizadas foram:

- max_df=0.1 - Que limita o vocabulário a termos cujo *document frequency* seja de até 10%
- max_features=1000 - Que limita o vocabulário aos 1000 termos com maior *term frequency*
- ngram_range=(2,3) - Que cria entradas no vocabulário contendo combinações de 2 e 3 palavras.
- ngram_range=(2,3) + max_features=5000 - Que combina a configuração de 2 e 3 n-gramas e limita o número aos 5000 mais frequentes.

A Figura 5.7 ilustra como essas variações foram combinadas com as diferentes representações dos vetores. Por exemplo, na representação *Bag-of-Words* duas das opções de parâmetros foram usadas tanto para TFiDF quanto para *Word Count*. Já a opção "sublinear_tf: True" só foi usada na vetorização com a técnica TFiDF.

Também foram realizados testes de classificação com diferentes modelos de *Word Embeddings*. Os primeiros testes foram baseados nos quatro modelos disponibilizados por DOS SANTOS et al. (2018), que contém textos clínicos em Português do Brasil produzidos com os algoritmos *Word2vec* e *FastText*, ambos com um versão em modo *CBOW* e outra em modo *Skipgram*. Essas variações também estão representadas na Figura 5.7.

Figura 5.7: Técnicas de vetorização, representações vetoriais e as respectivas variações exploradas



Cada um desses quatro modelos também foi expandido usando os textos disponíveis para os experimentos deste trabalho, gerando novos *Embeddings*.

5.6 Seleção e Transformação de Features

Os textos vetorizados com as técnicas de *TFiDF* e *Word Count* geram grandes matrizes esparsas representando as *features* dos documentos. A fim de selecionar aquelas mais relevantes para alimentar os classificadores, foram testadas algumas técnicas de seleção de *features* e de redução de dimensionalidade (já apresentadas na seção 2.2). Também foram realizados testes visando transformar as *features*, a fim escalar os valores e diminuir o impacto de eventuais discrepâncias entre eles.

Para a transformação das *features* foi testada a técnica LSA - *Latent Semantic Analysis* (cuja implementação na biblioteca *Scikit Learn* é chamada *Truncate SVD*), que reduziu os vetores a 100 componentes, isto é, 100 *features* para cada documento. Além disso, duas versões do *Quantile Transformer*, uma gerando distribuição normal e outra gerando distribuição uniforme, foram testadas a fim de escalar os valores das *features*, diminuindo a variância dos seus valores.

A técnica de seleção de *features* usada foi a *SelectKBest* com duas diferentes métricas: *chi-square* e Informação Mútua e três diferentes limites de *features* para cada uma: 1000, 2000 e 5000.

As técnicas foram aplicadas aos dados gerados na fase de vetorização, resultando em versões modificadas dos vetores que representavam os documentos. Nem todas as técnicas foram aplicadas a todos os dados pré-processados, por exemplo, para um texto vetorizado com o parâmetro *max_features=1000* a técnica *SelectKBest* não foi aplicada, já que a quantidade de *features* no vetor já era 1000 (o menor valor usado nos testes com

Figura 5.8: Técnicas de Seleção e Transformação de *Features* e as respectivas variações exploradas

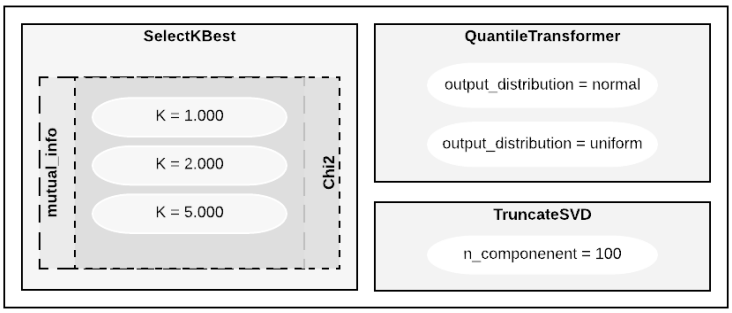
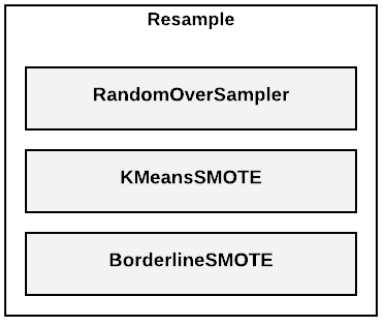


Figura 5.9: Técnicas de Resampling utilizadas nos experimentos



SelectKBest). A Figura 5.8 ilustra as técnicas e as variações de parâmetros exploradas.

5.7 Resampling

Conforme se pode observar na seção 5.1, existe um desbalanceamento entre as classes "positiva" e "negativa" no *dataset* utilizado neste trabalho. A fim de minimizar o impacto desse desequilíbrio no treinamento dos classificadores, técnicas de *resampling* foram utilizadas para reequilibrar a quantidade de exemplos em cada classe.

Foram utilizadas técnicas de *oversample* com diferentes métodos, conforme ilustra a Figura 5.9. Nessas técnicas, fundamentalmente, novos exemplos da classe positiva (encaminhamentos Aprovados) foram criados na porção de treinamento do *dataset*, gerando uma versão equilibrada do conjunto que era então utilizada para treinamento de um classificador.

A implementação usada para essas técnicas foi provida pela biblioteca $imbalanced-learn$, apresentada em LemaÎtre, Nogueira e Aridas (2017), por ser plenamente integrada[3]

[3]http://contrib.scikit-learn.org/

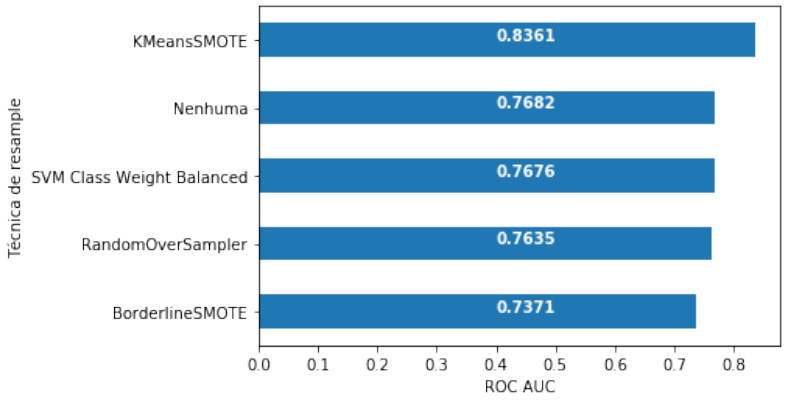
à biblioteca *Scikit Learn* já em uso nos experimentos.

Conforme demonstrado na Figura 5.1, a etapa de *resampling* é realizada após a vetorização, implicando que os dados de entrada para as técnicas de *resampling* foram os diferentes vetores numéricos produzidos na etapa de vetorização.

Além dessas técnicas aplicadas ao *dataset* antes de utilizá-lo para treinamento, para o classificador SVM também foi avaliado o uso do parâmetro *class_weight='balanced'* que procura ajustar a penalização do erro de acordo com o tamanho de cada classe no *dataset*.

Na tentativa de identificar qual das técnicas teria melhor impacto nos dados deste trabalho, um mesmo conjunto de dados vetorizado foi submetido a um classificador SVM passando antes por cada uma das diferentes técnicas. O resultado é ilustrado na Figura 5.10.

Figura 5.10: ROC AUC média para um mesmo conjunto de dados e classificador, com diferentes técnicas de *resampling*



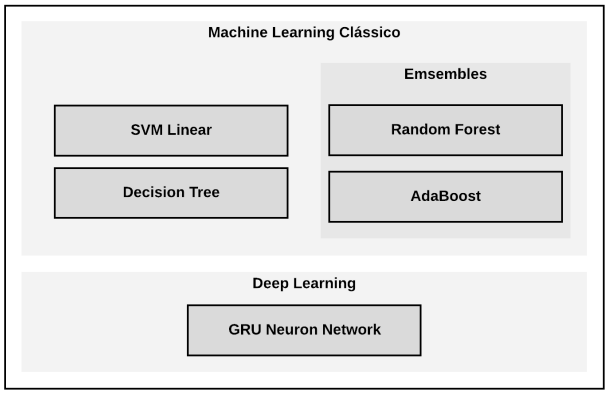
Como se observa, o recurso implementado pelo própria biblioteca (*Class Weight Balanced*) teve o mesmo impacto de não realizar qualquer tratamento (Nenhuma). As técnicas de *RandomOverSampler* e *BorderlineSMOTE* acabaram piorando o resultado em relação a não aplicação de um tratamento.

Por fim, os resultados mostram que a técnica *KMeansSMOTE* teve maior impacto, melhorando em $0,07$ a média da ROC AUC em relação a não aplicação de um tratamento. Assim, essa técnica foi aplicada nos testes subsequentes.

5.8 Treinamento e Avaliação dos Modelos

Independentemente das etapas executadas para processamento dos textos, a etapa final do experimento era a de treinamento e avaliação de um modelo de classificação

Figura 5.11: Modelos de classificação testados



utilizando os dados resultantes desse processamento. A Figura 5.11 apresenta os modelos de classificação que foram utilizados durante o experimentos. Cada um desses modelos foi usado em momentos diferentes, isto é, nem todos os conjuntos de dados processados foram submetidos a todos eles, já que alguns conjuntos foram sendo descartados por apresentarem resultados inferiores e outros foram criados especificamente para alguns modelos.

As avaliações foram baseadas no resultados da ROC AUC reportado para a classificação. Exceto quando utilizadas redes neurais, a avaliação foi feita usando a técnicas de validação cruzada, com 10 *folds*, por ser essa das formas comuns de apresentar resultados em estudos semelhantes. Para o caso das redes neurais, dado o custo computacional necessário para utilizar validação cruzada, optou-se pela técnica de *holdout*, criando-se um conjunto estratificado com 20% dos dados usados no treinamento. O conjunto de *holdout* já existente não foi utilizado, pois era desejável que ele fosse usado numa etapa posterior de validação do modelo, permitindo uma comparação da aplicação dos melhores modelos sobre um mesmo conjunto de dados totalmente novo. Para os modelos que apresentaram os melhores resultados em tempo de treinamento e teste, uma validação adicional foi realizada usando os dados de *holdout* separados conforme a seção 5.2. Os resultados apresentados sempre indicam a métrica e a forma como foram obtidos.

Cada um dos modelos usados continha algum conjunto de parâmetros que poderia ser ajustado a fim de melhorar o seu desempenho. Alguns desses parâmetros foram explorados em combinação com diferentes versões de textos processados. A medida que os resultados reportados foram melhorando, esses parâmetros foram sendo fixados para os testes seguintes.

O modelo mais utilizado foi o SVM Linear, na implementação disponibilizada

pela biblioteca *Scikit Learn*[4]. Ele serviu como uma espécie de *baseline* para avaliação dos resultados. Entre a motivações para que esse algoritmo tenha sido escolhido como *baseline* pode-se citar os bons resultados reportados com seu uso (ex.: Sarker e Gonzalez (2015) e Mujtaba et al. (2018)), bem como a sugestão da própria *Scikit Learn* em seu artigo "*Choosing the right estimator*" [5]. O principal parâmetro a ser configurado no caso do SVM Linear é a penalização ao erro (parâmetro “C”).

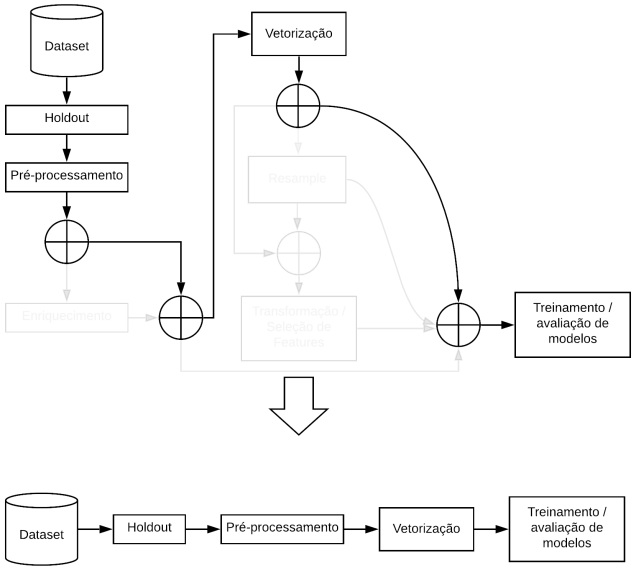
Como um hiper-parâmetro, o valor de “C” foi determinado com base em testes de diferentes valores dentro de um espaço de busca com 21 possibilidades diferentes ($[2^{-5}... 2^{15}]$), uma sequência de crescimento exponencial que segundo Hsu, Chang e Lin (2003) é uma maneira prática de encontrar bons valores para este parâmetro.

5.8.1 Diferentes Fluxos

No início do Capítulo 5 ilustrou-se as várias etapas dos experimentos com os possíveis fluxos alternativos. Observado apenas um dos possíveis caminhos percorridos desde a entrada dos dados até a avaliação de um classificador, pode-se considerar tal fluxo com um caminho simplificado, conforme ilustra a Figura 5.12.

Nas seções seguintes os resultados serão apresentados precedidos de uma dessas visões simplificadas, indicando qual fluxo levou a tais resultados.

Figura 5.12: Uma visão simplificada de um dos possíveis fluxos de experimento



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

[5]Disponível em https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

5.8.2 Fluxo 1

Figura 5.13: Fluxo com apenas pré-processamento e vetorização dos dados

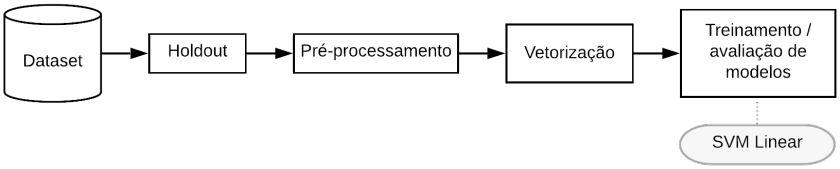


Figura 5.14: Fluxo 1 - Avaliado pela ROC AUC média (validação cruzada com 10 folds)

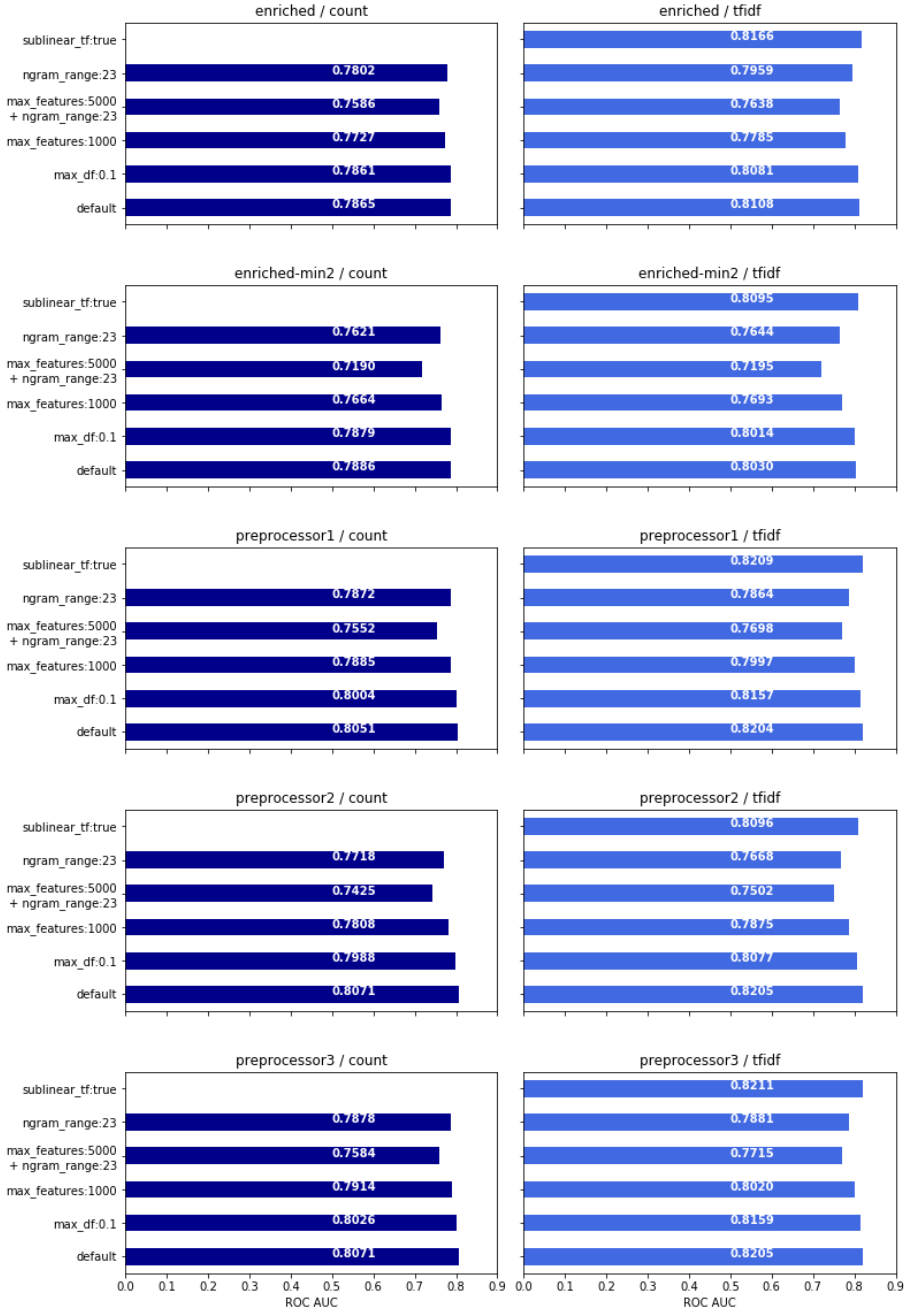


Figura 5.15: Melhores resultados para o Fluxo 1

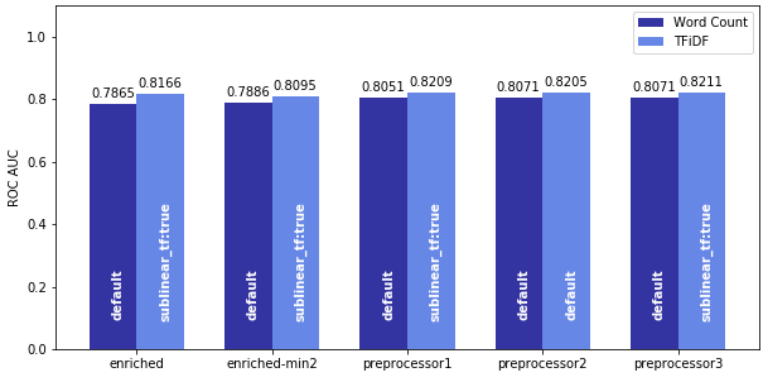


Tabela 5.8: Comparativo dos melhores resultados para o Fluxo 1

	ROC AUC				
	enriched	enriched-min2	preprocessor1	preprocessor2	preprocessor3
Média	0.817	0.809	0.821	0.821	0.821
Desvio Pad.	0.005	0.005	0.006	0.005	0.005
Mínimo	0.806	0.799	0.815	0.814	0.815
Máximo	0.824	0.817	0.833	0.831	0.831

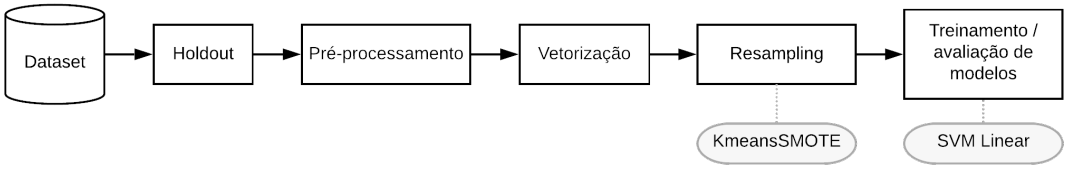
O preprocessor3 com vetorização usando TFiDF e o um valor ponderado da frequência de termos (parâmetro sublinear_tf como True) obtém o melhor resultado. Utilizando a técnica de *Word Count* também o melhor resultado foi com o preprocessor3, sendo que o mesmo valor foi obtido com o preprocessor2.

A técnica de *Word Count* teve piores resultados independentemente do pré-processamento realizado quando comparada com a técnica de TFiDF. A diferença, é em média de $0,0158$. A maior diferença foi com o preprocessamento 'enriched', onde a média da ROC AUC variou $0,0301$.

Interessante observar também que, com a técnica de *Word Count*, a vetorização *default* sempre teve melhores resultados que as alterativas. Esse mesmo comportamento ocorreu com a técnica de TFiDF quando usado o preprocessor2 (único pré-processamento que incluiu lematização), enquanto nos demais casos, ponderar a frequência dos termos sempre ajudou a classificação.

5.8.3 Fluxo 2

Figura 5.16: Visão simplificada do fluxo do experimento incluindo *resampling* dos dados



Este fluxo distingue-se do primeiro pela aplicação do *resampling* nos dados, usando a técnica KmeansSMOTE, que como visto na seção 5.7, é a que apresenta melhor resultado para os dados disponíveis.

Tabela 5.9: Comparativo dos melhores resultados para o Fluxo 2

	ROC AUC				
	enriched	enriched-min2	preprocessor1	preprocessor2	preprocessor3
Média	0.878	0.876	0.876	0.877	0.877
Desv. Pad.	0.006	0.006	0.004	0.005	0.005
Mínimo	0.868	0.868	0.871	0.869	0.869
Máximo	0.886	0.884	0.882	0.884	0.884

De maneira geral, a média da ROC AUC melhorou em relação ao fluxo 1 que não incluía *resampling*. Em média, o aumento foi de $0,0639$ quando usado *Word Count* e $0,0565$ com TFiDF, com destaque para o 'enriched' e o 'enriched-min2', que aumentaram cerca de $0,08$ a média da classificação em relação àquele fluxo.

Apesar da melhora nos números, a ordem dos melhores resultados quase não mudou. O pré-processador *enriched* superou preprocessor3, que havia se saído melhor antes do *resampling*, mas com uma diferença muito pouco significativa. A vetorização usando TFiDF ainda apresentou melhor resultado do que sua alternativa direta com *Word count*, porém agora a vetorização *default* superou as demais variações para a maioria dos casos.

A técnica de *Word Count*, que antes sempre foi melhor com vetorização *default*, agora teve a variação que utiliza n-gramas com 2 e 3 palavras figurando entre os melhores para a maior parte dos pré-processamentos.

No geral, os resultados com a técnica de TFiDF continuaram ligeiramente melhores do que a de *Word Count*. Quando se observa cada pré-processamento em separado percebe-se que a diferença entre os resultados de cada técnica continua pequena, em média na casa de $0,01$, tendo diminuído mais no pré-processamento 'enriched'.

Figura 5.17: Fluxo com pré-processamento, vetorização e resampling dos dados

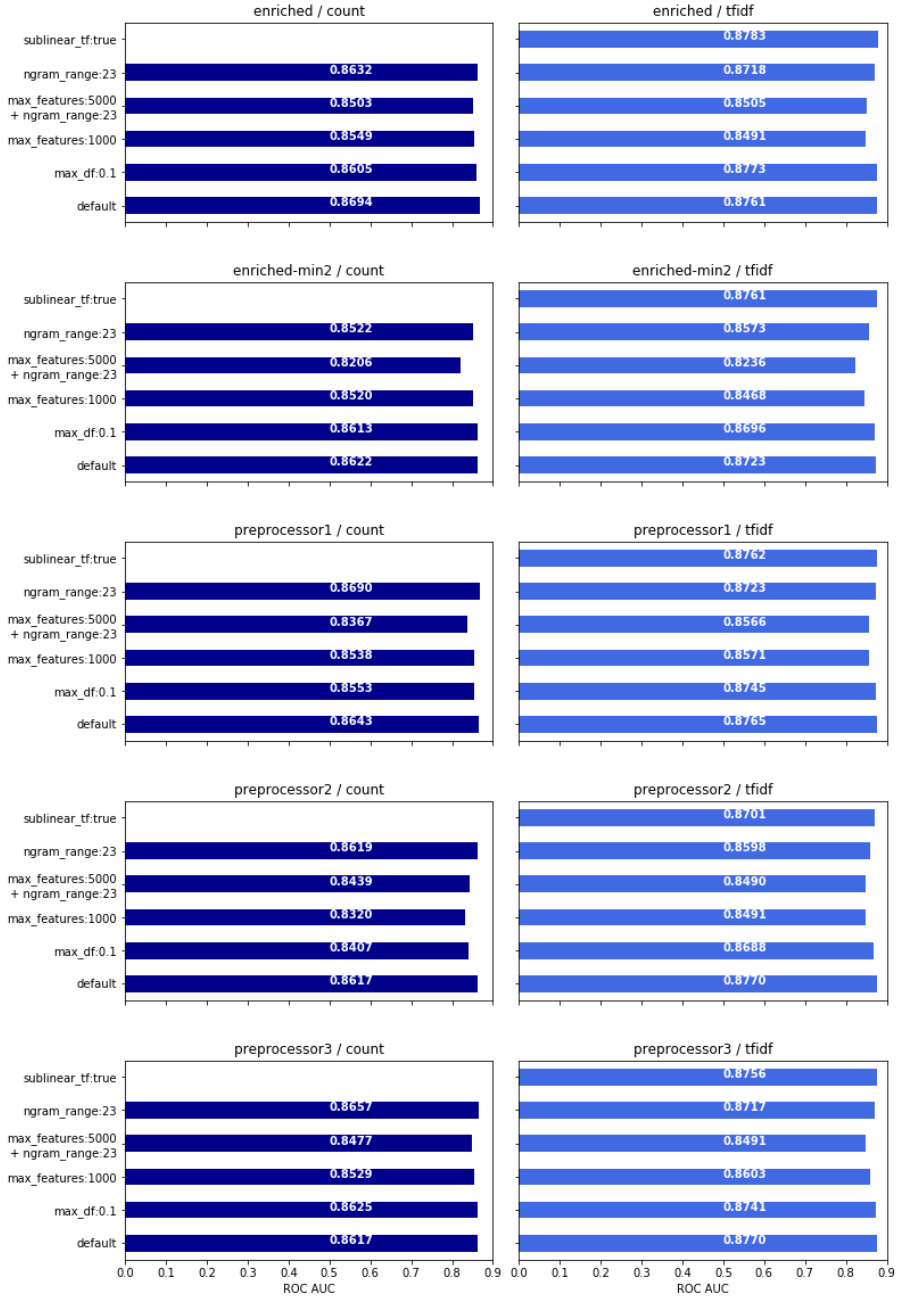
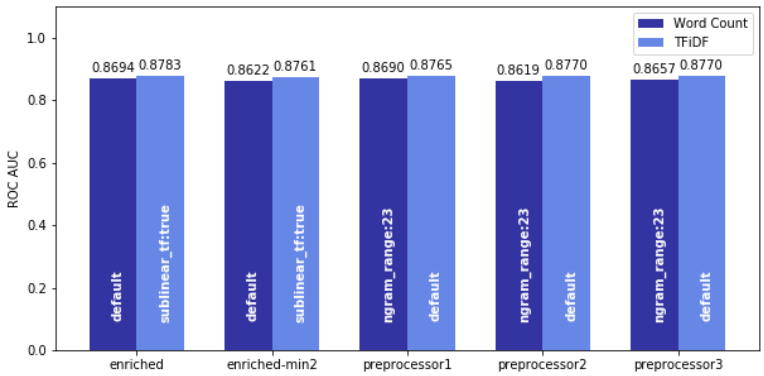


Figura 5.18: Melhores resultados para o Fluxo 2



5.8.4 Fluxo 3

Figura 5.19: Fluxo de experimento que incluiu transformação e seleção de *features*

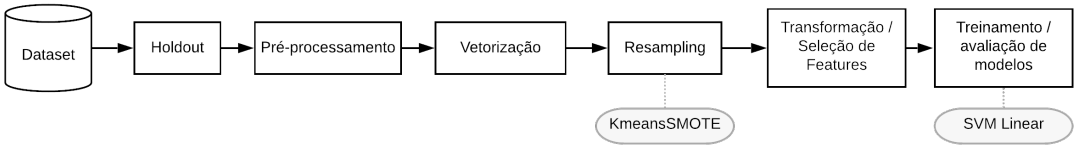
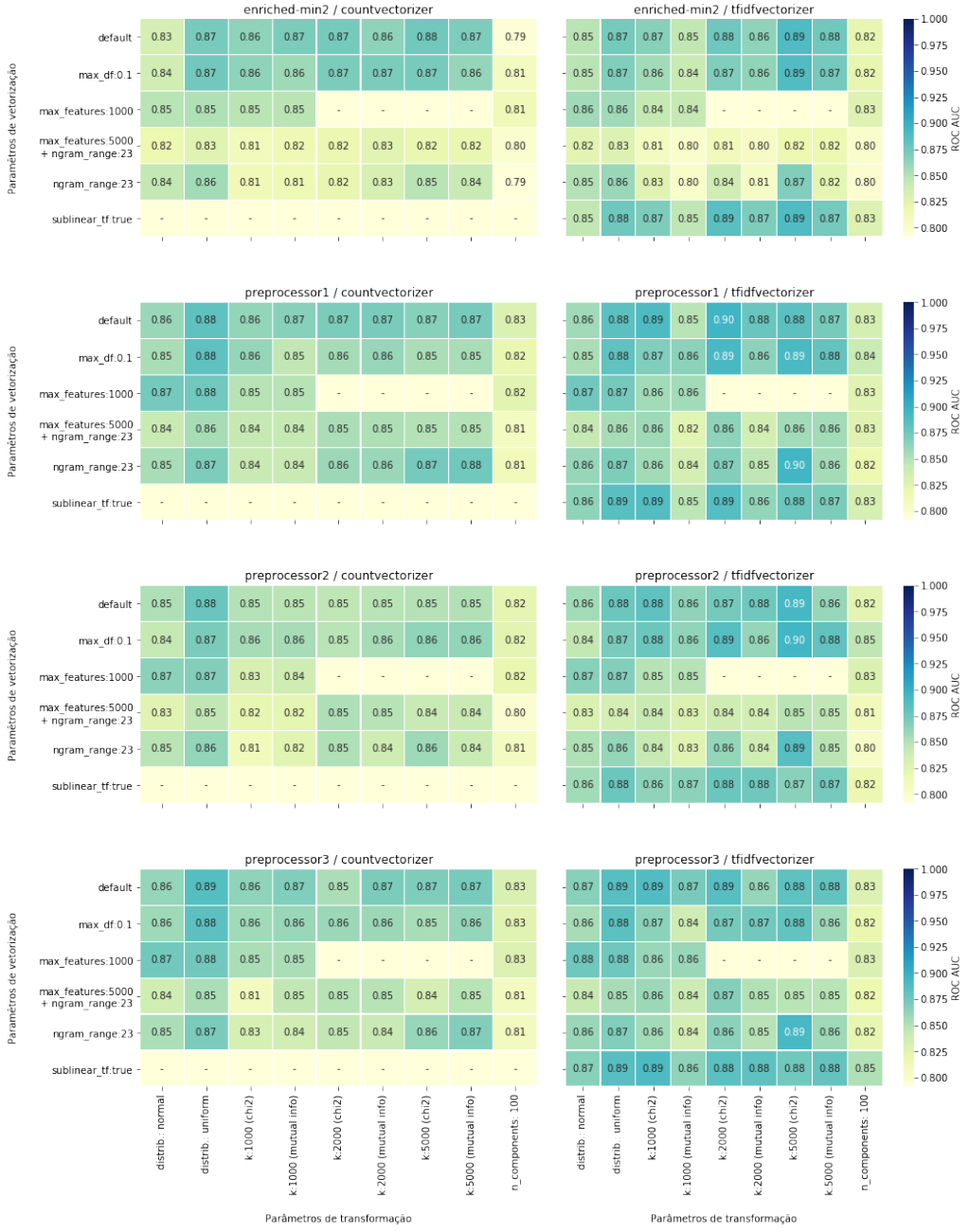


Figura 5.20: Resultados da classificação combinando diferentes pré-processamentos e técnicas de seleção de *features*)



O gráfico da Figura 5.20 apresenta os resultados em termos de ROC AUC média obtidos pela classificação usando um SVM Linear e validação cruzada de 10 folds.

Cada sub-gráfico corresponde a uma das combinações de pré-processamento e técnica de vetorização, as mesmas das seções anteriores com a exceção do pré-processamento 'enriched' não foi considerado nestes testes. Em cada sub-gráfico, a linha representa a técnica de vetorização (as mesmas testadas nos fluxos anteriores) e as colunas representam a técnica de transformação ou seleção de *features*.

As técnicas aplicadas são as representadas na Figura 5.8, a identificação das colunas está abreviada e os *labels* correspondem ao seguinte: *distrib: normal* e *distrib: uniform* referem-se, respectivamente, a aplicação da transformação usando quantis (*QuantileTransformer*) com distribuição normal e uniforme. As identificações *K:1000 (chi2)* e *K:1000 (mutual info)* correspondem a seleção de *features* com K*-best* limitado a 1000 *features*, usando a métrica *chi-square* ou *mutual info*, respectivamente. O mesmo vale para os demais números de *features* (2000 e 5000). Por fim, o *label n_components: 100* refere-se a aplicação do *TruncateSVD* com 100 componentes.

Analisando-se o gráfico é possível observar que a coluna *distrib:uniform* tem os melhores resultados quando usado *Word Count*, independentemente do pré-processamento envolvido ou dos parâmetros usados na vetorização. Ainda pelo tom de cor das colunas, é possível perceber que a técnica o *TruncateSVD* tem consistentemente o pior resultado quando comparada às demais técnicas aplicadas, tanto do lado do *Word Count* quanto nos gráficos do TFiDF.

Ao atentar para as linhas dos gráficos, vê-se uma concentração de melhores resultados nas técnicas de vetorização mais ao topo (default, e max_df:0.1), sobretudo no lado do *Word Count*, onde o sublinear_tf não é aplicável. Essa concentração é notada principalmente nos textos com pré-processamento 'enriched-min2'. A linha da combinação *max_features: 5000 + ngram_range:23* quase sempre apresenta tons mais claros que as a demais linhas em todos os gráficos.

Por fim, observando-se a Figura 5.20 como um todo se percebe tons de verde mais escuros nos gráficos lado do TFiDF, acompanhando os resultados dos fluxos anteriores, contudo aqui os melhores resultados são encontrados com o preprocessor 1 e preprocessor2 e não mais com o preprocessor3.

Os resultados da aplicação de técnicas de seleção e transformação de *features* apresentam melhora tímida, de aproximadamente $0,03$, na ROC AUC média. O melhor resultado obtido antes da aplicação das técnicas era 0.870 e aqui se obteve 0.896 na "Con-

Figura 5.21: Melhores resultados para o Fluxo 3

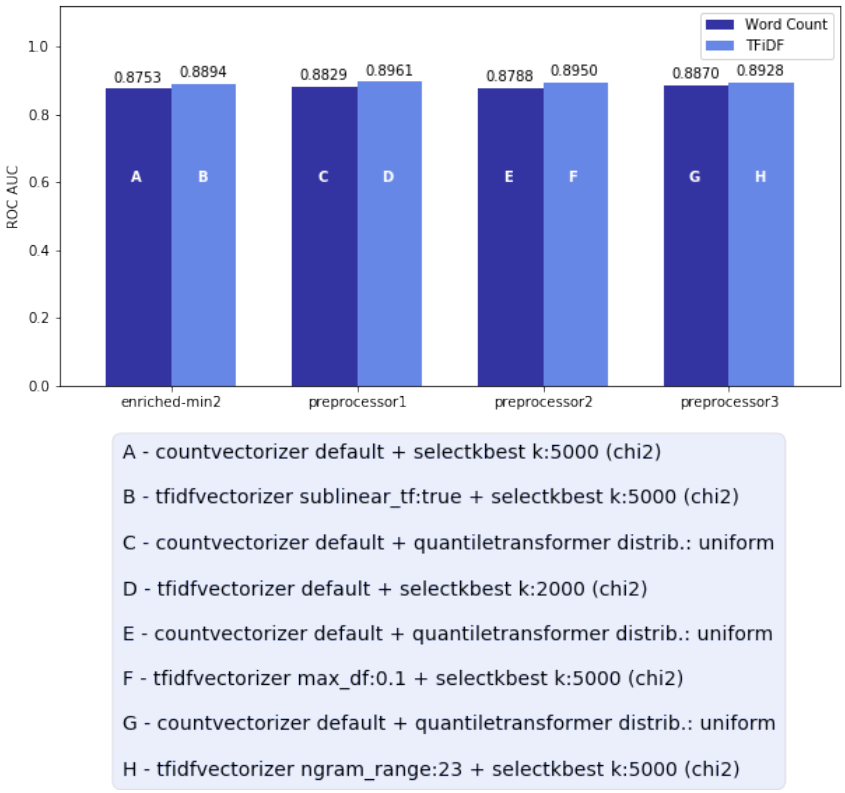


Tabela 5.10: Comparativo dos melhores resultados para o Fluxo 3

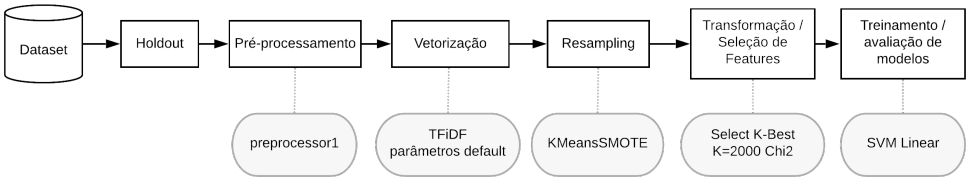
	ROC AUC			
	enriched-min2	preprocessor1	preprocessor2	preprocessor3
Média	0.893	0.896	0.896	0.893
Desvio Pad.	0.004	0.005	0.004	0.004
Mínimo	0.889	0.890	0.889	0.886
Máximo	0.900	0.903	0.899	0.898

figuração D", em que se aplicou a técnica de seleção de *features Select K-Best* com um limite de 2000 *features*, utilizando a métrica *Chi-square*.

Verifica-se, tanto pela Figura 5.20 quanto pelo seu resumo na Figura 5.21, que de maneira geral as diferenças entre os resultados são sempre bastante pequenas, independentemente das combinações aplicadas.

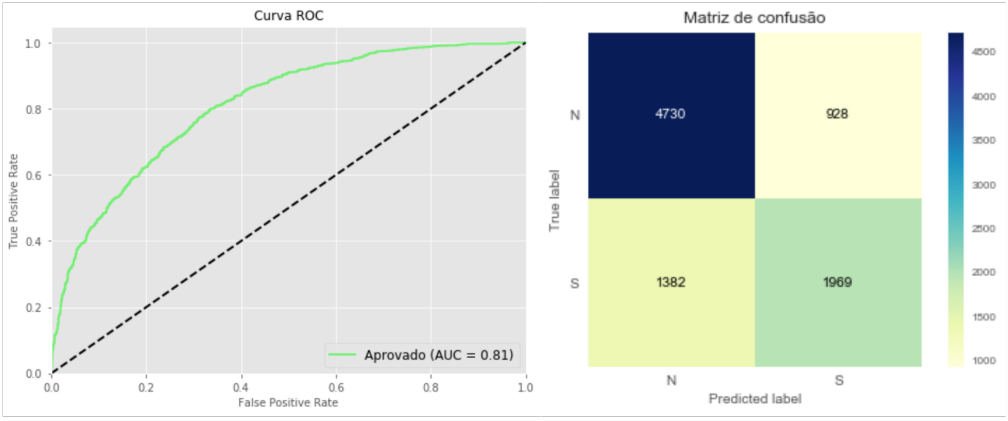
Apesar de serem pequenas as diferenças de resultados entre todas as variações testadas conclui-se que: Utilizando o SVM Linear, o melhor desempenho na classificação é alcançado utilizando-se os dados submetidos ao preprocessor1, com dados vetorizados com a técnica de TFiDF, utilizando os parâmetros *default* do vetorizador, seguido pelo balanceamento das classes usando a técnica KMeansSMOTE e pela posterior seleção das melhores *features* utilizando a técnica *Select K-Best* com um limite de 2000 *features*, e métrica *Chi-square*. Essa configuração é representada na Figura 5.22.

Figura 5.22: Melhor configuração do Fluxo 3, que usa SVM Linear



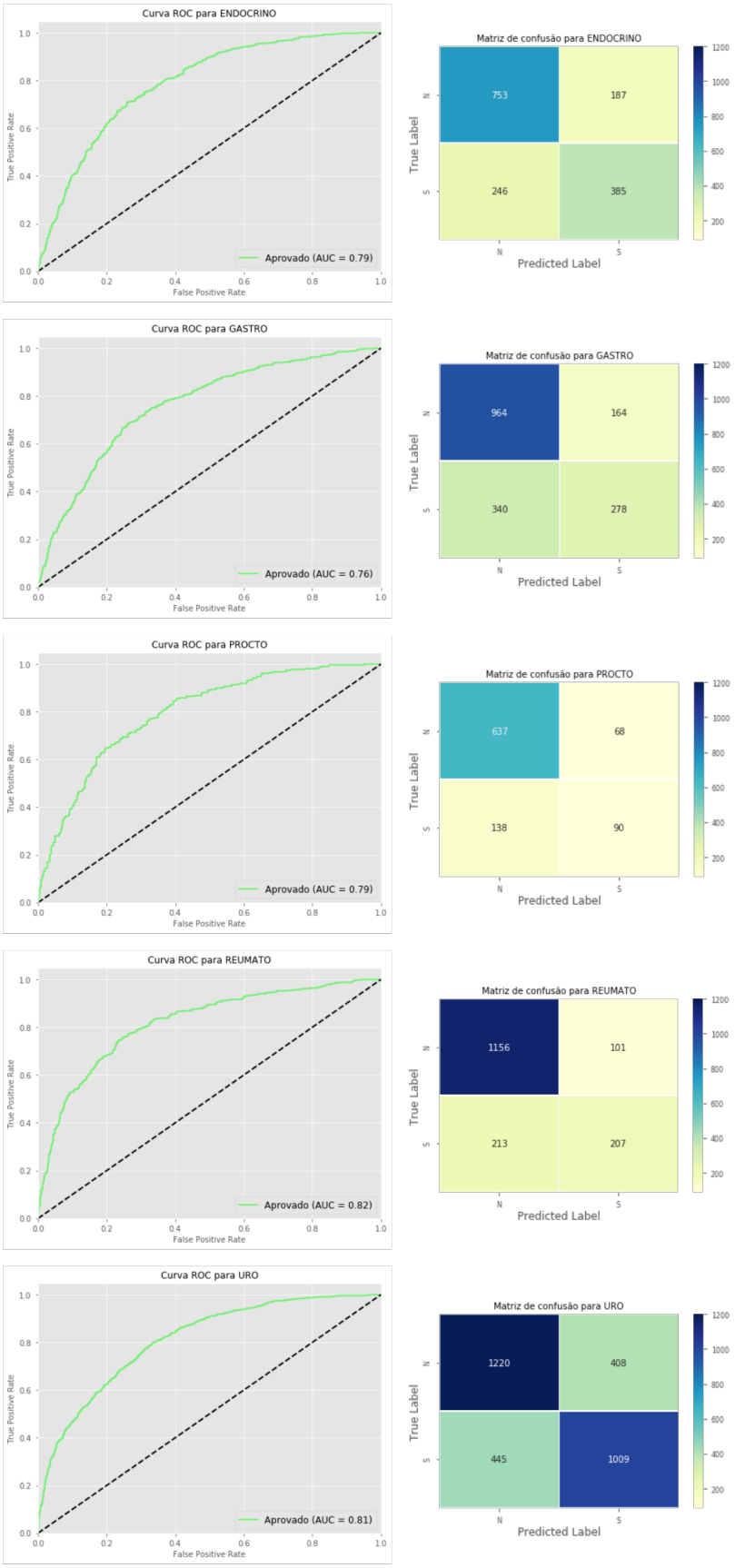
Após identificar qual a melhor configuração através de testes usando a técnica de validação cruzada, uma avaliação final foi executada nos dados de *holdout*, separados conforme descrito na seção 5.2. Os detalhes são apresentados a seguir.

Figura 5.23: Resultados para o melhor classificador do Fluxo 3 quando aplicado aos dados de *Holdout*



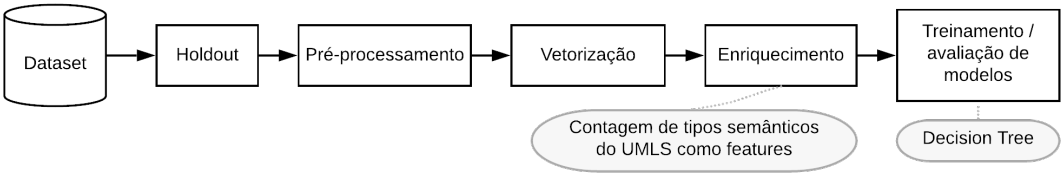
Ao se observar os resultados individualizados por especialidade, na Figura 5.24 vê-se que a curva ROC tem resultados abaixo dos 0.81 para Gastro, Reumato e Procto, os quais são compensados por resultados maiores nas especialidades de Reumato e URO.

Figura 5.24: Resultados do melhor classificador do Fluxo 3 por especialidade (*Holdout*)



5.8.5 Fluxo 4

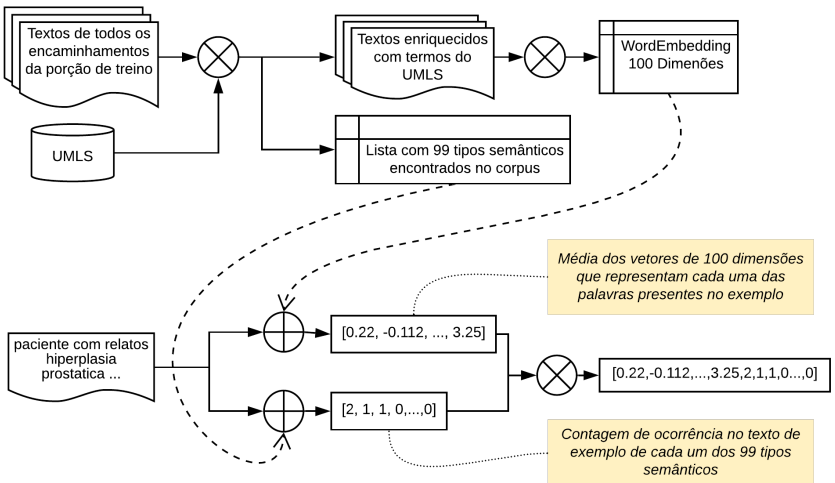
Figura 5.25: Fluxo de experimento utilizando Tipos Semânticos extraídos do UMLS em um classificador *Decision Tree*



Para o fluxo 4 foram usados os dados gerados pelo pré-processador *enriched*, cujo texto havia sido enriquecido com dados extraídos do UMLS, conforme ilustrado na Tabela 5.5. Os textos já enriquecidos foram usados para a construção de um *Word Embedding* de 100 dimensões usando o algoritmo Word2Vec e a técnica SkipGram.

Usando este *Word Embedding*, os textos foram então vetorizados e cada exemplo de encaminhamento passou a ser representado por um vetor de 100 dimensões. A esse vetor de 100 dimensões, acrescentou-se outras 99 que representavam a contagem de tipos semânticos encontrados naquele exemplo de encaminhamento. Esse processo é ilustrado na figura 5.26.

Figura 5.26: Detalhes da geração do vetor utilizado no Fluxo 4



O vetor resultante, de 199 dimensões, foi submetido a diferentes configurações dos algoritmos de *Decision Trees* e AdaBoost (também com *Decision Trees* como classificador base) e os resultados são apresentados na figuras 5.27 e 5.28.

Figura 5.27: Curvas ROC obtidas pela classificação usando a contagem de ocorrência dos tipos semânticos em diferentes configurações de DecisionTrees

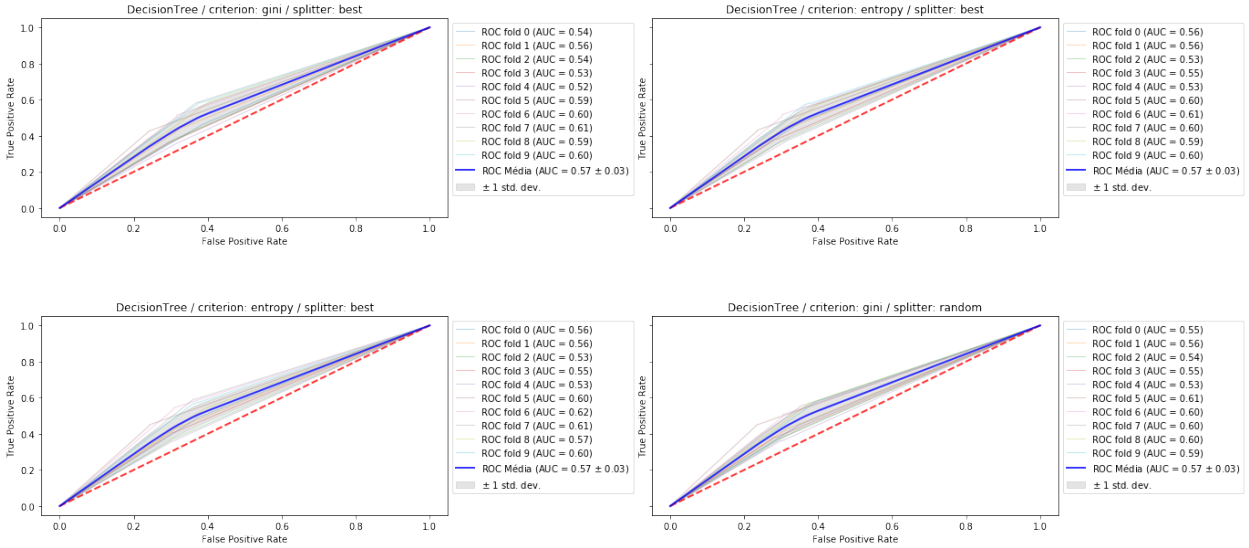
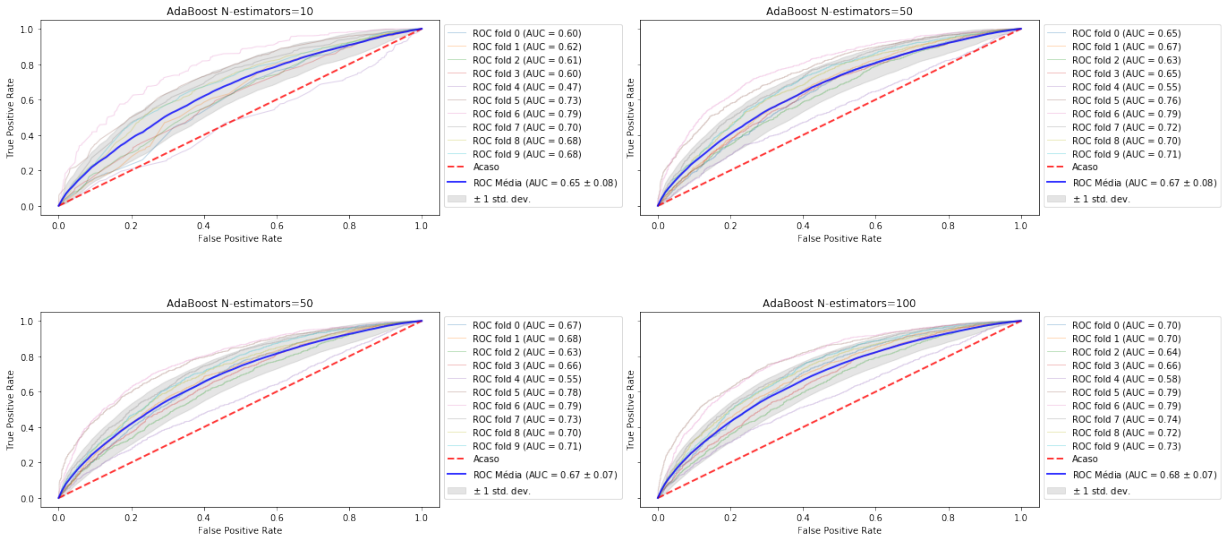


Figura 5.28: Curvas ROC obtidas pela classificação usando a contagem de ocorrência dos tipos semânticos em diferentes configurações de AdaBoost



Os resultados, avaliados com técnica de validação cruzada, apresentam valores de média de 0.57 e desvio padrão de +-0.03 para todas as configurações de *Decision Trees* testadas. Esses valores sobem para 0.68 e +-0.07, respectivamente, na melhor configuração de AdaBoost testada (N-estimators=100).

Conforme se verifica por esses números, os resultados obtidos com esse fluxo em nenhum caso são melhores do que aqueles já encontrados até aqui. Dessa forma, a aplicação desse fluxo sobre os dados de *holdout* não foi realizada.

5.8.6 Fluxo 5

Figura 5.29: Fluxo de experimento utilizando AdaBoost, sem e com resampling

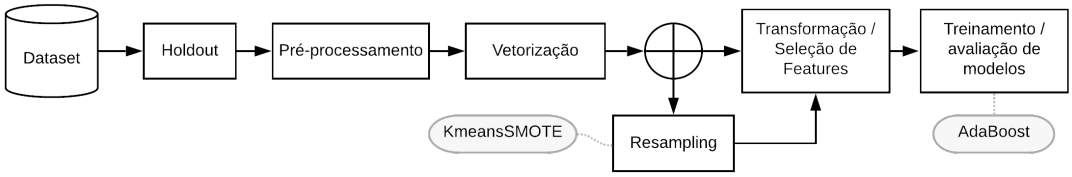


Figura 5.30: Resultados de experimento utilizando AdaBoost sem resampling

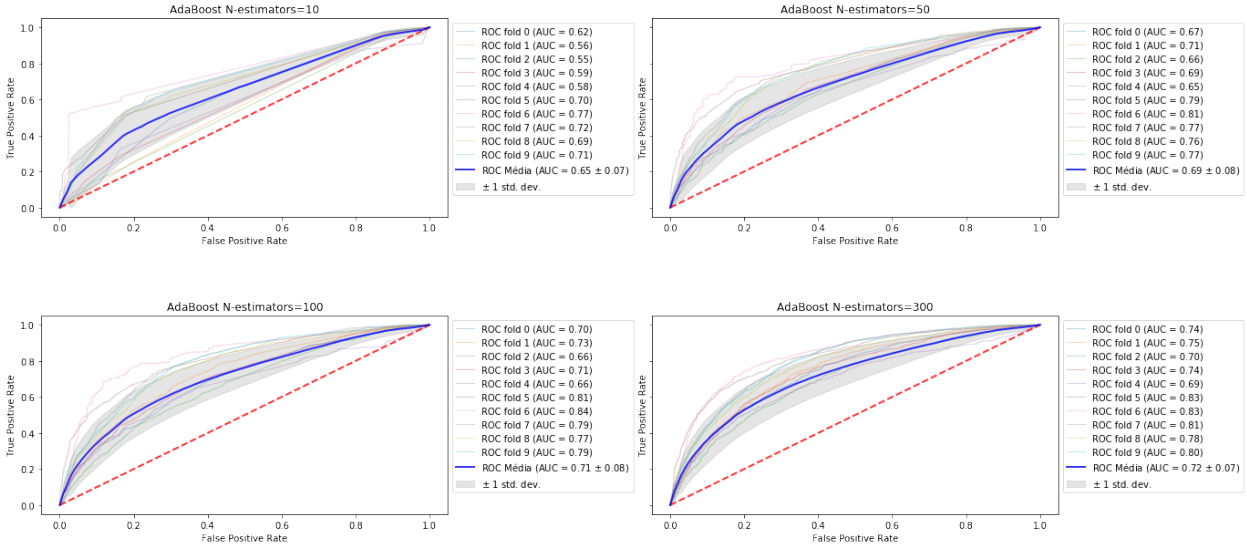
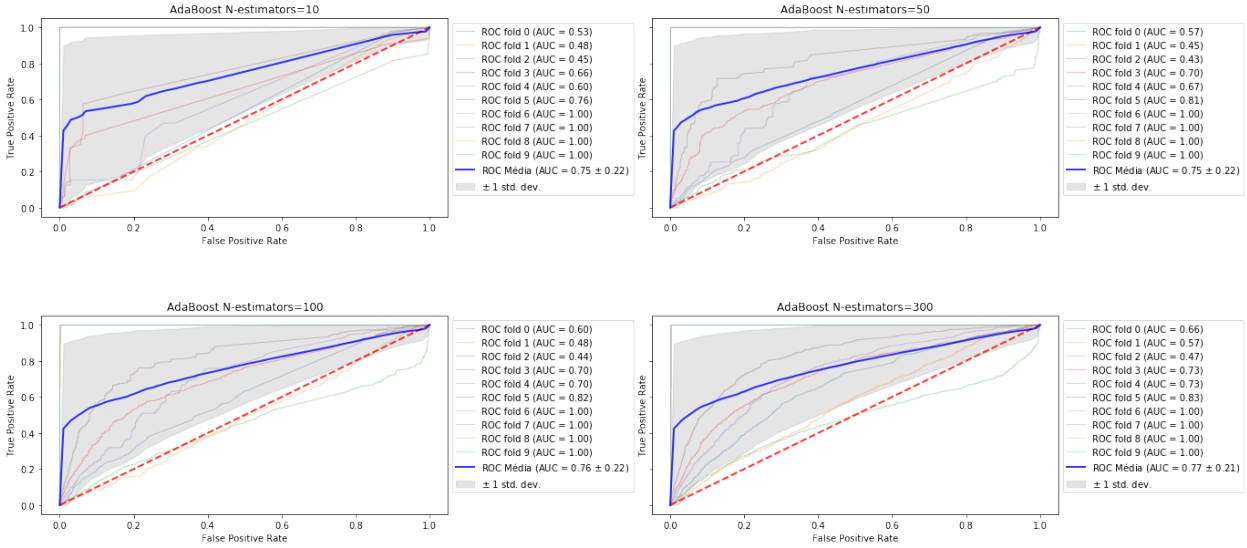


Figura 5.31: Resultados de experimento utilizando AdaBoost com resampling



Neste cenário, juntou-se os resultados de dois possíveis fluxos para destacar algo

que até aqui não havia sido analisado: O impacto do *resampling* no desvio padrão das classificações.

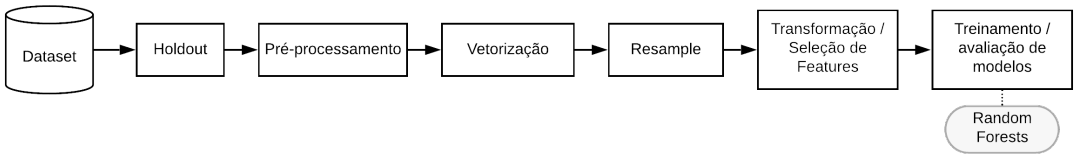
Os dados utilizados nas figuras 5.30 e 5.31 são os mesmos que obtiveram o melhor resultado para o SVM no Fluxo 3: Gerados pelo preprocessor1, vetorizados com TFiDF, usando os parâmetros *default*, com seleção de *features* feita pela técnica *Select K-Best* com 2000 *features* e métrica *Chi-square*. Os parâmetros explorados para o classificador são os mesmos entre as figuras e vão indicados acima de cada um dos gráficos. Assim, a única diferença entre processo que gerou os resultados de cada uma das duas figuras é o *resampling* usando KMeansSMOTE.

O que se observa quando o *resampling* é adicionado ao fluxo é um aumento da área sob a curva ROC, se considerada a curva média dos 10 *folds*. Os valores passam de 0.72 para 0.77 no melhor caso de cada figura. Entretanto, é claro também o aumento no desvio padrão dos resultados, indicando que a inclusão do *resampling* introduz uma variância significativamente maior nos resultados.

Novamente os resultados obtidos com o fluxo em nenhum caso são melhores do que aqueles já encontrados. Assim, não foram realizados testes com os dados de *holdout*.

5.8.7 Fluxo 6

Figura 5.32: Fluxo de experimento utilizando *Random Forests*

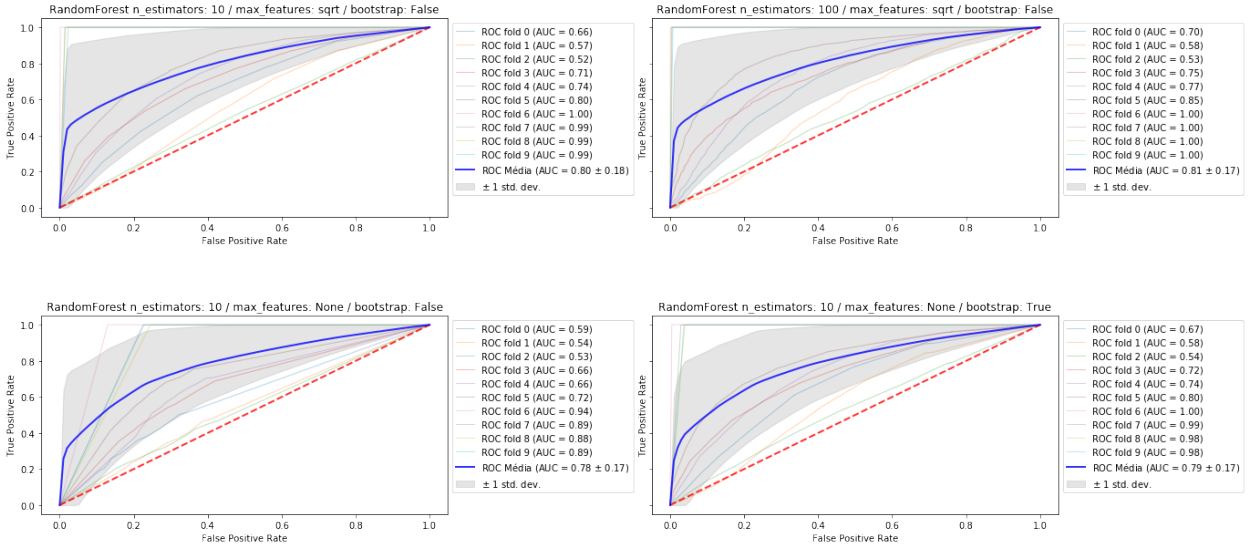


Para a experimentação com o algoritmo de *Random Forests* utilizou-se mais uma vez os mesmo dados que tiveram melhor resultado com SVM seguido pela aplicação do resample com KMeansSMOTE. Testes com dados preparados com outros preprocessadores e outras técnicas de seleção também foram realizados aqui, mas os resultados foram ligeiramente inferiores a estes reportados na Figura 5.33.

Como antes, se observa um considerável desvio padrão nos resultados apresentados pelos diferentes folds, mas com uma ROC AUC média chegando a 0.81 no caso da configuração com 100 árvores (sem *Bootstrap* e com limite de *features*).

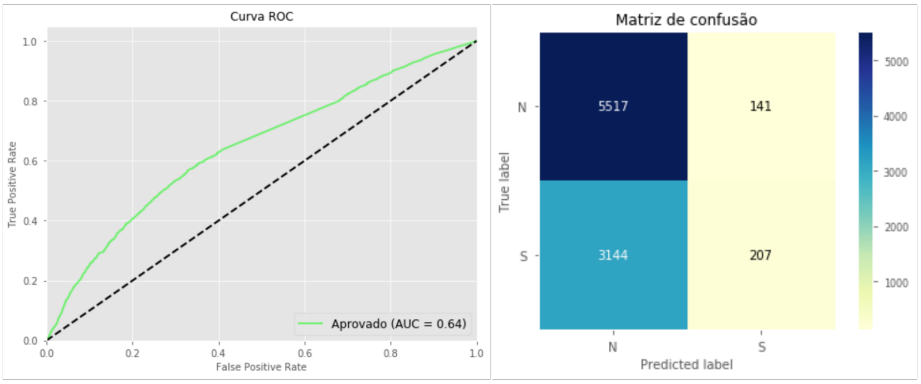
Sendo 0.81 um resultado mais próximo do encontrado com SVM, realizou-se um

Figura 5.33: Resultados de experimento utilizando *Random Forests*



teste sobre os dados de Holdout usando este classificador. Os resultados são apresentados na Figura 5.34

Figura 5.34: Resultados do classificador do Fluxo 6 aplicado aos dados de *Holdout*

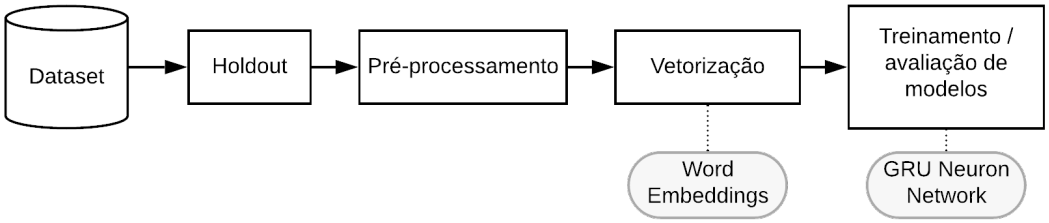


Conforme se verifica, os resultados apresentados sobre os dados de Holdout resultam numa ROC AUC de 0.64 e, portanto, não mantém a mesma qualidade de classificação obtida nos testes com validação cruzada. O decréscimo em relação ao valor (0.81) obtido nos testes com quele método de avaliação é de 0.17, maior do que o que se viu nos testes com SVM, onde a variação foi de 0.9 entre o valor de ROC AUC obtido com validação cruzada e com Holdout.

Testes individualizados por especialidade foram realizados também aqui, com os seguintes resultados de ROC AUC: 0.68 para Endocrino, 0.60 para Gastro, 0.59 para Procto, 0.66 para Reumato e 0.62 para Uro.

5.8.8 Fluxo 7

Figura 5.35: Fluxo de experimento utilizando *Word Enbeddings* e Rede Neural



Para os dados vetorizados com a técnica de *Word embeddings* a classificação foi realizada utilizando uma rede neural profunda. A rede foi implementada usando as bibliotecas Tensorflow[6] e Keras [7].

Após alguns testes preliminares com diferentes tipos de redes recorrentes, incluindo LSTM, Redes Convolucionais e GRU, tendo-se como critério a observação dos valores iniciais de acurácia das redes nesses testes, optou-se por investir em um modelo usando GRU.

Foi definida uma configuração inicial básica para a rede e essa mesma configuração foi mantida para os testes com os diferentes *embeddings* produzidos. A topologia utilizada é a da Figura 5.36, a mais simples possível, pois incluía apenas a camada de *embedding* e a própria GRU como camadas ocultas.

A camada GRU da rede usou 32 neurônios[8] e os parâmetros de *dropout* e *recurrent_dropout* foram definidos como 0.2[9]. A rede teve a função $sigmoide$ como ativação na camada de saída - para garantir valores entre 0 e 1, representando a classe predita - e a função de custo foi a *binary_crossentropy*[10] enquanto o otimizador foi o *adam*[11]. O tamanho de lote utilizado foi 128, um tamanho intermediário entre os que viriam a ser testados

[6]Uma das principais bibliotecas de código aberto para desenvolver e criar modelos de Aprendizado de Máquina. Disponível em https://www.tensorflow.org/?hl=pt-br

[7]Uma biblioteca que fornece uma API de alto nível que facilita o uso TensorFlow. Disponível em https://keras.io/

[8]Uma sugestão retirada de https://towardsdatascience.com/machine-learning-word-embedding-sentiment-classification-using-keras-b83c28087456

[9]Sugerido como bom ponto de partida em https://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/

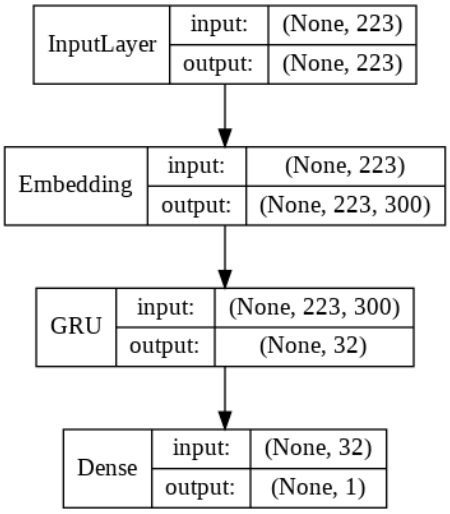
[10]Recomendada pela documentação do keras para classificações binárias (https://keras.io/api/losses/probabilistic_losses/binarycrossentropy-class), essa função é didaticamente explicada em https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a

[11]Uma das opções mais recomendadas para problemas de NLP, conforme explicado em https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

na etapa seguinte, e o treinamento foi limitado a 25 épocas, podendo ser interrompido antes disso se não houvesse melhora no custo por 5 épocas seguidas.

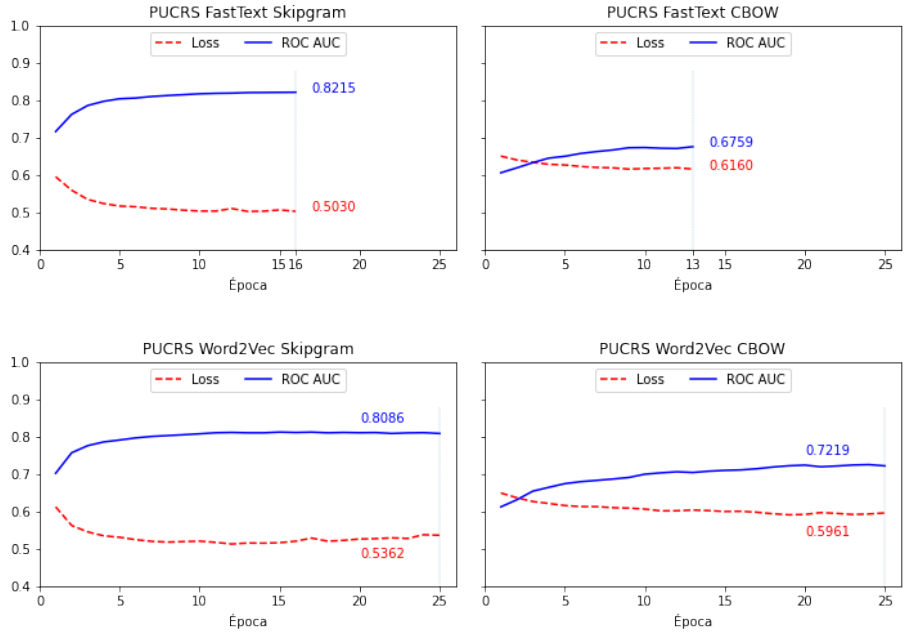
As redes foram treinadas com 28822 exemplos (80% dos dados disponíveis para treino) e a ROC AUC apresentada é a obtida pela classificação de outros 7208 exemplos.

Figura 5.36: Topologia da rede usada na avaliação dos *Word embeddings*



Inicialmente, como já apontado, testou-se a aplicação dos modelos já prontos de DOS SANTOS et al. (2018). Os resultados obtidos ao longo do treinamento são apresentados na Figura 5.37.

Figura 5.37: Resultados da aplicação dos *embeddings* de DOS SANTOS et al. (2018)

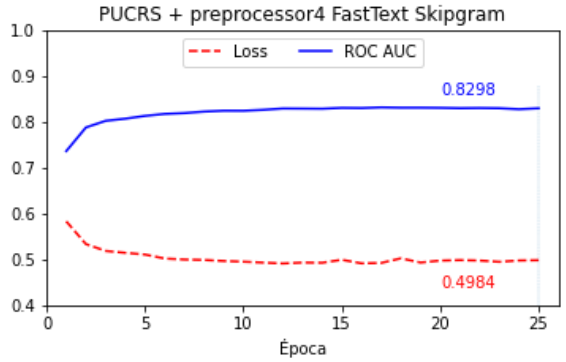


Verifica-se que com tais *embeddings* o melhor resultado em termos de ROC AUC foi o valor de 0.8215 obtido com o modelo que usou o algoritmo Fasttext e modo Skip-gram. O modelo deixou de melhorar após a 16ª época. O pior resultado foi o obtido com

FastText em modo CBOW, que não conseguiu reduzir o custo abaixo de 0.6160 e deixou de melhorar logo após a 13ª época.

A biblioteca utilizada para a geração dos modelos de DOS SANTOS et al. (2018) permite que o treinamento seja expandido atualizando o vetor de representação dos textos. Para valer-se dessa possibilidade de extensão, um novo pré-processador (preprocessor4) foi criado para gerar *tokens* no padrão dos encontrados nos modelos de DOS SANTOS et al. (2018) e uma versão estendida do *embedding* com melhor resultado foi gerada. A diferença no resultado da classificação usando essa versão, como se observa na Figura 5.38, é pequena. E embora a rede não tivesse convergido ao fim das 25 épocas, o gráfico demostra que a melhora obtida nas últimas épocas era irrisória, sugerindo que os resultados não alcançariam valores muito melhores em épocas seguintes.

Figura 5.38: Resultados da aplicação dos *embeddings* de DOS SANTOS et al. (2018) após estendê-lo com dados deste trabalho



A funcionalidade de extensão dos *embeddings* tem a limitação de que o vocabulário existente no modelo não é alterado, isto é, os novos termos presentes no *corpus* desse trabalho não eram adicionados no vetor expandido. Considerando essa limitação, foram criados novos *embeddings* usando apenas os textos do *corpus* deste trabalho.

Foram usados os textos gerados pelos vários pré-processadores já disponíveis e, além disso, nesse ponto foi criado o preprocessor5 combinado algumas tarefas que apresentavam bons resultados até aqui, mas mantendo os *tokens* próximos daqueles do preprocessor4.

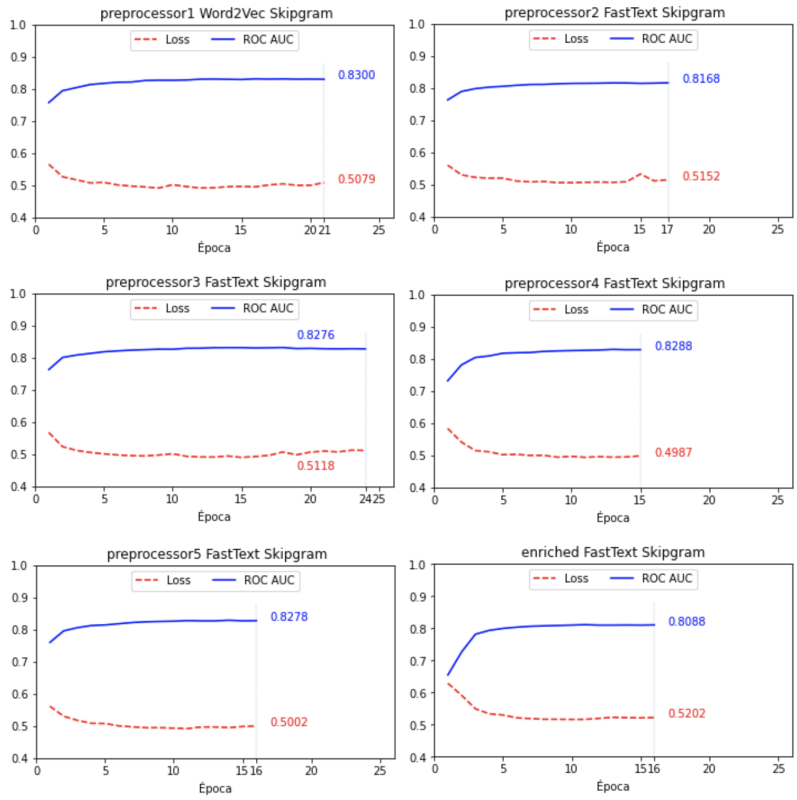
Para cada conjunto de textos pré-processados foram gerados 4 *embeddings*, criando as mesmas combinações dos dois modelos (Word2vec e FastText) e dos dois modos (CBOW e Skip-gram) criados por DOS SANTOS et al. (2018). Conforme apontado na Figura 5.7, todos os modelos foram gerados tendo 300 dimensões, uma dimensionalidade comumente encontrada na literatura, e uma janela de 5 *tokens*, o padrão da biblioteca utilizada, e incluindo todos os termos (*min_count:1*). Os melhores resultados obtidos para

cada configuração são apresentados na Figura 5.39.

Conforme se verifica, as variações dos resultados são pequenas nos valores da ROC AUC e Loss. A variação em termos de épocas para convergência, ou melhor, de pelo menos 5 épocas seguidas sem melhora, também é relativamente pequena. Em termos absolutos, o melhor resultado de ROC AUC é 0.8300 obtido pelo preprocessor1 (com Word2Vec em modo Skipgram). Considerando-se, entretanto, o tempo para atingir o resultado, preprocessor4 e preprocessor5, ambos em com algoritmo FastText em modo Skipgram podem ser considerados melhores já que obtém quase o mesmo resultado em menor tempo de treinamento (menos épocas).

Pela maneira como os experimentos foram conduzidos, os resultados do embedding gerado com o preprocessor5, que apresentava até então a melhor relação entre tempo de treinamento e valor de ROC AUC, foram conhecidos antes daqueles do preprocessor4. Por esse motivo, o modelo *preprocessor5 FastText Skipgram*, foi considerado o melhor e passou a ser utilizado para a otimização da rede.

Figura 5.39: Melhores resultados para os *embeddings* criados apenas com textos deste trabalho



Tendo identificado qual parecia ser o melhor *embedding* para alimentar a rede,

passou-se a avaliar diferentes possibilidades de meta-parâmetros para a sua configuração. Os testes a seguir reportam os resultados da rede com diferentes parâmetros, sempre usando o mesmo *embedding* (*preprocessor5 FastText Skipgram*) como entrada.

Seguindo sugestão de Bernico (2018), foram testados quatro diferentes tamanhos de lotes: 32, 64, 128 e 256 e três diferentes otimizadores: AdaDelta, Adam e RMSProp. Todas as possíveis combinações desses parâmetros foram avaliadas e um valor pseudo aleatório de *Dropout* foi usado ao testar cada uma delas. O espaço de busca para a opção de *Dropout* foi determinado pela função "*np.linspace(0.1, 0.5, 10)*" (que corresponde a: 0.1, 0.1444, 0.1889, 0.2333, 0.2778, 0.3222, 0.3667, 0.4111, 0.4556, 0.5) e em cada um dos testes que combinava um tamanho de lote e um otimizador, um desses possíveis valores foi selecionado aleatoriamente e aplicado na rede.

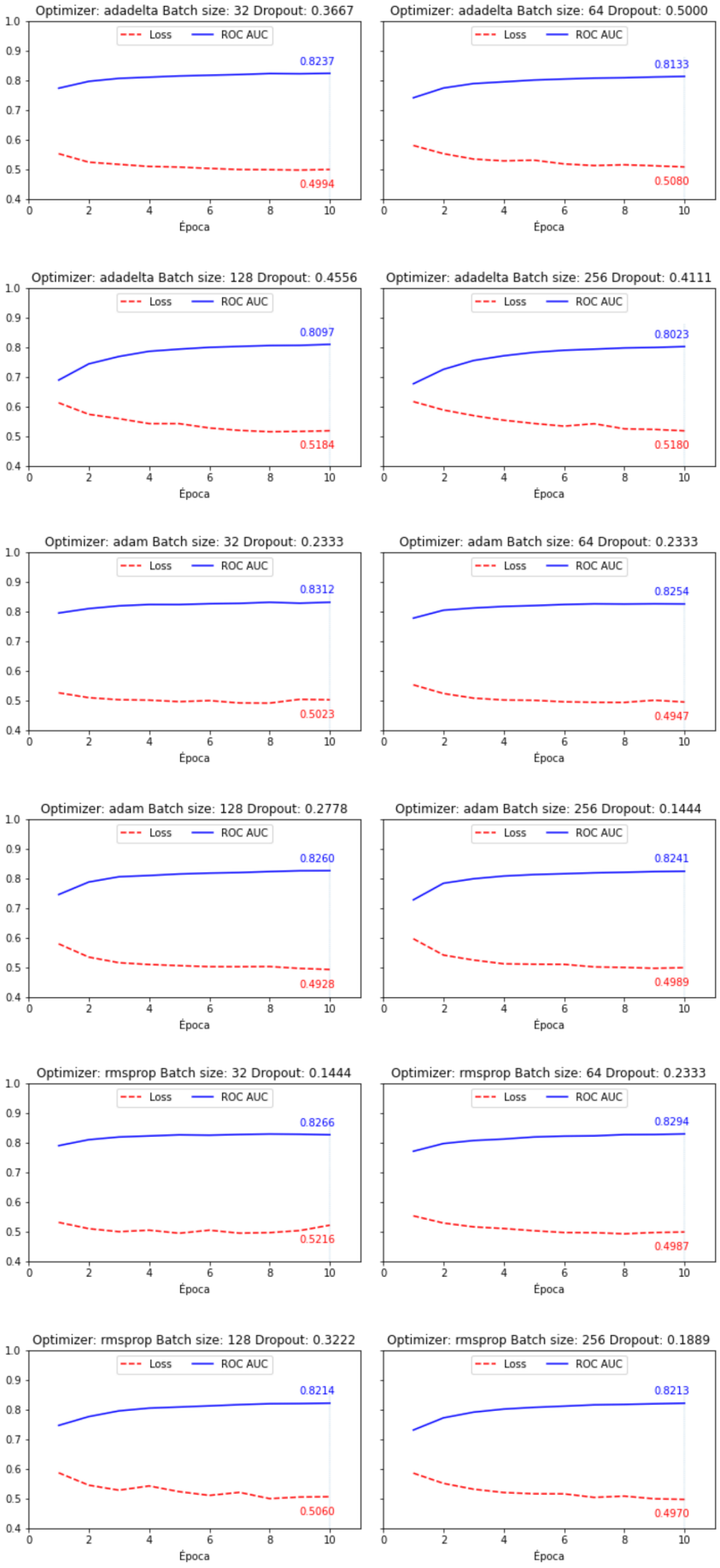
A partir destes testes, verificou-se que a configuração usando *Optimizer: adam Batch size: 32 Dropout: 0.2333* apresentava o melhor resultado em termos de ROC AUC (0.8312). Os detalhes constam na Figura 5.40.

Por fim, identificado esse melhor conjunto de parâmetros para a rede, passou-se a explorar diferentes topologias. A Figura a 5.41 mostra o resultado da ROC AUC para diferentes topologias usando o melhor *embedding* e os melhores parâmetros, recém identificados . A descrição de cada uma das topologias consta na Tabela 5.11

Após identificar a *topologia A* como melhor em uma execução limitada a 10 épocas, uma rede com esses mesmos parâmetros foi treinada, mas agora tendo como limite 100 épocas e uma tolerância de 25. Verifica-se que permitir que a rede fosse treinada por mais tempo não melhorou significativamente seu desemprenho. No novo treino a rede também deixou de melhorar após a época 11, sendo que o treinamento foi encerrado na época 36, após terem se passado 25 épocas sem melhora no acurácia da validação. A Figura 5.42 mostra os valores desse novo treinamento, destacando os valores obtidos nas épocas 11 e 36.

O que se observa desse novo treinamento, sobretudo analisando-se o gráfico comparativo entre treinamento e validação, é que a rede continua a melhorar no treinamento, reduzindo o valor de *Loss* a algo próximo de 0.4, porém ao validar a classificação esse valor volta a subir, ultrapassando 0.5. Disso pode-se deduzir que a rede está tendo um *overfit* e por isso, o treinamento mais longo não ajuda na melhora do valor da ROC AUC. Isso explica também porque topologias mais complexas, com mais neurônios e/ou mais camadas, tiveram desempenho pior do que a topologia A, uma das mais simples testadas, conforme e verifica pela Tabela 5.11.

Figura 5.40: Resultados da variação dos meta-parâmetros da rede



ID	Topologia							
Topologia A	[Input]	[Embedding]	[GRU (None, 32)]	[Output]				
Topologia B	[Input]	[Embedding]	[GRU (None, 64)]	[Dense (None, 64)]	[Dropout (None, 64)]	[Output]		
Topologia C	[Input]	[Embedding]	[GRU (None, 128)]	[Output]				
Topologia D	[Input]	[Embedding]	[GRU (None, 64)]	[GRU (None, 1)]	[Output]			
Topologia E	[Input]	[Embedding]	[Dense (None, 223, 64)]	[Dropout (None, 223, 64)]	[GRU (None, 223, 64)]	[GRU (None, 32)]	[Output]	
Topologia F	[Input]	[Embedding]	[GRU (None, 16)]	[Output]				
Topologia G	[Input]	[Embedding]	[GRU (None, 64)]	[Dense (None, 64)]	[RepeatVector (None, 6, 64)]	[Dense (None, 6, 64)]	[GRU (None, 64)]	[Output]
Topologia H	[Input]	[Embedding]	[GRU (None, 64)]	[Dense (None, 64)]	[RepeatVector (None, 6, 64)]	[Dense (None, 6, 64)]	[GRU (None, 64)]	[Output]
Topologia I	[Input]	[Embedding]	[GRU (None, 223, 64)	[GRU (None, 128)]	[Output]			
Topologia J	[Input]	[Embedding]	[GRU (None, 64)]	[Dense (None, 64)	[Dense (None, 32)]	[Dropout (None, 32)]	[Output]	
Topologia K	[Input]	[Embedding]	[GRU (None, 32)]	[Dense (None, 64)]	[Dropout (None, 64)]	[Output]		
Topologia L	[Input]	[Embedding]	[GRU (None, 16)]	[Output]				
Topologia M	[Input]	[Embedding]	[Dense (None, 223, 64)]	[Dropout (None, 223, 64)]	[Dense (None, 223, 64)]	[Dropout (None, 223, 64)]	[GRU (None, 64)]	[Output]
Topologia N	[Input]	[Embedding]	[GRU (None, 223, 8)]	[GRU (None, 8)]	[Output]			
Topologia O	[Input]	[Embedding]	[GRU (None, 128)]	[Output]				
Topologia P	[Input]	[Embedding]	[GRU (None, 8)]	[Output]				
Topologia Q	[Input]	[Embedding]	[GRU (None, 300)]	[Output]				
Topologia R	[Input]	[Embedding]	[Dense (None, 223, 128)]	[GRU (None, 64)]	[Dense (None, 32)]	[Dropout (None, 32)]	[Output]	
Topologia S	[Input]	[Embedding]	[Dense (None, 223, 128)]	[GRU (None, 64)]	[Dense (None, 32)]	[Dropout (None, 32)]	[Output]	
Topologia T	[Input]	[Embedding]	[LSTM (None, 128)]]	[Dropout (None, 128)	[Output]			
Topologia U	[Input]	[Embedding]	[GRU (None, 128)]	[Output]				

Tabela 5.11: Detalha as diferentes topologias testadas na rede neural

Figura 5.41: Resultados da variação da Topologia da Rede

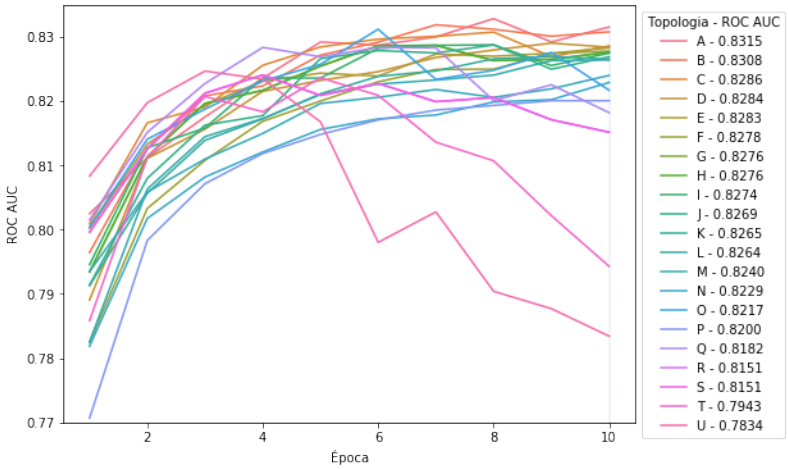
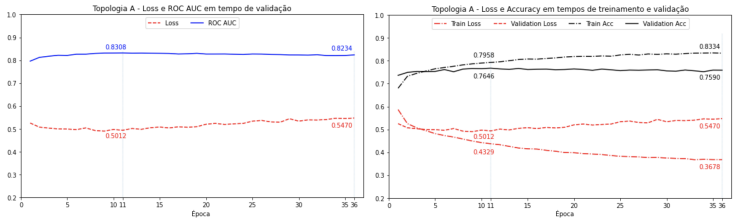


Figura 5.42: Resultados do novo treinamento da melhor topologia (Topologia A)



Tomando-se a versão da rede com melhor desempenho, isto é, com os pesos obtidos na época 11, aplicando-a aos dados de *holdout* obteve-se resultados bastante próximos daqueles apresentados em treinamento e teste, mantendo uma ROC AUC de 0.83, conforme se observa na Figura 5.43.

Figura 5.43: Resultados do melhor classificador do Fluxo 7 aplicado aos dados de *Holdout*

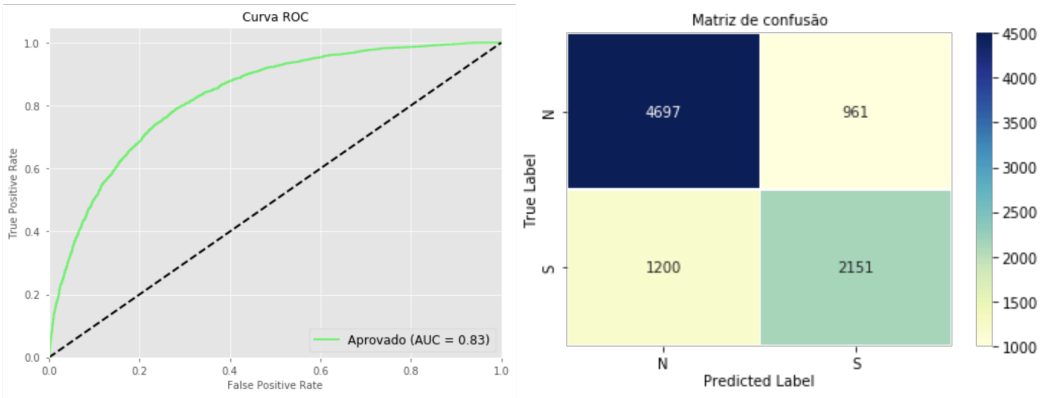
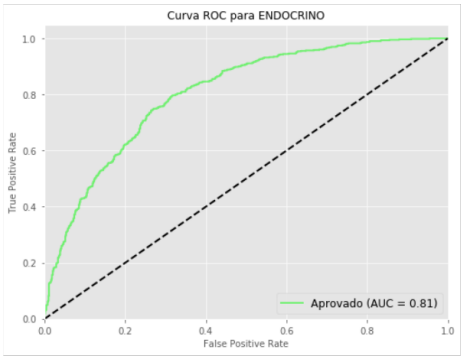
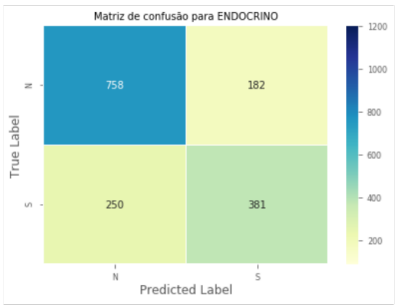
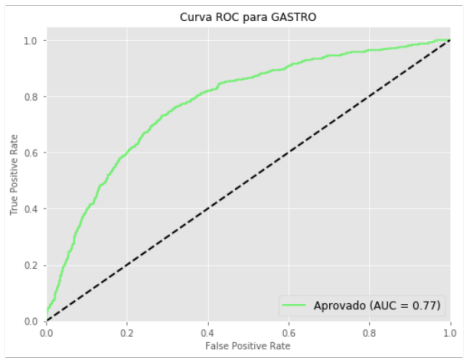
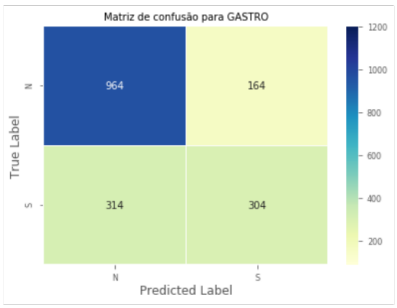


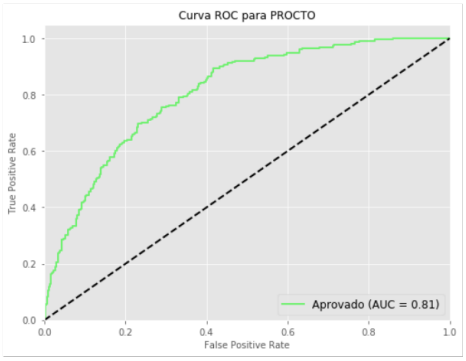
Figura 5.44: Resultados do melhor classificador do Fluxo 7, por especialidade (Holdout)

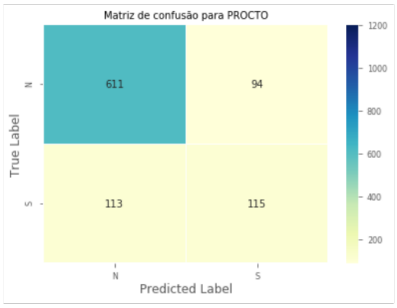


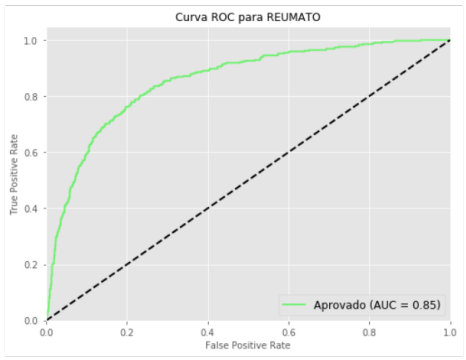


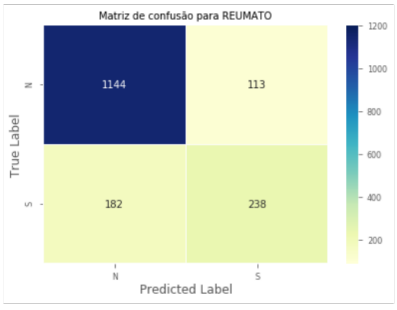


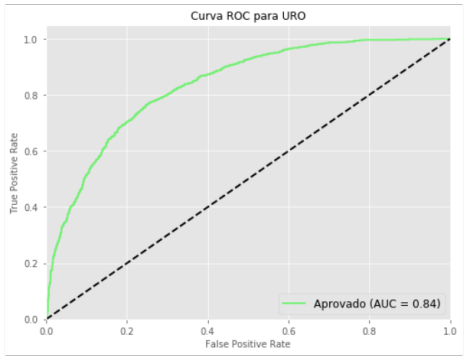


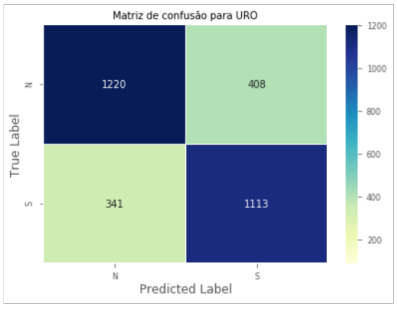












5.9 Resumo do Capítulo

Ao longo deste capítulo foram apresentados diversos fluxos de pré-processamento, engenharia de *features* e configuração de modelos de classificação que foram aplicados ao *corpus* disponível, a fim de encontrar o melhor configuração para a tarefa de classificação de textos apresentada.

Inicialmente, sob orientação de especialistas do domínio, os dados foram limpos e agrupados em 5 especialidades, em seguida um subconjunto estratificado de 20% deles foi criado para validação dos modelos (*holdout*).

Diferentes combinações de tarefas de pré-processamento foram realizadas, variando de alternativas simples como a conversão para letras minúsculas e a remoção de caracteres especiais, até outras complexas como a aplicação de *POS-Tagging* e o enriquecimento do texto com dados de uma base controlada, o UMLS. Para realização desta última etapa, após avaliar algumas das soluções disponíveis, desenvolveu-se um método próprio de consulta à base do UMLS que permitiu extrair de lá os Identificadores Únicos de Conceitos (CUI), sinônimos e tipos semânticos relacionados aos *tokens* encontrados no *corpus* deste trabalho e estendê-lo com estas novas informações semanticamente relacionadas.

Após o pré-processamento, foram criadas representações do *corpus* no formato *Bag-of-Words*, *Bag-of-ngrams* e *Word Embeddings* que foram em seguida submetidas a diferentes técnicas de seleção e transformação de *features*, criando representações simplificadas dos vetores a serem submetidas aos classificadores. Após os primeiros testes e o estabelecimento de um *baseline*, diferentes técnicas de *resampling* foram avaliadas, visando melhorar os resultados da classificação.

Vários modelos de classificação foram experimentados, reportando-se aqui os resultados de cinco deles. As comparações foram feitas com base no resultado da ROC AUC para as classificações, na maioria dos casos, considerando a sua média num procedimento de validação cruzada com 10 *folds*.

Entre o Fluxo 1 e o Fluxo 2, onde a diferença principal foi a inclusão da etapa de *resampling*, tomando-se a configuração que ficou com o melhor resultado no segundo fluxo (*enriched*, TFiDF, sublinear_tf:true) e observando-se seu valor no fluxo anterior percebe-se uma melhora na ROC AUC de cerca de 8%, com o valor subindo de $0,8166$ para $0,8783$. O que indica que a aplicação do *resampling* é relevante para a melhora dos resultados com o classificador SVM Linear.

Observando-se os melhores resultados dentro do Fluxo 2 vê-se pouca variação entre os valores obtidos com cada pré-processador. Entretanto, é interessante observar que há um padrão seguido pelos três pré-processadores que não incluem o enriquecimento dos textos e outro padrão para os dois que incluem esta etapa. No caso daqueles, sempre que usada a técnica TFiDF o melhor foi realizá-la usando os parâmetros *default* da implementação da Scikit Learn. Quando usada a técnica de *Word Count*, foi sempre melhor valer-se da criação de n-gramas de 2 e 3 tokens. Essa opção com n-gramas, no entanto, tem um custo computacional maior sendo, portanto, preferível utilizar o TFiDF com parâmetros *default*.

Já para os dois pré-processadores em que houve enriquecimento dos textos, vê-se que não utilizar os n-gramas funcionou melhor com a técnica de *Word Count*. Isso pode se dever ao fato de os textos terem ficado muito maiores com o enriquecimento, levando a um conjunto muito grande de *features* insignificantes quando criados os n-gramas. Já com a técnica TFiDF, para os textos enriquecidos, utilizar o sublinear_tf:true teve melhor resultado. Como os valores são muito próximos aos dos textos não enriquecidos, considerando-se o custo extra para o enriquecimento e as questões de licenciamento do UMLS, conclui-se que o melhor custo benefício é obtido sem enriquecimento dos textos.

No Fluxo 3, que avaliou as diferentes técnicas de seleção e transformação de *features*, os resultados demonstram que só se obtém uma pequena melhora, de aproximadamente $0,03$, na ROC AUC média, quando comparada aos resultados já obtidos. Ainda assim, pelos resultados do Fluxo 3, conclui-se que a técnica *Select K-Best* com limite de 2000 *features* e métrica *Chi-square* é a que atinge melhor resultado nos dados disponíveis.

No Fluxo 4, uma variação que levou em conta não só os dados textuais em si como também um conjunto de dados estruturados obtidos a partir desses textos, no caso, a contagem de tipos semânticos do UMLS presentes em cada exemplo, foi submetida a dois diferentes classificadores: *Decision Trees* e *AdaBoost*. Os resultados de ROC AUC foram inferiores aos obtidos até então com o SVM Linear.

No Fluxo 5 os dados que haviam levado ao melhor resultado de classificação com o SVM Linear foram submetidos novamente ao algoritmo *AdaBoost*. Dessa vez, comparou-se a classificação desse algoritmo antes e depois do *resampling*, sobretudo com a intenção de destacar a variância adicionada aos resultados após o *resampling*. Com os resultados desse fluxo foi possível observar que a média se eleva, porém há grandes discrepâncias nos valores de ROC AUC em cada *fold*.

No Fluxo 6, os mesmos dados da melhor classificação com o SVM Linear foram

submetidos a um *Ensemble Random Forest*. Verifica-se variância entre os *folds* semelhante àquela apresentada no fluxo anterior, fruto da aplicação o *oversample*, porém com média de ROC AUC de 0.81, portanto, próxima aos 0.83 obtidos com o SVM. Entretanto, ao aplicar o melhor classificador do Fluxo 6 sobre os dados de *holdout*, esta média caiu para 0.64.

No Fluxo 7 foram testados diferentes opções de *Word Embeddings*, submetidos a uma rede neural recorrente. Foram avaliados *embeddings* já existentes e também versões estendidas deles e novos embeddings criados do zero usando o *corpus* deste trabalho. Tendo-se obtido valores de ROC AUC bastante próximos com todos esses conjuntos, adotou-se o *embedding* criado unicamente com o *corpus* deste trabalho, identificado na Figura 5.39 como *preprocessor5 FastText Skipgram*.

Definido o *embedding*, buscou a melhor configuração para a rede neural, testando-se quatro diferentes tamanhos de lote e três opções de otimizadores, com valores de *dropout* aleatórios acompanhado as combinações desses parâmetros. Obteve-se os melhores resultados usando Adam como otimizador, tamanho de lote igual a 32 e *dropout* igual a 0.2333. Com esses parâmetros, vinte e uma topologias alternativas foram testadas para a rede, prevalecendo uma das mais simples.Essa configuração obteve ROC AUC de 0.83, superior a encontrada até então, e esse resultado se manteve nos testes com os dados de *holdout*.

Ao fim dos experimentos, concluiu-se que a melhor configuração dentre as avaliadas foi a do Fluxo 7, que utiliza uma rede neural recorrente composta de uma camada de *Embedding*, seguida por uma GRU de 32 neurônios e uma camada densa na saída, usando a função sigmoide. A camada de *Embedding* é construída com o vetor criado a partir do *corpus* deste trabalho, após um pré-processamento que incluiu conversão para letras minúsculas, remoção números, caracteres especiais, *stopwords*, acentos e *tokens* com apenas um caractere e a aplicação de *stemming*. Esse *embedding* usou o algoritmo FastText em modo SkipGram, tem 300 dimensões, e seu treinamento considerou uma janela de 5 *tokens* e incluiu todos os termos que - após o pré-processamento - apareciam ao menos uma vez. Os resultados desta configuração, mostrados nas Figuras 5.43 e 5.44 atingem ROC AUC de 0.83 para o conjunto de *holdout*. Quando observados por especialidade os resultados são de 0.81 para Endócrino, 0.77 para Gastro, $0,81$ para Procto, 0.85 para Reumato e 0.84 para Uro.

6 AVALIAÇÃO DA EFICÁCIA DO ALGORITMO

Conforme previsto no Capítulo 4, nesta etapa do trabalho o algoritmo com melhor resultado nos experimentos é comparado ao método atual, sendo aplicado a um novo conjunto de dados revisado por reguladores experientes.

Para esta etapa, dois profissionais com larga experiência na regulação de casos do TelessaúdeRS estão se dedicando a uma revisão de um conjunto de 350 casos de regulação para cada uma das cinco filas abordadas neste trabalho. Estão sendo usadas solicitações recebidas dentro do Município de Porto Alegre entre nov/2018 e nov/2019.

A resposta considerada como sendo a do método atual é aquela que consta atualmente no sistema para esses casos. A resposta do método experimental é a classificação dada pelo melhor algoritmo deste trabalho. A resposta correta é aquela indicada pelos dois reguladores, após a revisão dos casos.

A revisão dos casos está sendo conduzida da seguinte forma: Sem conhecer a resposta atual dada ao caso, cada um dos reguladores, individualmente, analisa a solicitação de encaminhamento e a regula, aprovando-a ou não. Quando ambos os reguladores dão à solicitação a mesma resposta, essa é assumida imediatamente como a resposta correta. Quando há divergência entre as respostas destes dois regulares, os profissionais discutem o caso entre si e chegam a um acordo a respeito da resposta. A partir dessa criteriosa revisão, estabelece-se um novo "padrão ouro" contra o qual podemos avaliar o nosso método e o método atual (forma padrão de regulação) e comparar seus resultados.

A criação deste novo padrão ouro ainda está em desenvolvimento. Já foram revisados casos para três das cinco filas abordadas neste trabalho, foram elas: Uro, Reumato e Procto. Para Procto, 1 dos 100 casos para os quais houve divergência ainda não foi discutido e será, portanto, desconsiderado na comparação aqui apresentada. Assim, a avaliação aqui apresentada utilizou 1049 exemplos (350 de Uro, 350 de Reumato e 349 de Procto).

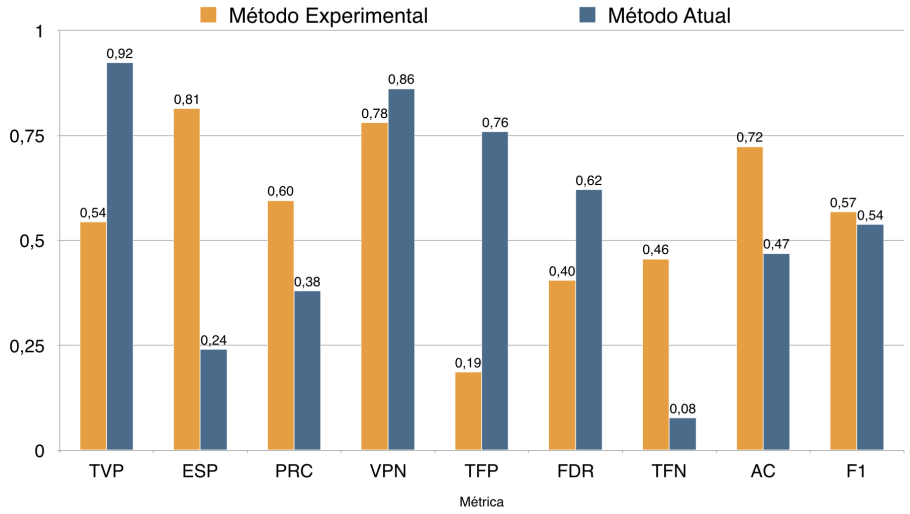
6.1 Resultados

As matrizes de confusão resultantes das classificações usando o método experimental, isto é, o melhor algoritmo deste trabalho e o método atual de regulação, que é a avaliação por médicos Reguladores, constam como apêndice na Figura A.1. Dessas matrizes de confusão foram extraídas as medidas apresentadas na Tabela 6.1 (cujas métricas foram introduzidas na Lista 2.10.1).

Tabela 6.1: Medidas de comparação entre classificações tendo como métodos o Melhor Algoritmo Experimental (Exp.) e a classificação por reguladores (Atual) para as filas de Procto, Reumato e Uro na Fase 3

	As 3 filas		REUMATO		URO		PROCTO		
Método	**Exp.**	**Atual**	**Exp.**	**Atual**	**Exp.**	**Atual**	**Exp.**	**Atual**	**Fórmula**
VP - Verdadeiros Positivos	191	324	61	80	78	118	52	126	
FP - Falsos Positivos	130	530	42	222	54	147	34	161	
VN - Verdadeiros Negativos	568	168	223	43	171	78	174	47	
FN - Falsos Negativos	160	27	24	5	47	7	89	15	
TVP - Sensibilidade (ou revocação)	0,54	0,92	0,72	0,94	0,62	0,94	0,37	0,89	TVP = VP / (VP + FN)
ESP - Especificidade (1 - TFP)	0,81	0,24	0,84	0,16	0,76	0,35	0,84	0,23	ESP = VN / (FP + VN)
PRC - Precisão	0,60	0,38	0,59	0,26	0,59	0,45	0,60	0,44	PRC = VP / (VP + FP)
VPN - Valor preditivo negativo	0,78	0,86	0,90	0,90	0,78	0,92	0,66	0,76	VPN = VN / (VN + FN)
TFP - Taxa de erro na classe negativa	0,19	0,76	0,16	0,84	0,24	0,65	0,16	0,77	TFP = FP / (FP + VN)
FDR - Taxa de descobertas falsas	0,40	0,62	0,41	0,74	0,41	0,55	0,40	0,56	FDR = FP / (VP + FP)
TFN - Taxa de erro na classe positiva	0,46	0,08	0,28	0,06	0,38	0,06	0,63	0,11	TFN = FN / (FN + VP)
AC - Acurácia	0,72	0,47	0,81	0,35	0,71	0,56	0,65	0,50	AC = (VP + VN) / (P + N)
F1-Measure - Medida F1	0,57	0,54	0,65	0,41	0,61	0,61	0,46	0,59	F1 = 2 (PRC * TVP) /(PRC + TVP)

Figura 6.1: Medidas de comparação entre classificações do método Experimental e Atual para as três filas na Fase 3



Como não se pode determinar limiares de confiança para as classificações do método atual, não há como apresentar uma comparação dos métodos em função da ROC AUC, como foi feito ao longo deste trabalho. Ainda assim, é possível observar diversas medidas objetivas na tabela 6.1 que permitem discutir as comparações.

De maneira geral, verifica-se uma Sensibilidade muito menor do método experimental em relação ao método atual. Consideradas as três filas juntas, esse valor cai de $0,92$ com o método atual para $0,54$ com o experimental. A Sensibilidade aqui equivale a taxa de detecção de encaminhamentos adequados, ou seja, a capacidade de reconhecer corretamente a necessidade de avaliação pelo especialista. Uma baixa sensibilidade, equivale a dizer que temos muitos Falsos Negativos, ou seja, muitos dos pacientes que deveriam ter tido seu encaminhamento aprovado, não o tiveram quando a avaliação foi feita usando o método experimental.

Essa baixa sensibilidade é contraposta por uma precisão maior. Quando considera-

das as três filas, esse valor sobe dos $0,38$ do método atual para $0,60$ no experimental. Uma precisão maior significa que a maioria dos pacientes cujo encaminhamento foi aprovado pelo algoritmo aqui proposto realmente precisavam da consulta com o especialista. Na prática, isso se traduziria em um menor número de pacientes se deslocando desnecessariamente à capital e menos recursos da Atenção Especializada alocados desnecessariamente. Essa ideia fica clara ao se observar a baixa taxa de Falsos Positivos do algoritmo.

Essa compensação nas medidas dá ao método experimental uma Medida F1 ligeiramente melhor no quadro geral, $0,57$ *versus* os $0,54$ do método atual e, conforme observado nos demais resultados apresentados ao longo desse trabalho, vemos que isso não é consistente entre todas as filas, sendo pior que o método atual para Procto, melhor para Reumato e igual para Uro.

6.2 Entregável

Ao final desta fase foi criado um repositório[1] contendo um componente de software que encapsula o melhor algoritmo experimental, permitindo o reuso da solução. O modelo em si, bem como os *embeddings* produzidos no trabalho, não estão registrados no repositório por questões de sigilo dos dados. Esses artefatos foram compartilhados diretamente com a equipe do TelessaúdeRS.

O componente desenvolvido permite que o classificador seja utilizado como um *Web service*, através de requisições HTTP. Isso facilita a integração com outros sistemas experimentais ou mesmo com o próprio sistema Gercon e também permite que *interfaces* para usuários finais sejam construídas para uso da solução.

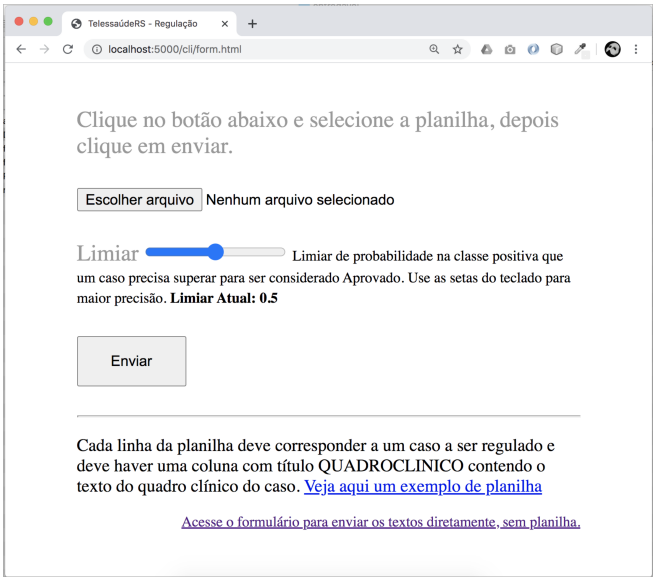
O serviço deve usado via solicitações HTTP em uma URL específica. Um parâmetro opcional pode ser adicionado na URL da requisição para determinar o limiar acima do qual os exemplos serão classificados na classe positiva. Os dados podem ser enviados de duas formas: como *JSON* ou em uma planilha em formato Microsoft Excel.

No repositório também foram disponibilizadas duas interfaces simples que servem de exemplo de como o serviço pode ser usado. Elas são apresentadas nas figuras abaixo.

Para utilizar este método deve ser feita uma requisição HTTP POST para o endereço do serviço. A requisição deve conter um arquivo em um campo nomeado como *spreadsheet*. O arquivo deve ser uma planilha contendo uma coluna intitulada *QUADRO-CLINICO*.

[1] https://github.com/pitervergara/telessauders-regulacao/

Figura 6.2: Exemplo de cliente para uso do software enviando dados em uma planilha



Cada linha da planilha deve corresponder a um caso a ser regulado e a coluna QUADROCLINICO deve conter o texto do quadro clínico do caso. As demais colunas que eventualmente existam serão ignoradas para fins de classificação. O retorno será uma cópia da mesma planilha com as seguintes três novas colunas: *_APROVADO*: Conterá 0 (Não-Aprovado) ou 1 (Aprovado); *_PROBA_0:* Probabilidade do caso pertencer à classe Não-Aprovado e *_PROBA_1:* Probabilidade do caso pertencer à classe Aprovado.

Para utilizar o serviço enviando os dados como JSON, uma requisição deve ser feita para o mesmo e endereço do serviço, mas contendo no corpo da requisição um atributo *solicitacoes* com lista dos textos de quadro clínico de cada caso a ser regulado. A resposta retornada será também um objeto JSON contendo dois atributos. Um deles, nomeado *pred_bins*, conterá uma lista com valores 0 ou 1 representando, respectivamente, a Não-autorização ou a Autorização do caso. O outro, nomeado *pred_probas*, conterá uma lista de listas, em que cada uma das sublistas contém dois valores que representam, respectivamente, a probabilidade do caso ser considerado Não-aprovado e a probabilidade do ser considerado Aprovado.

6.3 Resumo do Capítulo

Este capítulo apresentou uma comparação entre os resultados de classificação do melhor algoritmo deste trabalho e do método atual de regulação. Observando-se a diagonal principal das matrizes de confusão, vemos que o algoritmo aqui proposto tem um

Figura 6.3: Exemplo de cliente para uso do *software* enviando dados como *JSON*

conjunto maior de acertos em todos os casos, porém isso deve-se sobretudo a maior quantidade de acertos na classe negativa, que é a classe predominante. Os resultados indicam sobretudo uma taxa bastante maior de Falsos Negativos do algoritmo quando configurado no limiar padrão.

Cabe relembrar, entretanto, uma característica importante do método experimental, que é a possibilidade de calibrar o algoritmo de acordo com o uso pretendido. Os números aqui apresentados - bem como todos os demais reportados no trabalho - sempre levam em consideração um limiar de $0,5$ para que um exemplo seja classificado como positivo. Isso é, o algoritmo só classifica um exemplo como positivo se a probabilidade de pertencer a essa classe for superior a 50%. Esse valor, contudo, pode ser configurado. Por exemplo, adotando-se um limiar de $0,3$, a sensibilidade de algoritmo para os dados apresentados nesse capítulo subiria para $0,78$. Alternativamente, o limiar poderia ser configurado para $0,8$ o que aumentaria a especificidade para 0.98. Diversos possíveis valores já foram testados e estão reportados na Tabela B.1 para que possam ser considerados pelos especialistas do domínio no eventual uso da solução.

Para facilitar o uso do algoritmo, também foi apresentado nesse capítulo o componente de software final, o *entregável* deste trabalho, que permite o reúso do algoritmo tanto para eventuais classificações em ambiente de produção quanto para a continuidade da pesquisa.

7 CONCLUSÃO

Neste trabalho foi posta a avaliação de diferentes alternativas para o desenvolvimento de um Classificador de Textos baseado em Aprendizado de Máquina capaz de auxiliar no processo de regulação dos encaminhados de pacientes para a Atenção Especializada, realizado pelo TelessaúdeRS.

Foram avaliados e reportados os resultados de inúmeras combinações de pré-processamento, engenharia de *features* e seleção de modelos de classificação e ao final desse processo o melhor algoritmo encontrado apresentou uma ROC AUC de 0.83, tanto em tempo de treinamento quanto nos dados de *holdout*. Este algoritmo foi comparado ao método atual e verificou-se que em diversos aspectos ele se aproxima dos resultados obtidos atualmente, o que pode ser resumido pela Medida F1 de $0,58$ do algoritmo experimental, ligeiramente superior aos $0,53$ do método atual.

Mencionou-se no Capítulo 2, citando Faceli (2011), que, para ser aceitável, a acurácia preditiva de um classificador para um conjunto de dados desbalanceados deve ser maior do que a acurácia obtida atribuindo-se todo novo objeto à classe majoritária. Esse é o caso para o algoritmo resultante deste trabalho, já que, por exemplo, se nos dados apresentados no Capítulo 4.3 simplesmente atribuíssemos 0 a todos os exemplos, a acurácia seria de 0.63 e algoritmo atinge 0.73.

Além disso, tendo em vista a possibilidade de calibrar o algoritmo em tempo de uso, através do *entregável* fornecido, entende-se que ele pode auxiliar na agilização do processo de regulação dos encaminhamentos do TelessaúdeRS, por exemplo, aumentando-se bastante o limiar e autorizando automaticamente os casos que o algoritmo indicar como aprovados, já que o algoritmo tem - mesmo com a configuração padrão - uma baixa taxa de Falsos Positivos. Essa avaliação, naturalmente, caberá aos profissionais do TelessaúdeRS que ainda irão conduzir a conclusão da Fase3 e a avaliação final do algoritmo.

Pelo exposto, considera-se que os resultados deste trabalho têm como principais contribuições apresentar um extenso comparativo entre os resultados de diferentes tarefas de pré-processamento e engenharia de *features* para os dados de telerregulação do TelessaúdeRS, estabelecer uma *baseline* de classificação automática desses encaminhamentos, com o qual novos trabalhos poderão ser comparados, e oferecer uma versão funcional de software que poderá ser integrada desde já ao processo de regulação.

Não obstante esses resultados positivos, podemos indicar entre as limitações dso trabalho o fato de que a maioria de nossas tarefas de engenharia de *features* são funda-

mentalmente estatísticas, em contraste com as diversas opções de PNL encontradas na literatura, as quais extraem maior semântica dos dados. De igual forma, a falta de relação clara entre as decisões do algoritmo e os protocolos de regulação que norteiam a decisão dos Reguladores são um fator negativo, já que a explicabilidade das Redes Neurais ainda é um campo de pesquisa em aberto. Também pesa o fato de que foram explorados os métodos que atualmente são o estado da arte na área de Processamento de Linguagem Natural, como os métodos de *Transformers* e outros baseados em mecanismos de atenção, e ainda o fato de que não se explorou o impacto da adição de *features* não textuais - como idade e sexo do paciente - que também estavam disponíveis, ainda que esta tenha sido uma decisão de projeto, visando manter o foco nos dados textuais.

Dessas conclusões, podemos identificar como oportunidades de trabalhos futuros justamente a exploração de novas *features* mais voltadas à semântica do domínio da saúde, a busca de maior explicabilidade das decisões, a avaliação de métodos mais modernos de classificação e a integração do classificador resultante no processo de regulação do TelessaúdeRS.

REFERÊNCIAS

AGGARWAL, C. C. **Neural Networks and Deep Learning**. [S.l.: s.n.], 2018. ISBN 9783319944623.

ALLAHYARI, M. et al. A brief survey of text mining: Classification, clustering and extraction techniques. **CoRR**, abs/1707.02919, 2017. Available from Internet: <http://arxiv.org/abs/1707.02919>.

BENGIO, Y. et al. A neural probabilistic language model. **J. Mach. Learn. Res.**, JMLR.org, v. 3, n. null, p. 1137–1155, mar. 2003. ISSN 1532-4435.

BERNICO, M. **Deep Learning Quick Reference: Useful hacks for training and optimizing deep neural networks with TensorFlow and Keras**. Packt Publishing, 2018. ISBN 9781788838917. Available from Internet: <https://books.google.com.br/books?id=M5RRDwAAQBAJ>.

BIRD, S.; KLEIN, E.; LOPER, E. Natural language processing with python. In: ____. 1st. ed. [S.l.]: O'Reilly Media, Inc., 2009. chp. 6. ISBN 0596516495, 9780596516499.

BOJANOWSKI, P. et al. Enriching word vectors with subword information. **CoRR**, abs/1607.04606, 2016. Available from Internet: <http://arxiv.org/abs/1607.04606>.

BREIMAN, L. Bagging predictors. **Machine Learning**, v. 24, n. 2, p. 123–140, 1996. ISSN 08856125.

CHAWLA, N. et al. Smote: Synthetic minority over-sampling technique. **J. Artif. Intell. Res. (JAIR)**, v. 16, p. 321–357, 06 2002.

CHUNG, J. et al. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. p. 1–9, 2014. Available from Internet: <http://arxiv.org/abs/1412.3555>.

DAMASCENO, F. R. et al. Um estudo sobre o emprego da mineração textual para classificação de teleconsultorias no contexto do Projeto Telessaúde-RS. **Revista Eletrônica de Comunicação, Informação e Inovação em Saúde**, v. 10, n. 2, 2016. ISSN 1981-6278. Available from Internet: <https://www.reciis.icict.fiocruz.br/index.php/reciis/article/view/972/pdf_972>.

DEMNER-FUSHMAN, D.; CHAPMAN, W. W.; MCDONALD, C. J. **What can natural language processing do for clinical decision support?** 2009.

DOS SANTOS, H. D. P. et al. An Initial Investigation of the Charlson Comorbidity Index Regression Based on Clinical Notes. **Proceedings - IEEE Symposium on Computer-Based Medical Systems**, v. 2018-June, p. 6–11, 2018. ISSN 10637125.

DOUZAS, G.; BACAO, F.; LAST, F. Improving imbalanced learning through a heuristic oversampling method based on k-means and smote. **Information Sciences**, Elsevier BV, v. 465, p. 1–20, Oct 2018. ISSN 0020-0255. Available from Internet: <http://dx.doi.org/10.1016/j.ins.2018.06.056>.

FACEBOOK. **O que é a configuração de reconhecimento facial no Facebook e como ela funciona?** 2020. Available from Internet: <https://www.facebook.com/help/122175507864081?helpref=faq_content>.

FACELI, K. **Inteligência Artificial. Uma Abordagem de Aprendizado de Máquina (Em Portuguese do Brasil)**. [S.l.]: LTC, 2011. ISBN 8521618808.

GOOGLE. **Machine Learning Crash Course**. 2020. Available from Internet: <https://developers.google.com/machine-learning/crash-course>.

GUYON, I.; ELISSEEFF, A. **An Introduction to Feature Extraction**. [S.l.], 2006. Available from Internet: <http://clopinet.com/fextract-book/IntroFS.pdf>.

HAN, H.; WANG, W.-Y.; MAO, B.-H. Borderline-smote: A new over-sampling method in imbalanced data sets learning. In: HUANG, D.-S.; ZHANG, X.-P.; HUANG, G.-B. (Ed.). **Advances in Intelligent Computing**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. p. 878–887. ISBN 978-3-540-31902-3.

HINTON, G. **Lecture 6a - Overview of mini-batch gradient descent** . 2012. Available from Internet: <http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf>.

HOGARTH, M. Informática médica: Um pouco de história. **Informática Médica**, v. 1, n. 5, 1998.

HOYT, R. **Medical informatics : practical guide for the healthcare professional**. Pensacola, Fla. Raleigh, N.C: University of West Florida, School of Allied Health and Life Sciences, Medical Informatics Program Lulu.com, 2009. ISBN 978-0-557-13323-9.

HSU, C.-W.; CHANG, C.-C.; LIN, C.-J. **A Practical Guide to Support Vector Classification**. Department of Computer Science and Information Engineering, National Taiwan University, 2003. Available from Internet: <http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf>.

HUNT, D. L. et al. Effects of Computer-Based Clinical Decision Support Systems on Physician Performance and Patient OutcomesA Systematic Review. **JAMA**, v. 280, n. 15, p. 1339–1346, 10 1998. ISSN 0098-7484. Available from Internet: <https://doi.org/10.1001/jama.280.15.1339>.

KINGMA, D. P.; BA, J. L. Adam: A method for stochastic optimization. **3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings**, p. 1–15, 2015.

KREIMEYER, K. et al. Natural language processing systems for capturing and standardizing unstructured clinical information: A systematic review. **Journal of Biomedical Informatics**, Academic Press, v. 73, p. 14–29, sep 2017. ISSN 1532-0464. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S1532046417301685?via\%3Dihub>.

LABS chartbeat. **textacy: higher-level NLP built on spaCy**. 2016. Available from Internet: <http://textacy.readthedocs.io/en/latest/index.html>.

LANDAUER, T. et al. **Handbook of Latent Semantic Analysis**. Taylor & Francis, 2007. ISBN 9781135603274. Available from Internet: <https://books.google.pt/books?id=Jm_NgzzDntYC>.

LEMAÎTRE, G.; NOGUEIRA, F.; ARIDAS, C. K. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. **Journal of Machine Learning Research**, v. 18, n. 17, p. 1–5, 2017. Available from Internet: <http://jmlr.org/papers/v18/16-365.html>.

LENZ, A. L. **Development and Comparison of Machine Learning Methods for Subjective Refraction Prediction**. Bachelor's Thesis, 2018.

LOPER, E.; BIRD, S. Nltk: The natural language toolkit. In: **Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1**. Stroudsburg, PA, USA: Association for Computational Linguistics, 2002. (ETMTNLP '02), p. 63–70. Available from Internet: <https://doi.org/10.3115/1118108.1118117>.

MANNING, C. D.; RAGHAVAN, P.; SCHÜTZE, H. Introduction to information retrieval. In: ____. New York, NY, USA: Cambridge University Press, 2008. chp. 13, p. 253–286. ISBN 0521865719, 9780521865715.

MEDICINE, U. N. L. of. **MetamorphoSys - The UMLS Installation and Customization Program**. Acesso em: 23 de outubro de 2019. Available from Internet: <https://www.ncbi.nlm.nih.gov/books/NBK9683/>.

METZ, C. E. Basic principles of ROC analysis. **Seminars in Nuclear Medicine**, v. 8, n. 4, p. 283–298, 1978. ISSN 00012998.

MIKOLOV, T. et al. Efficient estimation of word representations in vector space. **Proceedings of Workshop at ICLR**, v. 2013, 01 2013.

MITCHELL, T. M. **Machine Learning**. New York: McGraw-Hill, 1997. ISBN 978-0-07-042807-2.

MONARD, M. C.; BATISTA, G. E. A. P. A. Learning with skewed class distributions. **Advances in Logic, Artificial Intelligence and Robotics**, p. 173–180, 2002.

MUJTABA, G. et al. Prediction of cause of death from forensic autopsy reports using text classification techniques: A comparative study. **Journal of Forensic and Legal Medicine**, Elsevier Ltd, v. 57, p. 41–50, 2018. ISSN 18787487. Available from Internet: <https://doi.org/10.1016/j.jflm.2017.07.001>.

NG, A. **Gradient Descent in Practice I - Feature Scaling**. 2020. Available from Internet: <https://www.coursera.org/lecture/machine-learning/gradient-descent-in-practice-i-feature-scaling-xx3Da>.

OLAH, C. **Understanding LSTM Networks**. 2015. Available from Internet: <http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf>.

OLEYNIK, M. et al. Performance analysis of a POS tagger applied to discharge summaries in portuguese. **Studies in Health Technology and Informatics**, v. 160, n. PART 1, p. 959–963, 2010. ISSN 09269630.

OLIVEIRA, L. E. S. E. et al. A rule-based method for continuity of care identification in discharge summaries. **Studies in Health Technology and Informatics**, v. 192, n. 1-2, p. 1221, 2013. ISSN 09269630.

ORENGO, V. M.; HUYCK, C. A Stemming Algorithm for Portuguese Language. In: **Proc. of Eigth Symposium on String Processing and Information Retrieval (SPIRE 2001) - Chile**. [S.l.: s.n.], 2001. p. 186–193.

PINHO, J. M. B. M. da C. **O que é a Informática Médica?** 2002. Available from Internet: <https://users.med.up.pt/~jbizarro/Intromed/Trabalho/IntroMed1_ficheiros/Introducao.htm>.

RAVI, J.; SRINIVASAN, V. **Introduction to ML Classification Models using scikit-learn**. 2018. https://www.udemy.com/introduction-to-ml-classification-models-using-scikit-learn/learn/v4/overview. Available from Internet: <https://www.udemy.com/introduction-to-ml-classification-models-using-scikit-learn/learn/v4/overview>.

ROY, B. **All about Feature Scaling**. 2020. Available from Internet: <https://towardsdatascience.com/all-about-feature-scaling-bcc0ad75cb35>.

SANTOS JOAQUIM; VIEIRA, R. **Evolução dos Modelos de Linguagem**. 2020. https://propor.di.uevora.pt/tutorials/. Available from Internet: <https://propor.di.uevora.pt/wp-content/uploads/2020/03/PROPOR_Modelos_de_linguagem.pdf.pdf>.

SARKER, A.; GONZALEZ, G. Portable automatic text classification for adverse drug reaction detection via multi-corpus training. **Journal of Biomedical Informatics**, v. 53, 2015. Available from Internet: <http://dx.doi.org/10.1016/j.jbi.2014.11.002>.

SAVOVA, G. K. et al. Mayo clinical Text Analysis and Knowledge Extraction System (cTAKES): architecture, component evaluation and applications. **Journal of the American Medical Informatics Association : JAMIA**, BMJ Group, v. 17, n. 5, p. 507–513, 2010. ISSN 1527-974X. Available from Internet: <https://www.ncbi.nlm.nih.gov/pubmed/20819853https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2995668/>.

SCIKIT-LEARN developers. **Documentation of scikit-learn 0.19.1**. 2017. http://scikit-learn.org/stable/documentation.html. Available from Internet: <http://scikit-learn.org/stable/documentation.html>.

SKIENA, S. S. **The data science design manual**. [s.n.], 2017. 1–445 p. ISBN 978-3-319-55443-3. Available from Internet: <http://link.springer.com/10.1007/978-3-319-55444-0>.

SRIVIDHYA, V.; ANITHA, R. Evaluating preprocessing techniques in text categorization. In: **International Journal of Computer Science and Application**. [S.l.]: Publishing Press, 2010. p. 49–51.

SUS. **Programa Telessaúde Brasil Redes**. 2020. Available from Internet: <https://www.saude.gov.br/telessaude>.

TELESSAUDERS. **Como Funciona o RegulaSUS**. 2019. Available from Internet: <https://www.ufrgs.br/telessauders/documentos/infografico_gercon.pdf>.

TELESSAUDERS. **Quem somos**. 2020. Available from Internet: <https://www.ufrgs.br/telessauders/quemsomos/>.

UZUNER, O. Recognizing obesity and comorbidities in sparse data. **Journal of the American Medical Informatics Association : JAMIA**, American Medical Informatics Association, v. 16, n. 4, p. 561–70, 2009. ISSN 1067-5027. Available from Internet: <http://www.ncbi.nlm.nih.gov/pubmed/19390096http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC2705260>.

VASCONCELOS SIMONE; CONCI, A. **Análise de Componentes Principais (PCA)**. 2020. Available from Internet: <http://www2.ic.uff.br/~aconci/PCA-ACP.pdf>.

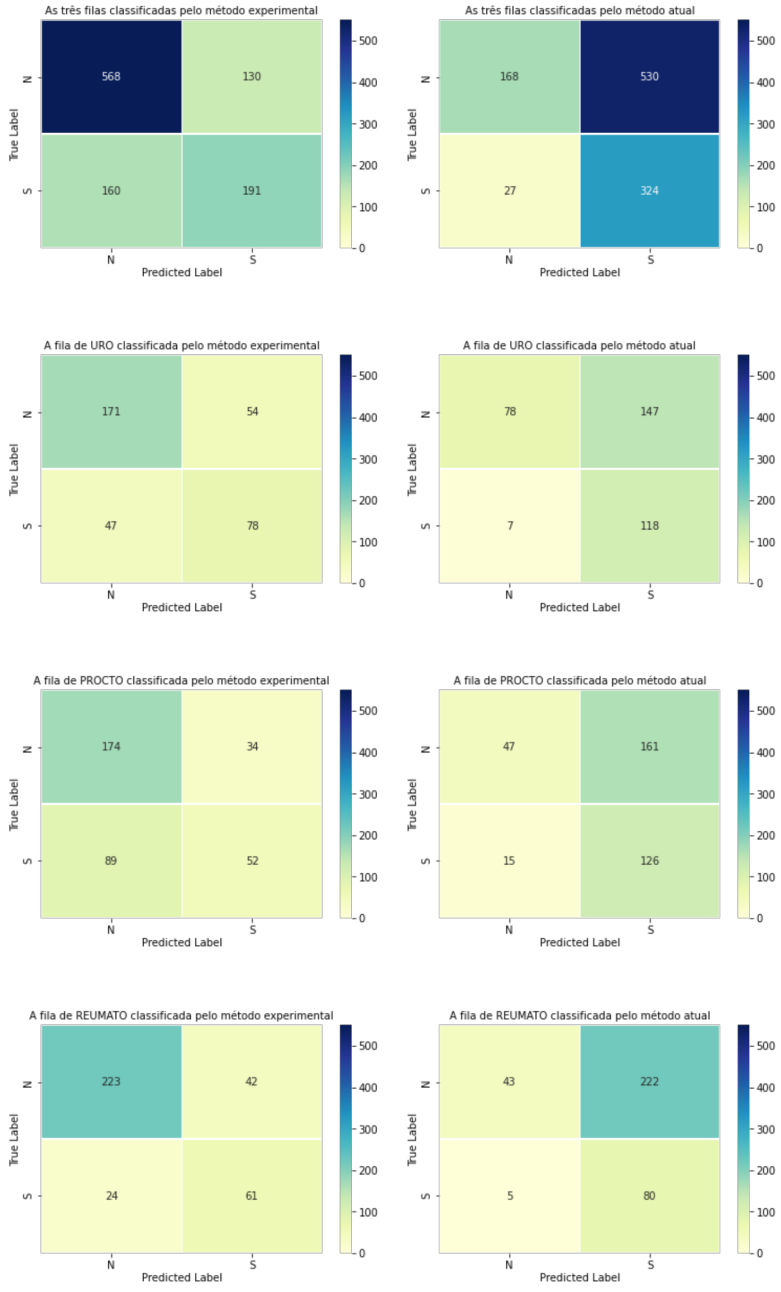
WÄHNER, K.; PENCHIKALA, S. **Data Preprocessing vs. Data Wrangling in Machine Learning Projects**. 2017. https://www.infoq.com/articles/ml-data-processing. Available from Internet: <https://www.infoq.com/articles/ml-data-processing>.

WEN, A. **Keeping your company data safe with new security updates to Gmail**. 2020. Available from Internet: <https://blog.google/products/g-suite/keeping-your-company-data-safe-new-security-updates-gmail/>.

ZEILER, M. D. ADADELTA: An Adaptive Learning Rate Method. 2012. Available from Internet: <http://arxiv.org/abs/1212.5701>.

ApêndiceA FASE 3 MÉTODOS EXPERIMENTAL *VS* MÉTODO ATUAL

Figura A.1: Matrizes de confusão comparando o melhor algoritmo de regulação experimental ao método atual de regulação para os dados da Fase 3



ApêndiceB MÉTRICAS DO ALGORITMO EM DIFERENTES LIMIARES

Tabela B.1: Métricas para a classificação dada pelo algoritmo para os 1049 casos utilizados na fase 3 (Capítulo 6) ao configurar o algoritmo com diferentes limiares.

Limiar	VP	FP	FN	VN	TVP	ESP	PRC	VPN	TFP	FDR	TFN	AC	F1
0.0	351	698	0	0	1,00	0,00	0,33	0,00	1,00	0,67	0,00	0,33	0,50
0.01	351	680	0	18	1,00	0,03	0,34	1,00	0,97	0,66	0,00	0,35	0,51
0.02	349	645	2	53	0,99	0,08	0,35	0,96	0,92	0,65	0,01	0,38	0,52
0.03	349	618	2	80	0,99	0,11	0,36	0,98	0,89	0,64	0,01	0,41	0,53
0.04	348	592	3	106	0,99	0,15	0,37	0,97	0,85	0,63	0,01	0,43	0,54
0.05	346	560	5	138	0,99	0,20	0,38	0,97	0,80	0,62	0,01	0,46	0,55
0.06	345	540	6	158	0,98	0,23	0,39	0,96	0,77	0,61	0,02	0,48	0,56
0.07	341	522	10	176	0,97	0,25	0,40	0,95	0,75	0,60	0,03	0,49	0,56
0.08	340	503	11	195	0,97	0,28	0,40	0,95	0,72	0,60	0,03	0,51	0,57
0.09	338	482	13	216	0,96	0,31	0,41	0,94	0,69	0,59	0,04	0,53	0,58
0.1	334	468	17	230	0,95	0,33	0,42	0,93	0,67	0,58	0,05	0,54	0,58
0.11	331	455	20	243	0,94	0,35	0,42	0,92	0,65	0,58	0,06	0,55	0,58
0.12	329	436	22	262	0,94	0,38	0,43	0,92	0,62	0,57	0,06	0,56	0,59
0.13	324	424	27	274	0,92	0,39	0,43	0,91	0,61	0,57	0,08	0,57	0,59
0.14	322	403	29	295	0,92	0,42	0,44	0,91	0,58	0,56	0,08	0,59	0,60
0.15	316	393	35	305	0,90	0,44	0,45	0,90	0,56	0,55	0,10	0,59	0,60
0.16	315	384	36	314	0,90	0,45	0,45	0,90	0,55	0,55	0,10	0,60	0,60
0.17	312	375	39	323	0,89	0,46	0,45	0,89	0,54	0,55	0,11	0,61	0,60
0.18	310	364	41	334	0,88	0,48	0,46	0,89	0,52	0,54	0,12	0,61	0,60
0.19	307	354	44	344	0,87	0,49	0,46	0,89	0,51	0,54	0,13	0,62	0,61
0.2	304	346	47	352	0,87	0,50	0,47	0,88	0,50	0,53	0,13	0,63	0,61
0.21	301	338	50	360	0,86	0,52	0,47	0,88	0,48	0,53	0,14	0,63	0,61
0.22	299	325	52	373	0,85	0,53	0,48	0,88	0,47	0,52	0,15	0,64	0,61
0.23	295	319	56	379	0,84	0,54	0,48	0,87	0,46	0,52	0,16	0,64	0,61
0.24	292	310	59	388	0,83	0,56	0,49	0,87	0,44	0,51	0,17	0,65	0,61
0.25	287	297	64	401	0,82	0,57	0,49	0,86	0,43	0,51	0,18	0,66	0,61
0.26	282	289	69	409	0,80	0,59	0,49	0,86	0,41	0,51	0,20	0,66	0,61
0.27	277	283	74	415	0,79	0,59	0,49	0,85	0,41	0,51	0,21	0,66	0,61
0.28	274	277	77	421	0,78	0,60	0,50	0,85	0,40	0,50	0,22	0,66	0,61
0.29	274	266	77	432	0,78	0,62	0,51	0,85	0,38	0,49	0,22	0,67	0,62
0.3	273	256	78	442	0,78	0,63	0,52	0,85	0,37	0,48	0,22	0,68	0,62
0.31	270	247	81	451	0,77	0,65	0,52	0,85	0,35	0,48	0,23	0,69	0,62
0.32	265	241	86	457	0,75	0,65	0,52	0,84	0,35	0,48	0,25	0,69	0,62
0.33	261	228	90	470	0,74	0,67	0,53	0,84	0,33	0,47	0,26	0,70	0,62
0.34	258	222	93	476	0,74	0,68	0,54	0,84	0,32	0,46	0,26	0,70	0,62
0.35	255	217	96	481	0,73	0,69	0,54	0,83	0,31	0,46	0,27	0,70	0,62
0.36	248	209	103	489	0,71	0,70	0,54	0,83	0,30	0,46	0,29	0,70	0,61
0.37	242	202	109	496	0,69	0,71	0,55	0,82	0,29	0,45	0,31	0,70	0,61

0.38	236	193	115	505	0,67	0,72	0,55	0,81	0,28	0,45	0,33	0,71	0,61
0.39	235	188	116	510	0,67	0,73	0,56	0,81	0,27	0,44	0,33	0,71	0,61
0.4	231	186	120	512	0,66	0,73	0,55	0,81	0,27	0,45	0,34	0,71	0,60
0.41	227	180	124	518	0,65	0,74	0,56	0,81	0,26	0,44	0,35	0,71	0,60
0.42	224	176	127	522	0,64	0,75	0,56	0,80	0,25	0,44	0,36	0,71	0,60
0.43	222	168	129	530	0,63	0,76	0,57	0,80	0,24	0,43	0,37	0,72	0,60
0.44	219	162	132	536	0,62	0,77	0,57	0,80	0,23	0,43	0,38	0,72	0,60
0.45	216	160	135	538	0,62	0,77	0,57	0,80	0,23	0,43	0,38	0,72	0,59
0.46	213	151	138	547	0,61	0,78	0,59	0,80	0,22	0,41	0,39	0,72	0,60
0.47	208	147	143	551	0,59	0,79	0,59	0,79	0,21	0,41	0,41	0,72	0,59
0.48	201	139	150	559	0,57	0,80	0,59	0,79	0,20	0,41	0,43	0,72	0,58
0.49	197	136	154	562	0,56	0,81	0,59	0,78	0,19	0,41	0,44	0,72	0,58
0.5	191	130	160	568	0,54	0,81	0,60	0,78	0,19	0,40	0,46	0,72	0,57
0.51	190	123	161	575	0,54	0,82	0,61	0,78	0,18	0,39	0,46	0,73	0,57
0.52	184	119	167	579	0,52	0,83	0,61	0,78	0,17	0,39	0,48	0,73	0,56
0.53	178	115	173	583	0,51	0,84	0,61	0,77	0,16	0,39	0,49	0,73	0,55
0.54	174	106	177	592	0,50	0,85	0,62	0,77	0,15	0,38	0,50	0,73	0,55
0.55	172	101	179	597	0,49	0,86	0,63	0,77	0,14	0,37	0,51	0,73	0,55
0.56	170	99	181	599	0,48	0,86	0,63	0,77	0,14	0,37	0,52	0,73	0,55
0.57	165	96	186	602	0,47	0,86	0,63	0,76	0,14	0,37	0,53	0,73	0,54
0.58	165	92	186	606	0,47	0,87	0,64	0,77	0,13	0,36	0,53	0,73	0,54
0.59	160	88	191	610	0,46	0,87	0,65	0,76	0,13	0,35	0,54	0,73	0,53
0.6	154	84	197	614	0,44	0,88	0,65	0,76	0,12	0,35	0,56	0,73	0,52
0.61	151	82	200	616	0,43	0,88	0,65	0,75	0,12	0,35	0,57	0,73	0,52
0.62	147	76	204	622	0,42	0,89	0,66	0,75	0,11	0,34	0,58	0,73	0,51
0.63	147	72	204	626	0,42	0,90	0,67	0,75	0,10	0,33	0,58	0,74	0,52
0.64	141	68	210	630	0,40	0,90	0,67	0,75	0,10	0,33	0,60	0,73	0,50
0.65	137	65	214	633	0,39	0,91	0,68	0,75	0,09	0,32	0,61	0,73	0,50
0.66	131	59	220	639	0,37	0,92	0,69	0,74	0,08	0,31	0,63	0,73	0,48
0.67	129	59	222	639	0,37	0,92	0,69	0,74	0,08	0,31	0,63	0,73	0,48
0.68	128	57	223	641	0,36	0,92	0,69	0,74	0,08	0,31	0,64	0,73	0,48
0.69	125	51	226	647	0,36	0,93	0,71	0,74	0,07	0,29	0,64	0,74	0,47
0.7	120	48	231	650	0,34	0,93	0,71	0,74	0,07	0,29	0,66	0,73	0,46
0.71	115	45	236	653	0,33	0,94	0,72	0,73	0,06	0,28	0,67	0,73	0,45
0.72	112	41	239	657	0,32	0,94	0,73	0,73	0,06	0,27	0,68	0,73	0,44
0.73	107	38	244	660	0,30	0,95	0,74	0,73	0,05	0,26	0,70	0,73	0,43
0.74	97	32	254	666	0,28	0,95	0,75	0,72	0,05	0,25	0,72	0,73	0,40
0.75	90	27	261	671	0,26	0,96	0,77	0,72	0,04	0,23	0,74	0,73	0,38
0.76	87	23	264	675	0,25	0,97	0,79	0,72	0,03	0,21	0,75	0,73	0,38
0.77	83	17	268	681	0,24	0,98	0,83	0,72	0,02	0,17	0,76	0,73	0,37
0.78	73	16	278	682	0,21	0,98	0,82	0,71	0,02	0,18	0,79	0,72	0,33
0.79	69	16	282	682	0,20	0,98	0,81	0,71	0,02	0,19	0,80	0,72	0,32
0.8	69	15	282	683	0,20	0,98	0,82	0,71	0,02	0,18	0,80	0,72	0,32

0.81	65	13	286	685	0,19	0,98	0,83	0,71	0,02	0,17	0,81	0,71	0,30
0.82	59	12	292	686	0,17	0,98	0,83	0,70	0,02	0,17	0,83	0,71	0,28
0.83	52	8	299	690	0,15	0,99	0,87	0,70	0,01	0,13	0,85	0,71	0,25
0.84	48	8	303	690	0,14	0,99	0,86	0,69	0,01	0,14	0,86	0,70	0,24
0.85	43	6	308	692	0,12	0,99	0,88	0,69	0,01	0,12	0,88	0,70	0,22
0.86	37	6	314	692	0,11	0,99	0,86	0,69	0,01	0,14	0,89	0,69	0,19
0.87	34	5	317	693	0,10	0,99	0,87	0,69	0,01	0,13	0,90	0,69	0,17
0.88	30	4	321	694	0,09	0,99	0,88	0,68	0,01	0,12	0,91	0,69	0,16
0.89	26	3	325	695	0,07	1,00	0,90	0,68	0,00	0,10	0,93	0,69	0,14
0.9	21	2	330	696	0,06	1,00	0,91	0,68	0,00	0,09	0,94	0,68	0,11
0.91	14	2	337	696	0,04	1,00	0,88	0,67	0,00	0,13	0,96	0,68	0,08
0.92	10	2	341	696	0,03	1,00	0,83	0,67	0,00	0,17	0,97	0,67	0,06
0.93	7	2	344	696	0,02	1,00	0,78	0,67	0,00	0,22	0,98	0,67	0,04
0.94	7	2	344	696	0,02	1,00	0,78	0,67	0,00	0,22	0,98	0,67	0,04
0.95	5	2	346	696	0,01	1,00	0,71	0,67	0,00	0,29	0,99	0,67	0,03
0.96	3	1	348	697	0,01	1,00	0,75	0,67	0,00	0,25	0,99	0,67	0,02
0.97	1	0	350	698	0,00	1,00	1,00	0,67	0,00	0,00	1,00	0,67	0,01
0.98	0	0	351	698	0,00	1,00	0,00	0,67	0,00	0,00	1,00	0,67	0,00
0.99	0	0	351	698	0,00	1,00	0,00	0,67	0,00	0,00	1,00	0,67	0,00
1.0	0	0	351	698	0,00	1,00	0,00	0,67	0,00	0,00	1,00	0,67	0,00