UNIVERSIDADE FEDERAL DE MINAS GERAIS
ENGENHARIA DE SISTEMAS

MACHINE LEARNING AUTOMATED MODEL COMPARISON: UM SOFTWARE PARA AUTOMATIZAR A COMPARAÇÃO ENTRE MÉTODOS DE APRENDIZADO

PAULO CIRINO RIBEIRO NETO

Orientador: Antônio de Pádua Braga
Universidade Federal de Minas Gerais

Coorientador: Gustavo Rodrigues Lacerda Silva
Universidade Federal de Minas Gerais

BELO HORIZONTE
NOVEMBRO DE 2017

PAULO CIRINO RIBEIRO NETO

MACHINE LEARNING AUTOMATED MODEL COMPARISON: UM SOFTWARE PARA AUTOMATIZAR A COMPARAÇÃO ENTRE MÉTODOS DE APRENDIZADO

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia de Sistemas da Universidade Federal de Minas Gerais, como requisito parcial para aprovação na disciplina de Trabalho de Conclusão de Curso I

Orientador: Antônio de Pádua Braga
Universidade Federal de Minas Gerais

Coorientador: Gustavo Rodrigues Lacerda Silva
Universidade Federal de Minas Gerais

UNIVERSIDADE FEDERAL DE MINAS GERAIS
ENGENHARIA DE SISTEMAS
BELO HORIZONTE
NOVEMBRO DE 2017

AGRADECIMENTOS

Agradeço ao meu orientador Prof. Dr. Antônio de Pádua Braga, pelo apoio ao longo dessa trajetória. Aos colegas do LITC Dr. Luiz Carlos Bambirra Torres, Dr. Frederico Gualberto Ferreira Coelho e, especialmente, ao meu coorientador Eng. Gustavo Rodrigues Lacerda. Aos meus colegas de sala, professores e especialmente ao secretário do curso de Engenharia de Sistemas Júlio César Pereira de Carvalho. Por fim, gostaria de mostrar o meu reconhecimento à minha família, pois acredito que sem o apoio deles seria muito difícil vencer esse desafio.

“The only excuse for making a useless thing is that one admires it intensely.” (Oscar Wilde, prefácio, O retrato de Dorian Gray)

RESUMO

O tema desse trabalho de conclusão de curso é a criação de um software para automatizar testes de modelos de aprendizado de máquina.

Ao fim do projeto, existirá um software, escrito na linguagem de programação R, capaz de comparar modelos de aprendizado de máquina consagrados na literatura com modelos criados pelo usuário. O software fará o tratamento das bases de dados padrões, suas chamadas e os testes estatísticos de qualidade e tempo para os problemas de classificação, regressão e clusterização.

Será discutido nesse trabalho, o impacto social de software livre e os aspectos técnicos e definições matemáticas do aprendizado supervisionado e não supervisionado.

Palavras-chave: Software Livre. Aprendizado de Máquina. Linguagem R. Teste de Modelos.

ABSTRACT

The theme of this work is the creation of a software capable of automating machine learning models testing.

At the end of the project, there will be a software written in the R programming language, capable of comparing established machine learning model with ones created by the user. The software will make standard database treatments, create function calls for testing, and define statistical tests routines of bolth quality and time for classification, regression and clustering problems.

The discussion of social impact of free software and the technical and mathematical aspects of supervised and unsupervised learning, will also be part of the project.

Keywords: Free Software. Machine Learning. R programming Language. Model Testing. Benchmarking.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

UFMG	Universidade Federam de Minas Gerais
TCC	Trabalho de Conclusão de Curso
GNU	*GNU's Not Unix*
GPL	*General Public License*
MIT	*Massachusetts Institute of Technology*
BSD	*Berkeley Software Distribution*
NSA	*National Security Agency*
SO	Sistema Operacional
CRAN	*Comprehensive R Archive Network*
CPU	*Central Processing Unit*
GPU	*Graphical Processing Unit*
POO	Programação Orientada à Objetos
RNA	Redes Neurais Artificiais
ML	*Machine Learning*
KNN	*K Nearest Neighbours*
NN	*Nearest Neighbour*
MLP	*Multilayer Perceptron*
PCA	*Principal Component Analasys*
SVM	*Support Vector Machines*

LISTA DE SÍMBOLOS

x	Entrada real
y	Saída real
$\hat{y}$	Saída estimada
$p(x)$	Distribuição de probabilidade de uma variável x
$p(y\|x)$	Probabilidade condicional de x dado y
μ	Média
σ	Desvio padrão
ϵ	Erro aleatório com $\mu = 0$
ω	Parâmetro de uma função
Ω	Conjunto de todos os parâmetros de uma função
$\mathbb{R}$	Conjunto dos números reais
R^+	Conjunto dos números reais positivos
$\mathbb{I}$	Conjunto dos números inteiros
$\mathbb{N}$	Conjunto dos números naturais
L	Função de Custo
f	Função do Aprendizado de Máquina
g	Função do Sistema
$d(r, s)$	Distância entre as variáveis r e s
$\in$	Pertence
$\wedge$	Simbolo de *e* lógico
$\vee$	Simbolo de *ou* lógico
$\rightarrow$	Implica

SUMÁRIO

Capítulo 1

Introdução

1.1 Justificativa

Um dos principais objetivos de softwares é auxiliar os usuários nos trabalhos com sistemas computacionais. Uma das formas de atingir esse objetivo é automatizando tarefas rotineiras, repetitivas, desinteressantes e que consomem muito tempo, liberando as pessoas para tarefas mais importantes (ZAMBIASI; RABELO, 2012).

Uma parte fundamental no ciclo do desenvolvimento de novos algorítimos em aprendizado de máquina é o processo de *benchmarking*. Esse procedimento consiste na comparação do novo método com outros modelos, além, é claro, de testes estatísticos que permitam extrair conclusões quantitativas e qualitativas.

Realizar *benchmarking* pode se tornar uma tarefa repetitiva, trabalhosa e consequentemente cara. O engenheiro precisa implementar todos os métodos padrões, baixar e pré-processar todas as bases de testes, definir o planejamento e análise dos experimentos, e por fim, realizar testes estatísticos e desenhos gráficos para que seja possível extrair suas conclusões.

Dessa forma, a automatização da tarefa de *benchmarking* seria benéfica ao engenheiro no sentido de economizar tempo para realizar tarefas mais importantes. Além disso, uma rotina de teste padronizada é interessante pois faz com que seja possível a comparação de estudos feitos separadamente.

1.2 Objetivo

Esse trabalho tem como objetivo a criação de um software, que seja capaz de automatizar a tarefa de *benchmarking* para o teste de novos modelos de aprendizado de máquina.

O projeto será executado na forma de pacote aberto, licença GNU GPLv3, da linguagem de programação estatística R. O pacote será implementado utilizando a própria linguagem R, sua API para C e $C++$ e outros pacotes livres feitos pela comunidade.

Ao fim do ciclo de desenvolvimento, o pacote deverá ser capaz de comparar modelos para tarefas de classificação, clusterização e regressão desenvolvidos pelos usuários com diversos modelos já presentes no pacote. O usuário poderá fornecer as bases de dados de testes ou utilizar as bases do pacote. O pacote deverá fornecer também a opção de testes estatísticos e visualizações dos resultados.

Após a conclusão do projeto, ele será enviado para publicação no repositório CRAN e submetido ao *The R Journal*, um periódico para divulgação de pacotes feitos pela comunidade R.

1.3 Organização do trabalho

O trabalho está estruturado de forma que o capítulo 2 tem como objetivo contextualizar e discutir os aspectos sociais e culturais do software livre, bem como falar brevemente da história desse movimento e das suas definições de liberdades sociais.

O Capítulo 3, discutirá sobre a linguagem de programação R, como ela está inserida dentro do movimento de software livre e como ela está organizada.

Por fim, os capítulos 4 e 5 discutirão, de forma sucinta, os conceitos de aprendizado de máquina e de estatísticas de testes que são as bases da implementação do software.

Nesse trabalho não serão discutidas as métricas e modelos que serão implementados, e tampouco as questões técnicas de implementação do projeto que serão abordadas no texto da disciplina de TCC II .

Capítulo 2

Software Livre

2.1 O Que é Software Livre

A definição de software livre apresenta os critérios para determinar se um programa de software é qualificado como livre. Essa definição pode mudar conforme o momento histórico, mas atualmente é definida pela GNU como : *software que os usuários têm a liberdade de executar, copiar, distribuir, estudar, alterar e melhorar* (INC, 2012) .

Nesse contexto, o conceito de livre diz respeito à "liberdade de expressão", e não à gratuidade (WILLIAMS; STALLMAN, 2010). Com essas liberdades, os usuários, tanto individualmente quanto coletivamente, controlam o programa e o que ele pode fazer. Quando os usuários não controlam o programa, chamamos ele de programa *não livre* ou *proprietário* (INC, 2012).

Em linhas gerais, um software livre deve, obrigatoriamente, obedecer a quatro liberdades (INC, 2012; WILLIAMS; STALLMAN, 2010; LESSIG; STALLMAN, 2002) :

1. A liberdade de executar o programa como desejar, para qualquer propósito;
2. A liberdade de estudar como funciona o programa e mudá-lo para que ele faça a sua computação como você deseja;
3. A liberdade de redistribuir cópias para que você possa ajudar seu vizinho;
4. A liberdade de distribuir cópias de suas versões modificadas para outros.

Naturalmente, essas liberdades estão relacionadas às informações divulgadas entre desenvolvedor e usuário. A liberdade **2**, por exemplo, implica na necessidade de o desenvolvedor divulgar abertamente o código fonte de um software e permitir que esse seja modificado (INC, 2012). Além disso, é obvio que para os usuários terem a liberdade de decidir sobre sua computação, o software livre, por sua vez, não pode utilizar nenhum código *não livre*.

É importante notar que não existem, nessas liberdades, quaisquer limitações relacionadas

ao uso comercial do software. Isso significa que empresas podem cobrar pelo uso do software em versões modificadas ou não.

Além das liberdades básicas de um software livre, existem as licenças de uso. Cada licença específica, restringe ou garante mais liberdades sobre o software, sem afetar as quatro fundamentais. De forma forma geral, existem quatro famílias de licenças : *Permissiva*, *Fracamente Protetiva*, *Fortemente Protetiva* e *Protetiva de Rede* (WILLIAMS; STALLMAN, 2010).

2.2 História do Software Livre

O surgimento do movimento de software livre está altamente atrelado à academia e ao desenvolvimento dos sistemas operacionais UNIX, GNU e Linux.

O UNIX tem suas origens na *joint venture*, lançada no final da década de 1960 pela *Bell Labs* e MIT para criar um novo sistema operacional chamado *Multics* (TOZZI; ZITTRAIN, 2017). Utilizando o conhecimento adquirido nesse projeto, alguns dos programadores desenvolveram, paralelamente, um sistema operacional para oferecer mais flexibilidade aos usuários, nomeado UNIX.

Em 1975, Ken Thompson juntamente com Bill Joy e Chuck Haley, começaram a distribuir uma versão *open source* do UNIX chamada BSD. No ano seguinte, o lançamento de uma edição revista foi denominada 2BSD (TOZZI; ZITTRAIN, 2017).

No ano de 1984, o programador Richard Stallman fundou o Projeto GNU. A GNU GPL permitia aos usuários modificar o código e distribuir a versão melhorada sob a mesma licença. O sistema operacional GNU não tinha um *kernel*, até que em 1991, Linus Torvalds desenvolveu o *kernel* do Linux que foi integrado no sistema operacional GNU em 1992 (TOZZI; ZITTRAIN, 2017).

Nos anos seguintes, foram introduzidas versões, comerciais e aprimoradas, do sistema operacional Linux por fornecedores como *Red Hat*, *Mandriva* e *Novell*.

Com a criação de sistemas operacionais que poderiam ser utilizados com total liberdade, surgiu então uma demanda por softwares livres que funcionassem nesses ambientes. Da mesma forma que a comunidade se juntou para aprimorar a base do *kernel* do Linux, eles se juntaram e fizeram os mais diversos softwares para atender essa demanda.

Nos dias atuais existem inúmeras comunidades que fazem os mais variados tipos de softwares utilizando os mesmos princípios criados por Richard Stallman. Os softwares livres difundiram na sociedade, e hoje são peças fundamentais para infraestrutura computacional, periféricos, celulares e virtualmente qualquer outro dispositivo computacional.

2.3 A Importância Social do Software Livre

Atualmente, os softwares livres são fundamentais principalmente em três áreas sociais: à proteção da liberdade individual, ao avanço da computação e à acessibilidade da educação e conhecimento.

2.3.1 Software Livre à Favor da Proteção da Liberdade Individual

Uma celebre frase da comunidade de software livre diz, 'Os softwares não livres, onde o usuários não controlam o programa, o programa controla os usuários' (WILLIAMS; STALLMAN, 2010).

Essa frase, resume uma preocupação crescente com o software proprietário, a de que sempre há alguma entidade, que controla o programa e através dele, exerce poder sobre seus usuários.

Esse poder é enxergado por alguns como uma afronte ao direito de privacidade do indivíduo. Hoje, serviços de busca e redes sociais, utilizam dos dados de navegação para gerar propagandas sob-medidas. Muitas pessoas, consideram a forma que essas empresas manipulam as informações como uma forma de censura e venda de informação confidencial.

Em alguns casos, empresas que desenvolvem softwares proprietários foram ligadas a escândalos onde propositalmente construíram *backdoors* em seus produtos que dão acesso a informações sem as devidas permissões dos usuários. Um exemplo é o caso do *Kindle*, que possui uma *backdoor* que permite apagar livros (GNU Operating System, 2017) .

Um outro senário onde é importante que o software seja livre, é para proteger os usuários de acesso externo indesejado. Um exemplo disso é o caso do ex analista da NSA (TATE et al., 2013), Edward Joseph Snowden, que em junho de 2013, revelou como a agência americana utilizava de falhas de seguranças, propositais ou não, para espiar na população mundial.

O principio que o software livre protege a liberdade Individual, contra empresas mau intencionadas ou governos abusivos, é que quando o código de um programa é aberto, a comunidade pode ver oque ele faz e testar todas as falhas que o mesmo possa ter (GNU Operating System, 2017).

2.3.2 Software Livre à Favor do Avanço da Computação

Após a construção dos sistemas operacionais livres, surgiram varias distribuições e variações, cada qual para atender um nicho. Esses avanços, tornaram possível que hoje, os sistemas operacionais baseados no Linux e UNIX dominassem o setor de infra estrutura

computacional. Possibilitando que empresas e órgãos governamentais customizassem esses softwares para criar soluções específicas para suas necessidades, que por sua vez não são necessariamente livres.

Essa abordagem se tornou tão prática que, dados da W3Techs (2017) e Top 500 project (2017), mostram que esses sistemas operacionais são utilizados em 66.8% de todos os servidores públicos de internet e 99.88% dos supercomputadores do mundo.

A ultima grande plataforma que popularizou a utilização do Linux foi o sistema operacional Android. Construído inicialmente para ser um software de telefones celulares, esse sistema operacional criado com base no *kernel* do Linux, se espalhou pelos mais diversos aparelhos, como televisões, *tablets* e até mesmo geladeiras (RILEY, 2012). Esse software se popularizou tanto que, segundo o CEO da Google, Sundar Pichai, é utilizado em mais de 2 bilhões de dispositivos ativos (RILEY, 2012).

Além dos avanços em sistemas operacionais, o movimento de software livre alavancou o desenvolvimento comunitário de softwares de computação livres. Um exemplo disso é a *Apache Software Foundation*, que é uma corporação americana sem fins lucrativos formada por uma comunidade descentralizada de desenvolvedores de código aberto.

Os projetos Apache são feitos em desenvolvimento colaborativo, baseado em consenso e uma licença de software aberta e pragmática. Cada projeto é gerenciado por uma equipe de especialistas técnicos auto-selecionados que são contribuidores ativos para o projeto.

O projeto inicial da Apache foi o *HTTP Server*, que era um sistema para processar protocolos web básicos na internet. Contudo, hoje são 315 projetos nas mais diversas áreas da computação, e incluem softwares de *Big Data* como o *Spark* e *Hadoop*, gerenciamento de projetos como o *Maven* e até mesmo software de escritório como o *Open Office*.

2.3.3 Software Livre à Favor da Acessibilidade da Educação e Conhecimento

As escolas e universidades, influenciam o futuro da sociedade através do que ensinam. Ensinar um programa proprietário é implantar a dependência de um artifício que não é de sua propriedade. Isso diminui a capacidade do futuro profissional em exercer os conhecimentos que lhe foram ensinados (Richard Stallman, 2017).

A utilização de software livre como ferramente de ensino, empodera os estudantes a utilizarem os conhecimentos técnicos na vida após a universidade (LESSIG; STALLMAN, 2002). Utilizando software livre, um profissional é capaz de fazer uso de programas como ferramenta básica de trabalho gratuitamente, ou ainda utilizar soluções livres como base para criar um produto próprio.

Além de ser útil para o estudante, os softwares livres são importantes para o avanço da pesquisa acadêmica na universidade. A comunidade de Software livre têm em suas liberdades básicas, a liberdade de estudar como um programa funciona e permitir que os usuários melhorem e redistribuam esse programa. Este espírito de comunidade permite que pesquisadores em locais diferentes do mundo, sem quaisquer dificuldades, compartilharem suas pesquisas e conhecimentos.

Além de promover o compartilhamento da informação, os softwares livres também promovem uma acessibilidade universal dos avanços técnico-científicos, uma vez que permitem pesquisadores e empresas de ponta à compartilhar seus resultados e códigos com o resto do mundo (LESSIG; STALLMAN, 2002). Isso permite que mesmo pesquisadores com recursos escaços, façam aplicações ou pesquisa com esses avanços .

Um exemplo disso, é o caso do software *Tensor Flow* (ABADI et al., 2016), feito pela Google. Esse é um programa extremamente complexo que funciona como motor de operações numéricas, utilizando a abstração de computação em grafos de forma escalável para CPU's e GPU's. Além de compartilhar o código do *Tensor Flow* com a comunidade, a empresa também disponibilizou inúmeros modelos de RNA, como a *LeNet* (SZEGEDY et al., 2015) que é um modelo treinado para identificar objetos em imagens. Utilizando esse avanço, pesquisadores do mundo inteiro foram capazes de utilizar esses modelos em suas próprias pesquisas.

Capítulo 3

Linguagem de Programação R

3.1 História da Linguagem R

R é uma linguagem e ambiente para computação estatística e gráfica, que foi criada na década de 90 por Ross Ihaka e Robert Gentleman, enquanto ambos trabalham na Universidade de Auckland. Esse é um projeto GNU que é semelhante a linguagem e ao ambiente *S* desenvolvida na *Bell Laboratories* por John Chambers.

O *R* pode ser considerado como uma implementação diferente da linguagem de programação *S*, ao ponto que código escrito em *S* pode ser executado de forma inalterada pelo interpretador de *R*. Contudo, a principal diferenças entre os dois ambientes, é que o *R* é um projeto de software livre sob licença GPL GNU e o *S* possui licença proprietária.

Atualmente esse ambiente é utilizando principalmente nas áreas de estatística e análise de dados. O sucesso da linguagem nessas tarefas pode ser explicado pela grande variedade de algorítimos em modelagem linear e não-linear, testes estatísticos clássicos, análise de séries temporais e técnicas gráficas. Nos dias atuais, a linguagem é utilizada na indústria e academia,de forma que já é a sexta linguagem de programação mais popular (CASS, 2017).

Apesar de ser uma linguagem livre e aberta, o desenvolvimento e manutenção do ambiente é controlada pela *R Foundation* e pelo grupo de 20 curadores chamados de *R core team*. Contudo, qualquer pessoa pode contribuir com o avanço da linguagem por meio de pacotes, tradicionalmente disponibilizados no repositório oficial CRAN e divulgados na revista chamada *The R Journal*.

3.2 Organização da Linguagem de Programação *R*

O *R* é uma linguagem de programação que não foi planejada para ser tão versátil. Inicialmente, foi criado um interpretador que executava comandos da linguagem *S*, onde o ambiente era composto por funções de matemáticas, estatísticas, gráficas e de leitura de dados.

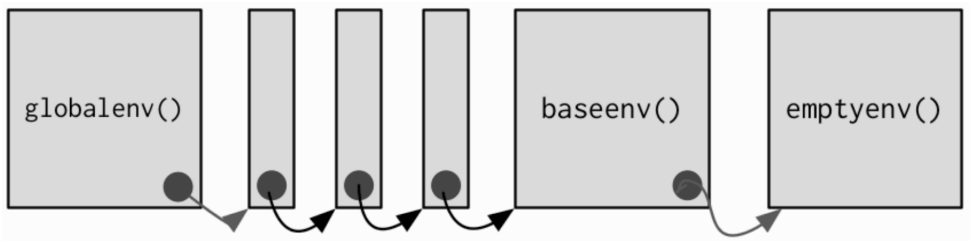
Como a linguagem foi criada por estatísticos para estatísticos, não foram definidos padrões de programação. Dessa forma ela é considerada uma linguagem de quarta geração, ou seja, uma linguagem de domínio.

Possui suporte para os principais paradigmas de programação, incluindo os modelos declarativo, imperativo, orientado à objeto, procedural, funcional e outros. É um linguagem naturalmente lenta, por conta de ser interpretada e concorrente. Entretanto, fornece interface para conectar com linguagens de alta performance, *C*, *C++* e *Fortran*, e programação paralela pela plataforma *OpenMP* .

Segundo o projeto *OpenHub*, o código do interpretador *R* possui mais de 750 mil linhas, das quais 39% são escritas em *C*, 27.1% em *Fortran*, 19.7% em *R*, 8.1% em *Autoconf* e 2% em *shell*.

Esse interpretador foi definido para trabalhar com o conceito de *ambientes*, que são grupos de funções e variáveis organizadas por precedência de contexto. O formato dos contextos podem ser observados na figura 1 .

Figura 1 – *Ambientes* da linguagem de programação R



Fonte: Wickham (2015)

Em linhas gerais, o ambiente onde o usuário trabalha é chamado de *globalenv*, as funções da linguagem *R* estão definidas no ambiente de *baseenv* e os gerenciadores de memória, o coletor de lixo e os demais controladores do próprio interpretador estão no ambiente *emptyenv*.

A linguagem *R*, interpreta as funções de forma a percorrer os ambientes hierarquicamente. Quando o usuário chama o interpretador, ele inicialmente procura as funções no *globalenv*, quando não encontra, ele busca recursivamente nos ambientes *pai*.

Todos os ambientes entre o *globalenv* e o *baseenv*, são chamados de *ambientes de pacote*. O ultimo pacote a ser invocado é chamado de pai do *globalenv*, e é filho do penúltimo pacote invocado.

Esses pacotes foram a forma encontrada pelos desenvolvedores da linguagem para permitir que a comunidade contribuísse com o avanço da linguagem, sem afetar as funções base. Além disso, esse processo recursivo de busca de *ambientes* é o que permite os pacotes da linguagem R utilizarem uns aos outros.

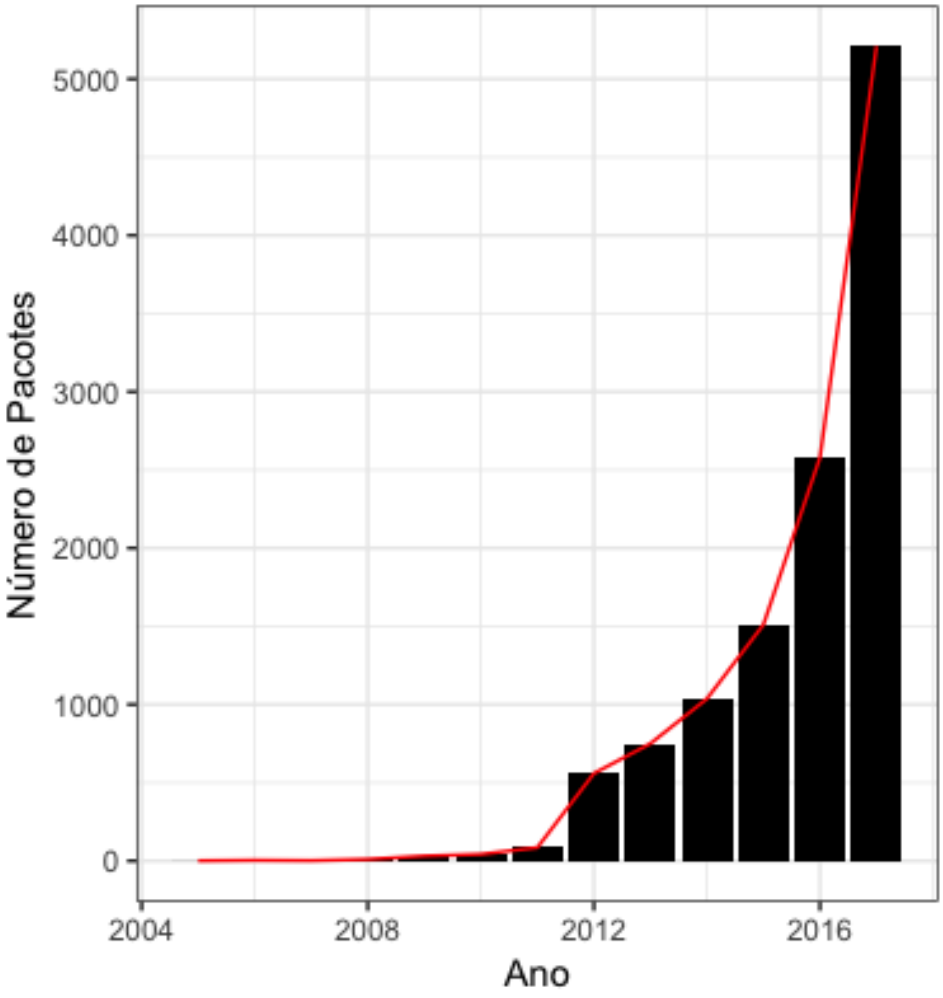
3.3 A comunidade R e seus Pacotes

A linguagem R surgiu dentro do meio universitário como uma alternativa à softwares proprietários. Dessa forma, a expansão da comunidade de usuários ocorreu inicialmente da mesma maneira, no âmbito acadêmico e por meio softwares e pacotes livres. Contudo, hoje o R possui grande aporte de empresas privadas, como Google e AT&T.

Atualmente o repositório CRAN possui mais de 11800 pacotes, que implementam modelos dos mais variados, incluindo mas não limitado à gráficos, biológicos, estatísticos, ecológicos e de aprendizado de maquina. É importante ressaltar que todos os pacotes devem ter documentação, código aberto e possuírem licença livre.

Esses padrões definidos pela comunidade têm se mostrado extremamente efetivos, no sentido da popularização da criação de pacotes, vide figura 2.

Figura 2 – Número de pacotes disponíveis no CRAN ao longo do tempo



Fonte: https://cran.r-project.org/web/packages/available_packages_by_date

Capítulo 4

Aprendizado de Máquina

4.1 Formulação e Caracterização do Aprendizado de Máquina

Nos primórdios da inteligência artificial, pesquisadores foram capazes de resolver problemas que são intelectualmente difíceis para os seres humanos, mas relativamente simples para os computadores. Exemplo disso são os problemas que podem ser descritos por uma sequência de regras ou que podem ser solucionados por operações matemáticas (GOODFELLOW; BENGIO; COURVILLE, 2016).

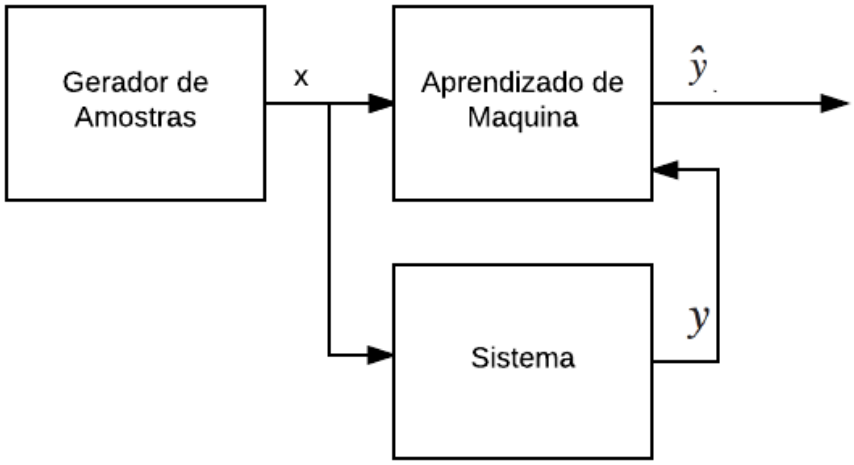
A dificuldade surgiu para fazer com que os computadores fossem capazes de resolver tarefas fáceis para os humanos, mas difíceis de serem descritas de forma algorítmica, como reconhecer escrita, identificar rostos em imagens ou ainda definir regras de decisão de forma autônoma (GOODFELLOW; BENGIO; COURVILLE, 2016).

Eis que surgiu o campo de Aprendizado Máquina, uma alternativa que permitia computadores aprender por exemplo e não por regras. Ao reunir o conhecimento da experiência, esta abordagem evita a necessidade de operadores humanos especificar formalmente todo o conhecimento que o computador precisa. Isso ocorre através da combinação do aprendizado de conceitos simples, em formato pré-definido, combinados de forma hierárquica para formar um aprendizado de conceitos complicados (GOODFELLOW; BENGIO; COURVILLE, 2016).

Segundo Cherkassky e Mulier (2007), o processo de aprendizado é definido como a estimação de uma relação ou estrutura, previamente desconhecida, entre entrada e saída. Segundo o autor, o processo de aprendizado normalmente envolve três componentes 3 : o *Gerador* de amostras, um *Sistema* que gera as saídas reais e o *Aprendizado de Máquina* que estima a relação desconhecida entre entrada e saída.

Cherkassky e Mulier (2007) explica que o *Gerador* produz aleatoriamente, com distribuição de probabilidade $p(x)$, vetores $x \in \mathbb{R}^n$, que são as características. O *Sistema* é capaz de

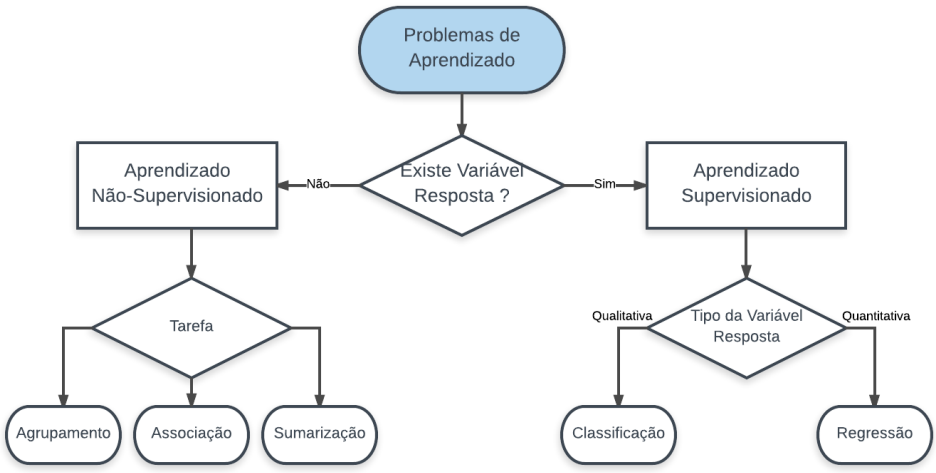
Figura 3 – Formulação do Aprendizado



produzir, com probabilidade $p(y|x)$, saídas $y = g(x) + \epsilon$ em que ϵ é um ruido branco de $\mu = 0$. De forma geral, o *Aprendizado de Máquina* é capaz de implementar uma função $\hat{y} = f(x, \omega)$, $\omega \in \Omega$, em que Ω é conjunto de parâmetros da função f.

O aprendizado de máquinas clássico diz respeito à aprender pelo exemplo, que é definido como um conjunto de dados, previamente obtidos pelo *Gerador* e *Sistema*, composto por variáveis quantitativas e qualitativas que são características do problema. Na situação onde existe uma variável resposta, o aprendizado é chamado de supervisionado, na situação onde essa variável não existe, o problema é definido como aprendizado não supervisionado. A figura 4 ilustra essa taxonomia, bem como algumas de suas sub-divisões.

Figura 4 – Principais Problemas em Aprendizado de Máquina



4.2 Aprendizado Supervisionado

No aprendizado supervisionado, o objetivo é prever o valor de uma variável resultado com base em variáveis características. Na situação que a variável resposta é quantitativa, o problema é considerado como uma tarefa de regressão. Quando essa variável resposta é quantitativa, a tarefa é chamada de classificação.

De acordo com Cherkassky e Mulier (2007), existem duas interpretações para o problema de aprendizado : identificação e imitação. A interpretação escolhida para fundamentar esse trabalho é a de imitação, descrita por Vapnik e Chervonenkis (1971).

O objetivo do aprendizado é encontrar uma função $f(x, \omega)$ que aproxima, da melhor forma possível, a saída y do *Sistema*. Para formular esse problema matematicamente, Cherkassky e Mulier (2007), assumem pares de características e respostas, (x_i, y_i) em que $i = (1, 2, ..., n)$ e $n \in \mathbb{N}$, e uma função de custo $L(y, f(x, \omega))$que mede a discrepância entra saída do *Sistema* y e do *Aprendizado de Máquina* $\hat{y}$. Então, formulam que a tarefa do aprendizado é descrita pela equação 1 .

$$\arg\min_{f} L(y, f(x, \omega)) \tag{1}$$

A principal diferença entre problemas de classificação e regressão é a função de custo. Como a variável de saída de cada um desses problemas tem um formato diferente, a função de custo deve ser adaptada especificadamente para ele. Contudo, a formulação do aprendizado feito na equação 1 permanece inalterada, pois o objetivo é sempre minimizar a discrepância entre *Sistema* e *Aprendizado de Máquina* (CHERKASSKY; MULIER, 2007).

4.2.1 Problemas de Classificação

A situação mais simples de classificação é aquele em que a saída y assume apenas dois valores, $y = 0$ ou $y = 1$, em que é chamada de problema de classificação binária. Nessa situação, a função de custo específica é do formato 2.

$$L(y, f(x, \omega)) = \begin{cases} 0, & \text{se } y = f(x, \omega) \\ l, & \text{se } y \neq f(x, \omega) \end{cases} \quad l \in \mathbb{R}^+ \tag{2}$$

O senário em que y assume q valores, em que $q > 2 \wedge q \in \mathbb{N}$, pode ser simplificado em q problemas de classificação binária em que o problema j tem forma $y = j \rightarrow y_j = 1$ e $y_j \neq\rightarrow y_j = 0$, para $0 \leq j \leq q \wedge j \in \mathbb{N}$.

4.2.2 Problemas de Regressão

O problema de regressão é caracterizado por $y \in \mathbb{R}$, dessa forma a função de custo normalmente é formulada como mostra a equação 3, onde d é uma métrica genérica de distância entre dois números reais.

$$L(y, f(x, \omega)) = d(y, f(x, \omega)) \tag{3}$$

4.3 Aprendizado Não Supervisionado

O aprendizado não supervisionado é caracterizado pela não existência de uma variável resposta y. Contudo, os problemas dessa tarefa podem ser divididos em três grandes grupos : agrupamento, associação e sumarização. Será discutido nessa seção apenas a tarefa de agrupamento, porque é a única que será contemplada pelo projeto final de software.

4.3.1 Agrupamento

A tarefa de agrupamento, diz respeito a atribuição de rótulos para amostras de forma que aquelas com rótulos iguais possuam mais similaridades entre sí do que entre aquelas com rótulos diferentes.

Não existe uma padronização, entre autores, na classificação das metodologias para realizar essa tarefa. Porém, segundo Berkhin (2006), eles podem ser divididos em três grandes grupos :

- **Hierárquicos :** São modelos baseados em agrupar amostras hierarquicamente, por meio de uma função de similaridade.
- **Baseado em densidade :** São modelos que utilizam a ideia de que agrupamentos estão definidos por amostras em regiões densas do espaço.
- **Baseado em partição :** São modelos que particionam o espaço amostral em regiões, normalmente esses métodos são formulados por otimização iterativa e incluem os algorítimos baseados em centros e distribuições de probabilidade.

Capítulo 5

Estatísticas de Teste

5.1 O que é Estatística de Teste

Uma estatística de teste é o valor que é calculado a partir de dados durante um teste de hipóteses. O valor dessa estatística mede o grau de concordância entre uma amostra de dado e a hipótese nula, que por sua vez pode ser rejeitada ou não (CASELLA; BERGER, 2002).

Em um teste estatístico, as hipóteses são premissas a serem testadas. Tradicionalmente existem duas hipóteses, a nula e a alternativa, em que o objetivo do teste é conservadoramente rejeitar a hipótese nula à favor da hipótese alternativa. Dessa forma, os testes são feitos tal que o objetivo que deseja-se testar é descrito pela hipótese alternativa.

Os tipos de testes que serão implementados no projeto são os testes pareados paramétricos e não-paramétricos.

5.2 Teste Pareado

Os testes pareados são aqueles utilizados para comparar um conjuntos de medidas de duas populações e avaliar se elas são diferentes. Nessa situação, tradicionalmente a hipótese nula é a diferença entre estatística entre as populações, e a hipótese alternativa pode ser que uma é maior que a outra ou que são diferentes.

Uma vantagem de utilizar os testes pareados, é que eles tornam possível a comparação de múltiplas populações, fazendo vários testes 2 à 2.

5.3 Testes paramétrico e não-paramétrico

REFERÊNCIAS

ABADI, M. et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. **CoRR**, abs/1603.04467, 2016. Disponível em: <http://arxiv.org/abs/1603.04467>. Citado na página 7.

BERKHIN, P. A survey of clustering data mining techniques. **Grouping multidimensional data**, Springer, v. 25, p. 71, 2006. Citado na página 15.

CASELLA, G.; BERGER, R. L. **Statistical inference**. [S.l.]: Thomson Learning, 2002. 660 p. ISBN 0534243126. Citado na página 16.

CASS, S. The 2017 Top Programming Languages. **IEEE Spectrum**, 2017. Disponível em: <https://spectrum.ieee.org/computing/software/the-2017-top-programming-languages>. Citado na página 8.

CHERKASSKY, V. S.; MULIER, F. **Learning from data : concepts, theory, and methods**. [S.l.]: IEEE Press, 2007. 538 p. ISBN 0471681822. Citado 2 vezes nas páginas 12 e 14.

GNU Operating System. **Proprietary Software Is Often Malware**. 2017. Disponível em: <https://www.gnu.org/proprietary/proprietary.html>. Citado na página 5.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>. Citado na página 12.

INC, F. S. F. **What is free software?** 2012. Disponível em: <http://www.gnu.org/philosophy/free-sw.html>. Citado na página 3.

LESSIG, L.; STALLMAN, R. **Free Software, Free Society: Selected Essays of Richard M. Stallman**. [s.n.], 2002. Disponível em: <https://www.gnu.org/philosophy/fsfs/rms-essays.pdf>. Citado 3 vezes nas páginas 3, 6 e 7.

Richard Stallman. **Why Schools Should Exclusively Use Free Software**. 2017. Disponível em: <https://www.gnu.org/education/edu-schools.en.html>. Citado na página 6.

RILEY, M. **Programming your home : automate with Arduino, Android, and your computer**. [S.l.]: Pragmatic Bookshelf, 2012. 216 p. ISBN 1934356905. Citado na página 6.

SZEGEDY, C. et al. Going deeper with convolutions. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2015. p. 1–9. Citado na página 7.

TATE, J. et al. **Edward Snowden says motive behind leaks was to expose 'surveillance state'**. 2013. Disponível em: <https://www.washingtonpost.com/politics/edward-snowden-says-motive-behind-leaks-was-to-expose-surveillance-state/2013/06/09/aa3f0804-d13b-11e2-a73e-826d299ff459{_}story.html?tid=pm{_}politics{_}pop{&}utm{_}ter>. Citado na página 5.

Top 500 project. **Operating system Family - Systems share**. 2017. Disponível em: <https://www.top500.org/lists/2017/11/>. Citado na página 6.

TOZZI, C.; ZITTRAIN, J. **For Fun and Profit: A History of the Free and Open Source Software Revolution**. MIT Press, 2017. (History of Computing). ISBN 9780262036474. Disponível em: <https://books.google.com.br/books?id=MXosDwAAQBAJ>. Citado na página 4.

VAPNIK, V. N.; CHERVONENKIS, A. Y. On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities. **Theory of Probability & Its Applications**, Society for Industrial and Applied Mathematics, v. 16, n. 2, p. 264–280, jan 1971. ISSN 0040-585X. Disponível em: <http://epubs.siam.org/doi/10.1137/1116025>. Citado na página 14.

W3TECHS. **Usage of operating systems for websites**. 2017. Disponível em: <https://w3techs.com/technologies/overview/operating{_}system/>. Citado na página 6.

WICKHAM, H. **Advanced R**. CRC Press, 2015. Disponível em: <https://englianhu.files.wordpress.com/2016/05/advanced-r.pdf>. Citado na página 9.

WILLIAMS, S.; STALLMAN, R. M. Free as in Freedom (2.0): Richard Stallman and the Free Software Revolution Second edition revisions. 2010. Disponível em: <https://sagitter.fedorapeople.org/faif-2.0.pdf>. Citado 3 vezes nas páginas 3, 4 e 5.

ZAMBIASI, S. P.; RABELO, R. J. Uma arquitetura aberta e orientada a serviços para softwares assistentes pessoais. **RITA**, v. 19, p. 93–119, 2012. Citado na página 1.