

**Fundação Getulio Vargas  
Escola de Matemática Aplicada**

**Gustavo Vianna Avila**

**Análise de sentimento para textos curtos**

Rio de Janeiro  
2017

Avila, Gustavo Vianna

Análise de sentimento para textos curtos / Gustavo Vianna Avila. – 2017.  
76 f.

Dissertação (mestrado) – Fundação Getulio Vargas, Escola de Matemática Aplicada.

Orientador: Flávio Codeço Coelho.

Inclui bibliografia.

1. Mineração de dados (Computação). 2. Processamento da linguagem natural (Computação). 3. Aprendizado do computador. 4. Modelagem de dados. I. Coelho, Flávio Codeço. II. Fundação Getulio Vargas. Escola de Matemática Aplicada. III. Título.

CDD – 006.312

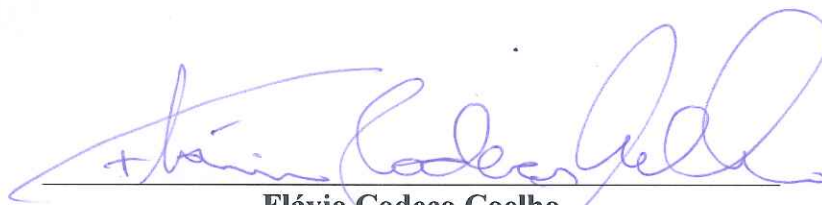
**GUSTAVO VIANNA AVILA**

**ANÁLISE DE SENTIMENTO PARA TEXTOS CURTOS.**

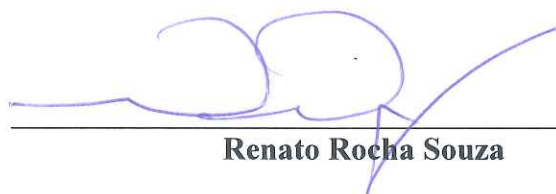
Dissertação apresentada ao Curso de Mestrado em Modelagem Matemática da Informação da Escola de Matemática Aplicada da Fundação Getúlio Vargas para obtenção do grau de Mestre em Modelagem Matemática da Informação.

Data da defesa: 10/03/2017.

**ASSINATURA DOS MEMBROS DA BANCA EXAMINADORA**

A handwritten signature in blue ink, appearing to read 'Flávio Codeço Coelho', is written over a horizontal line.

**Flávio Codeço Coelho**  
Orientador (a)

A handwritten signature in blue ink, appearing to read 'Renato Rocha Souza', is written over a horizontal line.

**Renato Rocha Souza**

A handwritten signature in blue ink, appearing to read 'Ligia Maria Arruda Café', is written over a horizontal line.

**Ligia Maria Arruda Café**

**Gustavo Vianna Avila**

**Análise de sentimento para textos curtos**

Dissertação submetida à Escola de Matemática Aplicada como requisito parcial para a obtenção do grau de Mestre em Modelagem Matemática da Informação.

Área de Concentração: Modelagem e Análise da Informação

Orientador: Flávio Codeço Coelho

Rio de Janeiro  
2017

## **Dedicatória**

Dedico este trabalho a minha esposa Patrícia pelo apoio incondicional.

Aos meus filhos Laura e Thomas, que são minha inspiração e motivo de alegria.

A minha mãe Regina Maria, meu pai Paulo Roberto, meus irmãos Rafael e Juliana, minha tia Eliane e a todos os meus familiares que sempre me apoiaram e incentivaram a ir atrás dos meus objetivos.

## **Agradecimentos**

Aos professores da Escola de Matemática Aplicada da Fundação Getúlio Vargas: Alexandre Rademaker, Asla Medeiros e Sá, Flávio Codeço Coelho, Hugo A. de La Cruz Cansino, Maria Soledad Aronna, Moacyr Alvim Horta Barbosa da Silva, Paulo Cezar P. Carvalho, Renato Rocha Souza, Vincent Gérard Yannick Guigues pelas suas atitudes, ensinamentos, exemplos e incentivos que contribuíram para a o meu desenvolvimento pessoal e profissional.

Ao professor Orlando Bernardo Filho do departamento de Engenharia de Sistemas e Computação da Universidade do Rio de Janeiro pelo apoio e incentivo desde os tempos de graduação até hoje.

À Petrobras por patrocinar e autorizar a realização desse curso.

Ao Américo Fernandes Figueiredo e Sá, Carlos José Carneiro de Vilhena, Luiz Felipe de Araújo por compreender e apoiar a minha dedicação de tempo ao curso de mestrado em Modelagem Matemática da Informação.

Ao Diego César Nascimento pela parceria e contribuição nas análises e discussões relacionadas a aplicação de análise de sentimento na Petrobras.

Ao meu orientador Flávio Codeço Coelho pela sua disponibilidade, pelas suas recomendações e proposições construtivas.

## Resumo

Um grande número de mensagens curtas informais são postadas diariamente em redes sociais, fóruns de discussão e pesquisas de satisfação. Emoções parecem ser importantes de forma frequente nesses textos. O desafio de identificar e entender a emoção presente nesse tipo de comunicação é importante para distinguir o sentimento presente no texto e também para identificar comportamentos anômalos e inapropriados, eventualmente oferecendo algum tipo de risco.

Este trabalho propõe a implementação de uma solução para a análise de sentimento de textos curtos baseada em aprendizado por máquina. Utilizando técnicas de aprendizado supervisionado, é desejado discernir se uma mensagem possui sentimento positivo, neutro ou negativo. As mensagens a serem analisadas serão pesquisas de satisfação de serviços de TI. Foram utilizados nas análises dois modelos, o primeiro modelo onde apenas o campo de texto livre "Comentário" foi considerado e o segundo modelo, onde além do campo de texto livre "Comentário", foram consideradas, adicionalmente, duas perguntas objetivas da pesquisa de satisfação.

Os resultados obtidos indicam que as técnicas utilizadas de aprendizado por máquina, não ficam atrás dos resultados produzidos por aprendizado humano. A acurácia obtida foi de até 86,8% de acerto para um modelo de três classes: "elogio", "neutro" e "reclamação". A acurácia foi significativamente superior, alcançando até 94,5% em um modelo alternativo, de apenas duas classes: "elogio" e "não-elogio".

## **Abstract**

A huge number of short informal messages are posted every day in social network sites, discussion forums and customer surveys. Emotions seem to be frequently important in these texts. The challenge of identifying and understanding an emotion present in this type of communication is important in distinguishing the sentiment in the text and also in identifying anomalous and inappropriate behaviors, eventually offering some kind of risk.

This work proposes the implementation of a sentiment analysis solution based on machine learning. Using supervised learning techniques, it is desired to discern whether a message has a positive, neutral, or negative sentiment. The messages to be analyzed are IT service satisfaction surveys. Two models were used in the analysis, the first model where only the "Comment", a non-structured text field was considered and the second model, where besides the "Comment" field, two objective questions were considered.

The results obtained indicate that the techniques of machine learning, are not behind the results produced by human-produced baselines. The accuracy obtained was up to 86.8% accuracy for a three class model: "praise", "neutral" and "complaint". Accuracy was significantly higher, reaching up to 94.5 % in an alternative model of only two classes: "praise" and "non-praise".



# Sumário

<b>Lista de Figuras</b>	<b>7</b>
<b>Lista de Tabelas</b>	<b>8</b>
<b>Lista de Códigos de Programas</b>	<b>9</b>
<b>1 Introdução</b>	<b>11</b>
<b>2 Referencial teórico</b>	<b>14</b>
2.1 Processamento de linguagem natural (PLN)	14
2.2 Aprendizado de máquina	16
2.2.1 Aprendizagem supervisionada	17
2.2.2 Aprendizagem não supervisionada	18
2.2.3 Aprendizagem por reforço	18
2.2.4 Aprendizagem semi-supervisionada	19
2.3 Análise de Sentimento	20
2.3.1 Técnicas baseadas em conhecimento	21
2.3.2 Técnicas baseadas em estatística	22
2.3.3 Técnicas híbridas	23
2.4 Desequilíbrio de classes	24
2.4.1 Técnicas em nível de dados	24
2.4.2 Técnicas em nível de algoritmo	25
2.5 Algoritmos de Classificação	27
2.5.1 <i>Naive Bayes</i> (NB)	27
2.5.2 <i>Support Vector Machine</i> (SVM)	29
2.5.3 <i>Stochastic Gradient Descent</i> (SGD)	30
2.5.4 <i>XGBoost</i>	33
<b>3 Metodologia</b>	<b>36</b>
3.1 Descrição dos dados	36
3.2 Pré-processamento	43
3.2.1 Valores atribuídos na manifestação	43
3.2.2 Sumarização no campo manifestação geral	44
3.2.3 Adequação de idioma	48
3.2.4 Padronização do texto livre	49
3.2.5 Correção ortográfica	50
3.3 Processamento dos dados	51

3.3.1	Seleção de <i>features</i> . . . . .	51
3.3.2	Algoritmos de aprendizado de máquina . . . . .	53
3.3.3	Utilização de <i>features</i> adicionais . . . . .	54
3.3.4	Modelos de aprendizado de máquina para análise de sentimento . . . . .	55
3.3.5	Tratando o desequilíbrio de classes . . . . .	59
3.3.6	Modelo binário de classes . . . . .	60
<b>4</b>	<b>Resultados</b>	<b>63</b>
4.1	Resultados do modelo <i>multiclass</i> . . . . .	63
4.2	Proposição de modelo de visualização da informação para análise de resultado de um classificador . . . . .	63
4.3	Técnicas para melhorar a qualidade do modelo . . . . .	65
4.4	Resultados do modelo binário de classes . . . . .	67
<b>5</b>	<b>Conclusão</b>	<b>70</b>
<b>6</b>	<b>Referências</b>	<b>72</b>

## Lista de Figuras

2.1	<i>Support Vector Machine</i> (SVM) . . . . .	30
2.2	<i>Stochastic Gradient Descent</i> (SGD) . . . . .	32
2.3	Amostra de uma árvore CART . . . . .	34
3.1	E-mail sobre a pesquisa de satisfação . . . . .	36
3.2	Tela da pesquisa de satisfação . . . . .	37
3.3	Histórico Global do Quantitativo . . . . .	39
3.4	Manifestação ao Atendimento . . . . .	45
3.5	Manifestação ao Serviço . . . . .	46
3.6	Manifestação Geral . . . . .	48
3.7	Modelo Binário - Manifestação Geral . . . . .	62
4.1	modelo de visualização da informação . . . . .	65
4.2	Curva ROC para classificador SVM Linear no modelo (2) . . .	69

## Lista de Tabelas

3.1	Campos da Pesquisa de Satisfação e sua descrição . . . . .	40
3.2	Exemplos de registros . . . . .	41
3.3	Resultados dos modelos de seleção de <i>features</i> por algoritmo .	53
3.4	Representatividade das classes . . . . .	60
4.1	Resultados dos modelos de análise de sentimento . . . . .	63
4.2	Resultados dos modelos de análise de sentimento binário . . .	68

## Lista de Códigos de Programas

3.1	Transformação de valores atribuídos na manifestação . . . . .	44
3.2	Sumarização da Manifestação . . . . .	47
3.3	Adequação de Idioma . . . . .	49
3.4	Padronização do Texto Livre . . . . .	50
3.5	Seleção de <i>features</i> . . . . .	53
3.6	Utilização de <i>features</i> adicionais . . . . .	55
3.7	SVM Linear . . . . .	56
3.8	SVM não Linear . . . . .	57
3.9	<i>Naive Bayes</i> . . . . .	58
3.10	SGD . . . . .	58
3.11	<i>XGBoost</i> . . . . .	59
3.12	Modelo Binário: Pré-Processamento . . . . .	61
4.1	Utilizando <i>GridSearch</i> . . . . .	66
4.2	Parâmetros utilizados no algoritmo SVM linear . . . . .	66
4.3	<i>Voting</i> . . . . .	67

# 1 Introdução

Um grande número de textos não estruturados, curtos e informais são publicados diariamente tanto na internet (ex: fórum, blog, twitter) como nas redes corporativas (ex: pesquisa de satisfação, comentários na intranet). Entender o que as outras pessoas pensam ou qual o seu sentimento sobre determinado assunto é uma informação relevante e normalmente utilizada para o processo de tomada de decisão [1].

Análise de sentimento ou *opinion mining* se refere ao campo que trata de forma computacional a opinião, o sentimento e a subjetividade de um texto. O termo computação afetiva também é usado quando se deseja que computadores reconheçam e expressem sentimentos.[1]. Segundo o dicionário Michaelis de língua portuguesa, a palavra sentimento significa: "a atitude mental a respeito de alguém ou de alguma coisa." Na análise de sentimento, estamos portanto, interessados em analisar essa atitude mental.

Emoções ou sentimentos possuem um importante papel na comunicação humana. Algumas vezes, a inteligência emocional é mais importante que o quociente de inteligência para o sucesso de uma interação. Existem inclusive evidências significantes de que o aprendizado nos humanos é dependente das emoções [2]. A análise de sentimento é fator importante também para o desenvolvimento da inteligência artificial e seus campos correlatos [3], visto que identificar e entender a emoção é importante não só para humanos, como para computadores.

Vamos analisar alguns conceitos gerais que serão importantes para o desenvolvimento do trabalho. Iremos comentar brevemente sobre: mineração de textos, análise de sentimento, *natural language processing* (NLP) ou, em português, processamento de linguagem natural e *machine learning* ou, em português, aprendizado por máquina. Muitos desses conceitos serão detalhados posteriormente na seção que trata do referencial teórico.

A mineração de textos surgiu com a finalidade de buscar padrões em um texto em linguagem natural, e pode ser definida como: o processo de análise do texto para extrair informação para um propósito particular [4].

O principal propósito deste trabalho com mineração de textos é a análise de sentimento. Podemos pensar na análise de sentimento como a classificação de textos em uma ou mais categorias predefinidas, utilizando técnicas de processamento de linguagem natural e *machine learning*.

As abordagens para tratar o problema de análise de sentimento se encaixam em três categorias principais: *knowledge-based*, *statistical-based* e *hybrid*.

*based*.

*Knowledge-based* é a abordagem que classifica o texto em categorias, como por exemplo positivo e negativo, de acordo com a presença de palavras específicas tais como "demorado", "excelente" e "satisfeito". *Statistical-based* é a abordagem que utiliza técnicas de inferência estatística para classificar o texto. A abordagem *hybrid-based* utiliza as duas abordagens citadas anteriormente, em conjunto.[5]

Em geral, algoritmos para análise de Sentimentos que utilizam a abordagem *statistical-based*, utilizam técnicas de *machine learning* para identificar *features* associadas a sentimentos negativos, positivos e neutros. Dois problemas recorrentes de *machine learning* estão relacionados à: escolha de *features* e escolha do Algoritmos de Classificação. Trataremos desses problemas, mais a frente, na seção sobre o referencial teórico.

Por hora, vale definir o conceito de *features* como o conjunto de atributos, geralmente representados como um vetor, associados à amostra de dados[6]. Por exemplo, no caso de um algoritmo para detecção de mensagens de correio eletrônico como *spam*, algumas *features* relevantes podem incluir: tamanho da mensagem, nome do destinatário, nome do remetente, título da mensagem, presença de determinada palavra chave no corpo da mensagem, presença de anexo, formato do anexo, etc.

Um fator complicante para a detecção de sentimento em textos não estruturados, curtos e informais é que, frequentemente, as regras gramaticais e de ortografia são ignoradas[7]. Um exemplo comum dessa situação, para a língua portuguesa, é a omissão de acentos das palavras. Podemos citar também o caso de abreviações, *emoticons* e sentenças truncadas.

É importante perceber a dificuldade em codificar todas as regras, palavras e sentenças que podem ocorrer em um texto não estruturado. Consequentemente, é um desafio tentar encontrar uma solução analítica para esse problema. A forma apropriada para tratar esse tipo de questão, em que não temos uma solução analítica disponível, porém temos dados suficientes para construir uma solução empírica, é utilizar a informação disponível a partir dos dados conhecidos ou históricos. Essa técnica é conhecida como *learning from data* [8] e será empregada nesse trabalho.

*Machine Learning* é a área da computação responsável por fazer os computadores agirem sem serem explicitamente programados, utilizando a técnica *learning from data* mencionada anteriormente. Exemplos práticos de aplicação de técnicas de *machine learning* são: carros que dirigem sozinhos em qualquer estrada, detecção de *spam* no serviço de correio eletrônico, recomendação

de produtos baseada no histórico de compras de um cliente e estratégias para compra e venda de ações baseada na movimentação de preços.

Algoritmos de *machine learning* que não utilizam conceitos linguísticos, em geral, identificam o sentimento baseado nas ocorrências de palavras, *bi-grams* (pares de palavras) e *trigrams* (trincas de palavras). Ou seja, as *features* utilizadas pelo algoritmo são relacionadas à presença de termos ou à frequência em que os termos aparecem no texto. O resultado desse tipo de algoritmo nem sempre é satisfatório devido aos fatores complicantes citados anteriormente, relacionados às regras gramaticais e de ortografia que são ignoradas. Iremos verificar com mais profundidade esse conceito na próxima seção.

Além de técnicas de *machine learning* brevemente comentadas acima, será necessário utilizar também técnicas de *Natural Language Processing* (NLP) nesse trabalho. NLP é a área da computação que explora a questão de como computadores podem ser utilizados para entender e manipular textos e áudio em linguagem natural, como inglês, português e espanhol. O objetivo da NLP é concentrar conhecimento de como o ser humano entende e utiliza a linguagem, para então desenvolver ferramentas e técnicas apropriadas, para que os sistemas de computador entendam e manipulem a linguagem natural apropriadamente.[9]

Os algoritmos modernos para análise de sentimento utilizam de técnicas de NLP e são baseados em *machine learning*, onde não existe codificação de regras específicas para cada caso. Esse paradigma de *machine learning* é baseado em inferência estatística, de forma que as regras são identificadas com base em um grande número de exemplos reais. Chamamos de *corpus* o conjunto de sentenças ou documentos que servem de exemplo e devem ser aprendidas pelos algoritmos.

Vale comentar nessa introdução que utilizaremos o *Natural Language Toolkit* (NLTK) e o *scikit-learn* para processamento de linguagem natural e implementação dos algoritmos de análise de sentimentos. O NLTK é uma suíte de programas, corpus e tutoriais com recursos de NLP e *machine learning*. [10] O NLTK é escrito em Python e distribuído abaixo da licença GPL Open Source.

O *scikit-learn* [11] é um pacote Python com diversas funcionalidades e algoritmos de *machine learning* para rápida utilização. A ênfase é disponibilizar ferramentas de *machine learning* para não especialistas, baseado em facilidade de uso e documentação extensa e consistente. *Scikit-learn* é distribuído abaixo da licença BSD.



## 2 Referencial teórico

### 2.1 Processamento de linguagem natural (PLN)

Como comentado na seção de introdução, o objetivo do processamento de linguagem natural (PLN), ou, do inglês, *natural language processing* (NLP), é concentrar conhecimento de como o ser humano entende e utiliza a linguagem [9]. Isso permite desenvolver ferramentas e técnicas apropriadas, para que os sistemas de computador entendam e manipulem a linguagem natural apropriadamente. O processamento de linguagem natural possui três etapas analíticas distintas: morfológica, sintática, semântica [12].

A análise morfológica, preocupa-se com as palavras isoladamente e não com a sua participação na frase. A morfologia está agrupada em dez classes, denominadas classes de palavras ou classes gramaticais: substantivo, artigo, adjetivo, numeral, pronome, verbo, advérbio, preposição, conjunção e interjeição.

A análise sintática preocupa-se com a associação das palavras dentro de uma frase. Examina funções das palavras, relações de interdependência. Por hora, vale comentar que algumas classes possuem palavras com pouca informação de significado e podem ser desconsideradas na análise de sentimento. Chamaremos essas palavras de *stopwords*. Como exemplo de *stopwords* pode-se citar os artigos definidos, que, em geral, pouco contribuem para a análise de sentimento.

A análise semântica preocupa-se com o significado e a organização dos significados das palavras, termos ambíguos, etc. Pode ser: descritiva, na qual se estuda a significação atual ou histórica, se preocupando com a evolução dos sentidos vocabulares através do tempo. Para esse trabalho, o foco será na semântica descritiva, visto que os textos analisados são recentes e não é esperado alteração de sentido da criação dos mesmos até o momento em que este trabalho foi escrito.

Realizar a análise de sentimento a partir de texto não estruturado em linguagem natural é uma tarefa desafiadora. Isso porquê, para realizar essa análise é necessário um profundo entendimento, tanto de forma implícita como explícita da linguagem [13]. Podemos citar como exemplos dos desafios do processamento de linguagem natural: tratamento de negação, ambiguidade, diferenciação do significado de textos afirmativos e interrogativos, etc.

Podemos incluir o assunto análise de sentimento como parte do universo do processamento de linguagem natural. A questão de processar computaci-

onalmente o problema de reconhecimento da emoção presente em um texto não estruturado é um dos grandes desafios da área de processamento de linguagem natural.

## 2.2 Aprendizado de máquina

A premissa básica da técnica de aprendizagem a partir de dados existentes, ou, do inglês *learning from data*, que foi comentado na introdução, é utilizar um conjunto de observações conhecidas para descobrir informações ainda desconhecidas[8]. Um campo muito importante dentro do universo de técnicas de aprendizado a partir de dados é o aprendizado de máquina ou, do inglês *machine learning*, onde o aprendizado é realizado por máquinas. O campo análogo ao *machine learning* é o *human learning*, onde o aprendizado é feito por humanos.

O aprendizado humano é extremamente lento, levando décadas para que o ser humano obtenha conhecimento suficiente sobre algum assunto. Dessa forma, existe uma motivação para utilizar técnicas de *machine learning* para simular o pensamento e aprendizado humano, conseguindo assim um ganho de tempo proveniente da utilização da computação [14].

Podemos dizer que *machine learning* consiste em construir de forma eficiente algoritmos que possuem uma predição precisa [6]. A preocupação com a complexidade é fundamental para projetar algoritmos eficientes em qualquer campo da computação, e não é diferente para o campo de *machine learning*. A diferença é que, agora, precisamos nos preocupar também com a complexidade e tamanho do conjunto de dados amostrais, que serão entrada do algoritmo.

De forma resumida, o objetivo de um algoritmo do campo de *machine learning*, consiste em aprender características sobre determinado conjunto de dados, que chamaremos de dados amostrais de treinamento, ou, do inglês *training sample*, e aplicar esse conhecimento em outro conjunto de dados, distinto, do qual chamaremos de dados amostrais de teste, ou, do inglês *test sample*, para prever suas características. Vale destacar que o conjunto de dados de teste não é utilizado durante a etapa de aprendizado do algoritmo.

Complementarmente, podemos dividir o conjunto de dados disponível para aprendizado em duas partes. Uma parte para fazer o treinamento propriamente dito e a outra parte para fazer a validação, refinando os parâmetros do algoritmo de *machine learning*. Fazendo essa divisão ficaríamos com três conjuntos de dados: dados amostrais de treinamento, dados amostrais de validação e dados amostrais de teste. Para simplificar chamaremos esses conjuntos de dados apenas de: dados de treinamento, dados de validação e dados de teste a partir de agora.

Foi utilizado neste trabalho o conceito de *Bag-of-words* para representação

e processamento dos textos não estruturados em linguagem natural. Esse conceito não leva em consideração nem a ordem das palavras nem a sua semântica. Existem alternativas já publicadas de outros modelos de representação como o *word2vec* [15]. Artigos científicos recentes estendem essa representação para sentenças e parágrafos [16].

Outra alternativa publicada recentemente aborda a utilização de modelos de *deep learning*, relacionados a *Convolutional Neural Networks (CNN)* para a classificação de sentenças utilizando o modelo *word2vec* como base[17].

Podemos organizar os problemas relacionados a aprendizagem nos seguintes paradigmas principais: *supervised learning*, *unsupervised learning*, *reinforcement learning*[8] e *semi-supervised learning*[18].

### 2.2.1 Aprendizagem supervisionada

A principal característica do paradigma *supervised learning* ou, aprendizagem supervisionada, é que o conjunto de dados disponível, ou de treinamento, contém exemplos explícitos de qual é a saída correta, baseado na entrada informada.

Dentro da abordagem de aprendizagem supervisionada, podemos dividir os problemas em basicamente duas categorias: classificação e regressão [6]. Na classificação, as amostras pertencem a uma ou mais classes e queremos identificar as classes dos dados de teste baseado no conhecimento adquirido a partir dos dados de treinamento. Na regressão, a ideia é similar a classificação, porém ao invés de saídas discretas, a saída do algoritmo consiste em uma variável contínua, como por exemplo um número real.

Um exemplo de problema de classificação é o de reconhecimento de dígitos escritos a mão. Podemos imaginar um conjunto de dados de treinamento com informação de imagem do que foi escrito na mão e a informação do dígito correto. Nesse caso, de forma supervisionada, o conjunto de treinamento tem cada uma de suas imagens analisadas e categorizadas em uma das dez categorias possíveis  $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

Um exemplo de problema de regressão é o de predizer o peso de um peixe baseado nas entradas de tamanho e idade do animal em anos. Nesse exemplo, o conjunto de entradas é uma variável contínua e uma variável discreta, e a saída é uma variável contínua, representando o peso do animal.

Outro conceito importante é o de *label*, ou em português, rótulo, que se refere aos valores ou categorias designados ao conjunto de dados. Em problemas de classificação, como no exemplo do reconhecimento de dígitos,

o *label* pode receber um dos dez valores possíveis, para cada um dos dez dígitos. Em problemas de regressão, como a predição do peso do animal, o *label* pode receber um valor dentro de infinitos valores reais, guardadas as devidas restrições, como por exemplo, para o caso do peso do animal, valores negativos não são válidos.

Vale reforçar que para o nosso problema relacionado a análise de sentimento de textos curtos, iremos utilizar a categoria de classificação do paradigma de aprendizagem supervisionada, e por isso a detalharemos com mais profundidade. Mais a frente, na seção de metodologia, detalharemos melhor o problema a ser tratado e o seu enquadramento nessa categoria.

### 2.2.2 Aprendizagem não supervisionada

No paradigma *unsupervised learning* ou, aprendizagem não supervisionada, o conjunto de dados disponível, ou de treinamento, não contém nenhuma informação a respeito da saída para determinada entrada. Todos os dados de treinamento se resumem a informações de entradas possíveis. Geralmente o problema nesse paradigma se resume a clusterização das entradas, onde as entradas possíveis são agrupadas por similaridade ou a questão de estimar a densidade do conjunto de entradas ou a detecção de *outliers*.

Um exemplo de problema ao qual pode ser aplicado esse paradigma é o de reconhecimento facial. Podemos aplicar a clusterização das entradas para identificar, por exemplo, em uma creche, através de similaridade, quantas professoras e quantas crianças estão presentes em determinado local, em um determinado momento. Isso porque a face do adulto em geral possui características diferentes da face de uma criança.

### 2.2.3 Aprendizagem por reforço

No paradigma *reinforcement learning* ou, aprendizagem por reforço, o conjunto de dados disponível, ou de treinamento, não contém exemplos explícitos de qual a saída correta. Ao invés disso, possui a informação de qual a saída para cada uma das diversas entradas e uma medida indicativa de quanto é boa ou ruim a determinada saída.

Um exemplo ao qual pode ser aplicado esse paradigma é o processo de aprendizagem de um determinado jogo. No xadrez, em geral, não é uma tarefa trivial identificar qual o melhor movimento em um determinado momento do jogo. Utilizando o paradigma de *reinforcement learning*, podemos

decidir por um dos movimentos possíveis e anotar qual foi o resultado. Essa informação será utilizada como conjunto de dados de treinamento para o algoritmo de aprendizado.

#### **2.2.4 Aprendizagem semi-supervisionada**

No paradigma *semi-supervised learning* ou, aprendizagem semi-supervisionada, o conjunto de dados de treinamento possui uma pequena quantidade de exemplos explícitos de qual a saída correta para cada entrada e uma grande quantidade de exemplos que não possui qualquer informação acerca da saída correta para cada entrada. Esse tipo de paradigma está entre a aprendizagem supervisionada e a aprendizagem não supervisionada.

Um dos principais motivadores para esse paradigma é que, estatisticamente, os resultados da aprendizagem semi-supervisionada superam os resultados de aprendizagem não supervisionada, melhorando a performance da curva de aprendizagem do algoritmo. Outro motivador é que muitas vezes é custoso produzir um conjunto de dados de entrada supervisionados, sendo possível definir a saída correta somente para um pequeno conjunto de entradas. Dessa forma a aprendizagem supervisionada não é aplicável e precisamos trabalhar com a aprendizagem semi-supervisionada[18].

## 2.3 Análise de Sentimento

Como vimos na seção introdutória, análise de sentimento se refere ao campo que trata de forma computacional, a opinião, sentimento e subjetividade de um texto [1].

As principais tarefas da computação afetiva e da análise de sentimentos são: reconhecimento de emoção e detecção de polaridade. Reconhecimento de emoção se preocupa em extrair o conjunto de informações relevantes de um texto, enquanto que a detecção de polaridade se preocupa em classificar as informações relevantes do texto entre: positivo, neutro ou negativo [5]. Complementar a tarefa de classificação da polaridade do texto, existem modelos de análise de sentimento que classificam também o grau de positividade ou negatividade do texto. Vale comentar que para esse trabalho, iremos nos concentrar na detecção da polaridade.

A análise de sentimento já é utilizada por empresas de forma comercial com o objetivo de analisar e aperfeiçoar estratégias de marketing e de vendas. O esforço das empresas consiste principalmente em coletar opiniões e críticas sobre a sua marca ou o seu produto na internet, em blogs, redes sociais, site corporativo da empresa, etc e processar toda essa informação para suportar no seu processo decisório [13].

Atualmente, existem algumas soluções comerciais disponíveis para atender a demanda relatada acima. Contudo, a identificação de sentimento e a sua respectiva polaridade são baseados em sentimentos expostos de forma explícita. Ou seja, ainda existe bastante espaço para evoluir na questão de identificação de sentimentos implícitos[13].

Vale comentar também outra funcionalidade já disponível nas soluções comerciais, a técnica *opinion-target*, que avalia o sentimento e a sua polaridade de um alvo específico ao invés do texto como um todo. Essa técnica é bastante utilizada na questão de avaliação de um produto ou uma marca.

Como comentado na seção de introdução, as abordagens para tratar o problema de análise de sentimento se encaixam em três categorias principais: *knowledge-based*, *statistical-based* e *hybrid-based*. Iremos apresentar a partir de agora algumas abordagens possíveis para tratar a análise de sentimento dentro de cada uma das categorias.

Os artigos científicos mais recentes com relevância de número de citações no assunto análise de sentimento de textos curtos utilizam majoritariamente a base do *Twitter*, uma das principais plataformas de *micro-blogging* da atualidade, para realizar suas análises.

Das contribuições importantes de artigos científicos recentes pode ser destacado: coletar *corpus* de *micro-blogging*s com sentimentos positivos, neutros e negativos de forma automática, de forma que nenhum esforço prévio seja necessário para a classificação dos textos [19].

Outra contribuição recente é relacionada a utilização de *micro-blogging features* como: *hashtags*, *emoticons* e intensificadores de texto como letras maiúsculas e caracteres repetidos para o treinamento e teste do algoritmo de classificação [20].

### 2.3.1 Técnicas baseadas em conhecimento

Técnicas baseadas em conhecimento ou, do inglês *knowledge-based*, são as mais acessíveis de serem utilizadas rapidamente. Isso porque, não necessitam de uma base de treinamento com *labels* identificados para iniciar o processo a análise de sentimento. Será visto que esse é um requisito da abordagem estatística mais a frente.

Dentro da categoria *knowledge-based*, uma abordagem para as tarefas de reconhecimento da emoção e detecção da polaridade é baseada em uma lista de *opinion words*, que são utilizadas para expressar sentimentos positivos e negativos, como por exemplo: bom e ruim [21]. Essa abordagem é chamada de léxica ou em inglês *lexicon-based* e também é conhecida como *bag of words* [5].

Como já comentamos na introdução, textos curtos e informais possuem algumas características específicas. Essas características, como por exemplo: utilização de *emoticons*, abreviações, expressões coloquiais, etc dificultam a abordagem léxica. Essas expressões podem até possuir emoção e polaridade bem definidas, porém não existem na lista de *opinion words*, que é a referência para a abordagem léxica. Dessa forma, essas expressões são identificadas como neutras ou sem opinião relevante para a análise de sentimento.

Para materializar esse assunto, seguem alguns exemplos de expressões coloquiais que possuem emoção e polaridade bem definidas, porém podem não constar em uma lista de *opinion words*: "sensacional!!!!!!!" e "xatiado:((((", visto as questões de erros gramaticais, erros de concordância, repetição de caracteres, uso de *emoticons*, etc. Percebemos por esses exemplos que é necessário algum tipo de pré-processamento nos dados para corrigir essas questões, antes do processamento da análise de sentimento *lexicon-based*.

Essa questão, leva a uma situação em que somente a aplicação da abordagem léxica para o problema de análise de sentimento para textos curtos, irá



trazer uma baixa revocação, ou do inglês *recall*. Isso porque é esperado que o método léxico não identifique um grande conjunto de expressões com alto teor de sentimento e polaridade bem definida [21]. Vale comentar também que apesar do *opinion lexicon* poder ser incrementado conforme necessidade, sempre teremos novas expressões e ajustes na linguagem coloquial que irão dificultar o tratamento da baixa revocação apenas com o incremento do *opinion lexicon*. Isso se deve, por exemplo, à criatividade e ao neologismo.

Ainda dentro da categoria *knowledge-based*, é possível estender a classificação utilizando outras técnicas como, por exemplo: regras linguísticas [21], *part of speech tagger* (POS-tagger) [22] e informação semântica [23]. Na técnica de regras linguísticas, vale destacar a regra de negação. A regra de negação basicamente reverte a polaridade do sentimento expressado no texto. Um exemplo é o texto: "o serviço de atendimento não foi bom".

Na técnica *POS-tagger* cada termo do texto é identificado e segregado de acordo com a sua categoria sintática. Ou seja, identificamos cada termo em uma das categorias: adjetivos, advérbios, verbos, pronomes, preposições, etc. A partir dessa categorização podemos nos beneficiar no desafio de realizar a análise de sentimento, utilizando, por exemplo, adjetivos superlativos [19] para indicar polaridade.

Uma forma de utilizar a informação semântica é identificar conceitos semânticos que, relacionados à determinada entidade, tem de forma consistente uma correlação positiva ou negativa. Assim, usamos esses termos de significado relevante para a verificação da polaridade. Um exemplo é utilizar o conceito semântico "produto Apple" aos termos: "Iphone", "Ipad", "Macbook" a medida que os mesmos aparecem nos textos. O racional é de que se em geral, textos que contém referência ao "Iphone" são positivos, isso pode ter alguma influência nos demais produtos referenciados da empresa "Apple", que fabricou o "Iphone".

### 2.3.2 Técnicas baseadas em estatística

Dentro da categoria de técnicas baseadas em estatística ou, do inglês *Statistical-based*, o mais comum é aplicar métodos de *machine-learning* para realizar a análise de sentimento [1].

Complementando o que foi comentado sobre *features* na introdução, para análise de sentimentos, utilizamos principalmente dois tipos de *features* em algoritmos de *machine-learning*. Uma delas é a presença de determinados termos no texto a ser analisado e a outra é a frequência em que os termos

aparecem no texto a ser analisado[13].

A *feature* de presença pode ser tratada como um vetor binário, que indica apenas se o termo está presente ou não no texto analisado. A utilização de termos que possuem recorrência e polaridade significativa para um determinado contexto são candidatos óbvios a serem selecionados para fazer parte do vetor binário. Presença é o meio mais efetivo para identificar a polaridade do texto, isso porquê apesar de termos recorrentes indicarem um assunto específico, a repetição de termos pode não refletir bem a polaridade do sentimento [13].

Vale comentar que quando fazermos referência a presença de termos como *features* do algoritmo de *machine-learning*, podemos utilizar também *bi-grams*, *trigrams* ou *n-grams* além de palavras. Podemos inclusive incluir a posição de determinado termo no texto como *feature*, se entendermos que o posicionamento tem influência no sentimento presente no texto.

### 2.3.3 Técnicas híbridas

Na categoria de técnicas híbridas ou, do inglês *hybrid-based*, ocorre uma combinação de métodos das categorias tratadas anteriormente, *Knowledge-based* e *Statistical-based*. A ideia é utilizar os conceitos linguísticos para pré-processamento em conjunto com uma análise de sentimento que utilize *machine-learning*. A justificativa para a utilização dessa categoria é o entendimento de que essas técnicas são complementares e a sua utilização em conjunto tende a trazer melhores resultados.

## 2.4 Desequilíbrio de classes

O problema de *class imbalance* ou desequilíbrio de classes, ocorre quando em uma situação de aprendizado supervisionado do tipo classificação, existem muito mais instâncias de uma determinada classe do que de outras [24]. Vale lembrar que na classificação, as amostras pertencem a uma ou mais classes e queremos identificar as classes dos dados de teste, baseado no conhecimento adquirido a partir dos dados de treinamento.

O que ocorre nesses casos é que o classificador tende a ser dominado pela classe com uma grande quantidade de instâncias e desconsiderar as classes com menor representatividade. Na literatura, entende-se que uma razão de aproximadamente uma instância com pouca representatividade para cada cem instâncias com grande representatividade caracteriza uma situação de *class imbalance*. Esse é um problema recorrente em situações de detecção de fraude, detecção de desvios de segurança, diagnóstico médico de doenças raras, etc.

Existe evidência, através de experimentos, de que *class imbalance* não necessariamente atrapalha o desempenho de um sistema de aprendizado. Na verdade, o problema é inerente ao aprendizado de poucas instâncias das classes minoritárias e a outros fatores complicadores, como por exemplo, a sobreposição de classes[25].

### 2.4.1 Técnicas em nível de dados

Existem algumas formas de tratar o problema de *class imbalance*, tanto no nível de dados como no nível de algoritmo. No nível de dados, existem algumas técnicas que podem ser aplicadas, como por exemplo, ajustar os dados de treinamento de forma a escolher menos amostras da classe mais significativa, em favor de escolher mais instâncias de classes menos significativas. Outra forma de tratar esse problema, é gerar amostras adicionais de acordo com a necessidade[24].

Ainda sobre técnicas relacionadas ao conjunto de amostras, é preciso observar alguns riscos inerentes as suas aplicações. *Random undersampling*, ou redução das amostras da classe majoritária de forma aleatória, pode remover importantes exemplos. A técnica de *random oversampling*, ou aumento das amostras da classe minoritária de forma aleatória, pode levar ao aumento do custo computacional e ao *overfitting*. O quanto de *undersampling* ou *oversampling* deve ser aplicado ao problema em questão é detectado de forma

empírica.

Além das técnicas não heurísticas, *random undersampling* e *random oversampling*, citadas acima, existem também técnicas heurísticas, onde ocorre uma investigação dos dados amostrais para manipulação de forma não aleatória.

Como técnicas heurísticas, podemos comentar o Synthetic Minority Over-sampling Technique (SMOTE)[26], que é um método de *oversampling*, onde a idéia é formar amostras das classes minoritárias através de interpolação de amostras existentes dessas classes. A consequência é que a fronteira referente a área minoritária se expande dentro do espaço da classe majoritária.

A ideia de uma amostragem mais inteligente demonstra que a criação de amostras sintéticas, ou adicionais, da classe minoritária ajudam a aumentar a região de decisão do algoritmo. Dessa forma, fazemos essas classes maiores e menos específicas. Esse método é chamado de *oversampling* sintético. De outro lado, é desejado uma redução de amostras da classe majoritária, que chamamos em inglês de *undersampling*. Essa combinação de métodos traz um maior equilíbrio entre as classes e resultados mais efetivos de classificação[26] de uma maneira geral.

A principal motivação de técnicas de *sampling* é remover instâncias que trazem ruído no treinamento do algoritmo, além de balancear a proporção de instâncias de cada classes no conjunto de dados.

Esse ruído é devido a essas instâncias específicas estarem no lado errado da fronteira. É como se tivessem características de uma determinada classe, porém não pertencem a mesma. A remoção dessas instâncias tem o objetivo de formar os *clusters* de instâncias de cada classe mais bem definidos. Conforme referência sobre o assunto [27], métodos de *oversampling* são mais eficientes em induzir os classificadores a atingirem resultados mais eficazes do que métodos de *undersampling*.

## 2.4.2 Técnicas em nível de algoritmo

No nível de algoritmo [28], as técnicas que podem ser aplicadas são referentes a parametrização do algoritmo de classificação. Por exemplo, pode ser feito um ajuste no processo de classificação para utilizar ao invés de um aprendizado discriminatório, onde o algoritmo tende a classificar a amostra em uma das classes possíveis, utilizar um aprendizado de reconhecimento, onde o algoritmo aprende apenas a partir da classe com menos representatividade. Esse método é conhecido, em inglês, como *one-class learning*.

Ainda no nível de algoritmo, outro aspecto importante para tratar pro-

blemas de análise de sentimento com a característica de *class imbalance* é uma escolha de *features* apropriada [29]. O aprendizado a partir de dados multidimensionais é um aprendizado caro do ponto de vista computacional e pouco preciso. Dados com várias dimensões levam a distorções na distribuição das instâncias das classes. Assim, é fundamental buscar uma seleção de *features* que privilegie a uma alta separabilidade entre as classes, para evitar sobreposições de suas fronteiras.

Apesar de existirem diversos artigos sobre o problema de *class imbalance*, descrevendo diversos métodos, não existe uma palavra final sobre o método específico que deve ser utilizado para tratar um problema. Dependendo do contexto, um método ou uma combinação de métodos será mais eficaz do que os demais.

## 2.5 Algoritmos de Classificação

### 2.5.1 *Naive Bayes* (NB)

*Naive Bayes* (NB) é um classificador simples utilizado há mais de cinquenta anos, principalmente na área de recuperação da informação[30]. Nos últimos vinte anos, o classificador vem novamente ganhando espaço na área de aprendizagem supervisionada, se estabelecendo inclusive, como um *baseline* para comparação com outros classificadores mais sofisticados [31].

A base deste classificador é o teorema da probabilidade, conhecido como teorema de Bayes ou regra de Bayes:

$$P(C = c_k | \mathbf{X}=\mathbf{x}) = P(C = c_k) \times \frac{P(\mathbf{X}=\mathbf{x} | C = c_k)}{P(\mathbf{x})} \quad (1)$$

onde:

$$P(\mathbf{X}=\mathbf{x}) = \sum_{k'=1}^{e_c} P(\mathbf{X}=\mathbf{x} | C = c_{k'}) \times P(C = c_{k'}) \quad (2)$$

Todas as pesquisas de satisfação são classificadas, exatamente, em apenas uma das  $e_c$  classes possíveis:  $(c_1, c_2, \dots, c_k, \dots, c_n)$ . No caso,  $C$  pode ser entendido como uma variável aleatória [32], cujo os valores possíveis são as classes supracitadas ou *labels* do problema de classificação. Materializando o conceito no problema em questão, um exemplo de classes possíveis seriam: "elogio", "neutro" e "reclamação".

Enquanto isso,  $\mathbf{X}$ , pode ser entendido como um vetor de variáveis aleatórias, cujo os valores  $\mathbf{x} = (x_1, x_2, \dots, x_n)$ , componentes do vetor, são as *features* da pesquisa de satisfação. Materializando o conceito no problema em questão, o vetor de *features* representa a frequência de vocábulos que aparecem em uma determinada pesquisa de satisfação.

Com o entendimento desses conceitos, podemos definir  $P(C = c_k | \mathbf{X}=\mathbf{x})$  como a probabilidade condicional que uma pesquisa de satisfação pertence à classe  $c_k$ , dado que é conhecido o seu vetor de *features*  $x$ . O teorema de Bayes especifica como essa probabilidade condicional pode ser calculada, a partir da probabilidade condicional ao conhecer vetores de *features* de cada classe  $\frac{P(\mathbf{X}=\mathbf{x} | C=c_k)}{P(\mathbf{x})}$  e a probabilidade incondicional de encontrar pesquisas de cada classe  $P(C = c_k)$ .

Considerando que  $c_k$  e  $\mathbf{x}$  são valores gerados a partir de variáveis aleatórias  $C$  e  $\mathbf{X}$  respectivamente, podemos simplificar a notação e representar o teorema de Bayes como:

$$P(c_k|\mathbf{x}) = P(c_k) \times \frac{P(\mathbf{x}|c_k)}{P(\mathbf{x})} \quad (3)$$

No caso do problema de classificação de pesquisas de satisfação, o interesse é minimizar o número de erros de classificação. Isso é feito atribuindo a pesquisa de satisfação com vetor de *features*  $\mathbf{x}$  à classe que maximiza  $P(c_k|\mathbf{x})$ .

A partir da equação (3), pode-se perceber, baseado no teorema de Bayes, que será necessário estimar  $P(c_k|\mathbf{x})$  a partir das probabilidades combinadas de:  $P(c_k)$ ,  $P(\mathbf{x}|c_k)$  e  $P(\mathbf{x})$ . O desafio é que, em geral, existem muitas possibilidades de valores de *features*  $\mathbf{x} = (x_1, x_2, \dots, x_k, \dots, x_n)$ . No problema de pesquisa de satisfação, cada vocábulo presente no corpus corresponde a um  $x_k$ .

A estratégia utilizada para tratar esse desafio é decompor a probabilidade condicional de  $\mathbf{x}$  em  $c_k$  conforme (4) e assumir a premissa de que cada valor particular de  $x_j$  é estatisticamente independente de qualquer outro valor particular  $x'_j$  dado uma pesquisa de satisfação de uma classe  $c_k$ .

$$P(\mathbf{x}|c_k) = \prod_{j=1}^n P(x_j|c_k) \quad (4)$$

substituindo (4) em (3) temos:

$$P(c_k|\mathbf{x}) = P(c_k) \times \frac{\prod_{j=1}^n P(x_j|c_k)}{P(\mathbf{x})} \quad (5)$$

onde agora temos:

$$P(\mathbf{x}) = \sum_{k'=1}^{e_c} P(c_{k'}) \times \prod_{j'=1}^n P(x'_{j'}|c'_{k'}) \quad (6)$$

A vantagem dessa estratégia de modelagem é a simplificação do problema em um número reduzido de parâmetros, possibilitando a classificação de pesquisas de satisfação em um das classes existentes. Devido a essa premissa de independência entre os valores  $x_j$  e  $x'_j$ , que não necessariamente ocorre em problemas reais, esse classificador é conhecido como Bayes ingênuo ou *Naive Bayes*. De qualquer forma, mesmo com essa simplificação, os resultados desse classificador são bastante aceitáveis, como veremos na seção 4.

### 2.5.2 *Support Vector Machine* (SVM)

*Support vector machine* ou simplesmente SVM é um método supervisionado de aprendizado, utilizado para: classificação, regressão e detecção de *outliers*. As principais vantagens são: efetividade em espaços com várias dimensões, efetividade em espaços onde o número de dimensões é maior do que o número de amostras e versatilidade, pois utiliza diferentes funções no seu *kernel* para definir a forma de decisão do algoritmo.

SVM é um método que permite uma análise matemática menos complexa, pois pode ser demonstrado a sua correspondência a um método linear em um espaço de *features* de várias dimensões, não linearmente relacionada ao espaço de entrada [33].

Classificadores do tipo *Support Vectors* ou vetores de suporte, são baseados em hiperplanos, ou seja, espaços com dimensão reduzida em relação ao ambiente espacial em questão. A máquina de vetores de suporte constrói um hiperplano ou um conjunto de hiperplanos capazes gerar modelos de classificação e regressão. Para a máquina de vetores de suporte uma boa separação é constituída de uma maior margem, dada pela menor distância entre as observações e o hiperplano que separa as classes. Quanto maior esta margem, maior é poder de generalização do modelo.

*Support Vector Machine* tenta separar as amostras baseadas em sua classe, no espaço  $n$ -dimensional, sendo  $n$ , o número de *features*, onde  $u$  e  $v$  são os vetores de suporte, perpendiculares ao hiperplano, e correspondentes aos exemplos que estão mais próximos dos limites da respectiva classe, conforme figura 2.1.

O vetor de suporte define, através do seu módulo, a margem entre o hiperplano e as amostras mais próximas de cada uma das classes. Para cada classe, o algoritmo tenta encontrar o vetor de suporte que maximiza a referida margem. Para classificar uma amostra de teste, ou sem o *label* associado, o algoritmo identifica em qual local no espaço essa amostra de teste está, e qual a sua posição em relação ao hiperplano [34]



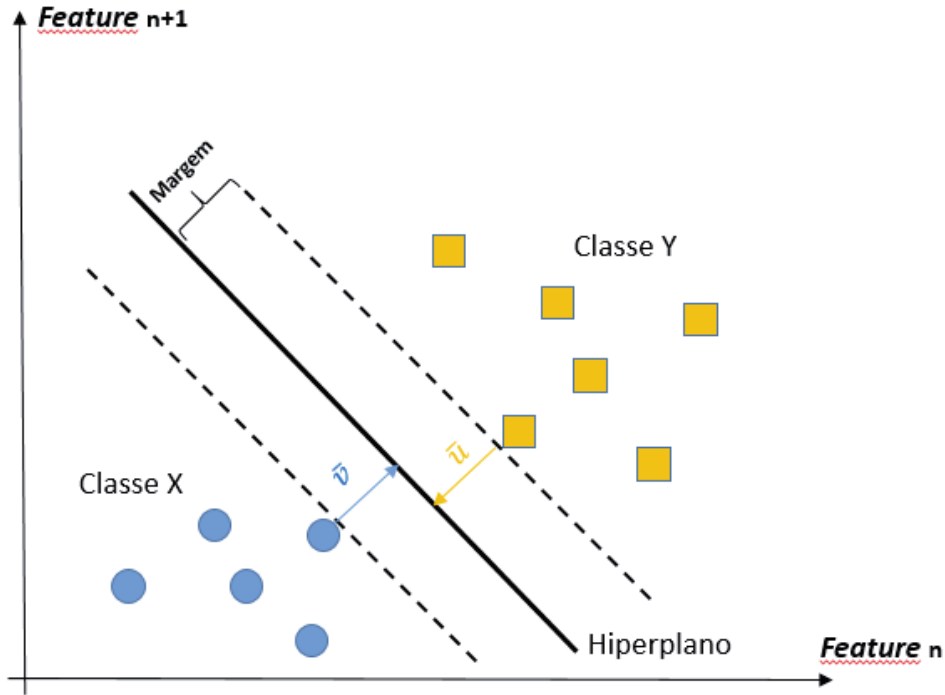


Figura 2.1: *Support Vector Machine* (SVM)

Já foi demonstrado que o algoritmo SVM é altamente efetivo na categorização de texto, tendo performance superior ao algoritmo *Naive Bayes*[35] e sendo uma referência para para o problema de análise de sentimento.

### 2.5.3 *Stochastic Gradient Descent* (SGD)

Antes de discutir o algoritmo *stochastic gradient descent* (SGD), vale comentar brevemente sobre o algoritmo *gradient descent* (GD) [36]. O algoritmo *gradient descent* pode ser descrito como um algoritmo que minimiza uma dada função, definida por um conjunto de parâmetros.

O algoritmo *gradient descent* evolui o conjunto de parâmetros de forma a minimizar a função, através de iterações no sentido negativo do gradiente da função. Para o caso do problema, foco deste trabalho, se está interessado

em minimizar a função de perda (*loss function*), que representa o preço pago pela falta de acurácia nas predições de classificação.

Dado  $\mathbf{X}$  o espaço vetorial com todas as entradas possíveis e  $Y = \{ \text{"elogio"}, \text{"neutro"}, \text{"reclamação"} \}$ , o espaço vetorial com todas as possíveis saídas, desejamos encontrar a função *loss function*  $f : X \rightarrow Y$  que melhor relaciona os dois espaços vetoriais. Devido a ruídos e falta de informações completas para essa classificação, é possível que um mesmo  $\mathbf{x} \in \mathbf{X}$  levar para mais de um  $y \in Y$ . Assim, o objetivo da aprendizagem de máquina, no contexto do algoritmo *gradient descent*, é minimizar a perda dessa função.

Resumindo, é desejado minimizar a função de perda, chamada no idioma inglês de *loss function*, que pode ser expressa como a perda média de todas as amostras do conjunto de treinamento. A computação dessa média consome um tempo proporcional ao número  $n$  de amostras. Essa constante vai aparecer de alguma forma no tempo de processamento de cada iteração de um algoritmo *gradient descent*.

O algoritmo *stochastic gradient descent* funciona de maneira ligeiramente diferente. Este algoritmo também deseja minimizar a função de perda *loss function*, porém, ao invés de avaliar todas as  $n$  amostras em cada iteração, ele avalia uma única amostra aleatória e atualiza os parâmetros baseado no resultado dessa única amostra. O algoritmo funciona porque o efeito médio de cada amostra é o mesmo. Apesar da convergência apresentar mais ruídos do que o *gradient descent*, a eliminação da constante  $n$  no processamento é uma grande vantagem para problemas de larga escala.

Na figura 2.2, é apresentado um exemplo de iterações de um algoritmo GD  $\{X_0, \dots, X_6\}$  e de um algoritmo SGD  $\{X'_0, \dots, X'_{16}\}$  para efeito de comparação. É possível perceber que além do ruído ser maior, é necessário um maior número de iterações no SGD. Em compensação, como já comentado, essas iterações tem um custo de processamento menor do que as iterações do GD. Ao final do processamento é esperado, para problemas de classificação, que exista uma convergência de ambos os algoritmos para a região de mínimo da função.

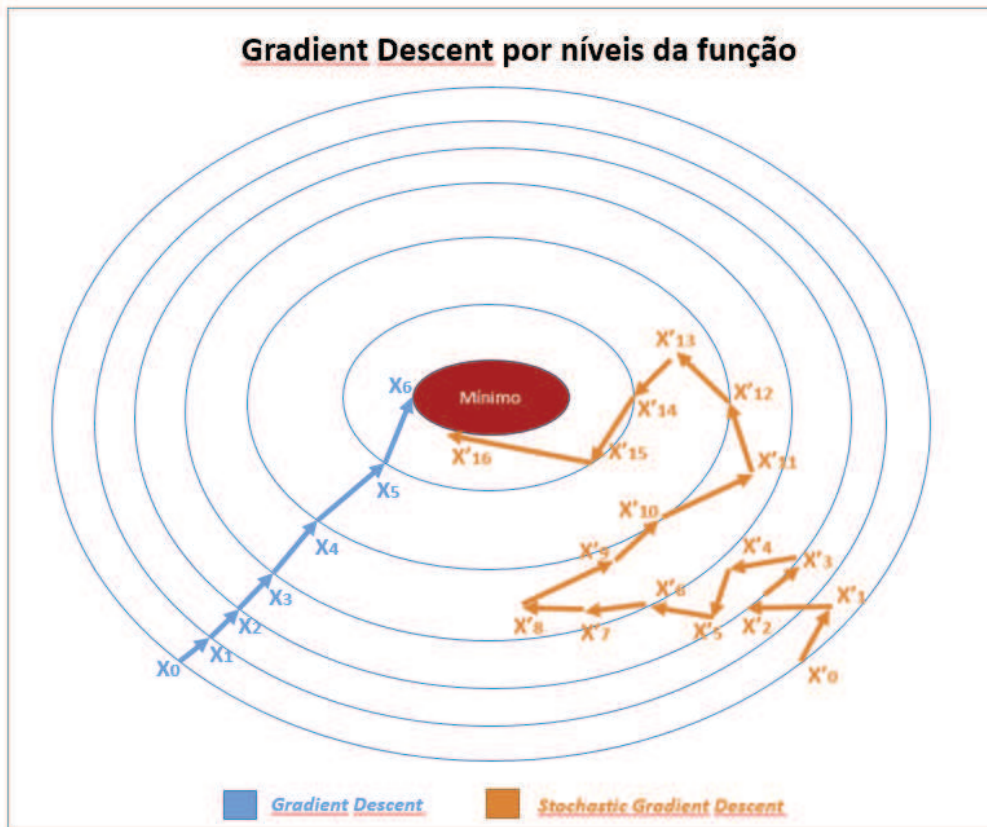


Figura 2.2: *Stochastic Gradient Descent* (SGD)

Vale comentar que o classificador SGD, da biblioteca scikit-learn, utilizado nesse trabalho, considera a função de perda *hinge*, relacionada ao algoritmo de *machine learning support vector machines* (SVM).

A função de perda é a função que relaciona os valores de uma ou mais variáveis em um número real que representa a perda associada neste relacionamento. A função de perda mede o erro empírico da classificação. Idealmente uma função de perda deve ser: convexa (otimização mais fácil, não correndo o risco de se perder em mínimos locais), robusta (tratar adequadamente *outliers*) e não necessitar de todas as *features* e amostras para especificar a solução. Vale destacar que é desejado que a função de perda

seja minimizada.

Em aprendizado de máquina, a função de perda *hinge* é utilizada para treinar o classificador, no contexto de minimizar a perda devida ao classificador prever a classe equivocada. Para um conjunto binário de saídas ou *labels* possíveis  $t = \{-1, +1\}$ , a função de perda *hinge* " $L$ " da predição do classificador  $y$  é:

$$L(y) = \max(0, 1 - t.y) \quad (7)$$

onde temos, na instância do SVM linear:

$$y = \mathbf{w} \cdot \mathbf{x} + b \quad (8)$$

Onde temos  $w$  e  $b$  como parâmetros do hiperplano e  $x$  como a entrada a ser classificada. Considerando que  $|y| \geq 1$ , quando o classificador prediz a classe correta,  $y$  e  $t$  terão o mesmo sinal de forma que  $L(y) = 0$ . Quando os sinais de  $y$  e  $t$  são opostos temos um incremento no erro  $L(y)$ .

Assim como o algoritmo SVM é comumente estendido para a classificação *multiclass*, também é possível estender a a função de perda *hinge* para o mesmo fim. Diferentes variações da função *multiclass hinge* já foram propostas, como a destacada abaixo [37]:

$$L(y) = \sum_{y \neq t} \max(0, 1 + \mathbf{w}_t \cdot \mathbf{x} - \mathbf{w}_y \cdot \mathbf{x}) \quad (9)$$

#### 2.5.4 XGBoost

O algoritmo *XGBoost* ou "*Extreme Gradient Boosting*" é baseado na técnica de *gradient boosting machine* (GBM) [38] e pode ser aplicado no contexto de aprendizagem supervisionada. Com o intuito de ser mais específico, vale comentar que o modelo GBM utilizado pelo *XGBoost*, algoritmo que utilizaremos nesse trabalho, é o *tree ensemble*, ou, conjunto de árvores.

*Tree ensemble* é um conjunto de árvores CART, *Classification and Regression Trees*, onde cada nó folha da árvore tem um *score* associado, possibilitando uma interpretação detalhada sobre o contexto em questão. Como o próprio nome já explicita, em geral, uma única árvore não é suficiente para realizar a classificação, sendo necessário utilizar um conjunto de árvores CART para fazer a predição.

Segue um amostra de árvore CART, extraída do processamento dos dados de pesquisa de satisfação na figura 2.3. Essa amostra trata de uma pequena parte de uma das árvores utilizadas no algoritmo.

Cada nó de decisão avalia se o registro de pesquisa de satisfação possui ou não determinada frequência de um vocábulo. De acordo com as respostas do conjunto de avaliações, a pesquisa é acomodada em um nó folha. Cada nó folha possui um score associado, que irá auxiliar na classificação da pesquisa de satisfação em uma das classes disponíveis.

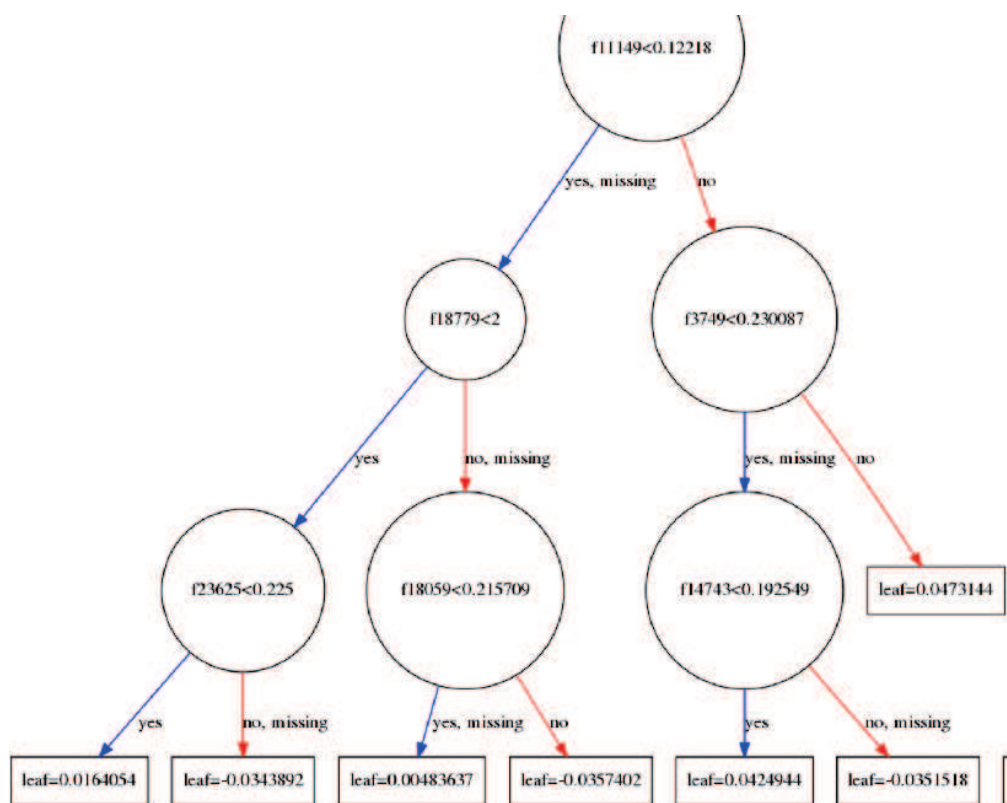


Figura 2.3: Amostra de uma árvore CART

Existe uma diferença importante entre o algoritmo de *gradient descent*, visto anteriormente, e o *gradient boosting machine*. Enquanto o *gradient descent* busca os melhores parâmetros para minimizar a função de perda, o

*gradient boosting machine* busca no espaço de funções, a melhor função para minimizar a função de perda.

O *gradient descent* atualiza os parâmetros da função passo a passo, com objetivo de encontrar o mínimo local da função de perda. Enquanto que o *gradient boosting machine* adiciona novos termos a função já existente com o objetivo de encontrar o mínimo local da função de perda.

A função resultante do algoritmo *gradient descent* é a mesma função utilizada no início do processo de minimizar a função de perda, com parâmetros otimizados, enquanto que no algoritmo *gradient boosting*, ao final do processamento, a função de perda é diferente da função utilizada no início do processamento.

## 3 Metodologia

### 3.1 Descrição dos dados

Os dados que serão utilizados neste trabalho são de pesquisas de satisfação de serviços de tecnologia da informação da Petrobras. A área de tecnologia da informação e comunicações, também conhecida como TIC, presta serviços para toda a empresa Petrobras. Após a conclusão de um serviço solicitado pelo usuário, o sistema de gestão de serviços de TIC gera automaticamente um e-mail para o usuário que solicitou o serviço. Esse e-mail contém um link, onde a pesquisa de satisfação pode ser acessada e respondida, conforme figura 3.1.

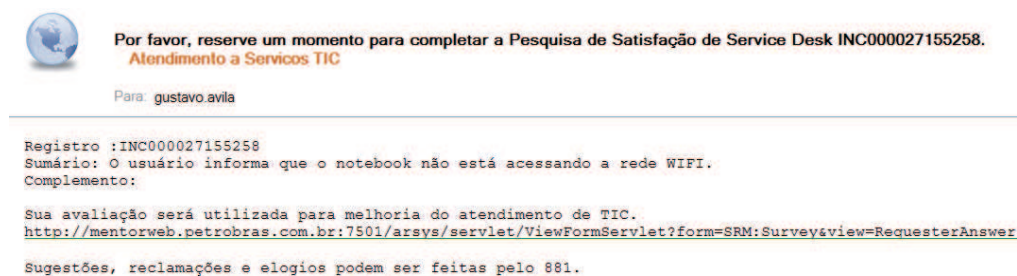


Figura 3.1: E-mail sobre a pesquisa de satisfação

No e-mail é possível observar as seguintes informações: Registro, com um número de identificação única, Sumário, com uma breve descrição sobre o serviço prestado ao usuário e Complemento, com um detalhamento adicional sobre o serviço. Essas informações são divulgadas para auxiliar o usuário na identificação do serviço prestado em que ele é convidado a avaliar.

Vale comentar que o serviço de tecnologia da informação prestado ao usuário é categorizado em dois grandes tipos: solicitação de serviço e incidente. Solicitação de serviço é definida como uma requisição formal de um usuário para algo a ser fornecido, por exemplo, uma requisição para informações ou aconselhamento, para redefinir uma senha ou para instalar uma estação de trabalho para um novo usuário. Incidente é definido como uma interrupção não planejada de um serviço de TIC ou uma redução da quali-

dade de um serviço de TIC [39]. Neste trabalho serão tratadas pesquisas de satisfação oriundas tanto de solicitações de serviço como de incidentes.

Ao clicar no link recebido por correio eletrônico, o usuário irá acessar a tela da pesquisa de satisfação, que contém novamente algumas informações relevantes sobre o registro, para auxiliar na identificação do serviço prestado em que ele é convidado a avaliar, como: número de identificação única, descrição da solicitação, complemento, status do registro, empresa e tipo de solicitação conforme figura 3.2.

Tecnologia da Informação e Telecomunicações

**Pesquisa de Satisfação**

ID Solicitação: INC000027155258 Status\*: Respondido

Descrição da Solicitação: O usuário informa que o notebook não está acessando a rede WIFI

Complemento:

Empresa\*: PETROBRAS

Tipo de Solicitação\*: Incidente

Responda a seguinte pergunta utilizando a escala de 1 a 5, onde a nota 1 = Muito Insatisfeito e a nota 5 = Muito Satisfeito.

A) Qual o seu nível de satisfação com esse atendimento? Nota (1-5) 5

B) Seu chamado foi resolvido? (Sim ou Não) Sim

Comentário: tudo ótimo

Os comentários registrados nesta pesquisa serão analisados, podendo dar origem a um elogio, reclamação ou sugestão. Neste caso, o tratamento será reportado posteriormente a você, através de nota de correio.

Enviar Fechar

Figura 3.2: Tela da pesquisa de satisfação

O campo empresa possui os valores: "PETROBRAS", "PETROBRAS - ENGLISH" e "PETROBRAS - ESPANOL", com o intuito de segregar pesquisas de satisfação de diferentes idiomas. Serão analisadas as pesquisas elaboradas e respondidas no idioma português. As pesquisas relacionadas aos



idiomas inglês e espanhol, são minoria e exigiriam tratamento diferenciado [40] para a análise de sentimento, e por isso serão desconsideradas.

O campo "Tipo de Solicitação" identifica se o registro é um incidente ou uma solicitação de serviço. No caso do registro INC000027155258, utilizado como exemplo nas figuras 3.1 e 3.2, se trata de um incidente. Isso porque se trata de uma interrupção não planejada do serviço de rede *wi-fi*, disponibilizado pela TIC.

O campo "Status" localizado no canto superior direito da tela de pesquisa de satisfação, indica se a pesquisa de satisfação já foi respondida ou não. Vale comentar que para o exemplo retratado na figura 3.2, o status está como "Respondido", visto que a pesquisa já foi preenchida anteriormente.

Ademais das informações sobre o registro, o usuário tem a possibilidade de responder duas perguntas objetivas e, opcionalmente, preencher o campo texto livre de comentário. A pergunta "A" é definida como "Qual o seu nível de satisfação com esse atendimento?" e aceita uma avaliação de um a cinco, sendo o menor valor para demonstração de insatisfação e o maior valor para demonstração de satisfação. A pergunta "B" é definida como "Seu chamado foi resolvido?" e aceita exatamente dois valores possíveis, sim ou não.

O campo de comentário é um campo de texto livre, onde o usuário pode manifestar detalhadamente a sua percepção sobre o serviço e o atendimento prestado pela TIC. Este trabalho será concentrado naquelas pesquisas de satisfação que possuem algum texto de comentário preenchido pelo usuário. A análise de sentimento será feita utilizando essencialmente esse campo.

Os dados coletados envolvem o período de janeiro de 2015 até junho de 2016 conforme figura 3.3. O sistema de gestão de serviços de TIC gerou uma média mensal de 158.841 pesquisas, que é proporcional a quantidade de serviços prestados no período supracitado. Dessas pesquisas geradas, em média, por mês, 29.416 pesquisas foram efetivamente respondidas, o que corresponde a aproximadamente 18,5% da média das pesquisas geradas. Dessas pesquisas respondidas, em média, por mês, 5.827 pesquisas foram comentadas, ou seja, tiveram o campo "Comentário" preenchido com algum valor, o que corresponde a aproximadamente 19,8% da média de pesquisas respondidas.

Em média, 3,67% da média de pesquisas geradas, foram respondidas com algum comentário no campo de texto livre denominado "Comentário". Conforme já mencionado neste trabalho, foram consideradas apenas pesquisas de satisfação no idioma português. Dessa forma, ainda será realizado um pré-processamento dos dados para adequação necessária, antes de iniciar a

análise de sentimento. Essas adequações serão explicitadas posteriormente.

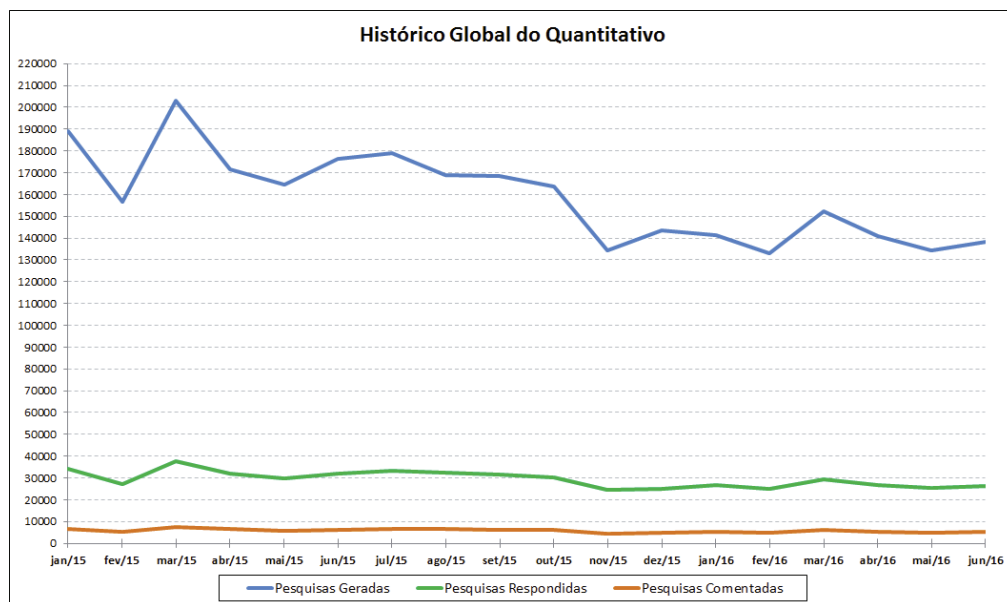


Figura 3.3: Histórico Global do Quantitativo

Os dados coletados pelo sistema de gestão de serviços de TIC relacionados à pesquisa de satisfação são descritos detalhadamente na tabela 3.1. Vale destacar em **negrito**, aqueles campos que iremos utilizar neste trabalho com o intuito de realizar a análise de sentimento. Esses dados foram extraídos do sistema de gestão de serviços de TIC em formato ".csv".

O campo "Nota Questão 1" está relacionado a resposta dada pelo usuário à pergunta "A" da tela de pesquisa de satisfação, conforme figura 3.2. Dessa forma, esse campo assume valores de um a cinco, sendo convencionado o menor valor para demonstração de insatisfação e o maior valor para demonstração de satisfação.

O campo "Nota Questão 2" está relacionado a resposta dada pelo usuário à pergunta "B" da tela de pesquisa de satisfação. Dessa forma, esse campo assume valores de um ou cinco, sendo convencionado o menor valor para indicação de que o chamado ou serviço não foi resolvido ou realizado. O maior valor é convencionado para indicação de que o chamado ou serviço foi

resolvido ou realizado.

CAMPO	DESCRIÇÃO
Número do Incidente	número de identificação única do registro
Status	situação do registro
Data da Resposta	Indica data da resolução do registro
Organização de Suporte	Organização de TIC que resolveu o registro
Grupo Designado	Grupo de TIC que resolveu o registro
Nível 1, Nível 2 e Nível 3	Categorização do registro
Questão 1	"Qual o seu nível de satisfação com esse atendimento?"
Questão 2	"Seu chamado foi resolvido?"
Questão 3 e Questão 4	não utilizado
<b>Nota Questão 1</b>	<b>Valor de 1 a 5 referente a Questão 1</b>
<b>Nota Questão 2</b>	<b>Valor 1=Não e 5=Sim referente a Questão 2</b>
Nota Questão 3 e Nota Questão 4	não utilizado
Organização, Localização e Departamento	Dados do usuário que solicitou o serviço
<b>Comentário</b>	<b>Texto livre preenchido pelo usuário</b>
P1 (Satisfação)	Calculado automaticamente baseado na "Nota Questão 1"
P2 (Resolução)	Calculado automaticamente baseado na "Nota Questão 2"
<b>Manif.Atendimento</b>	<b>Avaliação humana sobre o atendimento</b>
<b>Manif.Serviço</b>	<b>Avaliação humana sobre o serviço</b>
Ações e Registro Reabertura	Ações de gestão relacionadas a reabertura do registro
Data corrigida	Calculado baseado na "Data da Resposta"

Tabela 3.1: Campos da Pesquisa de Satisfação e sua descrição

O campo "Comentário" está relacionado diretamente ao comentário, em

texto livre, feito pelo usuário na tela da pesquisa de satisfação. Já os campos "Manifestação ao Atendimento" e "Manifestação ao Serviço" contém a avaliação humana sobre o registro, preenchida manualmente, com a percepção sobre o atendimento e o serviço respectivamente. Esses campos são preenchidos por um prestador de serviços da Petrobras e são auditados mensalmente para garantir a acurácia do processo. Existe a expectativa de que essa tarefa repetitiva possa ser automatizada, utilizando as técnicas de análise de sentimento.

Seguem alguns exemplos de registros na tabela 3.2 para ajudar no entendimento dos principais campos, destacados anteriormente:

<b>Nota Questão1</b>	<b>Nota Questão2</b>	<b>Comentário</b>	<b>Manif. Atendi- mento</b>	<b>Manif. Serviço</b>
5	1=NÃO	Software voltou a trava. Mas reporto muito bom atendimento da equipe.	Elogio	Reclamação
5	5=SIM	Ok	Neutro	Neutro
3	5=SIM	Achei muito tempo para resolver o problema	Reclamação	Neutro
5	5=SIM	O toner foi trocado com agilidade, e o colaborador foi muito educado, excelente atendimento.	Elogio	Elogio
5	5=SIM	Boa tarde Fernanda ,muito obrigado	Agradecimento	Neutro
2	1=NÃO	O computador deveria ser testado antes de considerar o serviço como concluído. O teclado não está funcionando.	Reclamação	Sugestão

Tabela 3.2: Exemplos de registros

Vale destacar que, ao preencher manualmente a percepção sobre o atendimento e sobre o serviço de um registro específico, existe um conjunto de valores válidos possíveis: "Elogio", "Reclamação", "Neutro", "Agradecimento",

"Sugestão", "Insatisfeito" e "Não-Resolvido". O prestador de serviço é obrigado a escolher um desses valores válidos em sua avaliação sobre o registro.

No modelo análise de sentimento que será construído, utilizando técnicas de *machine learning*, os campos "Nota Questão 1", "Nota Questão 2" e o campo texto livre "Comentário" são candidatos a serem *features*, enquanto que os campos "Manif. Atendimento" e "Manif. Serviço" são candidatos a serem *label* dos registros. A seção relacionada ao pré-processamento dos dados, detalhará como esses campos candidatos serão aproveitados.

## 3.2 Pré-processamento

Conforme comentado na seção introdutória, será utilizada a linguagem de programação Python em conjunto com a biblioteca Scikit-Learn para tratar do processamento de linguagem natural e *machine learning*. Conforme necessário, blocos de código serão destacados e expostos, com o intuito de facilitar o entendimento do texto e do que foi feito na construção do modelo de análise de sentimento de textos curtos.

### 3.2.1 Valores atribuídos na manifestação

O primeiro pré-processamento necessário é em relação aos valores possíveis a serem atribuídos aos campos "Manifestação ao Atendimento" e "Manifestação ao Serviço". Na base de dados, como foi comentado na seção de descrição dos dados, era possível atribuir os seguintes valores: "Elogio", "Reclamação", "Neutro", "Agradecimento", "Sugestão", "Insatisfeito" e "Não-Resolvido".

Com a visão da análise de sentimento e baseado no objetivo de identificar a neutralidade ou a polaridade de uma pesquisa, vale restringir esses valores para especificamente: "Elogio", "Neutro" e "Reclamação", representando polaridade positiva, neutralidade e polaridade negativa respectivamente. Dessa forma, se torna necessário uma transformação dos dados tanto do campo "manifestação ao serviço" como do campo "manifestação ao atendimento". Segue, abaixo, como essa transformação foi feita no código 3.1.

---

```

1  #Normalizar 'Manifestação' em: elogio, neutro e reclamação.
2  #Input : dataframe e campo X a ser normalizado
3  #Output: MANIFESTAÇÃO normalizada
4  def setmanifestacao2 (DF,X):
5      positivo=["elogio","agradecimento"]
6      neutro  =["neutro","sugestão"]
7      negativo=["reclamação","insatisfeito","não-resolvido"]
8      valido=positivo+neutro+negativo
9
10     if (str.lower(X) in negativo):
11         return "reclamação"
12     elif (str.lower(X) in positivo):
13         return "elogio"
14     elif (str.lower(X) in neutro):
15         return "neutro"
16     elif (str.lower(X) not in valido):
17         return "invalido"
18     else:
19         return "erro"

```

---

Código 3.1: Transformação de valores atribuídos na manifestação

### 3.2.2 Sumarização no campo manifestação geral

Vale comentar que, apesar de existirem dois campos, "Manifestação ao Atendimento" e "Manifestação ao Serviço", utilizados na base de dados, para registrar a manifestação geral do usuário em relação a sua experiência com a TIC, utilizaremos no nosso modelo um único campo para resumir a manifestação geral do usuário, adequando tanto os dados de treinamento como na classificação do algoritmo de análise de sentimento.

Isso porquê, foi observado que, de forma majoritária, as manifestações são referentes apenas ao atendimento, conforme figura 3.4, e os próprios responsáveis pela pesquisa de satisfação na Petrobras, enxergam pouco valor nessa informação segregada.

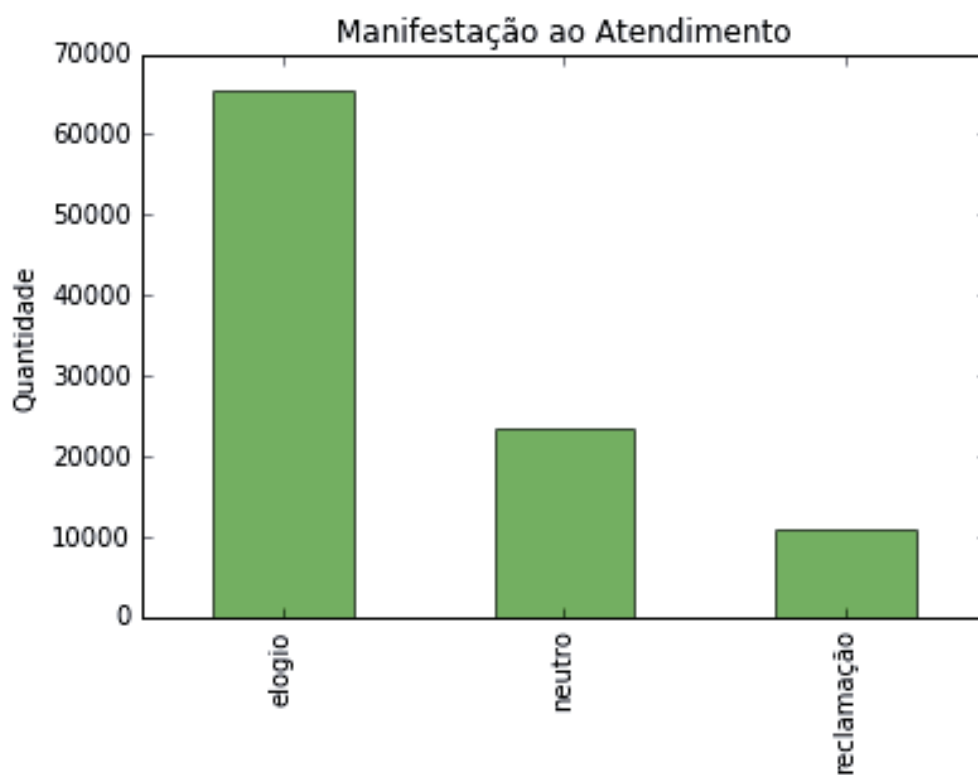


Figura 3.4: Manifestação ao Atendimento

Foi observado que, em média, menos de 5% das manifestações, conforme figura 3.5, fazem alguma referência ao serviço. Dessa forma, foi entendido que o esforço necessário para avaliar o atendimento e o serviço de forma segregada, utilizando análise de sentimento, não justificava o benefício.



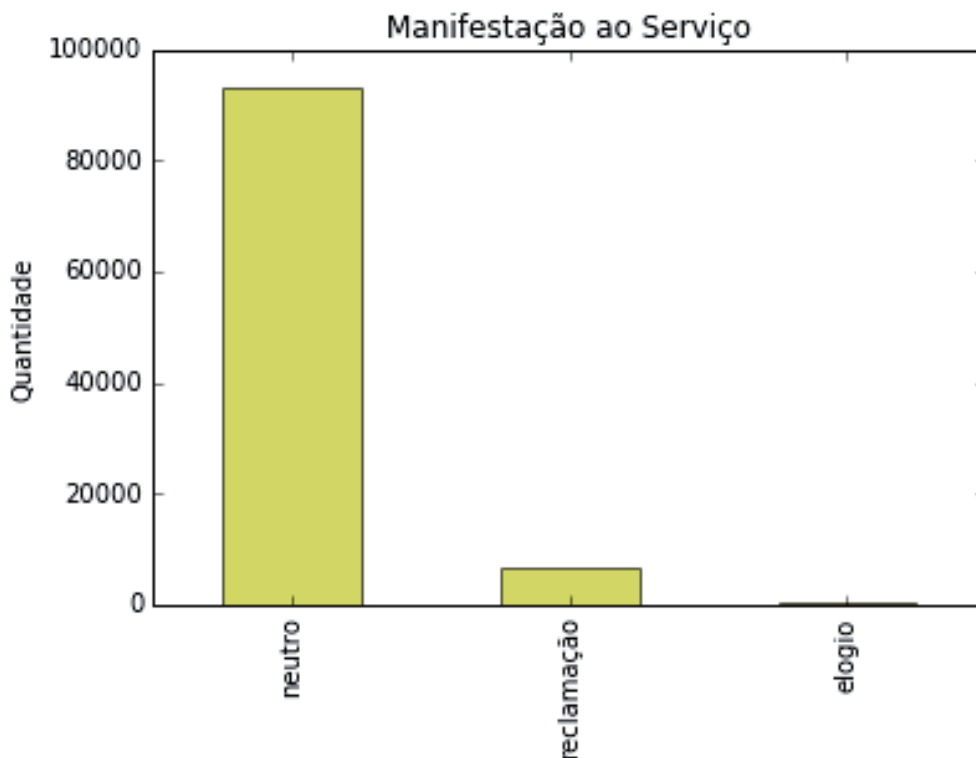


Figura 3.5: Manifestação ao Serviço

Utilizando essa premissa, ou seja, considerando um único *label* para a manifestação, conseguimos deixar o modelo a ser construído mais generalista, pois exige a necessidade de um conjunto de dados pré-classificado com um único *label* ao invés de dois.

Para essa sumarização, foi utilizado como premissa que qualquer manifestação de polaridade negativa, que chamamos de "Reclamação", é mais relevante e sobrepõe os demais tipos de manifestação. Outra premissa complementar é que, qualquer manifestação de polaridade positiva, que chamamos de "Elogio", sobrepõe a manifestação "Neutro". Dessa forma, o código 3.2, explicita como a regra de sumarização foi implementada.

---

```

1  #Normalizar 'Manifestação' summarizada
2  #Input : dataframe e campos de MANIFESTAÇÃO
3  #Output: MANIFESTAÇÃO normalizada
4  def setmanifestacao (DF,ATEN,SERV):
5      positivo=["elogio","agradecimento"]
6      neutro  =["neutro","sugestão"]
7      negativo=["reclamação","insatisfeito","não-resolvido"]
8      valido=positivo+neutro+negativo
9
10     if (str.lower(ATEN) in negativo) or
11         (str.lower(SERV) in negativo):
12         return "reclamação"
13     elif (str.lower(ATEN) in positivo) or
14         (str.lower(SERV) in positivo):
15         return "elogio"
16     elif (str.lower(ATEN) in neutro) and
17         (str.lower(SERV) in neutro):
18         return "neutro"
19     elif (str.lower(ATEN) not in valido) or
20         (str.lower(SERV) not in valido):
21         return "invalido"
22     else:
23         return "erro"

```

---

Código 3.2: Sumarização da Manifestação

Após a sumarização dos campos "Manifestação ao Atendimento" e "Manifestação ao Serviço" em um único campo para registrar a "Manifestação Geral" do usuário em relação a sua experiência com a TIC, ficamos com a seguinte segmentação ilustrada na figura 3.6.

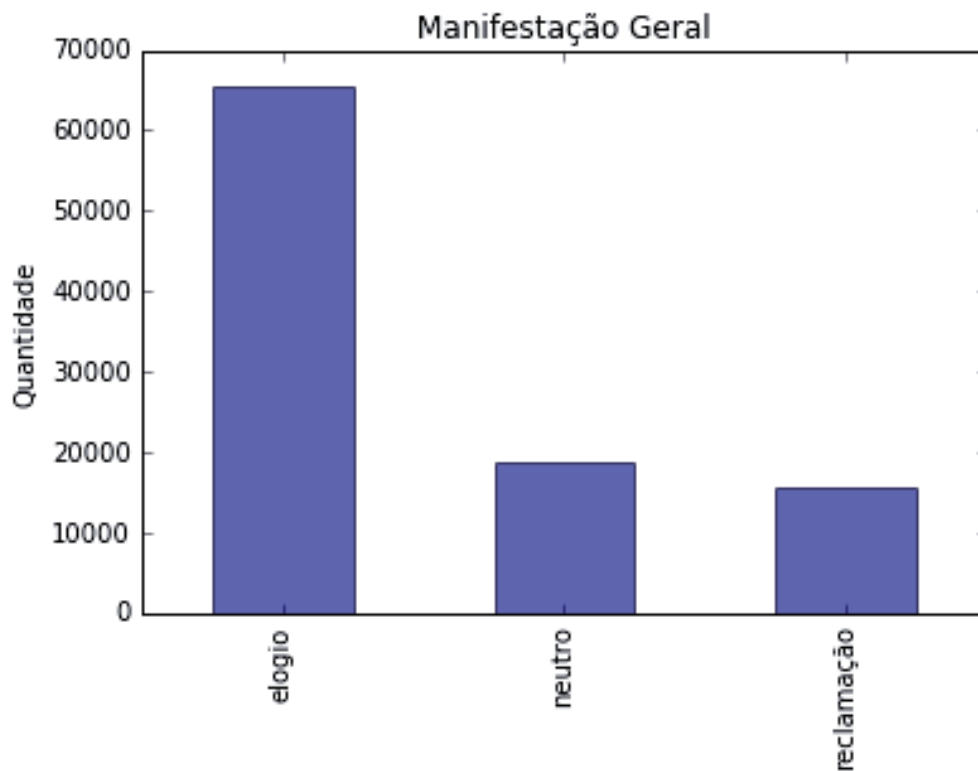


Figura 3.6: Manifestação Geral

### 3.2.3 Adequação de idioma

Como já comentado, o foco é na análise das pesquisas elaboradas e respondidas no idioma português. Dessa forma, se torna necessário identificar na base de dados apenas os registros com essa característica, como detalhado no código 3.3, através da função "setidioma", que recebe como parâmetro "Q", o campo "Questão 1" do registro.

---

```
1 def setidioma (DF,Q):
2     if Q == "Qual o seu nível de satisfação com esse atendimento":
3         return 1
4     elif Q == "Cual es su grado de satisfaccion con este servicio":
5         return 2
6     else:
7         return 3
```

---

Código 3.3: Adequação de Idioma

### 3.2.4 Padronização do texto livre

De forma a padronizar o campo "Comentário", do tipo string, foram aplicadas as seguintes funções nativas do python: *str.lower()*: que retorna uma cópia da string com todos os caracteres em caixa baixa, ou do inglês *lowercase* e *str.split()* e *str.strip(',')*: que retorna as palavras na string separadas por um delimitador e com os caracteres iniciais e finais passados como parâmetro removidos.

---

```

1  #STRIP-remove caracteres,
2  #LOWER-transforma caracteres em minúsculo
3  #SPLIT-Separa as palavras no "Comentário"
4
5  words3 = [(i.strip('.').lower().split(),category,n1,n2,m1,m2,ID,
6             data,id_inc,nivel1) for i,category,n1,n2,m1,m2,ID,
7             data,id_inc,nivel1 in
8             map(lambda*a:a,*zip (*itertools.zip_longest(
9             dict3.get('Comentário'),
10             dict3.get('Manifestação Ajustada'),
11             dict3.get('Nota Questão 1'),
12             dict3.get('Nota Questão 2'),
13             dict3.get('Manifestação ao Atendimento'),
14             dict3.get('Manifestação ao Serviço'),
15             dict3.get('ID'),
16             dict3.get('Data'),
17             dict3.get('Número do Incidente'),
18             dict3.get('Nível 1')))))]

```

---

Código 3.4: Padronização do Texto Livre

### 3.2.5 Correção ortográfica

Para a correção ortográfica, utilizaremos a biblioteca python *PyEnchant* [41] ou referenciada também como somente *Enchant* que suporta o idioma português do Brasil. Foi utilizado o arquivo "enchant-personal word list.txt", com as palavras reservadas, que possuem significado no contexto da pesquisa de satisfação e não devem ser tratados como equívocos, sujeitos a correção ortográfica. Seguem exemplos: "sap", que se refere a um sistema da empresa e "tic", que se refere a um departamento da empresa.

A partir da funcionalidade de sugestões de correção ortográfica, do *Enchant*, foi construído durante o desenvolvimento deste trabalho, o arquivo "enchant-correction word list.txt", com todas as correções necessárias a partir das sugestões do *Enchant*. Segue um exemplo de correção realizado: vocábulos "òtimo", "otmo" e "ótmo" substituídos pelo vocábulo "ótimo". Vale comentar que o tratamento foi dado, principalmente, a erros de acentuação e erros de digitação percebidos.

### 3.3 Processamento dos dados

A partir da realização do pré-processamento, temos os valores atribuídos a manifestação normalizados e sumarizados no campo "Manifestação Geral", as pesquisas de satisfação no idioma português segregadas das demais pesquisas em outros idiomas e o texto livre no campo "Comentário" padronizado e tratado com o pacote *Enchant*. Assim, os dados estão prontos para serem processados.

O processamento de dados utilizará a abordagem de aprendizagem supervisionada para tratar um problema de classificação de pesquisas de satisfação. O objetivo do processamento é classificar os dados de teste, baseando-se no conhecimento adquirido a partir dos dados de treinamento.

#### 3.3.1 Seleção de *features*

A partir de experimentos para análise e classificação de sentimento [42] pode-se concluir que utilizar técnicas para seleção de *features* baseadas no *corpus* são mais eficientes e apresentam melhores resultados do que as técnicas para seleção de *features* baseadas apenas na intuição.

Em relação às técnicas baseadas na intuição, estamos nos referindo a indicar determinados vocábulos para sentimentos positivos, neutros e negativos. Como exemplos de vocábulos para sentimentos positivos temos: "melhor" e "rápido". Como exemplos de vocábulos para sentimentos negativos temos: "pior" e "lento". Como exemplo de vocábulo para sentimento neutro temos: "ok". Nessa técnica, a pesquisa terá um sentimento positivo, caso tenha mais vocábulos positivos do que vocábulos neutros ou negativos.

Em relação a técnicas baseadas no *corpus*, estamos nos referindo à formas de extrair *features* numéricas que podem ser utilizadas nos algoritmos de *machine learning*, a partir do texto a ser analisado. No caso deste trabalho, o texto a ser analisado consiste no conjunto de vocábulos presentes no campo de texto livre "Comentário" de cada uma das pesquisas de satisfação pré-processadas.

Falando com mais detalhe das técnicas baseadas no *corpus*, inicialmente, é feito o *tokenizing*, onde cada vocábulo separado por vírgula ou ponto é isolado como um *token*, e recebe um identificador único. Após o *tokenizing*, realizamos a etapa de *counting*, que faz a contagem do número de ocorrências dos tokens em cada pesquisa de satisfação. Após o *counting*, é possível normalizar e balancear o peso de *tokens* que ocorrem na maioria das pesquisas

de satisfação.

Após as etapas de *tokenizing*, *counting* e, opcionalmente, a normalização e o balanceamento, temos o *corpus* representado como uma matriz, onde cada linha da matriz representa uma pesquisa de satisfação e cada coluna da matriz representa um *token*. Vale comentar que este processo de extração das *features* numéricas ignora a posição relativa dos vocábulos dentro do texto.

Para implementar a seleção de *features* numéricas foram utilizados quatro modelos: binário, *bag-of-words* ou *raw term frequency(tf)*, *term frequency (tf)* normalizado e *tf-idf (term frequency - inverted document frequency)* [43]. Esses modelos foram comparados entre si, utilizando um conjunto de cinco algoritmos de *machine learning*, que serão detalhados a seguir.

Ainda em relação aos quatro modelos a serem utilizados, vale comentar que todos possuem a etapa de *tokenizing* similar. Na etapa de *counting*, o modelo binário possui apenas dois valores válidos por coluna para as linhas da matriz: 1, no caso de presença do *token* na pesquisa e 0, no caso de ausência do *token* na pesquisa.

Na etapa de *counting* do modelo *bag-of-words*, cada linha da matriz terá um valor inteiro atribuído por coluna, diretamente relacionado à quantidade de vezes que o *token* aparece na pesquisa. Na etapa de *counting* do modelo *term frequency* normalizado, cada linha da matriz tem norma igual a um, com valores, por coluna, relacionados à quantidade de vezes que o *token* aparece na pesquisa.

Finalmente, o modelo *tf-idf*, utiliza um esquema bastante utilizado tanto em recuperação da informação como em classificação. A frequência normalizada do termo que utilizamos no modelo *tf* normalizado é agora submetida ao inverso da sua frequência nas pesquisas de satisfação do *corpus*. A ideia é diminuir o impacto provocado por *tokens* muito frequentes que, em geral, possuem pouca informação associada. Exemplos típicos que podem ser citados de *tokens* com pouca informação são os artigos definidos: "o", "a", "os" e "as".

Vale destacar que os artigos possuem sim valor semântico, inclusive modificando o significado da palavra porém como em: "o caixa" (funcionário) e "a caixa" (objeto). Porém no modelo *bag-of-words* utilizado, considerando somente *unigrams*, é esperado que os *tokens* segregados de artigos definidos não tenham informação associada relevante.

### 3.3.2 Algoritmos de aprendizado de máquina

Serão utilizados os algoritmos de *machine learning* apresentados na seção 2.5 para construção dos modelos de análise de sentimento. Utilizaremos os algoritmos: *Naive Bayes*, *Stochastic Gradient Descent(SGD)*, *XGBoost* e *Support Vector Machine (SVM)*.

Como existem resultados contraditórios em relação ao melhor modelo para seleção de *features* na área de categorização de texto e análise de sentimento, onde alguns artigos indicam a superioridade do modelo binário [42] e outros artigos indicam a superioridade do modelo relacionado a frequência [44], conforme o algoritmo utilizado, foi feita uma análise dos resultados de acurácia para a análise de sentimento de pesquisas de satisfação. Segue na tabela 3.3 a acurácia, em percentual, dos resultados por modelo e algoritmo.

	Modelo	Naive Bayes	SGD	XGB	SVM Linear
(1)	Binario	81,6%	84,7%	78,9%	83,8%
(2)	Bag of Words	81,6%	84,8%	78,9%	84,1%
(3)	TF normalizado	81,3%	84,1%	79,5%	<b>85,0%</b>
(4)	TF-IDF	81,4%	84,5%	79,5%	84,6%

Tabela 3.3: Resultados dos modelos de seleção de *features* por algoritmo

Como pode ser percebido na linha (3) da tabela 3.3, a melhor performance foi alcançada utilizando o modelo TF normalizado, utilizando o algoritmo SVM, com kernel linear, destacado em negrito. Como resultado dessa verificação, será utilizado esse modelo para os próximos experimentos. Segue código 3.5 com o detalhe da implementação da seleção de *features*.

```
1 count_vect = CountVectorizer()
2 count_vect.fit(df_scikit['text'].values)
3 data = count_vect.transform(df_scikit['text'].values)
4
5 tf_transformer = TfidfTransformer(use_idf=False).fit(data)
6 data_features_tf = tf_transformer.transform(data)
```

Código 3.5: Seleção de *features*, utilizando modelo TF Normalizado



Vale comentar que o modelo utilizado para seleção de *features*, contempla apenas *unigrams*, sem a utilização de *stemming*. Conforme referências [42], a utilização de *bigrams*, *n-grams*, *parts of speech (pos)*, restrição de *features* à apenas de adjetivos não contribui para melhorar a performance em relação a utilização apenas de *unigrams*, em cenários de análise e classificação de sentimento de textos curtos. Essa expectativa foi comprovada mediante testes, realizados no escopo deste trabalho.

É importante destacar que a afirmação no parágrafo anterior não diminui o valor da seleção de *features* utilizando *bigrams*. A utilização de *bigrams* é igualmente útil assim como a utilização de *unigrams*. Conforme referência,[45] já foi identificado inclusive que a utilização apenas de *bigrams* pode ser efetiva para evitar a ambiguidade de vocábulos. Entretanto, ao utilizar os dados de pesquisa de satisfação de serviços de TI com os modelos escolhidos, a utilização de *bigrams* não contribuiu para melhorar a qualidade do modelo.

Intuitivamente, poderia se esperar que os adjetivos carregassem uma grande carga de informação em relação ao sentimento da pesquisa de satisfação ou ao texto que está sendo analisado. Apesar disso, conforme referências[42], o modelo utilizando apenas adjetivos do *corpus*, possui resultados inferiores ao modelo de *unigrams*, não justificando o esforço para tratamento do *corpus*, em relação a identificação de adjetivos.

### 3.3.3 Utilização de *features* adicionais

Em relação a seleção de *features*, nas seções 3.3.1 e 3.3.2 foi tratado basicamente qual o melhor modelo para extração *features* a partir do texto livre do *corpus*. Porém, como comentado na seção 3.1, pode ser complementado ao modelo os campos "Nota Questão 1" e "Nota Questão 2", que possuem informações relevantes, referentes à satisfação com o atendimento e se o chamado foi resolvido ou não.

---

```

1 print(data_features_tf.shape)
2
3 data_features_tf2=
4     csr_matrix(hstack([coo_matrix(data_features_tf),
5         coo_matrix(n1_scaled)]))
6
7 data_features_tf2=
8     csr_matrix(hstack([coo_matrix(data_features_tf2),
9         coo_matrix(n2_scaled)]))
10
11 print(n1_scaled.shape)
12 print(n2_scaled.shape)
13 print(data_features_tf2.shape)
14
15 #Output
16 (99825, 27385)
17 (99825, 1)
18 (99825, 1)
19 (99825, 27387)

```

---

Código 3.6: Utilização de *features* adicionais

No código 3.6, com a implementação de como as *features* adicionais foram agregadas ao conjunto de *features*, na variável `data_features_tf`, resultante do código 3.5. Ao final do processamento desse bloco de código, temos a variável `data_features_tf2`, com as *features* de "Nota Questão 1" e "Nota Questão 2" agregadas.

### 3.3.4 Modelos de aprendizado de máquina para análise de sentimento

Serão cinco algoritmos de *machine learning*: *Naive Bayes*, *Stochastic Gradient Descent (SGD)*, *XGBoost* e *Support Vector Machine (SVM)*. No SVM utilizaremos duas variações: kernel linear e kernel não linear [46]. Em relação aos modelos para seleção de *features*, serão dois: TF normalizado e TF normalizado com agregação das *features* "Nota Questão 1" e "Nota Questão 2".

Após a escolha do modelo para seleção de *features* e dos algoritmos de

*machine learning* a serem utilizados, podemos iniciar o processamento dos dados. Será iniciada a apresentação pelo SVM linear, conforme código 3.7. A primeira etapa do algoritmo é a parametrização do estimador classificador "clf". Após a parametrização, executamos o processamento do algoritmo utilizando o conjunto de *features* construído no pré-processamento.

---

```
1 clf = svm.SVC(kernel='linear', C=1,cache_size=1000)
2
3 #Modelo TF Normalizado e SVM Linear
4 n_samples=data_features_tf.shape[0]
5 cv = cross_validation.ShuffleSplit(
6     n_samples, n_iter=5, test_size=0.2, random_state=23)
7 scores_SVM = cross_validation.cross_val_score(
8     clf, data_features_tf, df_scikit['label'].values , cv=cv)
9
10 #Modelo TF Normalizado + Nota1 + Nota2 e SVM Linear
11 n_samples=data_features_tf2.shape[0]
12 cv = cross_validation.ShuffleSplit(
13     n_samples, n_iter=5, test_size=0.2, random_state=23)
14 scores_SVM2 = cross_validation.cross_val_score(
15     clf, data_features_tf2, df_scikit['label'].values , cv=cv)
```

---

Código 3.7: SVM Linear

Deve ser observado que, todos os modelos utilizam a funcionalidade de *cross validation*, que consiste em segregar aleatoriamente os dados utilizados na aprendizagem dos dados utilizados para a classificação. Essa funcionalidade visa tratar o problema metodológico de utilizar os mesmos dados para treinamento e teste, que tem como consequência o *overfitting*. No caso desse trabalho, a divisão foi feita considerando 80% para treinamento e 20 % para teste e realizando cinco iterações. O resultado consolidado leva em consideração a média das cinco iterações. Essa abordagem é chamada de *k-fold cross validation*, onde para esse caso  $k=5$ .

Será apresentado agora o pelo o SVM *Radial Basis Function* (RBF) ou SVM não linear, conforme código 3.8.

---

```

1  clf3 = svm.SVC(kernel='rbf', C=1,cache_size=1000)
2
3  #Modelo TF Normalizado e SVM não Linear
4  n_samples=data_features_tf.shape[0]
5  cv = cross_validation.ShuffleSplit(
6      n_samples, n_iter=5, test_size=0.2, random_state=23)
7  scores_SVM3 = cross_validation.cross_val_score(
8      clf3, data_features_tf, df_scikit['label'].values , cv=cv)
9
10 #Modelo TF Normalizado + Nota1 + Nota2 e SVM não Linear
11 n_samples=data_features_tf2.shape[0]
12 cv = cross_validation.ShuffleSplit(
13     n_samples, n_iter=5, test_size=0.2, random_state=23)
14 scores_SVM4 = cross_validation.cross_val_score(
15     clf3, data_features_tf2, df_scikit['label'].values , cv=cv)

```

---

Código 3.8: SVM não linear

O mesmo processo é feito para os demais algoritmos: *Naive Bayes*, *SGD* e *XGBoost*., conforme códigos 3.9, 3.10 e 3.11, respectivamente, com pequenas variações, em geral, devido a parametrização distinta de cada um dos estimadores. O conceito porém, não é alterado, ou seja, primeiro temos a etapa de "fit", onde o estimador aprende sobre os dados, através do subconjunto de treinamento. Em seguida, na próxima etapa, o estimador tenta classificar o subconjunto de teste utilizando o aprendizado obtido na etapa imediatamente anterior.

---

```

1  clf5 = MultinomialNB()
2
3  #Modelo TF Normalizado e Naive Bayes
4  n_samples=data_features_tf.shape[0]
5  cv = cross_validation.ShuffleSplit(
6      n_samples, n_iter=5, test_size=0.2, random_state=23)
7  scores_SVM3 = cross_validation.cross_val_score(
8      clf5, data_features_tf, df_scikit['label'].values , cv=cv)
9
10 #Modelo TF Normalizado + Nota1 + Nota2 e Naive Bayes
11 n_samples=data_features_tf2.shape[0]
12 cv = cross_validation.ShuffleSplit(
13     n_samples, n_iter=5, test_size=0.2, random_state=23)
14 scores_SVM4 = cross_validation.cross_val_score(
15     clf5, data_features_tf2, df_scikit['label'].values , cv=cv)

```

---

Código 3.9: *Naive Bayes*

---

```

1  clf7 = SGDClassifier(loss="hinge", penalty="l2",n_iter=50)
2
3  #Modelo TF Normalizado e SGD
4  n_samples=data_features_tf.shape[0]
5  cv = cross_validation.ShuffleSplit(
6      n_samples, n_iter=5, test_size=0.2, random_state=23)
7  scores_SVM3 = cross_validation.cross_val_score(
8      clf7, data_features_tf, df_scikit['label'].values , cv=cv)
9
10 #Modelo TF Normalizado + Nota1 + Nota2 e SGD
11 n_samples=data_features_tf2.shape[0]
12 cv = cross_validation.ShuffleSplit(
13     n_samples, n_iter=5, test_size=0.2, random_state=23)
14 scores_SVM4 = cross_validation.cross_val_score(
15     clf7, data_features_tf2, df_scikit['label'].values , cv=cv)

```

---

Código 3.10: SGD

---

```

1  clf9=xgb.XGBClassifier(
2      max_depth=5, learning_rate=0.05, n_estimators=100)
3
4  #Modelo TF Normalizado e XGBoost
5  n_samples=data_features_tf.shape[0]
6  cv = cross_validation.ShuffleSplit(
7      n_samples, n_iter=5, test_size=0.2, random_state=23)
8  scores_SVM3 = cross_validation.cross_val_score(
9      clf9, data_features_tf, df_scikit['label'].values , cv=cv)
10
11 #Modelo TF Normalizado + Nota1 + Nota2 e XGBoost
12 n_samples=data_features_tf2.shape[0]
13 cv = cross_validation.ShuffleSplit(
14     n_samples, n_iter=5, test_size=0.2, random_state=23)
15 scores_SVM4 = cross_validation.cross_val_score(
16     clf9, data_features_tf2, df_scikit['label'].values , cv=cv)

```

---

Código 3.11: *XGBoost*

Na seção 4, referente a apresentação dos resultados, serão apresentados as métricas e os valores obtidos para cada um dos modelos treinados e testados. Antes disso, será discutido a aplicabilidade da técnica de *oversampling* no problema em questão e um modelo binário alternativo de classes, utilizando apenas duas classes para classificação.

### 3.3.5 Tratando o desequilíbrio de classes

Para o *corpus* escopo deste trabalho, que trata de pesquisas de satisfação de serviços de tecnologia da informação, foi analisado se será preciso tratar do problema conhecido como *class imbalance*. Isso porque, aparentemente, a grande maioria das pesquisas respondidas pertencem a classe de elogio, enquanto um pequeno número de pesquisas respondidas pertencem a classe de reclamação e neutro.

Avaliando os registros de pesquisa de satisfação disponíveis, podemos verificar a representatividades das classes:

	Classe	Registros	%
(1)	elogio	65.404	65,5%
(2)	neutro	18.756	18,8%
(3)	reclamação	15.665	15,7%
	<b>Total</b>	<b>99.825</b>	<b>100%</b>

Tabela 3.4: Representatividade das classes

Conforme a literatura, configura-se *class imbalance* quando temos uma razão de aproximadamente uma instância com pouca representatividade para cada cem instâncias com grande representatividade [24].

Analisando os quantitativos de registros na tabela 3.4, pode ser concluído que, apesar da classe "elogio" ter uma representatividade maior do que as demais, não é possível configurar precisamente uma situação de *class imbalance*. Isso porque a proporção no problema da pesquisa de satisfação é de aproximadamente uma instância de "reclamação" para cada quatro instâncias da classe "elogio".

Dessa forma, não será realizado nenhum tratamento específico para *class imbalance*, visto que o problema de desbalanceamento de classes não foi configurado, conforme a literatura.

### 3.3.6 Modelo binário de classes

Até agora, o problema de análise de sentimento de textos curtos, materializado através do exemplo de pesquisas de satisfação de serviços de TI, foi tratado como um problema *multiclass*. Ou seja, a classificação é feita para uma única classe, sendo que existem mais de duas classes possíveis. Vale lembrar que nas pesquisas de satisfação, os valores válidos eram: "elogio", "neutro" e "reclamação".

Será feito agora o processamento e análise de um modelo alternativo, binário, caso tivéssemos apenas duas classes possíveis, "elogio" e "não-elogio". O pré-processamento do modelo binário, conforme código 3.12, foi ligeiramente distinto do pré-processamento feito anteriormente na seção 3.2

---

```

1  #Função para separar os registros em: elogio e não-elogio
2  #Input : dataframe e campo 'label'
3  #Output: label tratado
4  def setmanifestacao_binario (DF,X):
5      positivo=["elogio","agradecimento"]
6      neutro  =["neutro","sugestão"]
7      negativo=["reclamação","insatisfeito","não-resolvido"]
8      valido=positivo+neutro+negativo
9
10     if (str.lower(X) in negativo):
11         return "não-elogio"
12     elif (str.lower(X) in positivo):
13         return "elogio"
14     elif (str.lower(X) in neutro):
15         return "não-elogio"
16     elif (str.lower(X) not in valido):
17         return "invalido"
18     else:
19         return "erro"

```

---

Código 3.12: Modelo Binário: Pré-Processamento

A regra utilizada para separação dos registros em duas classes, no modelo binário, foi agrupar os registros com polaridade positiva na classe "elogio" e agrupar os registros com polaridade negativa ou neutra na classe "não-elogio". A justificativa para essa divisão, foi principalmente pela representatividade dos registros com polaridade positiva, que são a grande maioria. A segmentação dos registros entre as duas classes pode ser visualizada na figura 3.7



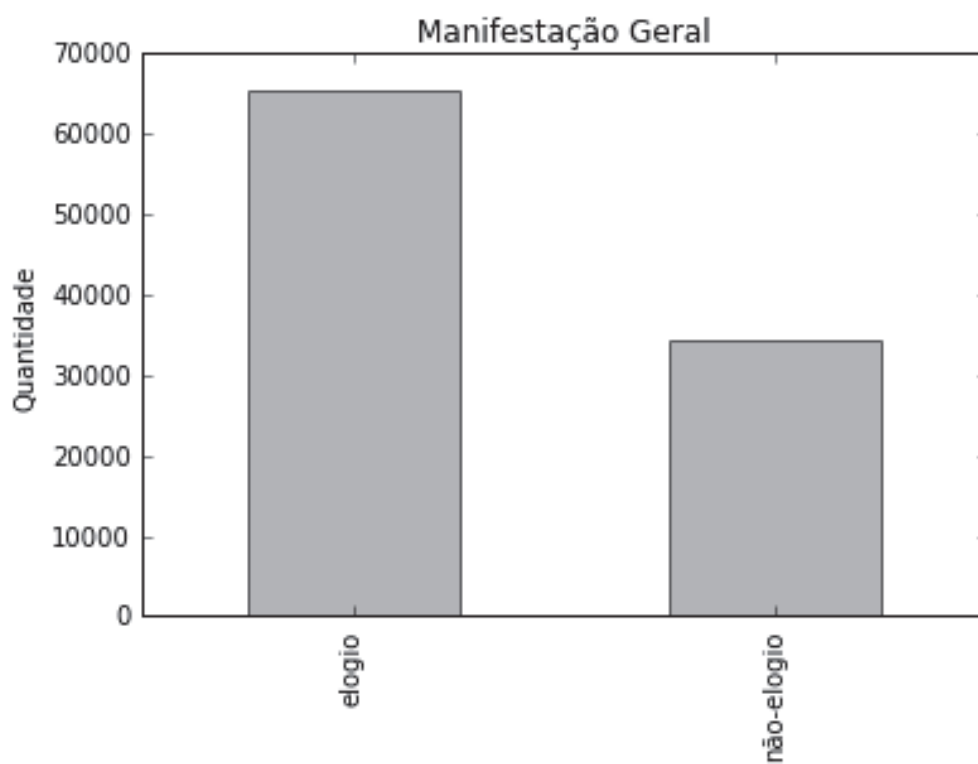


Figura 3.7: Modelo Binário - Manifestação Geral

## 4 Resultados

Nessa seção serão apresentados os resultados do processamento de dados dos modelos de *machine learning*, bem como comentários em relação a esses resultados. Será apresentado também o resultado do modelo binário alternativo de classes, onde, ao invés de três classes: "elogio", "neutro" e "reclamação", temos apenas duas classes: "elogio" e "não-elogio".

### 4.1 Resultados do modelo *multiclass*

Os resultados para o modelo de *multiclass* foram consolidados na tabela 4.1, que foi obtida a partir da média do *K-Fold cross validation*, utilizando cinco *folds* [47] e a métrica *score*.

	Modelo	SVM RBF	Naive Bayes	XGB	SGD	SVM Linear
(1)	TF Norm.	65,4%	81,3%	79,5%	84,1%	<b>85,0%</b>
(2)	TF Norm.+N1+N2	78,1%	81,3%	83,8%	86,1%	<b>86,8%</b>

Tabela 4.1: Resultados dos modelos de análise de sentimento

De uma maneira geral, é possível perceber que os algoritmos de *machine learning* superam um *baseline* de 65,5%, que é o número de registros da classe majoritária "elogio", conforme tabela 3.4 com exceção do Modelo (1) no algoritmo SVM RBF. Vale a pena fazer esse comentário, porque poderia ser alcançado um score próximo de 65,5% apenas classificando todos os registros como "elogio".

Vale lembrar que o modelo *Term Frequency* normalizado (TF Norm.) foi utilizado por apresentar melhor performance do que os demais modelos verificados na seção 3.3.1.

### 4.2 Proposição de modelo de visualização da informação para análise de resultado de um classificador

Durante o desenvolvimento deste trabalho foi identificado que a grande maioria dos livros e artigos que tratam o problema de classificação, ao apresentar os resultados obtidos, o fazem através de tabelas simples.

As tabelas são eficientes para apresentar de forma direta o resultado obtido nas experimentações, porém deixam a desejar no apelo visual e no fornecimento de eventuais detalhes complementares do modelo. Com o objetivo de atender essas questões de visualização e detalhamento dos resultados, foi construído um modelo de visualização.

O modelo construído foi específico para os dados escopo do trabalho, porém pode ser adaptado ou estendido para outros problemas de classificação similares. O modelo é baseado na visualização do tipo *waffle*, onde cada pesquisa de satisfação é tratada como um pequeno quadrado de dimensões mínimas.

Os tons de azul indicam qual o *label* da pesquisa: "elogio", "neutro" e "reclamação". No caso do algoritmo acertar a classificação o tom azul é mantido. Caso a classificação seja incorreta, a pesquisa é marcada em um tom de vermelho. O tom de vermelho mais escuro, indica um erro de classificação para a categoria "reclamação", enquanto que um tom de vermelho mais claro, indica um erro de classificação para a categoria "elogio".

Ao posicionar o mouse sobre uma determinada pesquisa, algumas informações relevantes sobre a mesma são apresentadas na parte inferior do modelo de visualização. Adicionalmente, é apresentado também o resultado da acurácia por classe e total.

Foram utilizados dados do mês de julho de 2016 para construção apresentação do modelo de visualização na figura 4.1. Ou seja, foram utilizados dados distintos dos utilizados para treinamento e teste do algoritmo.

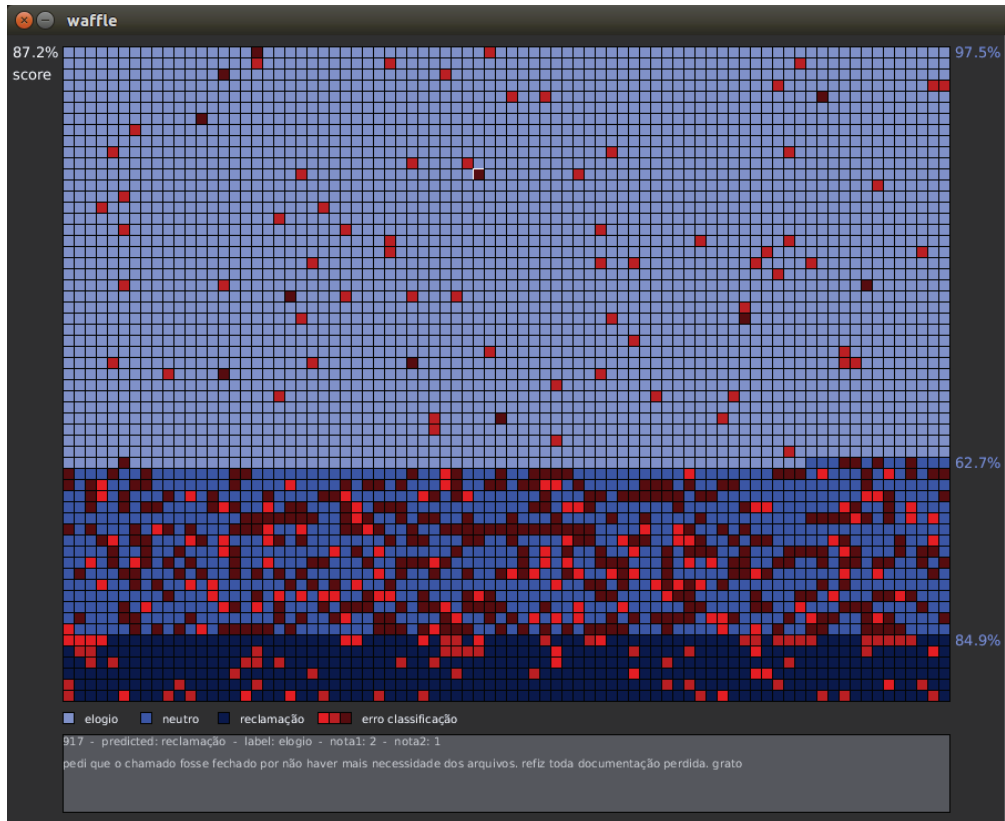


Figura 4.1: modelo de visualização da informação

### 4.3 Técnicas para melhorar a qualidade do modelo

Uma das técnicas para melhorar a qualidade do modelo é o *tuning* de parâmetros utilizados no estimador. Esses parâmetros não são aprendidos durante a etapa de treinamento e sim passados ao método construtor do estimador quando o mesmo é instanciado. Exemplos típicos são o parâmetro *"kernel"* e *"C"* do estimador SVM.

Existe uma funcionalidade no pacote *scikit-learn* chamada de *GridSearch* que realiza a busca exaustiva pelo melhor parâmetro a partir de um estimador. Segue um trecho do código utilizado para utilizar a funcionalidade *GridSearch* no modelo SVM Linear, para verificar o valor do parâmetro *'C'*

que maximiza o *score* do estimador:

---

```
1 param_grid = {"C": [0.5,1,2] }
2
3 # Executando GridSearch
4 grid_search = GridSearchCV(clf2, param_grid=param_grid)
5 grid_search.fit(test_comments2, test_labels2)
```

---

Código 4.1: Utilizando *GridSearch*

Segue o código 4.2 com o detalhamento dos parâmetros utilizados no estimador SVM Linear.

---

```
1 SVM-Linear: {
2   'C': 1,
3   'tol': 0.001,
4   'verbose': False,
5   'kernel': 'linear',
6   'shrinking': True,
7   'class_weight': None,
8   'probability': False,
9   'coef0': 0.0,
10  'random_state': None,
11  'decision_function_shape': None,
12  'max_iter': -1,
13  'cache_size': 1000,
14  'degree': 3,
15  'gamma': 'auto'
16 }
```

---

Código 4.2: Parâmetros utilizados no algoritmo SVM linear

Os resultados da tabela 4.1 demonstram a diferença entre os estimadores SVM RBF e SVM Linear que foram construídos quase da mesma maneira. A única diferença entre eles é referente ao parâmetro *kernel*, linha 5 do código 4.2 em que no primeiro caso, utiliza o *kernel* "rbf", e no segundo caso, o *kernel* "linear". Os demais parâmetros são precisamente os mesmos.

Outra funcionalidade do pacote *scikit-learn* interessante, que permite, eventualmente, melhorar os resultados é a funcionalidade de o "Voting".

Essa funcionalidade permite elencar diversos estimadores e utilizar os seus aprendizados distintos para classificar o registro de acordo com a maioria dos estimadores elencados. Funciona como se fossem um conselho consultivo de classificação. Por exemplo, no código 4.3, utilizando três classificadores: SVM Linear, SGD e *XGBoost* para classificar os registros.

---

```
1 clf_voting = VotingClassifier(  
2     estimators=  
3     [('SVM-Linear', clf2),  
4     ('SGD', clf8),  
5     ('XGB', clf10)],  
6     voting='hard')  
7  
8 clf_voting.fit(train_comments2, train_labels2)  
9 print(clf_voting.score(test_comments2, test_labels2))  
10  
11 #Output  
12 0.86942148760330573
```

---

Código 4.3: *Voting*

O resultado da técnica de voting foi aproximadamente 86,9%, que supera inclusive o melhor resultado obtido até então, utilizando apenas o algoritmo SVM Linear. Vale comentar que o modelo de seleção de *features* utilizado no código 4.3, foi o de TF normalizado como "nota1" e "nota2".

Finalmente, vale ser comentado que foram testadas as técnicas de remoção de *stopwords* e extensão do modelo de apenas *unigrams* para *unigrams* e *bigrams*. Os resultados de ambas as técnicas não alterou significativamente os resultados obtidos anteriormente. Em razão disso, essas técnicas não foram exploradas com mais profundidade neste trabalho.

## 4.4 Resultados do modelo binário de classes

Os resultados para o modelo binário foram consolidados na tabela 4.2, que foi obtido a partir da média do *K-Fold cross validation*, utilizando cinco *folds*

[47] e a métrica *score*.

	Modelo	SVM RBF	XGB	Naive Bayes	SGD	SVM Linear
(1)	TF Norm.	65,4%	90,8%	91,7%	93,2%	<b>94,1%</b>
(2)	TF Norm.+N1+N2	86,2%	91,1%	90,4%	93,5%	<b>94,5%</b>

Tabela 4.2: Resultados dos modelos de análise de sentimento binário

Como pode ser observado, ocorreu uma melhoria significativa na maior parte dos resultados do modelo binário em relação modelo *multiclass*. Novamente o modelo (2) com SVM Linear apresenta o melhor resultado. Pode ser observado na tabela 4.2 que, com exceção do Naive Bayes, o modelo (2), que tem agregado às notas da pesquisa de satisfação, apresenta resultados superiores ao modelo (1).

A performance de um classificador binário pode ser ilustrada com a utilização da curva de ROC (*receiver operating characteristic*). A curva de ROC é criada através da plotagem da taxa de TPR (*true positive rate*) contra a taxa de FPR (*false positive rate*).

A taxa de TPR consiste do somatório de amostras de polaridade positiva que foram classificadas como tal. Essa taxa de TPR também é chamada sensibilidade. No caso da pesquisa de satisfação, podemos associar como o percentual de registros da classe "elogio" que foram corretamente classificados.

A taxa de FPR consiste do somatório de amostras de polaridade negativa que foram classificadas como amostras positivas. Essa taxa de FPR também é chamada taxa de falso alarme. No caso da pesquisa de satisfação, podemos associar como o percentual de registros da classe "não-elogio" que foram classificados como sendo registros da classe "elogio".

Para materializar a ilustração, a curva ROC foi plotada para o classificador SVM Linear utilizando o modelo (2), ou seja, utilizando TF normalizado, concatenado às *features* nota1 e nota2.

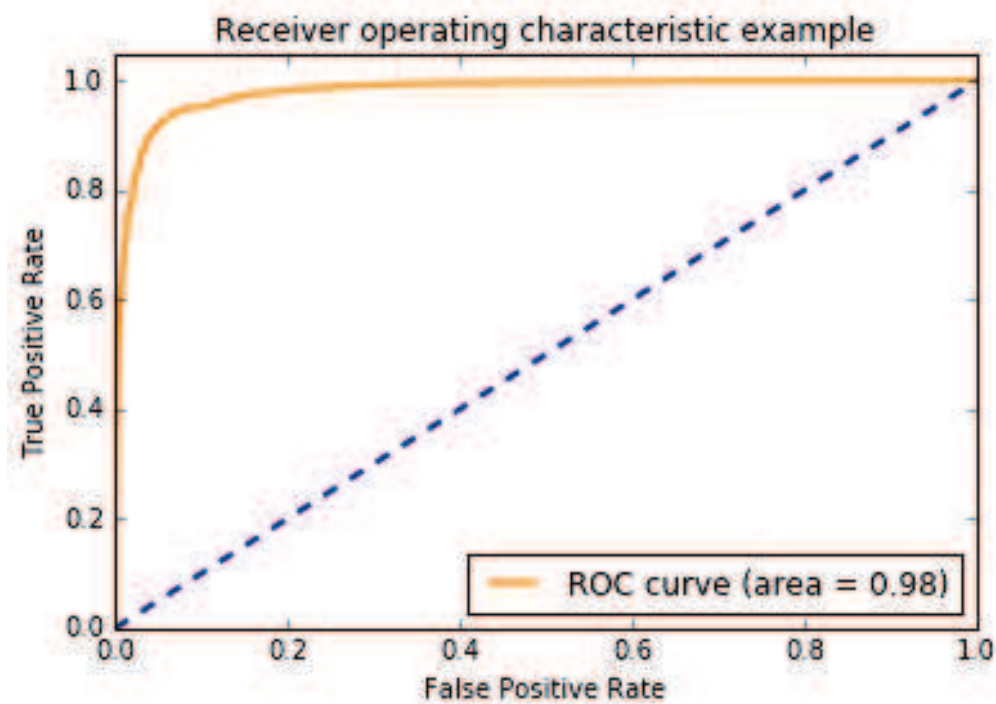


Figura 4.2: Curva ROC para classificador SVM Linear no modelo (2)

A curva ROC mostra o *tradeoff* entre as taxas de TPR e FPR. Quanto mais a curva se aproxima do canto superior esquerdo, mais preciso é o teste. Quanto mais a curva se aproxima da diagonal, destacada em azul pontilhado, menos preciso é o teste.



## 5 Conclusão

A análise de sentimentos em pesquisas de satisfação, com base em dados extraídos de um sistema de gerenciamento de serviços de TI, é possível através de aplicação de técnicas de processamento de linguagem de natural e aprendizagem de máquinas.

De uma maneira geral, algoritmos de *machine learning* apresentaram resultados de até 86,8% de acerto, considerando o modelo de três classes: "elogio", "neutro" e "reclamação". No modelo binário, os resultados foram de até 94,5% de acerto, utilizando as classes: "elogio" e "não-elogio".

Tanto no modelo com três classes, como no modelo binário de classes, em termos de performance, o melhor resultado foi alcançado utilizando o algoritmo SVM com kernel linear. O pior resultado, curiosamente, foi obtido utilizando também o algoritmo SVM, porém com o kernel rbf. Esse resultado inferior, no modelo TF normalizado (1) com três classes, é inclusive menor do que um eventual *baseline* de escolhermos todos os testes como a classe com maior representatividade. Ou seja, se hipoteticamente classificarmos todos os registros como "elogio", obteríamos um score de 65,5%, referente ao percentual de registros dessa classe, superando o SVM com kernel rbf.

Em compensação, o melhor resultado, obtido através do SVM com kernel linear, é aceitável e demonstra que o método proposto, de utilização de algoritmos de *machine learning* para a análise de sentimento, pode ser utilizado de forma a substituir as tarefas manuais de classificação, sem perdas significativas. Isso comprova que: técnicas de *machine learning* superam resultados produzidos por humanos, de forma manual[42].

Vale comentar que muitos dos trabalhos e dos fundamentos do problema de análise de sentimento são similares aos problemas encontrados na classificação de tópicos, onde é necessário automatizar a categorização de texto. Vale comentar que a classificação ou análise de sentimento é historicamente um problema mais desafiador do que uma classificação de tópicos tradicional. Classificadores semelhantes aos utilizados neste trabalho, alcançaram resultados superiores a 90% de acurácia [35][44], quando utilizados para classificação de tópicos, para algumas classes particulares, em modelos com mais de duas classes.

Nos últimos anos, as abordagens de análise de sentimento estão evoluindo do tratamento puramente léxico, para o tratamento semântico de textos, [5]. Esse tratamento semântico, é utilizado em conjunto de técnicas de *machine learning* para obter melhores resultados. Ou seja, para tratar problemas

relacionados a análise de sentimento, o estado da arte atual é a utilização de um modelo híbrido, com abordagem baseada em conhecimento e estatística.

Como próxima etapa para futuros estudos, aplicar modelos alternativos como o *word2vec*, *deep learning* e *Paragraph Vector* no *corpus* de pesquisa de satisfação e comparar os resultados com os obtido neste trabalho em que utilizamos o modelo de *bag-of-words* com o algoritmo de SVM.

Para futuros estudos, seria interessante evoluir também o trabalho de análise de sentimento, aplicando tratamento semântico, com objetivo de identificar, por exemplo: textos com sarcasmo ou ironia, distinção entre textos opinativos e informativos, textos com várias opiniões descritivas sobre um ou vários assuntos, distinção entre textos formais e informais, etc. Será necessário avaliar se algum desses padrões a ser identificado, efetivamente contribui ou não, para melhorar a acurácia do modelo de análise de sentimento para textos curtos.

## 6 Referências

### Referências

- [1] B. Pang and L. Lee, “Opinion Mining and Sentiment Analysis,” *Found. Trends Inf. Retr.*, vol. 2, pp. 1–135, Jan. 2008.
- [2] R. W. Picard, *Affective Computing*. MIT Press, 1997.
- [3] M. Minsky, *The emotion machine: Commonsense thinking, artificial intelligence, and the future of the human mind*. Simon and Schuster, 2007.
- [4] R. J. Mooney and R. Bunescu, “Mining Knowledge from Text Using Information Extraction,” *SIGKDD Explor. Newsl.*, vol. 7, pp. 3–10, June 2005.
- [5] E. Cambria, “Affective Computing and Sentiment Analysis,” *IEEE Computer Society*, 2016.
- [6] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*. Cambridge, MA: The MIT Press, Aug. 2012.
- [7] M. Thelwall, K. Buckley, G. Paltoglou, D. Cai, and A. Kappas, “Sentiment strength detection in short informal text,” *Journal of the American Society for Information Science and Technology*, vol. 61, pp. 2544–2558, Dec. 2010.
- [8] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin, *Learning From Data*. AMLBook, Mar. 2012.
- [9] G. G. Chowdhury, “Natural language processing,” *Annual review of information science and technology*, vol. 37, no. 1, pp. 51–89, 2003.
- [10] S. Bird, “NLTK: The Natural Language Toolkit,” in *Proceedings of the COLING/ACL on Interactive Presentation Sessions*, COLING-ACL ’06, (Stroudsburg, PA, USA), pp. 69–72, Association for Computational Linguistics, 2006.

- [11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, and others, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, no. Oct, pp. 2825–2830, 2011.
- [12] C. C. . L. Cintra, *Nova Gramatica Do Portugues Contemporaneo*. Lexikon, linguistica edition ed., 2009.
- [13] E. Cambria, B. Schuller, Y. Xia, and C. Havasi, “New Avenues in Opinion Mining and Sentiment Analysis,” *IEEE Intelligent Systems*, vol. 28, no. 2, pp. 15–21, 2013.
- [14] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, *Machine Learning: An Artificial Intelligence Approach*. Springer Science & Business Media, Apr. 2013.
- [15] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in *Advances in neural information processing systems*, pp. 3111–3119, 2013.
- [16] Q. V. Le and T. Mikolov, “Distributed Representations of Sentences and Documents,” in *ICML*, vol. 14, pp. 1188–1196, 2014.
- [17] Y. Kim, “Convolutional neural networks for sentence classification,” *arXiv preprint arXiv:1408.5882*, 2014.
- [18] O. Chapelle, B. Schölkopf, A. Zien, and others, “Semi-supervised learning,” 2006.
- [19] A. Pak and P. Paroubek, “Twitter as a Corpus for Sentiment Analysis and Opinion Mining,” in *LREc*, vol. 10, pp. 1320–1326, 2010.
- [20] E. Kouloumpis, T. Wilson, and J. D. Moore, “Twitter sentiment analysis: The good the bad and the omg!,” *Icwsm*, vol. 11, no. 538-541, p. 164, 2011.
- [21] L. Zhang, G. Riddhiman, M. Dekhil, M. Hsu, and L. Bing, “Combining Lexicon-based and Learning-based Methods for Twitter Sentiment Analysis,” *Hewlett-Packard Labs Technical Report HPL-2011-89*, 2011.

- [22] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python*. "O'Reilly Media, Inc.", 2009.
- [23] H. Saif, Y. He, and H. Alani, "Semantic sentiment analysis of twitter," in *The Semantic Web-ISWC 2012*, pp. 508–524, Springer, 2012.
- [24] N. V. Chawla, N. Japkowicz, and A. Kotcz, "Editorial: special issue on learning from imbalanced data sets," *ACM Sigkdd Explorations Newsletter*, vol. 6, no. 1, pp. 1–6, 2004.
- [25] R. C. Prati, G. E. Batista, and M. C. Monard, "Class imbalances versus class overlapping: an analysis of a learning system behavior," in *MICAI 2004: Advances in Artificial Intelligence*, pp. 312–321, Springer, 2004.
- [26] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, pp. 321–357, 2002.
- [27] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, "A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data," *SIGKDD Explor. Newsl.*, vol. 6, pp. 20–29, June 2004.
- [28] S. Dasgupta, C. Papadimitriou, and U. Vazirani, "Algorithms," 2008.
- [29] I. Guyon and A. Elisseeff, "An Introduction to Variable and Feature Selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, Mar. 2003.
- [30] M. E. Maron, "Automatic indexing: an experimental inquiry," *Journal of the ACM (JACM)*, vol. 8, no. 3, pp. 404–417, 1961.
- [31] D. D. Lewis, "Naive (Bayes) at forty: The independence assumption in information retrieval," in *European conference on machine learning*, pp. 4–15, Springer, 1998.
- [32] S. Ross, *Probabilidade: um curso moderno com aplicações*. Bookman Editora, 2009.
- [33] M. A. Hearst, S. T. Dumais, E. Osman, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their Applications*, vol. 13, no. 4, pp. 18–28, 1998.

- [34] J. I. Serrano, M. Tomeckova, and J. Zvarova, “Machine learning methods for knowledge discovery in medical data on Atherosclerosis,” *EJBI*, vol. 1, pp. 6–33, 2006.
- [35] T. Joachims, “Text categorization with support vector machines: Learning with many relevant features,” in *European conference on machine learning*, pp. 137–142, Springer, 1998.
- [36] L. Bottou, “Large-scale machine learning with stochastic gradient descent,” in *Proceedings of COMPSTAT’2010*, pp. 177–186, Springer, 2010.
- [37] U. Dogan, T. Glasmachers, and C. Igel, “A unified view on multi-class support vector classification,” *Journal of Machine Learning Research*, vol. 17, no. 45, pp. 1–32, 2016.
- [38] J. H. Friedman, “Greedy function approximation: a gradient boosting machine,” *Annals of statistics*, pp. 1189–1232, 2001.
- [39] Great Britain Cabinet Office, *ITIL Service Operation*. TSO, 2011.
- [40] A. Abbasi, H. Chen, and A. Salem, “Sentiment Analysis in Multiple Languages: Feature Selection for Opinion Classification in Web Forums,” *ACM Trans. Inf. Syst.*, vol. 26, pp. 12:1–12:34, June 2008.
- [41] P. C. Vinh, V. Alagar, E. Vassev, and A. Khare, *Context-Aware Systems and Applications: Second International Conference, ICCASA 2013, Phu Quoc Island, Vietnam, November 25-26, 2013, Revised Selected Papers*. Springer, Apr. 2014. Google-Books-ID: G3y5BQAAQBAJ.
- [42] B. Pang, L. Lee, and S. Vaithyanathan, “Thumbs Up?: Sentiment Classification Using Machine Learning Techniques,” in *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP ’02, (Stroudsburg, PA, USA), pp. 79–86, Association for Computational Linguistics, 2002.
- [43] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval: The Concepts and Technology Behind Search*. New York: Addison-Wesley Professional, edição: 2 ed., 2011.
- [44] A. McCallum, K. Nigam, and others, “A comparison of event models for naive bayes text classification,” in *AAAI-98 workshop on learning for text categorization*, vol. 752, pp. 41–48, Citeseer, 1998.

- [45] T. Pedersen, “A decision tree of bigrams is an accurate predictor of word sense,” in *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, pp. 1–8, Association for Computational Linguistics, 2001.
- [46] A. J. Smola and B. Schölkopf, “A tutorial on support vector regression,” *Statistics and computing*, vol. 14, no. 3, pp. 199–222, 2004.
- [47] R. Kohavi and others, “A study of cross-validation and bootstrap for accuracy estimation and model selection,” in *Ijcai*, vol. 14, pp. 1137–1145, 1995.