INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO CEARÁ
IFCE CAMPUS ARACATI
COORDENADORIA DE CIÊNCIA DA COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

José Wellington Pereira Júnior

Aplicação de Técnicas Supervisionadas de Aprendizado de Máquina para Análise de Sentimentos em Avaliações Docentes

ARACATI-CE
2020

José Wellington Pereira Júnior

Aplicação de Técnicas Supervisionadas de Aprendizado de Máquina para Análise de Sentimentos em Avaliações Docentes

Trabalho de Conclusão de Curso (TCC) apresentado ao curso de Bacharelado em Ciência da Computação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará - IFCE - Campus Aracati, como requisito parcial para obtenção do Título de Bacharel em Ciência da Computação.

Orientador (a): Prof. Msc. Silas Santiago Lopes Pereira

Aracati-CE
2020

Dados Internacionais de Catalogação na Publicação
Instituto Federal do Ceará - IFCE
Sistema de Bibliotecas - SIBI
Ficha catalográfica elaborada pelo SIBI/IFCE, com os dados fornecidos pelo(a) autor(a)

Júnior, José Wellington Pereira.
Aplicação de Técnicas Supervisionadas de Aprendizado de Máquina para Análise de Sentimentos em Avaliações Docentes / José Wellington Pereira Júnior. - 2020.
f. : il. color.

Trabalho de Conclusão de Curso (graduação) - Instituto Federal do Ceará, Bacharelado em Ciência da Computação, Campus Aracati, 2020.
Orientação: Prof. Me. Silas Santiago Lopes Pereira.

1. Avaliações Docentes. 2. Análise de Sentimentos. 3. Aprendizado de Máquina. 4. Classificação de Texto. I. Titulo.

JOSÉ WELLINGTON PEREIRA JÚNIOR

APLICAÇÃO DE TÉCNICAS SUPERVISIONADAS DE APRENDIZADO DE MÁQUINA PARA ANÁLISE DE SENTIMENTOS EM AVALIAÇÕES DOCENTES

Trabalho de Conclusão de Curso (TCC) apresentado ao curso de Bacharelado em Ciência da Computação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará - IFCE - Campus Aracati, como requisito parcial para obtenção do Título de Bacharel em Ciência da Computação.

Aprovada em 10 de Março de 2020

BANCA EXAMINADORA

Prof. Msc Silas Santiago Lopes Pereira (Orientador)
IFCE

Prof. Dr. Mário Wedney de Lima Moreira
IFCE

Prof. Dr. Reinaldo Bezerra Braga
IFCE

DEDICATÓRIA

À minha família que sempre deu o apoio necessário para que eu pudesse chegar até aqui.

AGRADECIMENTOS

Agradeço primeiramente a Deus, por me dá a saúde e a disposição necessária para enfrentar essa longa jornada.

Agradeço a minha família que sempre me apoiou e compreendeu meus momentos de ausência, em especial, meus pais Jane Cláudia Damasceno e José Wellington Pereira que nunca mediram esforços para me proporcionar uma educação de qualidade. Agradeço à minha namorada por todo apoio e companheirismo principalmente na etapa final dessa longa caminhada.

Agradeço a todos os meus professores que contribuíram de maneira significativa na minha formação e crescimento. Agradeço principalmente ao meu orientador professor Silas Santiago que foi fundamental na concepção do deste trabalho e sobretudo na minha formação. Em especial agradeço a meu grande mestre Mauro Oliveira que foi fundamental não só na minha formação acadêmica, mas também na minha formação como ser humano.

Agradeço a todos os meus amigos que de alguma forma participaram desta longa caminhada. Em especial agradeço a Wesley Oliveira e Ruan Gondim que foram fundamentais para a concepção desta pesquisa.

Agradeço ao Laboratório de Redes e Sistemas Multimídia (LAR) por possibilitar experiência e crescimento profissional ao longo de toda minha graduação. Em especial aos professores Reinaldo Braga e Carina Oliveira que conduzem de maneira brilhante o laboratório.

Enfim, agradeço a todos que de alguma forma contribuíram para minha formação acadêmica, profissional e pessoal.

RESUMO

Promover melhorias na educação pode proporcionar o crescimento de qualquer sociedade. As universidades, por exemplo, têm papel fundamental na formação de profissionais qualificados. Nesse contexto, criar mecanismos para possibilitar a avaliação dos alunos sobre os mais diversos aspectos que envolvem o ambiente educacional é um fator importante na melhoria do processo de ensino aprendizado e consequentemente na educação de uma sociedade. Em especial, é essencial saber a satisfação dos discentes quanto às metodologias usadas por professores em sala de aula. Nesse cenário, os Sistemas de Resposta Estudantis (SREs) permitem a coleta de avaliações contendo a satisfação dos alunos. Contudo, extrair informações úteis dessas avaliações docentes pode ser um processo lento e cansativo. Para isso, técnicas de Mineração de Dados Educacionais (MDE) surgem como uma aplicação que tem o intuito de otimizar o processo de extração de informações de dados educacionais. Considerando todas essas informações, neste trabalho são aplicadas um conjunto de técnicas de Aprendizado de Máquina (AM) a uma base contendo avaliações docentes. O objetivo é criar um modelo preditivo inteligente para extrair a satisfação de alunos (“Insatisfatório”, “Neutro” e “Satisfatório”) de avaliações docentes de maneira automática. Foi realizado um total de dezesseis experimentos a partir da combinação de diferentes algoritmos de AM e diferentes técnicas de representação de palavras. Com a realização dos experimentos, foi feito uma análise comparativa dos experimentos com base em uma série de métricas de desempenho para selecionar o modelo preditivo que melhor soluciona o problema abordado. A partir da análise dos resultados obtidos, no melhor dos casos, o algoritmo *Support Vector Machine* (SVM) obteve valor de acurácia superior a 86% na predição da satisfação dos alunos.

Palavras-chaves: Avaliações Docentes. Análise de Sentimentos. Aprendizado de Máquina. Classificação de Texto.

ABSTRACT

Promoting improvements in education can provide growth for any society. Universities, for example, have a fundamental role in training qualified professionals. In this context, creating mechanisms to enable students to evaluate the most diverse aspects that involve the educational environment is an important factor in improving the teaching-learning process and, consequently, in the education of a society. In particular, it is essential to know the students' satisfaction with the methodologies used by teachers in the classroom. In this scenario, Student Response Systems (SRS) allow for the collection of assessments containing student satisfaction. However, extracting useful information from these teacher evaluations can be a slow and tiring process. For this, Educational Data Mining (EDM) techniques emerge as an application that aims to optimize the process of extracting information from educational data. Considering all this information, in this work a set of Machine Learning (ML) techniques are applied to a base containing teacher evaluations. The goal is to create an intelligent predictive model to extract student satisfaction ("Unsatisfactory", "Neutral" and "Satisfactory") from teacher evaluations in an automatic way. A total of sixteen experiments were carried out based on the combination of different ML algorithms and different word representation techniques. With the performance of the experiments, a comparative analysis of the experiments was made based on a series of performance metrics to select the predictive model that best solves the problem addressed. From the analysis of the results obtained, at best, the Support Vector Machine (SVM) algorithm obtained an accuracy value greater than 86% in the prediction of student satisfaction.

Keywords: Teacher evaluations. Sentiment Analysis. Machine Learning. Text Classification.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

INEP	Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira
IES	Instituições de Ensino Superior
MDE	Mineração de Dados Educacionais
AS	Análise de Sentimentos
AM	Aprendizado de Máquina
API	*Application Programming Interface*
REST	*RepresentationalStateTransfe*
SRE	Sistema de Resposta Estudantil
PLN	Processamento de Linguagem Natural
BOW	*Bag-Of-Words*
TF-IDF	*Term Frequency-Inverse Document Frequency*
IA	Inteligência Artificial
AD	Árvores de Decisão
NB	*Naive Bayes*
SVM	*Support Vector Machine*
TAE	Teoria do Aprendizado Estatístico
RBF	*Radial Basis Function*
RNA	Rede Neural Artificial
MLP	*Multilayer Perceptron*
EAD	Educação a Distância
POC	Prova de Conceito
IFCE	Instituto Federal de Educação, Ciência e Tecnologia do Ceará
JSON	*Javascript Object Notation*

NLTK	*Natural Language Toolkit*
LSTM	*Long Short-Term Memory*

SUMÁRIO

1 INTRODUÇÃO

Promover melhorias na educação é um passo importante para viabilizar avanços em qualquer sociedade. As universidades têm papel fundamental na formação de profissionais qualificados. Por isso é primordial a adoção de medidas para manter qualidades dos cursos e, principalmente, combater a evasão universitária. A desistência em cursos superiores tem mostrado dados preocupantes nos últimos anos no Brasil. Dados do Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (INEP) mostram que só no ano de 2018 nas Instituições de Ensino Superior (IES) públicas e privadas houveram quase 3,5 milhões de matrículas desvinculadas ou trancadas (INEP, 2019).

A evasão pode ser atribuída a inúmeros motivos. Segundo (BARROSO; FALCÃO, 2004) há no mínimo três variáveis para justificar a evasão, são elas: 1) econômica: quando a situação socieconômica do aluno não permite que ele prossiga no curso; 2) vocacional: quando não existe a adaptação do discente com o curso; 3) institucional: no caso de não haver adequação dos métodos de estudo, ou por inadequação dos métodos de ensino, pois o aluno não consegue entender os conteúdos e acaba desmotivado por conta das reprovações. No combate à evasão, principalmente causadas por motivos institucionais, é crucial considerar a satisfação dos alunos em relação aos mais diversos aspectos que envolvem o ambiente universitário, inclusive o desempenho dos professores no processo de ensino.

Nesse contexto, o desempenho de professores nas salas de aulas tem sido objeto de pesquisas (BALAHADIA; FERNANDO; JUANATAS, 2016). Para avaliar e tentar melhorar a atuação dos docentes, inúmeros mecanismos vem sendo propostos nos últimos anos. Uma parte importante do processo de melhora do ensino é a coleta de opiniões dos alunos (SANTOS; SILVEIRA; LECHUGO, 2016). Essas opiniões podem impactar seriamente os programas de auto-aperfeiçoamento das universidades, fornecendo um ponto de vista que pode afetar desde decisões sobre os professores até a estrutura dos futuros semestres desses cursos (NEWMAN; JOYNER, 2018).

A partir do momento que um professor tem qualquer *feedback*, ele pode adaptar-se para melhorar seu desempenho em sala de aula. Assim, um professor eficiente é aquele que possui um amplo repertório de técnicas e é capaz de usá-las habilmente para atender as novas demandas de seus alunos (BALAHADIA; FERNANDO; JUANATAS, 2016).

Grande parte das avaliações institucionais no Brasil são apresentadas na forma de perguntas objetivas. Os modelos objetivos apresentam facilidades na hora

de tabular os dados (SANTOS; LECHUGO; SILVEIRA-MACKENZIE, 2016). Contudo, esses modelos acabam limitando os estudantes às perguntas pré-definidas nos questionários. Perguntas no modelo aberto ou subjetivo dão maior liberdade para os alunos reportarem suas impressões a respeito dos métodos de ensinos utilizados em aula. Contudo, esse modelo de perguntas demanda mais tempo de análise para que informações úteis sejam extraídas. Assim, é fundamental encontrar soluções que ajudem a automatizar o processo de extração de conhecimento de formulários subjetivos.

Nesse contexto, a Mineração de Dados Educacional (MDE) surge como uma aplicação que visa otimizar o processo de extração de informações no meio educacional (ALTRABSHEH; GABER; COCEA, 2013). A MDE possui um conjunto de técnicas que visa explorar grandes bases de dados com o intuito de encontrar padrões que ajudem no processo de tomada de decisão (SANTOS; LECHUGO; SILVEIRA-MACKENZIE, 2016). Dentro das diversas aplicações que podem ser exploradas pela MDE, pode-se destacar a Análise de Sentimentos (AS).

A AS ou mineração de opiniões é um método computacional para identificar, ou categorizar opiniões expressas em um texto (SCHÜTZE; MANNING; RAGHAVAN, 2008). Com a análise de sentimentos, pode-se encontrar polaridades expressas em textos, como: “Positivo”, “Negativo” ou “Neutro” (ALTRABSHEH; GABER; COCEA, 2013). Além disso, a AS também pode ser usada para encontrar outros sentimentos no texto, como: “Amor”, “Alegria” e “Raiva”.

Os estudos sobre AS espalharam-se da ciência da computação para outras áreas como ciências sociais e da administração, devido à sua grande importância para os negócios e para a sociedade (ZHANG; WANG; LIU, 2018). Essa relevância atribuída à área pode estar diretamente ligada à influência que opiniões de terceiros causam nas decisões de um determinado indivíduo. Essas opiniões podem influenciar desde a escolha de um determinado produto ou serviço, até influências em escolhas políticas ou sociais.

As pesquisas em análise de sentimentos vêm crescendo nos últimos anos. Nota-se uma correlação entre o crescimento da área e o avanço das mídias sociais na *Web* como, por exemplo, revisões, *blogs*, fóruns, *Twitter* e outras redes sociais. Pela primeira vez na história, há um volume gigantesco de dados opinativos registrados em formulários digitais (ZHANG; WANG; LIU, 2018). Dessa forma, diversas aplicações que utilizam mineração de opiniões foram produzidas, inclusive relacionadas ao processo de ensino-aprendizado.

Considerando todos esses fatores, este trabalho propões a construção de um mecanismo inteligente baseado em análise de sentimentos que visa classificar a avaliação de um aluno direcionado para uma determinada aula como: “Negativo”, “Neutro” ou “Positivo”. O objetivo é extrair de maneira automática a satisfação dos alunos com

relação aos métodos de ensino usados em sala de aula. Dessa forma, há uma otimização no tempo de análise dos *feedback's* estudantis por parte dos professores ou gestores educacionais. Assim, é possível tomar decisões mais rápidas em relação a mudanças nas metologias de ensino por parte de professores, para assim melhorar o processo de ensino aprendizado.

1.1 Objetivos

1.1.1 Objetivo Geral

Este trabalho objetiva a construção de um mecanismo inteligente baseado em análise de sentimentos para extrair a satisfação de avaliações estudantis de maneira automática.

1.1.2 Objetivos Específicos

Para que se alcance o objetivo geral deste trabalho, é destacado abaixo objetivos específicos da pesquisa:

- criar uma base de dados em português com avaliações relacionadas ao desempenho de professores em sala de aula.
- estudar as principais técnicas de análise de sentimentos usando AM disponíveis na literatura.
- submeter os dados coletados a uma série de experimentos combinando técnicas de pré-processamento de texto com algoritmos de AM, com o intuito de gerar modelos inteligentes capazes de classificar novas avaliações.
- selecionar o modelo que obtiver o melhor desempenho com base nas métricas de avaliação para compor o módulo inteligente do sistema.
- propor a arquitetura de uma Prova de Conceito (do inglês *Proof of Concept* - PoC) web para testar o modelo preditivo criado no trabalho.

1.2 Organização do Trabalho

Este trabalho está organizado da seguinte forma: o Capítulo 2 apresenta a fundamentação teórica, onde são abordados os principais conceitos envolvidos na realização da pesquisa; no Capítulo 3, são apresentados os trabalhos relacionados com

a área de estudo da pesquisa; no Capítulo 4, é feito um detalhamento da proposta, bem como a metodologia adotada para desenvolvimento e validação da pesquisa; o Capítulo 5 apresenta os resultados obtidos durante o decorrer do trabalho; por fim, no Capítulo 6 são mostradas as conclusões e os trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção é apresentada a fundamentação teórica necessária para um melhor entendimento da proposta do trabalho. No decorrer do capítulo, são explicados os contextos educacionais da pesquisa e os principais conceitos relacionados ao Processamento de Linguagem Natural, Análise de Sentimentos, classificação de textos e Aprendizado de Máquina.

2.1 Contexto Educacional

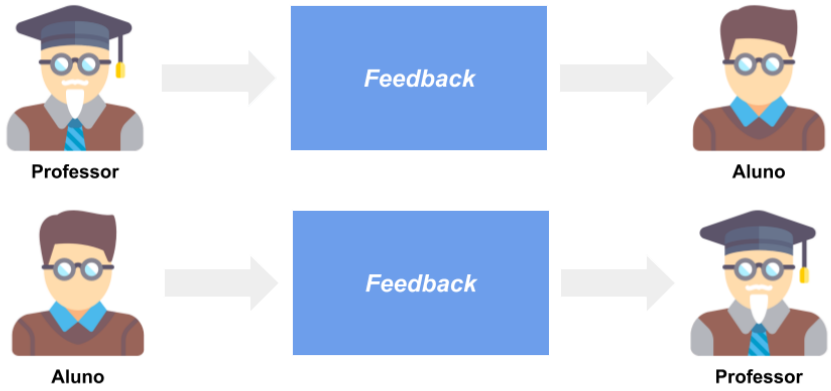
Para melhorar o processo de ensino-aprendizado, muitas estratégias vêm sendo adotadas por gestores e professores. Por exemplo, os Sistemas de Resposta de Estudantil (SRE) tem como objetivo receber avaliações de alunos a respeito dos mais diversos aspectos que envolvem o ambiente educacional, principalmente sua satisfação em relação às didáticas usadas em sala de aula. Assim, a coleta de opiniões de alunos torna-se parte importante na melhora do processo de ensino-aprendizado (SANTOS; SILVEIRA; LECHUGO, 2016). A avaliação de alunos permite que os docentes adaptem suas metodologias de ensino de acordo com a necessidade dos discentes (ALTRABSHEH; GABER; COCEA, 2013).

No contexto educacional, pode-se definir dois tipos de *feedback*: 1) *feedback* do professor para alunos, visando seu auto-aperfeiçoamento; 2) *feedback* dos alunos para professores, isso possibilita o docente tomar decisões sobre suas metodologias de ensino de maneira a maximizar seu desempenho em sala de aula (ALTRABSHEH; GABER; COCEA, 2013). A Figura 1 apresenta os dois tipos de avaliações no contexto educacional.

O *feedback* do estudante para o professor pode ser feito de maneira quantitativa objetiva ou qualitativa subjetiva. O modelo de avaliação objetiva apresenta uma maior facilidade na hora de tabular e extrair informações. Contudo, os modelos qualitativos permitem uma maior riqueza de informações, já que os alunos não ficam presos a um formulário pré-definido por um especialista e podem expressar suas opiniões livremente (SANTOS; LECHUGO; SILVEIRA-MACKENZIE, 2016).

Os Sistemas de Resposta Estudantil são ferramentas usadas para coletar a opinião/avaliação de alunos a respeito das entidades que compõem o ambiente educacional, inclusive o desempenho de docentes. Os SREs podem rodar sobre diversas plataformas, incluindo a Web e aplicações mobile. Além de pedir especificamente o *feedback*, os professores podem fazer perguntas aos alunos e os estudantes podem

Figura 1 – Tipos de *feedback's* estudantis.



Fonte: Elaborada pelo Autor.

fazer perguntas aos docentes. Portanto, os SREs podem ajudar na interação entre professor e aluno (ALTRABSHEH; GABER; COCEA, 2013).

Apesar de ser importante o processo de extrair informações das avaliações dos alunos coletadas por SREs, pode ser uma tarefa demorada e exaustiva, pois há necessidade de analisar inúmeras avaliações uma a uma até que o professor tenha um retorno satisfatório. Supondo que um professor ministre 4 disciplinas, cada uma com 30 alunos, e que os *feedback's* sejam feitos semanalmente, toda semana o educador tem 120 avaliações para ler, interpretar e tirar alguma conclusão, para depois tomar alguma decisão em relação as suas metodologias de ensino. Portanto, nas próximas seções são exploradas técnicas para automatizar e otimizar o processo de extração de informações dos *feedback's* passados pelos estudantes.

2.2 Processamento de Linguagem Natural

O tratamento de linguagem natural por máquinas é uma tarefa complexa que gera pesquisas desde os primórdios da computação. Essa complexidade é causada pelo alto grau de ambiguidade presente nas linguagens naturais, pois estas, diferentes das linguagens de programação, não possuem uma definição formal que facilite seu processamento por computadores (VIEIRA; LOPES, 2010).

O Processamento de Linguagem Natural (PLN) surge como uma subárea da Inteligência Artificial que estuda a elaboração de sistemas computacionais que analisam, reconhecem ou geram textos em linguagens humanas (VIEIRA; LOPES, 2010). Para (RUSSELL, 2013) existem duas principais razões para o desenvolvimento de

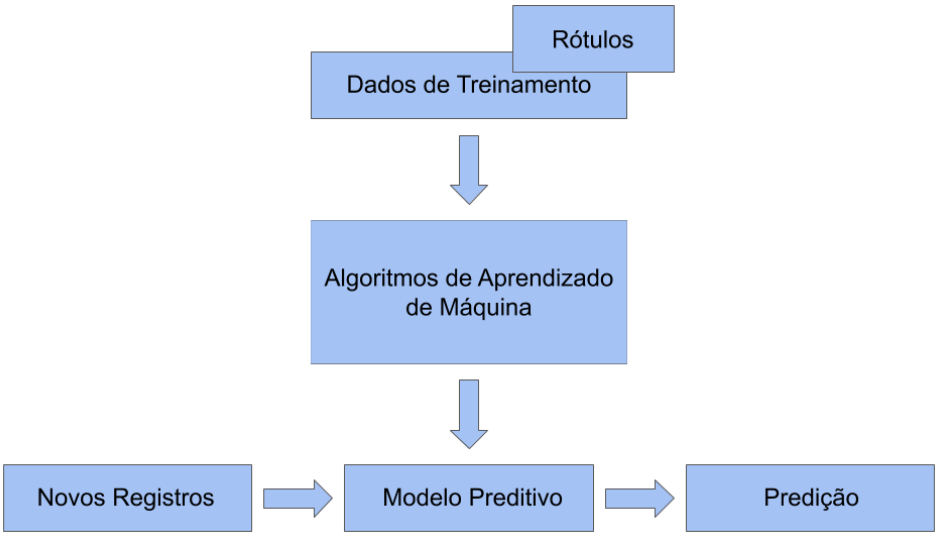
estudos na área de PLN: 1) fazer com que máquinas comuniquem-se com seres humanos; 2) adquirir e extrair informações a partir de linguagens escritas.

No PLN existem diversas tarefas que trabalham com diferentes aplicações. Três das tarefas mais utilizadas no PLN são: classificação de texto, recuperação de informação e extração de informação. Em especial, a classificação de texto é usada para inúmeras aplicações. Por exemplo, a AS que permite extrair sentimentos ou polaridade de textos.

2.2.1 Classificação de texto

Em um problema de classificação de texto, uma sentença é associada a um rótulo que representa alguma informação presente no texto. Por exemplo, o rótulo pode ser “positivo” ou “negativo”, que representa a polaridade daquele texto. Nesse método, após passar por todo processo de pré-processamento, o texto é submetido a um modelo de AM que foi treinado com um conjunto de dados rotulados relacionados ao contexto trabalhado. Esse modelo classificará a nova entrada e atribuirá o rótulo adequado ao texto. A Figura 2 representa o processo seguido na classificação de texto. Na Seção 2.3 será feito um detalhamento das técnicas de AM usadas na pesquisa proposta.

Figura 2 – Representação do processo classificação de texto através do Aprendizado de Máquina.



Fonte: Adaptado de (WALKER, 2018).

Análise de Sentimentos ou Mineração de Opinião é uma aplicação do PLN, linguística computacional e análise de texto. Este campo de estudo visa a identificação e extração de certas informações do texto (ALTRABSHEH; GABER; COCEA, 2013). Na AS, é extraído uma polaridade do texto. Em geral, essa polaridade é “positiva”

ou “negativa” mas também pode ser expressa em um intervalo. Essa faixa pode ser em uma escala de n pontos, como: “muito ruim”, “ruim”, “regular”, “bom” e “muito bom” (PRABOWO; THELWALL, 2009). Além de extrair a polaridade de textos, a AS também pode ser usada para obter diferentes emoções contidas em textos, como: “Amor”, “Alegria”, “Raiva”, “Frustração”, entre outros (TIAN et al., 2009).

A seguir são explicados os conceitos presentes no PLN que são fundamentais para compreensão deste trabalho. Esses conceitos são: coleta de dados, necessária para a construção de uma base de treinamento; pré-processamento, necessário para limpar todas as informações desnecessárias da base, com o intuito de melhorar o desempenho na etapa de classificação de texto; por fim, as técnicas de representação de palavras presentes na pesquisa.

2.2.2 Coleta de Dados

Para realizar a tarefa de classificação de texto supervisionada, é essencial a presença de textos já rotulados. Por isso, é essencial uma fase de coleta de dados para montar a base de treinamento. Atualmente as redes sociais são uma fonte excelente de dados, tendo em vista que usuários expressam milhares de opiniões a cada dia (ALTRABSHEH; GABER; COCEA, 2013). Outras fontes de dados incluem os sistemas específicos para coleta de opiniões e satisfação, como os serviços de avaliações de produtos ou compras e até mesmo os SREs. Para a coleta de dados do presente trabalho, é utilizado um formulário criado e disponível na ferramenta de formulários do *google*, *google forms* [1]. O formulário foi usado para coletar avaliações de alunos a respeito de sua satisfação com o desempenho de seus professores. O formulário e a metologia usada para coleta são detalhados na Seção 4.1.1.

2.2.3 Pré-Processamento de Dados

Depois da coleta, os dados devem passar por uma fase de pré-processamento antes do estágio de análise de sentimentos, para aumentar a precisão e diminuir o erro nos dados (ALTRABSHEH; GABER; COCEA, 2013). A seguir são apresentadas as técnicas de pré-processamento de texto adotadas nesta pesquisa.

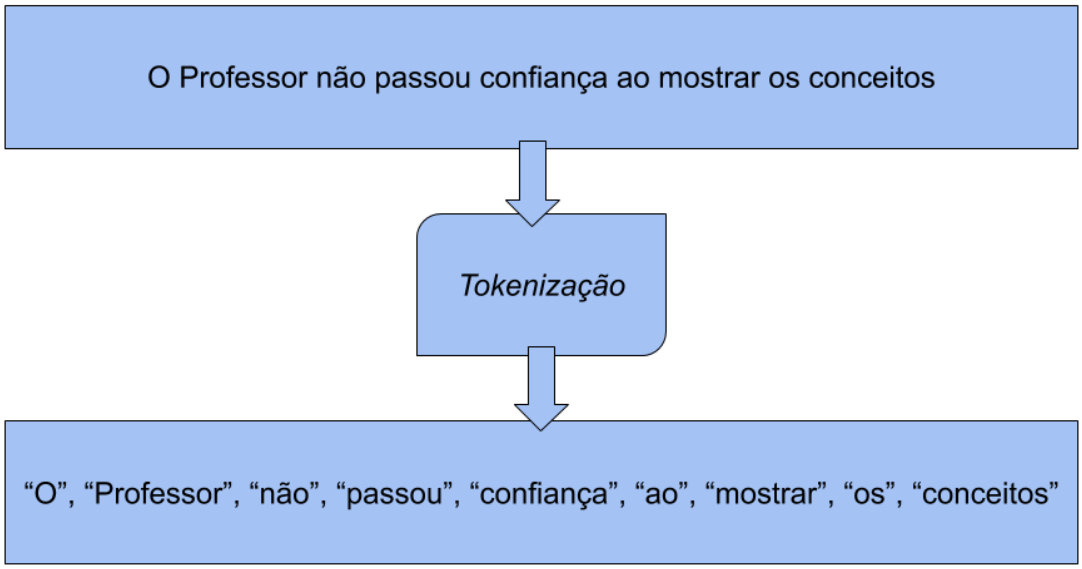
2.2.3.1 Tokenização

Geralmente, é a primeira etapa do pré-processamento e envolve a quebra ou a divisão do texto em partes menores chamadas de *tokens*. Os *tokens* às vezes

[1] https://www.google.com/intl/pt-BR/forms/about/

são sinônimos de palavras em uma frase, contudo, deve-se notar que as pontuações podem ser tratadas como *tokens* de acordo com o esquema de *tokenização* adotado (WALKER, 2018). A Figura 3, ilustra o exemplo de uma frase *tokenizada*.

Figura 3 – Exemplo de *tokenização*.



Fonte: Elaborada pelo Autor.

2.2.3.2 Normalização

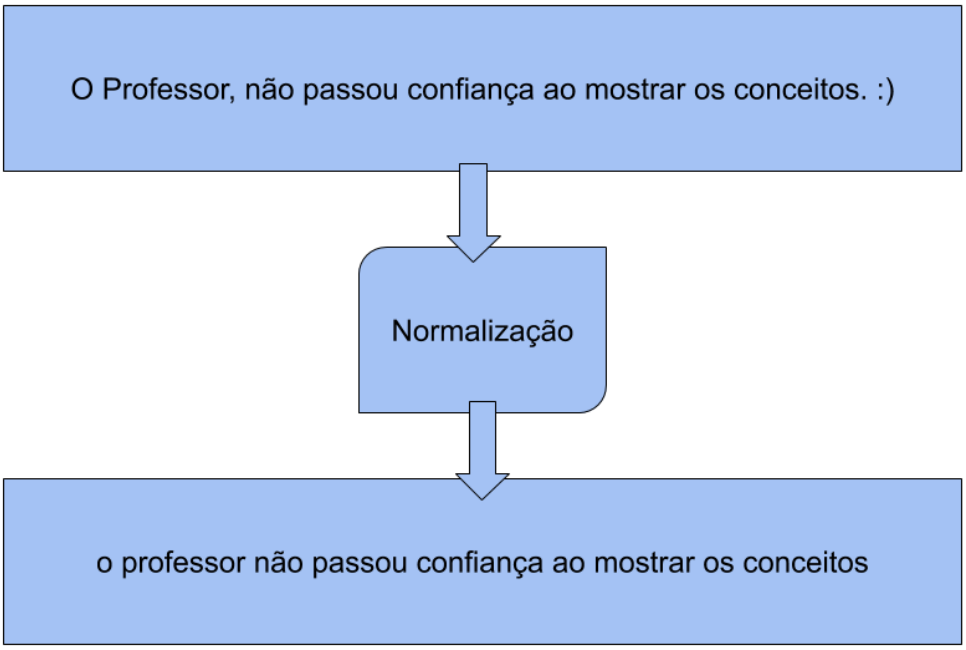
A normalização é uma das etapas mais importantes no processo de pré-processamento de texto. Ela envolve o processo de transformar os *tokens* em uma forma canônica. Ou seja, todo o texto é padronizado para mesma forma, independente de haver pequenas diferenças (WALKER, 2018).

Alguns processos envolvidos na etapa de normalização são: transformação de letrar maiúsculas para minúsculas; remoção de caracteres especiais, como **[’-’, ’+’, ’*’, ’/’, ’(’, ’)’, ’@’, ’{’, ’}’]**; a eliminação de quebras de linhas; remoção de pontuação entre outros. Resumindo, nesta etapa todos os *tokens* que não carregam valor semântico e não influenciam no sentimento ou polaridade do texto, são descartados. A Figura 4 exemplifica uma frase após passar por uma função de normalização de texto.

2.2.3.3 Remoção de Stopwords

É um dos métodos mais aplicados no pré-processamento de texto. Consiste na remoção de palavras comuns, como: “a”, “o”, “que” entre outras, pois na grande parte das vezes essas palavras não fornecem informações relevantes na construção

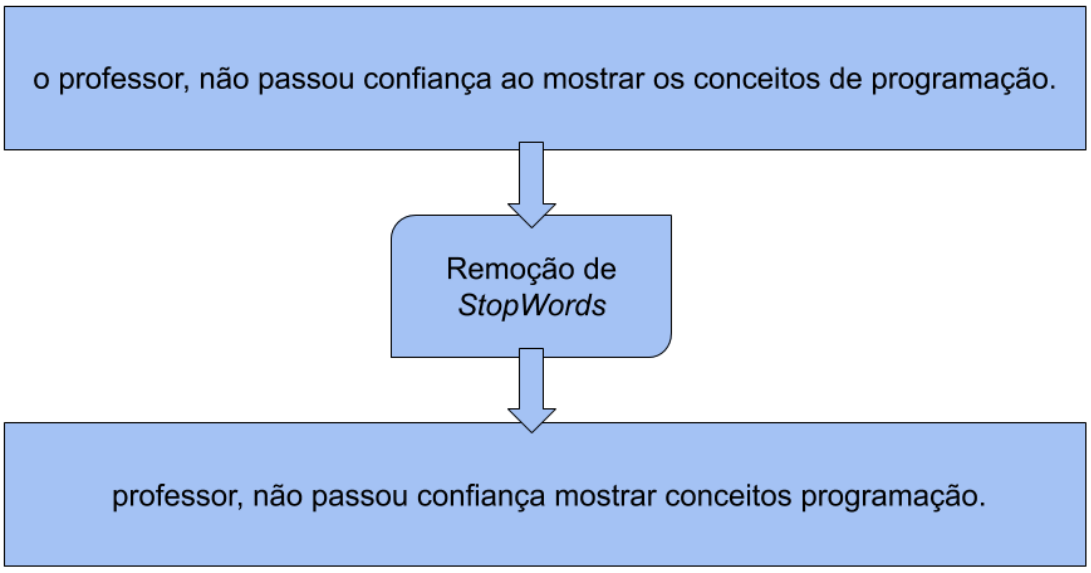
Figura 4 – Exemplo de normalização.



Fonte: Elaborada pelo Autor.

do modelo. Obviamente, as *stopword* variam de acordo com o idioma e com o contexto abordado, pois palavras que são insignificantes em determinado tema podem ter relevância em outro (AGUIAR, 2017). Supondo que temos uma lista de *stopwords* L = [“o”, “a”, “os”, “as”, “ao”, “aos”, ..., “de”, “do”], a Figura 5 mostra o exemplo de uma frase depois de passar pelo processo de remoção de *stopwords*.

Figura 5 – Exemplo de remoção de *stopwords*.



Fonte: Elaborada pelo Autor.

2.2.4 Representação de Palavras

Antes de ser submetido ao treinamento com algoritmos de AM, utilizados no trabalho, o texto da base de treinamento precisa de ser representado como um conjunto de atributos.

As técnicas de representação selecionadas para a execução do trabalho são baseadas em *n-gramas*. Um *n-grama* pode ser definido como uma cadeia de n palavras retiradas de determinado texto. Se uma palavra é tomada por vez, esta sequência é chamada de *unigrama*. Sequências de duas palavras são chamadas *bigrama* (CARVALHO et al., 2018).

Um vocabulário v representa todos os *n-gramas* presentes no corpus. De acordo com a Equação 2.1, cada elemento presente no vocabulário v recebe um índice i, onde w_i representa o *n-grama* de índice i e n é a quantidade de *n-gramas* existentes no corpus (CARVALHO et al., 2018).

$$v = \{w_1, w_2, ..., w_n\} i, n \in \mathbb{N} \tag{2.1}$$

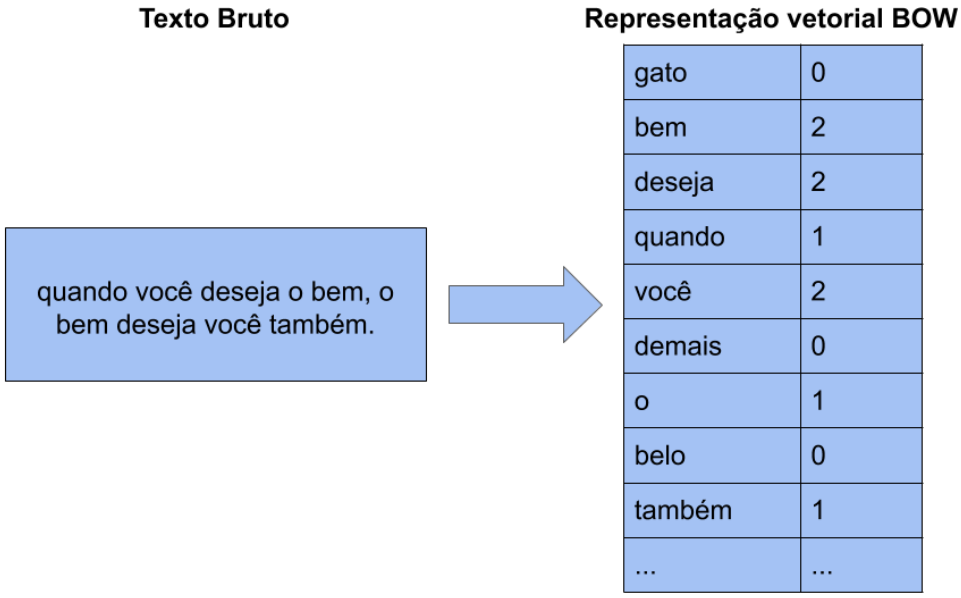
2.2.4.1 Bag-Of-Words

O método *Bag-Of-Words* (BOW) representa um documento de texto como um vetor que guarda a frequência de presença de cada *n-grama* do vocabulário. A Equação 2.2 descreve cada elemento c_i que representa a contagem das ocorrências de w_i no documento d (CARVALHO et al., 2018). O BOW pode ser tratado como uma representação *unigrama* ou *1-grama*.

$$d = [c_1, c_2, ..., c_n] i, n \in \mathbb{N} \tag{2.2}$$

A Figura 6 mostra um exemplo simples de uma frase sendo representada usando o BOW. O BOW é utilizado com frequência por conta de sua simplicidade no processo de classificação (MEDHAT; HASSAN; KORASHY, 2014). Apesar de eficiente, o BOW apresenta algumas limitações. Por exemplo, considerar apenas frequência das palavras. Em muitas ocasiões algumas palavras aparecem diversas vezes em um documento, mas a alta frequência dessa palavra não adiciona nenhum contexto para identificar o sentimento ou polaridade do documento (WALKER, 2018).

Figura 6 – Exemplo de uma frase representada com o *Bag-Of-Words*.



Fonte: Elaborada pelo Autor.

2.2.4.2 *Term Frequency-Inverse Document Frequency*

Frequência do termo-inverso da frequência nos documentos ou do inglês *term frequency-inverse document frequency* (TF-IDF) é uma técnica usada para mitigar o problema das palavras muito frequentes que não adicionam nenhum contexto ao documento (WALKER, 2018). O TF-IDF pode ser separado em duas fases. Na primeira, a do "TF", o algoritmo faz a contagem do número de palavras pela frequência normalizada (CARVALHO, 2018). A Equação 2.3 mostra a fórmula do passo "TF".

$$TF = f_{(t,d)} = \frac{f_{t,d}}{f_{t',d}} \tag{2.3}$$

Onde t representa o termo e d o documento. É realizado uma normalização dividindo pela frequência de t' o termo mais repetido no documento d (CARVALHO, 2018).

A segunda etapa do TF-IDF, é a do *Inverse Document Frequency* o "IDF". Nessa etapa, procura-se diminuir a influência das palavras que tem alta frequência nos documentos (SANTOS; LECHUGO; SILVEIRA-MACKENZIE, 2016). O "IDF" pode ser calculado usando:

$$IDF = log\frac{N}{nt} \tag{2.4}$$

Onde N representa o número total de documentos e nt a quantidade de documentos que contém o termo t. O cálculo final do TF-IDF pode ser representado pela

Equação 2.5.

$$\textit{TF-IDF} = f_{(t,d)*log\frac{N}{nt}} \tag{2.5}$$

2.3 Aprendizado de Máquina

Aprendizado de Máquina ou do inglês *Machine Learning* é uma subárea da Inteligência Artificial (IA) inspirado na ideia de que sistemas podem aprender através de casos passados, encontrar padrões e tomar decisões com o mínimo de intervenção humana (RUSSELL, 2013).

Existem três principais categorias de aprendizado de máquina. Cada um é determinado por um tipo de *feedback* diferente (RUSSELL, 2013).

Na **aprendizagem não supervisionada**, o agente inteligente aprende padrões na entrada, embora não seja passado nenhum tipo de *feedback* explícito para o algoritmo. A tarefa mais comum nesse tipo de aprendizado é o **agrupamento** (RUSSELL, 2013). Agrupamento ou *Clustering* (do inglês), é um conjunto de técnicas e algoritmos de AM que fazem um agrupamento automático de dados de acordo com um grau de semelhança.

Na **aprendizagem por reforço**, o agente aprende a partir de recompensas ou punições (RUSSELL, 2013). Com a repetição dos experimentos, espera-se que o agente comece a perceber as ações que levam a uma maior recompensa, e passe a evitar as ações que geram punição ou recompensa menor (HONDA, 2017).

Na **aprendizagem supervisionada**, o agente analisa alguns pares de entrada e saída e aprende uma função mapear ambas. Então, podemos dizer que a aprendizagem não supervisionada pode ser representada pelo seguinte:
Dado um conjunto de treinamento de N pares de exemplos de entrada e saída

$$(x_1, y_1), (x_2, y_2), ...(x_n, y_n)$$

onde cada y_j foi gerado por uma função desconhecida $y = f(x)$, o modelo supervisionado vai descobrir uma função h que se aproxime da função original f (RUSSELL, 2013). As principais técnicas do aprendizado supervisionado são: a classificação, quando se busca resultados discretos; e regressão, quando o objetivo é encontrar saídas contínuas.

A seguir é feito um detalhamento das técnicas supervisionadas usadas para treinar o modelo preditivo para classificação de texto baseado em AS proposto na pesquisa. Foram selecionadas duas técnicas difundidas na literatura para realizar a tarefa de classificação de texto.

2.3.1 Árvores de Decisão

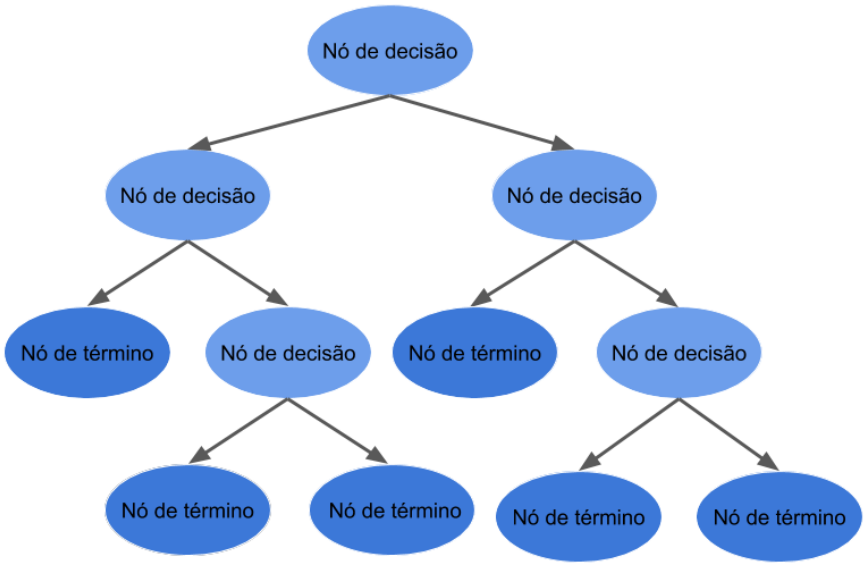
Árvores de Decisão (AD) é uma técnica supervisionada de AM que é usada para várias aplicações, inclusive para problemas de classificação de texto (WALKER, 2018).

Árvores de Decisão, como uma árvore natural, são formadas por nós, ramos e folhas. Os nós em uma árvore de decisão são chamados de nós de decisão, pois os exemplos no conjunto de treinamento geralmente são divididos em nós de acordo com recursos específicos (WALKER, 2018).

Uma AD faz uso do método de divisão e conquista para resolver um problema de decisão. Um problema complexo é decomposto em problemas mais simples, aos quais recursivamente é aplicado a mesma estratégia. As soluções dos problemas mais simples podem ser combinadas, na forma de uma árvore, para produzir a solução do problema maior (FACELI et al., 2011).

Formalmente, uma AD é um grafo acíclico direcionado (árvore) em que cada nó, ou é um nó de divisão (nó de decisão) com mais de um sucessor, ou é um nó folha (nó de término). Os nós condicionais possuem um teste condicional baseado nos valores de domínio de um atributo. Já os nós folhas são rotulados por uma função com um dos possíveis valores do atributo alvo (FACELI et al., 2011). A Figura 7 apresenta a estrutura de uma AD e seus respectivos nós de decisão e término.

Figura 7 – Estrutura de uma Árvore de Decisão.



Fonte: Elaborada pelo Autor.

Para criação de uma árvore de decisão, dois cálculos são necessários, o de entropia e o de ganho da informação. A entropia mede a aleatoriedade ou grau de

impureza (dificuldade para predizer) do atributo alvo, representando a falta de informação. A cada nó de decisão, o atributo que é escolhido para dividir os dados, é aquele que reduz mais a aleatoriedade da variável alvo. O cálculo da entropia é realizado pela Equação 2.6.

$$Entropia(S) = \sum_{i=1}^{c} -p_i \log_2 p_i \tag{2.6}$$

Onde S é o conjunto de entrada, c a quantidade de classes distintas para o problema e p_i a probabilidade de ocorrência da classe i. O ganho da informação, representa o quanto é preciso para completar a informação faltante. Dessa forma, o ganho da informação é calculado pela diferença da entropia do conjunto de entrada S e a soma ponderada da entropia das partições. O cálculo do ganho é representado pela Equação 2.7 , onde $Ganho(S, A)$ representa a redução na entropia de S, para cada atributo A.

$$Ganho(S, A) = Entropia(S) - \sum_{x \in valores(A)} \frac{|S_x|}{|S|} Entropia(S_x) \tag{2.7}$$

O cálculo da $Entropia(S)$ é realizado para todos os atributos do conjunto de treinamento. Em seguida, o $Ganho(S, A)$ é calculado para que o atributo mais importante seja selecionado para representar o nó raiz da árvore. Após essa etapa, são realizados novos cálculos para a seleção dos ramos da árvore (nós de decisão). Finalmente, são adicionados os nós folhas (nós de término) que representam os rótulos associados ao conjunto de dados de entrada (FACELI et al., 2011).

2.3.2 Naive Bayes

O *Naive Bayes* (NB) é um classificador poderoso e simples que é largamente utilizado na classificação de texto. O NB baseia-se no teorema das probabilidades de *Bayes* [2]. A principal característica do Teorema de *Bayes* é a suposição de independência, que afirma que todos os atributos são independentes uns dos outros (WALKER, 2018). A Equação 2.8 expressa o teorema de *Bayes*:

$$p(C_k|x) = \frac{p(C_k)p(x|C_k)}{p(x)} \tag{2.8}$$

A Equação 2.8 mostra que a probabilidade de uma classe C, dada as características x, é a probabilidade a *priori* da classe multiplicada pela probabilidade da característica x pertencer à classe C dividido pela probabilidade do atributo x ocorrer.

[2] $https://pt.wikipedia.org/wiki/Teorema_de_Bayes$

Vale ressaltar que o denominador é dado pela probabilidade de x sem a influência de um rótulo de classe, então o valor $p(x)$ é constante.

O *Naive Bayes* primeiro calcula a probabilidade a *priori* de cada classe do conjunto de dados. Em seguida, ele multiplica as probabilidades a priori de cada classe com a probabilidade de que cada atributo pertença a essa classe. A etapa final consiste na escolha do rótulo com maior probabilidade (WALKER, 2018).

Dado um conjunto de entrada X, em que $x_1, x_2, ..., x_d$ são os atributos da base. Então, pode-se dizer que x_j é j-ésimo atributo do exemplo X. Com isso, a probabilidade de um exemplo pertencer à classe C_k é dada por:

$$p(C_k|x) \propto P(C_k) \prod_{j=1}^{d} P(x_j|C_k) \tag{2.9}$$

O uso discriminante da Equação 2.9 obtém o classificador NB. O termo *naive* vem justamente da hipótese de que os valores dos atributos são independentes de sua classe (FACELI et al., 2011).

A fórmula do NB, pode ser ainda expressa em forma de somatório. Aplicando logaritmos a Equação 2.10, é obtido:

$$\log(P(C_k|x)) \propto \log(P(C_k)) + \sum_{j} \log(P(x_j|C_k)) \tag{2.10}$$

2.3.3 Support Vector Machine

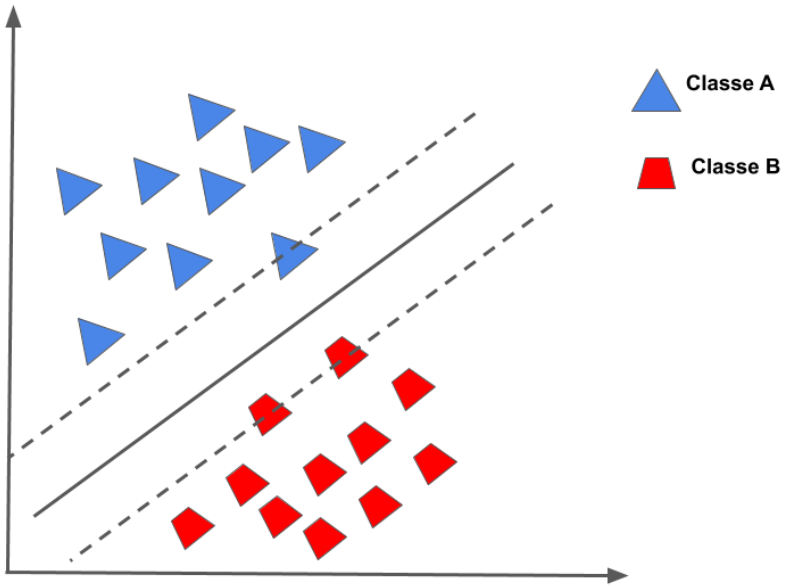
Máquinas de Vetores de Suporte, do inglês *Support Vector Machines* (SVMs), é uma técnica de AM que vem sendo bastante difundida nos últimos anos. Segundo (GOUDJIL et al., 2018) o SVM apresenta um excelente desempenho para tarefa de classificação de texto, principalmente quando o conjunto de treinamento é pequeno.

As SVMs são baseadas na teoria do aprendizado estatístico (TAE), desenvolvidas por *Vapnik* (VAPNIK, 1995). Nessa teoria são estabelecidas um conjunto de princípios que precisam ser seguidos para que uma boa capacidade de generalização seja obtida nos classificadores (LORENA; CARVALHO, 2007). Seja h um classificador e H o conjunto com todos os classificadores que uma determinada técnica de AM pode produzir, esse algoritmo faz uso de um conjunto de treinamento X composto de n pares (x_i, y_i) para gerar um classificador $\widehat{h} \in H$.

O aprendizado de uma SVM baseia-se em encontrar um hiperplano com margem máxima que representa a melhor reta existente para separar as instâncias a serem classificadas. Além do mais, o algoritmo procura encontrar o hiperplano de

margem máxima minimizando o erro dos dados em relação ao hiperplano encontrado. A Figura 8 mostra o exemplo de um hiperplano com margem máxima que melhor separa os triângulos (Classe A) dos trapézios (Classe B). Dessa forma, o novo objeto será classificado com triângulo ou trapézio dependendo de sua posição em relação ao hiperplano.

Figura 8 – Representação de um hiperplano de margem máxima.



Fonte: Elaborada pelo Autor.

Pode-se dizer que uma SVM é linear quando elas definem fronteiras lineares baseadas em dados linearmente separáveis. Assim, dado um conjunto X de treinamento com n elementos $x_i \in X$ e suas respectivas classes $y_i \in Y$. X é linearmente separável se é possível separar os objetos de classes diferentes com um hiperplano linear (FACELI et al., 2011). A equação usada pela SVM para encontrar um hiperplano de separação de dados é apresentada na Equação 2.11. Nela, $w * x$ representa o produto escalar entre os vetores w e x, $w \in X$ representa o vetor normal ao hiperplano e $\frac{b}{||w||}$ representa à distância do hiperplano à origem, sendo b uma constante $\in \Re$.

$$y = w * x + b \tag{2.11}$$

Nos cálculos realizados na SVM, existem inúmeras operações algébricas usando os pontos sobre os hiperplanos. Com essas manipulações, é possível a maximização da margem de separação dos objetos em relação a $w * x + b = 0$. A maximização da margem pode ser encontrada por meio da minimização de $||w||$ (CAMPBELL, 2000). Assim, recorre-se ao seguinte problema de otimização:

$$\underset{w,b}{Minimizar} \frac{1}{2}\|w\|^2 \tag{2.12}$$

Uma ideia das SVMs é alcançar o menor erro possível com a Equação 2.13, onde C é uma constante que impõe um peso à minimização dos erros nos dados de treinamento em relação à minimização na complexidade do modelo (FACELI et al., 2011). Tendo os erros dos vetores de suporte, dado pelo cálculo de $\sum_i^n \varepsilon_i$, a SVM atualiza o novo hiperplano cada vez que um valor menor que atual calculado é encontrado.

$$\underset{w,b,\varepsilon_i}{Minimizar} \frac{1}{2}\|w\|^2 + C\left(\sum_i^n \varepsilon_i\right) \tag{2.13}$$

Comumente, a técnica SVM é usada para trabalhar com dados linearmente separáveis. Contudo, pode haver a necessidade de tratar problemas que envolvem dados não linearmente separáveis. Para tratar esse problema, pode-se usar uma técnica chamada *Kernel Trick*. Essa técnica permite transformar um conjunto não linear em um conjunto linear. Uma função *Kernel* recebe dois pontos como entrada e faz o cálculo do produto escalar desses objetos no espaço de características (FACELI et al., 2011). Alguns *Kernels Trick* existentes são: linear, polinomial e *Gassian Radial Basis Function* (RBF).

2.3.4 Multilayer Perceptron

As Redes Neurais Artificiais (RNAs) são um conjunto de algoritmos de Aprendizado de Máquina que busca simular o comportamento do cérebro humano no processo de aprendizagem. Uma RNA, assim como o sistema nervoso de um humano, é composta por inúmeros neurônios interconectados. Os neurônios em uma Rede Neural Biológica possuem dendritos, corpo celular e axônio. De maneira análoga, uma RNA possui seus valores de entrada, função soma e função de ativação, respectivamente.

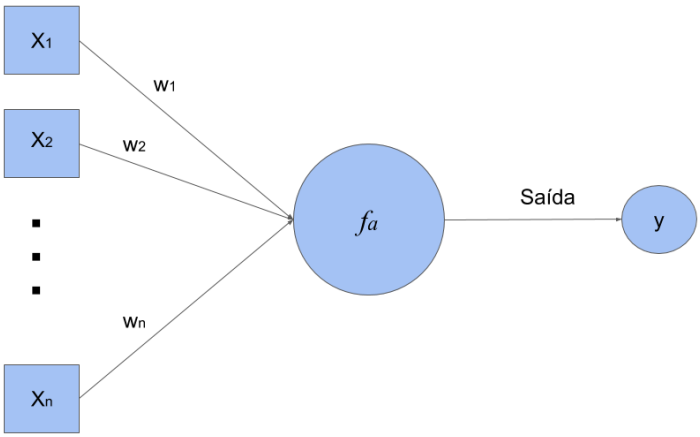
Os neurônios artificiais são dispostos em uma ou mais camadas e são interligados por uma grande quantidade de conexões, geralmente unidirecionais. Na maioria das arquiteturas, os neurônios artificiais simulam as sinapses biológicas e possuem pesos associados, que ponderam cada uma das entradas da rede. Os pesos podem assumir valores positivos ou negativos, dependendo do valor o comportamento da conexão pode ser excitatório ou inibitório, respectivamente (FACELI et al., 2011).

A Figura 9 mostra um modelo simples de um neurônio artificial. Cada um dos terminais de entrada de um neurônio recebe recebe um valor. Os valores recebidos são ponderados e combinados com uma função matemática f_a. Dessa forma, a saída da função é a resposta do neurônio para a entrada. Supondo uma entrada X com d atributos expressados na forma de um vetor $X = [x_1, x_2, ..., x_d]^t$ e um neurônio u com

pesos w_1, w_2,...,w_d, que podem ser representados pelo vetor $W = [w_1, w_2, ..., w_d]$. A entrada total recebida pelo neurônio u, pode ser definida pela Equação 2.14.

$$u = \sum_{j=1}^{d} x_j w_j \tag{2.14}$$

Figura 9 – Estrutura de um neurônio artificial.



Fonte: Elaborada pelo Autor.

A saída de um neurônio é definida pela aplicação de um função de ativação. Algumas das funções de ativação propostas na literatura, são: linear, limiar e sigmoidal (FACELI et al., 2011).

Algoritmos para ajustes de parâmetros, podem ser entendidos como a definição dos pesos relacionados as conexões de uma rede, que permitem com que o modelo preditivo obtenha um desempenho melhor. Esses algoritmos são conhecidos como algoritmos de treinamento e constituem o processo de aprendizado de uma RNA. Ao longo dos anos, diversos algoritmos de treinamento para RNAs foram propostos na literatura (FACELI et al., 2011).

A rede perceptron, foi a primeira RNA a ser construída e foi implementada por (ROSENBLATT, 1958). O treinamento da rede perceptron é feito por um algoritmo supervisionado de correção de erro e a função de ativação usada é a limiar. No decorrer do processo de treinamento, para um objeto x_i, os pesos recebem o ajuste de acordo com a Equação 2.15

$$w_j(t+1) = w_j(t) + nx_i^j(y_i - \widehat{f}(x_i)) \tag{2.15}$$

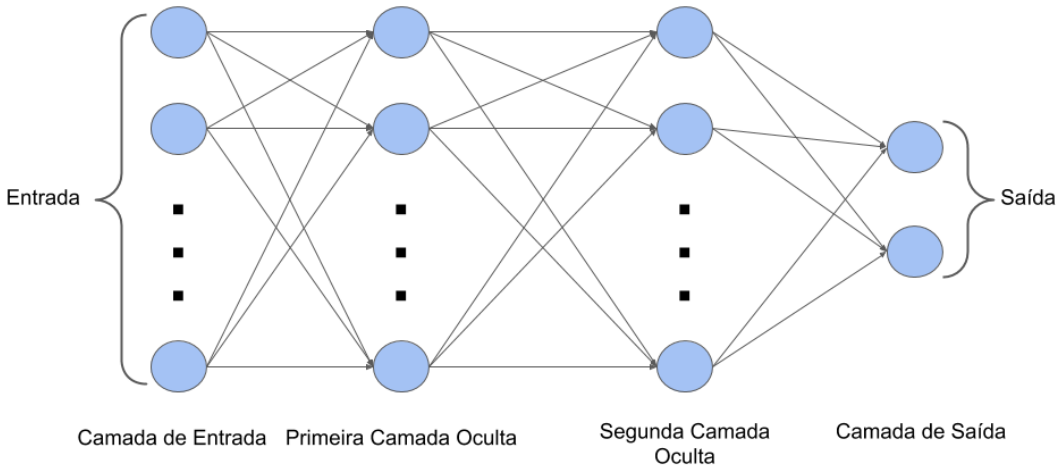
Onde $w_j(t)$ é o peso da j-ésima conexão de entrada no instante de tempo t, n é uma taxa de aprendizado, x_i^j representa o valor do j-ésimo atributo da entrada x_i,

$\widehat{f}(x_i)$ representa a saída produzida pela rede no instante de tempo t e y_i o rótulo de x_i esperado como saída da rede.

Um limitação para redes de uma camada, como o perceptron, é que essas só conseguem classificar problemas que tem enrtradas linearmente separáveis (FACELI et al., 2011).

Multilayer Perceptron (MLP) é uma rede com neurônios dispostos em várias camadas. Os neurônios que recebem diretamente os dados de entrada, compõem a camada de entrada. Os neurônios que recebem como entrada a saída produzida pela camada de entrada, constituem a segunda camada e assim sucessivamente até a última camada, a camada final. As camadas internas, que estão entre as camadas de entrada e saída, são geralmente chamadas de camadas ocultas (KOVÁCS, 2002). A Figura 10 mostra a representação gráfica da estrutura de uma MLP com duas camadas ocultas.

Figura 10 – Estrutura de uma MLP com duas camadas ocultas.



Fonte: Elaborada pelo Autor.

As MLPs surgiram como uma alternativa ao perceptron para solucionar problemas com entradas não linearmente separáveis, pois alternativa mais utilizada para esse problema é adicionar uma ou mais camadas intermediárias. As camadas intermediárias ou ocultas de uma MLP, faz uso de funções de ativação não lineares, como a função sigmoidal (FACELI et al., 2011).

Cada neurônio de uma MLP realiza uma função específica. A função implementada em um neurônio de uma camada específica, é a combinação de todas as funções de neurônios de camadas anteriores que estão ligadas a ela. Conforme os o processamento avança entre camadas, a função de ativação e o processamento realizado se tornam mais complexos. A combinação das funções associadas a cada um dos neurônios da rede, define a função da RNA como um todo.

Cada neurônio da camada de saída, representa uma das classes ou rótulos presentes na base de dados de treinamento. A rede classifica corretamente um objeto, quando o neurônio de saída que produz o valor mais alto, corresponde a classe do objeto. Quando nenhum ou mais de um neurônios produzem valor elevado, a rede não é capaz de classificar a entrada (FACELI et al., 2011).

2.4 Configuração de Parâmetros com GridSearchCV

Cada algoritmo citado nas seções anteriores possuem uma série de parâmetros que podem influenciar no desempenho do modelo preditivo. Dependendo da estrutura organizacional dos dados, diferentes combinações de parâmetros podem obter desempenhos distintos. Técnicas de hiperparametrização consistem na busca de uma combinação de parâmetros que extraem o melhor desempenho para uma determinada base de treinamento (ZHENG, 2015).

Na literatura existem diferentes técnicas para encontrar a melhor combinação de parâmetros para uma determinada base de dados. As técnicas de hiperparametrização vão desde a seleção manual de parâmetros, até métodos mais sofisticados que fazem a busca de maneira automática pela combinação ideal de parâmetros (ZHENG, 2015).

Para este trabalho, adotou-se a técnica de *Grid Search* para encontrar configuração de parâmetros que permite obter o melhor desempenho para as técnicas exploradas na pesquisa. O *Grid Search* consiste em montar uma *grid* de parâmetros definidos por um especialista e combina-los, avaliando cada uma das combinações com base em uma métrica de desempenho. Ao final do processo, o algoritmo retorna combinação que obteve o melhor desempenho do modelo preditivo.

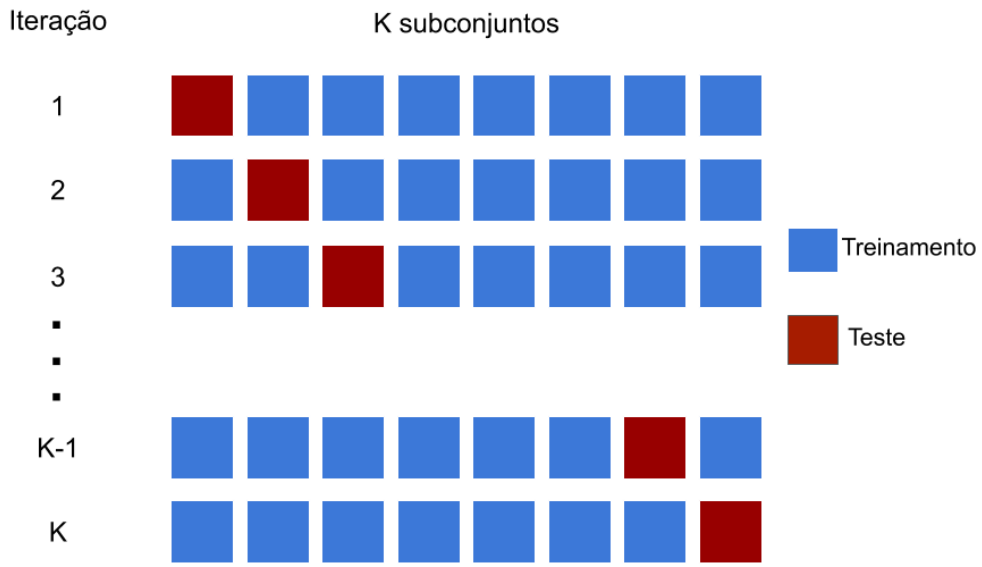
Nesta pesquisa, adotou-se a implementação *GridSearchCV* disponível na biblioteca do *Python*[3], *scikit-learn*[4]. O *GridSearchCV* usa a validação cruzada, apresentada na Seção 2.5.1, no processo de busca por melhores parâmetros (PEDREGOSA et al., 2011).

2.5 Avaliação dos Modelos Preditivos

A validação de um modelo preditivo obtido a partir da utilização de técnicas de AM, geralmente envolve a execução de experimentos dentro de um ambiente controlado. Durante esses experimentos são adotadas diversos artifícios para avaliar o

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

Figura 11 – Representação do processo de validação cruzada.



Fonte: Elaborada pelo Autor.

desempenho de um modelo. É o caso das métricas de avaliação e das técnicas de amostragem.

2.5.1 Validação Cruzada

Técnicas de amostragem consistem em dividir os dados de entrada em subconjuntos de treinamento e teste com o objetivo de obter estimativas mais confiáveis. Validação cruzada é um método de amostragem. A validação cruzada é comumente utilizada com o método *K-Fold*. Este método divide a base em treino e teste aleatoriamente e refaz esse processo K vezes (AGUIAR, 2017). A validação cruzada permite avaliar a capacidade de generalização dos modelos gerados. Capacidade de generalização de um modelo preditivo é conhecida pela habilidade que este tem de prever um determinado evento, mesmo que a entrada fornecida nunca tenha sido apresentada ao modelo antes. A Figura 11 apresenta o processo de validação cruzada com um determinado K.

2.5.2 Métricas de Avaliação

As métricas de avaliação permitem obter parâmetros sobre o desempenho dos modelos preditivos. Dessa forma, através da análise de um conjunto de métricas dentro de uma série de experimentos controlados, é possível escolher o modelo preditivo que melhor soluciona o problema abordado.

Tabela 1 – Exemplo de matriz de confusão.

		Predição	
		C	¬C
Real	C	VP	FN
	¬C	FP	VN

Fonte: Elaborada pelo Autor.

A matriz de confusão é um artifício usado para descrever os resultados em um problema de classificação. A Tabela 1 mostra um exemplo de matriz de confusão.

A Tabela 1 demonstra um problema de classificação, onde uma instância pode pertencer ou não a classe C, sendo ¬C a representação de não pertencimento a classe em questão. As linhas da tabela representam quantos elementos pertencem, ou não, a classe C. As colunas exibem quantas instâncias foram classificadas como pertencentes, ou não, a classe C (CARVALHO et al., 2018).

As siglas usadas na Tabela 1 representam as quantidades de instâncias que foram classificadas de maneira correta ou incorreta. A seguir, é feito uma descrição de cada sigla.

- Verdadeiros Positivos (**VP**): Representam a quantidade de instâncias pertencentes a classe C que foram classificadas como pertencendo à classe C;
- Falsos Positivos (**FP**): Representam a quantidade de instâncias não pertencentes a classe C que foram classificadas como pertencendo à classe C;
- Verdadeiros Negativos (**VN**): Representam a quantidade de instâncias que não pertencem à classe C e foram classificadas como não pertencendo à classe C;
- Falsos Negativos (**FN**): Representam a quantidade de instâncias pertencentes a classe C que foram classificadas como não pertencentes a classe C;

Os autores de (FACELI et al., 2011) citam as principais métricas para problemas de classificação: acurácia, precisão, sensibilidade (*recall*) e *F1-Score*. A seguir é feito uma breve explicação sobre cada uma das métricas citadas.

- **Acurácia**: É a taxa de acerto do classificador. Pode ser definida pela Equação 2.16:

$$ac = \frac{VP + VN}{VP + FP + VN + FN} \tag{2.16}$$

- **Precisão**: É a dimensão de instâncias positivas classificadas corretamente detre todos os positivos (AGUIAR, 2017). Pode ser definida pela Equação 2.17:

$$prec = \frac{VP}{VP + FP} \tag{2.17}$$

- **Sensibilidade**: É a taxa de acerto nas instâncias da classe positiva (AGUIAR, 2017). Pode ser definida pela Equação 2.18:

$$sens = \frac{VP}{VP + FN} \tag{2.18}$$

- **F1-Score**: É a média harmônica entre precisão e sensibilidade. Pode ser definida pela Equação 2.19:

$$F1 = \frac{2 * prec * sens}{prec + sens} \tag{2.19}$$

3 TRABALHOS RELACIONADOS

Nos últimos anos, pesquisadores tem aplicado inúmeras técnicas de AM para desenvolver soluções que melhorem o processo de ensino-aprendizado. Em seguida, serão explorados alguns desses trabalhos existentes na literatura.

(SANTOS; SILVEIRA; LECHUGO, 2016) apresenta um modelo computacional conceitual da aplicação de técnicas de Mineração de Dados Educacionais, com ênfase na Análise de Sentimentos, em uma Avaliação Institucional Docente. O modelo proposto realiza a captação dos dados através de um formulário eletrônico, no qual o aluno tem a opção de falar "bem" ou "reclamar" de seus professores. A partir da captura das respostas, inicia-se o processo de mineração desses dados. Nesta etapa, é realizado um processo de extração de informações que identifica categorias presentes nas afirmações dos alunos. Finalmente, as informações extraídas serão exibidas de forma mais amigável através de ferramentas de visualização de dados, para gestores e docentes. Vale ressaltar que, a pesquisa encontra-se em estado de desenvolvimento, portanto, o autor não publicou resultados concretos a respeito de seu trabalho.

(RANI; KUMAR, 2017) explora um sistema baseado em AS que visa melhorar o processo de ensino-aprendizado através da extração de informações em comentários multilíngues presentes na plataforma de cursos *Coursera*. O sistema proposto foi dividido em 5 componentes principais: coleta de dados, pré-processamento de dados, identificação de sentimentos e emoções, computação de satisfação e insatisfação e visualização de dados. O *corpus* usado consiste em *feedback* de alunos a respeito de cursos presentes na plataforma *Coursera*. Inicialmente, foram coletados cerca de 4.000 *reviews* do período de Agosto de 2015 até Agosto de 2016. Os cometários coletados foram feitos durante o período de curso. Posteriormente, foram coletados mais 1.700 avaliações realizadas após o período de curso. Depois da etapa de coleta de dados, foi usada a linguagem R para tratar o pré-processamento de dados e a classificação de sentimentos. Durante a etapa de pré-processamento, foram realizadas as etapas de: tokenização, conversão para minúsculas, normalização, *stemming*, remoção de conteúdo irrelevante e tradução de sentenças. Na etapa de identificação de sentimento e emoção, foi usado o *NRC Emotion Lexicon* ou *Emolex* [1], para associar palavras com sentimento positivo ou negativo.

Para validar a proposta, (RANI; KUMAR, 2017) analisou comentários feitos por estudantes em um Sistemas de Resposta Estudantil universitário para 25 cursos diferentes ao longo de dois anos. Foi comparada a porcentagem de sentimento posi-

[1] https://saifmohammad.com/WebPages/NRC-Emotion-Lexicon.htm

tivo nos comentários dos alunos a respeito de determinado curso com a nota média deste mesmo curso em uma escala de 0-100. Ao final das avaliações, notou-se que a diferença foi de menos de 20% entre os métodos. Apesar de promissor, o sistema apresenta algumas limitações. A precisão do modelo depende da qualidade das avaliações coletadas. Por isso, é preciso ter cuidado com a coleta das opiniões dos alunos. Os SREs devem ser bem projetados e deve haver um empenho de professores e gestores para que o maior número possível de alunos forneça um *feedback* completo e preciso. Nos trabalhos futuros o autor pretende criar uma Interface de Programação de Aplicações (API do inglês) para comunicar o sistema proposto com SREs e plataformas de aprendizado online para permitir um análise em tempo real dos comentários dos alunos.

(BALAHADIA; FERNANDO; JUANATAS, 2016) produziu uma pesquisa que objetiva o desenvolvimento de uma ferramenta de avaliação de desempenho do professor usando a mineração de opinião com análise de sentimentos. O estudo pode ajudar a identificar os pontos fortes e fracos dos membros do corpo docente com base no *feedback* positivo e negativo dos alunos em inglês ou em filipino. Inicialmente é realizada a coleta de avaliações dos alunos em classificação numérica e escala qualitativa usando uma ferramenta *online* de avaliação do professor. Após a coleta dos comentários dos alunos, será realizado um processo de pré-processamento dos dados. Nesta etapa, é realizado transformações sobre as sentenças a fim de remover caracteres indesejados que não afetam na classificação do texto.

Para realizar o pré-processamento, (BALAHADIA; FERNANDO; JUANATAS, 2016) envolveu os seguintes processos: 1) Converter todos os texto para letras minúsculas; 2) Remoção da pontuação; 3) Remoção de números; 4) Remoção de espaços em branco; 5) Remoção das *stopwords*; 6) Remoção de sufixos das palavras. Em seguida, as frases serão submetidas a um modelo baseado no algoritmo de ML *Naïve Bayes*. Nesta fase, será atribuída uma polaridade “positiva”, “negativa” ou “neutra” para cada avaliação presente no banco de dados. Finalmente, após o resultado da classificação, será exibido através ferramentas de visualizações os resultados das avaliações docentes para gestores e professores. Cada professor só terá acesso as suas avaliações. Futuramente, os pesquisadores pretendem adicionar um verificador de gramática e ortografia para validar as palavras presentes nos *feedback's* fornecidos pelos alunos. Também pretende-se fazer uma experimentação com outros algoritmos além do *Naïve Bayes* com a finalidade de encontrar um modelo mais preciso e robusto.

Em (AZEVEDO et al., 2017) é proposto a utilização de Análise de Sentimentos para prevenir a evasão nos cursos de Educação a Distância (EAD). Em resumo, o trabalho propõe uma técnica baseada em AS para o português, com o objetivo de identificar a motivação do aluno a partir de sua iteração em um fórum educacional.

Desta forma, a partir do resultado retornado, o professor pode buscar alternativas para motivar o aluno novamente, na tentativa de evitar sua evasão. São usados dois bancos de dados para avaliar a técnica. o primeiro formado por postagens geradas artificialmente e o segundo faz uso de postagens de diversas postagens turmas de um curso de Ciência da Computação. A técnica proposta identifica a polaridade da motivação do aluno como positiva ou negativa. É considerado a polaridade positiva com o aluno motivado e negativa para o desmotivado (que tem risco de evasão). A polaridade é obtida a partir de 4 etapas: pré-processamento, distribuição de pesos, tratamento da negação e classificação. Ao final de todas as etapas, a técnica mostrou uma taxa de acerto de 82% na classificação da motivação dos estudantes.

Analisando a literatura existente, percebe-se a importância do desenvolvimento da pesquisa proposta por esse trabalho. Apesar da grande relevância do tema, pode-se notar que ainda há uma carência de trabalhos relacionados a avaliação de docentes usando AS, principalmente quando o assunto é língua portuguesa. A maioria dos trabalhos existentes não apresentam resultados claros e concretos.

4 PROPOSTA

Neste Capítulo, é apresentada a proposta deste trabalho que consiste na aplicação de técnicas de AM para realização de análise de sentimentos em avaliações docentes. O objetivo é auxiliar professores e gestores educacionais em suas decisões pedagógicas. Também são apresentadas todas as etapas seguidas para concepção da pesquisa.

Durante o decorrer da pesquisa, foi desenvolvido um sistema baseado em análise de sentimentos para auxiliar professores no momento de selecionar quais metodologias adotar em sala de aula. Desta maneira, o objetivo da solução proposta é tornar o processo de ensino e aprendizagem mais dinâmico e eficiente, pois como citado nos Capítulos anteriores, aplicar AS nas avaliações docentes torna o processo de extração de informações desses *feedback's* mais rápido. A solução proposta possui dois fluxos de trabalho. O primeiro é a construção de um módulo inteligente para análise de sentimentos, e o segundo é a proposta de uma PoC web para avaliação docente.

O módulo inteligente faz uso de um modelo preditivo que tem o intuito de realizar a classificação de polaridade de avaliações docentes em uma das seguintes categorias: “Insatisfatório”, “Neutro” ou “Satisfatório”. A construção do modelo inteligente se deu pela realização de uma série de etapas apresentadas na Figura 12. Foi construída uma base com avaliações de alunos a respeito da satisfação dos mesmos em relação a suas aulas. Essa base foi submetida a uma série de técnicas de AM, gerando um conjunto de modelos inteligentes. Cada um dos modelos preditivos gerados foi avaliado a partir de um conjunto de métricas de avaliação de desempenho, comuns na literatura em análise de sentimentos. Por fim, o modelo que apresentou o melhor desempenho foi selecionado para compor o módulo inteligente. Cada uma das etapas realizadas é explicada de maneira mais detalhada ao longo da Seção 4.1.

A segunda etapa desse trabalho é a proposta de uma PoC web para avaliação docente. Com a escolha do melhor modelo inteligente, é proposto uma Interface de Programação de Aplicativos (do inglês *Application Programming Interface* - API) baseada no paradigma REST. O objetivo da API é receber uma nova avaliação docente, submeter esta ao modelo inteligente, e depois retornar ao cliente o rótulo associado aquele *feedback*. Os Códigos 4.1 e 4.2 mostram o pseudo formato de uma requisição e reposta HTTP na API que contém o modelo inteligente para classificar uma nova avaliação docente.

Código 4.1 – Pseudo requisição HTTP para submeter uma avaliação ao modelo inteligente.

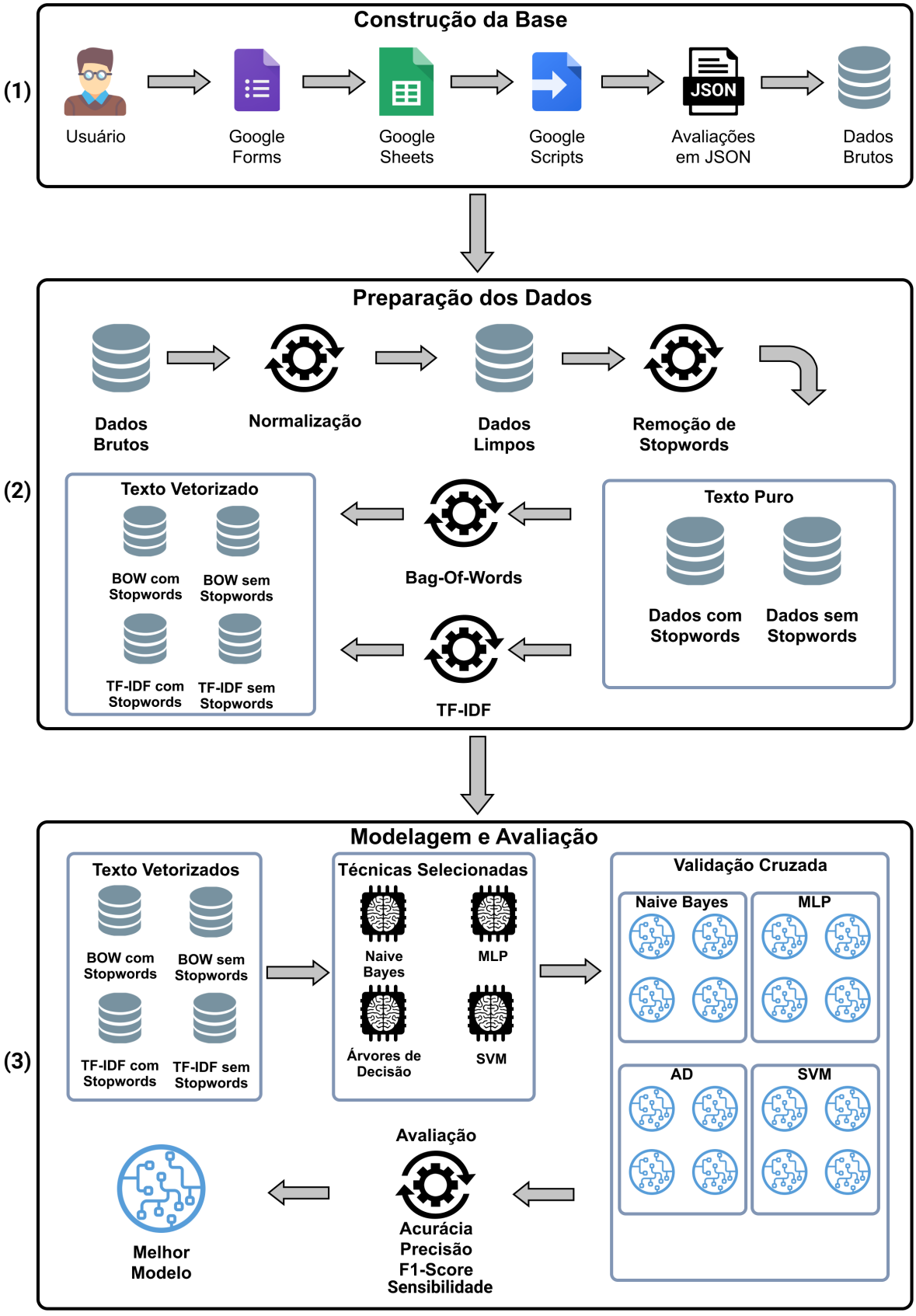
```
POST http://<server>:5001/predict
{
    "data": "A aula foi legal.",
    "model": "TfIdf-SVM",
}
```

Código 4.2 – Pseudo resposta HTTP com a classe predita para avaliação submetida.

```
Response: [{'class': 'satisfatorio'}]
```

Para testar a API e o módulo inteligente, é proposto um cliente web para avaliação docente. Nesse cliente, é permitido que um usuário faça a avaliação de determinada aula. A avaliação produzida é submetida a API que faz uso do módulo inteligente. A satisfação atribuída àquela avaliação pelo modelo inteligente é exibida ao usuário. Na Seção 4.4 a PoC Web é apresentada com mais detalhes.

Figura 12 – Etapas para o desenvolvimento do modelo inteligente.



Fonte: Elaborada pelo Autor.

4.1 Modelo Inteligente

Ao longo dessa seção, são explicadas todas as etapas seguidas para construção e seleção do melhor modelo preditivo para classificação de avaliações docentes de maneira automática. Então, nessa seção são explicadas as fases de: construção da base, onde ocorreu a criação de uma base contendo avaliações docentes e as respectivas polaridades associadas a estas; preparação dos dados, onde foram realizados processos com o intuito de melhorar a qualidade dos dados; por fim, a fase de modelagem e avaliação, onde os dados foram submetidos a algoritmos de Aprendizado de Máquina e através da avaliação perante métricas de desempenho foi selecionado o modelo com melhor desempenho.

Durante todo o processo de construção do modelo inteligente, foi usada a linguagem *Python* bem como algumas bibliotecas presentes na mesma. A escolha do *Python* se deu pela sua facilidade de codificação e por seu grande número de bibliotecas que facilitam o processo de aprendizado.

4.1.1 Construção da Base de Dados

Parte inicial do processo de criação do modelo preditivo inteligente foi a etapa da construção da base de treinamento. Inicialmente, foram realizadas buscas por bases abertas em português relacionadas ao contexto da pesquisa. Contudo, o resultado não foi satisfatório, pois a maioria das bases encontradas por meio dos trabalhos relacionados pertencem a outras línguas que não são a portuguesa. Os poucos trabalhos para língua portuguesa, não deixaram claro o local de acesso para as bases utilizadas em suas pesquisas.

Por esses motivos, optou-se pela construção uma base própria para realizar o trabalho. Para construção da base, foi usada uma estratégia inspirada no trabalho de (AZEVEDO et al., 2017). A estratégia consiste em gerar a base de treinamento de dados de maneira artificial. Ou seja, um grupo de pessoas foi selecionado, e esses tinham o objetivo de criar avaliações para simular um aluno que acabou de assistir uma aula e deixou um *feedback* a respeito da mesma.

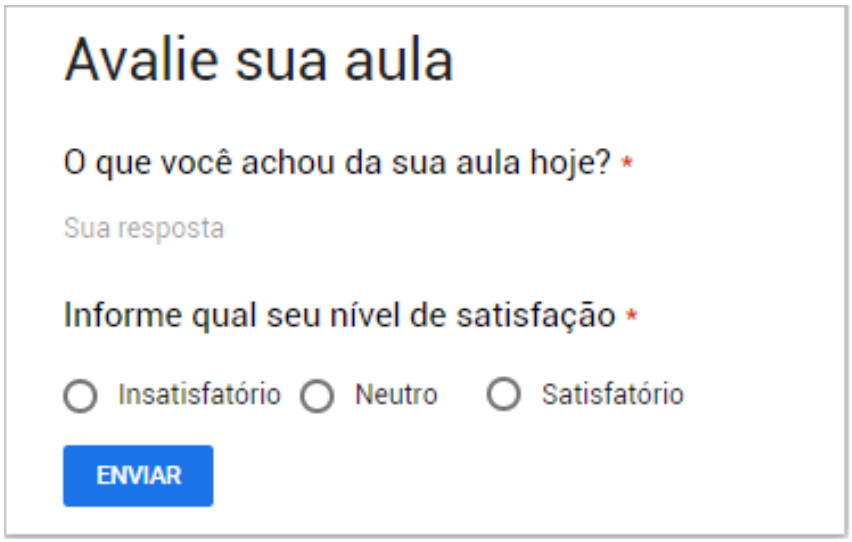
O grupo foi formado por um total de cinco pessoas, todos estudantes do curso de Ciência da Computação do Instituto Federal de Ciência e Tecnologia do Ceará (IFCE). A escolha desses, se deu pelo fato de todos estarem inseridos em um ambiente de ensino, tendo aulas constantemente, o que facilitou a criação das avaliações fictícias por meio dos mesmos. As avaliações foram coletadas no período de 20/08/2019 à 08/02/2020. Em seguida, são explicados a forma como as avaliações foram coletadas até a maneira como ela foi exportada para ser usada na construção

do modelo inteligente proposto no trabalho.

Durante todo processo de criação da base, foram utilizadas as ferramentas do *google*: *forms*, *sheets* e *app scripts*. A escolha dessas ferramentas se deu pelo fato das mesmas serem fáceis de manipular e da abstração de problemas como a hospedagem. Além do mais, para recursos usados neste trabalho, todas as ferramentas foram gratuitas.

Inicialmente, foi criado um formulário no *google forms* para coletar as novas avaliações. O formulário é composto por duas perguntas. A primeira é "O que você achou da sua aula hoje?", que está em forma de campo de texto aberto, para que os usuários possam colocar sua avaliação subjetiva a respeito de determinada aula. O outro campo, é formado pela pergunta "Informe qual seu nível de satisfação", que recebe como resposta um campo de seleção única, que pode ser preenchido com os valores "Insatisfatório", "Neutro" ou "Satisfatório", cada uma dessas equivale às polaridades: "Negativo", "Neutro", "Positivo" respectivamente. A Figura 13 mostra o formulário usado.

Figura 13 – Formulário usado para aquisição de avaliações rotuladas.



Fonte: Elaborada pelo Autor.

A ideia do formulário com duas perguntas é simples. A segunda pergunta serve como rótulo ou classe para primeira, pois, como mostrado na Seção 2.3 problemas de aprendizagem supervisionada (caso deste trabalho) necessitam de um conjunto de treinamento formado tuplas (x_i, y_i) onde y_i é um rótulo para o vetor de atributos x_i. Então, neste trabalho, o conjunto de treinamento é formado por tuplas do formato $(\text{Avaliação}, \text{Satisfação})$ as quais representam a primeira e segunda pergunta do formulário respectivamente.

As avaliações coletadas no formulário, ficam automaticamente salvas em uma planilha no *google sheets*, a ferramenta de planilhas do *google*. Com os dados salvos

em uma planilha, foi usado a ferramenta *google app scripts* [1] para manipular os dados da mesma. O *google app scripts* é uma ferramenta que permite trabalhar e manipular de maneira simples as outras plataformas do *google* como formulários, planilhas, etc. Dessa forma, as avaliações coletadas foram importadas, transformadas em *javascript object notation* (JSON) e disponibilizadas como uma API. Assim, nas próximas etapas para gerar o modelo inteligente, que são explicadas nas seções seguintes, ficou mais fácil importar as avaliações coletadas, pois bastou realizar uma requisição HTTP no *app script* criado, que todos os *feedback's* coletados foram importados. A parte (1) da Figura 12 representa todo fluxo seguido para a construção da base de treinamento.

Ao final do período de coleta, um total de 706 avaliações foram obtidas. O conjunto de dados construído é apresentado na Tabela 2, na qual pode ser observado a proporção das avaliações para cada uma das categorias consideradas.

Tabela 2 – Quantidade de avaliações coletadas por classe.

	Classes	**Quantidade**
	Insatisfatório	241
	Neutro	230
	Satisfatório	235
Total		706

Fonte: Elaborada pelo Autor.

A etapa de construção da base de dados é seguida pela etapa de preparação dos dados. Nesta, os processos explicados na Seção 2.2.3 foram aplicados, com o objetivo de melhorar a qualidade dos dados de entrada e consequentemente obter um modelo inteligente com o melhor desempenho possível. A seguir, são explicados de maneira detalhada como cada um dos processos de preparação de dados foi realizado durante o decorrer do trabalho.

4.1.2 Preparação dos Dados

Após a construção da base, deu-se início a fase de preparação dos dados antes. A fase preparação de dados foi dividida em duas etapas. A primeira é o pré-processamento dos dados textuais e a segunda é a etapa de representação de palavras. Em seguida, é explicado como estes dois processos foram realizados durante o decorrer do trabalho.

[1] https://www.google.com/script/start/

4.1.2.1 Pré-Processamento

Na etapa de pré-processamento, foram realizados os procedimentos de normalização e remoção de *stopwords*, explicados na Seção 2.2.3. O objetivo desta etapa é tratar os dados da base contendo avaliações docentes, com o intuito de melhorá-los para assim obter um modelo preditivo com o melhor desempenho possível.

O primeiro passo foi, a partir dos "dados brutos" produzidos na fase de construção da base de treinamento explicada na Seção 4.1.1, realizar o processo de normalização. O primeiro passo da etapa de normalização foi deixar todas as avaliações coletadas em letras minúsculas.

Após a conversão das avaliações para letras minúsculas, foram usadas expressões regulares, através da biblioteca re[2] do *Python*, para remover quebras de linhas e caracteres especiais do texto. Esse processo torna-se necessário, pois, caracteres como **-+*/"#&!?**, podem influenciar negativamente no processo de análise do texto. O Código 4.3 representa a função em *Python* usada para fazer a normalização do texto.

Código 4.3 – Pseudo código em *Python* para representar o processo de normalização do texto.

```
import re
def limpa_texto(texto):
    texto = texto.lower()
    texto = re.sub("[-+*/''()#\@$!?,;%&:~<>=|.${}]", "", texto)
    texto = re.sub("\n", " ", texto)
    texto = re.sub('"', "", texto)
    if texto == '[': texto = ""
    if texto == ']': texto = ""
    return texto
```

O processo de normalização das frases é uma etapa extremamente essencial. Por exemplo, na hora de submeter o texto ao treinamento nos algoritmos de AM, as palavras "excelente", "excelente!", "@excelente", "Excelente" serão todas tratadas como *features* diferentes pelo algoritmo. Com o termino da normalização, foi gerada a base de avaliações com os "dados limpos".

Com os "dados limpos", foi realizado o processo de remoção de *stopwrods* explicado na Seção 2.2.3.3. Nessa etapa, uma série de palavras que não influenciam de maneira significativa no sentimento do texto foram removidas. Para realizar o processo

[2] https://docs.python.org/3/library/re.html

de remoção de *stopwords* é necessária uma lista com um conjunto de *stopwords* na língua do texto usado como base de treinamento. No caso deste trabalho, o português. A lista usada neste trabalho foi a disponível na biblioteca do *Python*, *Natural Language Toolkit*[3] (NLTK) que é bastante popular em tarefas que envolvem PLN.

A lista de *stopwords* da NLTK possui um total de 203 palavras. Ao longo dos testes preliminares usando a base sem *stopwords* como base de treinamento, notou-se uma queda considerável na precisão da classe "Insatisfatório". Assim, foi feita uma análise nesta lista e notou-se que a palavra "não" pertencia à lista de *stopwords*. Pode-se verificar que a palavra "não" tem grande impacto sobre a negatividade de uma frase. Por exemplo, considere a frase "a aula não foi boa". Ao remover a palavra não, a nova frase será "a aula foi boa", o que remove o sentido de negatividade da frase. Também observou-se que ao longo de toda base existia 141 ocorrências da palavra "não" ao qual 106 estavam presente nas avaliações "Insatisfatório". Assim, percebeu-se que a palavra "não" encontra-se fortemente atrelada ao sentimento de "Insatisfatório". Por conta disso, decidiu-se remover a palavra "não" da lista original de *stopwords* resultando em uma nova lista com 202 palavras.

Ao final da etapa de pré-processamento, decidiu-se gerar duas bases de dados. Uma com as avaliações originais sem a remoção de *stopwords* e outra contendo as avaliações sem a presença de *stopwords*. O intuito disso é verificar o impacto de desempenho causado pela utilização ou não de uma lista de *stopwords* no problema de AS explorado nesta pesquisa. As duas bases serviram de entrada para o processo de representação de palavras que é explicado ao longo da próxima seção.

4.1.2.2 Representação de Palavras

Com os dados pré-processados, deu-se início a fase de representação de palavras. Nessa etapa, são usadas técnicas para transformar os "textos brutos" em um formato numérico ou vetorizado. Esse processo torna-se necessário dado que os algoritmos de AM não trabalham com texto em seu formato puro. Para realizar a representação de palavras neste trabalho, foram usadas duas técnicas comumente utilizadas na literatura em PLN para representação de palavras. A primeira foi o *Bag-Of-Words*, a qual é explanado com detalhes na Seção 2.2.4.1. E a segunda é o TF-IDF que é detalhado na Seção 2.2.4.2.

Para implementar a representação de palavras foram usadas duas classes presentes na biblioteca *scikit-learn* (PEDREGOSA et al., 2011): o *CountVectorizer*[4] e

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

[4] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

Tabela 3 – Descrição de cada uma das base de dados geradas na etapa de preparação dos dados.

Base de dados	Descrição
tf_idf_sem_stop	Base sem stopwords representada com TF-IDF
tf_idf_com_stop	Base com *stopwords* representada com TF-IDF
bow_sem_stop	Base sem *stopwords* representada com *Bag-Of-Words*
bow_com_stop	Base com *stopwords* representada com *Bag-Of-Words*

Fonte: Elaborada pelo Autor.

o *TfidfVectorizer*[5] que são usadas para representar o texto com as técnicas de representação de palavras BOW e TF-IDF, respectivamente. A escolha de utilização destas classes se deu pela abstração de complexidades de implementação proporcionadas por ambas.

Como na fase de pré-processamento foram geradas duas bases (com *stopwords* e sem *stopwords*), cada uma das bases foi submetida a uma das duas técnicas de representação de palavras presentes no trabalho. Assim, ao final desta etapa foram geradas quatro bases vetorizadas e prontas para serem usadas como base de treinamento das técnicas de AM exploradas no trabalho. A Tabela 3 mostra a nomenclatura atribuída a cada uma das bases de dados, bem como a descrição delas.

O Código 4.4 mostra um pseudo código baseado na linguagem *Python* que representa todos os processos implementados e seguidos durante a fase de preparação dos dados.

Código 4.4 – Pseudo código em *Python* para representar os processos realizados durante a etapa de preparação dos dados.

```
from sklearn import TfidfVectorizer
from sklearn import CountVectorizer

# Pre-processando e removendo stopwords
dados_limpos = pre_processamento(dados_brutos)

dados_sem_stop = remocao_stop(dados_limpas)
dados_com_stop = dados_limpos

# Representação com TF-IDF
tf_idf_sem_stop = TfidfVectorizer(dados_sem_stop)
tf_idf_com_stop = TfidfVectorizer(dados_com_stop)
```

[5] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

```
# Representação com BOW
bow_sem_stop = CountVectorizer(dados_sem_stop)
bow_com_stop = CountVectorizer(dados_com_stop)
```

Após representar cada uma das bases, com e sem *stopwords*, verificaram-se as dimensões de todas elas. Essa informação é importante, pois, alguns algoritmos como o SVM têm tendências a ter desempenhos melhores quando aplicados a bases com muitas colunas e poucas linhas (JOACHIMS, 1998) (GOUDJIL et al., 2018).

A Tabela 4 apresenta as dimensões das quatro bases de dados. Pode-se perceber que os valores das linhas nunca mudam, pois, todas as bases têm um total de 706 instâncias. Os valores das colunas são alterados apenas nas bases que tem ou não *stopwords*. Esse fato ocorre, pois, quando o texto é representado com TF-IDF ou BOW é criado uma matriz com a quantidade de colunas equivalente ao total de palavras distintas no corpus. Contudo, as implementações *CountVectorizer* e *TfidfVectorizer* trabalham com o conceito de matriz esparsa para otimizar o uso da memória (PEDREGOSA et al., 2011).

Tabela 4 – Dimensões das quatro bases depois de vetorizadas.

Base	Linhas	Colunas
tf_idf_sem_stop	706	657
tf_idf_com_stop	706	726
bow_sem_stop	706	657
bow_com_stop	706	726

Elaborada pelo Autor.

Na parte (2) da Figura 12, é sumarizado todo fluxo seguido durante a preparação dos dados, desde a entrada dos dados brutos até às quatro bases representadas de forma numéricas prontas para passar pelo treinamento perante as técnicas de AM com intuito de criar modelos preditivos inteligentes para classificação de avaliações docentes.

4.1.3 Modelagem e Avaliação

Com os dados pré-processados e representados numericamente, deu-se início ao processo de avaliação dos modelos preditivos inteligentes. As bases vetorizadas foram submetidas aos algoritmos de AM comuns na literatura, explicadas na Seção 2.3, para a tarefa de classificação de texto. No caso, as técnicas selecionadas foram: *Naive Bayes* detalhado na Seção 2.3.2, as Árvores de Decisão explanadas na Seção 2.3.1, o SVM explicado na Seção 2.3.3 e por fim o MLP explanado na Seção 2.3.4.

Foram escolhidas as implementações disponíveis na biblioteca *scikit-learn* do Python. A escolha da *scikit-learn* se deu por esta oferecer grande facilidade e eficiência em tarefas de predição e análise de dados (PEDREGOSA et al., 2011).

Cada uma das quatro bases geradas na etapa de preparação de dados passou processo de busca por hiperparâmetros com *GridSearchCV*. Em seguida foram realizados experimentos combinando às quatro bases e às quatro técnicas de AM para o problema de classificação de texto explorado no trabalho. Assim, foram avaliados um total de dezesseis modelos preditivos inteligentes. A Tabela 5 mostra a nomenclatura atribuída a cada um dos modelos avaliados e a descrição (base de dados e algoritmo) de cada um deles.

Tabela 5 – Descrição de cada modelo preditivo usado na pesquisa.

Base de dados	Descrição
svm_tf_idf_sem_stop	SVM treinado com a base sem stopwords representada com TF-IDF
svm_tf_idf_com_stop	SVM treinado com a base com stopwords representada com TF-IDF
svm_bow_sem_stop	SVM treinado com a base sem stopwords representada com Bag-Of-Words
svm_bow_com_stop	SVM treinado com a base com stopwords representada com Bag-Of-Words
mlp_tf_idf_sem_stop	MLP treinado com a base sem stopwords representada com TF-IDF
mlp_tf_idf_com_stop	MLP treinado com a base com stopwords representada com TF-IDF
mlp_bow_sem_stop	MLP treinado com a base sem stopwords representada com Bag-Of-Words
mlp_bow_com_stop	MLP treinado com a base com stopwords representada com Bag-Of-Words
ad_tf_idf_sem_stop	AD treinada com a base sem stopwords representada com TF-IDF
ad_tf_idf_com_stop	AD treinada com a base com stopwords representada com TF-IDF
ad_bow_sem_stop	AD treinada com a base sem stopwords representada com Bag-Of-Words
ad_bow_com_stop	AD treinada com a base com stopwords representada com Bag-Of-Words
nb_tf_idf_sem_stop	NB treinado com a base sem stopwords representada com TF-IDF
nb_tf_idf_com_stop	NB treinado com a base com stopwords representada com TF-IDF
nb_bow_sem_stop	NB treinado com a base sem stopwords representada com Bag-Of-Words
nb_bow_com_stop	NB treinado com a base com stopwords representada com Bag-Of-Words

Fonte: Elaborada pelo Autor.

4.2 Definição de Hiperparâmetros

Foi realizado o processo de seleção de hiperparâmetros para cada técnica de AM explorada no trabalho. Para selecionar os hiperparâmetros foi usada a classe *GridSearchCV* da *scikit-learn*, apresentada na Seção 2.4. O uso dessa técnica consiste em selecionar um conjunto de parâmetros específicos para cada algoritmo de AM. Esses parâmetros selecionados são submetidos a cada algoritmo através do uso de uma *cross-validation* com o número de partições igual a 10, parâmetro comumente

utilizado na literatura em experimentos de AM, usando todo o conjunto de dados para realizar o treinamento e retornar a melhor combinação encontrada. A seguir é feito um detalhamento da configuração de parâmetros usada para cada algoritmos de AM utilizado nesta pesquisa.

O primeiro algoritmo a ser utilizado foi o *Naive Bayes*. Essa foi a única das quatro técnicas que não precisou passar pelo processo de seleção de hiperparâmetros com *GridSearchCV*, pois, essa não necessita de nenhum parâmetro para ser treinada com as bases de dados. O uso do NB nessa pesquisa se deu pelo uso da implementação da *scikit-learn*, *MultinomialNB*[6]. O MultinomialNB é uma variação do NB clássico ideal para trabalhar com classificação de características discretas, por exemplo, a classificação de texto. O Código 4.5 mostra como o *MutinomialNB* foi usado com cada uma das quatro bases presentes na pesquisa.

Código 4.5 – Utilização do MultinomialNB na pesquisa.

```
from sklearn.naive_bayes import MultinomialNB
''' Modelo treinado com NB e a base sem stopwords representada
    com TF-IDF '''
nb_tf_idf_sem_stop = MultinomialNB()
nb_tf_idf_sem_stop.fit(tf_idf_sem_stop, classes)
''' Modelo treinado com NB e a base com stopwords representada
    com TF-IDF '''
nb_tf_idf_com_stop = MultinomialNB()
nb_tf_idf_com_stop.fit(tf_idf_com_stop, classes)
''' Modelo treinado com NB e a base sem stopwords representada
    com BOW '''
nb_bow_sem_stop = MultinomialNB()
nb_bow_sem_stop.fit(bow_sem_stop, classes)
''' Modelo treinado com NB e a base com stopwords representada
    com BOW '''
nb_bow_com_stop = MultinomialNB()
nb_bow_com_stop.fit(bow_com_stop, classes)
```

O método *fit* representa a função que realiza o treinamento do conjunto de dados perante o algoritmo de AM. A função pode ser usada no formato $.fit(x, y)$, onde x é o conjunto de dados de treinamento, no caso desta pesquisa uma das quatro representações com a base de avaliação docente, e y representa o conjunto de classes relacionadas a base de treinamento, no caso desse trabalho a satisfação atribuída as

[6] https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html

Tabela 6 – Grid usada no processo de seleção de parâmetros com *GridSearchCV* para o *DecisionTreeClassifier*.

Parâmetro	Descrição
criterion	Função para medir a qualidade de uma divisão. É a métrica essencial para a construção da árvore de decisão.
splitter	Representa estratégia utilizada para escolher a divisão em cada nó.
max_depth	É a profundida máxima da árvore. Profundidade demais pode gerar um sistema super especializado nos dados de treinamento. Pouca profundidade diminuirá a capacidade de generalização do modelo.

Fonte: Elaborada pelo Autor.

avaliações (Insatisfatório, Neutro, Satisfatório). Essa configuração do método *fit* se aplica a todas as técnicas explicadas nesse trabalho.

Para o treinamento com Árvores de Decisão, foi usado a classe da *scikit-learn*, *DecisionTreeClassifier*[7]. O *DecisionTreeClassifier* é ideal para realizar classificação multicitasses que é quando existem mais de dois atributos alvos, como é o caso deste trabalho. Foi montada uma *grid* de parâmetros com o objetivo de achar a melhor combinação para o *DecisionTreeClassifier* e às quatro bases com *GridSearchCV*. A Tabela 6 mostra o conjunto de parâmetros selecionados para montar a *grid* de busca por melhores parâmetros para o *DecisionTreeClassifier* e suas respectivas descrições.

O Código 4.6 ilustra o uso das quatro bases com o algoritmo de AD selecionado, bem como os parâmetros que foram selecionados para cada uma das bases após o processo de *GridSearchCV*.

Código 4.6 – Ultilização do *DecisionTreeClassifier* na pesquisa.

```
from sklearn import tree
''' Modelo treinado com AD e a base sem stopwords representada
    com TF-IDF '''
ad_tf_idf_sem_stop = tree.DecisionTreeClassifier(criterion = '
    gini', max_depth = 100, splitter = 'random')
ad_tf_idf_sem_stop.fit(tf_idf_sem_stop, classes)
''' Modelo treinado com AD e a base com stopwords representada
    com TF-IDF '''
ad_tf_idf_com_stop = tree.DecisionTreeClassifier(criterion = '
    gini', max_depth = 100, splitter = 'random')
ad_tf_idf_com_stop.fit(tf_idf_com_stop, classes)
```

[7] https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier

```
''' Modelo treinado com AD e a base sem stopwords representada
    com BOW '''
ad_bow_sem_stop = tree.DecisionTreeClassifier(criterion = '
    entropy', splitter = 'random', max_depth = 100)
ad_bow_sem_stop.fit(bow_sem_stop, classes)
''' Modelo treinado com AD e a base com stopwords representada
    com BOW '''
ad_bow_com_stop = tree.DecisionTreeClassifier(criterion = 'gini',
     splitter = 'random', max_depth = 100)
ad_bow_com_stop.fit(bow_com_stop, classes)
```

O *MLPClassifier*[8] da *scikit-learn* foi a implementação das redes MLP usada neste trabalho. Com o intuito de melhorar o desempenho do classificador, também foi montada uma *grid* de parâmetros para passar pelo *GridSearchCV*. A Tabela 7 mostra o conjunto de parâmetros selecionados e suas respectivas descrições.

Tabela 7 – Grid usada no processo de seleção de parâmetros com *GridSearchCV* para o *MLPClassifier*.

Parâmetro	Descrição
solver	O solver é um solucionador para otimização de pesos na MLP. O solver 'adam', tem um desempenho melhor para conjunto de dados grandes. Para bases pequenas o 'lbfgs' é mais recomendado.
hidden_layer_sizes	Representa o número de neurônios na i-ésima camada oculta. Quanto maior o valor do hidden_layer_sizes, maior o tempo de treinamento do modelo.
alpha	Alfa é um parâmetro para o termo de regularização, também conhecido como termo de penalidade, que combate o sobreajuste dos modelos ao restringir o tamanho dos pesos.

Fonte: Elaborada pelo Autor.

O Código 4.7 representa a utilização do algoritmo perante às quatro bases, bem como o conjunto de parâmetros que foi selecionado após o uso do *GridSerachCV* em cada uma das bases.

Código 4.7 – Ultilização do *MLPClassifier* na pesquisa.

```
from sklearn.neural_network import MLPClassifier
''' Modelo treinado com MLP e a base sem stopwords representada
    com TF-IDF '''
mlp_tf_idf_sem_stop = MLPClassifier(alpha = 1e-3,
    hidden_layer_sizes = (100,), solver = 'lbfgs')
```

[8] https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

```
mlp_tf_idf_sem_stop.fit(tf_idf_sem_stop, classes)
''' Modelo treinado com MLP e a base com stopwords representada
   com TF-IDF '''
mlp_tf_idf_com_stop = MLPClassifier(alpha = 1e-3,
   hidden_layer_sizes = (100,), solver = 'lbfgs')
mlp_tf_idf_com_stop.fit(tf_idf_com_stop, classes)
''' Modelo treinado com MLP e a base sem stopwords representada
   com BOW '''
mlp_bow_sem_stop = MLPClassifier(alpha = 1e-05,
   hidden_layer_sizes = (10,), solver = 'adam')
mlp_bow_sem_stop.fit(bow_sem_stop, classes)
''' Modelo treinado com MLP e a base com stopwords representada
   com BOW '''
mlp_bow_com_stop = MLPClassifier(solver='adam',
   hidden_layer_sizes=(10,),  alpha = 1e-05)
mlp_bow_com_stop.fit(bow_com_stop, classes)
```

A *scikit-learn* possuí diferente implementações para o algoritmo SVM. Nesse trabalho adotou-se a implementação SVC[9] que permite trabalhar com diferentes tipos de *kernel*, e é ideal para classificação multiclasse. Para configurar o SVC foi selecionado uma *grid* de parâmetros com o objetivo de extrair o melhor desempenho possível do classificador. A Tabela 8 mostra os parâmetros selecionados para passar pelo processo de busca por melhores parâmetros com *GridSearchCV* e suas respectivas descrições.

Tabela 8 – Grid usada no processo de seleção de parâmetros com *GridSearchCV* para o *SVC*.

Parâmetro	Descrição
kernel	Especifica o tipo de kernel para ser usado no algoritmo.Pode ter o valor “linear”, para trabalhar melhor com dados linearmente separáveis. Pode ser também “rbf”, “poly”, “sigmoid” e “precomputed”.

Fonte: Elaborada pelo Autor.

O Código 4.8 representa a utilização do SVC no trabalho, bem como os parâmetros que foram achados para cada uma das bases.

Código 4.8 – Utilização do SVC na pesquisa.

```
from sklearn.svm import SVC
```

[9] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

```
''' Modelo treinado com SVM e a base sem stopwords representada
    com TF-IDF '''
svm_tf_idf_sem_stop = SVC(kernel = 'linear')
svm_tf_idf_sem_stop.fit(tf_idf_sem_stop, classes)
''' Modelo treinado com SVM e a base com stopwords representada
    com TF-IDF '''
svm_tf_idf_com_stop = SVC(kernel = 'linear')
svm_tf_idf_com_stop.fit(tf_idf_com_stop, classes)
''' Modelo treinado com SVM e a base sem stopwords representada
    com BOW '''
svm_bow_sem_stop = SVC(kernel = 'linear')
svm_bow_sem_stop.fit(bow_sem_stop, classes)
''' Modelo treinado com SVM e a base com stopwords representada
    com BOW '''
svm_bow_com_stop = SVC(kernel = 'linear')
svm_bow_com_stop.fit(bow_com_stop, classes)
```

4.3 Avaliação dos modelos preditivos

Com o processo de busca por hiperparâmetros com *GridSearchCV* realizado, foram executados dezesseis experimentos combinando às quatro bases já explicadas e as quatro técnicas de AM. Foi realizado o processo de Validação Cruzada, explicado na Seção 2.5.1, que consiste em dividir a base em treino e teste de maneira aleatória e refaz o procedimento K vezes. O objetivo do uso da validação cruzada nesse trabalho é verificar a capacidade de generalização dos modelos preditivos inteligentes. A capacidade de generalização de um modelo preditivo inteligente, é a habilidade que este modelo tem de classificar dados corretamente mesmo que este nunca tenha visto o dado de entrada.

Para este trabalho, foi adotado um valor de K=10. Apesar de não existir uma regra que justifique o valor de K=10, esse é um valor normalmente adotado para problemas de classificação que realizam validação cruzada, pois, esse valor foi mostrado empiricamente para produzir estimativas de taxa de erro de teste que não sofrem viés excessivamente alto nem variação muito alta (JAMES et al., 2013). Para realizar o processo de validação cruzada, foi usada a classe da *scikit-learn*, *cross_val_predict*[10].

Com o processo de validação cruzada realizado, os modelos preditivos foram avaliados perante as métricas discutidas na Seção 2.5.2. Essas métricas são: a acurácia que mede o desempenho geral do modelo, ou seja, quantas instâncias o

[10] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_predict.html#sklearn-model-selection-cross-val-predict

modelo classificou corretamente; a Precisão apresenta quantas instâncias de uma determinada classe o algoritmo classificou corretamente; a Sensibilidade mostra quantas instâncias o modelo classificou como uma classe em específico estão corretas; o *F1-Score* representa a média harmônica entre precisão e sensibilidade de uma classe.

Com a avaliação realizada, o modelo que apresentou melhor desempenho perante as métricas citadas, foi escolhido para compor o módulo inteligente para classificação automática de avaliações docentes. A avaliação e escolha do melhor modelo preditivo são detalhadas no Capítulo 5.

A parte (3) da Figura 12 mostra todo o fluxo seguido durante a etapa de modelagem e avaliação realizada no decorrer desta pesquisa, desde o treinamento com as técnicas de AM selecionadas para o trabalho até a escolha do melhor modelo preditivo.

4.4 Prova de Conceito Web

Com todo processo de análise e desenvolvimento do melhor modelo preditivo para classificar a satisfação dos alunos, é proposto uma PoC Web para permitir o uso do módulo inteligente. A ideia consiste da construção de uma API em *Flask*[11] que recebe uma nova avaliação submete ao melhor modelo preditivo e retorna como resposta para o usuário a satisfação extraída daquele *feedback*.

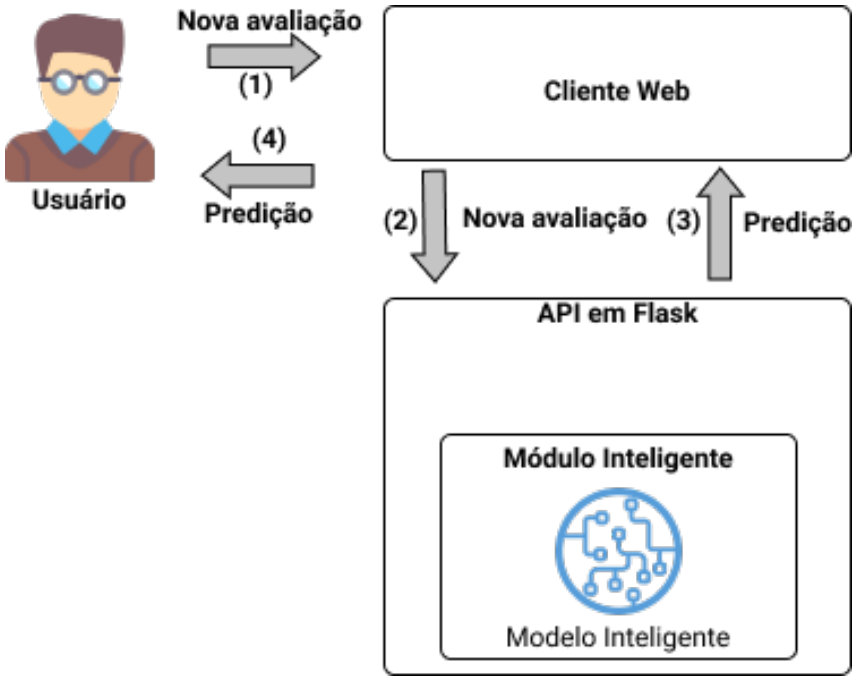
A partir da construção da API é possível seguir duas vertentes de trabalho. Uma é fornecer os serviços da API para sistemas de resposta estudantis já existentes. Ou seja, SREs já existentes podem simplesmente integrar o serviço de classificação de avaliações docente proposto no trabalho. A segunda é a construção de um sistema próprio onde os alunos possam avaliar determinada aula de um disciplina em específico. A avaliação gerada pode ser submetida ao módulo inteligente, que irá classificar a satisfação daquele aluno e direcionar a mesma para o professor da disciplina avaliada. Assim, o professor através de ferramentas como *dashboards* poderá visualizar de maneira mais intuitiva e condensada a satisfação de seus alunos, Assim, o mesmo será capaz de tomar alguma decisão sobre suas metodologias de ensino e consequentemente tentar promover melhorias no ambiente de aprendizado.

Para o escopo deste trabalho, é proposto um cliente web apenas para testar o modelo inteligente. A ideia é a interface fornecer um espaço para que o usuário entre com uma nova avaliação. O cliente web irá consumir o serviço oferecido pela API já explicada e exibirá para o usuário a satisfação atribuída pelo modelo inteligente para aquela avaliação. Para o desenvolvimento desta etapa, escolheu-se o *framework web*

[11] https://flask.palletsprojects.com/en/1.1.x/

Vue.js[12] que permite a criação de clientes web de maneira fácil e rápida. A Figura 14 mostra o fluxo seguido na PoC proposta.

Figura 14 – Fluxo da PoC web proposta para o trabalho.



Fonte: Elaborada pelo Autor.

Neste fluxo de interação entre usuário e sistema, pode-se identificar as seguintes etapas: em (1), o usuário usa o cliente web para digitar uma nova avaliação; em (2), é feito uma requisição HTTP na API proposta passando o *feedback* como parâmetro, este será pré-processado, vetorizado e submetido ao módulo inteligente; em (3), a API retorna o valor predito (“Insatisfatório”, “Neutro”, “Satisfatório”) para aquela avaliação; por fim, em (4), o cliente web exibe a satisfação predita para o usuário.

[12] https://vuejs.org

5 RESULTADOS

No Capítulo 4, foi apresentada a proposta de um sistema baseado em Análise de Sentimentos para auxiliar no processo de ensino e aprendizado bem como todos os passos seguidos para o desenvolvimento da solução proposta. Este capítulo apresenta os resultados obtidos com o desenvolvimento da proposta através da análise de desempenho dos modelos preditivos.

5.1 Análise de Desempenho dos Modelos Preditivos

Os primeiros resultados exibidos consistem de uma abordagem comparativa da utilização de diferentes modelos preditivos que satisfazem o problema abordado no trabalho. Assim, através do uso de um conjunto de métricas apresentadas ao longo da leitura e uma série de experimentos controlados usando algoritmos de AM, foi possível escolher o modelo preditivo que melhor resolveu o problema discutido.

Como citado no Capítulo 4, foram realizados diferentes experimentos com uma base inicial de avaliações docentes que foi duplicada ao manter uma base mantendo *stopwords* e outra removendo e as vetorizando com duas técnicas diferentes, gerando um total de quatro bases de treinamento. Ao todo, são comparados dezesseis modelos preditivos resultantes da combinação das quatro bases com as quatro técnicas de AM explanadas na Seção 2.3 gerando um total de dezesseis experimentos que são analisados a seguir.

Com a realização do processo de validação cruzada explicado na Seção 4.3, foram produzidos uma série de resultados que são minuciosamente avaliados nesta seção. A primeira análise feita sobre o desempenhos dos modelos preditivos foi a respeito da acurácia global dos modelos. O intuito de comparar a acurácia dos modelos é obter um parâmetro sobre o desempenho geral dos modelos, ou seja qual a capacidade dos mesmos classificarem uma avaliação docente de maneira correta como “Insatisfatório”, “Neutro” e “Satisfatório”.

A Tabela 9 mostra os valores de acurácia obtidos com a validação cruzada realizada com a combinação das quatro bases com as quatro técnicas de AM. Na tabela 9, é possível observar que as técnicas que obtiveram a melhor acurácia global foram a SVM aplicada a base com *stopwords* representada com TF-IDF e a MLP aplicada a base com *stopwords* representada com BOW.

Tabela 9 – Acurácia Global - Técnicas Supervisionadas.

Base	AD	NB	MLP	SVM
tf_idf_sem_stop	$77,47\%$	$82,29\%$	$82,43\%$	$83,99\%$
tf_idf_com_stop	$78,32\%$	$83,14\%$	$84,13\%$	**86,40%**
bow_sem_stop	$76,48\%$	$81,86\%$	$85,26\%$	$81,86\%$
bow_com_stop	$76,91\%$	$82,57\%$	**86,40%**	$83,71\%$

Fonte: Elaborada pelo Autor.

O segundo ponto analisado na comparação dos experimentos é a sensibilidade para classe “Insatisfatório”. Essa métrica revela quantas avaliações que foram consideradas “Insatisfatório” na base de treinamento realmente foram classificadas como “Insatisfatório” pelos modelos preditivos. Resolveu-se dar mais relevância a classe “Insatisfatório”, pois entende-se que a insatisfação dos alunos é o que causa algum tipo de mudança nas metodologias de ensino usadas por um professor em sala de aula. Pois, a partir do momento que um professor percebe que seus alunos estão insatisfeitos com sua maneira de ministrar seus ensinamentos ele procura novos métodos de ensino para assim melhorar a satisfação de seus alunos bem como os desempenho dos mesmos. A Tabela 10 mostra a sensibilidade obtida pelos dezesseis experimentos para a classe “Insatisfatório”. Observando a Tabela 10 percebe-se que o experimento que apresentou o melhor resultado para a métrica avaliada é novamente o algoritmo SVM aplicado a base de dados com *stopwords* representada com TF-IDF.

Tabela 10 – Sensitividade da Categoria Insatisfatório.

Base	AD	NB	MLP	SVM
tf_idf_sem_stop	$74,00\%$	$85,00\%$	$83,00\%$	$85,00\%$
tf_idf_com_stop	$76,00\%$	$84,00\%$	$83,00\%$	**87.00%**
bow_sem_stop	$82,00\%$	$85,00\%$	$86,00\%$	$86,00\%$
bow_com_stop	$74,00\%$	$80,00\%$	$86,00\%$	$86,00\%$

Fonte: Elaborada pelo Autor.

A precisão da categoria “Insatisfatório” representa todas as instâncias que o modelo classificou como “Insatisfatório” que realmente estão corretas. Resolveu-se atribuir uma relevância a essa métrica, pois se um modelo classifica muito as avaliações de maneira negativa e essas classificações estão erradas e os alunos estão satisfeito com a forma de ensino do professor, isso pode causar mudanças desnecessárias nas metologias do professor que nem sempre podem ser boas. A Tabela 11 mostra os valores da precisão da classe “Insatisfatório” obtidas com a realização dos experimentos. Observando a Tabela 11 percebe-se que a configuração que obteve melhor precisão da classe “Insatisfatório” foi o modelo gerado pela base com *stopwords* representa com TF-IDF treinada com SVM.

Tabela 11 – Precisão da Categoria Insatisfatório.

Base	AD	NB	MLP	SVM
tf_idf_sem_stop	77,00%	79,00%	83,00%	83,00%
tf_idf_com_stop	79,00%	82,00%	85,00%	**87,00%**
bow_sem_stop	72,00%	79,00%	86,00%	80,00%
bow_com_stop	78,00%	83,00%	86,00%	84,00%

Fonte: Elaborada pelo Autor.

Como a precisão e sensibilidade da categoria "Insatisfatório" foram consideradas, o *F1-Score* da mesma também foi considerada, tendo em vista que o *F1-Score* representa a média harmônica da precisão e sensibilidade de uma classe. A Tabela 12 apresenta os valores do *F1-Score* da classe insatisfatório com a realização dos experimentos. Analisando a Tabela 12 percebe-se que novamente a configuração que obteve o melhor desempenho na métrica avaliada foi o SVM aplicado a base de dados com *stopwords* representada com TF-IDF.

Tabela 12 – F1-Score da Categoria Insatisfatório.

Base	AD	NB	MLP	SVM
tf_idf_sem_stop	76,00%	82,00%	83,00%	84,00%
tf_idf_com_stop	77,00%	83,00%	84,00%	**87,00%**
bow_sem_stop	77,00%	82,00%	86,00%	83,00%
bow_com_stop	76,00%	82,00%	86,00%	85,00%

Fonte: Elaborada pelo Autor.

Com a observação dos resultados, percebe-se que apesar da rede MLP e o SVM apresentarem melhores resultados, o algoritmo SVM destacou-se nas métricas escolhidas para avaliação, principalmente quando aplicado a base de dados com *stopwords* representada com TF-IDF.

Com o bom desempenho apresentado, o modelo escolhido para compor o módulo inteligente desse trabalho foi o que combina o algoritmo SVM com a técnica de representação de palavras TF-IDF. Também optou-se pela não remoção de *stopwords* das avaliações docentes.

Tabela 13 – Matriz de confusão para o SVM aplicado a base com *stopwords* representada TF-IDF.

	Insatisfatório	Neutro	Satisfatório
Insatisfatório	210	14	17
Neutro	25	181	24
Satisfatório	7	9	219

Fonte: Elaborada pelo Autor.

A Tabela 13 apresenta a matriz de confusão obtida para o experimento que combinou o SVM com TF-IDF. As demais matrizes de confusão estão presentes nos Anexos deste trabalho.

5.1.1 Discussão Sobre o Desempenho do SVM

O algoritmo SVM é uma das técnicas amplamente difundidas na literatura para a tarefa de classificação de texto. Os autores de (JOACHIMS, 1998) debatem algumas características envolvidas no processo de classificação de texto que justificam o bom desempenho do algoritmo na tarefa. Uma das principais características citadas pelo autor é a alta dimensionalidade dos dados de entrada. Segundo os autores, o bom desempenho do SVM nessas situações ocorre dado que o mesmo usa proteção de *overfitting*, que não depende necessariamente do espaço de dimensional da base de treinamento.

Outra característica que normalmente permite que o SVM obtenha um bom desempenho são bases de treinamento com poucas instâncias (GOUDJIL et al., 2018). Ambas estas as características citadas estão presentes na base de treinamento utilizada neste trabalho. Como já citado na Seção 4.1.1, a base de treinamento construída contém um total de 706 instâncias, um número relativamente pequeno para problemas de classificação. Outro aspecto a ser considerado é a base de dados após ser representada com a técnica TF-IDF. A base resultante é uma matriz com 726 atributos, uma dimensão consideravelmente alta para problemas de classificação. Esses são alguns pontos estudados que teoricamente justificam o bom desempenho do algoritmo SVM para o problema em questão.

6 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Evitar a evasão universitária tornou-se um grande desafio para as IES e existem poucos mecanismos que ajudam professores e gestores educacionais a tomar decisões para mitigar esse problema. A evasão de um universitário pode ser causada por inúmeros fatores internos e externos à instituição de ensino. Uma variável importante no processo de evasão é a satisfação dos alunos com relação às metodologias adotadas por seus professores. Os sistemas de resposta estudantil possibilitam a avaliação de alunos a respeito do desempenho de seus professores. Contudo, analisar e extrair alguma informação útil para os professores e gestores educacionais é um processo maçante e pode ser demorado.

Nesse contexto, este trabalho apresentou um mecanismo baseado em análise de sentimentos para extrair informação de maneira automática de avaliações docentes. O objetivo do mecanismo proposto é extrair a satisfação (“Insatisfatório”, “Neutro”, “Satisfatório”) de maneira automática de uma avaliação docente. Dessa forma, há uma otimização no tempo de análise do *feedback* estudantil, o que permite aos professores adequar de maneira mais eficiente suas metodologias de ensino para assim satisfazer as necessidades dos alunos e melhorar o processo de ensino aprendizado. Assim, pode-se auxiliar a criar um ambiente mais propício à permanência dos discentes.

Este trabalho contou com uma série de experimentos controlados que permitiram avaliar o desempenho de diferentes técnicas de AM combinadas com diferentes bases de dados. Com a realização dos experimentos, foi possível escolher um modelo preditivo que melhor soluciona o problema abordado. Ao longo da análise dos experimentos, foi possível perceber que a técnica que apresentou melhor desempenho foi o algoritmo SVM, que obteve acurácia superior a 86% quando aplicado à base de dados representada com a técnica TF-IDF.

Como apresentado ao longo do trabalho, uma das grandes dificuldades para problemas de classificação de texto em português é encontrar bases relacionadas ao contexto abordado, para realizar o treinamento e avaliação dos modelos preditivos. Neste trabalho não foi diferente. Por isso, durante o desenvolvimento da pesquisa optou-se pela construção de uma base com avaliações docentes fictícias. Embora a base de treinamento utilizada neste trabalho tenha sido gerada artificialmente, ao longo da análise dos resultados foi possível perceber que o desempenho dos modelos preditivos foi satisfatório.

A partir de tudo que foi apresentado neste trabalho, é proposto como trabalhos futuros, aumentar a base de avaliações usada no desenvolvimento da pesquisa

e agregar novas fontes de dados relacionadas ao contexto abordado. Pretende-se explorar outras técnicas convencionais de Aprendizado de Máquina e também analisar a classificação de texto usando *Deep Learning* por meio das Redes Neurais Recorrentes, em especial as redes LSTM (*Long Short-Term Memory*). Uma vez que as LSTM estão sendo bastante difundidas para problemas de classificação de texto.

REFERÊNCIAS

AGUIAR, E. J. D. *Análise de Sentimento em Redes Sociais Utilizando Combinção de Classificadores*. Monografia (Graduação) — Universidade Estadual do Norte do Paraná, Bandeirantes, Paraná, 2017. Citado 3 vezes nas páginas 24, 36 e 38.

ALTRABSHEH, N.; GABER, M.; COCEA, M. Sa-e: sentiment analysis for education. In: *International Conference on Intelligent Decision Technologies*. Sesimbra, Portugal: Springer, 2013. v. 255, p. 353–362. Citado 5 vezes nas páginas 16, 19, 20, 21 e 22.

AZEVEDO, D. et al. Aplicação de análise de sentimento em fóruns educacionais para prevenir evasão. In: *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*. Recife-PE, Brasil: Sociedade Brasileira de Computação – SBC, 2017. v. 28, n. 1, p. 1097. Citado 2 vezes nas páginas 40 e 45.

BALAHADIA, F. F.; FERNANDO, M. C. G.; JUANATAS, I. C. Teacher's performance evaluation tool using opinion mining with sentiment analysis. In: IEEE. *Region 10 Symposium (TENSYMP), 2016 IEEE*. Bali, Indonesia, 2016. p. 95–98. Citado 2 vezes nas páginas 15 e 40.

BARROSO, M. F.; FALCÃO, E. B. Evasão universitária: o caso do instituto de física da ufrj. *IX Encontro Nacional de Pesquisa em Ensino de Física*, v. 9, p. 1–14, 2004. Citado na página 15.

CAMPBELL, C. An introduction to kernel methods radial basis function network: design and applications. *Springer Verlag, Berlin*, v. 1, p. 31, 2000. Citado na página 31.

CARVALHO, C. M. A. d. et al. Estudo comparativo de análise de sentimentos aplicado à notícias públicas. Universidade Federal do Maranhão, 2018. Citado 2 vezes nas páginas 25 e 37.

CARVALHO, M. H. D. Estudo comparativo dos métodos de word embedding na análise de sentimentos. Recife-PE, Brasil, 2018. Citado na página 26.

FACELI, K. et al. *Inteligência Artificial: Uma Abordagem de Aprendizagem de Máquina*. Rio de Janeiro: LTC, 2011. Citado 9 vezes nas páginas 28, 29, 30, 31, 32, 33, 34, 35 e 37.

GOUDJIL, M. et al. A novel active learning method using svm for text classification. *International Journal of Automation and Computing*, Springer, v. 15, n. 3, p. 290–298, 2018. Citado 3 vezes nas páginas 30, 51 e 64.

HONDA, H. *Os Três Tipos de Aprendizado de Máquina*. 2017. https://lamfo-unb.github.io/2017/07/27/tres-tipos-am/. Acessado em 10/03/2019. Citado na página 27.

INEP. *Sinopse Estatística da Educação Superior 2018*. Brasília: Inep, 2019. Disponível em: <http://portal.inep.gov.br/basica-censo-escolar-sinopse-sinopse>. Acesso em: 20 jan. 2020. Citado na página 15.

JAMES, G. et al. *An introduction to statistical learning*. New York: Springer, 2013. Citado na página 58.

JOACHIMS, T. Text categorization with support vector machines: Learning with many relevant features. In: SPRINGER. *European conference on machine learning*. Berlin, Heidelberg, 1998. p. 137–142. Citado 2 vezes nas páginas 51 e 64.

KOVÁCS, Z. L. *Redes neurais artificiais*. São Paulo-SP, Brasil: Editora Livraria da Fisica, 2002. Citado na página 34.

LORENA, A. C.; CARVALHO, A. C. de. Uma introdução às support vector machines. *Revista de Informática Teórica e Aplicada*, v. 14, n. 2, p. 43–67, 2007. Citado na página 30.

MEDHAT, W.; HASSAN, A.; KORASHY, H. Sentiment analysis algorithms and applications: A survey. *Ain Shams engineering journal*, Elsevier, v. 5, n. 4, p. 1093–1113, 2014. Citado na página 25.

NEWMAN, H.; JOYNER, D. Sentiment analysis of student evaluations of teaching. In: SPRINGER. *International Conference on Artificial Intelligence in Education*. London, UK, 2018. p. 246–250. Citado na página 15.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011. Citado 4 vezes nas páginas 35, 49, 51 e 52.

PRABOWO, R.; THELWALL, M. Sentiment analysis: A combined approach. *Journal of Informetrics*, Elsevier, v. 3, n. 2, p. 143–157, 2009. Citado na página 22.

RANI, S.; KUMAR, P. A sentiment analysis system to improve teaching and learning. *Computer*, IEEE, v. 50, n. 5, p. 36–43, 2017. Citado na página 39.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, American Psychological Association, v. 65, n. 6, p. 386, 1958. Citado na página 33.

RUSSELL, P. N. S. *Inteligência Artificial*. 3ª edição. ed. Rio de Janeiro–RJ, Brasil: Elsevier Editora Ltda, 2013. ISBN 978-85-352-3701-6. Citado 2 vezes nas páginas 20 e 27.

SANTOS, F.; SILVEIRA, I. F.; LECHUGO, C. Mineração da percepção do aluno: Mineração de dados educacionais na geração de indicadores para a avaliação de práticas pedagógicas docentes. In: *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*. Uberlândia-MG, Brasil: Sociedade Brasileira de Computação – SBC, 2016. v. 5, p. 1473. Citado 3 vezes nas páginas 15, 19 e 39.

SANTOS, F. de P.; LECHUGO, C. P.; SILVEIRA-MACKENZIE, I. F. "speak well" or "complain" about your teacher: A contribution of education data mining in the evaluation of teaching practices. In: IEEE. *Computers in Education (SIIE), 2016 International Symposium on*. Salamanca, Spain, 2016. p. 1–4. Citado 3 vezes nas páginas 16, 19 e 26.

SCHÜTZE, H.; MANNING, C. D.; RAGHAVAN, P. *Introduction to information retrieval*. New York-NY, United States: Cambridge University Press, 2008. Citado na página 16.

TIAN, F. et al. Can e-learner's emotion be recognized from interactive chinese texts? In: IEEE. *2009 13th International Conference on Computer Supported Cooperative Work in Design*. Santiago, Chile, 2009. p. 546–551. Citado na página 22.

VAPNIK, V. N. The nature of statistical learning theory. Springer-Verlag, New York, 1995. Citado na página 30.

VIEIRA, R.; LOPES, L. Processamento de linguagem natural e o tratamento computacional de linguagens científicas. *EM CORPORA*, p. 183, 2010. Citado na página 20.

WALKER, M. *Introduction to Natural Language Processing: Concepts and Fundamentals for Beginners*. Sepapaja 6, Tallinn, 15551, Estonia: AI Sciences, 2018. Citado 7 vezes nas páginas 21, 23, 25, 26, 28, 29 e 30.

ZHANG, L.; WANG, S.; LIU, B. Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, v. 8, n. 4, p. e1253, 2018. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/widm.1253>. Citado na página 16.

ZHENG, A. Evaluating machine learning models: a beginner's guide to key concepts and pitfalls. O'Reilly Media, 2015. Citado na página 35.

Apêndice

APÊNDICE A - MATRIZES DE CONFUSÃO DOS EXPERIMENTOS REALIZADOS DURANTE A PESQUISA

Matrizes de confusão obtidas com a realização dos dezesseis experimentos controlados presentes na pesquisa.

Tabela 14 – Matriz de confusão para o experimento nb_bow_com_stop.

	Insatisfatório	Neutro	Satisfatório
Insatisfatório	194	30	17
Neutro	29	175	26
Satisfatório	12	9	214

Fonte: Elaborada pelo Autor.

Tabela 15 – Matriz de confusão para o experimento nb_bow_sem_stop.

	Insatisfatório	Neutro	Satisfatório
Insatisfatório	204	23	14
Neutro	41	165	24
Satisfatório	12	14	209

Fonte: Elaborada pelo Autor.

Tabela 16 – Matriz de confusão para o experimento nb_tfidf_com_stop.

	Insatisfatório	Neutro	Satisfatório
Insatisfatório	202	25	14
Neutro	31	173	26
Satisfatório	14	9	212

Fonte: Elaborada pelo Autor.

Tabela 17 – Matriz de confusão para o nb_tf_idf_sem_stop.

	Insatisfatório	Neutro	Satisfatório
Insatisfatório	204	23	14
Neutro	37	170	23
Satisfatório	16	12	207

Fonte: Elaborada pelo Autor.

Tabela 18 – Matriz de confusão para o ad_bow_com_stop.

	Insatisfatório	Neutro	Satisfatório
Insatisfatório	178	31	32
Neutro	35	159	36
Satisfatório	15	14	206

Fonte: Elaborada pelo Autor.

Tabela 19 – Matriz de confusão para o experimento ad_bow_sem_stop.

	Insatisfatório	Neutro	Satisfatório
Insatisfatório	198	24	19
Neutro	48	156	26
Satisfatório	28	21	186

Fonte: Elaborada pelo Autor.

Tabela 20 – Matriz de confusão para o experimento ad_tf_idf_com_stop.

	Insatisfatório	Neutro	Satisfatório
Insatisfatório	182	22	37
Neutro	33	161	36
Satisfatório	15	10	210

Fonte: Elaborada pelo Autor.

Tabela 21 – Matriz de confusão para o experimento ad_tf_idf_sem_stop.

	Insatisfatório	Neutro	Satisfatório
Insatisfatório	179	30	32
Neutro	36	156	38
Satisfatório	16	7	212

Fonte: Elaborada pelo Autor.

Tabela 22 – Matriz de confusão para o experimento mlp_bow_com_stop.

	Insatisfatório	Neutro	Satisfatório
Insatisfatório	207	23	11
Neutro	27	186	17
Satisfatório	7	11	217

Fonte: Elaborada pelo Autor.

Tabela 23 – Matriz de confusão para o experimento mlp_bow_sem_stop.

	Insatisfatório	Neutro	Satisfatório
Insatisfatório	208	24	9
Neutro	27	181	22
Satisfatório	6	16	213

Fonte: Elaborada pelo Autor.

Tabela 24 – Matriz de confusão para o experimento mlp_tf_idf_com_stop.

	Insatisfatório	Neutro	Satisfatório
Insatisfatório	200	29	12
Neutro	21	191	18
Satisfatório	15	17	203

Fonte: Elaborada pelo Autor.

Tabela 25 – Matriz de confusão para o experimento mlp_tf_idf_sem_stop.

	Insatisfatório	Neutro	Satisfatório
Insatisfatório	200	27	14
Neutro	31	179	20
Satisfatório	10	22	203

Fonte: Elaborada pelo Autor.

Tabela 26 – Matriz de confusão para o experimento svm_bow_com_stop.

	Insatisfatório	Neutro	Satisfatório
Insatisfatório	207	16	18
Neutro	25	176	29
Satisfatório	15	12	208

Fonte: Elaborada pelo Autor.

Tabela 27 – Matriz de confusão para o experimento svm_bow_sem_stop.

	Insatisfatório	Neutro	Satisfatório
Insatisfatório	207	19	15
Neutro	33	165	32
Satisfatório	19	10	206

Fonte: Elaborada pelo Autor.

Tabela 28 – Matriz de confusão para o experimento svm_tf_idf_com_stop.

	Insatisfatório	Neutro	Satisfatório
Insatisfatório	210	14	17
Neutro	25	181	24
Satisfatório	7	9	219

Fonte: Elaborada pelo Autor.

Tabela 29 – Matriz de confusão para o experimento svm_tf_idf_sem_stop.

	Insatisfatório	Neutro	Satisfatório
Insatisfatório	204	19	18
Neutro	36	168	26
Satisfatório	5	9	221

Fonte: Elaborada pelo Autor.