Universidade Federal Rural de Pernambuco
Departamento de Estatística e Informática

Uma aplicação de impacto social com aprendizagem de máquina

Johann Gomes Barros Lima

Recife

Agosto de 2014

Johann Gomes Barros Lima

Uma aplicação de impacto social com aprendizagem de máquina

Orientador: Teresa Medeiros Maciel

Co-Orientador: Tiago Alessandro Espíndola Ferreira

Monografia apresentada ao Curso Bacharelado em Sistemas de Informação da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Recife

Agosto de 2014

À minha mãe, Olga
Aos meus orientadores, Teresa e Tiago
Aos meus queridos amigos.

Agradecimentos

À minha mãe, Olga, por sempre ter me mostrado a importância de estar continuamente aprendendo e buscando conhecimentos, até o fim de nossas vidas. Ela sempre acreditou no poder transformador da educação, não só no sentido de fazer com que posições sejam galgadas profissionalmente, mas, principalmente como uma forma de fazer com que sejamos seres humanos cada vez melhores, e deixemos um legado positivo na sociedade.

À minha professora orientadora, Teresa, que sempre esteve muito disposta a me ajudar a desenvolver esse trabalho, principalmente no que diz respeito a como desenvolê-lo e, posteriormente, escrevê-lo. Na correria diária, apesar de ser uma pessoa de muitos compromissos, ela sempre conseguiu encaixar algum momento dos seus dias no acompanhamento deste trabalho, sempre com muito empenho, vontade e ânimo. Além de uma força motivadora, Teresa é um pessoa repleta de grandes ideias, ela foi fundamental durante a etapa inicial deste trabalho.

Ao meu co-orientador, Tiago, pelas ideias que transformaram o trabalho em algo com profundidade computacional muito relevante. Tiago atuou como um guia de importância fundamental para a conclusão desse projeto, mostrando o "caminho das pedras", sempre que precisei.

Aos meus amigos, que representam para mim uma família de grande valor. Eles indiretamente me ajudaram a compor esse trabalho, cada um da forma que lhes foi possível. Deixo aqui agradecimentos especiais à minha amiga Aline Castilho Gomes Ribeiro (Graduanda em Ciências Sociais - UFRPE - PE) por estar presente na apresentação oral do trabalho. Agradeço também, especialmente, aos meus amigos Edigleisson Alcântara (Mestrando em Psicologia - UFPE - PE), Leonardo Dias Silveira (Graduado em Publicidade e Propaganda - FACHA - RJ), Thais Moura de Freitas (Graduanda em Sistemas de Informação - UFRPE - PE) e Tiago Rafael (Doutorando em Biologia - UFPE - PE) por me ajudarem na revisão do trabalho escrito e/ou apresentação.

Resumo

A desigualdade social ocorre quando um grupo limita, censura, discrimina e/ou prejudica o status de outro grupo, classe ou círculo social, e diminui o acesso destes às suas liberdades fundamentais, privando-os dos seus direitos assegurados como seres humanos. O *software*, como um atuante transformador em nossa sociedade, tem um grande potencial para produzir um impacto social positivo, podendo inclusive diminuir a desigualdade e fazer justiça social, considerando que ele está presente na vida de cada cidadão em qualquer lugar do mundo. Este trabalho, tem como objetivo trazer mudanças sociais relevantes às comunidades através de um *software*, utilizando como ferramenta a tecnologia. O *software* desenvolvido, permite que as pessoas vítimas de discriminação elaborarem denúncias por meio de um aplicativo que facilita o processo de escrita, e o torna mais rápido para o usuário, utilizando um algoritmo de classificação para categorizar o tipo em que elas se enquadram, automaticamente, por mineração de texto (*Text-Based Mining*). Ao fim do trabalho, é feito um estudo que compara o desempenho observado de vários algoritmos de classificação para o problema proposto, de categorização automática de denúncias, com a massa de dados utilizada neste trabalho para o treinamento dos algoritmos.

Palavras-chave: desigualdade social, software de impacto social, algoritmos de classificação, aprendizado de máquina, support vector machine, k-nearest neighbor, random forest

Abstract

Social inequality occurs when a group limit, reproach, discriminate and / or affect the status of another group, class or social circle, decreasing their access to their fundamental freedoms, depriving them of their rights assured as humans. Software, acting as a ?game-changer? in our society, have great potential to produce a positive social impact and can even reduce inequality through social justice, considering its ubiquity in the modern world. This work, aims to bring relevant social change to communities through technology. The implemented software enables victims of discrimination, to write complaints through an application that makes this process easier and faster, using an algorithm to categorize the type of discrimination it represents, automatically, by text mining (Text-based Mining). Additionally, a study comparing the performance, observed in different Classification Algorithms for the proposed problem of automatic categorization of complaints, is conducted. This study is done using our mass of data, created to train the algorithm, with the objective of predict types of complaints.

Keywords: social inequality, social impact software, classification algorithms, machine learning, support vector machine, k-nearest neighbor, random forest

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

1.1 Apresentação

A desigualdade social ocorre quando um grupo tem o efeito de limitar, censurar e/ou prejudicar o status de um outro grupo, classe ou círculo social. Diminuindo o acesso do outro às suas liberdades fundamentais como seres humanos.

As áreas de desigualdade social incluem o acesso aos direitos de voto, a liberdade de expressão, a extensão dos direitos de propriedade e de acesso à educação, saúde, habitação de qualidade, mobilidade, férias, assim como outros bens e serviços sociais. [12]

As desigualdades sociais estão relacionadas principalmente á falta de equidade entre gênero, raça e classe. A forma com que as pessoas se comportam em sociedade, com comportamentos/práticas racistas e sexistas, assim como outras formas de discriminação, afeta os direitos e oportunidades das outras pessoas. [23]

O software, como um atuante transformador na sociedade atual, tem grande potencial para produzir um impacto social positivo e provocar mudanças sociais, visto que ele está muito presente em nossa vida, praticamente em todos os lugares o tempo todo [22].

Há várias iniciativas de grandes empresas de tecnologia no sentido de apoiar e desenvolver softwares com esse propósito, elas vêm sendo as grandes fomentadoras de projetos com esse objetivo. Temos por exemplo, o Google, com o *Google Impact Challenge* [18], uma competição realizada em várias regiões do mundo, que premia, anualmente, quatro projetos

criativos e inovadores, que usam a tecnologia para tentar atacar problemas de desigualdade social.

Outra empresa que podemos usar como exemplo é a ThoughtWorks, uma multinacional de desenvolvimento e consultoria em software, que possui escritórios espalhados em todo o mundo. A corporação possui o ideal de causar transformações positivas de impacto social e garantir os direitos humanos por meio da tecnologia como um dos pilares que sustentam a missão da empresa, o pilar é denominado Social Impact Program [20]. Um exemplo ilustrativo de um software desse tipo é visto na figura 1.1.

Figure 1.1: Exemplo de software mobile desenvolvido para ajudar populações carentes de informaçoes de pré-natal e neonatal em Gana.

Aprendizado de Máquina é uma área de Inteligência Artificial, cujo objetivo é o desenvolvimento de técnicas computacionais sobre o aprendizado, bem como a construção de sistemas capazes de adquirir conhecimento de forma automática. Um sistema de aprendizado é um programa de computador que toma decisões baseadas em experiências acumuladas através da solução bem sucedida de problemas anteriores [1].

Os algoritmos de classificação fazem parte do conjunto de técnicas do Aprendizado de

Máquina para previsão de associações, por meio de um conjunto de experiências já acumuladas. Com essa técnica, é facilitado o processo de ensinar uma máquina algo que ela deve fazer. Aprender se torna um processo de análise de histórico.

1.2 Breve descrição do produto resultante do TCC

Esse trabalho objetiva a construção de um *software*, cuja característica fundamental é permitir a qualquer pessoa vítima de alguma situação discriminatória, ultrajante ou humilhante, a possibilidade dela denunciar o ocorrido, preenchendo um formulário descritivo básico sobre o fato e com alguns dados pessoais.

O *software* utiliza, como forma de facilitar o preenchimento do formulário, algoritmos para classificar automaticamente a denúncia escrita em algum tipo de denúncia de discriminação, visando facilitar o preenchimento de formulários dessa natureza. O usuário não necessita preencher um campo identificando o tipo da denúncia, o *software* identifica-o por mineração de dados.

A existência de muitos campos para o preenchimento de um formulário, do ponto de vista do usuário, é intimidador e dificultoso, podendo resultar na perda de interesse em usar a aplicação, pois cria uma barreira entre ele e a máquina [7]. Podemos simplificar esse preenchimento eliminando a necessidade de o usuário preencher alguns campos, fazendo com que o software preencha-os automaticamente.

Os usuários da aplicação serão vítimas de alguma situação discriminatória ou testemunhas de uma situação desse tipo, que desejam reportá-la.

1.3 Trabalhos Relacionados

Em pesquisa realizada, encontrou-se apenas um *software* semelhante ao implementado neste trabalho, desenvolvido por uma Organização Não-Governamental denominada *American-Arab Anti-Discrimination Committee*, com sede no estado americano de Michigan nos Estados Unidos [4]. ADC é a maior organização árabe-americana nos Estados Unidos. Ela está empenhada em apoiar os direitos civis de todas as pessoas, e se opõe ao racismo e á in-

tolerância. O *software* desenvolvido por ela, denomina-se '*ADC Michigan Action App*'.Trata-se de um aplicativo móvel, criado para ajudar a capacitar as comunidades para criar mudanças sociais em um nível individual. O aplicativo fornece aos usuários o poder de denunciar casos de discriminação, assédio, étnica, entre outras formas de intolerância. Estas denúncias são enviadas para a organização, a partir dos dispositivos móveis, instantaneamente. Esse aplicativo não possui os atributos de aprendizagem de máquina que classificam automaticamente a denúncia, facilitando o preenchimento do formulário pela vítima ou testemunha, uma das preocupações deste trabalho.

1.4 Relevância

1.4.1 Relevância Social

Lidar com o problema da desigualdade social é, até hoje, um dos maiores desafios mundiais [27]. É considerado um enorme problema cultural que é intensificado por práticas de consumo, propaganda e diferenças de oportunidades.

Atualmente, a sociedade está muito mais alerta a esse tipo de problema e suas consequências, do que há algumas décadas.

Particularmente para denúncias, existe *software* similar, como por exemplo o '*ADC Michigan Action App*', mas devido à sua pouca abrangência e o seu requisito de fazer o usuário preencher grandes formulários, percebe-se a necessidade de aplicações que tornem esse processo mais simples.

1.4.2 Relevância Técnica

Nos últimos anos, a Inteligência Artificial (IA) ganhou um papel de protagonista mundial de estudos relacionados à computação. Mesmo com esta nova notoriedade vinda à tona somente nos tempos atuais, o estudo de IA não é algo tão novo assim. As pesquisas foram iniciadas logo após o fim da Segunda Guerra Mundial e o seu nome foi definido no final da década de 50 [21]. É sabido que existe uma ampla área de atuação em IA, informação que contrasta com a falta de interesse dos profissionais de ingressarem em estudos relacionados neste setor.

Em [21], Russel e Norving afirmam que devido à falta de pesquisas em IA, ainda existe espaço para o nascimento de muitos 'Einsteins', 'Galileus' e 'Newtons'. Eles fazem menção à inexistência de grandes pensadores revolucionários nesta área de estudos.

Atualmente, a IA cobre uma vasta área de estudos com muitas subáreas. Neste trabalho, o foco principal são as formas de Aprendizado de Máquina (AM). AM é uma peça fundamental para a evolução da robótica e automações de modo geral.

Essa subárea da IA, estuda a forma como a máquina deve aprender automaticamente a fazer previsões a partir de observações de dados passados. A AM lida com problemas de classificação de diferentes exemplos em categorias, a partir das já vistas anteriormente, fazendo uma previsão baseada em histórico. [19]

Dentre as vantagens e motivações do uso do AM está:

- Maior precisão de acerto para a máquina do que se for escrita uma série de regras criadas por seres humanos, para que estas sejam interpretadas pela própria máquina (desde que tenhamos um grande conjunto de dados passados).
- Os humanos normalmente são incapazes de expressar o que sabem para que a máquina consiga imitar (exemplos: regras gramaticais, ou como reconhecer letras do alfabeto por meio de imagens). Fornecer à maquina uma grande massa de exemplos, mostrando o que se quer fazer com que ela aprenda, sendo feita do jeito certo, fazendo-a aprender por meio de uma análise desse histórico, induzindo-a a reconhecer padrões e repetir procedimentos é muito mais eficaz do ponto de vista computacional. Usa-se o que a máquina tem de mais poderoso, que são os recursos de capacidade de uma grande quantidade de cálculos de forma quase imediata.
- Barato e flexível. Pode-se aplicar os algoritmos a praticamente qualquer tipo de problema de classificação. Além disso o código deles é aberto ao público, existindo variadas bibliotecas que servem de interface para eles, disponibilizadas para muitas linguagens de programação, na licença *opensource*.

Para o concluinte do curso, especificamente, é importante gerar e repassar esse tipo de conhecimento, como forma de difundir a área na comunidade, assim como sua relevância e importância aplicada em um projeto real, assim como enriquecer debates da comunidade a partir do que foi apreendido.

1.5 Objetivo

1.5.1 Objetivo Geral

Desenvolver um software web para submissão de denúncias. Deve ser facilitado o seu uso e acesso, podendo a aplicação ser acessada facilmente por meio da maioria dos dispositivos móveis atuais: *notebooks, tablets* ou *smartphones*. Algoritmos de classificação devem estar imbutidos no software, como forma de facilitar o preenchimento do formulário de denúncia, categorizando o tipo de denúncia automaticamente de acordo com as palavras-chave encontradas no corpo da denúncia.

1.5.2 Objetivos Específicos

Dentre os objetivos específicos deste trabalho, temos:

- Gerar conhecimento em torno de algoritmos de classificação utilizando aprendizagem de máquina.
- Desenvolver uma aplicação que utilize um algoritmo de classificação (*Support Vector Machine*) para categorizar as denúncias por meio das palavras-chave utilizadas.
- O *software* deve ser usável na forma de um protótipo executado em ambiente controlado de desenvolvimento, tendo sido implementados suas características fundamentais, descritas neste trabalho em 3.1.1.
- Disponibilizar uma análise comparativa entre alguns algoritmos de classificação que podem ser utilizados (*Support Vector Machine* com diferentes funções *kernel, K-Nearest Neighbor* e *Random Forest*), em termos de tempo de treinamento necessário e precisão das previsões de cada um (taxa de acerto), de acordo com os dados utilizados no contexto deste trabalho.

Capítulo 2

Fundamentação Teórica

2.1 Desigualdade Social e Justiça Social

A desigualdade social, em síntese, é a privação de alguns direitos e liberdades fundamentais, que deveriam ser inerentes a todos os indivíduos. Essa privação é feita por ação de outros grupos sociais dominantes.

Thomas M. Shapiro apresenta um exemplo de desigualdade social hipotético em seu livro, *'The Hidden Cost of Being African American'*, para exemplificar o termo. A demonstração é feita na passagem *'playing field for blacks and whites'*. É ilustrada a recusa de um empréstimo bancário a uma família negra com intenção de comprar uma moradia, ao passo que o mesmo empréstimo é aprovado para uma família branca em mesmas condições financeiras. Ao passo que possuir uma moradia é um importante método de ascender social e economicamente, essa situação faz com que a família negra possua menos oportunidades de prosperar, produzindo desigualdade social. O exemplo mostra a privação de direitos que deveriam ser inerentes a todos os seres humanos. O direito fundamental, neste caso, seria a igualdade de oportunidades. [15]

Justiça social é uma construção moral e política baseada na igualdade de direitos e na solidariedade coletiva. Em termos de desenvolvimento, a justiça social é vista como o cruzamento entre o pilar econômico e o pilar social. A justiça social pode ser vista como o que tenta 'consertar' a desigualdade social, provocada por disfunções culturais e do próprio capitalismo. Para ilustrar o conceito, diz-se que, enquanto a justiça tradicional é 'cega' [23], a

justiça social deve tirar a sua venda para que ela enxergue a realidade e possa compensar as desigualdades que nela se produzem.

Em Uma Teoria da Justiça (*'A Theory of Justice'*), publicado em 1971, Rawls defende que uma sociedade será justa se respeitar três princípios:

- Garantia das liberdades fundamentais para todos.
- Igualdade equitativa de oportunidades.
- Manutenção de desigualdades apenas para favorecer os mais desfavorecidos.

A definição de justiça social como é conhecida hoje foi formulada no século XIX pelos denominadados tomistas, seguidores das ideias de São Tomás de Aquino, padre e filósofo nascido no século XIII.

São Tomás de Aquino pregava que a fé e a razão não podem ser contraditórias e, de acordo com esse pensamento, o conceito de justiça social foi desenvolvido. Ele previa que em uma sociedade democrática, todos os serem humanos são dignos e têm a mesma importância. Portanto, todos possuem direitos e deveres iguais não apenas em aspectos econômicos, mas também relativos á saúde, educação, trabalho, direito á justiça e manifestação cultural.

2.2 Algoritmos de Aprendizado Supervisionado

Os Algoritmos de Aprendizado Supervisionado, são a categoria de algoritmos utilizados de AM, em que estão os algoritmos de classificação, estudados nesse trabalho.

Pela definição de Bigus [8], no aprendizado supervisionado, o objetivo do algoritmo é induzir/prever classificações por meio de rótulos denominados 'classes'. Essa previsão ocorre a partir de exemplos que estão pré-classificados, ou seja, exemplos que estão rotulados com uma classe conhecida. Esses exemplos são mostrados ao algoritmo antes do processo de previsão, de forma que o algoritmo 'aprenda' com eles, percebendo padrões.

Um algoritmo de aprendizado supervisionado produz uma função de inferência a partir dos dados mostrados ao algoritmos, que são posteriormente analisados, esta função é usada para classificar os novos exemplos. Um cenários ótimo permite ao algoritmo determinar

corretamente qual a classe de instâncias associada aos dados ainda não vistos, a partir da observação e análise dos já disponibilizados e rotulados a uma classe.

Com o propósito de lidar com o problema de aprendizagem supervisionada proposto, uma pessoa deve seguir os seguintes passos:

1. **Determinar o tipo dos exemplos de dados de treinamento.** Antes de fazer qualquer coisa, o usuário deve decidir o tipo de dado que será usado para treinamento. Em caso de análise de uma frase escrita à mão, por exemplo, o tipo de dado pode ser um caracter escrito à mão, uma palavra inteira escrita à mão ou mesmo uma linha inteira.

2. **Coletar os dados para compor o conjunto de dados de treinamento.** O conjunto de treinamento deve ser representativo quanto ao uso da função no mundo real. O conjunto de dados pode ser coletado tanto de sistemas automatizados como de seres humanos usando seu próprio conhecimento e raciocínio.

3. **Determinar a representação da função de aprendizado.** A precisão da função de aprendizado depende fortemente de como o objeto é representado. Tipicamente, o objeto a ser previsto, assim como treinado, é transformado em um vetor, contendo os atributos que caracterizam o objeto. O número de características não deve ser tão grande, por conta da chamada '*curse of dimensionality*', mas deve conter informação suficiente para a previsão precisa de um resultado classificatório.

4. **Determinar a estrutura da função de aprendizado e seu correspondente algoritmo de aprendizado.** Por exemplo, o engenheiro de software pode escolher entre usar Support Vector Machines (SVMs) ou Random Forests, ambos algoritmos de classificação.

5. **Completar o design.** Executar o algoritmo de aprendizado a partir dos dados de treinamento coletados. Alguns algoritmos de aprendizado supervisionado requerem que o usuário determine alguns parâmetros de controle. Esses parâmetros devem ser ajustados para otimizar a performance de um subconjunto (conjunto de validação) do conjunto de treinamento.

6. **Validar a precisão da função de aprendizagem.** Isso é feito por meio de ajuste de parâmetros e aprendizado do algoritmo.

Há diversos algoritmos de aprendizado supervisionado, com o propósito de prever classificações, cada um possui suas vantagens e desvantagens. Não há um algoritmo que funcione melhor em todos os problemas de aprendizado (ou seja, não há um algoritmo 'bala de prata' ou *silver bullet*), mas sim um melhor algoritmo para um determinado tipo de problema.

Neste trabalho, foram usados diversos algoritmos de classificação populares para lidar com o nosso problema de classificação de dados, de forma que pudéssemos implementar o nosso sistema, e também, comparar a performance entre eles. Os algoritmos são: Support Vector Machine, K-Nearest Neighbor e Random Forest. Nas próximas sessões, todas as tecnologias usadas na implementação do software serão brevemente descritas, de forma que possámos ter um entendimento melhor de como se deu essa construção.

2.2.1 Support Vector Machine (SVM)

São modelos de aprendizado associados com algoritmos de aprendizagem de máquina, que permitem analisar dados e reconhecer padrões. O SVM costuma ser usado para classificação de dados a partir de um conjunto de exemplos que é usado para o treinamento do algoritmo, tendo cada exemplo uma categoria pré-associada. O dado que queremos classificar é classificado então a partir de exemplos anteriores que usamos para treinar o algoritmo.

Ao adicionarmos exemplos para o treinamento do algoritmo, eles são mapeados no mesmo espaço da categoria ao qual ele pertence. Por exemplo, se temos três categorias de espécies: Anelídeos, Mamíferos e Protozoários, e adicionarmos um Cavalo (já pré-associado por nós como Mamífero) como dado de exemplo para treinamento da rede, o SVM alocará esse cavalo no espaço de Mamíferos existente no algoritmo, de forma que ele consiga organizar a sua própria informação.

Essa é uma forma de organização para classificação e previsão linear, em adição a essa metodologia, o SVM pode performar classificações não lineares usando funções *kernel*, mapeando novos dados em espaços ultra-dimensionais.

As funções *kernels* são uma classe de algoritmos para análise de padrões que podem ser combinados com o SVM, elas facilitam a separação dos dados, podendo ser definidas como funções de similaridade. [16] Com o uso dessas funções, podemos encontrar relações em tipos de dados aparentemente diferenciados, como documentos de texto, conjuntos, vetores,

imagens, gráficos e etc.

A característica principal das funções *kernel*, entretanto, é a sua forma distinta de visualizar o problema. Funções *kernel* mapeiam os dados em espaços hiper dimensionais (*Hiperdimensional Spaces*), na esperança de que nesses espaços, os dados se mostrem melhor estruturados do ponto de vista da análise de padrões, dividindo-os por grau de similaridade encontrada.

Escolher o *kernel* mais apropriado para o algoritmo de classificação SVM, em questões de eficiência e desempenho, depende muito do problema que temos em mãos, podendo facilmente tornar-se uma tarefa tediosa e demorada. A escolha de um *kernel* depende diretamente do problema que está sendo modelado. As funções *kernel* que serão testadas com o algoritmo SVM neste trabalho são: Linear, Polinomial, *Radial Basis Function* (RBF) e Sigmóide.

2.2.2 K-Nearest Neighbor (k-NN)

Assim como o SVM, também são modelos de aprendizagem para classificação de dados. O princípio básico do k-NN é encontrar uma porção de exemplos de dados de treinamento que seja o mais próximo possível em distância, do novo ponto, e prever a classe associada a ele a partir dessa redução de escopo (daí o nome *Nearest Neighbor*, que vem de 'vizinho mais próximo').

Apesar de sua aparente simplicidade, a utilização do k-NN tem sido bem sucedida em um grande número de problemas de classificação, incluindo análise de dígitos escritos à mão ou de imagens de satélites [9].

2.2.3 Random Forest

O *Random Forest* é um modelo de aprendizagem para classificação e regressão de dados alternativo ao SVM e k-NN, operando na construção de árvores de decisão em tempo de treinamento e produzindo uma classe que é o *mod* das classes produzidas por árvores individuais pré-divididas pelo algoritmo. O pseudo-código que intuiu o algoritmo final do *Random Forest* foi desenvolvido por Leo Breinan e Adele Cutler [10].

O desenvolvimento do algoritmo de *Random Forests* também foi influenciado pelo trabalho de Amit e Geman [11], que introduziram a ideia de busca por meio de um conjunto randômico

de decisões possíveis, fatiando um nódo de árvore no contexto do crescimento de uma árvore unificada. Os trabalhos mais recentes acerca do algoritmo foram desenvolvidos pela Microsoft Research [2], que vem extendendo o trabalho produzido por Breiman.

2.2.4 Ruby on Rails

O Ruby on Rails (frequentemente abreviado como RoR) é o *framework* para desenvolvimento de aplicações (*web* utilizado para a implementação do protótipo deste trabalho. Trata-se de um *framework* de código aberto que tem como principal destaque sua promessa de aumentar a velocidade e facilidade no desenvolvimento de aplicações *web* orientadas a bancos de dados (*database-driven web sites*), utilizando a arquitetura *Model-View-Controller* (MVC). O uso do RoR tem sido considerada a forma mais produtiva, eficiente e barata de produzir aplicações *web*, tendo sido reportados pelo Business Insider, inúmeros cases de sucesso associados ao uso do RoR em grandes corporações, com excelentes resultados. Dentre essas corporações, estão: Amazon, BBC, Cisco, IBM, JP Morgan, NASA e Yahoo!. [25]

O RoR utiliza a linguagem de programação Ruby, que foi criada na década de 90 pelo jovem programador japonês Yukihiro Matsumoto. De acordo com seu autor, ela foi influenciada por Perl, Smalltalk, Ada, Eiffel e Lisp. Vários são seus paradigmas suportados: funcional, orientado a objetos, dentre outros. Entre outras características relevantes do Ruby, estão o fato de ser uma linguagem dinâmica e com suporte ao gerenciamento de memória automático, muito comum também em linguagens de *script* como Python.

Matsumoto resume a filosofia da linguagem em uma simples frase, publicada por ele em uma lista de discussão muito popular acerca de Ruby[29]:

> *"simples em aparência, porém, muito complexa internamente, assim como o corpo humano"*

O Crescimento da Linguagem Ruby

Desde seu lançamento ao público, em 1995, Ruby conseguiu conquistar vários admiradores ao redor do mundo. Em 2006, Ruby conseguiu chegar no seu estopim de aceitação entre os programadores, com vários grupos de usuários formados nas maiores cidades do mundo e de

conferências relacionadas ao seu uso.

Ruby-Talk, a primeira lista de discussão da linguagem de programação Ruby, subiu para uma média de 200 mensagens por dia em 2006. A lista caiu recentemente devido á divisão da comunidade em vários grupos menores, seu crescimento pode ser acompanhado na figura 2.1. [28]

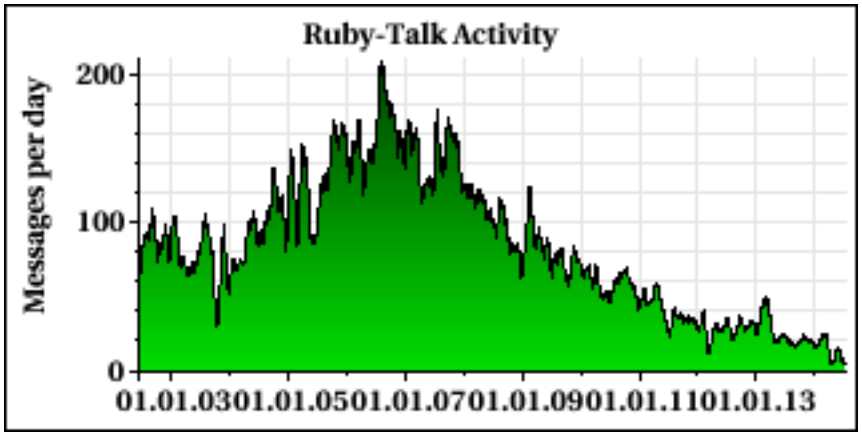

Figure 2.1: Gráfico de popularidade da linguagem Ruby, a partir de mensagens por dia na lista de discussão Ruby-Talk, de acordo com o tempo.

Ruby está rankeado entre as linguagens de programação com maiores índices de popularidade no mundo pelo índice TIOBE. [24] Muito de seu crescimento é particularmente associado ao sucesso e grande *user adoption* do *framework web* Ruby on Rails.

2.2.5 PostgreSQL

O PostgreSQL é um banco de dados de código aberto considerado exemplar para a especificação SQL, por ser extremamente aderente a esse padrão. [5] O projeto atual surgiu em 1995, e tem como concorrente principal o MySQL, por também ser um banco de dados software livre.

O desempenho do PostgreSQL é similar ao do MySQL, entretanto, o PostgreSQL é mais adequado quando estamos lindando com *hardware* multiprocessado, devido à suíte de funcionalidade que ele disponibiliza para que possamos lidar com esse ambiente de desenvolvimento.

2.2.6 Twitter Bootstrap

O Bootstrap foi desenvolvido pelos desenvolvedores Mark Otto e Jacob Thorton, no período em que trabalhavam na rede social e microblog *microblog* Twitter. Ele foi desenvolvido para encorajar consistência de páginas *web* entre diferentes *browsers*. [13]

Com o Bootstrap é possível termos uma única tela implementada (base de código única), que se adapte ao tamanho de todos os tipos de dispositivos no qual ela precisará ser exibida, de *smartphones*, a *tablets* ou *desktops*, simplificando assim, o processo de desenvolvimento de uma aplicação *web* responsiva. [14]

2.2.7 Bibliotecas libsvm-ruby-swig e scikit-learn

A biblioteca libsvm-ruby-swig é uma implementação em Ruby do algoritmo de classificação *Support Vector Machine* (SVM), tendo sido usada neste presente trabalho. O algoritmo foi implementado por Tom Zeng da Tomz Consulting [26] e distribuído na forma dessa biblioteca, que permite uma interação com ele por meio de uma interface programática. O libsvm é usado atualmente no `http://www.tweetsentiments.com`, uma aplicação *web* para análise de sentimentos no Twitter que utiliza algoritmo de classificação SVM.

Scikit-learn é uma interface Python para vários algoritmos de classificação, sendo uma biblioteca que reúne implementações de vários deles, incluindo o *Nearest Neighboor* e o *Random Forest*. Essa biblioteca foi usada nesse trabalho para comparar o desempenho do SVM com estes algoritmos [6].

Capítulo 3

Metodologia

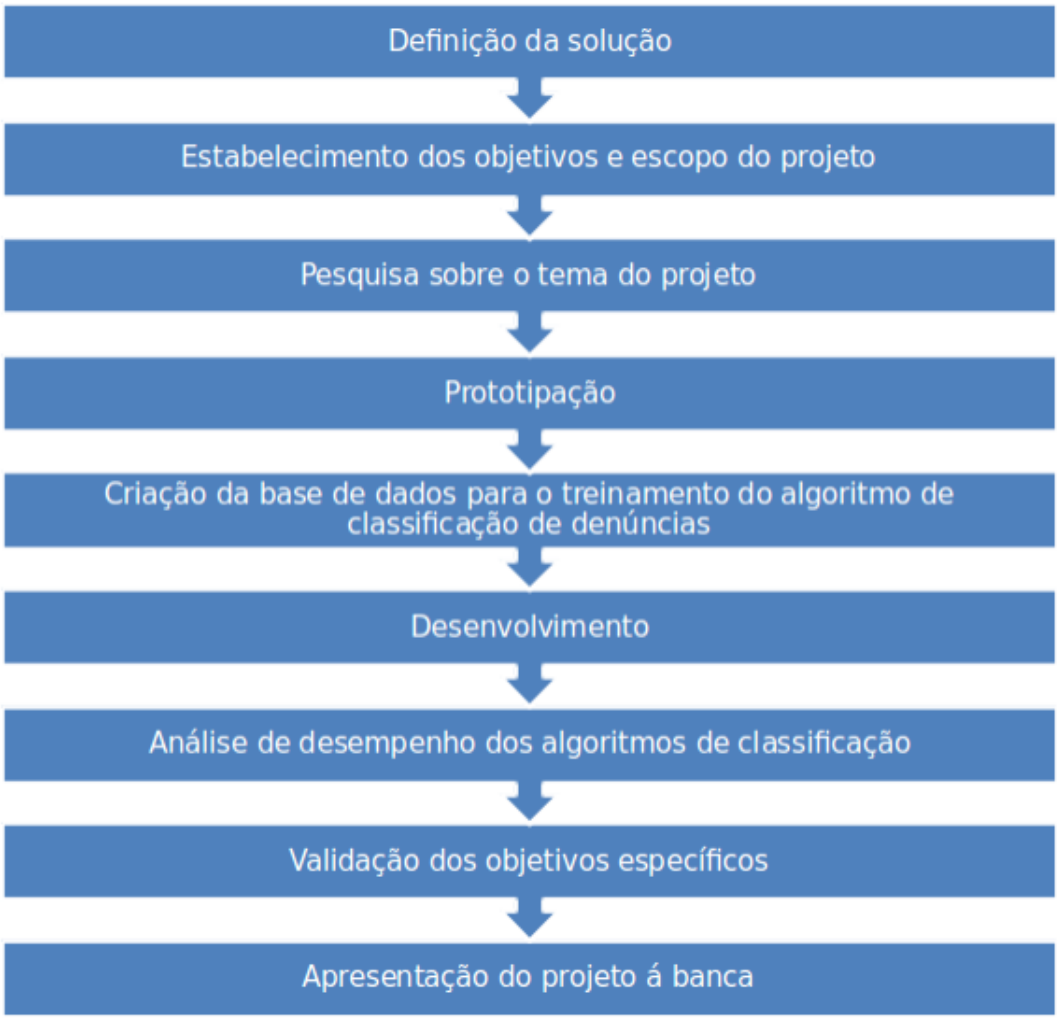

Figure 3.1: Fluxograma mostrando a sequência de etapas realizadas para o desenvolvimento deste trabalho.

O fluxograma ilutrado na figura 3.1, mostra as etapas de desenvolvimento deste trabalho, desde a definição da solução até a implementação da aplicação, passando pela validação dos resultados e apresentação do produto resultante.

As etapas de desenvolvimento do projeto serão explanadas, didaticamente, e em detalhes, nos tópicos que se seguem, de forma que o experimento possa ser replicado com facilidade.

3.1 Etapa 1: Definição da Solução

A solução do software para denúncia de situações discriminatórias foi idealizada em um evento denominado "Programação Insubordinada" (ver figura 3.2), que ocorreu durante os dias 5 e 6 de Abril, no escritório da empresa *ThoughtWorks* em Recife. O Programação Insubordinada foi um treinamento para pessoas envolvidas com tecnologia que possuíssem o desejo de aprender disciplinas avançadas da computação, aplicadas para construir software de qualidade com o objetivo de mudar os paradigmas da sociedade. Foi proposto aos participantes do evento, estudantes da área de computação, a seguinte situação-problema:

> *"em uma sociedade de grandes injustiças e problemas sociais, como a nossa, de que forma podemos, através do software, produzir um impacto social positivo?"*

A partir dessa problemática, foram coletadas ideias de software com o objetivo de produzir impacto social positivo, por meio de um *Brainstorm*.

Brainstorm (ou *Brainstorming*) é um processo de geração de ideias que explora a capacidade criativa das pessoas, por meio de produção massiva dessas ideias em um curto espaço de tempo, para posterior avaliação. As ideias formuladas pelos participantes eram resumidas em uma breve frase, essa frase era escrita em um *post-it* que era colado na parede, junto com outras ideias.

Após essa etapa de formulação, os participantes do *Brainstorm* votavam nas ideias que mais lhes simpatizassem. A ideia do *software* para denúncia de situações discriminatórias com o uso de algoritmos de classificação para melhorar a experiência de usuário, diminuindo o número de campos que ele precisa preencher para formular a denúncia, tema desse trabalho, foi a mais votada dentre as ideias mostradas.

Figure 3.2: Evento “Programação Insubordinada”, no escritório da *ThoughtWorks* em Recife.

3.2 Etapa 2: Estabelecimento dos objetivos e escopo do projeto

Com base na definição da solução, no evento Programação Insubordinada (Etapa 1 da Metodologia), os objetivos do trabalho foram definidos na seção de Objetivos, dentro do capítulo Introdução.

3.3 Etapa 3: Pesquisa sobre o tema do projeto

Durante esta etapa foram realizadas atividades de estudo sobre a desigualdade social e seus desdobramentos, seu impacto na sociedade, assim como o conceito de justiça social, que é o contra-atuante a esse problema. Foi também estudado o que tem sido feito em software com relação ao problema social da desigualdade.

Foram buscadas informações sobre o tema do projeto em fontes diversas. A maior parte da pesquisa tem base em referências seguras na internet, como o Mashable [30], para obser-

varmos o que tem sido feito em tecnologia para produzir impacto social. O objetivo dessas pesquisas foi permitir uma melhor compreensão acerca do tema, como forma de facilitar o desenvolvimento do tema do projeto.

Dentre algumas iniciativas populares em softwares desse segmento podemos citar:

Open MRS: É um sistema de diagnósticos online que permite armazenar dados de um paciente e compartilhar ao redor do mundo, facilitando o acesso a médicos de qualquer lugar do mundo, em especial, áreas menos favorecidas em relação a serviços de saúde.

Keep America Beautiful: Com cada foto que o usuário compartilha, Johnson & Johnson doa US$ 1 a uma causa de sua escolha.

Feedie: Ao consumir alimentos e tirar uma foto do mesmo antes do consumo em um dos restaurantes participantes, uma refeição é doada a alunos de escolas na África.

Doe Sua Energia: Numa iniciativa em Pernambuco, a agência Ampla, numa iniciativa encabeçada pelas Baterias Moura, desenvolveu um aplicativo (denominado Doe Sua Energia) que mede a distância que o usuário percorre em suas corridas diárias. A cada quilômetro corrido, R$1 é doado para o Lar do Nenem e a Casa de Maria, ambas instituições de caridade da Cidade do Recife. O aplicativo deixou de operar concomitantemente ao encerramento da ação publicitária, foram arrecadados R$ 62.279.

3.4 Etapa 4: Prototipação

Antes da implementação da aplicação, foi feita uma prototipação preliminar de suas telas em papel.

A prototipação em papel ajuda a entender melhor o propósito do software e ajuda-nos a visualizá-lo mais facilmente, fazendo com que identifiquemos uma melhor forma de implementar o produto graficamente, respeitando a preocupação com a usabilidade pelo usuário, sem que se precise implementar as telas, via código, diminuindo a velocidade do processo de prototipação. A figura 3.3 mostra a prototipação em papel da aplicação deste trabalho.

Quanto ao nome da aplicação, houve uma preocupação no sentido de termos um nome, que por meio de sua leitura, seja facilmente identificável o propósito do software, e sua

funcionalidade básica, apenas por meio dele. O nome escolhido foi "Boca No Trombone". Tal nome foi escolhido por remeter facilmente a algo que se usa para dinfundir alguma coisa, se apropriando do ditado popular *"colocar a boca no trombone"*.

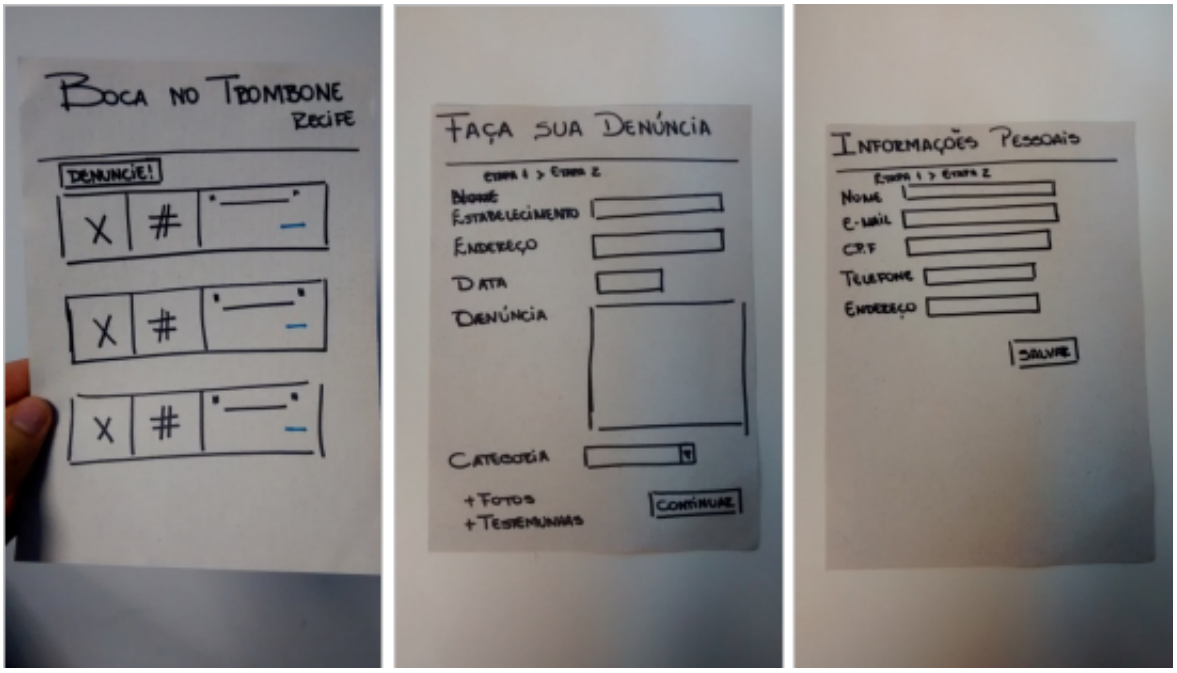

Figure 3.3: Prototipação da aplicação, mostrando algumas de suas telas planejadas.

3.5 Etapa 5: Criação da base de dados para treinamento do algoritmo de classificação de denúncias

Primeiramente, foi feita uma divisão, entre os tipos possíveis de denúncia que se enquadram no desrespeito aos direitos fundamentais, advindos da desigualdade social. Observaram-se 11 tipos de desigualdades catalogadas na literatura, nas quais as denúncias podem ser enquadradas.

- Gênero (mulheres)
- Gênero (homosexualidade)
- Racial (cor da pele)
- Racial (xenofobia)

- Idade
- Classe social
- Assistência médica
- Educação
- Digital
- Castas

3.5.1 Definição de exemplos de palavras-chave

Foram coletadas 128 palavras-chave que representassem palavras comumente encontradas em denúncias dos mais variados tipos, sendo elas palavras ofensivas ou não (é comum encontrar simplesmente a palavra "hospitais" ao encontrar denúncias de condições precárias de hospitais). Cada palavra foi associada a uma "classe", essa classe é uma classificação, que representa o tipo da denúncia a que essa palavra-chave normalmente está enquadrada, a tabela 3.1 mostra um exemplo dessa associação.

Palavra-chave	**Tipo de discriminação (classe)**
"escurinho"	Racial (cor da pele)

Table 3.1: Exemplo de relação palavra-chave para tipo de discriminação (classe)

Esse tipo de coleta foi necessária para treinar o algoritmo de classificação, de forma que ele consiga prever a categoria das denúncias recebidas, a partir dessa base de dados de 128 palavras que possibilitou seu aprendizado.

Na tabela 3.2, foi separada uma lista dos tipos de palavras catalogadas, com dois exemplos de cada. A classe "Neutro" categoriza palavras que por si só não se enquadram em um tipo de denúncia específica, por serem palavras genéricas. Elas só ganham uma classe quando associada com outra palavra já associada a uma outra classe (ganhando a classe dessa outra palavra).

Tipo de discriminação (classe)	Exemplos de palavras-chave
Gênero (mulheres)	“vadia”, “puta”
Gênero (homosexualidade)	“afeminado”, “maricona”
Racial (cor da pele)	“escurinho”, “macaco”
Idade	“velho”, “gagá”
Classe Social	“favelado”, “pobre”
Assistência Médica	“hospitais”, “fila”
Educação	“professor”, “aula”
Digital	“notebook”, “internet”
Castas	“evangélico”, “dalit”
Neutro	“nojento”, “repugnante”

Table 3.2: Palavras-chave comumente encontradas em denúncias de situações de discriminação.

3.5.2 Definição de exemplos de combinações de palavras-chave associadas a uma categoria

Foram cruzadas 128 palavras em 24.870 combinações, sendo cada combinação associada a uma classe. Essas combinações, usadas para treinar os algoritmos de classificação, são denominadas “*samples*”. As palavras-chave são relacionadas pelo seu número de identificação na tabela referente no banco de dados (atributo *id*, uma chave-primária encontrada em cada um deles).

Essa é a massa de dados que de fato é usada para treinar o algoritmo de classificação. As combinações foram feitas, ou por cruzamento das palavras neutras com as palavras classificadas (resultando na classe das classificadas) ou por alguns cruzamentos pensados e escritos “à mão”, com inteligência humana. Segue exemplo de combinação na tabela a seguir (3.3).

ID de uma palavra-chave	ID de outra palavra-chave	Classe
90	124	Discriminação de Castas

Table 3.3: Exemplo de combinação de palavras-chave, resultando em um tipo de discriminação (classe).

Quanto maior a massa de dados utilizada no treinamento, junto com a qualidade dos dados, maior será a precisão do algoritmo de classificação em suas previsões. Tal qual a combinação de duas palavras relacionando a uma classe, o algoritmo irá buscar nos textos das denúncias recebidas por duas palavras, fazendo uma mineração baseada em palavras chave, e irá relacioná-la a uma classe por meio de seus processos estatísticos de previsão.

3.5.3 Desenvolvimento

De posse das desigualdades sociais elencadas em 11 categorias possíveis de denúncias, e com a base de dados de palavras e suas combinação já formada para o treinamento do algoritmo de aprendizagem, o sistema foi implementado utilizando as tecnologias mostradas na tabela 3.4.

Tipo de tecnologia	**Nome da tecnologia**	**Mais informações**
Linguagem de programação	Ruby 2.0.x	`https://www.ruby-lang.org/en`
Framework web	Ruby on Rails 4.x	`http://rubyonrails.org/`
Biblioteca CSS	Bootstrap 3.2.0	`http://getbootstrap.com/`
Gerenciador de Banco de Dados	PostgreSQL 9.4	`http://www.postgresql.org/`
Algoritmo de Classificação SVM	libsvm-ruby-swig	`https://rubygems.org/`

Table 3.4: Tecnologias usadas na implementação da aplicação.

3.5.4 Análise de desempenho dos Algoritmos de Classificação

O projeto foi implementado usando o algoritmo de classificação SVM para categorização das denúncias recebidas, por meio de previsão. O algoritmo foi treinado com as 24.870 combinações das 128 palavras-chave.

Além desse algoritmo, foram também treinados os algoritmos de classificação *k-NN* e *Random Forest*, de forma que pudéssemos comparar o seu desempenho com o do SVM, referente ao tempo necessário para treinamento e previsão. A comparação foi feita usando 4 *kernels* SVM diferentes.

Capítulo 4

Cronograma

	Abril	**Maio**	**Junho**	**Julho**	**Agosto**
Pesquisa inicial sobre o tema	X	-	-	-	-
Implementação do *software*	-	X	X	-	-
Escrita e entrega do relatório	-	-	-	X	-
Apresentação do trabalho	-	-	-	-	X

Table 4.1: Cronograma ilustrando o desenvolvimento das atividades deste trabalho, em função do tempo.

Capítulo 5

Resultados Obtidos

5.1 Conhecimento gerado em torno de algoritmos de classificação utilizando aprendizagem de máquina

Foi adquirido expertise em torno de algoritmos de classificação, através de estudos aplicados. Inicialmente dedicou-se á leitura sobre o tópico "Aprendizagem de Máquina", de forma a identificar a melhor solução para o problema encontrado na aplicação, para que posteriormente os algoritmos de classificação fossem vistos como a tecnologia mais adequada para a solução. Os algoritmos se mostraram a solução mais adequada, devido às suas propriedades, que permitem prevermos classificações a partir de palavras-chave encontradas em um texto (*Text-Based Mining*).

Foram procuradas bibliotecas Ruby e Python para auxiliar no uso desses algoritmos. A biblioteca Ruby "libsvm-ruby-swig" foi usada para implementar a aplicação "Boca no Trombone" em si, pois esta foi desenvolvida utilizando Ruby, e precisava-se de uma implementação nesta linguagem do algoritmo de classificação SVM. A "libsvm-ruby-swig" proporcionou isso.

Para desenvolver a comparação entre o SVM e os demais algoritmos de classificação (k-NN e *Random Forest*), foi utilizada a linguagem Python, devido ao fato de termos encontrado uma biblioteca que possui a implementação de todos esses algoritmos de classificação numa só biblioteca, chamada "scikit-learn", facilitando essa etapa do projeto.

O processo de previsão por algoritmos de classificação, foi então, aplicado em um projeto

real, tema deste trabalho (aplicação *web* "Boca no Trombone"), utilizando uma grande massa de dados para treinamento do algoritmo. A partir desse processo, foi gerado conhecimento prático em torno de sua utilização, para o concluinte do curso de Bacharelado em Sistemas de Informação.

5.2 Aplicação desenvolvida

A aplicação foi desenvolvida utilizando as tecnologias decritas na metodologia deste trabalho. Seguindo os esboços realizados na etapa de prototipação, foram implementadas quatro telas, cujas ilustrações estão nas figuras a seguir (5.1, 5.2, 5.3 e 5.4).

Figure 5.1: Tela inicial.

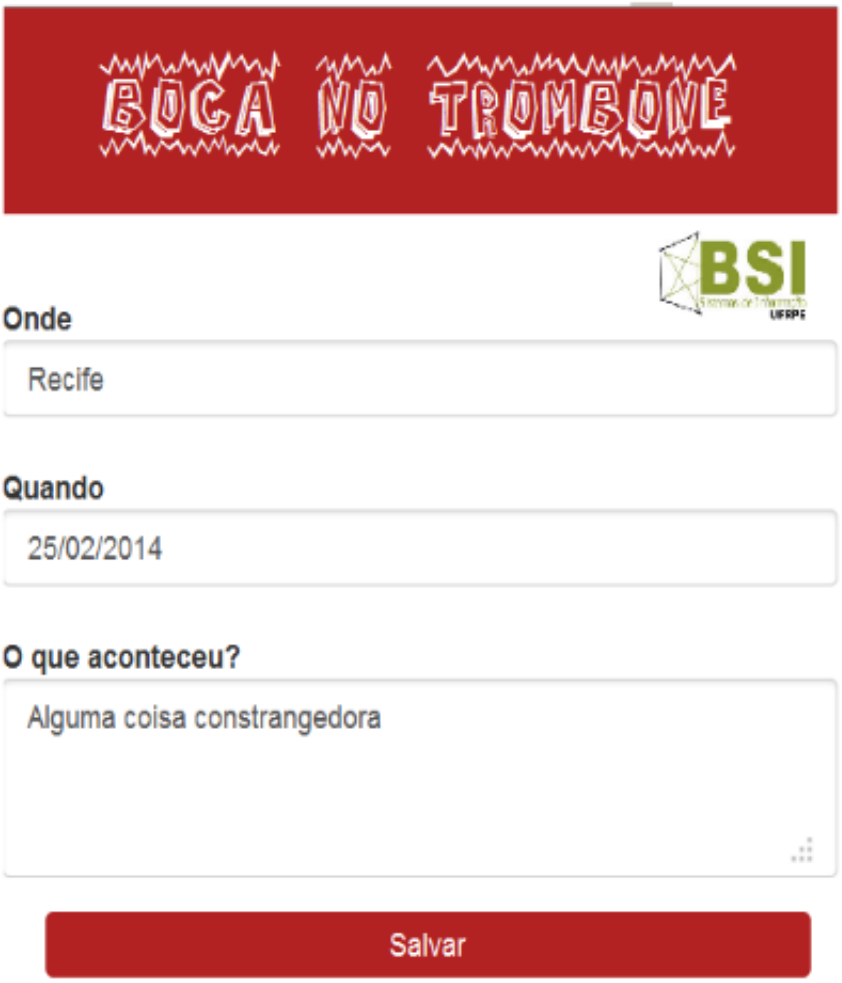

Figure 5.2: Tela de preenchimento de dados da denúncia.

Figure 5.3: Tela de preenchimento de dados do denunciante (dados pessoais).

Figure 5.4: Tela de resultados.

5.3 Análise comparativa de performance dos algoritmos de classificação

Foi feita uma comparação quanto ao tempo de treinamento levado pelos algoritmos de classificação SVM (com kernels Linear, Polinomial, RBF e Sigmóide), Random Forest e k-NN para carregar os dados e treinar a si próprios com os dados passados (24.480 combinações de palavras-chave, já previamente associadas a uma classe).

Posteriormente, foi feita uma comparação quanto á precisão da previsão, ou seja, a taxa de acerto de cada algoritmo quanto à corretude da previsão. Para isso, foi usado um conjunto de exemplos de combinações de palavras-chave encontradas em algumas denúncias.

5.3.1 Etapa 1: Tempo levado por cada algoritmo para treinamento a partir da massa de dados

Nessa etapa, o estudo compara o tempo que cada algoritmo levou para ler a base de dados de 24.870 exemplos previamente classificados e treinar o próprio algoritmo, de forma que a partir da análise e posterior percepção de padrões a partir desses dados, ele possa prever futuras classificações.

Seguem na tabela 5.1 os resultados dessa etapa. Os tempos observados foram arredondados para a terceira casa decimal dos milésimos de segundo.

Algoritmo de Classificação	**Kernel**	**Tempo de leitura e treinamento**
Support Vector Machine	Linear	28.875 segundos
Support Vector Machine	Polinomial	1973.653 segundos
Support Vector Machine	*Radial Basis Function* (RBF)	93.450 segundos
Support Vector Machine	Sigmóide	23.226 segundos
K-Nearest Neighbor	Não utiliza *kernel*	0.118 segundos
Random Forest	Não utiliza *kernel*	0.798 segundos

Table 5.1: Tempo necessário para o treinamento dos algoritmos de classificação.

5.3.2 Etapa 2: Precisão na previsão das classes

Nesta etapa será calculada a taxa de acertabilidade das previsões de cada algoritmo. O processo utilizado para medir esse índice está descrito a seguir.

1. Foi feita a seleção de 30 denúncias de situações de discriminação, encontradas em redes sociais. Essas denúncias costumam ser publicadas em redes como Facebook, em sua maioria, por usuários que sofreram algum tipo de discriminação em algum lugar, e usam a rede como um meio de desabafo para os seus amigos, na intenção de que a mensagem se espalhe o tanto quanto possível. Há dois pré-requisitos para a coleta de uma denúncia:

   - A denúncia deve conter, no mínimo, duas palavras-chave comumente encontradas em denúncias de discriminação ou situação de desigualdade social em geral.
   - A denúncia deve configurar algum tipo de discriminação de qualquer espécie, ou situação de desigualdade social em geral.

2. Tendo sido feita essa seleção de 30 denúncias, foram extraídas duas palavras-chave das denúncias selecionadas, comumente encontradas em denúncias de discriminação ou situação de desigualdade social. Ao final desse passo temos 30 pares de palavras-chave.

3. Com 30 pares de palavras-chave em mãos, esses pares são fornecidos aos algoritmos SVM (variando o *kernel* utilizado), k-NN e Random Forest, para que eles tentem acertar de que tipo de denúncia de discriminação/situação discriminatória vieram os pares de palavras-chave.

4. Para cada um dos algoritmos, foi calculada a taxa de acerto dos tipos de denúncias dos quais se originaram as palavras-chave. Por exemplo, obtendo-se um índice de precisão de 50%, significa que o algoritmo conseguiu acertar a classificação correta de metade de toda a amostra exemplo de 30 pares.

5.3.3 Etapa 3: Conclusões

Todos os objetivos específicos definidos neste trabalho foram cumpridos, desde o uso dos algoritmos de classificação, á implementação da aplicação. E posterior análise algorítmica

Algoritmo de Classificação	**Kernel**	**Precisão das previsões**
Support Vector Machine	Linear	55%
Support Vector Machine	Polinomial	60%
Support Vector Machine	*Radial Basis Function* (RBF)	65%
Support Vector Machine	Sigmóide	40%
K-Nearest Neighbor	Não utiliza *kernel*	60%
Random Forest	Não utiliza *kernel*	70%

Table 5.2: Precisão de previsão dos algoritmos.

de eficácia e desempenho entre eles.

Nossa análise comparou inicialmente o tempo de treinamento levado por cada algoritmo. Foi notada uma diferença relevante entre o tempo necessário para treinar os algoritmos k-NN e Random Forest, e o mesmo observado para treiná-lo utilizando o SVM, independente do kernel. O algoritmo de classificação, leva mais do que 22 segundos para ser treinado com a massa de dados de 24.870 exemplos previamente classificados. Ao passo que, os algoritmos k-NN e Random Forest levam em torno de menos de 1 segundo. Sendo assim, o SVM leva no mínimo, em torno de 22 vezes mais tempo para ser treinado do que os demais algoritmos.

A partir do trabalho de Colas [17], foi encontrada uma explicação para este fenômeno. Antes de ser chegada essa conclusão, é necessário que vejamos alguns fundamentos a respeito desses algoritmos, delineando algumas peculiaridades.

Peculiaridades observadas entre SVM, k-NN e Random Forest, quanto á forma de lidar com problemas de classificação

O SVM é um algoritmo projetado para lidar com problemas de classificação binária [3]. Entenda-se problemas de classificação binária aqueles nos quais só é possível escolher entre dois rótulos. Exemplo: Sim ou Não, Dia ou Noite, Ligado ou Desligado, etc, como ilustra a figura 5.5.

Sendo assim, o SVM é bastante adequado quando temos apenas duas classificações/classes possíveis.

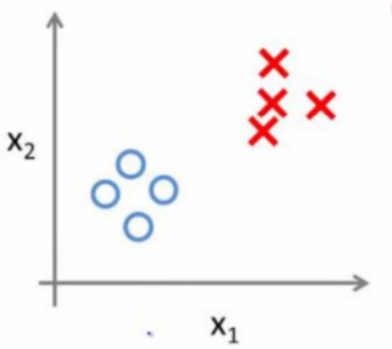

Figure 5.5: Exemplo ilustrativo de um problema de classificação binária.

Por outro lado, os algoritmos de classificação Random Forest e k-NN foram projetados para lidar com problemas de classificação multiclasse. Problemas de classificação multiclasse envolvem a possibilidade de escolher de 1...n classes possíveis, sem um limíte específico. Ocorre quando temos mais de duas classificações possíveis. Exemplo: Mamífero, Ovíparo, Réptil, Inseto (problema de taxonomia), como ilustra a figura 5.6.

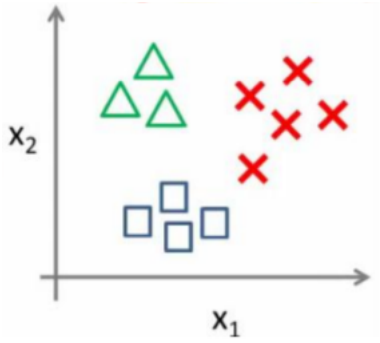

Figure 5.6: Exemplo ilustrativo de um problema de classificação multiclasse.

Ao lidar com problemas de classificação que envolvem mais de duas classes (multiclasse), o SVM, que é um algoritmo de classificação binária por padrão, apela para a estratégia "*One Versus All*" [17]. Nesta estratégia de classificação existem tantos classificadores binários quanto classes envolvidas no problema. Existe portanto, nessa estratégia, um classificador para cada classe. Quando usamos os algoritmos k-NN ou Random Forest para lidar com o mesmo tipo de problema, percebemos que por eles serem classificadores, por padrão, projetados para lidar com problemas que envolvem a classificação de mais de duas classes, é usado apenas um classificador, como ilustra a figura 5.7.

Portanto, ao executar experiências em tarefas de classificação envolvendo mais de duas classes, na verdade estamos comparando n classificadores SVM (para n classes) a um único

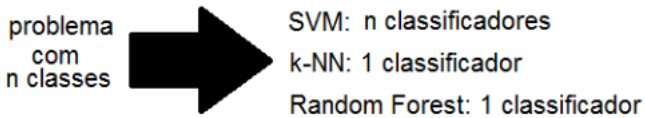

Figure 5.7: Classificadores existentes em um problema com n classes possíveis, dependendo do algoritmo de classificação utilizado.

classificador multiclasse Random Forest ou k-NN. Colas [17] considera esse tipo de comparação injusta em termos de desempenho, visto que o SVM será invariavelmente prejudicado ao lidar com problemas multiclasse.

5.3.4 Diferenças de precisão observadas

Na segunda análise feita na comparação entre os algoritmos, é coletada a precisão entre eles. Como visto na tabela 5.2, notou-se uma superioridade do Random Forest, com uma precisão de 70% em suas previsões, mas ainda seguido de perto do SVM com *kernel* RBF, que obteve 65%.

Não foi possível afirmar com certeza o motivo de esses dois algoritmos terem tido um desempenho superior. Entretanto, podemos cogitar várias possibilidades, todas elas ligadas á quantidade e/ou qualidade da informação usada para treinar os algoritmos, que podem ser motivo de estudo para trabalhos futuros:

- As palavras-chave definidas não foram suficientes, devido ao seu número em quantidade (apenas 128 palavras-chave).
- A combinações de palavras-chave usadas para treinar o algoritmo não foram suficientes, devido ao seu número em quantidade (apenas 24.870 combinações).
- As palavras coletadas e/ou combinações coletadas podem não representar as palavras e/ou combinações mais encontradas em denúncias de discriminação (qualidade da informação precária).

Bibliografia

[1] Baranauskas J. A. Aprendizado de máquina: Conceitos e definições. USP Departamento de Física e Matemática (FFCLRP-USP), 2012. Acessado em: 01 de Agosto de 2014. Disponível em: `http://dcm.ffclrp.usp.br/~augusto/teaching/ami/AM-I-Conceitos-Definicoes.pdf`.

[2] J. Shotton A. Criminisi and E. Konukoglu. Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. *Foundations and Trends® in Computer Graphics and Vision*, 7, Issue 2-3:5–32, 2012.

[3] Cortes C. and Vapnik V. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

[4] Dearborn. Adc-michigan introduces civil rights mobile app to commemorate 50th anniversary of civil rights act. 11 de Abril de 2014. Acessado em: 01 de Agosto de 2014. Disponível em: `http://bit.ly/XDi6v8`.

[5] Manga F. O que é o postgresql? 4Linux Open Software Specialists, 2014. Acessado em: 01 de Agosto de 2014. Disponível em: `http://www.4linux.com.br/postgresql/o-que-e-postgresql`.

[6] Alexandre G. Vincent M Fabian P., Gael V. and Bertrand T. Scikit-learn: Machine learning in python. *Journal of Machine Learning*, 12:2825–2830, 2011.

[7] Mattew H. 10 common ui/ux mistakes and how to avoid them. Fearless Flyer, Junho de 2014. Acessado em: 01 de Agosto de 2014. Disponível em: `http://fearlessflyer.com/10-common-ui-mistakes-and-how-to-avoid-them`.

[8] Bigus J. *Data mining with neural networks: solving business problems-from application development to decision support*. McGraw-Hill, New Jersey, 1996.

[9] Vanderplas J. Nearest neighbors. Sklearn lib, 2014. Acessado em: 01 de Agosto de 2014. Disponível em: `http://scikit-learn.org/stable/modules/neighbors.html`.

[10] Breiman L. Random forests. *Springer*, 45, Issue 1:5–32, 2001.

[11] Breiman L. Shape quantization and recognition with randomized trees. *Springer*, 9, No. 7:5–32, 2001.

[12] J. R. Kluegel L. Cohen and C. Kenneth. Social inequality and predatoy criminal victimization: An exposition and test of a formal theory. *America Sociological Review*, 46:505–524, 1981.

[13] Otto M. Bootstrap in a list apart [342]. 17 de Janeiro de 2012. Acessado em: 01 de Agosto de 2014. Disponível em: `http://www.markdotto.com/2012/01/17/bootstrap-in-a-list-apart-342`.

[14] Otto M. and Thornton J. Get bootstrap. Twitter, inc., 2014. Acessado em: 01 de Agosto de 2014. Disponível em: `http://getbootstrap.com`.

[15] Shapiro T. M. *The Hidden Cost of Being African American*. Universidade de Oxford, 2004.

[16] Garg N. What are kernels in machine learning. Quora, Agosto de 2012. Acessado em: 01 de Agosto de 2014. Disponível em: `http://www.quora.com/Machine-Learning/What-are-Kernels-in-Machine-Learning-and-SVM`.

[17] Colas F. P. R. Data mining scenarios for the discovery of subtypes and the comparison of algorithms. Leiden Institute of Advanced Computer Science (LIACS), Faculty of Science, Leiden University, 4 de Março de 2009. Acessado em: 01 de Agosto de 2014. Disponível em: `https://openaccess.leidenuniv.nl/handle/1887/13575`.

[18] Kumar R. Google announces winners of its global impact challenge in india. DNA Webdesk, 31 de Outubro de 2013. Acessado em: 01 de Agosto de 2014. Disponível em: `http://www.dnaindia.com/scitech/report-google-announces-winners-of-its-global-impact-challenge-in-india-1911920`.

[19] Schapire R. Machine learning algorithms for classification. Princeton University, 10 de Março de 2012. Acessado em: 01 de Agosto de 2014. Disponível em: http://www.cs.princeton.edu/~schapire/talks/picasso-minicourse.pdf.

[20] Vishy R. Your vision. our software. world-changing. introducing thoughtworks social impact program. echsangam, 29 de Agosto de 2012. Acessado em: 01 de Agosto de 2014. Disponível em: http://www.thoughtworks.com/social-impact.

[21] S. Russell and P. Norving. *Inteligência Artificial*. Elsevier, Rio de Janeiro, 2004.

[22] Meira S. A ubiquidade, as vantagens e o custo do software. Terra Magazine, 18 de Agosto de 2009. Acessado em: 01 de Agosto de 2014. Disponível em: http://terramagazine.terra.com.br/SilvioMeira/blog/2009/08/18/a-ubiquidade-as-vantagens-e-o-custo-do-software.

[23] Rugaber C. S and Boak J. Wealth gap: A guide to what it is, why it matters. AP News, 27 de Janeiro de 2014. Acessado em: 01 de Agosto de 2014. Disponível em: http://apnews.excite.com/article/20140127/DABJ40P00.html.

[24] TIOBE Software. Tiobe index for july 2014. TIOBE Software, Julho de 2014. Acessado em: 01 de Agosto de 2014. Disponível em: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html.

[25] Mornini T. Here's why ruby on rails is hot. Business Insider, 11 de Maio de 2011. Acessado em: 01 de Agosto de 2014. Disponível em: http://www.businessinsider.com/heres-why-ruby-on-rails-is-hot-2011-5.

[26] Zeng T. Ruby interface to libsvm (using swig). Tomz Consulting, 27 de Abril de 2012. Acessado em: 01 de Agosto de 2014. Disponível em: http://github.com/tomz/libsvm-ruby-swig/tree/master.

[27] Kevin W. World economic and social survey 2013 sustainable development challenges. United Nations, 2013. Acessado em: 01 de Agosto de 2014. Disponível em: http://sustainabledevelopment.un.org/content/documents/2843WESS2013.pdf.

[28] Matsumoto Y. About ruby. Ruby-Lang.org, 2014. Acessado em: 01 de Agosto de 2014. Disponível em: https://www.ruby-lang.org/en/about.

[29] Matsumoto Y. Speak on the rubytalk mailing list. RubyTalk, 12 de Março de 2000. Acessado em: 01 de Agosto de 2014. Disponível em: `http://blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-talk/2773`.

[30] Fox Z. 10 empowering apps for social good. Mashable, 20 de Setembro de 2013. Acessado em: 01 de Agosto de 2014. Disponível em: `http://mashable.com/2013/09/20/social-good-apps`.