UNIVERSIDADE DE CAXIAS DO SUL
ÁREA DO CONHECIMENTO DE CIÊNCIAS EXATAS E ENGENHARIAS

RAFAEL BELLINI

APLICAÇÃO DE MÁQUINAS DE SUPORTE VETORIAL NA CLASSIFICAÇÃO TEXTUAL

CAXIAS DO SUL

2020

RAFAEL BELLINI

APLICAÇÃO DE MÁQUINAS DE SUPORTE VETORIAL NA CLASSIFICAÇÃO TEXTUAL

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Ciência da Computação na Área do Conhecimento de Ciências Exatas e Engenharias da Universidade de Caxias do Sul.

Orientador: Profa. Dra. Carine Geltrudes Webber

CAXIAS DO SUL

2020

RAFAEL BELLINI

APLICAÇÃO DE MÁQUINAS DE SUPORTE VETORIAL NA CLASSIFICAÇÃO TEXTUAL

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Ciência da Computação na Área do Conhecimento de Ciências Exatas e Engenharias da Universidade de Caxias do Sul.

Aprovado em 01/12/2020

BANCA EXAMINADORA

Profa. Dra. Carine Geltrudes Webber
Universidade de Caxias do Sul - UCS

Profa. Dra. Elisa Boff
Universidade de Caxias do Sul - UCS

Profa. Dra. Maria de Fátima Webber do Prado Lima
Universidade de Caxias do Sul - UCS

RESUMO

A área de classificação de dados textuais envolve a coleta, o processamento, a análise e a construção de modelos para classificar textos. Ela se vale de algoritmos de *machine learning*. Dentre eles, destaca-se o algoritmo de Máquinas de Suporte Vetorial (SVM, do inglês, *Support Vector Machines*), amplamente utilizado com conjuntos de dados textuais. O objetivo deste trabalho é aplicar o SVM na construção de um modelo de classificação textual, usando como exemplo textos da área da saúde. As amostras textuais utilizadas neste trabalho foram coletadas por uma equipe de médicos especialistas da Universidade de Caxias do Sul e encontram-se em linguagem natural, na língua portuguesa brasileira, tendo sido previamente rotuladas em termos de suas características, tais como: descrição do tratamento, benefícios do tratamento, consequências do tratamento, influência na qualidade de vida do paciente e riscos do tratamento. Os textos de cada uma dessas categorias foram previamente classificados como positivos, negativos e regulares, configurando um problema do tipo multiclasse. A fase de pré-processamento dos textos foi realizada utilizando a biblioteca chamada *Natural Language Toolkit* (NLTK), já para os testes do algoritmo SVM utilizou-se a biblioteca chamada *Scikit-learn* e para o balanceamento das classes foi utilizado o algoritmo *Synthetic Minority Over-sampling Technique* (SMOTE), da biblioteca *imblearn*. Também foi utilizada a ferramenta Anaconda para Windows, que possibilitou executar tanto a linguagem *Python* quanto aplicativo, como o *Jupyter Notebook*. Os resultados obtidos através dos testes revelaram respostas satisfatórias para demonstrar a possibilidade de classificação supervisionada para os dados textuais das diversas categorias mencionadas, tendo apresentado resultados superiores a 90.0% de acurácia. Um dos desafios encontrados foi o desbalanceamento das classes, que necessitou de estudo e uso de métodos apropriados a fim de que se pudesse obter resultados satisfatórios.

Palavras-chaves: Aprendizagem de Máquina. Classificação. Máquinas de Suporte Vetorial. *Scikit-learn*. Supervisionada.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AC	Aprovados Corretamente
ADASYN	*Adaptive Synthetic*
AI	Aprovados Incorretamente
AM	Aprendizagem de Máquina
ARFF	*Attribute-Relation File Format*
CSV	*Comma-Separated Values*
FP	Falsos Positivos
FN	Falsos Negativos
GUI	*Graphical User Interface*
IDE	*Integrated Development Environment*
IDFT	*Inverse Document Frequency Transform*
JSON	*JavaScript Object Notation*
LibSVM	*Library for Support Vector Machines*
MVC	*Model-View-Controller*
NDF	*Normalization Form Canonical Decomposition*
NLTK	*Natural Language Toolkit*
RC	Reprovados Corretamente
RI	Reprovados Incorretamente
RNA	Redes Neurais Artificiais
RSLP	Removedor de Sufixos da Língua Portuguesa
SMOTE	*Synthetic Minority Over-sampling Technique*
SVM	*Support Vector Machines*
SRM	*Structural Risk Minimization*
TFT	*Term Frequency Transform*

VC	*Vapnik-Chervonenkis*
VP	Verdadeiros Positivos
VN	Verdadeiros Negativos
WEKA	*Waikato Environment for Knowledge Analisys*

SUMÁRIO

1 INTRODUÇÃO

A era da informação é caracterizada pela crescente expansão no volume de dados gerados e armazenados, conforme caracterizado por (HAND, 1997). Atualmente já existe um número considerável de documentos digitais, e esse é um número que só tende a aumentar. Tais documentos representam uma quantidade massiva de informações disponíveis e o desafio está em buscar valor neste amplo universo.

Todas as áreas do conhecimento podem se beneficiar pelo tratamento automático de textos. Esta tarefa inclui o processamento para fins de classificação automática de textos, área tratada pelos algoritmos de aprendizagem de máquina. Em especial, a área da saúde tem estado atenta aos avanços e tecnologias que apoiam tais processos.

A classificação automática de textos pode ser considerada uma aplicação útil para que as pessoas, sejam pacientes ou profissionais da área da saúde, possam contar com métodos automáticos de tratamento textual. Os textos da área da saúde são produzidos e consumidos em grande volume. Garantir a qualidade tanto da produção, quanto do que está sendo lindo, é relevante e pode ser feito por meio de métodos computacionais.

No campo de pesquisa da Inteligência Artificial, denomina-se de Aprendizagem de Máquina (AM, do inglês, *Machine Learning*), a área em que estuda-se o desenvolvimento de métodos automáticos capazes de extrair conhecimento a partir de amostras de dados (MITCHELL, 1997). Em geral, os diversos algoritmos de AM são utilizados de forma a gerar classificadores para um conjunto de exemplos.

A classificação automática de textos é o processo em que documentos construídos em linguagem natural são submetidos a categorias predefinidas considerando-se o seu conteúdo. Para essa finalidade existem alguns algoritmos úteis, tais como Árvore de Decisão, Redes Neurais e Máquinas de Suporte Vetorial (SVMs).

O SVM é um algoritmo introduzido por Cortes e Vapnik (1995) como uma nova técnica para resolver problemas de reconhecimento de padrões. Segundo a Teoria das SVMs (CORTES; VAPNIK, 1995), as técnicas tradicionais de reconhecimento de padrões são baseadas na minimização do risco empírico — isto é, na tentativa de otimizar o desempenho do conjunto de treino. Já as SVMs minimizam o risco estrutural — isto é, a probabilidade de classificar mal padrões de dados desconhecidos segundo uma distribuição de probabilidade. As SVMs se tornam atrativas devido à sua capacidade de condensar a informação contida no conjunto de treino e ao uso de famílias de decisão de relativa baixa dimensão (PONTIL; VERRY, 1998).

Algumas das principais características das SVMs que tornam seu uso atrativo são: a obtenção de boa capacidade de generalização de um classificador; a robustez diante de objetos de grandes dimensões, como imagens; a otimização de uma função quadrática da função objetiva

e a base teórica bem definida dentro da matemática e estatística.

Este algoritmo é amplamente utilizado no reconhecimento de padrões, como reconhecimento facial (KOJI, 2014) e reconhecimento de caracteres escritos à mão (UJJWAL; SK; SWAPAN, 2011). Na área da saúde já foi aplicado em diversas ocasiões, como na detecção do câncer de pulmão (KURESHI; ABIDI; BLOUIN, 2014) e mesmo na oftalmologia, para prever a qualidade de vida relacionada à visão a partir de dados de acuidade e campo visual em pacientes com glaucoma (HIRASAWA *et al.*, 2014). Esse tipo de abordagem também já foi utilizado para o diagnóstico de ceratocone. Arbelaez (AL., 2015) utilizou dados de tomografia corneana para construir um modelo de SVM e obteve sensibilidade de 95% para detecção de ceratocone e especificidade de 97,2%.

Em outro exemplo, na assistência médica, durante o processo de atendimento, são produzidos e armazenados dados para fins administrativos ou para verificação científica. Esses dados podem ser recuperados e reutilizados através de técnicas de mineração de textos, com o objetivo de encontrar um novo sentido para estas informações, possibilitando um aumento de conhecimento e entendimento das necessidades biológicas, bioquímicas, patológicas, psicossociais e ambientais, processos pelos quais a saúde e a doença são mediadas.

1.1 PROBLEMA E QUESTÃO DE PESQUISA

A tarefa de aprendizado realizada pelos classificadores de texto representa um novo conjunto de desafios para área de AM. Entre as principais dificuldades relacionadas à mineração de textos está a enorme quantidade de documentos disponíveis. Além disso, o habitual paradigma de programação baseado em lógica encontra grandes dificuldades no reconhecimento das relações difusas e muitas vezes ambíguas dos documentos de texto. Outra dificuldade é que a mineração de textos visa revelar as informações ocultas através de métodos capazes de lidar com um grande número de estruturas de palavras em linguagem natural, conciliado com a capacidade de lidar com a imprecisão, a incerteza e a imperfeição (HOTHO; NÜRBBERGER; PAASS, 2005).

A questão de pesquisa para este projeto é a seguinte: a utilização de um paradigma estatístico, aplicável por meio da técnica das SVMs, é capaz de produzir um modelo de classificação de textos da área da saúde?

1.2 OBJETIVOS

O objetivo deste trabalho é analisar como as SVMs comportam-se na classificação de textos, tendo como amostra constituída exemplos textuais da área da saúde.

Para chegar a tal objetivo, os seguintes objetivos específicos foram definidos:

1. identificar os principais parâmetros do algoritmo de classificação baseado em SVM;

2. aplicar e testar a técnica estudada em um conjunto de amostras de textos previamente classificados por especialistas;

3. analisar os resultados obtidos, listando as vantagens e limitações do método de classificação textual usando SVMs.

1.3 ESTRUTURA DO TRABALHO

O presente trabalho está estruturado de acordo com a seguinte descrição: no Capítulo 2 são apresentados os conceitos básicos sobre AM e SVMs, necessários para definir os conceitos que nortearam o trabalho. No Capítulo 3 é escolhida a plataforma de Aprendizagem de Máquina, a seleção do método e métricas e a seleção dos textos. No Capítulo 4 são apresentadas a carga dos textos, a preparação dos textos, os testes do algoritmo SVM, a variação dos parâmetros, a tabulação dos resultados, a análise dos resultados com identificação do melhor caso e as justificativas da escolha. Por fim, no Capítulo 5 são apresentadas a síntese do trabalho, as contribuições do trabalho e os trabalhos futuros.

2 MÁQUINAS DE SUPORTE VETORIAL

A Aprendizagem de Máquina é uma área da Inteligência Artificial que busca desenvolver modelos computacionais de aprendizagem. Ela se materializa em programas de computador que melhoram seu desempenho em determinadas tarefas através da experiência (MITCHELL, 1997).

Os modelos computacionais de aprendizagem tratam de soluções para problemas que utilizam o princípio de inferência lógica, chamado de "indução", e possibilitam alcançar conclusões genéricas a respeito de um conjunto particular de exemplos (FACELI *et al.*, 2011). Na indução, um conceito é aprendido realizando-se inferências indutivas a respeito dos exemplos fornecidos. Dessa forma, obtêm-se hipóteses, que podem ser satisfatórias ou não. As hipóteses satisfatórias são aquelas que possuem uma boa capacidade de generalização. O aprendizado indutivo se subdivide em não supervisionado e supervisionado.

No aprendizado não supervisionado, os exemplos não são rotulados. O algoritmo de AM aprende a agrupar as entradas segundo uma medida de qualidade. Neste caso, o indutor tenta determinar se os exemplos podem ser agrupados de alguma maneira, formando agrupamentos ou *clusters*. Após essa etapa, é realizada uma análise para cada agrupamento, para determinar seu significado no contexto do problema.

Já o aprendizado supervisionado possui um algoritmo de aprendizagem ou indutor, que tem por objetivo retirar um bom classificador de um conjunto de exemplos rotulados. Através desse classificador é possível identificar novos exemplos que ainda não tenham sido rotulados. Para as classes que possuem rótulos com valores discretos $1, \ldots, k$, o problema é chamado de "classificação"; para as que possuem rótulos com valores contínuos, ele é chamado de "regressão". Um problema de classificação com $k = 2$ denomina-se binário. Se $k > 2$, tem-se um problema multiclasse.

Cada exemplo, também chamado de "dado" ou "caso", é representado por um vetor de características, onde cada característica, também chamada de "atributo", expressa um determinado aspecto do exemplo. Existem dois tipos de atributos: nominal e contínuos. Um atributo é considerado nominal ou categórico quando não há uma ordem entre os valores que ele pode assumir (por exemplo, cores). No caso de atributos contínuos, existe uma ordem linear para os valores assumidos (por exemplo, entre pesos $\in \mathbb{R}$). (REZENDE, 2003)

Neste trabalho foi utilizado o aprendizado supervisionado, com a utilização das Máquinas de Suporte Vetorial — que representam uma abordagem eficiente para resolver problemas de classificação. Esta técnica foi recentemente desenvolvida por (VAPNIK, 1999) como um novo método de Aprendizagem de Máquina com base na Teoria da Aprendizagem Estatística fundamentada no princípio da minimização de risco estrutural (do inglês, *structural risk minimization*

— *SRM*). O SVM deriva da técnica de aprendizado por RNA [1] (Redes Neurais Artificiais), na sua forma computacional não algorítmica, para obtenção do conhecimento.

As Máquinas de Suporte Vetorial utilizam como base a Teoria do Aprendizado Estatístico, detalhada na Seção 2.1. Posteriormente são apresentados: os conceitos básicos para treinamento das SVMs, na Seção 2.2; hiperplano ótimo para padrões separáveis, na Seção 2.3; hiperplano ótimo para padrões não separáveis, na Seção 2.4 e classificadores não lineares, na Seção 2.5.

2.1 TEORIA DO APRENDIZADO ESTATÍSTICO

Proposta por (VAPNIK; CHERVONENKIS, 1974), a Teoria do Aprendizado Estatístico, conhecida por "Dimensão VC" (de Vapnik-Chervonenkis), determina vários princípios utilizados no funcionamento das SVMs. Esses princípios podem ser utilizados na obtenção de classificadores com boa generalização, ou seja, é a habilidade de prever novos casos a partir das hipóteses geradas durante o processo de aprendizagem.

O objetivo da Teoria do Aprendizado Estatístico é propor um cenário de estudo para os princípios e problemas de inferência, que se baseia no reconhecimento de padrões, na aquisição de conhecimento, na tomada de decisões ou na formação de modelos a partir de um conjunto de dados.

A proposta das SVMs é a construção de um hiperplano ótimo que minimize, simultaneamente, o risco empírico e a Dimensão VC (e que minimize esta última já ao maximizar a margem de separação das classes, satisfazendo a minimização do risco estrutural).

Para que um classificador f possua um bom desempenho, é necessário que ele tenha obtido o menor erro ao separar as amostras de duas classes durante o treinamento. Esse erro é mensurado através do número de predições incorretas de f. Dessa forma, denomina-se o risco empírico $R_{emp}(f)$ como sendo a medida de perda entre a resposta almejada e a resposta real. A Equação 2.1 demonstra a definição do risco empírico.

$$R_{emp}(f) = \frac{1}{n}\sum_{i=1}^{n} c(f(\mathbf{x}_i), y_i) \tag{2.1}$$

onde $c(\cdot)$ é a função de custo relacionada à previsão $f(\mathbf{x}_i)$ com a saída desejada y_i (LORENA; CARVALHO, 2007), na qual um tipo de função de custo é a "perda 0/1" definida pela Equação 2.2. O processo de busca por uma função f' que represente um menor valor de R_{emp} é denominado

[1] Redes Neurais Artificiais são técnicas computacionais que apresentam um modelo matemático inspirado na estrutura neural de organismos inteligentes e que adquirem conhecimento através da experiência adotando o princípio de minimização do risco empírico (ERM, do inglês, *empirical risk minimization*).

de Minimização do Risco Empírico.

$$c(f(\mathbf{x}_i), y_i) = \begin{cases} 1, & \text{se } y_i f(x_i) < 0 \\ 0, & \text{caso contrário} \end{cases} \tag{2.2}$$

A Dimensão VC serve para determinar a capacidade de um espaço de hipóteses. Essa capacidade é uma medida da complexidade e considera a força expressiva, a riqueza ou a flexibilidade que um conjunto de funções possui (VAPNIK, 1999).

Quanto maior o valor da Dimensão VC, mais complexas são as funções de classificação que podem ser induzidas. Dado um problema de classificação binário, essa dimensão é definida como o número máximo de exemplos que podem ser particionados em duas classes.

O princípio da minimização de risco estrutural, baseia-se no erro de treinamento $(v_{treino}(w))$ ou seja, na frequência de erros cometidos por uma máquina de aprendizagem com um vetor de peso **w** durante a sessão de treinamento. Similarmente, o erro de generalização $(v_{gene}(w))$ é definido como a frequência dos erros cometidos pela máquina quando é testada com exemplos não vistos anteriormente. Assume-se que os dados de teste são retirados da mesma população de onde foram retirados os dados de treinamento (HAYKIN, 2001).

2.2 TREINAMENTO DE UMA SVM

O treinamento de uma SVM é obtido a partir da solução de um problema de programação quadrática, ou seja, o número de variáveis no problema de otimização é ou igual ao número de amostras do conjunto de treinamento (SVM para classificação de padrões) ou duas vezes este número (SVM para regressão). Durante o treinamento dos dados é muito importante que as regras de decisão geradas pelo algoritmo de aprendizagem possam generalizar adequadamente os exemplos desconhecidos.

Existe um parâmetro de custo C, ou termo de regularização, que permite flexibilizar a separação das classes que controlam o compromisso entre possibilitar erros de treinamento ou forçar margens rígidas. Através desse parâmetro é possível criar uma margem flexível que torna possíveis alguns erros de classificação.

O aumento do valor de C se dá devido ao custo de exemplos classificados incorretamente, o que impõe o estabelecimento de um modelo mais complexo que pode não generalizar adequadamente exemplos desconhecidos (AL., 2004).

Na Seção 2.2.1 são apresentadas as SVMs para problemas binários; na Seção 2.2.2, as SVMs para problemas multiclasses e na Seção 2.2.3, as SVMs com opção de rejeição.

2.2.1 SVMs para problemas binários

O problema de classificação binária é o mais simples de todos. Talvez por isso seja considerado o mais importante. Na verdade, ele possibilita que qualquer problema seja reduzido a ele com apenas algumas suposições. São utilizadas duas classes ou rótulos que são definidos como -1 e 1. A interpretação do significado de cada uma das duas classes pode variar bastante. Nesse caso, procura-se separar duas classes, colocando-se cada classe de um lado diferente de um hiperplano ótimo.

2.2.2 SVMs para problemas multiclasses

As SVMs foram propostas originalmente como ferramentas de classificação binária. Porém, muitas aplicações necessitam do emprego de mais de duas classes. Isso não inviabiliza a utilização das SVMs. Para possibilitar que as SVMs tratem de problemas desse tipo, foram propostas diversas técnicas para estender as SVMs binárias.

Formalmente, um sistema multiclasse possui um conjunto de treinamento composto por pares (x_i, y_i), tal que $y_i \in \{1, ..., k\}$, com $k > 2$, onde k é o número de classes.

As duas abordagens utilizadas para realização dessa tarefa são chamadas de "decomposição Um-Contra-Todos" e "Todos-Contra-Todos".

O método de decomposição Um-Contra-Todos se baseia na construção de k classificadores binários, sendo que k representa o número de classes. Cada classificador f_i é responsável pela classificação de uma classe i das demais.

Segundo um novo padrão x a classe à qual esse novo padrão pertence é a representada pelo classificador que encontrou o valor máximo entre k classificadores, definida por:

$$f(x) = arg \max_{1 \leq i \leq k}(f_i(x)). \tag{2.3}$$

O método de decomposição Todos-Contra-Todos consiste na comparação das classes duas a duas, sendo necessário $k.(k-1)/2$ SVMs, onde k representa o número de classes. Para decidir-se a qual classe pertence um padrão x novo, é utilizado um esquema de votação por maioria, onde cada uma das SVMs fornece um resultado. A solução final é dada pela classe com maior número de votos.

É necessário um alto tempo de processamento caso sejam executadas várias SVMs. Por exemplo, se $k = 5$, seriam necessárias 5 SVMs segundo o método Um-Contra-Todos, enquanto seriam necessárias 10 SVMs se o método utilizado fosse o Todos-Contra-Todos.

2.2.3 SVMs com opção de rejeição

Nos problemas de decisão onde os erros implicam graves perdas, faz-se necessária a construção de mecanismos para evitar a classificação de exemplos ambíguos. Os exemplos de rejeição têm sido investigados desde os primeiros dias do reconhecimento de padrões. Em particular, (CHOW, 1970) analisa como a taxa de erro pode ser diminuída graças à opção de rejeição.

A opção de rejeição é muito útil para que as aplicações de reconhecimento de padrões possam se proteger contra o excesso de erros de classificação, onde a alta confiabilidade é necessária. Chow definiu uma regra de classificação ótima com opção de rejeição no âmbito do princípio da minimização do risco estrutural. No caso mais simples, onde os custos de classificação não dependem das classes, a regra de Chow consiste em rejeitar um padrão cuja probabilidade posterior máxima for inferior a um determinado limite. A otimização dessa regra depende do conhecimento exato da probabilidade posterior. No entanto, nas aplicações práticas, as probabilidades posteriores geralmente são desconhecidas. As SVMs não fornecem saídas probabilísticas. Nesse caso deve ser usada uma técnica de rejeição voltada para o classificador.

A opção de rejeição é implementada segundo duas abordagens. A primeira utiliza como medida de confiança de classificação a distância $d(x)$ de um padrão de entrada x de um hiperplano de separação ótimo, no espaço de característica induzida pelo *kernel* escolhido. A regra de rejeição consiste em recusar padrões onde $d(x)$ for inferior ao limite predefinido. Uma vez que o valor absoluto do $|f(x)|$ de saída da SVM é proporcional a $d(x)$, esta regra é implementada aplicando-se um limite de rejeição $|f(x)|$. A segunda abordagem que implementa a opção de rejeição nas SVMs consiste em mapear as saídas das probabilidades posteriores, de forma que a regra de Chow possa ser aplicada. Normalmente, para classificadores de distância (como o discriminador linear de (FISHER, 1936)), o mapeamento é implementado usando uma função sigmoide [2]. Esse método também foi proposto para as SVMs utilizando a função sigmoide, definido por:

$$P(y = +1|x) = \frac{1}{1 + exp(af(x) + b)}. \tag{2.4}$$

Aqui, o rótulo da classe é indicado como $y = +1, -1$, enquanto que a e b são termos constantes a serem definidos com base nos dados de exemplo. Nesse caso, as constantes a e b são escolhidas de modo que $P(y = +1|x) = 0.5$, se $f(x) > 0$, para padrões que caiam a uma distância $-1/||w||$ do hiperplano de separação ótimo.

Isso implica que a regra de Chow aplicada a tais estimativas é equivalente à regra de rejeição obtida através da aplicação direta de um limite de rejeição sobre o valor absoluto da saída $|f(x)|$. Ambas as regras proporcionam uma região de rejeição cujos limites consistem

[2] A função sigmoide é uma função matemática amplamente utilizada em áreas, tais como, economia e computação. O nome "sigmoide" vem da forma em S do seu gráfico.

de um par de hiperplanos paralelos ao hiperplano de separação ótimo e equidistantes dele. A distância de tais hiperplanos ao hiperplano de separação ótimo depende do valor do limite de rejeição (FUMERA; ROLI, 2002).

2.3 HIPERPLANO ÓTIMO PARA PADRÕES SEPARÁVEIS

Segundo uma amostra de treinamento $\{(\mathbf{x}_i, d_i)\}_{i=1}^{N}$, onde $\mathbf{x}_i$ é o padrão de entrada para o i-ésimo exemplo e d_i é a resposta desejada correspondente (saída-alvo) e assumindo-se que o padrão (classe) representado pelo subconjunto $d_i = +1$ e o padrão representado pelo subconjunto $d_i = -1$ são linearmente separáveis, a equação de uma superfície de decisão na forma de um hiperplano que realiza essa separação.

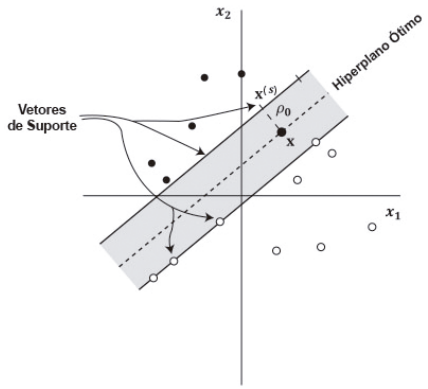
$$\mathbf{w}^T\mathbf{x} + b = 0 \tag{2.5}$$

onde **x** é um vetor de entrada, **w** é um vetor de peso ajustável e b é um limiar também conhecido como "bias".

$$\begin{cases} \mathbf{w}^T\mathbf{x}_i + b \geq 0, & \text{para } d_i = +1 \\ \mathbf{w}^T\mathbf{x}_i + b < 0, & \text{para } d_i = -1 \end{cases} \tag{2.6}$$

Para um dado vetor **w** de peso e b de bias, a separação entre o hiperplano da Equação 2.6 e o ponto de dado mais próximo é denominado "margem de separação", representada por ρ. O objetivo da técnica das SVMs é encontrar o hiperplano particular para qual a margem de separação ρ é máxima. Sob essa condição, a superfície de decisão é referida como o hiperplano ótimo conforme figura 1 a seguir.

Figura 1 – Hiperplano ótimo para um espaço de entrada bidimensional



Fonte: HAYKIN (2001)

Os vetores $\mathbf{w}_0$ e b_0 representam os valores ótimos do vetor peso e do bias, respectivamente. Consequentemente, o hiperplano ótimo, representando uma superfície de decisão linear

multidimensional no espaço de entrada é definido por:

$$\mathbf{w}_0^T \mathbf{x}_0 + b_0 = 0. \tag{2.7}$$

A função discriminante, definida por:

$$g(\mathbf{x}) = \mathbf{w}_0^T \mathbf{x} + b_0. \tag{2.8}$$

Esta equação fornece uma medida algébrica da distância de **x** até o hiperplano. É possível expressar **x** da seguinte forma:

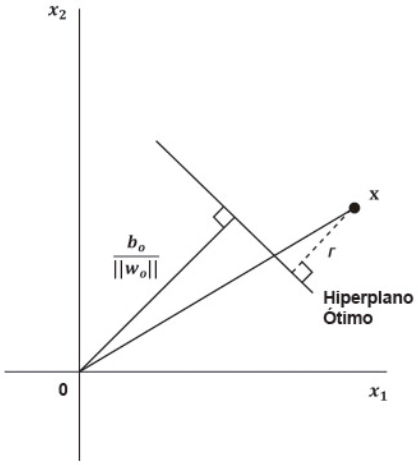
$$\mathbf{x} = \mathbf{x}_\rho + r\frac{\mathbf{w}_0}{||\mathbf{w}_0||} \tag{2.9}$$

onde $\mathbf{x}_\rho$ é a projeção normal de **x** sobre o hiperplano ótimo e r é a distância algébrica desejada; r é positivo se **x** estiver no lado positivo do hiperplano ótimo e negativo se **x** estiver no lado negativo, de forma que, por definição, $g(\mathbf{x}_\rho) = 0$, resultando:

$$g(\mathbf{x}) = \mathbf{w}_0^T \mathbf{x}_0 + b_0 = r||\mathbf{w}_0||. \tag{2.10}$$

A distância da origem (i. e., **x** = 0) até o hiperplano ótimo é dada por $b_0/||\mathbf{w}_0||$. Se $b_0 > 0$, a origem está do lado positivo do hiperplano ótimo; se $b_0 < 0$ ela está no lado negativo. Se $b_0 = 0$, o hiperplano ótimo passa pela origem, conforme representação geométrica da figura 2, a seguir.

Figura 2 – Distância entre os pontos de um hiperplano ótimo



Fonte: HAYKIN (2001)

Para encontrarem-se os parâmetros $\mathbf{w}_0$ e b_0 do hiperplano ótimo, dado o conjunto de treinamento $\tau = \{(\mathbf{x}_i, d_i)\}$, com base na Figura 2, tem-se que o par $(\mathbf{w}_0, b_0)$ deve satisfazer as

seguintes restrições:

$$\begin{cases} \mathbf{w}_0^T \mathbf{x}_i + b_0 \geq 1, & \text{para } d_i = +1 \\ \mathbf{w}_0^T \mathbf{x}_i + b_0 \leq 1, & \text{para } d_i = -1 \end{cases} \tag{2.11}$$

se este par for válido, pode-se concluir que os padrões são linearmente separáveis e sempre é possível escalar $\mathbf{w}_0$ e b_0 de maneira que a Equação 2.11 é válida.

Os pontos de dados particulares $(\mathbf{x}_i, d_i)$ para os quais a primeira ou a segunda linha da Equação 2.11 é satisfeita com o sinal de igualdade são chamados de "vetores de suporte". Eles desempenham um papel importante na operação na classe de máquinas de aprendizagem, pois representam os pontos em que os dados se encontram mais próximos da superfície de decisão e são, portanto, os mais difíceis de classificar. Dessa forma, eles têm uma influência direta na localização ótima da superfície de decisão.

Para um vetor de suporte $\mathbf{x}^{(s)}$, para o qual $d^{(s)} = +1$, por definição tem-se:

$$g(\mathbf{x}^{(s)}) = \mathbf{w}_0^T \mathbf{x}^{(s)} + b_0 = \pm 1. \tag{2.12}$$

Assim, assume-se a distância algébrica do vetor de suporte $\mathbf{x}^{(s)}$ até o hiperplano ótimo, definido por:

$$\begin{aligned} r &= \frac{g(\mathbf{x}^{(s)})}{||\mathbf{w}_0||} \\ &= \begin{cases} \frac{1}{||\mathbf{w}_0||} & \text{para } d^{(s)} = +1 \\ -\frac{1}{||\mathbf{w}_0||} & \text{para } d^{(s)} = -1 \end{cases} \end{aligned} \tag{2.13}$$

aqui, o sinal positivo indica que $\mathbf{x}^{(s)}$ se encontra no lado positivo do hiperplano ótimo e o sinal negativo indica que $\mathbf{x}^{(s)}$ está no lado negativo do hiperplano ótimo. Considera-se que ρ represente o valor ótimo da margem de separação entre as duas classes que constituem o conjunto de treinamento τ, resultando:

$$\begin{aligned} \rho &= 2r \\ &= \frac{2}{||\mathbf{w}_0||}. \end{aligned} \tag{2.14}$$

A Equação 2.14 indica que maximizar a margem de separação entre classes é equivalente a minimizar a norma euclidiana do vetor peso $\mathbf{w}$. O hiperplano ótimo definido pela Equação 2.14

é único no sentido de que o vetor peso $\mathbf{w}_0$ fornece a máxima separação possível entre exemplos positivos e negativos. Esta condição ótima é alcançada minimizando-se a norma euclidiana do vetor peso **w** (HAYKIN, 2001).

2.4 HIPERPLANO ÓTIMO PARA PADRÕES NÃO SEPARÁVEIS

Existem casos em que através do conjunto de dados de treinamento não é possível construir-se um hiperplano de separação sem que ocorram erros de treinamento. Nesses casos faz-se necessário encontrar um hiperplano ótimo que minimize a probabilidade de erro de classificação, calculada como a média sobre o conjunto de treinamento.

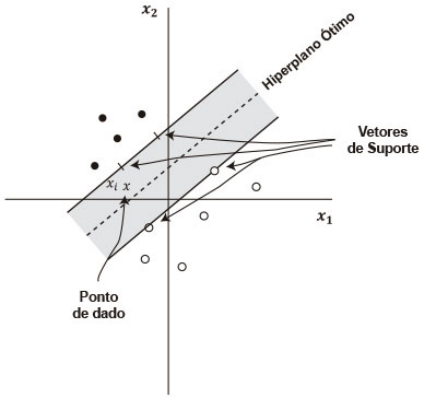
A margem de separação entre classes é considerada suave se um ponto de dado $(\mathbf{x}_i, d_i)$ violar a condição da equação, de acordo com:

$$d_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \quad i = 1, 2, \ldots N. \tag{2.15}$$

Tal violação pode surgir de duas formas:

- O ponto de dado $(\mathbf{x}_i, d_i)$ encontra-se dentro da região de separação, mas do lado correto da superfície de decisão, como ilustrado na figura 3.

Figura 3 – O ponto de dado $\mathbf{x}_i$ (pertencente à classe C_1)

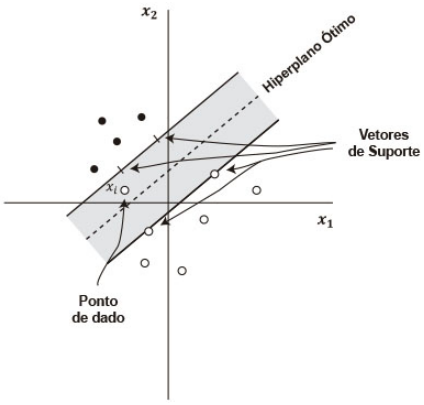


Fonte: HAYKIN (2001)

- O ponto de dado $(\mathbf{x}_i, d_i)$ se encontra no lado errado da superfície de decisão, como ilustrado na Figura 4.

Nota-se que a classificação é correta na Figura 3 e incorreta na Figura 4.

Figura 4 – O ponto de dado $\mathbf{x}_i$ (pertencente à classe C_2)



Fonte: HAYKIN (2001)

Para preparar-se o terreno para um tratamento formal para o caso de pontos de dados não separáveis, introduz-se um novo conjunto de variáveis escalares não negativas, $\{\xi_i\}_{i=1}^{N}$, na definição do hiperplano de separação (i.e., superfície de decisão), de acordo com:

$$d_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \quad \text{para } i = 1, 2, \ldots N. \tag{2.16}$$

As ξ_i são chamadas de variáveis soltas e servem para medir o desvio de um ponto de dado da condição ideal de separação de padrões. Para $0 \geq \xi_i \leq 1$, o ponto de dado encontra-se dentro da região de separação, mas no lado correto da superfície de decisão, como ilustrado na Figura 4. Para $\xi_i < 1$ ele se encontra no lado errado do hiperplano de separação, como ilustrado na Figura 5. Os vetores de suporte são aqueles pontos de dados particulares que satisfazem a Equação 2.16 precisamente, mesmo se $\xi_i > 0$. Nota-se que, se um exemplo com $\xi_i > 0$ for deixado de fora do conjunto de treinamento, a superfície de decisão não muda. Assim, os vetores de suporte são definidos exatamente do mesmo modo tanto para o caso linearmente separável como para o caso não separável.

O objetivo é encontrar um hiperplano de separação para o qual o erro de classificação, como média sobre o conjunto de treinamento, é minimizado. Pode-se fazer isso minimizando o funcional, definida por:

$$\Phi(\xi) = \sum_{i=1}^{N} I(\xi_i - 1). \tag{2.17}$$

Em relação ao vetor peso **w** sujeito à restrição descrita na Equação 2.16, e à restrição sobre $||\mathbf{w}||^2$, função $I(\xi)$ é uma função indicadora, definida por:

$$I(\xi) = \begin{cases} 0, & \text{para } \xi \leq 0 \\ 1, & \text{para } \xi > 0. \end{cases} \tag{2.18}$$

A minimização de $\Phi(\xi)$ em relação a **w** é um problema de otimização não-convexo, que é NP-completo (não determinístico em tempo polinomial). Para tornar o problema matematicamente tratável, aproximamos o funcional $\Phi(\xi)$, definida por:

$$\Phi(\xi) = \sum_{i=1}^{N} \xi_i. \tag{2.19}$$

Além disso, simplificamos a computação, formulando o funcional a ser minimizado em relação ao vetor peso **w**, de acordo com:

$$\Phi(\mathbf{w}, \xi) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N} \xi_i. \tag{2.20}$$

Como anteriormente, a minimização do primeiro termo da Equação 2.20 está relacionada à minimização da Dimensão VC da SVM. Assim como para o segundo termo $\sum_i \xi_i$, ele é um limite superior para o número de erros de teste. A formulação da função de custo $\Phi(\mathbf{w}, \xi)$ a Equação 2.20 está, portanto, em perfeito acordo com o princípio da minimização estrutural de risco.

O parâmetro C deve ser selecionado pelo usuário. Isso pode ser feito de duas formas:

- O parâmetro C é determinado experimentalmente através do uso padrão de um conjunto de treinamento/teste (validação), o que é uma forma grosseira de repetição da amostragem.

- Ele é determinado analiticamente estimando-se a Dimensão VC, usando-se limites do desempenho de generalização da máquina baseados na Dimensão VC.

De qualquer forma, o funcional $\Phi(\mathbf{w}, \xi)$ é otimizado em relação a **w** e a $\{\xi_i\}_{i=1}^{N}$, bem como à restrição descrita na equação 2.16 e a $\xi_i, \geq 1$. Fazendo isso, a norma quadrada de **w** é tratada como uma quantidade a ser minimizada simultaneamente em relação aos pontos não separáveis, e não como uma restrição imposta sobre a minimização do número desses pontos.

O problema de otimização para padrões não separáveis, assim formulado, inclui o problema de otimização para padrões linearmente separáveis como um caso especial. Especificamente, fazer $\xi_i = 0$ para todo i nas Equações 2.16 e 2.20 as reduz às formas correspondentes para o caso linearmente separável.

Pode-se agora formalizar o problema primordial para o caso não separável. Dada a amostra de treinamento $(\mathbf{x}_i, d_i)_{i=1}^{N}$, encontram-se os valores ótimos dos vetores de peso **w** e do

bias[3] b de modo que satisfaçam à restrição, de acordo com:

$$\begin{aligned} d_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, -\xi_i, \quad \text{para } i = 1, 2, \ldots N \\ \xi_i \geq 0, \quad \text{para todo } i. \end{aligned} \tag{2.21}$$

Deste modo, o vetor peso **w** e as variáveis soltas ξ_i minimizam o funcional de custo, como na Equação 2.20, onde C é um parâmetro positivo especificado pelo usuário.

Usando-se o método dos multiplicadores de Lagrange [4], pode-se formular o problema dual para padrões não separáveis da mesma forma da Equação 2.22.

Dada a amostra de treinamento $\{(\mathbf{x}_i, d_i)\}_{i=1}^N$ encontram-se os multiplicadores de Lagrange $\{\alpha_i\}_{i=1}^N$ que maximizam a função objetivo, definida por:

$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i\alpha_j d_i d_j \mathbf{x}_i^T\mathbf{x}_j \tag{2.22}$$

essa equação está sujeita às restrições:

$$\begin{aligned} \sum_{i=1}^{N} \alpha_i d_i = 0 \\ \text{e} \quad 0 \leq \alpha_i \leq C \quad \text{para } i = 1, 2, \ldots N. \end{aligned} \tag{2.23}$$

Aqui, C é um parâmetro positivo especificado pelo usuário. Nota-se que nem as variáveis soltas ξ_i nem os multiplicadores de Lagrange aparecem no problema dual. O problema dual para o caso de padrões não separáveis é, dessa forma, similar àquele para o caso simples de padrões linearmente separáveis, exceto por uma diferença pequena, mas importante. A função objetivo $Q(\alpha)$ a ser maximizada é a mesma em ambos os casos. O caso não separável difere do caso separável pelo fato de que a restrição $\alpha_i \geq 0$ é substituída pela restrição mais rigorosa $0 \leq \alpha_i \leq C$. Exceto por essa modificação, a otimização restrita para o caso não separável e os cálculos dos valores ótimos do vetor peso **w** e do bias b procedem do mesmo modo como no caso linearmente separável.

A solução ótima para o vetor peso **w** é definida por:

$$\mathbf{w}_0 = \sum_{i=1}^{N_s} \alpha_{0,i} d_i \mathbf{x}_i \tag{2.24}$$

onde N_s é o número de vetores de suporte (HAYKIN, 2001).

[3] "Bias" ou "viés" é um termo usado em estatística para expressar o erro sistemático ou tendenciosidade.

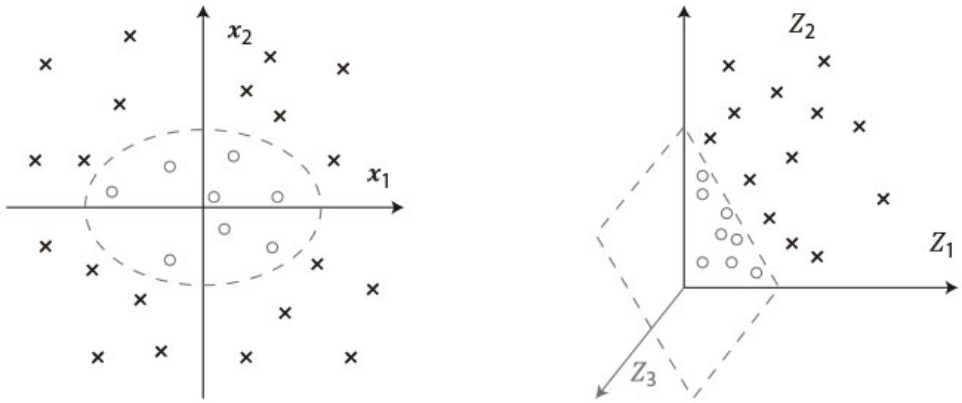
[4] Em matemática, em problemas de otimização, o método dos multiplicadores de Lagrange permite encontrar extremos (máximos e mínimos) de uma função de uma ou mais variáveis suscetíveis a uma ou mais restrições.

2.5 CLASSIFICADORES NÃO LINEARES

As SVMs utilizam estruturas lineares cujo resultado obtido é um hiperplano capaz de maximizar a margem de separação entre duas classes. Pode-se considerar que uma máquina linear seja bastante limitada (MINSKY; PAPERT, 1969) para os problemas complexos de classificação de padrões encontrados hoje em dia. Por isso, se faz necessário buscar alguma forma que permita a uma SVM atuar em problemas de classificação não linear. Para isso utilizam-se funções não lineares de *kernel* ou núcleo, úteis para mapear vetores de entrada em um espaço de alta dimensionalidade denominado "espaço de características". O hiperplano gerado pelas SVMs nesse espaço de características, ao ser mapeado de volta ao espaço de entrada, torna-se uma superfície não linear. Desta forma, o hiperplano de separação passa a ser uma função linear de vetores do espaço de características (Figura 5).

Uma característica fundamental da técnica das SVMs é a margem de separação entre as classes que estão associadas ao erro de classificação permitido. O usuário controla essa margem de separação como um parâmetro extra de treinamento. A formulação do problema de separação consiste em um problema de programação quadrática que possui restrições lineares e que dependem dos dados de treinamento e dos parâmetros dos *kernels*, além da margem de separação. O resultado do problema de programação quadrática consiste em vetores de suporte dentro da margem de separação definida pelo usuário (REZENDE, 2003).

Figura 5 – Exemplo de uso de funções *kernel*



Fonte: SMOLA; SCHÖLKOPF (2002)

2.6 CONSIDERAÇÕES FINAIS

Neste capítulo foi apresentado um tipo de técnica de aprendizado estatístico denominado SVMs que utiliza a Teoria do Aprendizado Estatístico, constituída por princípios, tais como, minimização do risco empírico, minimização do risco estrutural e dimensão VC para resolver problemas de classificação. Na fase de treinamento pode-se utilizar o aprendizado supervisionado (no qual as classes possuem um rótulo preestabelecido) ou o aprendizado não supervisionado

(em que as classes não possuem rótulos, e cabe ao algoritmo de SVM a separação das classes). Existem problemas binários (formados por duas classes) e problemas multiclasses (formados por mais de duas classes). Para que as classes estejam separadas umas das outras, é necessário que exista um hiperplano de separação entre elas. Quando não existem exemplos posicionados entre as margens de decisão, pode-se considerar o hiperplano como ótimo para padrões separáveis. Se existirem, há que se minimizar a probabilidade de erro dos testes, durante a classificação, para estabelecer-se o hiperplano como ótimo para padrões não separáveis. Também são introduzidos os conceitos de classificadores lineares e não lineares, através dos quais se estabelece o uso ou não de *kernels* para classificarem-se, da melhor maneira, as classes.

O aprofundamento teórico realizado neste capítulo demonstrou que, para a mineração de textos realizada neste trabalho, foi necessária a utilização das SVMs para problemas multiclasses, já que estas consistem em um número maior que dois. Além disso, o tipo de aprendizado empregado na fase de treinamento foi o supervisionado, pois o algoritmo de aprendizagem irá retirar um classificador de exemplos já rotulados como positivos, negativos e regulares.

No próximo capítulo serão referenciadas algumas das principais ferramentas disponíveis e que foram consideradas para a realização deste trabalho, bem como o uso e compatibilidade destas com as SVMs.

3 MATERIAIS E MÉTODO

Este capítulo apresenta os materiais e o método proposto neste trabalho. Inicia-se apresentando as etapas do trabalho, na forma de um percurso metodológico. Em seguida, apresenta-se a relação de ferramentas de desenvolvimento relacionada à área de aprendizagem de máquina (AM) e suas respectivas bibliotecas ou extensões para utilização da técnica de SVMs, com o objetivo de classificação de textos. As métricas que podem ser usadas nesta área também são descritas neste capítulo.

3.1 PERCURSO METODOLÓGICO

Para este trabalho seguiu-se um percurso desenvolvido em dez etapas:

1. Escolha da plataforma de Aprendizagem de Máquina
2. Seleção do método e métricas
3. Seleção dos textos
4. Carga dos textos
5. Preparação dos textos
6. Testes dos algoritmos
7. Variação dos parâmetros
8. Tabulação dos resultados
9. Análise dos resultados com identificação do melhor caso
10. Justificativas da escolha

As seções seguintes descrevem o planejamento das três primeiras etapas. As demais sete etapas são apresentadas no Capítulo 4, referente a descrição do desenvolvimento e testes da implementação deste trabalho.

3.2 PRIMEIRA ETAPA: ESCOLHA DA PLATAFORMA DE APRENDIZAGEM DE MÁQUINA

As ferramentas analisadas neste trabalho foram as seguintes: *Weka, Scikit-learn* e a biblioteca *LibSVM*. Elas foram selecionadas por serem amplamente utilizadas, gratuitas e bem

avaliadas para projetos de aprendizado de máquina, nas diversas áreas do conhecimento humano (PEDREGOSA *et al.*, 2020). Além delas, existem outras bibliotecas para aprendizado de máquina, com funcionalidades similares que poderia ser utilizadas. A escolha se deu baseada em critérios de afinidade da linguagem *Python* e potencial de uso e resultados das ferramentas.

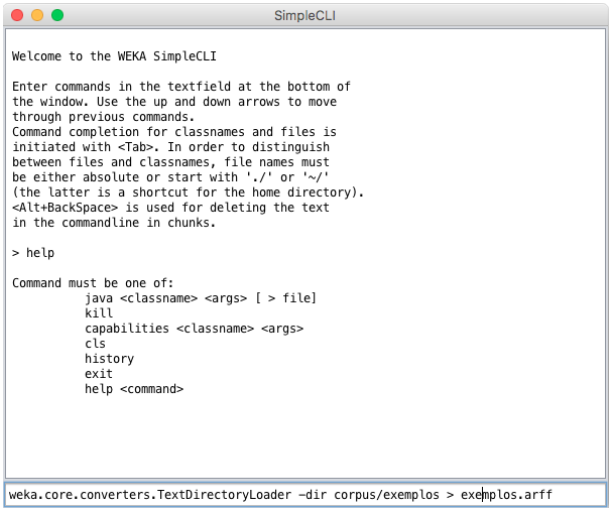
3.2.1 *Weka*

O pacote de programa *Weka*[1] *(Waikato Environment for Knowledge Analisys)* está correntemente na versão 3.8.4, é implementado em *Java*, possui *software* de código aberto, é multiplataforma e contém uma coleção de algoritmos utilizados nas tarefas de mineração de dados.

O *Weka* possui uma interface gráfica (GUI, do inglês, *Graphical User Interface*) que possibilita utilizar vários recursos do aplicativo de forma mais simples. Em uma das janelas desta interface, chamada de "*Explorer*", são disponibilizadas as seguintes abas: *Preprocess, Classify, Cluster, Associate, Select attributes* e *Visualize*.

Para utilizar um arquivo no *Weka* é necessário que ele esteja no formato padrão ARFF[2] (do inglês, *Attribute-Relation File Format*) ou no formato CSV (do inglês, *Comma-separated Values*). Posteriormente, na aba *Classify*, é possível utilizar vários algoritmos de classificação. Os algoritmos não disponíveis na instalação podem ser obtidos através do gerenciador de pacotes disponibilizado pelo próprio aplicativo. Ele também é composto por um terminal para executar aplicações *Java* via linha de comando através da janela *SimpleCLI*, conforme Figura 6.

Figura 6 – Exemplo de tela do *Weka SimpleCLI*



Fonte: o autor (2020)

Os algoritmos podem ser aplicados diretamente a um conjunto de dados ou utilizados a partir de um código escrito em *Java*. O *Weka* dispõe de bibliotecas padrão para utilização

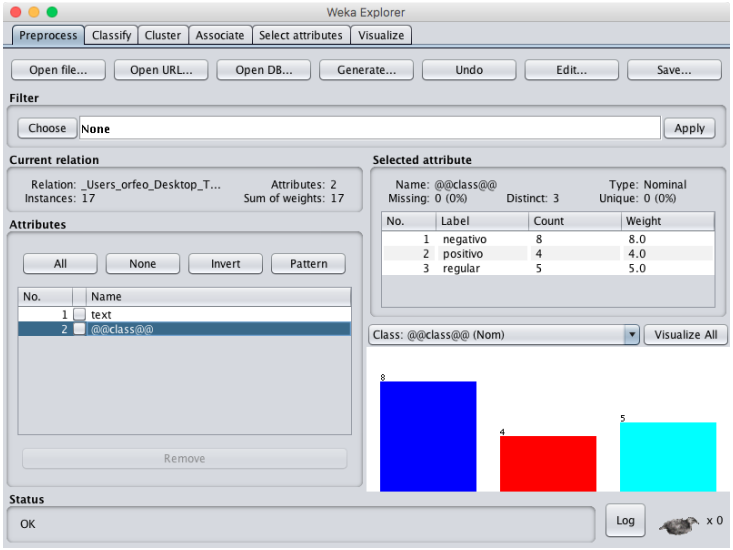
[1] http://www.cs.waikato.ac.nz/ml/weka/
[2] https://www.cs.waikato.ac.nz/ ml/weka/arff.html

do algoritmo SVM, porém, somente para padrões lineares. No caso de utilização de *kernels* para padrões não lineares é utilizada a biblioteca *LibSVM*[3]. Esta biblioteca é desenvolvida para aplicação do SVM que agrega as funções mais usadas em muitas ferramentas de produção em um único programa. Possui extensões e interfaces para diversas linguagens e IDEs. Desenvolvido por (CHANG; LIN, 2020), tem o código disponibilizado de forma livre e suporta a classificação multiclasse.

O aplicativo *Weka* possui a função *TextDirectoryLoader*, que carrega todos arquivos de um diretório com seus respectivos subdiretórios, já nomeados de acordo com a classe a que se referem, em um arquivo único no formato ARFF. Esta função pode ser executada via linha de comando direto na janela do *SimpleCLI*, facilitando a utilização da ferramenta.

A partir da execução da função de carregamento é possível salvar o arquivo no formato ARFF para posterior utilização desses dados no aplicativo. O carregamento de um arquivo se dá através da aba *Preprocess*, localizada na janela do *Weka Explorer*, pelo botão *Open file*, conforme a Figura 7. A partir disso, é possível observar os diferentes pesos das classes rotuladas de forma gráfica ou tabular.

Figura 7 – Exemplo de tela do *Weka Explorer* na aba *Preprocess*



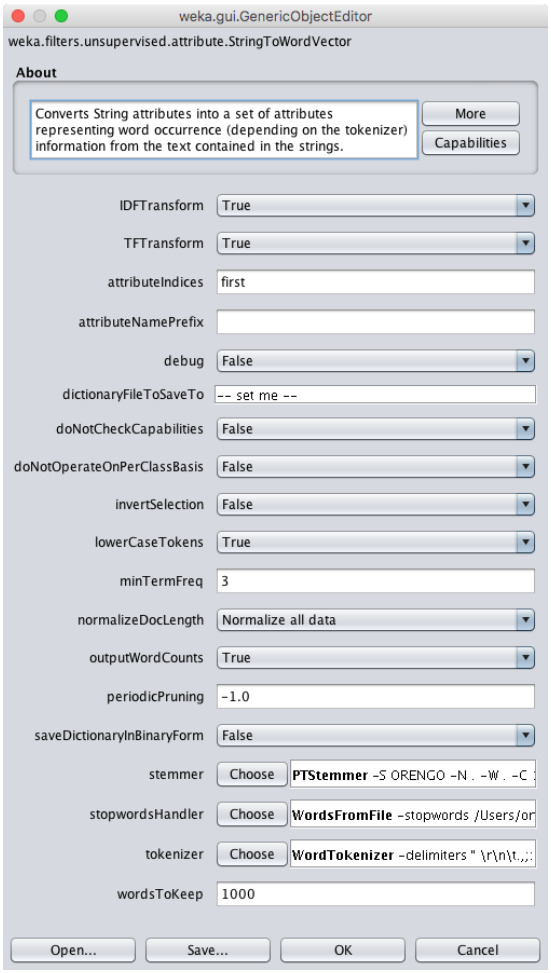
Fonte: o autor (2020)

Segundo os dados utilizados para os testes das ferramentas deste trabalho, conforme a Figura 7, o *Weka* apresentou o *corpus* subdividido em 8 classes negativas, 4 classes positivas e 5 classes regulares.

Para realizar o pré-processamento dos textos e converter os *strings* em um conjunto de atributos que representam a ocorrência das palavras, é necessária a utilização da função *StringToWordVector*, de acordo com a Figura 8. Essa função é encontrada ainda na aba *Preprocess*, na pasta dos filtros não supervisionados.

[3] https://www.csie.ntu.edu.tw/ cjlin/libsvm/

Figura 8 – Exemplo de tela para função *StringToWordVector*



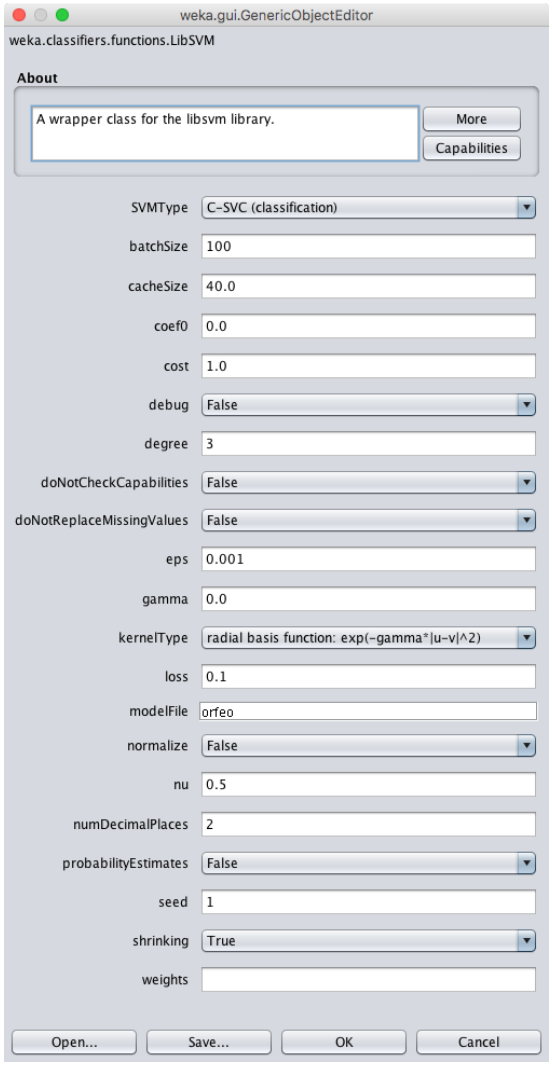
Fonte: o autor (2020)

Esta função possui vários parâmetros importantes para o pré-processamento de textos, tais como, a verificação da frequência dos termos nos documentos (*IDFTransform*, do inglês, *Inverse Document Frequency Transform*) e a frequência inversa dos termos (*TFTransform*, do inglês, *Term Frequency Transform*), que busca diminuir o peso dos termos que ocorrem com mais frequência nos documentos e aumentar o peso dos que ocorrem mais raramente. Também existem parâmetros para contagem dos termos (*outputWordCounts*), para conversão e padronização em letras minúsculas (*lowerCaseTokens*), para determinar uma frequência mínima para que um termo seja considerado (*minTermFreq*) e para normalizar a frequência de utilização dos termos em relação ao número total de termos do documento (*normalizeDocLength*). Ainda é possível a escolha de filtros que removem os afixos morfológicos, mantendo apenas o radical dos termos e consequentemente reduzindo o volume total destes, e para a eliminação das *stopwords*, que são alguns dos termos mais comuns empregados nos textos, mas que não agregam sentido.

A execução da função *StringToWordVector* retorna os dados pré-processados de forma vetorial. Após, é executado o algoritmo SVM para iniciar a fase de treinamento de acordo com a Figura 8. Para selecionar essa função deve ser escolhida a aba *Classify*, segundo Figura 7.

O *wrapper* para biblioteca *LibSVM*, conforme a Figura 9, apresenta inúmeros parâmetros empregados na fase de treinamento do algoritmo SVM.

Figura 9 – Exemplo de tela do *wrapper* para função *LibSVM*

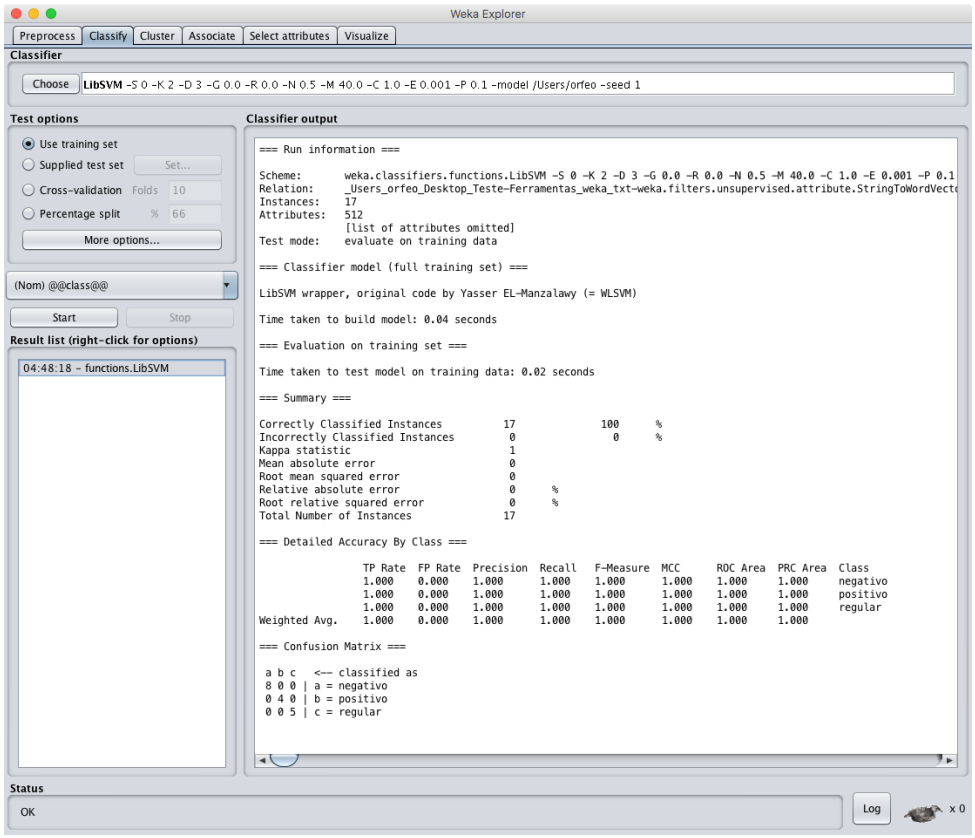


Fonte: o autor (2020)

Após executar este passo, o *Weka* fornece um relatório, no campo *Classifier output*, de acordo com a Figura 10.

Depois de executado o processo de treinamento, é possível escolher a opção *Supplied test set*, de acordo com a Figura 10, para carregar um conjunto de textos para teste de classificação. Finalizados os processos de treinamento e teste, é possível verificar os resultados da classificação automática através do *log Classifier output*.

Figura 10 – Exemplo de tela do *log* dos resultados obtidos



Fonte: o autor (2020)

3.2.2 *Scikit-learn*

O *Scikit-learn*[4] é um módulo *Python* que contém ferramentas para análise e mineração de dados com foco em AM. Possui código aberto e utiliza o *SciPy*[5] e o *NumPy*[6] como base. O projeto foi iniciado em 2007 por David Cournapeau, como um projeto *Google Summer of Code*, e desde então tem sido mantido por voluntários (PEDREGOSA *et al.*, 2020). Vários algoritmos de classificação, regressão, agrupamento e SVM são disponibilizados. O módulo *sklearn.svm* inclui vários algoritmos para utilização com a técnica de SVMs (Figura 11), tais como, *SVC, NuSVC e LinearSVC*, para execução de classificação multiclasse. Essa ferramenta possui ótima documentação atualizada e um amplo canal de suporte. Além disso, é possível contar com o módulo *NLTK*[7] (do inglês, *Natural Language Toolkit*) utilizado no pré-processamento de textos e que conta com várias bibliotecas.

Para carregar os arquivos de texto para teste no *Scikit-learn* foi utilizado o módulo *sklearn.datasets*, que, através da função *load_files*, carregou os arquivos de texto com suas

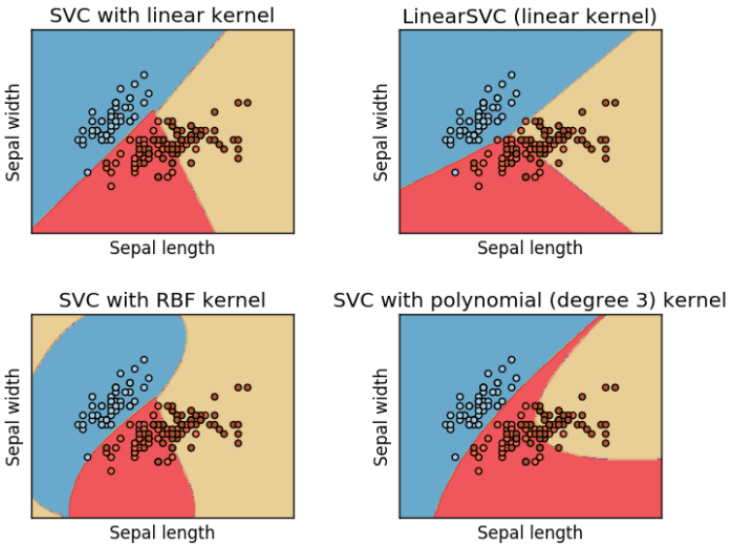
[4] http://scikit-learn.org/

[5] *SciPy* é um ecossistema baseado em *Python* de *software* de código aberto para a matemática, ciência e engenharia. https://www.scipy.org/

[6] *NumPy* é o pacote fundamental para computação científica com *Python*. http://www.numpy.org/

[7] http://www.nltk.org/

Figura 11 – Exemplo de aplicação de *kernel*

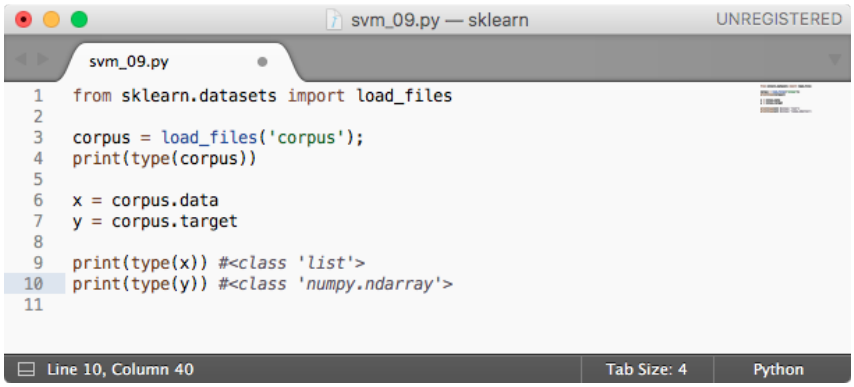


Fonte: PEDREGOSA *et al.* (2020)

respectivas classes conforme os nomes atribuídos aos diretórios que o conjunto continha.

Esses textos são carregados em um objeto de classe *Bunch*. Este objeto possui o atributo data que aponta para um objeto do tipo *list*, que contém todos os textos carregados. A classe *Bunch* também possui um atributo *target*, que é um objeto do tipo *numpy.ndarray* — um vetor de inteiros. Esse vetor é ordenado de acordo com os textos carregados no objeto *list* e contém as respectivas classes dos textos, de acordo com a Figura 12.

Figura 12 – Exemplo de carregamento do *corpus*

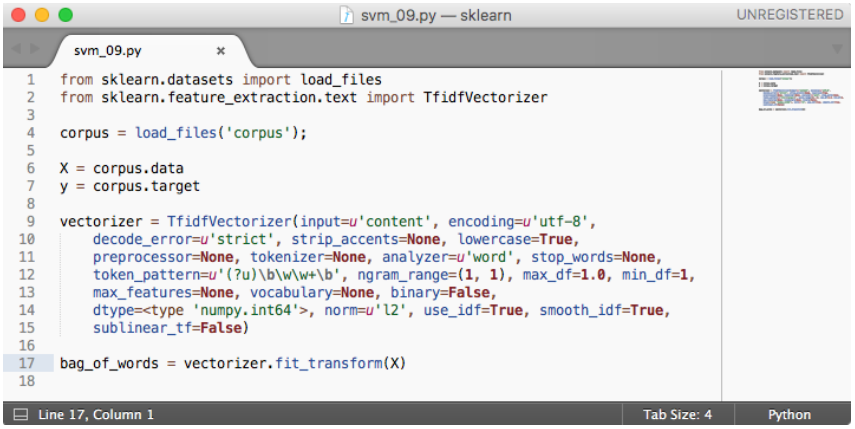


Fonte: o autor (2020)

Após carregados os dados através da função *load_files*, é possível executar a função *TfidfVectorizer* do módulo *feature_extraction.text*, com o objetivo de extrair as características vetoriais dos textos. Essa função aplica diversos parâmetros de pré-processamento dos textos, tais como: decodificar o formato de codificação dos termos; remover acentos dos termos para padronizar como não acentuados (pois os textos podem conter termos incorretamente não

acentuados); converter os termos para letra minúscula; remover termos *stopwords*; estabelecer parâmetros *ngram* e estabelecer as frequências máxima e mínima de termos, de acordo com a Figura 13.

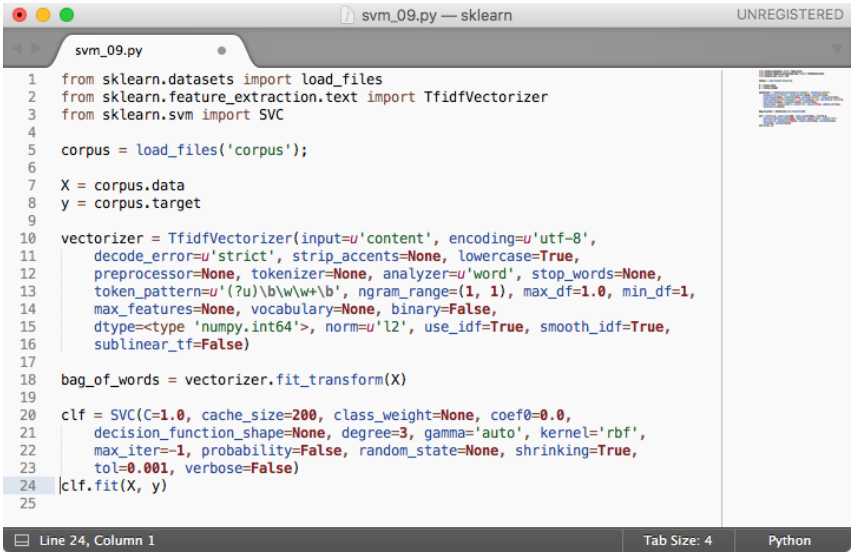
Figura 13 – Exemplo da função de vetorização



Fonte: o autor (2020)

Também é possível utilizar módulos especializados em pré-processamento de texto, como o *NLTK*, com o objetivo de entregar os textos já pré-processados para vetorização. Isso possibilita a aplicação da função com um número menor de parâmetros e, ao mesmo tempo, uma cópia dos dados tratados. Já a aplicação da função é indicada caso se queira obter as características vetoriais dos dados e não se queira realizar o pré-processamento dos textos. Com os dados vetorizados é possível aplicar o método *fit* da classe *SVC* que realiza o treinamento dos dados de acordo com a linha 24 da Figura 14.

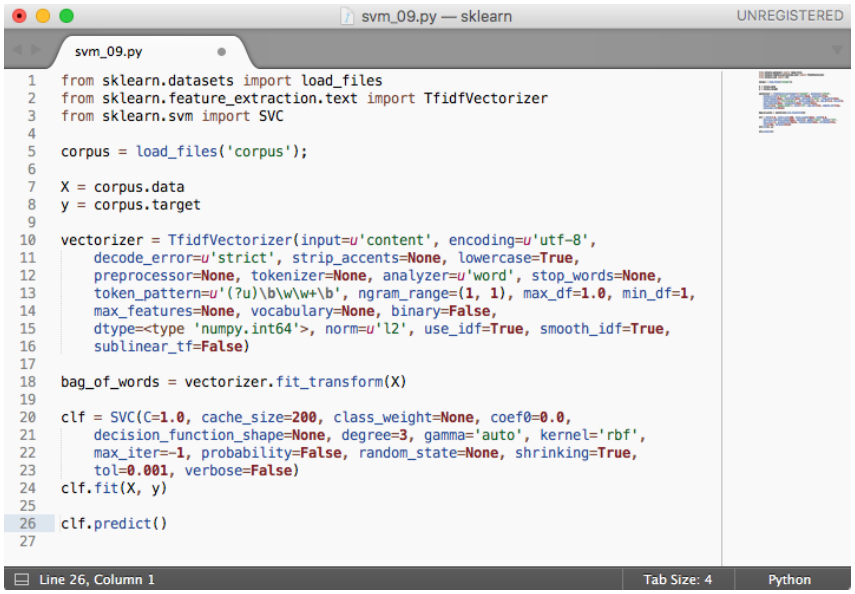
Figura 14 – Exemplo da função de treinamento



Fonte: o autor (2020)

A utilização da função de treinamento possui diversos parâmetros para melhorar o desempenho do algoritmo. A partir da realização da fase de treinamento, é possível efetuar os testes com novos dados, pela função *predict*, de acordo com a linha 26 da Figura 15, para verificar a eficiência do algoritmo de classificação automática. Desta forma, tem-se todo o processo de treinamento e testes realizados.

Figura 15 – Exemplo da função *predict*



Fonte: o autor (2020)

3.2.3 Definição da plataforma de aprendizagem de máquina

Nesta seção foram apresentadas algumas ferramentas para utilização com o algoritmo SVM, buscando-se destacar suas principais vantagens e desvantagens em relação à utilização neste trabalho. A ferramenta *Weka* demostrou possuir uma interface que contribui enormemente no sentido de facilitar a tarefa de parametrizações das funções e também ser uma ferramenta completa, com todos os recursos necessários para a execução deste trabalho. Posteriormente, foi analisado a ferramenta *Scikit-learn*, que também se mostrou extremamente poderosa, com vasto material de suporte em constante atualização, além de já ser adotada pela comunidade científica e grandes empresas que desenvolvem produtos na área de AM.

Neste trabalho optou-se pela utilização do *Scikit-learn* por dispor de amplo suporte, ser gratuito e sempre atualizado. Concluída aqui a escolha da ferramenta, a próxima seção apresenta outro aspecto importante do trabalho, que diz respeito a avaliação dos resultados.

3.3 SEGUNDA ETAPA: SELEÇÃO DO MÉTODO E MÉTRICAS

No contexto deste trabalho selecionou-se o algoritmo SVM e como métricas para avaliar a eficiência do sistema de classificação automática neste experimento foi utilizado um método de extração de atributos. Em problemas de classificação como a desenvolvida neste trabalho, é natural medir a performance em termos da taxa de erro. O classificador prediz a classe para cada instância. Se estiver correto, é contabilizado com um sucesso, senão, como um erro. A taxa de erros é proporcional aos erros obtidos em relação ao número total de instâncias e desta forma é medida a performance geral do classificador. Neste trabalho os dados já estão classificados e buscamos resultados para as melhores taxas de erro apenas na fase de treinamento, pois estas taxas de erro não representam um indicador para novos casos.

A partir da escolha do algoritmo de classificação deve-se escolher um critério de desempenho. Os resultados obtidos se dão por meio da matriz de confusão que indica a quantidade de instâncias classificadas como verdadeiros positivos (VP), verdadeiros negativos (VN), falsos positivos (FP) e falsos negativos (FN). Nas seções seguintes serão utilizadas as respectivas abreviações:

- RC indica a quantidade de textos reprovados corretamente
- AC indica a quantidade de textos aprovados corretamente
- RI indica a quantidade de textos reprovados incorretamente
- AI indica a quantidade de textos aprovados incorretamente

Através destes valores podem ser calculados os indicadores: precisão, *recall*, acurácia e *F-measure* como apresentados a seguir.

A precisão mede a relação entre a quantidade de textos positivos que o sistema classificou de maneira incorreta (falsos positivos). Quanto maior o número de RI, menor será a precisão. A precisão é definida por:

$$precisao = \frac{RC}{RC + RI}. \tag{3.1}$$

A desvantagem desta métrica é que ela desconsidera os textos que deveriam ter sido reprovados, mas foram aprovados. Para suprir esta perspectiva da avaliação, utiliza-se a métrica denominada *recall*.

O *recall* (ou sensibilidade, ou revocação) mede a quantidade de textos reprovados que o sistema aprovou. Esta métrica é definida por:

$$recall = \frac{RC}{RC + AI}. \tag{3.2}$$

A acurácia mede o quão efetivo o sistema é do ponto de vista da classificação (texto aprovado ou reprovado). A acurácia é uma métrica global para avaliação do desempenho do classificador, sendo definida por:

$$acuracia = \frac{RC + AC}{RC + AC + RI + AI}. \tag{3.3}$$

Para complementar, a medida denominada *F-measure* corresponde a uma média harmônica entre o *recall* e a precisão. Ela é definida por:

$$F_\beta = 1 + \beta^2 \frac{precisao \cdot recall}{\beta^2 \cdot precisao + recall}. \tag{3.4}$$

3.4 TERCEIRA ETAPA: SELEÇÃO DOS TEXTOS

A terceira etapa constitui na seleção de amostras textuais da área da saúde considerando problemas e doenças da coluna vertebral. Estas amostras foram coletadas por uma equipe de médicos especialistas da Universidade de Caxias do Sul e encontram-se em linguagem natural, na língua portuguesa brasileira, tendo sido rotuladas em termos de suas características, tais como: descrições do tratamento, benefícios do tratamento, consequências do tratamento, influência na qualidade de vida do paciente e riscos do tratamento, de acordo com o Quadro 1.

Quadro 1 – Distribuição dos fragmentos textuais segundo seus rótulos

Rótulos	Negativos	Positivos	Regulares
Descrições do tratamento	57	15	7
Benefícios do tratamento	43	14	7
Consequências do tratamento	21	8	6
Influência na qualidade de vida do paciente	10	0	1
Riscos do tratamento	10	8	7

Fonte: o autor (2020)

Dando continuidade aos trabalhos de (ABEL, 2016) e (SILVA, 2018) que investigavam dados textuais na área da saúde (Tabela 1), este trabalho tem como ênfase a utilização do algoritmo SVM no mesmo conjunto de dados. A partir de 2018, tendo em vista a diversidade das avaliações dos especialistas sobre um mesmo texto, segundo critérios variados de classificação, optou-se por aumentar a granularidade utilizando-se partes dos documentos de maneira a avaliar separadamente aspectos do texto. Os especialistas consideram como critério de qualidade do conteúdo textual como sendo: positivo, negativo ou regular. Os textos classificados como positivos são considerados bons exemplares, enquanto que os negativos e os regulares representam conteúdo parcialmente ou totalmente inadequados.

Tabela 1 – Algoritmos selecionados em cada categoria

Categoria	Algoritmo	Convergência com especialistas
Descrições do tratamento	SVM	97,47%
Benefícios do tratamento	Naïve Bayes	85,94%
Consequências do tratamento	SVM	88,57%
Influência na qualidade de vida do paciente	Naïve Bayes	100%
Riscos do tratamento	Naïve Bayes	80%

Fonte: MARTINS (2019)

3.5 CONSIDERAÇÕES FINAIS

Tendo em vista o planejamento apresentado, partiu-se para o desenvolvimento (implementação e testes). As ferramentas selecionadas para o desenvolvimento foram as seguintes: a biblioteca para aprendizado de máquina de código aberto para a linguagem *Python Scikit-learn*, a biblioteca para processamento de língua natural *NLTK*, o pacote para linguagem *Python Numpy* que suporta *arrays* e matrizes multidimensionais, além de uma larga coleção de funções matemáticas, a biblioteca *Matplotlib*[8] para criação de gráficos e visualizações de dados em geral. Além destas, outras poderão ser necessárias à medida em que o desenvolvimento do projeto avançar. O capítulo seguinte apresenta o desenvolvimento, a partir da quarta etapa do percurso metodológico, culminando na demonstração e análise dos resultados finais do uso do algoritmo SVM para análise de dados textuais.

[8] http://www.https://matplotlib.org/

4 DESENVOLVIMENTO E TESTES DA IMPLEMENTAÇÃO

O presente trabalho tem como objetivo implementar e testar uma solução baseada no algoritmo SVM para classificação de textos na área da saúde. Neste contexto, propõe-se a plataforma *Scikit-learn* e a linguagem *Python*, como ferramentas de desenvolvimento. Este capítulo descreve como transcorreu o trabalho, destacando-se os resultados e análises decorrentes dele.

4.1 RETOMADA DO PERCURSO METODOLÓGICO

Este trabalho foi desenvolvido nas seguintes etapas:

1. Escolha da plataforma de Aprendizagem de Máquina
2. Seleção do método e métricas
3. Seleção dos textos
4. Carga dos textos
5. Preparação dos textos
6. Testes dos algoritmos
7. Variação dos parâmetros
8. Tabulação dos resultados
9. Análise dos resultados com identificação do melhor caso
10. Justificativas da escolha

As etapas 1, 2 e 3, consideradas iniciais para o desenvolvimento proposto, são apresentadas em capítulo precedente uma vez que representam a base para as definições seguintes. Já as etapas de 4 a 10, que compõe a implementação propriamente dita, são descritas nas seções seguintes.

4.2 QUARTA ETAPA: CARGA DOS TEXTOS

Na quarta etapa utilizou-se a classe *CategorizedPlaintextCorpusReader* da biblioteca *NLTK* com a função de carregar textos simples, onde os parágrafos são divididos usando-se linhas em branco, já com suas respectivas classes atribuídas de acordo com o nome do diretório e dos arquivos que contém os dados textuais, conforme Figura 16.

Figura 16 – Classe *CategorizedPlaintextCorpusReader* da biblioteca *NLTK*

```
In [1]:
1  import nltk
2
3  from nltk.corpus.reader.plaintext import CategorizedPlaintextCorpusReader
4
5  DOC_PATTERN = r'(?!\.)[\w_\s]+/[\w\s\d\-]+\.txt'
6  CAT_PATTERN = r'([\w_\s]+)/.*'
7
8  corpus = CategorizedPlaintextCorpusReader(
9   'Corpus/beneficios', DOC_PATTERN, cat_pattern=CAT_PATTERN
10 )
```

Fonte: o autor (2020)

Uma vez que um *corpus* está estruturado e organizado de maneira persistente (arquivos) é possível a utilização de uma interface programática para ler, buscar, transmitir e filtrar documentos. Um *Corpus Reader* é instanciado passando-se um caminho para o diretório raiz que contém os arquivos e mais duas expressões regulares seguindo o padrão dos nomes dos documentos especificados no caminho sob a raiz do *corpus* de forma que haja uma letra, dígitos, espaços ou sublinhados, seguido pelo caractere "/" e uma ou mais letras, dígitos, espaços ou hífens seguidos pelo formato de arquivo .txt.

A Figura 17 ilustra o método utilizado para representar as classes automaticamente geradas a partir dos nomes dos diretórios contidos em cada categoria de dados textuais.

Figura 17 – Distribuição das classes

```
In [2]:
1  corpus.categories()
Out[2]: ['Negativo', 'Positivo', 'Regular']
```

Fonte: o autor (2020)

Após ter os dados textuais carregados em memória é possível realizar a contagem das classes de acordo com o exemplo da Figura 18, onde é demonstrado o somatório das classes no *corpus* dos Benefícios do Tratamento. Este somatório está disponibilizado no Quadro 1 para todas classes.

A partir deste ponto já temos os fragmentos textuais carregados em memória com suas respectivas categorias mapeadas possibilitando iniciar a próxima etapa de preparação dos dados.

4.3 QUINTA ETAPA: PREPARAÇÃO DOS TEXTOS

A preparação dos textos, também conhecida por pré-processamento, visa essencialmente remover os dados desnecessários. É nesta etapa que os dados textuais são organizados, limpos e padronizados através de diversos filtros para a posterior utilização pelo aprendizado de máquina.

Figura 18 – Fragmentos textuais dos benefícios segundo seus rótulos

```
In [3]:
for cat in corpus.categories():
    n = len([f for f in corpus.fileids(cat)])
    print("Para categoria", cat, ":", n)

Para categoria Negativo : 43
Para categoria Positivo : 14
Para categoria Regular : 7
```

Fonte: o autor (2020)

A remoção das *stopwords* (Figura 19) nada mais é que um processo de filtragem, onde palavras comuns e que não agregam informações relevantes durante a extração do conhecimento são removidas, restando apenas palavras essenciais. Estes *tokens* que compõe os *stopwords* são compostos por artigos, preposições, conjunções, números, símbolos, dentre outros. O *corpus* de *stopwords* da biblioteca de código aberto para língua portuguesa *NLTK* é composto por 204 *tokens*.

Figura 19 – Remoção das *Stopwords*

```
In [4]:
import string
stop = nltk.corpus.stopwords.words('portuguese')
documents = [([p for p in corpus.words(i) if p.lower(
) not in stop and p.lower() not in string.punctuation],
              i.split('/')[0]) for i in corpus.fileids()]
```

Fonte: o autor (2020)

Um próximo filtro importante é o lematizador (do inglês, *stemming*), que representa o processo de fundir as formas variantes de uma palavra em uma representação comum, o radical. O algoritmo lematizador RSLP (Removedor de Sufixos da Língua Portuguesa) é responsável pela remoção de sufixo das palavras, tais como, plural, gerúndio, prefixos, sufixos, gênero e número (Figura 20). Por exemplo, as palavras: apresentação, apresentado, apresentando podem ser reduzidas a uma raiz comum "presente". Parte-se do pressuposto de que colocar uma consulta com o termo apresentação implica um interesse por documentos que contenham as palavras apresentação e apresentado.

Figura 20 – A utilização do algoritmo lematizador RSLP

```
In [5]:
from nltk.stem import RSLPStemmer
rslps = RSLPStemmer()  # ('portuguese')
documents = [([rslps.stem(p) for p in d[0]], d[1]) for d in documents]
```

Fonte: o autor (2020)

Este processo contempla também uma função utilizada para a eliminação de hifens,

pontuação, acentos e transformação de caracteres maiúsculas em minúsculas (Figura 21). Primeiramente os *strings* são normalizados ou padronizados (*Canonication*) de maneira que cada caractere seja decomposto por seu equivalente canônico através do algoritmo NFD (do inglês, *Normalization Form Canonical Decomposition*) e posteriormente recombinado se o caractere não possui uma classe de combinação canônica atribuída.

Figura 21 – Remoção dos acentos

```
In [6]:
1 import unicodedata
2 def remove_acentos(string: str) -> str:
3     normalized = unicodedata.normalize('NFD', string)
4     return ''.join([l for l in normalized if not unicodedata.combining(l)]
5                    ).casefold()
6 documents = [([remove_acentos(p) for p in d[0]], d[1]) for d in documents]
```

Fonte: o autor (2020)

É possível observar uma redução na contagem dos *tokens* de 101 iniciais para 56 finais, posteriormente a utilização dos filtros acima mencionados (Figura 22).

Figura 22 – Documento processado e reduzido

```
In [7]:
1 c = corpus.fileids()[-1]
2 d = documents[-1][0]
3 print('Arquivo: ', c)
4 print(corpus.words(c)[:len(corpus.words(c))], len(corpus.words(c)))
5 print('Pré-processado:')
6 print(d, len(d))
```

```
Arquivo:  Regular/017.txt
['A', 'microcirurgia', 'possibilita', 'incisões', 'menores', 'e', 'recuperaçã
o', 'mais', 'rápida', '.', 'Diminui', 'o', 'sangramento', ',', 'o', 'período',
'de', 'hospitalização', 'e', 'de', 'recuperação', 'Proporciona', 'menor', 'mobi
lização', 'das', 'estruturas', 'nervosas', 'minimizando', 'a', 'reação', 'infla
matória', 'Trata', '-', 'se', 'de', 'um', 'procedimento', 'simples', 'e', 'segu
ro', 'que', 'pode', 'trazer', 'grande', 'alívio', 'para', 'alguns', 'paciente
s', 'selecionados', 'pelo', 'neurocirurgião', ',', 'principalmente', 'aqueles',
'que', 'apresentam', 'dor', 'lombar', 'intensa', 'O', 'alívio', 'é', 'imediat
o', 'Entre', 'as', 'vantagens', 'da', 'microcirurgia', 'temos', ':', 'incisõe
s', 'menores', 'e', 'mais', 'precisas', ',', 'sangramento', 'menor', ',', 'meno
r', 'movimentação', 'das', 'estruturas', 'nervosas', 'adjacentes', 'à', 'hérni
a', '.', 'Em', 'consequência', ',', 'a', 'recuperação', 'é', 'mais', 'rápida',
'e', 'a', 'hospitalização', 'mais', 'curta'] 101
Pré-processado:
['microcirurg', 'possibilit', 'incis', 'men', 'recuper', 'rapid', 'diminu', 'sa
ngr', 'period', 'hospit', 'recuper', 'proporc', 'men', 'mobil', 'estrut', 'ner
v', 'minimiz', 'reac', 'inflamator', 'trat', 'proced', 'simpl', 'segur', 'pod',
'traz', 'grand', 'alivi', 'algum', 'paci', 'selecion', 'neurocirurg', 'princi
p', 'apresent', 'dor', 'lomb', 'intens', 'alivi', 'imediat', 'vantag', 'microci
rurg', 'incis', 'men', 'precis', 'sangr', 'men', 'men', 'moviment', 'estrut',
'nerv', 'adjac', 'hern', 'consequ', 'recuper', 'rapid', 'hospit', 'curt'] 56
```

Fonte: o autor (2020)

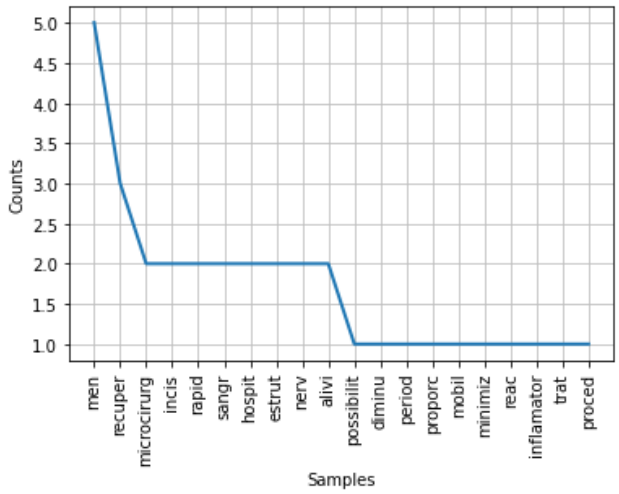
Após os dados terem sido tratados por todos os filtros de pré-processamento, faz-se necessária a aplicação de um conceito de *Bag of Words* que nada mais é que a representação dos

textos em um espaço vetorial. Este modelo de espaço vetorial é a representação matemática dos textos não estruturados em vetores numéricos.

Desta forma, cada documento do *corpus* pré-processado é convertido em um vetor numérico capaz de ser utilizado junto ao algoritmo SVM em que cada dimensão é representada por uma palavra específica a partir da frequência com que ela ocorre nos documentos (Figura 23).

Figura 23 – Frequência das palavras

```
In [8]:
frequency_dist = nltk.FreqDist(d)
frequency_dist.plot(20,cumulative=False)
```




```
Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x173246e6310>
```

Fonte: o autor (2020)

Essa conversão utiliza o módulo *Pipeline* que encadeia uma sequência fixa de funções fornecendo uma conveniência para chamada de diversos estimadores de uma só vez (Figura 24).

Para converter o texto dos documentos em uma matriz de contagem de *tokens* utilizou-se a função *CountVectorizer* e para transformar esta matriz em uma representação normalizada pela frequência de termos e pela frequência de termos inversa utilizou-se a função *TfidfTransformer*.

Figura 24 – Instanciação do *Pipeline*

```
In [9]:
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import (
    CountVectorizer, TfidfTransformer)
pipeline = Pipeline([
    ('vect', CountVectorizer(max_df = 0.9, min_df = 0.1)),
    ('tfidf', TfidfTransformer()),
])
```

Fonte: o autor (2020)

Para a função *CountVectorizer* utilizou-se os parâmetros *max_df* e *min_df* de maneira a desprezar os termos que possuem uma frequência estritamente superior, no primeiro caso, ou inferior no segundo, aos limites fornecidos. A função *fit_transform* treina e transforma o modelo com o estimador final produzindo os parâmetro X e o y (*target*) que representam os dois parâmetros referentes aos dados textuais a serem utilizados em conjunto com o algoritmo SVM (Figura 25).

Figura 25 – Ajusta e transforma o modelo com o estimador final

```
In [10]:
1 documents_tmp = [' '.join(d) for d in documents]
2 X = pipeline.fit_transform(documents_tmp).todense()
```

Fonte: o autor (2020)

Tendo os dados pré-processados e vetorizados já é possível iniciar os testes de classificação com o algoritmo SVM conforme descrito na seção subsequente.

4.4 SEXTA ETAPA: TESTES DOS ALGORITMOS

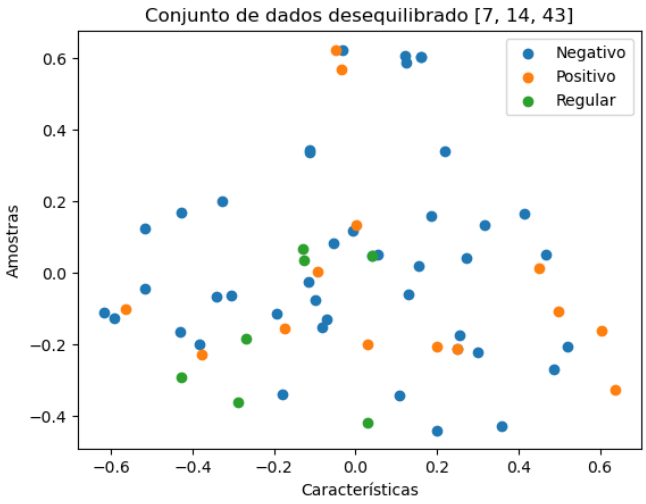
Para a finalidade de executar os testes deste trabalho optou-se por utilizar os *datasets* que tratam da descrição do tratamento, dos benefícios do tratamento e das consequências do tratamento que são os que possuem uma maior quantidade de fragmentos textuais. Além disto, no decorrer dos testes, percebeu-se que havia uma distribuição de classes desigual e tecnicamente desequilibrada em cada uma das categorias mencionadas (Quadro 1) prejudicando o desempenho do classificador. Este desequilíbrio de classes ocorre quando o número de exemplos representando uma classe é muito menor do que outras classes. Portanto, uma ou mais classes podem estar sub-representadas no conjunto de dados.

A Figura 26 apresenta um gráfico de dispersão (do inglês, *Scatter Plot*) utilizando o *dataset* que compõe a categoria dos benefícios do tratamento com a finalidade de ilustrar o desequilíbrio entre as classes. Ele é um tipo de gráfico ou diagrama matemático que se vale de coordenadas cartesianas para exibir valores para duas ou mais variáveis de um conjunto de dados. Os pontos em azul no gráfico, indicam as instâncias classificadas como negativas e a sua contagem é de 43 ocorrências. Os pontos em laranja são instâncias classificadas como positivas e a sua contagem é de 14 ocorrências. Por fim, os pontos em verde indicam instâncias classificadas como regulares e a sua contagem é de 7 ocorrências. Observa-se que há um desbalanceamento entre as classes.

Em detrimento deste desequilíbrio de classes optou-se por ajustar a distribuição do conjunto de dados por meio de um pacote chamado *Imbalanced-Learn*[1] que oferece uma série

[1] https://github.com/scikit-learn-contrib/imbalanced-learn

Figura 26 – Exemplo de *dataset* desbalanceado

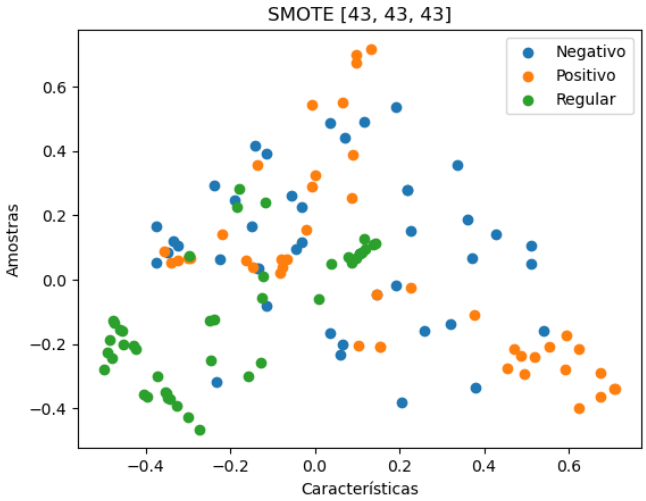


Fonte: o autor (2020)

de técnicas de reamostragem comumente utilizadas, tais como, SMOTE (do inglês, *Synthetic Minority Over-sampling Technique*) e ADASYN (do inglês, *Adaptive Synthetic*).

O algoritmo SMOTE (Figura 27) realiza uma abordagem de sobreamostragem para reequilibrar o conjunto de treinamento original. Ao invés de aplicar uma replicação simples das instâncias de classe minoritária, a ideia principal do SMOTE é apresentar exemplos sintéticos. Esses novos exemplos são criados por interpolação entre várias instâncias positivas que estão juntas (FERNÁNDEZ *et al.*, 2018).

Figura 27 – Categoria dos benefícios do tratamento após balanceada (SMOTE)



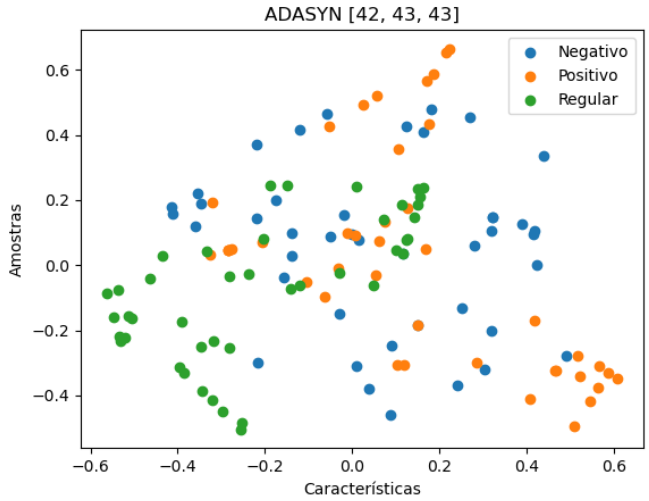
Fonte: o autor (2020)

Já o algoritmo ADASYN (Figura 28) é baseado na ideia de gerar exemplos minoritários

de forma adaptativa de acordo com suas distribuições: mais dados sintéticos são gerados para amostras de classes minoritárias que são mais difíceis de aprender em comparação com amostras de minorias que são mais fáceis de aprender. O método ADASYN pode não apenas reduzir o viés de aprendizagem introduzido pela distribuição de dados de desequilíbrio original, mas também pode alterar de forma adaptativa o limite de decisão para se concentrar nas amostras difíceis de aprender (CHAWLA *et al.*, 2002).

Neste trabalho são testados dois modelos de balanceamento de classes, o SMOTE e o ADASYN, sendo que optou-se por utilizar o SMOTE devido aos melhores resultados apresentados com relação a distribuição dos dados.

Figura 28 – Categoria dos benefícios do tratamento após balanceada (ADASYN)



Fonte: o autor (2020)

Após o balanceamento, cada categoria passou a ter um número de instâncias equivalentes por classe de acordo com o Quadro 2.

Quadro 2 – Distribuição dos fragmentos textuais segundo seus rótulos após balanceamento

Rótulos	Negativos	Positivos	Regulares	Total
Descrições do tratamento	57	58	55	170
Benefícios do tratamento	43	42	43	128
Consequências do tratamento	21	21	23	65

Fonte: o autor (2020)

Esse aumento no número de casos das classes minoritárias nos conjuntos de dados, leva uma melhora no desempenho de classificação oportunizando iniciar os testes dos parâmetros do algoritmo SVM para detectar os melhores resultados.

4.5 SÉTIMA ETAPA: VARIAÇÃO DOS PARÂMETROS

Foram feitos testes para identificar a melhor parametrização e avaliar a eficiência do sistema de classificação automática. Fez-se necessário a variação de alguns dos parâmetros do algoritmo SVM, tais como: o C (de custo, ou termo de regularização - vide Seção 2.2), o tipo do *kernel* (linear, poly e rbf), o *degree* (grau da função polinomial quando o *kernel* for do tipo poly) e o gamma que serve para o definir o coeficiente de *kernel* (para rbf, poly e sigmoid).

Com a finalidade de definir um sistema de teste, empregou-se o utilitário *train_test_split* que divide o conjunto de dados em subconjuntos de treinamento e teste. Neste caso, foi utilizado 80% do *dataset* para treinamento e os 20% remanescentes para teste (Figura 29).

Figura 29 – Utilização da função *train_test_split*

```
In [12]:
1 from sklearn.model_selection import train_test_split
2
3 X_train, X_test, y_train, y_test = train_test_split(
4     X, y, test_size=0.4, random_state=0)
```

Fonte: o autor (2020)

A relação dos resultados coletados durante a fase de testes para avaliar a necessidade de balancear as classes são apresentadas, segundo os parâmetros que resultaram no melhor desempenho de classificação por tipo de *kernel* e por categoria (Tabela 2).

Tabela 2 – Resultados obtidos nos testes antes do balanceamento

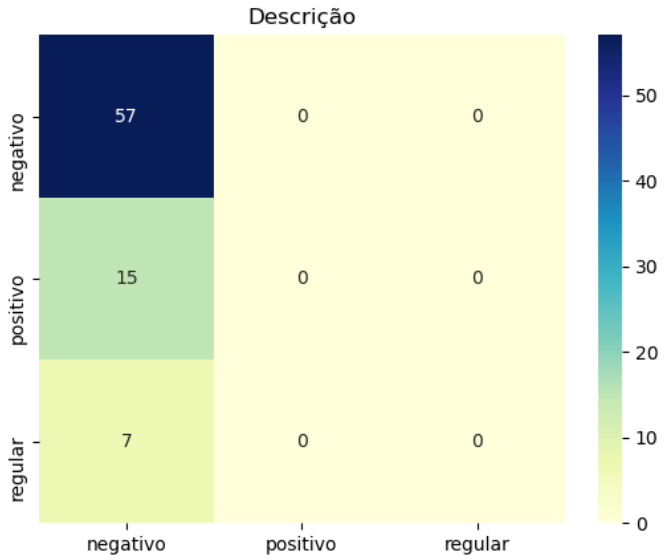
Categoria	*kernel*	*C*	Gamma	*Degree*	*Score*
Descrições do tratamento	linear	0.1	0.1	-	56.25%
	rbf	0.1	0.1	-	56.25%
	poly	0.1	0.1	0	56.25%
Benefícios do tratamento	linear	0.1	0.1	-	69.23%
	rbf	10	0.1	-	76.92%
	poly	0.1	10	2	76.92%
Consequências do tratamento	linear	0.1	0.1	-	57.14%
	rbf	0.1	0.1	-	57.14%
	poly	0.1	0.1	0	57.14%

Fonte: o autor (2020)

A partir desses resultados pode-se determinar que, em virtude das amostras da classe negativa serem em maior número que as demais classes, o modelo estava convergindo para resultados que apresentam um elevado número de falsos negativos, de acordo com a matriz de confusão da categoria "descrição do tratamento", conforme ilustrado na Figura 30.

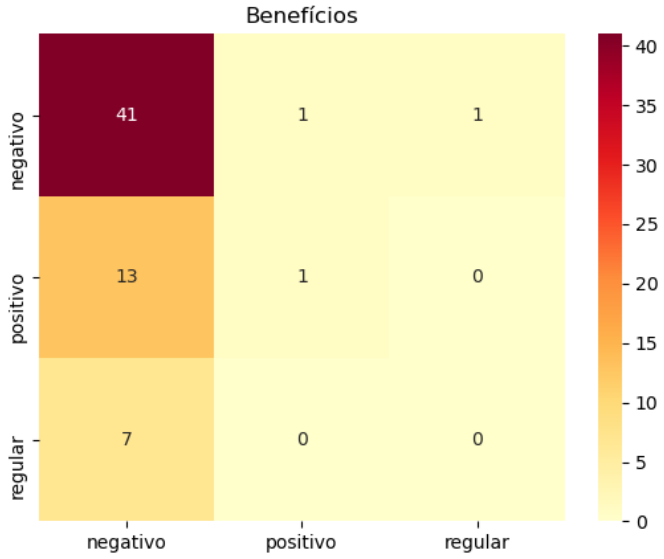
A categoria dos benefícios do tratamento também apresenta uma convergência na classificação das instâncias classificadas como negativas. Além disso, pode-se confirmar esta análise através da matrix de confusão resultante dos testes (Figura 31).

Figura 30 – Matriz de confusão da categoria das descrições do tratamento desbalanceada



Fonte: o autor (2020)

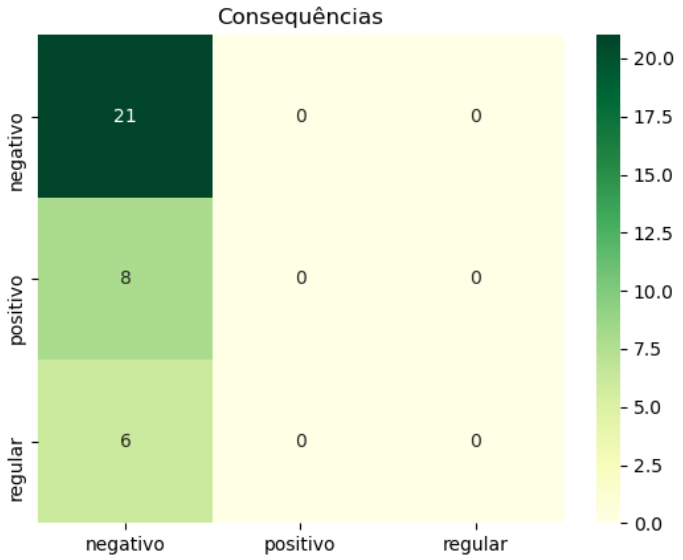
Figura 31 – Matriz de confusão da categoria dos benefícios do tratamento desbalanceada



Fonte: o autor (2020)

Do mesmo modo, a categoria das descrições do tratamento apresenta as mesmas características de acordo com a matriz de confusão da Figura 32. A partir da aplicação do modelo de balanceamento de classes SMOTE, que resultou em nova distribuição de instâncias por classe, foram realizados novos testes, obtendo-se um desempenho melhor, conforme observado na Tabela 3. Para os testes com as classes balanceadas, utilizou-se o método de *cross validation*.

Figura 32 – Matriz de confusão da categoria das consequências do tratamento desbalanceada



Fonte: o autor (2020)

Tabela 3 – Resultados obtidos nos testes após o balanceamento

Categoria	*kernel*	*C*	Gamma	*Degree*	*Score*
Descrições do tratamento	linear	10	0.1	-	91.17%
	rbf	1	1	-	97.05%
	poly	0.1	10	4	100.0%
Benefícios do tratamento	linear	10	0.1	-	65.38%
	rbf	10	1	-	84.61%
	poly	1	1	6	80.76%
Consequências do tratamento	linear	10	0.1	-	100.0%
	rbf	1	10	-	100.0%
	poly	0.1	10	2	100.0%

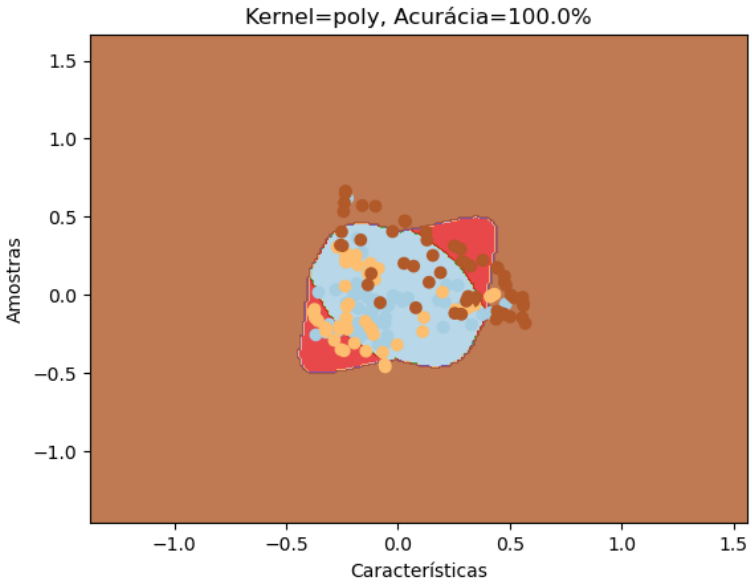
Fonte: o autor (2020)

Foram utilizados 3 diferentes tipos de *kernel* nos testes deste trabalho. Um *kernel* representa um conjunto de funções matemáticas e tem como função receber dados e transformá-los em um formato de acordo. Essas funções podem ser de diferentes tipos, como as lineares, as polinomiais e as rbf, que são funções de base radial. A classificação da categoria das descrições do tratamento alcançou o melhor desempenho ao utilizar o *kernel* do tipo polinomial, com uma acurácia de 100.0%, como podemos observar na Figura 33, que foi dividida em 3 regiões de cores diferentes, representando assim as 3 classes através de uma cor para cada uma delas.

Da mesma forma, a classificação da categoria dos benefícios do tratamento alcançou o melhor desempenho ao utilizar o *kernel* do tipo rbf, com uma acurácia de 85.0% (Figura 34).

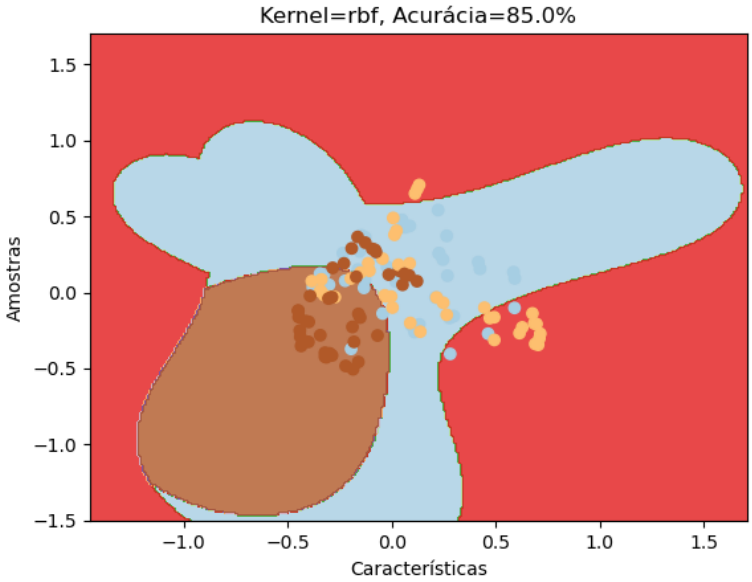
A classificação da categoria das consequências do tratamento atingiu a acurácia de

Figura 33 – *kernel* poly com melhor classificação para categoria das descrições do tratamento



Fonte: o autor (2020)

Figura 34 – *kernel* rbf com melhor classificação para categoria dos benefícios do tratamento

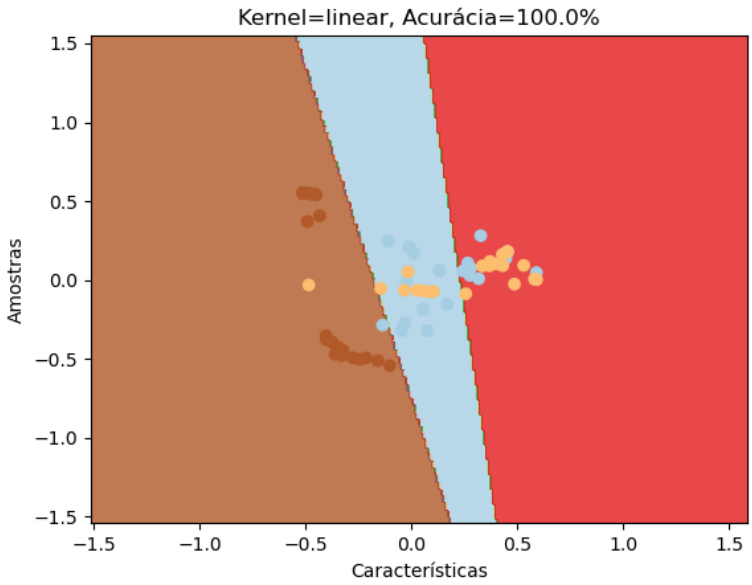


Fonte: o autor (2020)

100.0%, com a utilização de todos os tipos de *kernel*, já a Figura 35 ilustra a utilização de *kernel* do tipo linear.

Tendo realizado o balanceamento das classes e testes de parametrização, passou-se a fazer uso da validação cruzada através da função *cross_val_predict*. Esta função possui como objetivo avaliar a capacidade de generalização do modelo do SVM e recebe um parâmetro chamado “cv”

Figura 35 – *kernel* linear com melhor classificação para categoria das consequências do tratamento



Fonte: o autor (2020)

Tabela 4 – Métricas da classificação das categorias testadas

Categoria	Classe	Precisão	*Recall*	*F1-score*
Descrições do tratamento	negativo	0.91	0.88	0.89
	positivo	0.96	0.93	0.95
	regular	0.92	0.98	0.95
Benefícios do tratamento	negativo	0.92	0.79	0.85
	positivo	0.87	0.93	0.90
	regular	0.93	1.00	0.97
Consequências do tratamento	negativo	0.80	0.95	0.87
	positivo	0.94	0.76	0.84
	regular	1.00	1.00	1.00

Fonte: o autor (2020)

utilizado para estipular quantas partições dos dados são separados para treinamento e quantas para teste. No caso deste trabalho, as categorias foram separadas em 10 partições. Tendo definido os método e adequado o balanceamento das classes, a próxima etapa consiste em demonstrá-los.

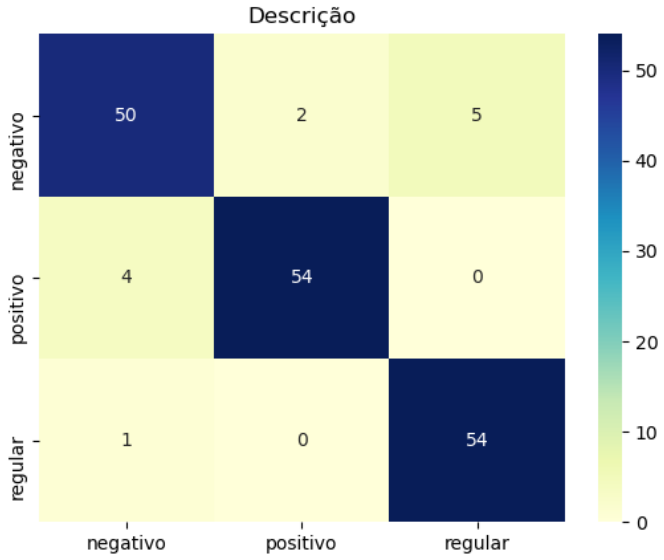
4.6 OITAVA ETAPA: TABULAÇÃO DOS RESULTADOS

A partir das etapas de teste e treinamento pode-se obter as métricas de classificação do modelo SVM a partir da função *accuracy_score*, que constrói um relatório de performance demonstrado na Tabela 4. Conforme a Seção 3.3, as métricas apresentadas são precisão, *recall* e *F1-score*.

A matriz de confusão da Figura 36, permite a visualização do desempenho de classificação do modelo SVM. Ela é utilizada para representar a frequência de classificação para a categoria das descrições do tratamento. Pode-se observar os erros de classificação, tais como:

- sete textos negativos foram classificados, sendo dois como positivos e cinco como regulares;
- quatro textos positivos foram classificados como negativos;
- um texto regular foi classificado como negativo.

Figura 36 – Matriz de confusão da categoria das descrições do tratamento balanceada



Fonte: o autor (2020)

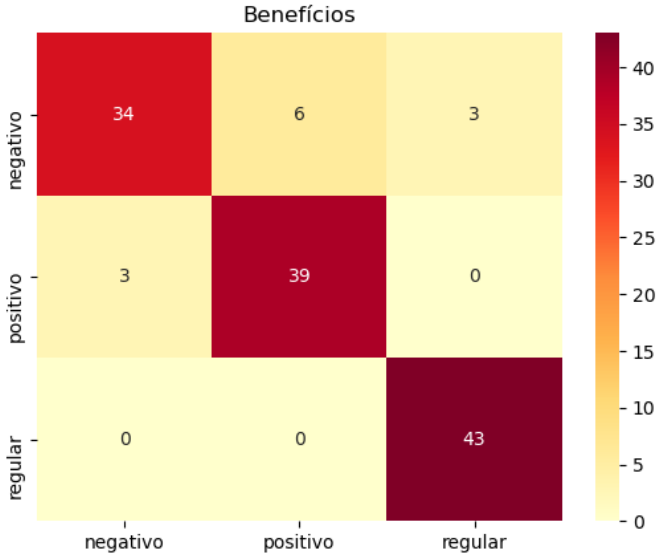
A partir da Figura 37 pode-se observar os erros de classificação, tais como:

- nove textos negativos foram classificados, sendo seis como positivos e três textos como negativos;
- três textos positivos foram classificados como negativos.

A Figura 38, também demonstra os seguintes erros de classificação:

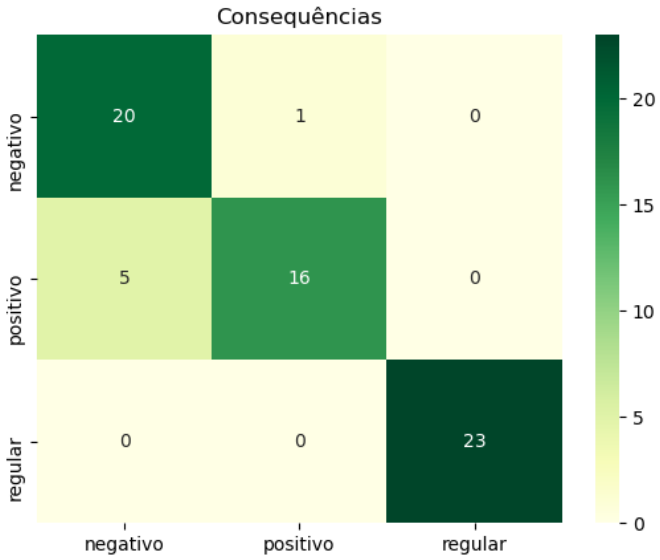
- um texto negativo foi classificado como positivo;
- cinco textos positivos foram classificados como negativos.

Figura 37 – Matriz de confusão da categoria dos benefícios do tratamento balanceada



Fonte: o autor (2020)

Figura 38 – Matriz de confusão da categoria das consequências do tratamento balanceada



Fonte: o autor (2020)

4.7 NONA ETAPA: ANÁLISE DOS RESULTADOS E IDENTIFICAÇÃO DO MELHOR CASO

Baseando-se nos resultados obtidos durante a fase dos testes (Tabela 3), observa-se que cada categoria obteve um melhor desempenho de acordo com um tipo de *kernel* específico,

com exceção da categoria das consequências do tratamento, que se saiu bem para cada tipo de *kernel*. Para obter-se acurácia de 93.0%, durante a fase dos testes de classificação da categoria das descrições do tratamento, foi utilizado um *kernel* do tipo polinomial. Já para a categoria benefícios do tratamento, em que obteve-se 91.0% de acurácia, foi utilizado um *kernel* do tipo rbf. A categoria das consequências do tratamento obteve acurácia de 100.0% para todos os tipos de *kernel*.

4.8 DÉCIMA ETAPA: JUSTIFICATIVAS DA ESCOLHA

Durante os primeiros testes do percurso metodológico e mesmo com a melhor parametrização, os resultados apresentaram uma performance muito aquém da proposta inicial, confirmando o que a literatura afirma em relação ao grande volume de dados necessários para o treinamento de um modelo tipo SVM. Foi a partir deste ponto que o balanceamento das classes precisou ser introduzido ao trabalho (*oversampling*), abrindo a possibilidade de utilizar bases de dados reduzidas e corrigir essas pequenas deturpações.

A partir do balanceamento das classes em todas as categorias, passou-se a obter resultados satisfatórios demonstrando a possibilidade de classificação supervisionada para os dados textuais testados. Tendo em vista os testes realizados, onde tratou-se de um problema do tipo multiclasse (três classes, neste caso), conforme já era esperado, o *kernel* linear demonstrou baixa eficácia nos três cenários, pois o *kernel* linear é mais utilizado em problemas binários.

5 CONCLUSÃO

Este capítulo destina-se a apresentar a síntese das atividades desenvolvidas neste trabalho de conclusão, descrevendo também, as contribuições deste trabalho e as ideias para trabalhos futuros.

5.1 SÍNTESE DO TRABALHO

O problema de classificação automática de documentos é de grande importância prática, dada a crescente expansão do volume de dados gerados e armazenados na Web. A classificação automática de textos é o processo em que dados textuais construídos em linguagem natural, são submetidos a categorias predefinidas, considerando-se o seu conteúdo.

A Aprendizagem de Máquina é uma área da Inteligência Artificial responsável por desenvolver modelos computacionais de aprendizagem, para fins de classificação automática de textos. Dentre eles, destaca-se o algoritmo de Máquinas de Suporte Vetorial, amplamente utilizado em conjuntos de dados textuais.

O objetivo deste trabalho foi aplicar o SVM na construção de um modelo de classificação textual, usando amostras textuais da área da saúde coletadas por uma equipe de médicos especialistas da Universidade de Caxias do Sul. Estes dados textuais foram divididos em cinco categorias, das quais três foram suficientes para execução deste trabalho. Também, cada categoria foi dividida em três classes definindo um problema multiclasses, e não binário, o que é mais desafiador. Para que esse objetivo fosse alcançado, foram apresentados estudos sobre a teoria do aprendizado estatístico e o treinamento de uma SVM.

Com base no conhecimento adquirido, tornou-se possível identificar os principais parâmetros do algoritmo de classificação baseado em SVM, que são o *kernel*, o *C*, o *degree* e o gamma. Todos esses parâmetros se demonstraram relevantes ao longo dos testes. A partir dos primeiros testes de treinamento do SVM percebeu-se que havia uma distribuição de classes desigual prejudicando o desempenho do classificador. Fez-se necessário adicionar uma nova ferramenta ao trabalho para rebalancear as classes, chamada de SMOTE, que gera novas instâncias para as classes minoritárias de maneira aleatória ou não determinística.

Após o balanceamento, cada categoria passou a dispor de um número equivalente de instâncias por classe. Ao aplicar e testar, obteve-se sucesso tanto na aplicação, como nos testes utilizando a validação cruzada. Como etapa final analisou-se os resultados obtidos, listando as vantagens e limitações. Os resultados colhidos foram bastante satisfatórios como comprovação da eficácia de todo processo deste trabalho.

O algoritmo SVM demonstrou que, a partir de bases de dados menores, é possível formar

um modelo preditivo com qualidade suficiente para novos casos. A única observação para que isso aconteça é de que todas categorias devam estar balanceadas para que não ocorra nenhum tipo de convergência para as classes maiores.

É importante destacar a maneira eficiente e robusta em que a linguagem *Python* e todas as bibliotecas utilizadas neste trabalho se demonstraram. Através de um código simples e prático, porém muito expressivo, pode-se representar todas as funcionalidades requeridas ao longo do trabalho.

5.2 CONTRIBUIÇÕES DO TRABALHO

O campo de pesquisa denominado *Machine Learning* está em constante desenvolvimento e quando aplicado a análise de dados textuais da área da saúde traz inovações importantes no que diz respeito a evolução das ferramentas que possibilitam automatizar o processo de classificação de textos e artigos escritos na língua Portuguesa acerca de sua qualidade. Nesse sentido, este trabalho buscou enriquecer este campo de pesquisa com mais conhecimento aplicado.

A partir dos temas abordados neste trabalho será possível contar como uma fonte de referência para futuros projetos e estudos relacionados. Ficam disponíveis todas etapas de uma aplicação de *Machine Learning* para classificação de dados textuais, inclusive a etapa imprevista em que tratou-se do balanceamento das classes, para que outros alunos ou pessoas interessadas possam seguir este mesmo processo e refletir.

5.3 TRABALHOS FUTUROS

Para trabalhos futuros a ideia é testar novos algoritmos de aprendizagem de máquina, tais como, Árvore de Decisão e Redes Neurais. Também, sugere-se aprofundar ainda mais o entendimento de outros conceitos relacionados a área de *Machine Learning*, como por exemplo, a aprendizagem não supervisionada em contrapartida a aplicada neste trabalho.

REFERÊNCIAS

ABEL, F. A. **Análise textual automática: apreensibilidade e qualidade da informação na área da saúde**. 2016. Monografia (Trabalho de Conclusão de Curso), Universidade de Caxias do Sul, Curso de Bacharelado em Ciência da Computação, Caxias do Sul - RS, Brasil.

AL., D.-R. C. et. Support vector machine soft margin classifiers: error analysis. **Journal of Machine Learning Research**, JMLR Editorial, v. 55, n. 5, p. 1143–1175, 2004.

AL., M. C. A. et. Use of a support vector machine for keratoconus and subclinical keratoconus detection by topographic and tomographic data. **Revista Brasileira de Oftamologia**, v. 74, n. 6, p. 383–385, 2015.

CHANG, C.-C.; LIN, C.-J. **A Library for Support Vector Machines**. 2020. (Acessado: 30.08.2020). Disponível em: <https://www.csie.ntu.edu.tw/~cjlin/libsvm/>.

CHAWLA, N. V. *et al.* Smote: synthetic minority over-sampling technique. **Journal of Artificial Intelligence Research**, AI Access Foundation and Morgan Kaufmann Publishers, n. 16, p. 321–357, 2002.

CHOW, C. On optimum recognition error and reject tradeoff. **IEEE Journals & Magazines**, v. 16, p. 41–46, 1970.

CORTES, C.; VAPNIK, V. Support vector networks: machine learning. Editora Springer, v. 20, n. 3, p. 273–297, 1995.

FACELI, K. *et al.* **Inteligência artificial: uma abordagem de aprendizado de máquina**. [S.l.]: LTC, 2011.

FERNÁNDEZ, A. *et al.* **Learning from Imbalanced Data Sets**. [S.l.]: Springer Nature Switzerland AG, 2018. 98–101 p.

FISHER, R. A. The use multiple measurements in toxonomic problems. **Annals of Eugenics**, Annals of Eugenics, v. 7, p. 179–188, 1936.

FUMERA, G.; ROLI, F. **Support vector machines with embedded reject option: svm '02: Proceedings of the First International Workshop on Pattern Recognition with Support Vector Machines**. [S.l.]: Springer-Verlag, 2002. 68–82 p.

HAND, D. **Construction and assessment of classification rules**. [S.l.]: John Wiley & Sons, 1997.

HAYKIN, S. **Redes neurais: princípios e prática**. [S.l.]: Bookman, 2001.

HIRASAWA, H. *et al.* Evaluation of various machine learning methods to predict vision-related quality of life from visual field data and visual acuity in patients with glaucoma. **British Journal of Ophthalmology**, v. 98, p. 1230–1235, 2014.

HOTHO, A.; NÜRBBERGER, A.; PAASS, G. A brief survey of text mining. **GLDV Journal for Computational Linguistics and Language Technology**, v. 20, p. 19–62, 2005.

KOJI, K. **A brain-computer interface for potential nonverbal facial communication based on eeg signals related to specific emotions**. [S.l.]: In Frontiers in Neuroscience, 2014. v. 8. 244 p.

KURESHI, N.; ABIDI, S. S. R.; BLOUIN, C. **A predictive model for personalized therapeutic interventions in non-small cell lung cancer**. [S.l.]: In IEEE Journal of Biomedical and Health Informatics, 2014. v. 20. 424–431 p.

LORENA, A. C.; CARVALHO, A. **Uma introdução às support vector machines**. [S.l.]: Revista de Informática Teórica e Aplicada, 2007. v. 14. 43–67 p.

MARTINS, E. R. **Princípios e aplicações da computação no Brasil**. [S.l.]: Atena Editora, 2019. v. 2. 39–53 p.

MINSKY, M.; PAPERT, S. **Perceptrons: an introduction to computational geometry**. [S.l.]: MIT Press, 1969.

MITCHELL, T. **Machine learning**. [S.l.]: McGraw-Hill International Editions, 1997.

PEDREGOSA, F. *et al.* **Scikit-learn: Machine Learning in Python**. 2020. (Acessado: 30.08.2020). Disponível em: <http://scikit-learn.org/>.

PONTIL, M.; VERRY, A. **Properties of support vector machines. Mit Press Journals. In Neural Computation**. [S.l.]: Mit Press Journals. In Neural Computation, 1998. v. 10. 955–974 p.

REZENDE, S. de O. **Sistemas inteligentes: fundamentos e aplicações**. [S.l.]: Manole, 2003.

SILVA, C. R. da. **Avaliação da apreensibilidade e da qualidade da imformação em saúde com o software Spinefind**. 2018. Monografia (Trabalho de Conclusão de Curso), Universidade de Caxias do Sul, Curso de Bacharelado em Ciência da Computação, Caxias do Sul - RS, Brasil.

SMOLA, A.; SCHÖLKOPF, B. **Learning with kernels: support vector machines, regularization, optimization and beyond**. [S.l.]: MIT Press, 2002.

UJJWAL, B.; SK, M.; SWAPAN, K. P. **Unconstrained bangla online handwriting recognition based on mlp and svm. Proceedings of the 2011 joint workshop on multilingual ocr and analytics for noisy unstructured text data**. [S.l.: s.n.], 2011.

VAPNIK, V. N. **The nature of statistical learning theory**. [S.l.]: Springer-Verlag, 1999.

VAPNIK, V. N.; CHERVONENKIS, A. Y. **Theory of pattern recognition: statistical problems of learning**. [S.l.]: Nauka, 1974.