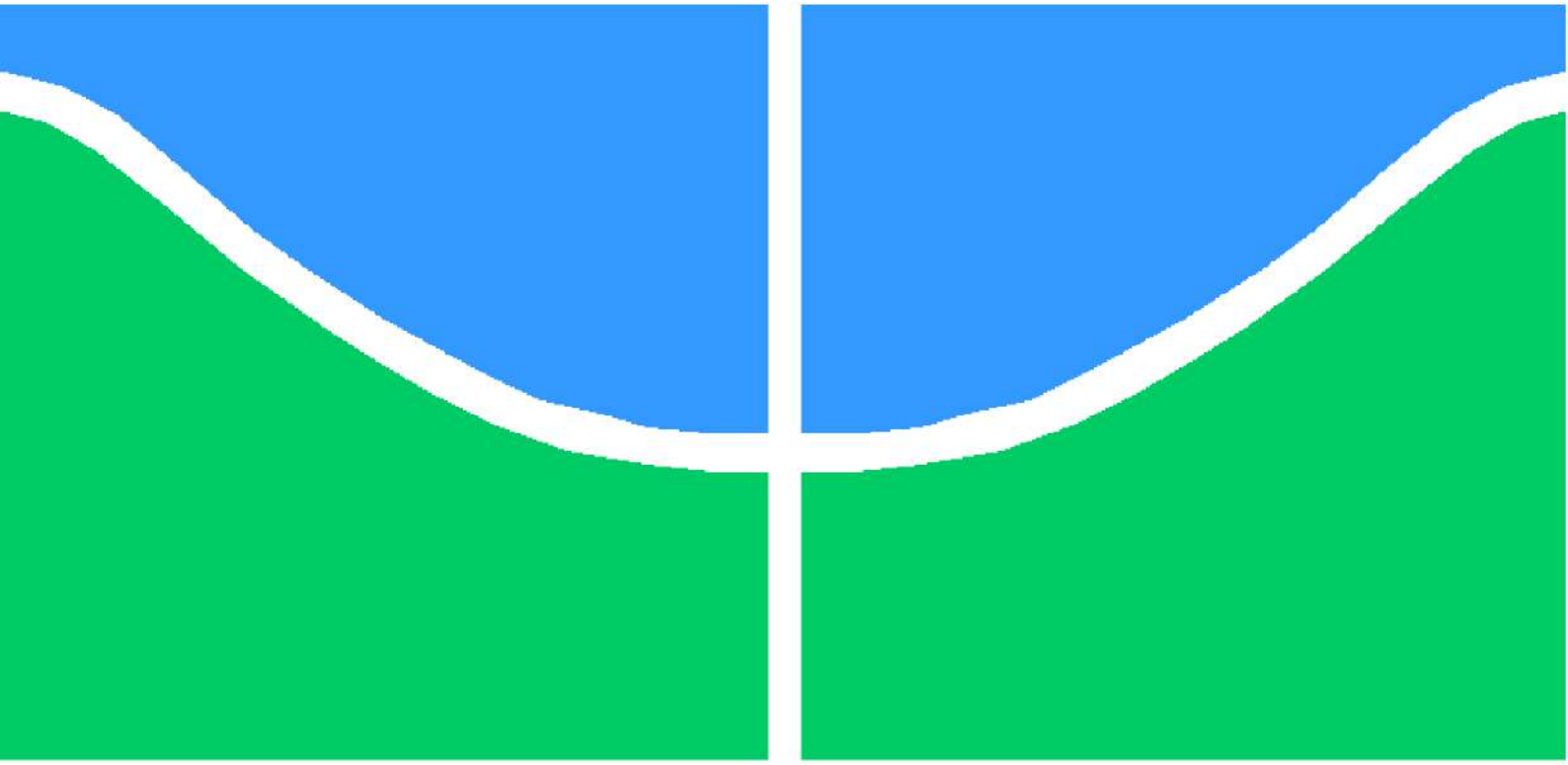

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Engenharia de Software

Aprendizado de Máquina e Processamento de Linguagem Natural aplicados à identificação de discurso de ódio

Autor: Lucas Henrique Araújo Malta, Marco Antonio Rodrigues Loureiro Kuroiva

Orientador: Prof. Dr. Fábio Macêdo Mendes

Brasília, DF

2019

Lucas Henrique Araújo Malta, Marco Antonio Rodrigues Loureiro Kuroiva

Aprendizado de Máquina e Processamento de Linguagem Natural aplicados à identificação de discurso de ódio

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Prof. Dr. Fábio Macêdo Mendes

Brasília, DF

2019

Lucas Henrique Araújo Malta, Marco Antonio Rodrigues Loureiro Kuroiva
Aprendizado de Máquina e Processamento de Linguagem Natural aplicados à identificação de discurso de ódio/ Lucas Henrique Araújo Malta, Marco Antonio Rodrigues Loureiro Kuroiva. – Brasília, DF, 2019-
54 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Fábio Macêdo Mendes

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA , 2019.

1. Aprendizado de máquina. 2. Inteligência artificial. I. Prof. Dr. Fábio Macêdo Mendes. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Aprendizado de Máquina e Processamento de Linguagem Natural aplicados à identificação de discurso de ódio

CDU 02:141:005.6

Lucas Henrique Araújo Malta, Marco Antonio Rodrigues Loureiro Kuroiva

Aprendizado de Máquina e Processamento de Linguagem Natural aplicados à identificação de discurso de ódio

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 18 de julho de 2019:

Prof. Dr. Fábio Macêdo Mendes
Orientador

Prof. Dr. Renato Coral Sampaio
Convidado 1

Ms. Ricardo Augusto Poppi Martins
Convidado 2

Brasília, DF
2019

Agradecimentos

Lucas Malta

- A minha família, meus pais, Jefferson Aguiar Malta e Gláucia Araújo Silva, que me apoiaram nos momentos de alegria e dificuldade, dando o apoio necessário para meu sucesso até o fim deste curso.
- A minha noiva, Ariele Louise Bomfim, que me motiva e se torna meu motivo todos os dias.
- Aos meus colegas de curso, que me trouxeram momentos de alegria, companhia e foram essenciais para o meu progresso.
- Ao meu orientador, Fábio Macêdo Mendes, pelos ensinamentos, correções, paciência e pelo dom de ensinar.

Marco Kuroiva

Primeiramente, gostaria de agradecer à pessoa mais importante para mim, que é a minha mãe, Teresinha Nessi Rodrigues Loureiro, que sempre esteve ao meu lado em todas as situações. Da mesma forma, gostaria de agradecer à toda a minha família, que acompanhou a minha jornada, não apenas ao longo da graduação, mas da minha vida.

Deixo meus agradecimentos aos meus amigos, por terem sido pacientes e me alegrarem até nos momentos mais difíceis. Agradeço também aos meus colegas, que, mais do que ninguém, conhecem bem os obstáculos que enfrentei durante esses anos de graduação.

Por fim, agradeço ao meu orientador, Fábio Macêdo Mendes, por todo suporte prestado durante a execução deste trabalho

Resumo

O Empurrando Juntos é uma plataforma de participação que aborda temas de cunho cívico, minimizando os efeitos da polarização. Uma preocupação natural do projeto é a manutenção de um ambiente livre de comentários com linguagem abusiva, incitação à violência e discurso de ódio. Para tal, a plataforma conta com um sistema de moderação de comentários manual. A fim de reduzir o esforço e a interferência humana nesta tarefa, este projeto propõe o uso de técnicas de Processamento de Linguagem Natural e Aprendizado de Máquina num sistema de moderação de comentários automático. Para isso, a metodologia CRISP-DM foi adotada e uma série de experimentos foram conduzidos a fim de encontrar um modelo adequado para a classificação dos comentários realizados na plataforma.

Palavras-chaves: Aprendizado de Máquina, Processamento de Linguagem Natural, Moderação de Comentários, Plataforma Web, Empurrando Juntos.

Abstract

Empurrando Juntos is a platform for participation that addresses civic issues, minimizing the effects of polarization. A natural concern of the project is the maintenance of an environment free of comments with abusive language, incitement to violence and hate speech. For this, the platform has a manual moderation system of comments. In order to reduce effort and human interference in this task, this project proposes the use of Natural Language Processing and Machine Learning techniques in an automatic comment moderation system. To that end, the CRISP-DM methodology was adopted and a series of experiments were conducted in order to find a suitable model for the classification of the comments made on the platform.

Key-words: Machine Learning, Natural Language Processing, Comment Moderation, Empurrando Juntos.

Lista de ilustrações

Figura 1 – Diagrama sobre tipos de aprendizado de máquina 28
Figura 2 – Diagrama SVM para 2 features 29
Figura 3 – Diagrama SVC. A esquerda, C baixo, a direita, C alto. 29
Figura 4 – Diagrama MLP, 4 features de entrada, 1 camada oculta, 8 neurônios . 31
Figura 5 – Precision, Recall e F1-score do algoritmo Random Forest. 34
Figura 6 – FONTE: scikit-learn.org 39
Figura 7 – Captura de tela no Jupyter Notebook 40
Figura 8 – O ciclo de vida da mineração de dados. FONTE: (IBM, 2015) 41
Figura 9 – Métricas do primeiro ciclo. 44
Figura 10 – Métricas dos modelos com o Bagging. 46
Figura 11 – Comparação dos modelos. 48
Figura 12 – Linha de tendência da precisão x recall dos modelos. 48

Lista de tabelas

Tabela 1 – Exemplo de *Bag-of-Words* 23
Tabela 2 – Exemplo de representação do modelo *n-grams* 25
Tabela 3 – Exemplo 2 de representação do modelo *n-grams* 25
Tabela 4 – Exemplo de votação com probabilidades 32
Tabela 5 – Matriz de confusão para a classe **C1** - Discurso de ódio 34
Tabela 6 – Informações do dataset 39
Tabela 7 – Trechos do dataset OffComBR 40
Tabela 8 – Resultado dos experimentos com os modelos 44
Tabela 9 – Resultado dos experimentos com Bagging 46
Tabela 10 – Resultados do segundo ciclo 47

Lista de abreviaturas e siglas

AM	Aprendizado de máquina.
EJ	Empurrando Juntos, rede social.
IA	Inteligência artificial.
MLP	Multilayer Perceptron, modelo de treinamento.
MNB	Multinomial Naive Bayes, modelo treinamento.
PLN	Processamento de Linguagem Natural.
SVM	Support Vector Machine, modelo de treinamento.

Sumário

1 INTRODUÇÃO 19
1.1 Contextualização 19
1.2 Problematização 19
1.3 Objetivos 20
1.4 Organização dos capítulos 20

2 FUNDAMENTAÇÃO TEÓRICA 23
2.1 Processamento de Linguagem Natural 23
2.1.1 Bag-of-Words 23
2.1.2 N-grams 24
2.1.3 TF-IDF 25
2.1.4 Limpeza do Texto 26
2.2 Aprendizado de máquina 27
2.3 Classificação 27
2.3.1 Support Vector Machines 28
2.3.2 Decision Tree/Random Forest 30
2.3.3 Naive Bayes/Multinomial NB 30
2.3.4 Multilayer Perceptron 31
2.3.5 Comitês de Máquina 31
2.3.5.1 Voting Classifier 32
2.3.5.2 Bagging Classifier 33
2.3.5.3 Custom Classifier 33
2.4 Métricas 33
2.5 Trabalhos Relacionados 35

3 FERRAMENTAS DE APOIO 37
3.1 NLTK 37
3.2 scikit-learn 37
3.3 Jupyter notebook 39
3.4 Conjunto de dados 39

4 METODOLOGIA 41
4.1 Fluxo de atividades 41

5 RESULTADOS E DISCUSSÕES 43
5.1 Resultados dos classificadores 43

5.1.1 Primeiro ciclo do CRISP-DM 44
5.1.2 Segundo ciclo do CRISP-DM 45

6 CONSIDERAÇÕES FINAIS 51

REFERÊNCIAS 53

1 Introdução

1.1 Contextualização

A popularização da internet foi um fenômeno que possibilitou e facilitou a criação e o compartilhamento de conteúdo, além de ter criado uma enorme abertura para a exposição de opiniões. Da mesma maneira, existe o risco da manifestação de discursos ofensivos, incitação à violência, discriminação contra grupos em virtude da etnia, religião, orientação sexual, entre outros. Em redes sociais, é algo bastante recorrente, onde é possível observar o exercício abusivo da liberdade de expressão, com as pessoas assumindo uma postura mais ativa, potencializado pela alta velocidade de propagação e aparente possibilidade de anonimato (STROPPA; ROTHENBURG, 2015).

O projeto Empurrando Juntos foi criado como uma plataforma de participação que objetiva minimizar os efeitos das bolhas de opinião e polarização, através da horizontalidade de comentários e de elementos de *gamificação*. Nesse contexto, há a necessidade de uma camada moderadora, uma vez que opiniões controversas podem gerar atrito entre as partes de uma discussão.

1.2 Problematização

A plataforma Empurrando Juntos busca discutir diferentes assuntos, políticos e sociais, com o objetivo de dar voz à quem precisa e de construir uma comunidade mais crítica e mais aberta a opiniões diferentes. A EJ é considerada uma plataforma de participação com elementos de rede social como a visualização e o voto de comentários, mesmo que esses não possam ser acessados de maneira direta. Por este motivo, pode ser considerada um “local público” (REBS, 2017). Apesar disso, há algumas diferenças de um local público físico, como a replicabilidade, a persistência e a buscabilidade de informações, uma vez que a mesma sempre fica disponível. Além disso, a EJ possui um paradigma alternativo com relação à interação de usuários, pois existem meios que impedem bolhas e opiniões totalitárias, como a ausência de "links"para um comentário específico. Esse paradigma visa dificultar a realização de ataques orquestrados em um determinado comentário específico, já que os comentários, por não apresentarem um link direto, possuem um caráter randômico em sua apresentação. Dada a necessidade da plataforma, o foco na moderação é um dos requisitos para o seu sucesso, e a moderação automatizada é a opção ideal para sistemas com grandes quantidades de entrada de informação e interação, visto que atualmente os comentários criados são submetidos à uma fila, onde o criador da conversa ou um moderador podem aceitar ou rejeitar um comentário, com a opção de

especificar o motivo para a rejeição do mesmo.

O principal canal de informação da plataforma EJ se dá a partir das conversas e dos comentários. A fim de promover um ambiente de discussão mais saudável, eliminando xingamentos, ofensas gratuitas e o discurso de ódio, é realizada uma moderação de comentários. Esta atividade é executada manualmente por membros do projeto, que devem ler os comentários individualmente para poder aprovar a sua exibição na plataforma.

O atual trabalho propõe a automatizar o processo de moderação ao máximo possível, de modo que, nem todos os comentários precisem ser submetidos à moderação manual, sendo estes, automaticamente classificados como discurso comum ou discurso de ódio. Como ferramenta de apoio da plataforma, o moderador automático não promove a substituição completa da moderação manual, apenas a diminuição da carga de trabalho destinada ao processo.

1.3 Objetivos

Este trabalho possui como objetivo geral facilitar a identificação de comentários com o linguajar inapropriado e conteúdo incompreensível para aplicações web, mais especificamente no projeto "Empurrando Juntos", utilizando de técnicas e ferramentas de processamento de linguagem natural e aprendizado de máquina.

Visando atingir o objetivo geral, alguns objetivos específicos foram traçados:

- Selecionar conjunto de dados de treinamento.
- Treinar e avaliar modelos de classificação.
- Construir, treinar e avaliar um classificador baseado em comitê de máquina.

1.4 Organização dos capítulos

O atual trabalho está organizado em capítulos de acordo com a lista abaixo:

- **Introdução:** atual capítulo, onde são apresentadas a contextualização, a problematização, os trabalhos relacionados e os objetivos.
- **Fundamentação Teórica:** capítulo dedicado a fornecer o embasamento teórico necessário para a realização e o entendimento deste trabalho.
- **Ferramentas de apoio:** apresenta as ferramentas que deram suporte necessário para a execução do trabalho, tanto na parte experimental quanto em seu desenvolvimento.

- **Metodologia:** detalha a abordagem e os procedimentos seguidos na execução deste trabalho.
- **Resultados e Discussões:** apresentação e discussão dos resultados obtidos através da coleta das métricas dos experimentos realizados.
- **Considerações Finais:** capítulo dedicado à uma análise interpretativa do trabalho como um todo.

2 Fundamentação teórica

Este capítulo destina-se a apresentar a fundamentação teórica realizada por meio de revisões da literatura. Seu objetivo é oferecer um catálogo de conceitos que servirá como base para o entendimento do atual trabalho.

2.1 Processamento de Linguagem Natural

O Processamento de Linguagem Natural é uma ciência que abranje um conjunto de técnicas e métodos que facilitam a análise e textual por um computador (ARANHA, 2007). No atual projeto, técnicas de PLN são aplicadas para extrair as *features* que, por sua vez, serão utilizadas para a realização das classificações. No contexto do aprendizado de máquina, as *features* são as variáveis intrinsecamente presentes nos dados, sendo elas essenciais na identificação de padrões pelo algoritmo.

As técnicas citadas são utilizadas na etapa de extração de atributos, comumente referida como *Feature Extraction*. É a partir dela que se define quais aspectos do texto servirão de insumo para o classificador. Há diversas formas de se abordar a extração de características de um texto, sendo as principais: *Bag-of-Words*, *n-grams*, *Word2Vec* e TF-IDF.

2.1.1 Bag-of-Words

O BoW (*Bag-of-Words*) é um modelo de extração de características de texto simples e flexível. Ele se baseia no número de ocorrências de palavras de uma frase. Para isso, cada frase é representada por um vetor com n elementos, onde n é o número de palavras do vocabulário considerado. Cada posição representa uma palavra e o elemento é o número de ocorrências daquela palavra na frase. Observe os exemplos a seguir:

(1) u = "A mochila está leve, então quero que a leve"

(2) v = "Levei um lanche leve na mochila"

Imagine um vetor para cada uma dessas frases. No BoW, eles seriam semelhantes a estes:

Tabela 1 – Exemplo de *Bag-of-Words*

	a	**mochila**	**está**	**leve**	**então**	**quero**	**que**	**levei**	**um**	**lanche**	**na**
$\vec{u}$	2	1	1	2	1	1	1	0	0	0	0
$\vec{v}$	0	1	0	1	0	0	0	1	1	1	1

Como é possível observar na Tabela 1, a palavra "leve"foi utilizada duas vezes na frase v, sendo, na primeira, um adjetivo e, na segunda, um verbo. São palavras homônimas, que possuem a mesma grafia, mas significados distintos. Este modelo, porém, desconsidera o papel que as palavras desempenham na frase e a sua estrutura.

Com apenas duas pequenas frases, o vocabulário é bastante limitado, apenas 11 palavras, porém, o que é mais comum em situações reais são vocabulários muito maiores, o que pode resultar em vetores igualmente maiores e com quase todos os seus elementos iguais a 0. Na linguagem *Python*, utilizada neste projeto, esses vetores são representados por matrizes esparsas. Observe a seguir como a classe *CountVectorizer*, da biblioteca scikit, trabalha com essas matrizes.

```
>>> from sklearn.feature_extraction.text import CountVectorizer
>>> corpus = [
        'A mochila está leve, então quero a leve primeiro',
        'Levei um lanche leve na mochila',
    ]
>>> vectorizer = CountVectorizer()
>>> x = vectorizer.fit_transform(corpus)
>>> print(vectorizer.get_feature_names())

['então', 'está', 'lanche', 'leve', 'levei', 'mochila', 'na', 'primeiro', '
    quero', 'um']

>>> print(x.toarray())

[[1 1 0 2 0 1 0 1 1 0]
 [0 0 1 1 1 1 1 0 0 1]]
```

Listing 2.1 – Matriz esparsa da classe csr_matrix

Note no Listing 2.1, que o *CountVectorizer* aplica o modelo de *Bag-of-Words* na construção de um objeto do tipo *csr_matrix*, mas não leva em consideração palavras com apenas um caractere.

2.1.2 N-grams

O modelo *n-grams*, assim como o *Bag-of-Words*, baseia-se em uma contagem de ocorrências. A diferença está na *vetorização*, que se dá por um processo diferente. Ao invés de realizar a contagem por palavras individuais, o *n-grams* propõe uma contagem por uma sequência contínua de n de palavras. Desta forma, este modelo se torna mais adequado para capturar nuances do texto (LUNDBORG, 2017), onde a ordem das palavras é

relevante e não apenas a presença delas. Utilizar este modelo também favorece na identificação de palavras compostas como "primeiro-ministro", "Rio de Janeiro", "Aprendizado de Máquina".

(1) u = “Eu moro no Distrito Federal”

(2) v = “O Distrito Federal fica no Centro-Oeste”

Tabela 2 – Exemplo de representação do modelo *n-grams*

	eu moro	**moro no**	**no distrito**	**distrito federal**	**o distrito**	**federal fica**	**fica no**	**no centro**	**centro oeste**
u	1	1	1	1	0	0	0	0	0
v	0	0	0	1	1	1	1	1	1

Observe, na Tabela 2, como os vetores ficariam para $n = 2$ e note que este modelo é capaz de identificar a presença do *bigrama*[1] "distrito federal"em ambas as frases. Outra situação que favorece o uso de *n-grams* é quando, por exemplo, há a presença das palavras "distrito"e "federal", mas elas estão separadas e com outro sentido, como na frase z abaixo. O modelo BoW não seria capaz de realizar essa distinção e identificaria uma falsa semelhança entre as frases u e z pela presença de termos que, individualmente são iguais, mas que dentro de seus contextos, possuem outros sentidos.

(3) z = “Neste distrito, conheci um policial federal”

Tabela 3 – Exemplo 2 de representação do modelo *n-grams*

	eu moro	**moro no**	**no distrito**	**distrito federal**	**neste distrito**	**distrito conheci**	**conheci um**	**um policial**	**policial federal**
u	1	1	1	1	0	0	0	0	0
z	0	0	0	0	1	1	1	1	1

2.1.3 TF-IDF

Em algumas situações, apenas avaliar a frequência de termos pode não ser o ideal. A forma com que a língua é estruturada favorece a dominância de uma série de termos dos quais não carregam em si conteúdo significativo para a realização de classificações. Termos frequentes não são bons indicadores de contexto por aparecerem em muitas frases independentes de suas classes.

[1] *n-grams* com $n = 2$

A fim de lidar com o grau desbalanceado da relevância desses termos, o modelo TF-IDF[2] propõe um fator de ponderação (*idf*) para que aqueles que mais aparecem nos documentos tenham uma representatividade menor (MATSUBARA; MARTINS; MONARD, 2003).

O cálculo do TF-IDF, definido pela Equação 2.1, descreve a frequência do termo t_j em dado documento d_i multiplicado pelo fator de ponderação, que varia entre 0 e o $\log N$, onde N é o número de documentos do conjunto de dados e $d(t_j)$ é o número de documentos onde há a ocorrência do termo pelo menos uma vez.

$$tfidf(t_j, d_i) = freq(t_j, d_i) \times \log \frac{N}{d(t_j)} \tag{2.1}$$

Ao aplicar a fórmula, o valor do fator de ponderação é igual a 0 quando o termo aparece em todos os documentos, mas quando aparece em apenas um, ele é $\log N$. Note também que, o valor do $tfidf$ favorece a frequência do termo t_j no documento d_i e desfavorece a sua frequência na coleção $d(t_j)$. Em outras palavras, as particularidades de cada documento são acentuadas.

É possível também abordar representações alternativas deste modelo. Em uma proposta denominada tf_{linear} (MATSUBARA; MARTINS; MONARD, 2003), basicamente, o fator de ponderação logarítmico é substituído por um fator linear que varia entre 0 e 1. Observe na Equação 2.2

$$tf_{linear}(t_j, d_i) = freq(t_j, d_i) \times \left(1 - \frac{d(t_j)}{N}\right) \tag{2.2}$$

No atual projeto, o TF-IDF foi utilizado em conjunto com o *n-grams*, ou seja, os termos podem ser compostos por mais de uma palavra. Assim como os hiperparâmetros[3] dos modelos de classificação, o valor de n é definido durante a fase experimental, podendo, inclusive assumir vários valores dentro de um intervalo.

2.1.4 Limpeza do Texto

Parte do pré-processamento do texto consiste na eliminação de informações desnecessárias para a classificação. Para tal, foram aplicadas algumas técnicas como remoção de acentuação, conversão de caracteres maiúsculos em minúsculos, entre outras. Após a normalização, é comum realizar as seguintes operações:

- ***Stemização***: Do inglês, *stemming*, refere-se ao processo de reduzir as palavras à uma forma primitiva, como um radical. Este processo visa garantir que variações

[2] *Term Frequency - Inverse Document Frequency*

[3] Hiperparâmetros são valores definidos na construção de um modelo de AM. Esses valores influenciam diretamente no processo de aprendizagem e em seu comportamento.

de uma mesma palavra sejam interpretadas pelo computador como uma palavra só. Dessa forma, o conjunto de palavras do vocabulário é reduzido, o que resulta numa economia de recursos. Outra vantagem é a generalização de algumas variações das palavras, como em gênero e número, identificando que elas carregam em si o mesmo sentido. Os termos "*historiador*" e "*historiadoras*" seriam reduzidas a um mesmo radical "*histori*", por exemplo.

- **Remoção das *stop words***: As *stop words* (palavras vazias) são palavras que agregam pouco ou nenhum valor semântico que deva ser levado em consideração pelo classificador. Geralmente, são as palavras mais comuns da língua, incluindo artigos, preposições, verbos de ligação, entre outras. Não existe um conjunto bem definido de quais palavras devem ser classificadas como palavras vazias. Naturalmente, esse conjunto depende do idioma em questão e em alguns casos, termos recorrentes do contexto, como jargões e gírias, também podem ser inclusos. Para o atual projeto, foi utilizado o conjunto de *stopwords* em português do **NLTK**.

2.2 Aprendizado de máquina

Aprendizado de máquina (AM) pode ser definido como uma forma de Inteligência Artificial (IA) que permite que um sistema aprenda através de dados ao invés de uma programação explícita (HURWITZ; KIRSCH, 2000). No início dos anos 90, a abordagem do AM para a classificação textual começou a ganhar popularidade e, eventualmente, tornou-se a técnica dominante na academia (SEBASTIANI, 2002).

Os modelos de Aprendizado de Máquina costumam ser divididos em 2 tipos: Supervisionado, do qual um conjunto de observações previamente rotuladas serve de insumo, o modelo tem como objetivo definir rótulos para novas observações, e o Não-Supervisionado, onde o modelo busca encontrar padrões ou estruturas intrínsecas nos dados de treino (MATH WORKS, 2016). Observe a Figura 1.

A escolha de qual tipo de AM deve ser utilizado depende da natureza do problema que está sendo abordado e dos dados disponíveis. No atual trabalho, serão utilizados modelos de classificação, do tipo de Aprendizado Supervisionado. Estes modelos estão detalhados a seguir.

2.3 Classificação

A classificação textual é a tarefa de atribuir um valor *booleano* para cada par de observação e classe (d_i, c_i). O valor é dado como **verdadeiro** se a observação d_i for atribuída à classe c_i, caso contrário, **falso**. Existem diferentes tipos de classificações, que variam de acordo com a situação. Por exemplo, quando uma observação pode ser

Figura 1 – Diagrama sobre tipos de aprendizado de máquina

atribuída à apenas uma classe, ela é apelidada de *single-label*, caso contrário, *multi-label*. Um tipo caso especial de *single-label* é a classificação binária, quando a observação deve, obrigatoriamente, ser atribuída à uma classe ou à sua classe complementar (SEBASTIANI, 2002).

O atual trabalho contempla uma classificação binária, sendo as classes **C0**, que representa um **discurso normal**, e **C1**, representando **discurso de ódio ou linguagem inapropriada**.

Independente do modelo de AM, é necessário utilizar um conjunto de dados para que as classificações sejam realizadas. Os dados de treino são traduzidos em propriedades qualitativas ou quantitativas, as *features*. Múltiplas *features* são combinadas em um vetor e utilizadas como informação de entrada para o algoritmo com a sua classe correspondente (LUNDBORG, 2017). Neste projeto, diversos modelos de classificação são utilizados. Nas subseções deste capítulo, temos suas descrições.

2.3.1 Support Vector Machines

O termo Support Vector Machines engloba um conjunto de modelos que buscam classificar os dados em 2 classes diferentes. Para tal, as features são analisadas e inseridas em um plano com N-1 features de dimensão, de modo que se forme um hiperplano que divide as 2 classes na maior margem possível entre elas. Esse hiperplano é formada com o auxílio dos dois pontos mais próximos de cada classe. Através delas, dois hiperplanos de suporte são traçadas, e a divisão principal é formada. Daí vem o nome “support vector”.

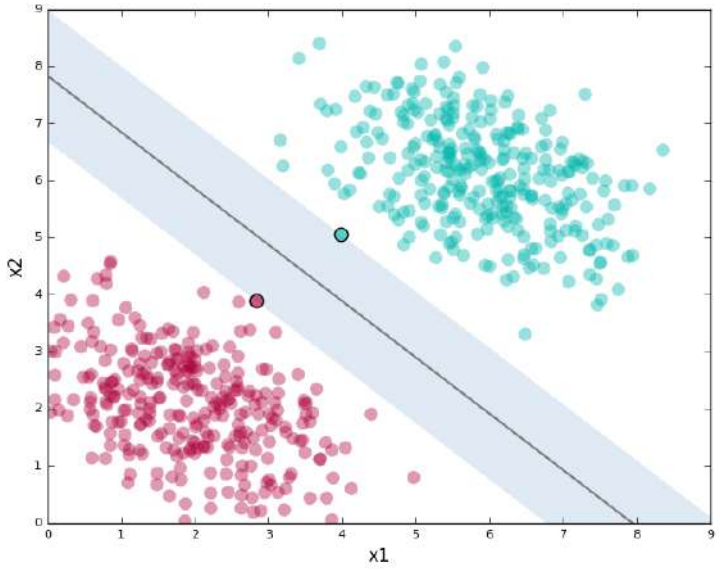

Figura 2 – Diagrama SVM para 2 features

O outro modelo baseado nesse princípio, é o Vetor de Suporte-C, ou SVC, implementado pela libsvm (CHANG; LIN, 2018). Diferente de alguns modelos de vetores de suporte que, para realizar uma atualização de parâmetro necessita computar todas as features, o SVC faz uma iteração mais leve, de modo que apenas um ou um grupo de features é verificado a cada atualização. Além disso, existe o parâmetro C, que quando maior seu valor, menores as margens do hiperplano de divisão serão, de modo que todos os pontos sejam classificados corretamente. Caso o valor seja pequeno, o hiperplano de divisão busca por margens maiores, mesmo que algumas amostras sejam classificadas incorretamente.

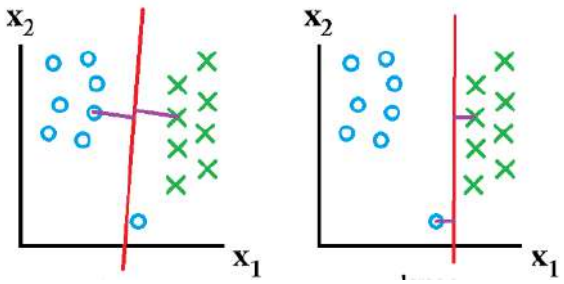

Figura 3 – Diagrama SVC. A esquerda, C baixo, a direita, C alto.

Neste trabalho, o SVC foi utilizado em conjunto do TF-IDF, de modo que se defina várias features para o funcionamento correto do algoritmo. Contudo, o melhor valor para a variável C foi de 4.0, um valor considerado alto. Por causa desse valor, é possível presumir que o hiperplano que classifica as frases fica bem próxima do limite de ambas, uma vez que uma margem grande, alcançada através de um C pequena, não é desejado para a maior performance.

2.3.2 Decision Tree/Random Forest

Árvores de decisão são um modelo usado tanto para regressão, quanto para classificação (LIBERMAN, 2017). São fáceis de serem interpretadas, justamente por se tratar de um modelo baseado em uma estrutura de dados em que estamos acostumados a lidar. Além disso, o modelo suporta classificação numérica ou categórica, e consegue obter um desempenho por qualidade muito bom com relação a outros modelos de classificação, independente do tamanho do dataset (ASHARI; PARYUDI; TJOA, 2013). Contudo, por se tratar de um algoritmo guloso, alguns problemas podem ser vistos nos resultados, como o sobreajuste, ou "overfitting"[4]. Com o objetivo de resolver isso, criou-se o modelo de Floresta randômica. O *Random Forest* consiste em um combinado de árvores de decisão, onde o resultado de cada uma é unido, com o objetivo de encontrar uma predição mais estável e preciso.

No atual projeto, o algoritmo de *Random Forest* com *n-grams* de até 4 palavras foi o que apresentou melhores resultados, de modo que as árvores possam crescer, dada a quantidade de *features* fornecida. A profundidade das árvores não foi limitada, mas foi definido um total de 100 árvores. Desse modo, mesmo que as árvores alcancem uma alta profundidade, o "*overfitting*" dos resultados não ocorre.

2.3.3 Naive Bayes/Multinomial NB

O modelo de *Naive Bayes* é baseado no teorema de Bayes, e tem como forte premissa de que as *"features"*, ou palavras de cada sentença, são fortemente independentes. O classificador de Naive Bayes analisa as palavras, de acordo com a probabilidade da *feature* ser "positiva" ou "negativa", caso seja utilizado para se analisar o sentimento de uma frase, por exemplo.

O *Multinomial Naive Baye*s é uma extensão do modelo de *Naive Bayes*, mas com a utilização da distribuição multinomial para cada *feature*. Desse modo, o modelo Multinomial NB funciona bem para dados que podem ser contados, se tornando um bom modelo para classificação de tópicos, por exemplo (RUSSELL; NORVIG, 2003).

No atual projeto, tanto o modelo tradicional quanto o multinomial funcionam melhor com o *dataset* isento de *stop words*, pois a alta frequência das mesmas pode atrapalhar a eficiência do algoritmo. O *n-gram* que gerou o melhor resultado foi com apenas uma palavra. Isso pode ser explicado pelo fato de que muitas frases que contém discurso de ódio são caracterizadas pelo uso de palavras específicas, raramente utilizadas.

[4] O *overfitting* define um modelo estático que se adequa muito bem a um conjunto de dados já conhecido, mas que não consegue prever novos resultados caso um conjunto de dados diferente seja apresentado. Esse caso piora mais e mais, proporcionalmente igual à profundidade da árvore.

2.3.4 Multilayer Perceptron

O MLP, ou Perceptron de múltiplas camadas, é um modelo de treinamento baseado em redes neurais. Seu funcionamento mais básico consiste em ao menos 3 camadas: A camada de entrada (onde as *features* serão passadas), a camada oculta (onde os cálculos são realizados), e a camada de saída. As camadas são compostas por neurônios, que realizam o cálculo utilizando de informações passadas pelos neurônios das camadas anteriores. Num MLP, podem haver várias camadas ocultas, dependendo do tipo de problema a ser resolvido. Cada camada oculta busca transformar o valor da camada anterior, utilizando da soma linear dos pesos e de uma função de ativação não linear, como a função tangente hiperbólica. Ao chegar na última camada oculta, a camada de saída recebe seu valor e o transforma em valores de saída.

No atual trabalho, apenas uma camada oculta foi utilizada, contendo 100 neurônios. Este valor busca cobrir casos em que haja muitas *features* (frases muito grandes podem resultar em altos valores de *features*), de modo que todas sejam levadas em consideração. Na maioria dos casos, menos de 20 neurônios seriam necessários para classificar a frase corretamente. Além disso, como a escolha é binária, apenas uma camada oculta é necessária, de modo que se crie uma "linha" que classifica a frase.

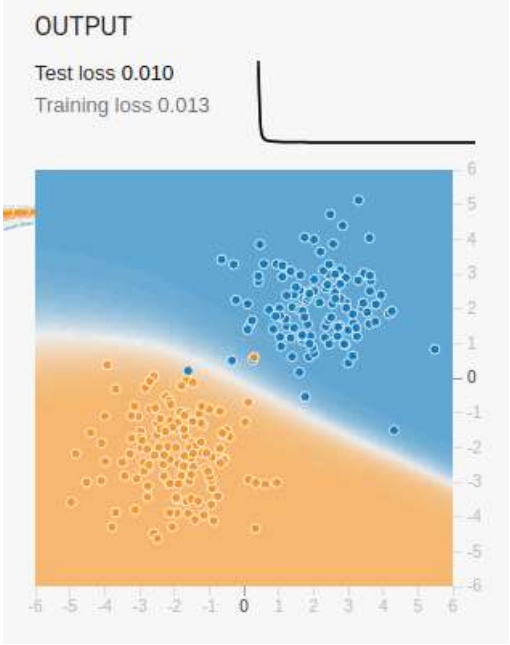

Figura 4 – Diagrama MLP, 4 features de entrada, 1 camada oculta, 8 neurônios

2.3.5 Comitês de Máquina

Os Comitês de Máquina explorados no atual trabalho são baseados em métodos de *Ensemble* (Conjunto) descritos na documentação do *scikit-learn*. O objetivo é de combinar as predições de múltiplos estimadores base construídos com um ou mais modelos

de aprendizado com a intenção de melhorar a generalização e robustez em relação à um classificador único (PEDREGOSA et al., 2011).

O comportamento do comitê varia de acordo com o tipo, com os classificadores base utilizados e com os hiperparâmetros definidos. Neste trabalho, foram conduzidos experimentos com o *Voting Classifier* (Classificador por Voto), *Bagging Classifier* (Classificador em Bolsas) e *Custom Classifier* (Classificador Personalizado).

2.3.5.1 Voting Classifier

A ideia por trás de um *Voting Classifier*[5] é combinar múltiplos modelos na realização de uma classificação. Tal classificador pode ser útil com um conjunto de diferentes modelos balanceando suas fraquezas individuais (PEDREGOSA et al., 2011). Esta abordagem é considerada intuitiva, imitando a tendência de buscar por opiniões de diferentes fontes antes de uma tomada de decisão (ROKACH, 2010).

Utilizando o *scikit-learn*, a classificação realizada de um *Voting Classifier* pode ser estabelecida de duas formas: de acordo com a maioria dos votos (*hard*) ou com a média das probabilidades previstas pelos classificadores (*soft*).

Tabela 4 – Exemplo de votação com probabilidades

Modelo	Classe 1	Classe 2
Modelo 1	**0.6**	0.4
Modelo 2	**0.7**	0.3
Modelo 3	0.1	**0.9**

Acompanhe na Tabela 4 um exemplo que apresenta três modelos e as probabilidades atribuídas para as classes 1 e 2. Se o tipo de votação definido for *soft*, então a classificação final seria Classe 2, pois a sua probabilidade é, em média, maior. No entanto, utilizando o sistema de votação *hard*, a classificação final seria Classe 1, pois, individualmente, esta classe recebeu mais votos dos modelos. É possível também atribuir pesos aos votos dos modelos e, assim, ressaltar os votos de alguns em detrimento de outros.

No atual trabalho, ambos os sistemas de votação são explorados, assim como diferentes configurações na ponderação dos pesos. Utilizando o *Grid Search*[6], é possível realizar uma busca pela combinação ótima dos pesos que resulte no melhor desempenho.

[5] https://scikit-learn.org/stable/modules/ensemble.html#voting-classifier
[6] Técnica que visa o refinamento de hiperparâmetros

2.3.5.2 Bagging Classifier

O *Bagging Classifier*[7] é um meta-estimador que, com um modelo de classificação base, realiza vários treinamentos com subconjuntos aleatórios do *dataset* original e agrega suas predições em uma predição final (PEDREGOSA et al., 2011). Esta agregação, assim como no *Voting Classifier*, pode ser dada através de uma votação ou pelo cálculo da média das probabilidades.

Qualquer modelo de classificação pode servir como base para o *Bagging Classifier*. Sua aplicação visa a melhora do desempenho do classificador base. No atual trabalho, utilizaremos o *Bagging Classifier* em todos os modelos avaliados e compararemos os resultados com os dos modelos sozinhos.

2.3.5.3 Custom Classifier

Por fim, o *Custom Classifier* é definido como um meta-estimador completamente customizável. Para tal, utilizamos de ferramentas oferecidas pelo sklearn[8], de modo que a única preocupação é de assimilar as entradas e saídas esperadas de cada método do classificador.

Através do *Custom Classifier*, podemos definir, por exemplo, um classificador com "voto de minoria", que dado uma lista de classificadores, levaria em conta se ao menos um deles votasse positivamente. Também é possível definir votos absolutos baseados em alguma métrica, como a precisão, ou até mesmo criar um classificador idêntico ao *Voting Classifier* em modo *hard* ou *soft*. Para isso, variações são criadas com o objetivo de maximizar métricas específicas, como a precisão e/ou o recall.

Como o *Custom Classifier* construído neste trabalho utiliza de outros classificadores para seu funcionamento, estes precisam estar no seu estado ideal de calibragem, para que os melhores resultados sejam gerados.

2.4 Métricas

A fim de determinar quais métricas devem ser levadas em consideração ao avaliar os modelos de classificação, é necessário ter um entendimento do problema e de como métricas em questão são geradas. No contexto de uma classificação binária, elas se derivam dos conceitos apresentados na matriz de confusão na Tabela 5.

[7] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html
[8] https://scikit-learn.org/stable/developers/contributing.html#rolling-your-own-estimator

Tabela 5 – Matriz de confusão para a classe **C1** - Discurso de ódio

TP - True Positive (Verdadeiro Positivo) Discurso de ódio corretamente classificado como tal.	FP - False Positive (Falso Positivo) Discurso comum erroneamente classificado como discurso de ódio.
FN - False Negative (Falso Negativo) Discurso de ódio erroneamente classificado como discurso normal.	TN - True Negative (Verdadeiro Negativo) Discurso de comum corretamente classificado como tal.

Observando a matriz de confusão na Tabela 5, torna-se evidente que os eventos representados de vermelho são indesejáveis, pois representam erros. Contudo, deve-se avaliar o impacto dos dois tipo erros e definir os seus custos, e, dessa forma, apontar o mais indesejável. Esta análise serve de base na definição de qual métrica será utilizada na avaliação de desempenho dos algoritmos. Isso porque o algoritmo pode ser tendencioso à realização de determinada classificação, sendo benéfico para uma métrica em detrimento outra, tornando imprescindível o uso de múltiplas métricas para tornar a avaliação mais balanceada.

A biblioteca scikit-learn oferece uma série de métodos que retornam métricas e nos ajudam a fazer esta análise. O método *"classification_report"*[9], utilizado neste trabalho, retorna as principais métricas no seguinte formato:

	precision	recall	f1-score	support
0	0.85	0.82	0.84	267
1	0.43	0.50	0.46	74
micro avg	0.75	0.75	0.75	341
macro avg	0.64	0.66	0.65	341
weighted avg	0.76	0.75	0.75	341

Figura 5 – Precision, Recall e F1-score do algoritmo Random Forest.

A precisão, dada como ρ, é definada da seguinte forma:

$$\rho = \frac{TP}{(TP + FP)} \tag{2.3}$$

O *recall*(τ), ou cobertura, pode ser definida de modo semelhante:

$$\tau = \frac{TP}{(TP + FN)} \tag{2.4}$$

O valor F1, que é uma média harmônica entre precisão e recall, é definido pela equação:

$$F1 = 2 * \left(\frac{\rho\tau}{\rho + \tau}\right) \tag{2.5}$$

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

Por fim, a acurácia é dada por:

$$\alpha = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.6}$$

2.5 Trabalhos Relacionados

A detecção de conteúdos impróprios na internet tem sido alvo de diversos trabalhos. No âmbito textual, muitas abordagens distintas são utilizadas. Os métodos comuns e mais simples de se lidar com o conteúdo impróprio fazem uso de uma "lista negra"com uma coleção de termos ofensivos, porém, esses métodos falham na detecção de tipos mais sutis e menos obscenos de discurso de ódio (NOBATA et al., 2016).

Um estudo que analisa práticas comuns para escapar de filtros de "lista negra"é o "Detecting Hate Speech on the World Wide Web"(WARNER; HIRSCHBERG, 2012). Na definição deste trabalho, o discurso de ódio usa uma quantidade pequena de termos específicos para o grupo minoritário alvo. Em seguida, realiza classificações de discusso anti-semita alcançando a acurácia de 94% e recall de 60%.

Utilizando Redes Neurais Profundas, o trabalho "Deep Learning for Hate Speech Detection in Tweets"trata-se de uma pesquisa com múltiplas arquiteturas de redes neurais visando otimizar a classificação comentários como "ofensivo", "racista"ou "nenhum". Utilizando um *dataset* com mais de 16 mil observações, o trabalho alcança um F1-score 18 pontos acima de outras técnicas avaliadas que utilizam o *n-grams*.

Das contribuições em português brasileiro, vale citar o trabalho "Offensive comments in the brazilian web: a dataset and baseline results"(PELLE; MOREIRA, 2017), que trata do *Hate2Vec*, uma solução baseada em um conjunto de classificadores, (tal como a proposta do atual trabalho) que apresenta resultados melhores do que o classificador BoW tradicional [10], com o F1-score acima de 0.9.

[10] Modelo baseado na frequência de termos com Bag Of Words

3 Ferramentas de apoio

3.1 NLTK

Dado como um conjunto de bibliotecas para processamento de linguagem natural, o NTLK oferece interfaces de processamento para classificação, *"tokenização"*, *"stemming"*, *"tagging"* e análise. Apesar de ser uma biblioteca voltada para o inglês, os principais módulos do NLTK contam com suporte para o português, como o *stemming* e o módulo de *stopwords*.

O módulo de *stemming* tem como objetivo retirar o sufixo das palavras, de modo que se mantenha apenas o seu radical. Esse método aumenta consideravelmente a chance de palavras sinônimas se corresponderem (BONZANINI, 2015), melhorando assim a performance geral do analisador de frases. O módulo do *stopwords* também é focado na melhoria da performance, visto que tais palavras agregam pouco valor semântico à frase (mais informações sobre as técnicas citadas na seção 2.1.4). Observe em 3.1 um exemplo de remoção de *stopwords*.

```
>>> from nltk.corpus import stopwords
>>> from nltk.tokenize import word_tokenize

>>> stop_words = set(stopwords.words('portuguese'))
>>> example_sent = "Este é um exemplo de frase que podemos remover as
    palavras vazias"

>>> word_tokens = word_tokenize(example_sent)
>>> filtered_sentence = [w for w in word_tokens if not w in stop_words]

>>> print(filtered_sentence)
['Este', 'é', 'exemplo', 'frase', 'podemos', 'remover', 'palavras', 'vazias
    ', '.']
```

Listing 3.1 – Removendo *stopwords* usando a biblioteca NLTK

3.2 scikit-learn

A biblioteca scikit-learn[1] provê algoritmos de aprendizado supervisionado e não-supervisionado através de interfaces pré-definidas. Todos os algoritmos de classificação considerados neste trabalho estão implementados nesta biblioteca.

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

Além da classificação, o scikit-learn contém outros módulos, como os de regressão e clusterização. Por se tratar de uma biblioteca para mineração e análise de dados, uma série de pacotes é necessária para o seu aproveitamento, como o matplotlib[2], o numpy[3] e o pandas[4]. Esses pacotes fazem parte de um stack(conjunto) chamado Pydata. Neste projeto, todos os classificadores utilizados e testados foram providos pelo scikit-learn.

A biblioteca do scikit-learn também possui suporte para a aplicação de técnicas de treinamento e organização do código, como:

- **Pipeline**: uma classe que permite a construção de métodos, compostos por diversos passos e parâmetros variáveis. Nos experimentos realizados neste trabalho, esses passos são as classes responsáveis por extrair os atributos do texto e pelo modelo da classificação.
- **K-fold cross-validation**: uma técnica que visa eliminar o viés gerado ao realizar a divisão do *dataset* entre dado de treino e dado de teste. A técnica propõe solucionar este problema dividindo os dados de treino em K partes e realizando o experimento proposto K vezes, revezando, em cada iteração, a parte a ser utilizada no teste.
- **GridSearchCV**: uma técnica que realiza uma busca por "força bruta", para encontrar os hiperparâmetros do modelo de classificação que melhor atendem o problema. Para isso, utilizando o scikit-learn, é definido um conjunto de parâmetros a serem avaliados e o critério de avaliação do método. Assim, o desempenho do modelo utilizando todas as possíveis combinações dos parâmetros sugeridos é avaliado. De acordo com a documentação do scikit-learn, a busca pelos parâmetros é otimizada pelo uso da validação cruzada (PEDREGOSA et al., 2011).Para um melhor entendimento observe a Figura 6 que ilustra o uso do K-fold na etapa de busca de parâmetros.

[2] https://matplotlib.org/
[3] https://www.numpy.org/
[4] https://pandas.pydata.org/

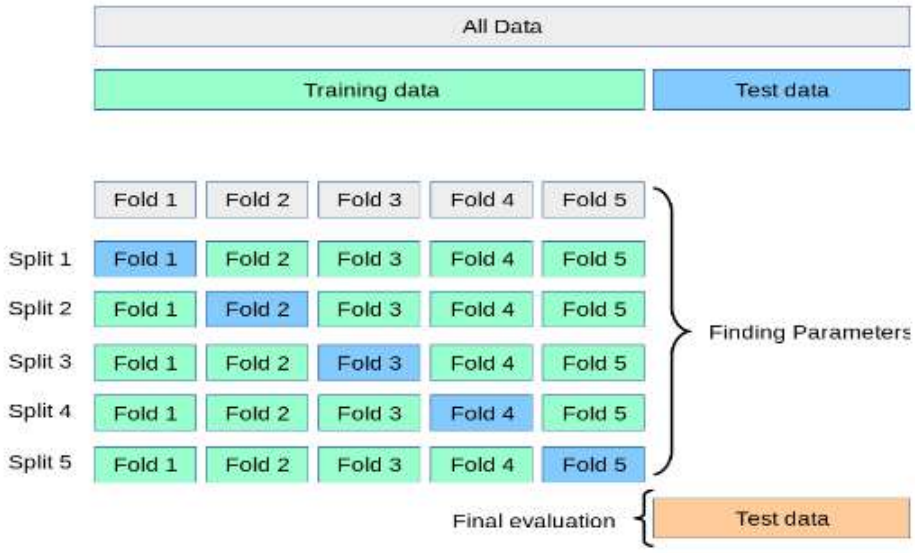

Figura 6 – FONTE: scikit-learn.org

3.3 Jupyter notebook

O Jupyter notebook é uma ferramenta que permite não só a execução de código python no navegador, mas também seu gerenciamento, organização e apresentação. Com ela, o desenvolvimento dos algoritmos é facilitado, visto que a execução dos mesmos pode ser separada por "blocos". Seguindo essa mesma lógica, é possível utilizar-se de comentários em *markdown* para documentar esses blocos de algoritmo, de modo que se facilite tanto a organização, quanto a leitura posterior desses dados. Com todas essas vantagens, o jupyter notebook se tornou nossa principal ferramenta para a realização dos testes. Esta também é uma ferramenta do conjunto Pydata.

3.4 Conjunto de dados

Um dos desafios enfrentados no atual trabalho foi a busca por um conjunto de dados (*dataset*) categorizado de discurso de ódio que atenda aos requisitos: ser em português brasileiro, utilizar linguagem coloquial e ser composto por textos pequenos. Estes requisitos foram definidos para que os dados utilizados nos treinos sejam o mais próximo possível do contexto onde o classificador será aplicado.

Tabela 6 – Informações do dataset

Tamanho	1250 frases
Discurso de ódio	419 frases
Média de caracteres.	72.5
Média de palavras	12.8

Figura 7 – Captura de tela no Jupyter Notebook

O conjunto de dados encontrado que melhor atende aos requisitos do projeto é o OffComBR (PELLE; MOREIRA, 2017). A fim de se obter familiaridade com os dados, e como parte da etapa de Entendimento dos Dados do CRISP-DM (mais informações no capítulo 4.1), uma análise exploratória dos dados foi conduzida e algumas informações foram obtidas, como: o *dataset* é composto por mais de mil e duzentos comentários, sendo 419 classificados como discurso de ódio ou conteúdo impróprio. No conteúdo das observações, diferentes assuntos são abordados, como política, economia, futebol e religião. As frases contém uma média de 72 caracteres, porém nenhuma ultrapassa 100, além de um alto grau de coloquialismo e erros de digitação. Racismo, sexismo, homofobia, xenofobia e intolerância religiosa estão entre os critérios adotados na classificação manual realizada pelos seus colaboradores. Ainda assim, é importante ressaltar que este *dataset* não é uma representação perfeita dos dados que serão classificados posteriormente na plataforma EJ, pois sua origem vem de uma rede social com um paradigma de comunicação diferente, e isso pode influenciar os dados coletados. Observe, na Tabela 7, as cinco primeiras observações do *dataset*.

Tabela 7 – Trechos do dataset OffComBR

	Classe	Frase
0	C1	Votaram no PEZAO agora tomem no **ZAO
1	C0	So conversa mole desde que golpeou o poder nao e Temer
2	C0	FORA TEMER ABAIXO A REDEGLOBODECORRUPCAO.
3	C1	VAO BATE PANELASSEUS BURROSBEM FEITO
4	C0	Podiam retirar dos lucros dos bancos

4 Metodologia

4.1 Fluxo de atividades

A sequência de atividades que engloba este trabalho baseia-se no modelo **CRISP-DM**, uma abreviação de CRoss Industry Standard Process for Data Mining (Processo Padrão de Vários Segmentos de Mercados para Mineração de Dados). Este é um modelo de processo e metodologia da IBM que busca orientar os esforços da organização descrevendo as fases típicas e tarefas envolvidas do ciclo de vida em um projeto de mineração de dados (IBM, 2015).

A Figura 8 mostra o fluxograma padrão que representa uma visão geral das fases do CRISP-DM.

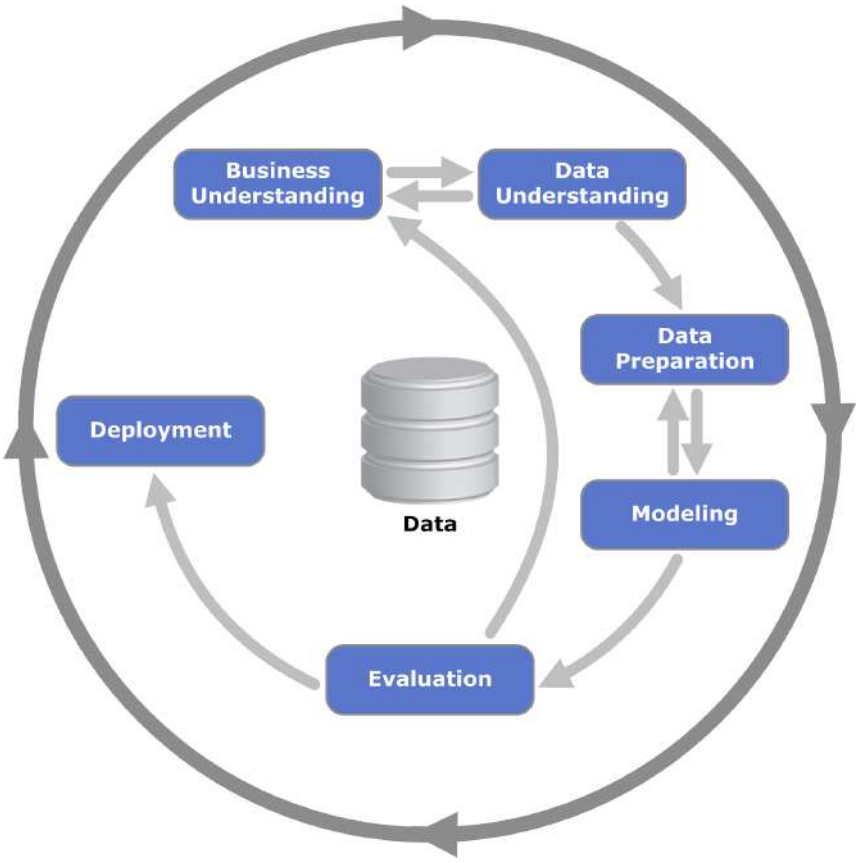

Figura 8 – O ciclo de vida da mineração de dados. FONTE: (IBM, 2015)

O modelo CRISP-DM é bastante flexível, podendo ser customizado de acordo com as necessidades do projeto ou da organização. Ele pode ser executado em múltiplos ciclos antes da fase de Implantação. No momento, a primeira rodada de experimentos e avaliações já foi concluída e o trabalho se encontra no segundo ciclo. Segue abaixo uma

breve descrição das fases ilustradas no modelo.

- **Business Understanding** - Entendimento de Negócios: Trata-se da fase inicial do projeto, onde se investiga as metas de negócios da organização, reúne-se informações básicas e define critérios para o sucesso.
- **Data Understanding** - Entendimento dos Dados: Etapa de verificação dos dados disponíveis a fim de evitar problemas posteriores. Normalmente, esta é a etapa mais longa de um projeto (IBM, 2015), porém, no atual projeto, onde são utilizados apenas textos e labels, não há uma complexidade muito grande na execução desta fase. Mais detalhes sobre o dataset na seção 3.4.
- **Data Preparation** - Preparação de Dados: "Estima-se que a preparação de dados normalmente consome 50-70% do tempo e esforço do projeto" (IBM, 2015). Como não havia a necessidade de derivar novos atributos, mesclar registros, ou remover registros inválidos, nesta etapa, foram utilizadas técnicas de limpeza de texto e processamento de linguagem natural, como stemização e remoção de stop words. Mais detalhes sobre as técnicas utilizadas na seção 2.1.4.
- **Modeling** - Modelando: Nesta fase, a natureza do problema e das metas definidas no Entendimento de Negócios e os dados disponíveis definem as técnicas de modelagem a serem utilizadas. No contexto de classificação de texto com um conjunto já rotulado, opta-se pelo uso de algoritmos de classificação, de aprendizado supervisionado. Estima-se que esta etapa seja realizada em várias iterações. Primeiramente, com os hiperparâmetros padrão e, nas iterações seguintes, refinando-os com o GridSearchCV a fim de buscar por resultados cada vez mais satisfatórios.
- **Evaluation** - Avaliação: Fase onde se analisa e faz inferências sobre o esforço realizado. Para que seja possível cumprir esta etapa de forma objetiva, foram estabelecidos critérios de avaliação para os modelos. As métricas são coletadas e analisadas a fim de encontrar e elencar os modelos que melhor atendem aos objetivos do projeto.
- **Deployment** - Implantação: Fase em que serão realizados estudos sobre a arquitetura da plataforma EJ, onde o classificador será implantado, levando em consideração que o modelo será constantemente retroalimentado, visando seu aperfeiçoamento com dados de situações mais próximas ao contexto da plataforma. Para se chegar à esta fase, é necessário repetir o ciclo até o ponto de avaliação ao menos uma vez, de modo que o sistema satisfaça os requisitos do projeto. Esta fase também envolve planejar e monitorar a implementação dos resultados, além de uma revisão do projeto.

5 Resultados e Discussões

Este capítulo apresenta os resultado obtidos através dos experimentos que configuram as fases de *Data Preparation*, *Modeling* e *Evaluation* do CRISP-DM. Nela, foram utilizados diferentes modelos de classificação, sendo eles: SVM, Random Forest, Multinomial Naive Bayes, Decision Tree, MLP e SVC, além dos modelos de Ensemble na construção de um comitê de máquina, sendo eles: *Voting Classifier*, *Bagging Classifier* e o *Custom Classifier*. Todos os modelos citados passaram pelo processo descrito a seguir, para que, por fim, eles pudessem ser avaliados a e comparados através das métricas obtidas em cada fase de teste.

- **Realizar limpeza do *dataset***: aplicação das técnicas descritas na seção 2.1.4.
- **Refinar hiperparâmetros**: utilizando uma técnica otimização de hiperparâmetros, GridSearchCV, busca-se por hiperparâmetros mais adequados para o atual modelo e para o TF-IDF.
- **Treinar e testar classificador**: trata-se de uma bateria de treinos e testes do classificador. As métricas obtidas são resultados das médias e desvios-padrão entre os 10 *folds* da validação cruzada (k-fold cross validation)

5.1 Resultados dos classificadores

Na análise das métricas, é importante definir o que é mais custoso: **Aprovar erroneamente um comentário impróprio, propagando-o na plataforma, ou rejeitar um comentário legítimo?** Para o primeiro caso, a métrica que se deve dar mais atenção é o recall (visando evitar os falsos negativos), para o segundo, a precisão (para evita os falsos positivos). Considerando o contexto do EJ, e que este trabalho almeja apontar comentários impróprios para serem avaliados por um moderador posteriormente, o custo de permitir que um comentário indevido seja exibido na plataforma é mais alto do que um comentário legitimo rejeitado por engano, já que, neste último caso, um moderador pode intervir. Portanto, a principal métrica para as nossas avaliações dos modelos será o **recall da classe C1** (discurso de ódio), que penaliza justamente as classificações errôneas, onde um comentário contendo discurso de ódio é classificado como discurso comum. Ou seja, o recall da classe C1 indica o percentual de comentários da classe C1 que foi corretamente identificado.

5.1.1 Primeiro ciclo do CRISP-DM

Observe na figura 9 e na Tabela 8 uma comparação entre os modelos de classificação avaliados utilizando as cinco métricas adotadas no projeto[1]. Note que C0 representa o discurso comum e C1, discurso de ódio.

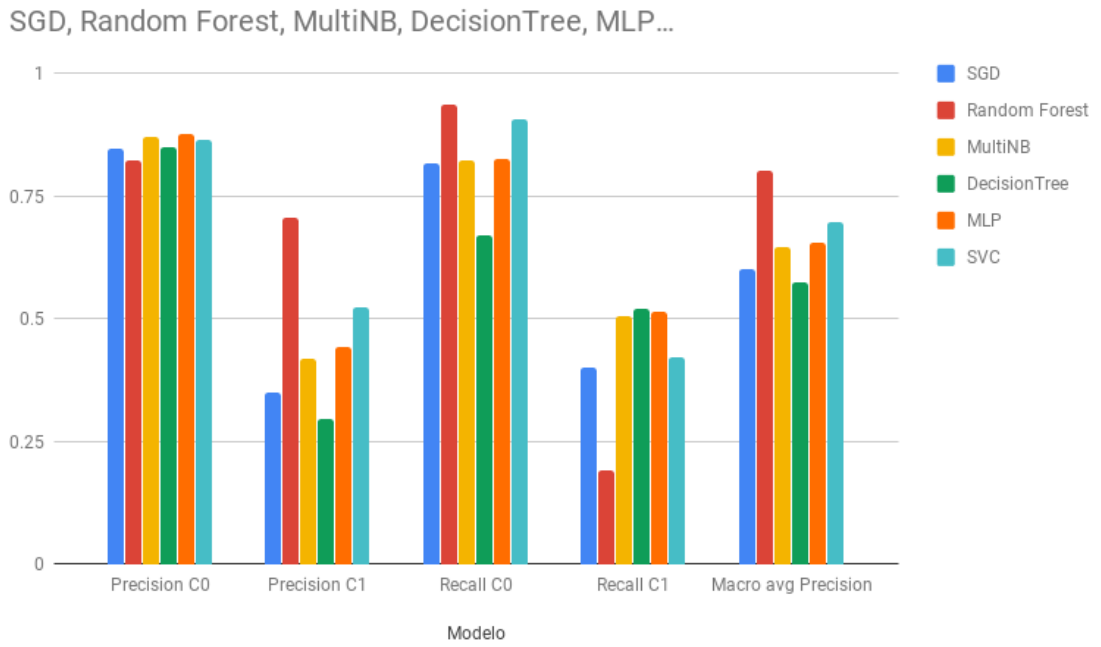

Figura 9 – Métricas do primeiro ciclo.

Tabela 8 – Resultado dos experimentos com os modelos

Modelo	Precisão C0	Precisão C1	Recall C0	Recall C1	Macro avg
SGD	0.84 (±0.02)	0.35 (±0.06)	0.81 (±0.05)	0.40 (±0.10)	0.60 (±0.03)
Random Forest	0.82 (±0.01)	0.70 (±0.30)	0.93 (±0.11)	0.19 (±0.10)	0.80 (±0.06)
MultiNB	0.87 (±0.02)	0.42 (±0.08)	0.82 (±0.04)	0.50 (±0.07)	0.64 (±0.05)
DecisionTree	0.85 (±0.03)	0.29 (±0.07)	0.67 (±0.14)	0.52 (±0.12)	0.57 (±0.05)
MLP	0.87 (±0.01)	0.44 (±0.09)	0.82 (±0.07)	0.51 (±0.05)	0.65 (±0.04)
SVC	0.86 (±0.02)	0.52 (±0.09)	0.90 (±0.03)	0.42 (±0.07)	0.70 (±0.05)

[1] Os valores apresentados são médias aritméticas e desvios-padrão gerados a partir de cinco rodadas de experimentos.

Através desses dados, é possível identificar que o modelo do *Random Forest* se destacou dos demais por sua precisão ao identificar a classe 1. Isso significa que, ao classificar um comentário como discurso de ódio, este modelo é relativamente confiável. O alto desvio padrão também revela uma variação nesse valor, ou seja, nem sempre o desemprenho é tão alto. O Recall C1 esteve abaixo da média, mostrando uma tendência ao falso negativo para a mesma classe, ou seja, altas chances de se deixar passar um discurso de ódio.

Enquanto o *Random Forest* se destaca pela disparidade em uma das métricas, o modelo *Multi-Layer Perceptron* (MLP) se apresenta com bons valores para o recall da classe 1, sem comprometer as outras. Ao compará-lo com os outros modelos, ele se apresenta entre os melhores em todas as cinco métricas exibidas acima.

5.1.2 Segundo ciclo do CRISP-DM

Durante a segunda rodada do CRISP-DM, novos experimentos foram conduzidos buscando uma melhoria de performance em relação ao primeiro ciclo. As métricas adotadas também passaram por mudanças visando apenas a Classe C1, e para dar uma visão geral do desempenho, a acurácia também foi coletada.

Na construção de um comitê de máquina, utilizamos três abordagens: o *Voting Classifier*, o *Bagging Classifier* e o *Custom Classifier*. A fim de encontrar a melhor combinação de pesos possível para o *Voting Classifier*, foi realizada uma busca utilizando o GridSearch. Como resultado, obtivemos a combinação de peso: 1 para o Multinominal Naive Bayes, e 0 para os outros. Isso significa que não foi possível encontrar combinação que resultasse em um recall (métrica de referência) maior do que o próprio Multinominal Naive Bayes sozinho. Sendo assim, a possibilidade de um comitê de máquina utilizando o *Voting Classifier* foi descartada.

Através dos experimentos conduzidos com o *Bagging Classifier*, não foi possível notar uma melhora significante nas métricas coletadas. Mesmo após o refinamento dos parâmetros, em muitos casos, o modelo teve seu desempenho piorado em relação ao obtido sem o uso do *Bagging*, observe as métricas na Figura 10. Na Tabela 9, as métricas obtidas com os modelos base, acompanhados das diferenças geradas após a utilização do *Bagging*. Os valores de diferença positivos representam melhoras e negativos, pioras. Note que, em nenhum caso de melhora, a diferença foi maior do que o erro calculado.

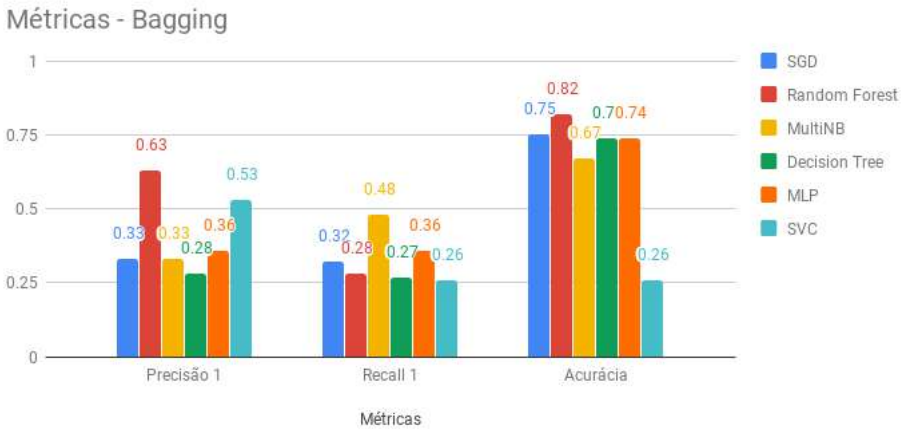

Figura 10 – Métricas dos modelos com o Bagging.

Tabela 9 – Resultado dos experimentos com Bagging

Modelo	Precisão C1	Recall C1	Acurácia
SGD	0.32 (±0.08)	0.36 (±0.14)	0.72 (±0.04)
Diferença	+0.01 (±0.16)	-0.04 (±0.24)	+0.03 (±0.08)
Random Forest	0.62 (±0.17)	0.27 (±0.08)	0.82 (±0.04)
Diferença	+0.01 (±0.29)	+0.01 (±0.16)	+0.00 (±0.08)
MultiNB	0.35 (±0.10)	0.55 (±0.13)	0.70 (±0.07)
Diferença	-0.02 (±0.21)	-0.07 (±0.08)	-0.03 (±0.14)
Decision Tree	0.32 (±0.09)	0.48 (±0.12)	0.68 (±0.08)
Diferença	-0.04 (±0.22)	-0.21 (±0.33)	+0.06 (±0.18)
MLP	0.28 (±0.14)	0.37 (±0.17)	0.71 (±0.05)
Diferença	+0.18 (±0.25)	-0.01 (±0.27)	+0.03 (±0.12)
SVC	0.48 (±0.15)	0.38 (±0.09)	0.79 (±0.03)
Diferença	+0.03 (±0.22)	-0.12 (±0.20)	-0.53 (±0.14)

Por fim, todas as versões do *Custom Classifier* foram construídas levando em consideração os resultados dos procedimentos anteriores, ou seja, analisando o desempenho singular dos modelos, algumas combinações foram criadas a fim de explorar as suas características individuais.

A primeira combinação utilizada, *Custom A*, leva em conta o "voto de minoria", quando, caso apenas um considere o texto impróprio, o mesmo será considerado impróprio. Os classificadores para esta combinação e o seu comportamento buscam maximizar o recall 1, ou seja, inibir ao máximo que comentários impróprios passem despercebidos. Os classificadores base utilizados foram: Random Forest, MultiNB, e SVC.

Já a combinação *Custom B* busca aproveitar a alta precisão 1 do Random Forest, ao mesmo tempo em que a opinião do MLP e SVC são levados em conta. Esta combinação pode ser interpretada como: Random Forest OU (MLP E SVC). Desse modo, aproveitamos a alta precisão do Random Forest como um voto absoluto, enquanto aprimoramos suas fraquezas com outras duas opiniões dependentes.

A combinação *Custom C* se assemelha a *Custom A*, mas utilizando todos os seis classificadores disponíveis. Com essa aplicação, conseguimos obter o maior recall 1 de todos os modelos, porém, sacrificando a precisão.

Os tempos de processamento representados na Tabela 10 dizem respeito aos tempos de classificação para uma amostra contendo 341 observações. Note que, como esperado, os modelos de Comitê de Máquina são os que apresentam os maiores tempos de execução, visto que, estes, precisam realizar múltiplas classificações preliminares antes de chegarem à uma classificação final.

Tabela 10 – Resultados do segundo ciclo

Modelo	Precisão C1	Recall C1	Acurácia	Tempo de processamento
SGD	0.32 (±0.08)	0.36 (±0.14)	0.72 (±0.04)	0.017 s
Random Forest	0.62 (±0.17)	0.27 (±0.08)	0.82 (±0.04)	0.059 s
MultiNB	0.35 (±0.10)	0.55 (±0.13)	0.70 (±0.07)	0.010 s
Decision Tree	0.32 (±0.09)	0.48 (±0.12)	0.68 (±0.08)	0.022 s
MLP	0.28 (±0.14)	0.37 (±0.17)	0.71 (±0.05)	0.021 s
SVC	0.48 (±0.15)	0.38 (±0.09)	0.79 (±0.03)	0.025 s
Custom A	0.34 (±0.09)	0.64 (±0.10)	0.67 (±0.08)	0.069 s
Custom B	0.48 (±0.15)	0.45 (±0.11)	0.77 (±0.04)	0.136 s
Custom C	0.27 (±0.05)	0.77 (±0.11)	0.53 (±0.10)	0.182 s

Na Figura 12, uma representação recall em relação à precisão dos melhores classificadores construídos até aqui. Os modelos com o prefixo "B_"são os aqueles que estão usando o *Bagging*. Note que a linha de tendência deixa claro que quanto maior a precisão, menor o recall. Tendo em mente este relacionamento, destacam-se os modelos que se apresentam acima da linha. Dos apresentados, o modelo mais equilibrado é o *Custom B*, pois ele possui uma precisão relativamente alta com o recall acima da média.

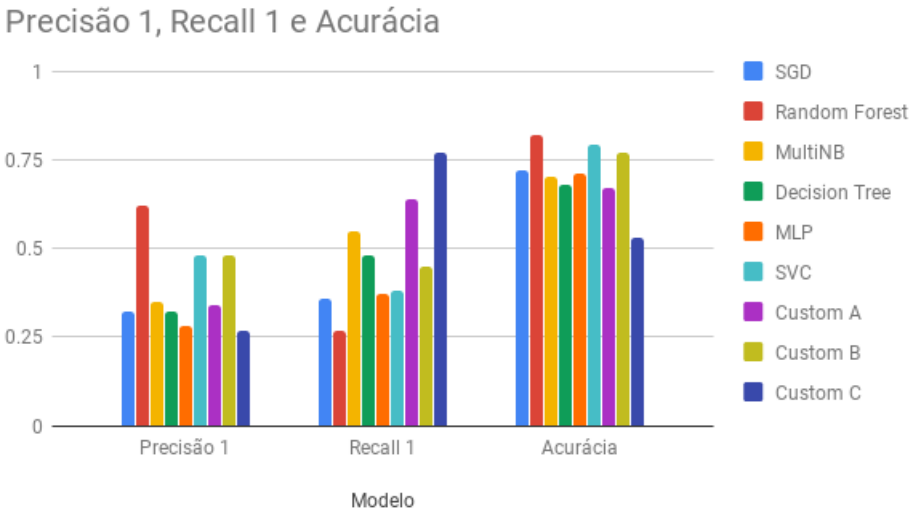

Figura 11 – Comparação dos modelos.

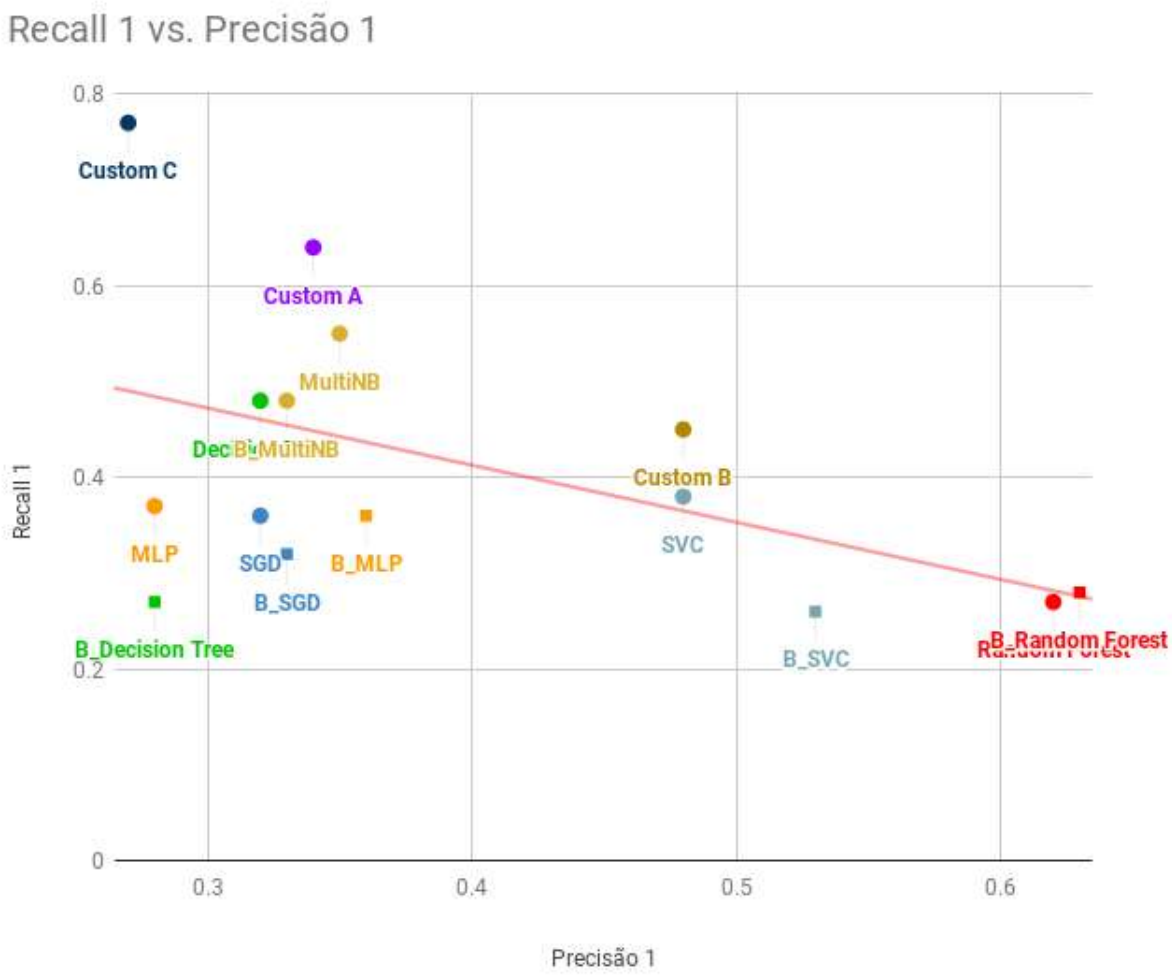

Figura 12 – Linha de tendência da precisão x recall dos modelos.

Ainda que o modelo *Custom B* possua uma característica desejável em termos de equilíbrio entre precisão e recall, todos os modelos próximos à linha de tendência podem ser considerados "equilibrados"no sentido de não favorecer fortemente nenhum dos dois aspectos. O *Custom B* e o SVC se localizam próximos ao centro da linha de tendência em relação à precisão e um pouco acima em relação ao recall. A depender da necessidade de negócio e como se pondera o risco de aceitar um comentário de ódio ou deixar de aceitar automaticamente um comentário comum, é possível favorecer outros modelos ao longo dessa linha como o *Random Forest* (favorecendo a precisão) ou o *Multinomial Naive Bayes* e *Decision Tree* (favorecendo o recall).

6 Considerações Finais

A plataforma Empurrando Juntos foi apresentada como um ambiente inovador de discussão e engajamento social e que busca trazer um ambiente que favoreça um debate mais produtivo, partindo de um ponto de vista completamente diferente das redes sociais atuais. Com um enfoque em tópicos de cunho social, a plataforma pode tratar de assuntos delicados, como preconceito, homofobia e racismo. Existe, portanto, a necessidade de se filtrar o conteúdo, principalmente pelo objetivo de manter o decoro do ambiente, impedir o assédio moral e disseminação do discurso de ódio. Com base nisso, este projeto propõe uma solução que automatiza, mesmo que parcialmente, a moderação de comentários em plataformas de conversa ou discussão.

Utilizando de classificadores já conhecidos, refinados os seus parâmetros para adequá-los ao escopo de classificação definido, testados os seus funcionamentos individuais e, com a intenção de melhorar os resultados, este trabalho propôs diferentes abordagens e mesclagens com os classificadores para atingir métricas satisfatórias. Com isso, alguns resultados foram alcançados, destacando os três deles, dados pelo *Custom Classifier*: O primeiro resultado, onde em torno de 65% de todos os comentários impróprios foram identificados, mas com uma precisão de 34%. Um segundo resultado mais equilibrado, onde tivemos 45% de todos os comentários impróprios identificados, com uma precisão de 48%. E por fim, o resultado mais discrepante, utilizando do método de voto de minoria, com 77% de todos os comentários impróprios barrados, ao preço de uma precisão de 27%. Nota-se que, o preço que se paga ao elevar o recall C1 (elevar o percentual de comentários impróprios capturados) é reduzir a precisão dessa mesma classe. Isso significa que, apesar do alto nível de captura de comentários impróprios, muitos comentários próprios também estão sendo barrados, dada a baixa precisão da classe C1.

Cada resultado gera um caso de uso diferente. Para uma moderação mais equilibrada, o classificador *Custom B* apresenta resultados acima da média tanto no ponto de vista da precisão quanto no ponto de vista do recall. Para o caso em que já existe uma moderação inclusa em todos os comentários (Como no EJ), a implementação do *Custom Classifier Custom C* se torna uma opção viável, visto que ele foi capaz de detectar quase todos os comentários impróprios, apesar de sua baixa precisão indicar que a redução de custos seria pequena, já que quando o classificador erra muito, ele gera trabalho para o moderador. Desse modo, concluímos que podemos utilizar um classificador simples ou um comitê customizado composto por um ou vários classificadores. A escolha dependerá do caso de uso em que essa classificação será inserida e quais métricas são as mais importantes para o contexto em questão.

Referências

ARANHA, C. N. Uma abordagem de pré-processamento automático para mineração de textos em português: Sob o enfoque da inteligencia computacional. 2007. Citado na página 23.

ASHARI, A.; PARYUDI, I.; TJOA, A. min. Performance comparison between naïve bayes, decision tree and k-nearest neighbor in searching alternative design in an energy simulation tool. *International Journal of Advanced Computer Science and Applications*, v. 4, 2013. Citado na página 30.

BONZANINI, M. Stemming, lemmatisation and pos-tagging with python and nltk. 2015. Disponível em: <https://marcobonzanini.com/2015/01/26/stemming-lemmatisation-and-pos-tagging-with-python-and-nltk/>. Citado na página 37.

CHANG, C.-C.; LIN, C.-J. Libsvm – a library for support vector machines. 2018. Citado na página 29.

HURWITZ, J.; KIRSCH, D. *Machine Learning for dummies*. [S.l.], 2000. Citado na página 27.

IBM. Ibm spss modeler crisp-dm guide. 2015. Disponível em: <ftp://public.dhe.ibm.com/software/analytics/spss/documentation/modeler/17.1/br_po/ModelerCRISPDM.pdf>. Citado 3 vezes nas páginas 11, 41 e 42.

LIBERMAN, N. Decision trees and random forests. 2017. Citado na página 30.

LUNDBORG, A. Text classification of short messages. *LU-CS-EX 2017-14*, 2017. ISSN 1650-2884. Disponível em: <http://lup.lub.lu.se/student-papers/record/8928009>. Citado 2 vezes nas páginas 24 e 28.

MATH WORKS. Introducing machine learning. 2016. Citado na página 27.

MATSUBARA, E. T.; MARTINS, C. A.; MONARD, M. C. Pretext: uma ferramenta para pré-processamento de textos utilizando a abordagem bag-of-words. 2003. Disponível em: <https://bdpi.usp.br/single.php?_id=001344333>. Citado na página 26.

NOBATA, C. et al. Abusive language detection in online user content. In: *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2016. (WWW '16), p. 145–153. ISBN 978-1-4503-4143-1. Disponível em: <https://doi.org/10.1145/2872427.2883062>. Citado na página 35.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011. Citado 3 vezes nas páginas 32, 33 e 38.

PELLE, R. P. de; MOREIRA, V. P. Offensive comments in the brazilian web: a dataset and baseline results. 2017. Citado 2 vezes nas páginas 35 e 40.

REBS, R. R. O excesso no discurso de ódio dos haters. 2017. Disponível em: <https://periodicos.ufsc.br/index.php/forum/article/view/1984-8412.2017v14nespp2512/35377>. Citado na página 19.

ROKACH, L. *Pattern Classification Using Ensemble Methods*. [S.l.]: World Scientific, 2010. Google-Books-ID: yr9pDQAAQBAJ. ISBN 978-981-4271-06-6. Citado na página 32.

RUSSELL, S. J.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. [S.l.: s.n.], 2003. 499 p. Citado na página 30.

SEBASTIANI, F. Machine learning in automated text categorization. *ACM Computing Surveys (CSUR)*, v. 34, n. 1, p. 1–47, Jan 2002. ISSN 0360-0300. Citado 2 vezes nas páginas 27 e 28.

STROPPA, T.; ROTHENBURG, W. C. Liberdade de expressÃo e discurso do Ódio: O conflito discursivo nas redes sociais. *Revista Eletrônica do Curso de Direito da UFSM*, v. 10, 12 2015. Citado na página 19.

WARNER, W.; HIRSCHBERG, J. Detecting hate speech on the world wide web. In: *Proceedings of the Second Workshop on Language in Social Media*. Association for Computational Linguistics, 2012. (LSM '12), p. 19–26. Disponível em: <http://dl.acm.org/citation.cfm?id=2390374.2390377>. Citado na página 35.