



Universidade de Brasília

Instituto de Ciências Exatas
Departamento de Ciência da Computação

Automação de classificador SVM para aplicação em projetos de consultoria de gestão

Filipe Guedes de O. Almeida

Dissertação apresentada como requisito parcial para conclusão do
Mestrado Profissional em Computação Aplicada

Orientador
Prof. Dr. Gladston Luiz da Silva

Brasília
2019

**Ficha catalográfica elaborada automaticamente,
com os dados fornecidos pelo(a) autor(a)**

GF483a

Guedes de Oliveira Almeida, Filipe
Automação de classificador SVM para aplicação em projetos
de consultoria de gestão / Filipe Guedes de Oliveira
Almeida; orientador Gladston Luiz da Silva. -- Brasilia,
2019.
62 p.

Dissertação (Mestrado - Mestrado Profissional em
Computação Aplicada) -- Universidade de Brasilia, 2019.

1. Automated Machine Learning. 2. SVM. 3. Support Vector
Machine. 4. Machine Learning. 5. Mineração de Dados. I. Luiz
da Silva, Gladston, orient. II. Título.



Universidade de Brasília

Instituto de Ciências Exatas
Departamento de Ciência da Computação

Automação de classificador SVM para aplicação em projetos de consultoria de gestão

Filipe Guedes de O. Almeida

Dissertação apresentada como requisito parcial para conclusão do
Mestrado Profissional em Computação Aplicada

Prof. Dr. Gladston Luiz da Silva (Orientador)
IE/UnB

Prof. Dr. Donald Matthew Pianto Dr. André Carlos Ponce de Leon Ferreira de Carvalho
EST/UnB ICMC/USP

Prof.a Dr.a Aletéia Patrícia Favacho de Araújo
Coordenadora do Programa de Pós-graduação em Computação Aplicada

Brasília, 22 de julho de 2019

Dedicatória

Dedico esse trabalho aos meus pais, família, namorada e amigos pela compreensão e apoio imprescindíveis para a realização desse trabalho.

Agradecimentos

Agradeço ao meu orientador Prof. Dr. Gladston Luiz da Silva, pela contribuição essencial no desenvolvimento do trabalho.

Aos professores do Programa de Pós-Graduação em Computação Aplicada (PPCA) pelos aprendizados ao longo do curso.

Aos meus colegas de curso, pelo apoio e compartilhamento de conhecimento ao longo dessa trajetória.

Resumo

O trabalho propôs a criação de um protótipo de ferramenta para auxiliar os consultores de uma consultoria de gestão empresarial no melhor entendimento e aprofundamento do problema de seus clientes bem como na tomada de decisões e proposições de soluções. Isso é feito a partir da automatização do processo de mineração de dados, podendo ser realizado com pouca necessidade de interação com o usuário. O desenho da ferramenta tomou como base conceitos e estudos de ferramentas disponíveis para automated machine learning realizados por meio de ampla revisão bibliográfica. A partir dos estudos, foi possível estruturar a lógica da ferramenta e suas funcionalidades. Essa lógica tem como base algumas das etapas do CRISP-DM, passando pelo entendimento dos dados, preparação, modelagem e avaliação. A validação da aplicabilidade da ferramenta foi feita utilizando bases de dados públicas. Os resultados mostram que com a utilização da ferramenta, mesmo com pouco conhecimento de mineração de dados, é possível construir modelos consistentes.

Palavras-chave: automated machine learning, SVM, support vector machine, machine learning, mineração de dados

Abstract

This work proposed the creation of a prototype tool to assist the consultants of a business management consultancy in the best understanding of the problem of its clients as well as in the decision making and propositions of solutions. This was done by automating the data mining process, which can be accomplished with little need for user interaction. The tool design was based on concepts and studies of available tools for automated machine learning supported by a wide bibliographic review. From the studies, it was possible to structure the logic of the tool and its functionalities. This logic is based on some of the steps of CRISP-DM, including data understanding, data preparation, modeling and evaluation. The tool applicability validation was done using public databases. The results show that with the use of the tool, even with little knowledge of data mining, it is possible to construct consistent models.

Keywords: automated machine learning, SVM, support vector machine, machine learning, data mining

Sumário

1	Introdução	1
1.1	Problema de Pesquisa	1
1.2	Justificativa	2
1.3	Objetivos	3
1.3.1	Objetivo Geral	3
1.3.2	Objetivos Específicos	3
1.4	Metodologia	4
1.4.1	Caracterização da Pesquisa	4
1.4.2	Estruturação da Pesquisa	5
1.5	Estrutura do trabalho	6
2	Fundamentação Teórica	7
2.1	<i>Machine Learning</i> e Mineração de Dados	7
2.2	<i>Automated Machine Learning</i>	8
2.2.1	Técnicas de otimização da configuração dos algoritmos	8
2.2.2	<i>Metalearning</i> como insumo para otimização dos parâmetros de algoritmo de <i>Machine Learning</i>	10
2.2.3	Ferramentas de <i>Automated Machine Learning</i> disponíveis	10
2.3	<i>Support Vector Machine</i>	13
2.3.1	Tipo do <i>kernel SVM</i>	15
2.4	<i>Pré-Processamento</i>	16
2.4.1	<i>Seleção de Atributos</i>	17
2.4.2	<i>Tratamento de missing values</i>	18
2.4.3	<i>Transformação de variáveis categóricas em numéricas</i>	19
2.5	Indicadores de Performance de Classificadores	20
3	Ferramenta Proposta para Automação do SVM e Resultados Obtidos	21
3.1	Análise Comparativa das Ferramentas de <i>Automated Machine Learning</i> . . .	21
3.1.1	Instalação	22

3.1.2	Bases de dados utilizadas	22
3.1.3	Configuração da Máquina utilizada	23
3.1.4	Análise Preliminar das Ferramentas	23
3.1.5	Análise dos Resultados das Ferramentas	23
3.1.6	Conclusões da Comparaçao de Ferramentas de AutoML	25
3.2	Automatização do Classificador	26
3.2.1	Entendimento dos Dados	27
3.2.2	Preparação dos Dados	27
3.2.3	Modelagem dos Dados	29
3.2.4	Avaliação dos Resultados	30
3.3	Desenvolvimento da Interface Gráfica da Ferramenta	30
3.4	Lógica de Utilização da Ferramenta	31
3.4.1	Entendimento dos Dados	31
3.4.2	Preparação dos Dados	32
3.4.3	Modelagem	33
3.4.4	Avaliação	33
3.5	Testes e Resultados Preliminares - Parte 1	35
3.6	Análise da Ferramenta	39
3.7	Testes e Resultados Preliminares - Parte 2	40
4	Conclusões	43
4.1	Trabalhos Futuros	45
A	Código Etapa Entendimento dos Dados	46
B	Código Etapa Preparação dos Dados	48
C	Código Etapa Modelagem	50
D	Código Etapa Avaliação	52
E	Código Interface Gráfica - Configuração de Tela	54
F	Código Interface Gráfica - Botões e Funcionalidades	55
G	Código Completo da Ferramenta	57
Referências		64

Lista de Figuras

1.1 Proposta de execução da ferramenta	4
1.2 Visão das Etapas da Metodologia	5
2.1 Representação cálculo margem SVM [1]	14
2.2 Exemplo de categorias interligadas	15
3.1 Erro instalação ATM	22
3.2 Tempo de processamento em minutos bases bases Iris e Wine	24
3.3 Tempo de processamento em minutos bases Abalone, Magic Gama e Adult .	24
3.4 Interface Gráfica Ferramenta - Etapa Entendimento dos Dados	32
3.5 Interface Gráfica Ferramenta - Etapa Preparação dos Dados - Parte 1	32
3.6 Interface Gráfica Ferramenta - Etapa Preparação dos Dados - Parte 2	33
3.7 Interface Gráfica Ferramenta - Etapa Avaliação	34
3.8 Fluxo de Atividades da Ferramenta	35
3.9 Limitação Interface Gráfica Ferramenta	40

Lista de Tabelas

2.1 Exemplo transformação dados ordinais	19
2.2 Exemplo transformação dados nominais - <i>One Hot Encoding</i>	20
3.1 Acurácia TPOT, Hyperopt e Auto Weka	25
3.2 Comparaçao ferramentas AutoML	26
3.3 Características das bases de dados testadas	37
3.4 Indicadores de Performance das Bases de Dados	37
3.5 Acurácia ferramenta desenvolvida e ferramentas de AutoML	38
3.6 Tempo processamento (em minutos) ferramenta desenvolvida e ferramentas de AutoML	39
3.7 Indicadores de performance utilizando validação cruzada	41
3.8 Comparaçao acurácia com e sem validação cruzada	41
3.9 Comparaçao tempo de processamento (em minutos) com e sem validação cruzada	41

Listings

A.1 Código fonte em Python para etapa de Entendimento dos Dados	46
B.1 Código fonte em Python para etapa de Preparação dos Dados	48
C.1 Código fonte em Python para etapa de Modelagem	50
D.1 Código fonte em Python para etapa de Avaliação dos Resultados	52
E.1 Código fonte em Python para criação e configuração inicial da tela da interface gráfica	54
F.1 Código fonte em Python para inserção dos demais botões que habilitam as funcionalidades via interface gráfica	55
G.1 Código completo da Ferramenta em Python	57

Capítulo 1

Introdução

1.1 Problema de Pesquisa

Segundo Witten *et al.* [2], o volume total de informações armazenados em bancos de dados dobra a cada 20 meses, além disso, mostram que a medida que a quantidade de dados disponíveis aumenta, a proporção de pessoas, empresas e organizações que consegue entendê-los diminui. Adicionalmente, Provost *et al.* [3] defendem que a capacidade de extrair conhecimentos úteis a partir dos dados, deve ser enxergado como um ativo estratégico. Campos [4], conhecido consultor de gestão empresarial, afirma que empresas e governos estão cheios de informação em seus computadores, mas não sabem o que fazer com elas. E assim como Provost *et al.* [3], conclui que o verdadeiro poder das organizações está no conhecimento que é extraído das informações. Diante desse cenário, trabalhar com bases de dados massivas, nem sempre estruturadas, vem se tornando um desafio para organizações públicas e privadas do mundo inteiro.

A empresa de consultoria de gestão empresarial que motivou o desenvolvimento do trabalho, atua no aperfeiçoamento da gestão das organizações para as quais trabalha, auxiliando-as a obter resultados de excelência e crescer de maneira sustentável. Em Brasília, atua basicamente em projetos de melhoria na administração pública, com diversos casos de sucesso em órgãos federais, como Ministério da Economia, Ministério da Educação, Ministério da Saúde, Banco Central do Brasil, TCU, entre outros. Com o objetivo de sempre gerar melhores resultados aos seus clientes públicos e, consequentemente, melhores serviços à população, a consultoria busca constantemente identificar oportunidades de melhoria em seu método de aperfeiçoamento de serviços públicos.

Atualmente, uma das dificuldades enfrentadas pelos consultores durante a execução dos projetos, está no grande volume de dados os quais os órgãos possuem. Essa limitação é causada principalmente pela dificuldade de se trabalhar com bases massivas e gerar informações e conhecimentos a partir das mesmas. Não são raros os casos onde há dispo-

nibilidade de bases de dados e são enfrentadas dificuldades de realizar análises e previsões baseadas nos dados existentes.

Um dos desafios da empresa é que a grande maioria dos mais de 200 consultores que fazem parte do corpo de colaboradores da consultoria não possui uma formação acadêmica que forneça conhecimentos e habilidades necessários para realização de análises de dados de forma ágil.

1.2 Justificativa

Consultoria de gestão consiste no atendimento personalizado a clientes em áreas específica como planejamento, inovação e melhoria de processos que auxiliam as empresas a melhorarem seu desempenho. Pode ser entendido de forma simplificada como o médico das organizações. Durante um projeto de consultoria em gestão empresarial, o consultor realiza uma análise extensa do cliente, tanto interna quanto externa, para formular as suas orientações à instituição. Esse diagnóstico bem detalhado é essencial para o entendimento do problema e geração de resultado para o cliente. É percebido também, que em diversas situações, um fator limitante para o diagnóstico é o fato da empresa cliente não conhecer bem seus dados e processos, o que pode restringir o diagnóstico da consultoria. Esses fatores, somados ao curto espaço de tempo para entendimento, podem limitar o resultado do projeto.

Dessa maneira, além das formas tradicionais já adotadas para a coleta de informações durante o diagnóstico do projeto, como por exemplo, entrevista e análises qualitativas, sugere-se a utilização de ferramentas de análise de dados para auxiliar na extração e análise de grande quantidade de dados de maneira automatizada, possibilitando um entendimento mais amplo e assertivo da situação do cliente. Acredita-se que, dessa forma, os dados serão analisados de maneira otimizada e possibilitarão à consultoria propor soluções com maior valor agregado.

Surgem assim, diversas possibilidades de aplicações dos conceitos de *Big Data* durante o projeto de consultoria, como por exemplo: mineração de texto para análise de sentimento da população com relação ao produto ou serviço prestado; mineração de processos para mapeamento da situação atual dos processos; mensuração do desempenho, identificação de gargalos ou inconformidades; análise de utilização de recursos humanos; mineração de dados para identificação de falhas, inconformidades, pontos de atenção e pontos de melhoria. Dessa maneira, espera-se obter um completo diagnóstico da situação do cliente antes da etapa de construção de soluções.

1.3 Objetivos

1.3.1 Objetivo Geral

No âmbito de Ciência de Dados, o objetivo desse trabalho é automatizar um classificador Máquina de Vetores de Suporte, do inglês, *Support Vector Machine* (SVM) e criar um protótipo de ferramenta que possa ser utilizada pelos consultores no contexto de consultoria empresarial, de modo a auxiliar no processo de tomadas de decisões baseado em informações. Espera-se que os consultores, a partir de uma base de dados fornecida pelo cliente, possam realizar a construção de classificadores sem a necessidade do conhecimento conceitual de Ciência de Dados.

1.3.2 Objetivos Específicos

Tendo em vista que a grande maioria do corpo de consultores não possui o conhecimento de Ciência de Dados, o projeto consiste na construção de uma arquitetura que automatize um classificador SVM que forneça *insights* e informações aos consultores a partir dos dados, sem a necessidade de possuírem conhecimentos técnicos aprofundados para isso, além de reduzir o tempo de análises. Dessa maneira, espera-se gerar melhores análises e soluções aos clientes. Para execução do trabalho, foram definidos os seguintes objetivos específicos:

- programar algoritmo para automatizar a parametrização do classificador classificador;
- desenvolver interface da ferramenta para utilização do usuário; e,
- testar ferramenta desenvolvida e analisar resultados.

A Figura 1.1 resume a proposta a ser desenvolvida, onde o consultor irá carregar a base de dados com a qual precisa realizar a classificação, responder algumas perguntas apresentadas pela, cujas respostas irão gerar, automaticamente, parâmetros para construção do modelo. Em seguida será realizado o treino e teste do classificador e o resultado será fornecido para o consultor complementar com outras análises do projeto.

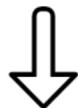
Figura 1.1: Proposta de execução da ferramenta



1) Consultor irá carregar a base no sistema e responder algumas perguntas de modo a se entender as características da base de dados e análises que devem ser feitas



2) A partir do input do usuário, o sistema selecionará os melhores parâmetros que maximize a performance do classificador



3) Após o processamento, será fornecido ao consultores os resultados do classificador para complementar as análises

1.4 Metodologia

1.4.1 Caracterização da Pesquisa

De acordo com Gil [5], uma pesquisa exploratória consiste no levantamento bibliográfico e testes de ferramentas para proporcionar maior familiaridade ao problema em estudo. Isto posto, essa pesquisa é caracterizada como exploratória, pois a base conceitual para realização do trabalho buscou levantamento bibliográfico de técnicas e conceitos de aprendizado de máquina, algoritmos de classificação e automatização de aprendizado de máquina e testes e comparações de ferramentas de *automated machine learning*. Com relação à natureza do trabalho, foi definida como Pesquisa Aplicada, que conforme definido por Silveira e Córdova [6], busca gerar conhecimentos para gerar aplicação prática, dirigido à solução de problemas específicos. O interesse da pesquisa consiste na aplicação de Mine-

ração de Dados para automatizar um classificador e permitir sua utilização em projetos de consultoria. Portanto, foi desenvolvida uma aplicação prática (ferramenta de automação) e é dirigido a um problema específico (desconhecimento de mineração de dados e programação por parte dos consultores).

1.4.2 Estruturação da Pesquisa

A execução da metodologia foi estabelecida a partir de quatro etapas: Escolha do Tema, Entendimento, Desenvolvimento e Avaliação conforme mostra Figura 1.2.

Figura 1.2: Visão das Etapas da Metodologia



A escolha do tema vai de encontro aos interesses profissionais do mestrando que atua como Consultor de Gestão em projetos para clientes do setor público e privado, e pelos interesses do Programa de Computação Aplicada da Universidade de Brasília (PPCA). Com vistas a se ambientar na temática e delimitar o escopo do trabalho foi realizada uma revisão de literatura preliminar em livros e artigos. Durante esse estudo preliminar, foi identificado um tema com relevância tanto para a empresa quanto para a academia.

Para a etapa de Entendimento, foi realizada uma ampla revisão de literatura baseada em livros e artigos científicos relacionada a Machine Learning, Mineração de Dados, Automated Machine Learning, técnicas de pré-processamento e indicadores de performance de classificadores. Essa revisão de literatura complementar, fundamentou a base teórica do trabalho. Também durante a etapa entendimento, foi realizado o mapeamento das principais ferramentas disponíveis no que diz respeito à AutoML e em seguida, o teste das mesmas.

Para a etapa de Desenvolvimento, foi realizada a idealização da ferramenta e codificação com base em dois aspectos: interface da ferramenta, que permite a navegação pelo usuário e a automação do classificador.

Por fim, a etapa de Avaliação buscou testar a ferramenta qualitativamente e quantitativamente com o apoio de cinco bases de dados públicas. Quantitativamente, analisou-se os

indicadores de performance dos modelos selecionados pela ferramenta e qualitativamente foi analisada a usabilidade e funcionalidade da ferramenta.

1.5 Estrutura do trabalho

A presente dissertação está dividido da seguinte forma. O Capítulo 1 contempla a delimitação do problema e sua relevância para a organização, os objetivos a serem cumpridos e a metodologia de pesquisa, detalhando as etapas executadas para garantir o cumprimento do objetivo do projeto.

No Capítulo 2 é apresentada uma revisão de literatura referente a algoritmos de mineração de dados, onde foram revisados e analisados os principais algoritmos de classificação; conceitos de *automated machine learning*, área da mineração de dados que estuda formas de automatizar o aprendizado de máquina e suas ferramentas disponíveis; técnicas de otimização da configuração de parâmetros de algoritmos de mineração de dados, etapa que mapeou diferentes formas de selecionar parâmetros para otimizar o desempenho do classificador; e técnicas de preparação de dados, etapa que permitiu subsidiar algumas das funcionalidades da ferramenta. A revisão de literatura auxiliou a mapear e entender as possíveis alternativas de execução do projeto e embasar a seleção do classificador e forma de seleção dos parâmetros, que servirão de insumo para as outras etapas do projeto.

O Capítulo 3 apresenta detalhadamente a proposta de ferramenta que automatiza o SVM, mostrando a lógica de desenvolvimento da ferramenta. São apresentados como foi realizada a automatização do classificador, o desenvolvimento da interface gráfica para permitir a interação com o usuário. Para subsidiar algumas decisões a respeito da ferramenta, foi feita uma análise comparativa das principais soluções de AutoML disponíveis *open source*. Por fim, são descritos os testes e resultados preliminares obtidos com a utilização da ferramenta.

Por fim, no Capítulo 4 são apresentadas as conclusões e sugestões de trabalhos futuros e para permitir a replicação e o aperfeiçoamento da ferramenta, os códigos desenvolvidos durante a construção da solução estão disponibilizados nos Apêndices A até G.

Capítulo 2

Fundamentação Teórica

2.1 *Machine Learning* e Mineração de Dados

Uma das grandes aplicações de Inteligência Artificial (IA) é o *Machine Learning*, ou em sua tradução, aprendizado de máquina. Tom Mitchell [7] caracteriza *Machine Learning* como programas que automaticamente melhoram com a experiência. Marsland [8] define como o processo de fazer o computador modificar ou adaptar suas ações para que obtenham maior acurácia, quando esta for a medida de desempenho adotada, onde acurácia representa a medida do quanto bem as ações escolhidas refletem as corretas. São portanto, algoritmos utilizados para construção de classificadores, reconhecimentos de padrões, análise de sentimentos etc. Esse campo da Ciência da Computação é responsável pela construção de algoritmos capazes de gerar aprendizado e realizar previsões. Ao longo dos últimos anos, vários algoritmos e abordagens de *Machine Learning* foram desenvolvidos e aperfeiçoados.

Adicionalmente, mineração de dados é definida por Witten *et al.* [2] como o processo de descoberta de padrões em dados, onde esse processo deve, preferencialmente, ser automatizado ou semi automatizado. Hong *et al.* [9] apresentam o conceito de mineração de dados como o processo de extração de conhecimento ou padrões a partir de bases de dados. Esses padrões identificados, devem auxiliar as organizações a entenderem melhor os seus clientes e processos com objetivo de identificar oportunidades de melhoria. Portanto, a mineração de dados deve auxiliar a entender o contexto do passado recente, para a partir disso possibilitar a tomada de decisões e resolução de problemas.

Uma pesquisa bibliográfica realizada na base de artigos *Web of Science* utilizando o argumento de pesquisa “*Machine Learning*” localizou mais de 48 mil artigos. Dentre esses artigos, as principais abordagens utilizadas são *Support Vector Machine*, Redes Neurais, Árvores de Decisão, *clustering* e Redes Bayesianas. Em uma pesquisa mais extensa, Wu *et al.* [10] mapearam os 10 principais algoritmos de mineração de dados utilizados na IEEE

International Conference on Data Mining 2006 em Hong Kong. Esses algoritmos estão entre os 10 mais influentes na comunidade acadêmica, a saber: C4.5, k-Means, SVM, Apriori, EM, PageRank, AdaBoost, kNN, Naive Bayes, and CART.

2.2 *Automated Machine Learning*

É percebido tanto na literatura quanto no mercado o crescente uso de modelos de machine learning para resolução de problema. A sua construção consiste no esforço de selecionar o algoritmo e testar seus parâmetros para maximizar a performance do modelo. Conforme apresenta [11], por conta do sucesso obtido com machine learning em aplicações diversas, surge cada vez mais frequente a demanda por sistemas de aprendizado de máquina que podem ser utilizados de forma imediata por não especialistas. Surgiu então o desenvolvimento de *Automated Machine Learning* que fornece métodos e processos que conseguem automaticamente gerar a melhor configuração de algoritmos de aprendizado de máquina sem a intervenção humana. Feurer *et al.* [11] mostram que AutoML automaticamente seleciona um bom algoritmo e etapas de pré-processamento para uma nova base de dados e também seleciona a melhor configuração dos hiperparâmetros para construção do modelo. Olson *et al.* [12] complementam esse conceito como sistemas que automatizam o processo de projetar e otimizar algoritmos de mineração de dados. Esse tipo de solução permite gerar vários benefícios para seus usuários, principalmente para aqueles que não possuem domínio conceitual de *Machine Learning* ou que não possuem tempo suficiente para realizar a parametrização dos algoritmos. Chung *et al.* [13] defendem que *Automated Machine Learning* é a solução ideal para a falta de cientistas de dados, pois pode aumentar drasticamente o desempenho e a produtividade dos cientistas de dados, precisão do modelo e, em última análise, até mesmo possivelmente substituir a necessidade de cientistas de dados.

2.2.1 Técnicas de otimização da configuração dos algoritmos

Os valores dos hiperparâmetros de um classificador tem impacto direto no sucesso dos algoritmos de mineração de dados e ajustar esses parâmetros para otimizar a performance do modelo não é uma atividade trivial [14]. A otimização dos hiperparâmetros, de acordo com Bergstra *et al.* [14] é o processo de procurar a partir de um conjunto de possíveis combinações das variáveis de um algoritmo que permitam que ele alcance melhores resultados. Adicionalmente, Kotthof *et al.* [15] defendem que as soluções mais robustas de *Automated Machine Learning* são em geral problemas de seleção de algoritmos e hiperparâmetros, denominados *CASH problem*, do inglês, *Combined Algorithm Selection and Hyperparameter*. Consiste então em um função de otimização onde a definição dos hiperparâmetros é

condicionada ao algoritmo que foi selecionado. A otimização dessa função é o componente principal de um sistema de *Automated Machine Learning*. Existem disponíveis, diversas formas de realizar a otimização dessa função. As mais utilizadas na literatura analisada são: Otimização Bayesiana, *Selection Criteria* e *Grid Search*.

As técnicas de Otimização Bayesiana consistem em modelos probabilísticos que capturam a relação entre as configurações dos hiperparâmetros e o desempenho do indicador. Em seguida, usa esse modelo para selecionar a configuração de hiperparâmetro mais promissora, avalia essa configuração de hiperparâmetro, atualiza o modelo com o resultado e repete [16]. De acordo com Brochu *et al.* [16], Otimização Bayesiana é uma das abordagens mais eficientes em termos do número de avaliações da função analisada. A estratégia de Otimização Bayesiana pode ser realizada de duas maneiras. Otimização baseada em processo gaussiano e otimização baseada em árvore. Por sua vez, a Otimização Bayesiana baseada em árvore pode ser realizada de duas maneiras. *Sequential Model-Based Algorithm Configuration* (SMAC) e *Tree Parzen Estimator* (TPE). Feurer *et al.* [11] defendem que a otimização dos parâmetros do classificador baseada em árvore é mais adequada para problemas com muitas dimensões, estruturados e parcialmente discretos, enquanto a otimização baseada em processo gaussiano é mais adequada quando se tem problema com poucas dimensões a hiperparâmetros numéricos.

Xuesi *et al.* [17] mostram que as abordagens de *evaluation criteria* ou *selection criteria* são baseadas no método *Leave-one-out* (LOO), o qual estima o limite superior das taxas de erro para encontrar os valores ótimos dos parâmetros. Conforme apresentado por Ayat *et al.*, [18] os principais modelos para *selection criteria* são *Generalized Approximate Cross-Validation* (GACV) e *Vapnik-Chernovenkis dimension* (VC). O GACV foi proposto por Wahba [19] e VC dimension por Vapnik e Chernovenkis [20]. Ayat *et al.* [18] demonstram que diferentemente do *VC dimension*, o GACV é uma função linear de otimização e dessa forma, é mais simples de ser calculada.

Além das técnicas mencionadas acima, existem diversas outras heurísticas e métodos de otimização. A forma mais extensa de busca é o *Grid Search*. Consiste em realizar uma busca extensiva em um conjunto finito de valores e manter aqueles que geram a menor taxa de erro. Ayat *et al.* [18] argumentam que o *Grid Search* não é fácil de ser executado sem um conhecimento a priori do problema. Adicionalmente, ressaltam que quando tem-se mais e dois hiperparâmetros, esse tipo de busca pode se tornar inviável. Por outro lado, Hsu *et al.*, [21] defendem a utilização da busca em *grid* argumentando que talvez não seja seguro a utilização de métodos que trabalham com aproximações ou heurísticas, pois não realizam uma busca exaustiva nos parâmetros. Hsu *et al.* [21] complementam a argumentação mostrando que o tempo requerido computacionalmente para localizar bons parâmetros com o *Grid Search*, dependendo do caso, não é muito maior do que outros

métodos avançados. Adicionalmente, caso os parâmetros sejam independentes, é possível paralelizar a busca, reduzindo o tempo necessário de processamento.

2.2.2 *Metalearning* como insumo para otimização dos parâmetros de algoritmo de *Machine Learning*

Brazdil et al. [22] definem *Metalearning* como o estudo de métodos que exploram o meta-conhecimento para obter modelos e soluções eficientes, adaptando processos de *Machine Learning* e Mineração de Dados. A partir desse conceito, é possível inferir que uma das possíveis aplicações do *Metalearning* seja para seleção de algoritmos ou parâmetros de um algoritmo de *Machine Learning*. Diversos autores utilizaram *Metalearning* como insumo para otimização dos parâmetros de um algoritmo. Kuba et al. [23] utilizaram *Metalearning* para auxiliar a encontrar relações entre as características de bases de dados e boas configurações de parâmetros. Utilizando informações de 14 meta *features* de 42 bases diferentes, foi possível prever a performance de novas bases de dados. Gomes et al. [24] utilizaram *Metalearning* para recomendar os parâmetros de um SVM baseado em uma série de meta exemplos armazenados em uma base de dados. Dessa forma, foi possível sugerir parâmetros iniciais para o classificador para na sequência, realizar o *Grid Search*. Buscando a validação da proposta, foi comparado com o desempenho utilizando otimização sem o *Metalearning*. Como conclusão, foi possível perceber que a primeira forma convergiu para soluções ótimas mais rapidamente.

2.2.3 Ferramentas de *Automated Machine Learning* disponíveis

É possível identificar algumas soluções tanto comerciais quanto open source que abordam o tema *Automated Machine Learning*. Uma das primeiras ferramentas desenvolvidas foi o Auto-Weka, por Thornton et al [25] e aperfeiçoado por Kotthoff et al. [15] em sua versão 2.0, quando foram corrigidos bugs da versão anterior, atualizado para as novas versões do Weka e Java e adicionadas novas funcionalidades. O Auto-Weka surgiu de evoluções do WEKA [26], plataforma *open source* de *Machine Learning* amplamente utilizada para mineração de dados. Assim como seu antecessor, o Auto-Weka possui uma interface para auxiliar o usuário que desconhece de programação. A proposta do Auto-Weka surgiu de uma necessidade dos usuários do WEKA, que tipicamente não sabem como escolher entre os diversos algoritmos disponíveis na ferramenta, bem como as configurações ideais para se obter um bom desempenho do modelo desenvolvido, tendo em vista o conjunto extenso de configurações alternativas. Dessa forma, utilizando Otimização Bayesiana baseada em árvore com *random-forest-based* SMAC, o Auto-Weka considera o espaço combinado dos algoritmos com seus hiperparâmetros para encontrar a combinação que minimiza o erro

de validação cruzada. O Auto-Weka é, portanto, uma ferramenta que realiza seleção de algoritmo e configuração dos parâmetros. A versão 2.0 contempla 15 algoritmos de classificação e 13 de regressão e sua principal vantagem é que sua interface permite a utilização mesmo de usuários sem conhecimento dos algoritmos e seus hiperparâmetros. As únicas opções que o usuário configura são: tempo limite de processamento, limite de memória e quantidade de processamentos em paralelo.

A partir das discussões do Auto-Weka, Komer [27] desenvolveu um sistema com objetivos similares, denominado Hyperopt-Sklearn, utilizando a biblioteca de aprendizado de máquina Scikit-learn [28], disponível em Python. Apesar das similaridades com o Auto-Weka, a principal justificativa para o desenvolvimento do Hyperopt-Sklearn, de acordo com Komer [27] é de que o Auto-Weka não foi desenvolvimento com vistas a ser uma solução escalável e, portanto, seriam necessárias soluções alternativas. O Hyperopt-Sklearn é constituído de seis classificadores: SVC, KNN, *Random Forest*, *ExtraTrees*, SGD e *Multinomial Naive Bayes* e cinco técnicas de pré-processamento: PCA, *StandardScaler*, *MinMaxScaler*, *Normalizer* e TF-IDF [14]. Para realizar a busca do classificador e dos seus parâmetros ideais, é possível selecionar entre três algoritmos de busca: *Random Search*, TPE ou *Annealing*. É intuitivo imaginar que realizar uma busca utilizando o conjunto de ferramentas de pré-processamento e classificadores pode demandar um alto volume de tempo e computacional. Dessa forma, o Hyperopt-Sklearn permite especificar um espaço de busca mais restrito, de forma a explorá-lo de forma mais profunda. É possível limitar tanto o espaço de classificadores, quanto módulos de pré-processamento quanto hiperparâmetros. Para seleção da melhor combinação de classificador e parâmetro, são utilizadas duas métricas de performance: f-measure e acurácia do modelo.

Posteriormente, Feurer *et al.* [11] evoluíram o Hyperopt-Sklearn implantando conceitos de *Metalearning* de forma a complementar a Otimização Bayeasiana já existente. A principal justificativa para essa mudança era de que a inicialização da Otimização Bayesiana era devagar para grandes espaços de parâmetros. Dessa maneira, a partir de conhecimentos de outras bases de dados, são selecionadas k configurações baseadas no *Metalearning* para serem utilizadas como parâmetros iniciais da Otimização Bayesiana. Feurer *et al.* [11] também compararam o desempenho do Hyperopt-Sklearn com o Auto-Weka para vinte e uma bases de dados diferentes, onde em seis tiveram desempenho superior, empate em doze e perda em três das bases.

Mais recentemente, foi proposto por Olson *et al.* [29], [12], outra ferramenta de *Automated Machine Learning*, denominada *Tree-based Pipeline Optimization Tool* (TPOT). A solução, utilizando uma versão da programação genética, se propõe a otimizar uma série de atributos de pré-processamento e o desenvolvimento de modelos de *Machine Learning* de forma a maximizar a acurácia do modelo. Possui quatro algoritmos de *Machine Le-*

arning (*Decision Tree*, *Random Forest*, *Extreme Gradient Boosting*, *Logistic Regression* e *Knearest Neighbor*) e doze ferramentas de pré-processamento, divididas em seleção de atributos e manipulação de dados [12].

Outra ferramenta foi proposta por Swearingen *et al.* [30], denominada Auto-Tuned Models (ATM), sistema distribuído, colaborativo e escalável para AutoML. Diferentemente das demais soluções, a otimização é realizada a partir de uma combinação da Otimização Bayesiana com processo gaussiano e otimização *Multi-Armed Bandit*. Nesta ferramenta, o utilizador seleciona o algoritmo de *Machine Learning* que deseja utilizar e são identificados os hiperparâmetros desse algoritmo, onde é construída uma árvore com todas as combinações de parâmetros possíveis. A partir da árvore, são criadas hiperpartições, onde para partição corresponde a um braço do *Multi-Armed Bandit*. Dessa forma, o processo gaussiano aprende sobre as partições e é capaz de identificar parâmetros que geram melhores resultados e, a partir disso, altera a distribuição dos braços do *multi-armed bandit*. Também é possível selecionar a medida de performance que irá avaliar o modelo.

A solução de *Automated Machine Learning* mais recente é o Auto-Keras, proposto por Jin *et al.* [31] do DATA Lab da Texas A&M University no final de 2018. Assim como os anteriores, busca auxiliar pessoas que não dominam as tecnologias de aprendizagem de máquina a utilizarem de forma simples. Porém, de acordo com Jin *et al.* [31], sua principal diferença é que o Auto-Keras foca em atividades de *deep learning*. Essa ferramenta, disponível em Python, promete treinar uma rede neural com apenas três linhas de código. Assim como o ATM, também utiliza Otimização Bayesiana com processo gaussiano, porém complementa o processo com uma *neural network kernel*.

Por sua vez, o Google também desenvolveu sua solução de *Automated Machine Learning*. A solução comercial permite a criação de modelos personalizados de *Machine Learning*. Possui a vantagem de fácil integração com outros aplicativos ou sites. A solução, disponibilizada na nuvem via Google Cloud, se divide em três produtos. AutoML *Natural Language*, voltado para classificação de documentos e textos; AutoML *Translation*, dedicado a tradução de documentos; e AutoML *Vision*, aplicado para processamento e classificação de imagens. Todas as informações a respeito da solução do Google foram encontradas no sítio da própria empresa (<https://cloud.google.com/automl/>). Não foram localizados artigos científicos que explicam o funcionamento e conceitos utilizados para desenvolvimento do Google Cloud AutoML.

Assim como o Google Cloud AutoML, o Driverless.AI também é uma plataforma comercial da H2O.ai, empresa de tecnologia voltada para Inteligência Artificial. Contempla desde o pré-processamento da base, com técnicas automatizadas de seleção de atributos, até a construção dos modelos e implantação do modelo. Pode ser implantado em nuvem ou localmente. Outro diferencial da ferramenta é que permite gerar visualizações e gráfi-

cos com base nas estatísticas dos dados, o que ajuda o usuário a entender mais facilmente seus dados e facilita na criação do modelo. Suas principais aplicações são para previsão de séries temporais e processamento de linguagem natural para classificação de textos. Todas as informações a respeito da solução do Google foram encontradas no sítio da própria empresa (<http://docs.h2o.ai/driverless-ai/latest-stable/docs/userguide/index.html>). Não foram localizados artigos científicos que explicam o funcionamento e conceitos utilizados para desenvolvimento do H2O Driverless.AI.

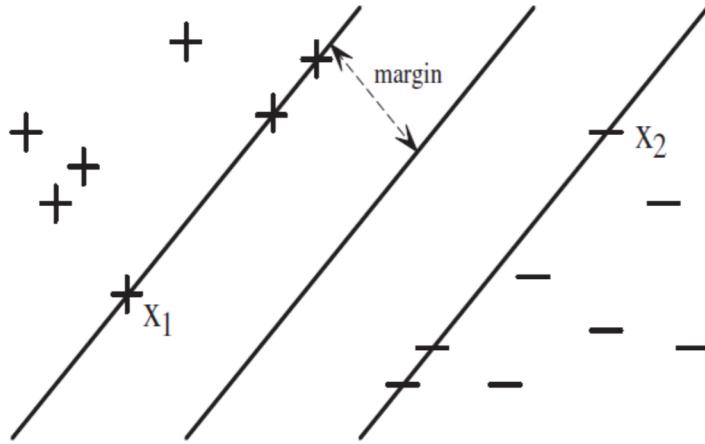
Por fim, a última ferramenta de AutoML identificada foi o DataRobot [32]. Também uma ferramenta comercial, se propõe a ser uma solução completa da empresa de mesmo nome. Possui processos automatizados desde o pré-processamento da base até a implantação do modelo e conta com algoritmos voltados para construção de modelos preditivos, classificações, regressões e mineração de texto. Além disso, também trabalha tanto com aprendizagem supervisionada quanto não supervisionada. Todas as informações a respeito da solução da Data Robot foram encontradas no sítio da própria empresa (<https://www.datarobot.com/wiki/>). Não foram localizados artigos científicos que explicam o funcionamento e conceitos utilizados para desenvolvimento do DataRobot.

2.3 *Support Vector Machine*

As máquinas de vetores de suporte, do inglês *Support Vector Machines* (SVM), propostas por Vapnik [33] é um dos algoritmos de aprendizagem mais populares [34], podendo ser utilizando tanto para problemas de classificação quanto de regressão a partir de dados estruturados. O objetivo do SVM é conceber uma maneira eficiente de aprender a separar hiperplanos em uma espaço de alta dimensão [35]. Dessa forma, o treinamento do SVM produz uma função que minimiza o erro de treinamento enquanto maximiza a margem que separa as classes de dados. A margem pode ser calculada como a distância perpendicular que separa o hiperplano e os hiperplanos gerados a partir dos pontos mais próximos, conforme Figura 2.1. A maximização da margem consiste em uma forma de reduzir a complexidade da solução [18].

Então, a partir de um conjunto de dados de treinamento, o SVM tentará generalizar e fazer previsões corretas sobre novos dados. Para os dados de treinamento, tem-se um conjunto de vetores de entrada, denotado x_i , com cada vetor de entrada tendo um número de características de seus componentes. Esses vetores de entrada são emparelhados com as categorias correspondentes, definida como y_i , com seus respectivos m pares ($i = 1, \dots, m$) [1]. Utilizando como exemplo uma base de dados onde os dados estejam classificados em duas classes, cada classe receberá valor -1 ou +1 e assim, cada registro da base y_i receberá um desses valores e a partir dos valores de x_i e y_i , o SVM definirá um hiperplano que

Figura 2.1: Representação cálculo margem SVM [1]



maximiza a distância entre as duas classes. O hiperplano de separação será dado por $w \cdot x + b = 0$, onde x representa os pontos dentro do hiperplano, w os pesos que determinam a orientação do hiperplano, b o viés ou deslocamento e $w \cdot x$ corresponde ao produto interno ou escalar.

Grande atenção tem sido dada ao SVM por conta de seus fundamentos e bom desempenho em diversos domínios quando comparado com outros algoritmos [34] e [23]. A partir de uma revisão de estudos relacionados ao SVM, Bennett e Campbel [36] mostram que diversas aplicações foram desenvolvidas com sucesso, desde identificação de partículas, identificação de rostos e categorização de texto até detecção de detonação de motores, bioinformática e marketing de banco de dados. Morgueza e Munoz [37] mostram que a vantagem do SVM com relação aos demais algoritmos é a possibilidade de determinar a função de classificação ou regressão a partir de uma pequena amostra, o que facilita sua aplicação em problemas com uma grande quantidade de dados, como processamento de texto ou bioinformática.

Porém, Cristianini e Shawe-Taylor [35] e Gomes *et al.* [24] reforçam que apesar do grande poder de generalização do SVM, a sua performance depende diretamente da escolha adequada dos seus parâmetros, incluindo a escolha da função *kernel* e os valores dos parâmetros do *kernel*, bem como a regularização desses parâmetros.

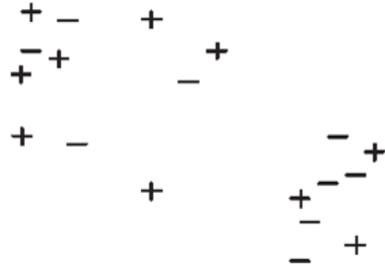
Com a evolução da utilização do SVM, foram desenvolvidas diversas variações de kernel. Hsu *et al.* [21] mostram que os principais tipos de *kernel* para o SVM são linear, polinomial e *radial basis function* (RBF)

2.3.1 Tipo do *kernel* SVM

O *kernel* consiste na função do SVM que irá auxiliar na aprendizagem do modelo. Dessa forma, mudar a função do *kernel* e seus parâmetros irá afetar o resultado do classificador. Apesar de existirem diversos tipos de *kernel* para o SVM, neste estudo serão analisados os mais utilizados na literatura. Além de Hsu *et al.*[21] que mostraram que os principais tipos de *kernel* para o SVM são linear, polinomial e *radial basis function* (RBF), Swearingen *et al.* [30] complementaram a lista dos *kernels* mais relevantes incluindo o *kernel* sigmoide. Por sua vez, Friedman *et al.* [38] citam *kernel* polinomial, *radial basis* e *neural network* (ou sigmoide) como os mais populares. Para cada *kernel*, é percebido pelo menos um parâmetro, cujo valor deve ser encontrado de forma a otimizar o resultado do classificador. Swearingen *et al.* [30] afirmam que definir o *kernel* mais adequado e a melhor configuração dos seus hiperparâmetros não é trivial. Portanto, entender cada *kernel* é relevante para selecionar o mais adequado para cada problema.

O *kernel* mais simples para o SVM é o linear. Esse tipo de *kernel* é definido como mais simples, pois gera hiperplanos lineares e, dessa maneira, só é possível separar as classes a partir de linhas. Ele é calculado a partir do produto interno de x e y , somado a uma constante c . A sua vantagem é que por conta da sua simplicidade, tende a resolver os problemas de otimização de forma muito mais rápida. Porém, sua performance, em geral, é inferior aos demais. Isso se deve pelo fato de que as classes podem ter características semelhantes, de forma que o *kernel* linear não consegue separar as categorias de forma eficiente, pois os dados não são linearmente separáveis. O exemplo da Figura 2.2 mostra um exemplo, de categorias interligadas, onde para duas categorias em um espaço bidimensional, não é possível utilizar um *kernel* linear.

Figura 2.2: Exemplo de categorias interligadas



Percebe-se, dessa forma, que a construção de modelos lineares pode ser um fator limitador. Foram desenvolvidos então, modelos cujas funções são não lineares. O *kernel* polinomial permite modelar funções polinomiais de grau n , aumentando a complexidade

da função e permitindo a construção de modelos lineares. É calculado a partir de três parâmetros: a constante C , o grau do polinômio d e o grau de declive α . Outro *kernel* muito utilizado para problemas de classificação é o *radial basis function* (RBF), também conhecido como *kernel* gaussiano. Ao mesmo tempo que o RBF pode gerar ótimos resultados, pode gerar resultados ruins caso não seja bem parametrizado. Sua grande vantagem é que a partir dos parâmetros σ e a constante C , permite criar hiperesferas para separar as classes. Hsu *et al.* [21] defendem que durante a construção do modelo, o RBF é uma boa primeira escolha, pois consegue trabalhar com relações não lineares que o *kernel* linear não trata e por ter menos parâmetros que o polinomial e, portanto, menos complexo. De forma complementar, Ali e Smith-Miles [34] realizaram um estudo empírico onde compararam *kernel* polimomial, RBF e linear e verificaram que o RBF obteve melhor performance média.

Por fim, o *kernel* sigmoide, também conhecido por *Hyperbolic Tangent Kernel* ou *Multilayer Perceptron Kernel*, tem sua origem a partir da teoria de redes neurais. É utilizada uma função sigmoide como função de ativação dos neurônios. Utiliza os parâmetros α e a constante C [39].

Após a análise do problema e definição do *kernel* a ser aplicado, uma das formas mais utilizadas de se encontrar os parâmetros ideais do *kernel*, segundo Campbell e Ying [1], é a partir da separação da base de dados em treino, validação e teste. Então, a partir da variação de opções de valores para os parâmetros do *kernel*, são selecionados aqueles onde são obtidos menores erros nos dados de validação.

Para casos onde não se tem conhecimento de qual *kernel* selecionar para o classificar, é possível utilizar uma abordagem mais completa, denominada *multiple kernel learning* (MKL), onde a partir de um conjunto de *kernels* candidatos, é selecionado aquele mais adequado para a base de dados com seus respectivos valores para os parâmetros. Essa abordagem visa reduzir o viés da seleção a partir da automação da análise.

2.4 Pré-Processamento

Na maior parte das vezes, os dados extraídos não estão prontos para serem trabalhados utilizando algoritmos de mineração de dados. Isso acontece frequentemente porque os dados coletados não são utilizados somente para mineração de dados. Eles podem servir para inúmeras outras utilidades e por isso, para tornar os dados possíveis de serem utilizados, pode ser necessária a adoção de técnicas de pré-processamento de dados. O pré-processamento pode ser utilizado para remover inconsistência dos registros, tratar dados ausentes, transformar dados incompatíveis com as características do algoritmo, etc.

Dessa maneira, busca-se remover atributos redundantes ou irrelevantes para o modelo que se deseja construir.

O principal objetivo do pré-processamento dos dados é prepara-los para a próxima etapa, de modelagem dos dados. Durante essa etapa, busca-se aumentar a qualidade dos dados, seja removendo problemas dos dados ou habilitando seu uso para as ferramentas de mineração. De acordo com Liu e Motoda [40], o pré-processamento pode ser dividido em três fases: uma de limpeza dos dados, onde serão removidas inconsistências e *outliers* nos dados; uma de transformação dos dados, processo de conversão dos dados de um formato para outro, com vistas a facilitar sua manipulação; e outra de redução dos dados, ou seleção de atributos, onde os dados serão transformados e reduzidos para seu correto uso de acordo com a aplicação que se deseja utilizar. Espera-se que ao final das duas fases os dados estejam tratados e preparados para modelagem.

2.4.1 Seleção de Atributos

Liu e Motoda [40] definem seleção de atributos como o processo que seleciona um conjunto ótimo de atributos de acordo com um determinado critério, ou seja, são técnicas utilizadas para selecionar os atributos mais relevantes de base de dados. Esse processo é relevante pois, conforme Chizi *et al.* [41], como resultado da seleção de atributos, a performance do modelo é melhorada, tanto para tempo de aprendizado, acurácia quanto simplicidade das regras. Adicionalmente, percebe-se que muitas vezes, por conta do grande volume de dados, esbarra-se na restrição computacional, o que dificulta o aprendizado. A seleção de atributos, a partir da remoção de atributos irrelevantes, auxilia na redução do tamanho da base de dados, pois são mantidos somente os atributos que mais contribuem para o classificador. Portanto, a seleção de atributos consiste em ferramentas de manipulação de dados sem alterar as características originais dos dados. Tendo em vista que o presente trabalho se propõe a automatizar um classificador, os esforços foram direcionados para estudo de técnicas de seleção de atributos voltadas para classificadores.

Foram identificadas três principais abordagens para seleção de atributos: *filter*, *wrapper* e *embedded* [41]. A primeira, é a abordagem mais antiga e mais simples consiste em utilizar heurísticas baseadas nas características dos dados. Trabalha independente do algoritmo de mineração de dados que será utilizado posteriormente e define o conjunto de atributos a partir do cálculo de informações dos dados. Consequentemente, tem por vantagem ser muito mais rápida do que as outras. Por sua vez os *wrapper methods*, de acordo com Kohavi e John [42], realizam uma busca por uma amostra ideal utilizando um algoritmo de indução como função para avaliar cada amostra de atributos. Uma série de conjunto de atributos são selecionados e para cada conjunto o classificador é gerado a partir dos dados dos atributos. A acurácia é registrada e aquele conjunto de atributos com

melhor acurácia é selecionado. Como consequência, apesar de obter melhor resultado que o anterior, tende a ser mais lento. Percebe-se que para bases de dados muito grandes, os *wrapper methods* podem ser tornar computacionalmente proibitivos. Por fim, os *embedded methods*, se caracterizam pela combinação dos dois anteriormente mencionados, de forma a obter melhores resultados combinando o melhor de cada um.

2.4.2 Tratamento de missing values

Durante a análise de dados e construção dos modelos, há casos onde alguns dos valores de variáveis pode estar ausente. Considerando que previamente à construção do modelo não sabemos qual o impacto dos dados ausentes no modelo, para esses casos, se faz necessário algum tratamento e manipulação dos dados de forma a torná-los mais adequados para construção do modelo. É preciso ter atenção ao se selecionar a técnica de tratamento, pois uma escolha errada, de acordo com Acock [43], pode produzir estimativas tendenciosas, estatísticas distorcidas e conclusões incorretas. Dessa maneira, a presente seção apresenta algumas das técnicas de tratamento de dados ausentes ou missing values mais utilizadas na literatura.

A técnica mais simples, conservadora e mais utilizada para tratamento de dados ausentes consiste em simplesmente excluir da base todo o registro quando alguma informação está ausente e o modelo será construído com os dados restantes que estiverem completos. De forma alternativa, é possível tratar os dados ausentes com o preenchimento da informação ausente a partir de algum determinado critério antes do treinamento do modelo. Friedman [38] argumenta que a primeira abordagem pode ser a melhor alternativa quando o volume de informações faltantes não é alto, caso contrário, deverá ser evitado, pois a base de dados ficaria muito reduzida e dificilmente representaria a população. Outra possibilidade, é a exclusão do atributo como um todo, ou seja, aquela variável é desconsiderada para todos os registros. Intuitivamente, é possível concluir que essa decisão deve ser tomada somente quando a maior parte dos dados desse atributo estão ausentes, caso contrário, é melhor utilizar outra abordagem. Dentre as abordagens de tratamento de dados faltantes por imputação de dados, em geral utilizam algum conceito estatístico. A mais comum consiste em adicionar o valor médio, moda ou mediana dos demais registros não faltantes. A substituição utilizando essas medidas estatísticas tem a vantagem de serem uma boa suposição para uma amostra aleatória, além da boa velocidade para execução. Porém, apesar de rápido, esse tipo de imputação de dados pode reduzir a variância dos dados. Outra forma mais complexa de tratar os dados ausentes é por meio da imputação múltipla. Alisson [44] demonstra que a imputação múltipla permite gerar ganhos superiores aos métodos tradicionais. A imputação múltipla permite agrupar as

estimativas de parâmetro para obter uma estimativa de parâmetro aprimorada. Múltiplas imputações produzem uma solução um pouco diferente para cada imputação.

2.4.3 *Transformação de variáveis categóricas em numéricas*

Frequentemente encontramos bases de dados com atributos numéricos e/ou categóricos, sendo que alguns algoritmos de mineração de dados suportam dados categóricos, enquanto outros, como o SVM, não. Dessa maneira, quando se deseja trabalhar com uma base com dados categóricos utilizando o SVM, se fez necessário uma etapa extra de transformação dos dados categóricos em dados numéricos.

Conforme Agresti [45], os dados categóricos podem ser classificados em dois tipos: ordinais ou nominais e para cada tipo pode haver um ou mais tratamentos. Os dados categóricos ordinais, são aquele os quais há de alguma forma, uma ordem implícita. Por exemplo, um atributo que possui os valores “ruim”, “bom” e “ótimo”. Eles são considerados dados ordinais, pois é possível ordenar esses valores em ordem crescente ou decrescente. Já para os dados categóricos nominais, não é possível estabelecer uma ordem para eles. É o caso de atributos com nomes de cidades, por exemplo. Dessa maneira, para dados ordinais ou nominais, podem existir diferentes formas de tratamento.

Para os dados ordinais, a partir do mapeamento da ordem entre eles, é possível definir valores numéricos para cada registro e realizar a substituição. Para o exemplo mencionado anteriormente, poderia ser feita a seguinte substituição dos dados ordinais conforme tabela 2.1 abaixo.

Tabela 2.1: Exemplo transformação dados ordinais

Dado categórico	Dado numérico
Ruim	-1
Bom	0
Ótimo	1

Já para os dados nominais, a abordagem bastante utilizada é o *One Hot Encoding*, que consiste em transformar cada atributo nominal em um conjunto de atributos binários com valores de 0 e 1. Para o exemplo das cidades, supondo que existam três registros, um com o valor “Brasília”, outro “São Paulo” e outro “Rio de Janeiro”, essa abordagem transformaria em três atributos conforme a tabela 2.2 abaixo

Uma restrição dessa abordagem é que para atributos com um grupo maior de valores, pode-se criar um número grande de atributos, o que pode gerar problemas de memória para o processamento, graças à alta cardinalidade.

Tabela 2.2: Exemplo transformação dados nominais - *One Hot Encoding*

Cidade_Brasilia	Cidade_SaoPaulo	Cidade_RiodeJaneiro
1	0	0
0	1	0
0	0	1

2.5 Indicadores de Performance de Classificadores

Essa seção apresenta o resumo dos principais indicadores de performance de classificadores encontrados na literatura. Os indicadores auxiliam na avaliação do resultado do classificador e quando combinados, e conforme mostra Kelleher [46], permitem determinar qual dos modelos desenvolvidos é mais adequado para uma determinada tarefa e estimar qual modelo performará melhor quando colocado em implantação. Portanto, podem auxiliar na comparação dos modelos.

Acurácia - é a medida mais comum e simples para avaliar um classificador. Ela define o percentual de instâncias classificadas corretamente. Kelleher [46] mostra que para bases de dados desbalanceadas, a acurácia pode gerar análises equivocadas. Uma medida alternativa é o cálculo da acurácia ponderado pelo tamanho de cada classe.

Curva ROC/AUC - A AUC - abreviação para “área sobre a curva” ou curva ROC, representa a precisão de um classificador e mostra o *tradeoff* entre a taxa de verdadeiros positivos e a taxa de falsos positivos. Quanto maior a área, melhor será o classificador. A curva em questão se refere à curva ROC (Receiver Operating Characteristic), que tenta mostrar a relação entre a sensibilidade e a especificidade de um método, sendo baseado na probabilidade de detecção, ou taxa de verdadeiros positivos e na probabilidade de falsos alarmes, ou taxa de falsos positivos. Witten et. al [2] defendem que a Curva ROC descreve o desempenho de um classificador com a vantagem de desconsiderar a distribuição de classes. Witten et. al [2] mostram que em geral as pessoas utilizam o cálculo da área sob a Curva ROC para mensurar o desempenho, onde quanto maior a área, melhor o modelo.

Recall – O Recall ou Taxa de Verdadeiros Positivos mostra o grau de confiança que temos com o modelo a respeito dos registros classificados como positivo. É calculado a partir da relação entre os registros classificados como verdadeiros dividido pelo total de registros de fato verdadeiros. [46]

Precisão - A precisão é o valor dado pelo número de elementos positivos que foram classificados corretamente em relação ao número total de elementos classificados como positivos. [46]

F1-Score - É a medida que combina precision e recall. Esta medida é definida a partir da média harmônica entre as duas variáveis.

Capítulo 3

Ferramenta Proposta para Automação do SVM e Resultados Obtidos

Para viabilizar a automatização do classificador para que os consultores possam utilizar durante projetos de consultoria, foi desenvolvido uma ferramenta com uma interface para navegação. Nesse Capítulo são detalhados os passos para elaboração da ferramenta, como utilizar a ferramenta após sua construção e os resultados preliminares obtidos com o uso da solução proposta.

A construção da ferramenta pode ser dividida em duas grandes etapas desenvolvidas de forma paralela. A primeira de construção do código que permitiu automatizar as etapas da construção do modelo e a segunda de desenvolvimento da interface da ferramenta onde o consultor poderá realizar todo o processamento.

Préviamente ao desenvolvimento da ferramenta, foi realizado estudo comparativo das soluções de AutoML disponíveis. Essa etapa foi importante para verificar pontos positivos que poderiam ser aproveitados e *gaps* que as outras ferramentas possuem e que poderia ser aperfeiçoado.

3.1 Análise Comparativa das Ferramentas de *Automated Machine Learning*

Tendo em vista que a aplicação principal do Auto-Keras, Google Cloud AutoML e H2O Driverless.AI não é voltada para construção de classificadores, essas não serão analisadas em detalhes no presente estudo. Complementarmente, por ser uma ferramenta comercial,

o DataRobot não foi analisado. Dessa maneira, foram analisados o Auto-Weka, Hyperopt-Sklearn, TPOT e ATM.

3.1.1 Instalação

De modo a testar as soluções *open source* de *Automated Machine Learning* identificadas na seção anterior, foram instalados alguns pacotes. Para o Hyperopr-Sklearn e o TPOT, bastou a instalação do pacote para Python para começar sua utilização. Para poder utilizar o Auto-Weka, se faz necessária a instalação prévia do Weka e posterior instalação do complemento do Auto-Weka. Apesar de ser um pouco mais demorado que os anteriores, a instalação se mostrou simples. Por sua vez, a instalação do ATM foi mais difícil que das demais soluções. A dificuldade ocorreu por conta de uma dependência do pacote “btb” disponível para Python durante instalação do ATM. Ao solicitar a instalação do ATM, não é possível instalar o btb, conforme pode ser percebido na Figura 3.1.

Figura 3.1: Erro instalação ATM

```
(base) C:\Users\USUARIO>pip install atm
Collecting atm
  Using cached https://files.pythonhosted.org/packages/69/9a/83b34f647fdf191ce2c1f41d6c8a07d76d6b607ac29de7e925e0d9f118f/atm-0.0.9-py3-none-any.whl
Requirement already satisfied: pyyaml>=3.12 in c:\users\usuario\anaconda3\lib\site-packages (from atm) (3.12)
Requirement already satisfied: numpy>=1.13 in c:\users\usuario\anaconda3\lib\site-packages (from atm) (1.16.1)
Collecting btb>=0.0.1 (from atm)
  Could not find a version that satisfies the requirement btb>=0.0.1 (from atm) (from versions: )
No matching distribution found for btb>=0.0.1 (from atm)

(base) C:\Users\USUARIO>
```

Ao buscar entender o motivador do problema em diversos fóruns relacionados ao assunto, chegou-se à conclusão que esse pacote não está mais disponível para a versão utilizada do Python. Dessa maneira, não foi possível testar o Auto Tuned Models.

3.1.2 Bases de dados utilizadas

A fim de testar e comparar as ferramentas disponíveis para AutoML, foram selecionadas cinco bases de dados públicas disponíveis no UCI Machine Learning Repository (<https://archive.ics.uci.edu/ml/datasets.php>) acessadas em 09 de abril de 2019.. A primeira denominada *Iris Plants Dataset*, contém registros de cento e cinquenta plantas divididas igualmente em três classes (setosa, versicolour e virginica) e compostas por quatro atributos. A segunda base de *Wine dataset*, consiste em classificar qualidade de vinhos a partir de informações químicas e é composto por cento e setenta e oito registros, divididos em treze atributos e três classes. A terceira base (*adult*), extraída do censo de 1994 contém 48.842 registros divididos em quatorze atributos e duas classes. A quarta base (abalone) contém 4.177 registros de um molusco marinho contendo oito atributos e vinte e oito classes. A última base utilizada (Magic Gama) possui informações de partículas visualizadas em um telescópio, contendo 19.020 registros, duas classes e onze atributos.

3.1.3 Configuração da Máquina utilizada

Para rodar o Auto-Weka, Hyperopt-Skearn e o TPOT, foi utilizada uma máquina com Windows 10 Pro, processador Intel Core i7-7700HQ 2.8GHz, 32gb de memória ram, 64 bits e placa de vídeo Nvidia GeForce GTX1050.

3.1.4 Análise Preliminar das Ferramentas

Antes da sua utilização, foram definidos critérios qualitativos para comparação das soluções disponíveis. As ferramentas estudadas foram comparadas utilizando os seguintes critérios: necessidade de programação, necessidade de conhecimento de Mineração de Dados, existência de etapas de pré-processamento, quantidade de algoritmos disponíveis e indicadores de performance disponíveis.

Do ponto de vista da necessidade de programação, Hyperopt-Sklearn, TPOT e ATM demandam conhecimentos intermediários de Python. O AutoWeka, por sua vez, possui uma interface gráfica que dispensa o conhecimento de linguagem de programação. Apesar disso, permite a utilização via código. Nesse item, o Auto-Weka se mostra mais flexível, podendo ser utilizado em ambas as situações. Para as demais ferramentas, foram encontrados tutoriais e códigos já prontos, que podem auxiliar programadores menos experientes.

Comparando do ponto de vista de conhecimentos de Mineração de Dados, as quatro ferramentas analisadas (Auto Weka, Hyperopt-Sklearn, TPOT e ATM) demandam conhecimentos prévios de Mineração para seleção de alguns parâmetros. Esse fato pode ser positivo caso queira customizar a aplicação, porém torna-se impeditiva a utilização por pessoas que desconheçam a teoria de Mineração de Dados.

O Hyperopt-Sklearn, AutoWeka e TPOT apresentam eficientes técnicas de pré-processamento, que podem ser utilizadas conforme necessidade do usuário. Porém, o ATM não possui essa funcionalidade, sendo necessária a utilização de pacotes adicionais e de forma manual.

Do ponto de vista da quantidade de algoritmos disponíveis, todas as ferramentas possuem ao menos 6 classificadores implementados. O TPOT apresenta algoritmos mais encontrados na literatura, como Decision Tree, Random Forest, Knearest Neighbor etc. O Auto Weka sem mostrou a ferramenta mais completa nesse quesito, com 15 classificadores implementados.

3.1.5 Análise dos Resultados das Ferramentas

Do ponto de vista quantitativo, as ferramentas foram comparadas utilizando de dois critérios: tempo de processamento e resultados dos indicadores de performance do melhor

modelo. Os tempos de processamento de cada base em cada ferramenta, em minutos, estão representados nas Figuras 3.2 e 3.3.

Figura 3.2: Tempo de processamento em minutos bases Iris e Wine

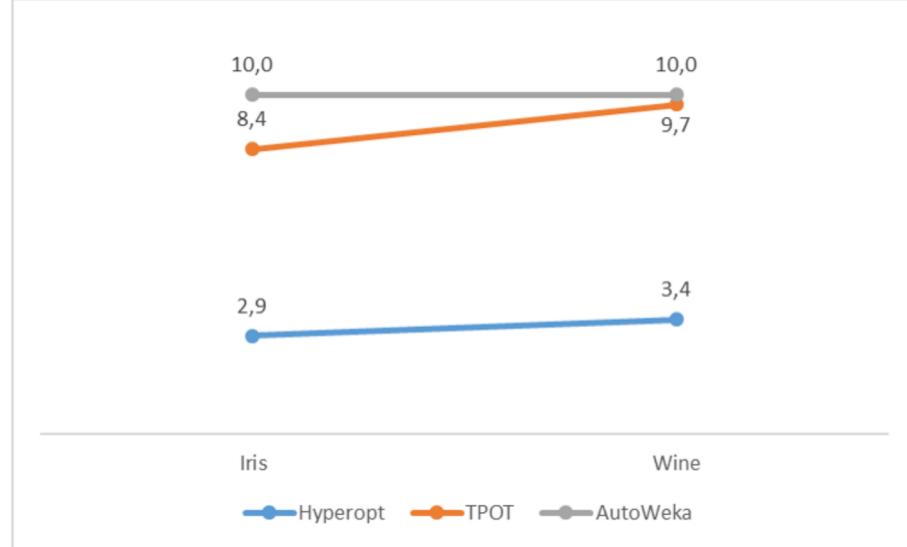
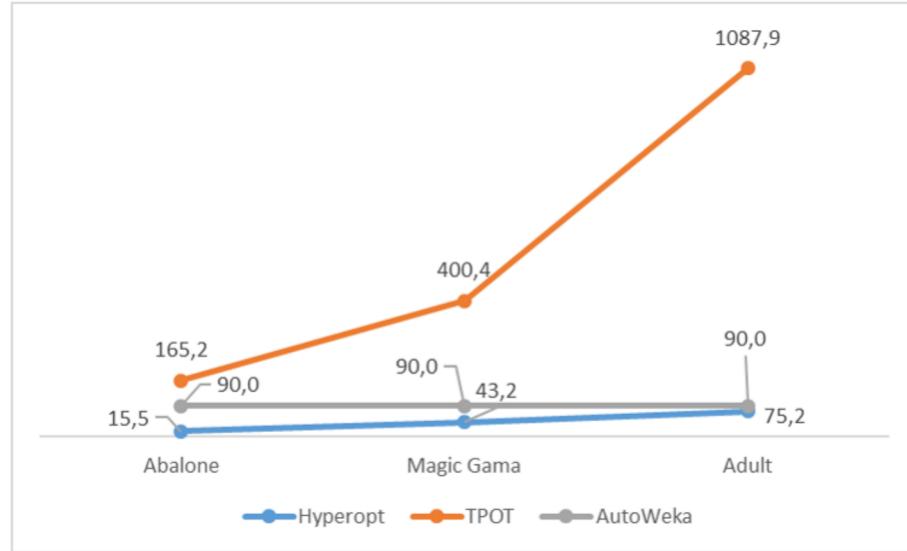


Figura 3.3: Tempo de processamento em minutos bases Abalone, Magic Gama e Adult



Considerando que o Hyperopt-Sklearn permite maior customização dos parâmetros, foi possível reduzir o espaço de busca e consequentemente, menor tempo de processamento que os demais. O Auto-Weka tem uma diferença onde o tempo de processamento é um dos parâmetros iniciais definidos. Por isso, seu tempo de processamento para as bases Iris e Winebase no maior tempo entre o TPOT e Hyperopt. Foi atribuído para o Auto-Weka,

tempo limite de dez minutos de processamento, enquanto para as bases Abalone, Magic Gama e AutoWeka, foi definido 90 minutos.

Um fato que chamou atenção é o crescimento exponencial do tempo de processamento do TPOT a medida que a complexidade da base é aumentada. A fim de validar se o tempo de processamento impacta na qualidade do classificador, foi mensurada a acurácia de cada classificador e os resultados estão apresentados na Tabela 3.1.

Tabela 3.1: Acurácia TPOT, Hyperopt e Auto Weka

	Hyperopt	TPOT	AutoWeka
Iris	0,9	0,96	0,98
Wine	1	0,972	0,994
Abalone	0,27	0,24	0,33
Adult	0,76	0,86	0,85
Magic Gama	0,83	0,86	0,98

Os resultados mostram que para a Base Wine, o Hyperopt teve um desempenho ligeiramente inferior que os demais, porém desempenho superior para a base Wine. Isso mostra que mesmo com tempo bem inferior de processamento, não houve perda considerável de qualidade. Para a base Abalone, todas obtiveram resultados ruins, porém o AutoWeka performou ligeiramente melhor. Por fim, para as bases Adult e Magic Gama, ambas as bases tiveram resultados bons, porém TPOT e AutoWeka tiveram melhores resultados, respectivamente.

3.1.6 Conclusões da Comparação de Ferramentas de AutoML

A análise das ferramentas nos mostra que o Auto Weka possui como vantagens a maior quantidade de algoritmos disponíveis, sendo portanto, uma ferramenta mais completa e consequentemente, tende a obter melhor performance. Possui como vantagem também a possibilidade de utilização sem necessidade de programação, o que permite a utilização por um público maior.

O Hyperopt-Sklearn, por sua vez, apresenta vantagens de maior possibilidade de customização e maior velocidade de processamento quando comparado aos demais. Complementarmente, o Hyperopt possui maior facilidade de integração com outras ferramentas.

Por fim, o TPOT se mostrou inferior aos demais para as bases analisadas, onde o maior tempo de processamento não se traduziu em um melhor desempenho do classificador.

A Figura 3.2 apresenta a análise comparativa dos critérios qualitativos.

Tabela 3.2: Comparaçāo ferramentas AutoML

	Auto Weka	Hyperopt	TPOT	ATM
Instalação	Simples	Simples	Simples	Difícil
Necessidade de programação	Não	Sim	Sim	Sim
Quantidade de algoritmos	28 (15 classificadores)	6	6	14
Indicadores de Performance	Diversos	Acurácia e F-measure	Acurácia	F-measure

Com o exposto, apesar do AutoWeka ser uma ferramenta mais robusta, entende-se que para o contexto ao qual está sendo trabalhado, o Hyperopt-Sklearn é a melhor opção por conta da sua facilidade de customização e possibilidade de aproveitamento de funções em Python.

3.2 Automatização do Classificador

A utilização da ferramenta foi pensada utilizando como base as etapas do processo CRISP-DM (*Cross-Industry Standard Process for Data Mining*) e, consequentemente, as etapas de desenvolvimento da ferramenta. De acordo com Azevedo e Santos [47] o CRISP-DM foi desenvolvido a partir dos esforços da DaimlerChrysler, SPSS e NCR e é composto por seis etapas:

- **Entendimento do Negócio** - A primeira etapa do CRISP-DM consiste em definir os objetivos e requisitos do projeto olhando do ponto de vista do negócio e assim, transformar em um problema de Mineração de Dados.
- **Entendimento dos Dados** - A etapa de entendimento dos dados inicia com a coleta dos dados e posterior análises para entender as características dos dados e potenciais problemas dos dados.
- **Preparação dos Dados** - A partir da base de dados inicial, a preparação dos dados consiste nas atividades de limpeza e transformação os dados para se obter a base de dados final para modelagem.
- **Modelagem** - Nessa fase, diversas técnicas de modelagem são aplicadas para se definir os parâmetros de cada algoritmo e obter o modelo.
- **Avaliação** - Análise dos resultados do(s) modelo(s) desenvolvido(s).

- **Implementação** - Fase final, de disponibilizar o modelo selecionado para produção.

A primeira decisão ao iniciar o desenvolvimento do código de automatização do classificador, foi escolher a linguagem de programação. Após estudo das linguagens mais utilizadas para Mineração de Dados, ficou-se entre duas alternativas. R e Python. Ambas as linguagens possuem inúmeros pacotes específicos para Mineração de Dados e que poderiam atender as necessidades do desenvolvimento da ferramenta. Porém, os principais fatores que influenciaram na decisão da utilização do Python, foram a possibilidade de desenvolver a interface e automatização na mesma linguagem e o fato de que algumas das ferramentas de *Automated Machine Learning* existentes atualmente foram desenvolvidas em Python, o que poderia facilitar no aproveitamento de códigos já existentes.

3.2.1 Entendimento dos Dados

Para a utilização da ferramenta, parte-se de premissa que a etapa de Entendimento do Negócio será feita pelo consultor previamente à utilização da ferramenta. Dessa maneira, definido o problema de Mineração de Dados, a ferramenta será utilizada a partir da etapa de entendimento dos dados. Para auxiliar o consultor no entendimento dos dados, ao carregar a base de dados, será possível visualizar as seguintes informações:

- quantidade de registros na base;
- quantidade de atributos na base;
- tipo de dado de cada atributo;
- resumo estatístico dos atributos numéricos.

Essas informações permitirão entender as características da base de dados e se existem outliers na base que podem distorcer as análises ou até mesmo se existem dados preenchidos incorretamente. A Listing A.1, representa o código utilizado para a etapa de Entendimento dos Dados.

Para o entendimento dos dados foram utilizados três pacotes Python: Tkinter, Pandas e Numpy. O Tkinter, já vinculado à interface, permite a abertura da janela para carregamento da base de dados, enquanto as bibliotecas Numpy e Pandas, permitem as análises estatísticas e manuseio da base.

3.2.2 Preparação dos Dados

Após a obtenção dos dados e seu entendimento, passo importante é realizar a "limpeza" e seleção dos dados para realizar a Mineração de Dados. A preparação é importante para

fornecer os dados da melhor maneira possível para que a performance do classificador seja otimizada. Para a ferramenta conseguir realizar a modelagem, o consultor precisará definir dentro da base de dados, qual a variável resposta, ou seja, qual o atributo que refere-se às classes do problema que ele pretende trabalhar.

Além disso, foi percebido que em muitas situações, o consultor recebe uma base de dados pura extraída dos sistemas do cliente e, portanto, pode possuir atributos que não serão utilizados para a modelagem. Dessa maneira, buscando evitar que atributos desnecessários interfiram na modelagem, a ferramenta permite que o consultor, com o conhecimento do negócio e dos dados, exclua aquelas variáveis que não serão relevantes para o modelo.

Outra situação que é possível encontrar nas bases de dados, refere-se aos dados ausentes. Dessa maneira, a etapa seguinte do desenvolvimento foi responsável por automatizar a seleção dos registros que permanecerão na base. De acordo com Acock [43], a escolha incorreta da técnica de tratamento de dados ausentes pode produzir estimativas tendenciosas, estatísticas distorcidas e conclusões incorretas. Tendo em vista as principais alternativas de tratamento de dados ausentes mencionadas na seção 2.4.2, optou-se pela utilização daquela que é a mais conservadora, que consiste em simplesmente excluir os registros ausentes. Apesar do risco de perda de informação, acredita-se que a alternativa mais segura.

Alguns algoritmos de Mineração de Dados conseguem trabalhar com dados numéricos, outros com dados categóricos. O SVM, algoritmo utilizado para construção do modelo trabalha somente com atributos numéricos. Porém, as bases de dados podem ter tanto atributos numéricos como categóricos. Dessa maneira, será necessário transformar as variáveis categóricas em um formato que o SVM conseguirá trabalhar melhor. A opção escolhida para realizar essa transformação foi o *One-Hot-Encoding*, que é o processo de transformar cada atributo nominal em um conjunto de atributos binários com valores de 0 e 1, onde o valor 0 significa que aquele valor não está presente na base e o valor 1 está presente na base.

Por fim, para finalizar a etapa de pré-processamento a base de dados é separada em treino e teste que serão utilizadas na etapa anterior. A base de treino fica com 80% da base e 20% para teste. A Listing B.1, representa o código utilizado para a etapa de Preparação dos Dados.

Para a preparação dos dados foram utilizados quatro pacotes Python: Tkinter, Pandas, Numpy e Scikit-Learn. O Tkinter, vinculado à interface, permite a interação com usuário onde fornece *input* para definição da variável classe e exclusão de colunas, as bibliotecas Numpy e Pandas, permitem a limpeza e manipulação da base e a biblioteca Scikit-Learn possibilita a separação da base de treino e teste.

3.2.3 Modelagem dos Dados

Para a modelagem dos dados com o *Support Vector Machine*, foi utilizado o pacote Hyperopt-Sklearn, que auxilia na seleção automática dos parâmetros do SVM. Ao comparar as possibilidades de utilização para modelagem dos dados, foi escolhido o Hyperopt-Sklearn por conta da sua possibilidade de customização e facilidade de uso. Komer et al. [27] mostram que para a utilização do Hyperopt o usuário define três parâmetros: o domínio de busca, a função objetivo e o algoritmo de otimização. O desafio dessa etapa, portanto, foi automatizar a definição desses parâmetros. O domínio de busca consiste nos parâmetros de busca para otimizar a função objetivo que consiste em percorrer o espaço de busca, analisar e retornar a configuração de parâmetros com a melhor performance. A função objetivo é baseada no método de validação de dados do pacote Scikit-Learn, chamado de *zero-one-loss*, o qual atribui 0 de perda para uma classificação correta e 1 para uma classificação incorreta. A função objetivo *fmin*, portanto, busca minimizar o erro de cada modelo construído. Para o algoritmo de otimização, o Hyperopt oferece três alternativas: *Random Search*, *Annealing Search* e *Tree Parzen Estimator*. Conforme visto na seção 2.2.1 as técnicas de otimização da configuração dos algoritmos utilizando Otimização Bayesiana mostra-se eficiente para problemas de natureza distintas e tem a vantagem quando comparada à busca randomiza por conta da velocidade de processamento. Dessa maneira, foi selecionado o TPE como algoritmo de busca.

Como parte da definição do domínio de busca, o Hyperopt oferece um conjunto de funcionalidades de pré-processamento. Tendo em vista que o pré-processamento da base já foi realizado na etapa de preparação dos dados, não foi utilizada essa alternativa. Adicionalmente, o Hyperopt oferece um conjunto de outros classificadores. Considerando que nesse momento será testado somente o SVM, os demais classificadores disponíveis não foram utilizados. Com relação ao kernel do SVM, considerando a variedade de problemas e bases que a ferramenta pode receber, optou-se por uma abordagem mais completa, denominada *multiple kernel learning* (MKL) e portanto, foram utilizados os quatro tipos disponíveis: linear, RBF, sigmoide e polinomial. Komer et al. comprovam [27] que percorrer todo o domínio pode ser computacionalmente extensivo e, portanto, de forma a reduzir o tempo total de processamento, foi definido que a modelagem será limitada a até 100 modelos e cada um terá no máximo 60 segundos para construção. Acredita-se que com 100 modelos desenvolvidos, será possível selecionar aquele com a melhor performance com um resultado satisfatório. A Listing C.1, representa o código utilizado para a etapa de Preparação dos Dados.

3.2.4 Avaliação dos Resultados

A correta interpretação e avaliação dos resultados é essencial para uma boa tomada de decisão. Para a ferramenta desenvolvida, a avaliação dos resultados é realizada inicialmente a partir das métricas de desempenho do melhor modelo. A métricas selecionadas foram: acurácia, *precision*, *recall*, área sob a curva ROC e F1-score. Espera-se que com o conjunto de métricas, somado com o auxílio na análise dos resultados, permita ao consultor a melhor tomada de decisão se o modelo é válido ou não. Além das métricas de desempenho, a ferramenta retorna ao consultor as informações referentes ao melhor modelo para caso se deseje implementá-lo. A Listing D.1, representa o código utilizado para a etapa de Avaliação dos Resultados.

Para a avaliação dos resultados foi utilizado o pacote Scikit-Learn para representar as métricas e o Hyperopt-Sklearn para representar os hiperparâmetros do melhor modelo.

3.3 Desenvolvimento da Interface Gráfica da Ferramenta

Paralelamente à execução da etapa de Automatização do classificador, foi construída a interface gráfica da ferramenta que permitirá a interação com o usuário. Para suportar essa etapa, foram testadas duas formas de construção. Programação em Django ou utilizando o pacote Tkinter. O estudo das duas alternativas mostrou que a programação em Django gerava mais liberdade para construção, pois permitia maior customização e *layout* mais amigável para o usuário final. Além disso, tinha a vantagem de a ferramenta poder ser disponibilizada em versão *web*. Em contrapartida, o Tkinter apesar das vantagens mencionadas para o Django, possui a facilidade de utilização tanto para Linux, Windows ou Mac, além de maior facilidade de desenvolvimento e, portanto, seu custo de construção era consideravelmente menor quando comparado ao Django. Diante dos motivos expostos, dada a natureza de prototipação da ferramenta, optou-se pela programação utilizando o Tkinter. Conforme explicam Welch et al. [48], a biblioteca Tkinter consiste em uma série de comandos para desenvolvimento e manipulação de *widgets*, definido como uma janela em uma interface gráfica do usuário (GUI) que possui uma aparência e comportamento específico. Com ele é possível incluir botões, menus, caixas de texto etc.

O primeiro passo para a construção da interface é a criação da tela, onde são definidos o tamanho da tela, título e configuração de cor conforme apresentado na Listing E.1. Foi configurado para que o tamanho da janela da interface seja adaptável de acordo com a tela do usuário. A inclusão de objetos na tela do Tkinter é feita fornecendo a posição do elemento, o tipo do elemento (botão, caixa de mensagem, etc) e sua configuração

(tamanho, cor, etc). A posição é definida como se a tela fosse um *grid* imaginário e é informada a posição da linha e coluna onde se deseja inserir o objeto. Para os botões, é possível vinculá-los a uma função, que são os códigos desenvolvidos na etapa anterior.

Na sequência, foram inseridos os botões que chamam as funções da ferramenta. Buscando minimizar a chance de erro na utilização da ferramenta, inicialmente aparecem somente o botão que irá permitir que o usuário faça o upload do arquivo contendo a base de dados e um botão para encerrar a aplicação. À medida que o usuário vai percorrendo as etapas do CRISP-DM e trabalhando com a base de dados, os demais botões vão sendo habilitados para uso.

De forma complementar, a medida que as funções vão sendo executadas, os resultados são disponibilizados na tela para o usuário. Não serão apresentados o código detalhado, porém a codificação completa da ferramenta está apresentada no Apêndice A.

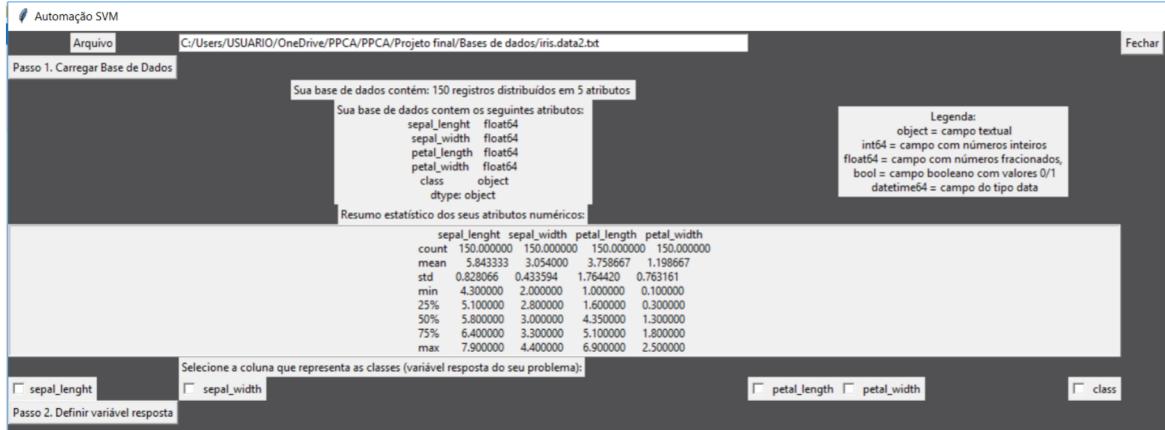
3.4 Lógica de Utilização da Ferramenta

Essa seção apresenta a lógica de utilização da ferramenta. Conforme mencionado nas seções anteriores, a ferramenta foi construída baseando-se nas etapas do CRISP-DM, para que a sua utilização também fosse dessa maneira. Assim sendo, após o consultor realizar a etapa de Entendimento do Negócio e definir o problema de Mineração de Dados, ele estará habilitado para utilização da ferramenta. Ou seja, parte-se da premissa de que para uso da ferramenta, o consultor já passou pela etapa de entendimento do negócio e a ferramenta já começa com o entendimento dos dados

3.4.1 Entendimento dos Dados

Ao abrir a ferramenta, o consultor precisará inserir a sua base de dados na ferramenta, que permite o carregamento de arquivos em formato .txt, .xlsx ou .csv. Ao carregar a base de dados, será apresentado ao consultor, informações gerais da base como quantidade de atributos e registros presentes na base fornecida, tipo de cada atributos e resumo estatístico dos atributos numéricos. As informações apresentadas para subsidiar o entendimento dos dados estão representadas na Figura 3.4.

Figura 3.4: Interface Gráfica Ferramenta - Etapa Entendimento dos Dados

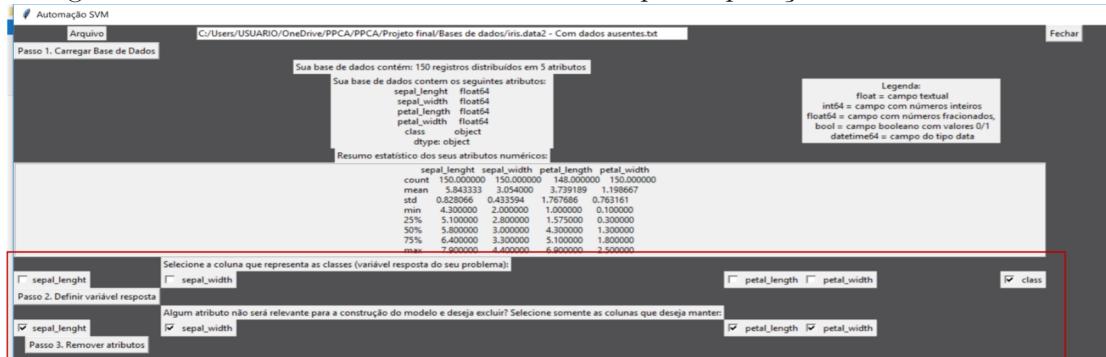


Tendo em vista o potencial desconhecimento dos usuários a respeito de alguns dos conceitos apresentados na ferramenta, em algumas situações foram adicionados comentários para facilitar a interpretação das informações apresentadas na ferramenta. O primeiro comentário adicionado diz respeito à explicação de cada tipo de dado conforme apresentado na Figura 3.4.

3.4.2 Preparação dos Dados

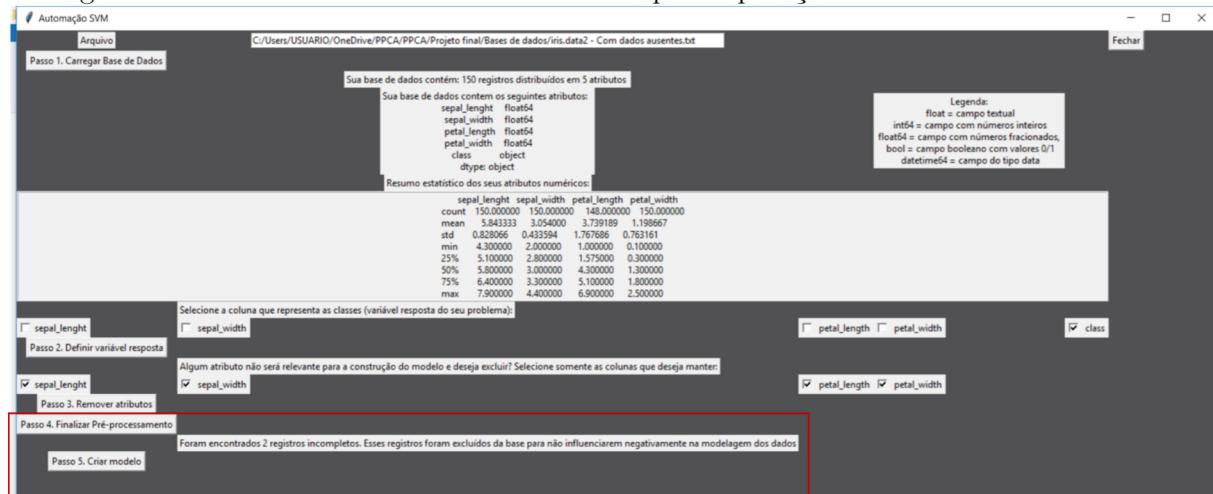
Durante a Etapa de Preparação dos Dados, a primeira informação que o usuário fornece à ferramenta diz respeito à variável que representa a classe do problema. Essa informação será importante para a etapa de modelagem. Na sequência, há outra interação com o usuário para ele selecionar os atributos que deseja manter para a construção modelo. Os demais atributos não selecionados serão retirados da base de dados. A Figura 3.5 apresenta as interações com o usuário para coleta das informações.

Figura 3.5: Interface Gráfica Ferramenta - Etapa Preparação dos Dados - Parte 1



Buscou-se adequar a linguagem da ferramenta aos conhecimentos do utilizador, de maneira que seja evitado a sua utilização incorreta. Em seguida, é finalizada a etapa de Preparação dos dados sem a interação com o usuário. Nesse momento serão excluídos os registros com dados ausentes, transformadas as variáveis categóricas. A ferramenta retorna ao consultor a informação da quantidade de registros excluídos por conta de dados ausentes bem como a explicação dos motivadores para essa exclusão, conforme Figura 3.6. Apesar da base de dados original não conter dados ausentes, foram excluídas informações da base de forma artificial para que fosse possível mostrar a funcionalidade.

Figura 3.6: Interface Gráfica Ferramenta - Etapa Preparação dos Dados - Parte 2



Apesar da base de dados original estar completamente preenchida, foram excluídas informações da base de forma artificial para que tivesse dados ausentes e fosse possível demonstrar a funcionalidade.

3.4.3 Modelagem

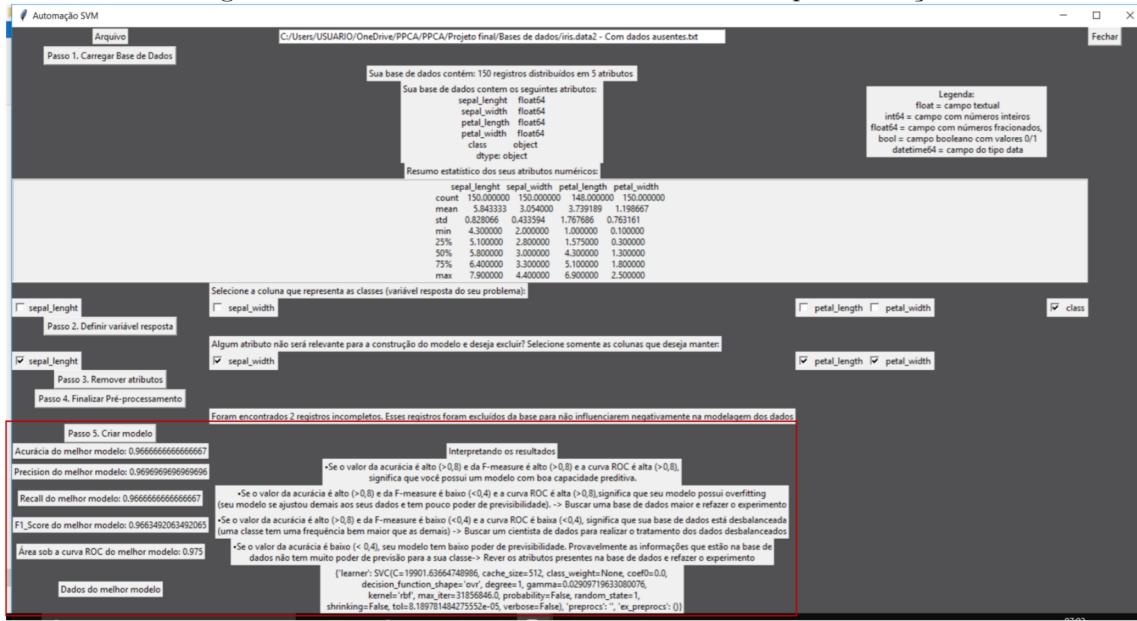
Para a etapa de modelagem, o consultor não precisa ter interação com a ferramenta. Basta clicar para iniciar o processamento e a ferramenta realiza o treino e teste dos modelos desenvolvidos.

3.4.4 Avaliação

A última etapa dentro da ferramenta diz respeito a Avaliação dos resultados. Essa etapa é sensível, pois uma interpretação equivocada pode gerar conclusões incorretas. Para subsidiar a análise, são disponibilizados cinco indicadores de performance que combinados,

podem gerar conclusões mais assertivas. Para facilitar a interpretação dos resultados, é apresentado brevemente ao consultor o conceito de cada indicador e a melhor forma de interpretação e ação a ser tomada para cada situação conforme apresentado na Figura 3.7.

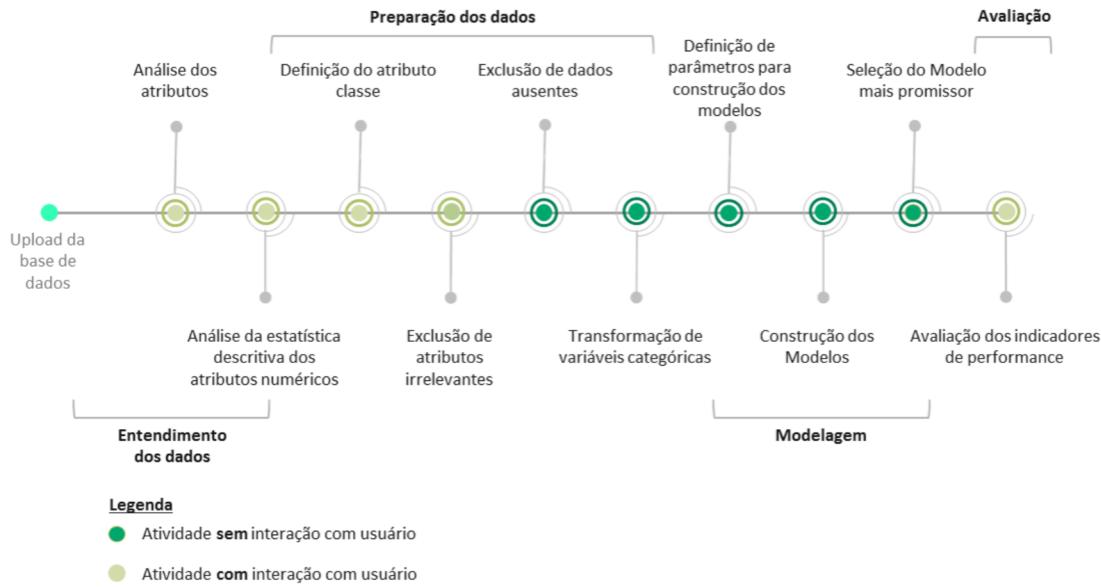
Figura 3.7: Interface Gráfica Ferramenta - Etapa Avaliação



Conforme destacado dentro do quadro vermelho, são apresentados os resultados do melhor modelo sob o ponto de vista de cada indicador de desempenho, junto com informações para auxiliar na análise. Após a análise, também são apresentados os hiperparâmetros do melhor modelo, para que caso seja necessário, o modelo possa ser colocado em produção.

A Figura 3.8 apresenta um fluxo resumido do passo a passo da ferramenta, desde o momento onde o usuário faz o *upload* da base de dados até a avaliação dos indicadores de desempenho.

Figura 3.8: Fluxo de Atividades da Ferramenta



A ferramenta possui então, cinco etapas onde não há necessidade de interação ou análise pelo usuário e cinco etapas onde o usuário precisa ter alguma interação ou análise.

3.5 Testes e Resultados Preliminares - Parte 1

Esse seção apresenta alguns resultados preliminares da ferramenta obtidos a partir do teste com bases de dados que simulam a sua utilização por um usuário sem conhecimentos de Mineração de Dados. Os testes buscaram simular a operação da ferramenta, identificar limitações e validar os resultados. Após a primeira rodada de testes, voltou-se para a etapa de desenvolvimento, onde foram desenvolvidas novas funcionalidades e ajustados alguns erros encontrados. Na sequência, os testes foram refeitos e apresentados abaixo.

A fim de validar a usabilidade e aplicabilidade da ferramenta, foram utilizadas cinco bases de dados públicas disponíveis no UCI Machine Learning Repository (<https://archive.ics.uci.edu/ml/datasets.php>) acessadas em 09 de abril de 2019. Foram utilizadas as bases Iris, Abalone, Wine, Adult e Magic Gama Telescopa.

IRIS

Uma das bases de dados mais utilizadas para reconhecimento de padrões. A partir das dimensões de comprimento e largura de pétalas e sépalas de flores, busca-se a classificação entre três tipos de flor Iris: versicolor, setosa e virginica. As três classes possuem a mesma proporção de registros e não foram encontrados dados ausentes

ADULT

A partir de informações numéricas e categóricas extraídas do censo norte americano como

idade, grau de instrução, sexo, status de relacionamento, cor de pele, horas de trabalho semanal, país nativo, etc. Busca-se prever se a pessoa irá receber mais de 50 mil por ano ou não. Existem 3620 registros incompletos na base, porém a ferramenta não capturou, pois estão preenchidos com "?" e a base está desbalanceada, pois aproximadamente 75% dos registros possuem salário anual inferior a 50 mil por ano.

WINE

Base de dados extraída a partir da análise química de três diferentes vinícolas de uma mesma região da Itália. A partir de treze atributos numéricos extraídos da análise química, como teor acólico, alcalinidade, total de fenóis, etc, busca-se prever de onde o vinho é originado. A base está relativamente balanceada e não foram encontrados dados ausentes.

ABALONE

Abalone é um tipo de molusco marinho e para o cálculo da sua idade, é realizado através da realização de um corte em sua casca e contagem do número de anéis utilizando um microscópio. A idade é definida a partir do número de anéis mais 1,5 anos. Acredita-se que a idade também pode ser calculada a partir de informações do abalone, como sexo, comprimento, diâmetro, peso total do abalone, peso da sua carne, peso das vísceras e peso após secagem. A amostra possui quantidade de anéis variando entre um e vinte e nove e sua distribuição se aproxima de uma distribuição normal com média igual a 9. Não foram encontrados dados ausentes.

MAGIC GAMA TELESCOPE

Baseado em informações de partículas em um telescópio, busca-se classificar os raios em gamma ou hadron utilizando as dimensões dos eixos menores e maiores da elipse, distância do centro à origem da elipse, ângulo entre o maior eixo e o vetor de origem e outras informações técnicas das partículas. Aproximadamente 65% da amostra corresponde a radiação gamma, enquanto o restante radiação hadron.

A escolha das bases de dados levou em consideração a característica de cada uma. Foram escolhidas bases que tivessem características distintas, como quantidade de atributos, registros, características dos dados e número de classes distintas. Com a heterogeneidade das bases, foi possível testar e avaliar todas as funcionalidades implementadas na ferramenta, bem como a sua performance em diferentes cenários. A Tabela 3.3 resume as características gerais de cada base de dados.

Inicialmente foi realizado o estudo de cada base de dados coletada para entender como o problema de Mineração de Dados seria estruturado para cada caso e em seguida, foram percorridas todas as etapas da ferramenta. Os testes foram realizados utilizando uma máquina com as seguintes configurações:

- Windows 10 Pro 64 bits
- Processador Intel Core i7-7700HQ 2.8GHz

Tabela 3.3: Características das bases de dados testadas

Base de dados	Qtd Atributos	Registros	Característica Dados	N classes
Iris	4	150	Numéricos	3
Adult	14	48842	Numéricos Categóricos	2
Wine	13	178	Numéricos	3
Abalone	8	4177	Numéricos Categóricos	28
MAGIC Gamma	11	19020	Numéricos	2

- 32gb de memória ram
- Placa de Vídeo Nvídia GeForce GTX1050

Os resultados dos indicadores de performance de cada modelo gerado com a utilização da ferramenta para as cinco bases testadas estão apresentados na Tabela 3.4.

Tabela 3.4: Indicadores de Performance das Bases de Dados

Base de dados	Acurácia	Precision	Recall	F1-Score	Curva ROC
Iris	0.933	0.933	0.933	0.933	0.949
Adult	0.764	0.585	0.764	0.663	0.5
Wine	1.0	1.0	1.0	1.0	1.0
Abalone	0.228	0.118	0.228	0.190	0.554
MAGIC Gamma	0.834	0.836	0.834	0.829	0.793

A análise dos indicadores de performance mostram que a ferramenta obteve resultados satisfatórios para as bases Iris, Wine e Magic Gama, o que leva a gerar uma conclusão preliminar que para bases com atributos numéricos, tende a performar de forma satisfatória, enquanto para bases com atributos categóricos, o resultado tende a ser inferior. Uma potencial justificativa é que pelo fato de durante o tratamento dos dados a ferramenta realizar a transformação dos dados categóricos utilizando o one hot encoding pode aumentar a quantidade de atributos e consequentemente a complexidade do modelo. Uma possível solução a ser testada é a inclusão de técnicas de redução de dimensionalidade na ferramenta.

A comparação dos resultados obtidos com a ferramenta e as demais soluções já existentes, apresentado na Tabela 3.5, apresentou resultados muito parecidos para as bases Iris e Wine, o que mostra que mesmo com somente um classificador implementado, possui potencial de superar as demais.

Tabela 3.5: Acurácia ferramenta desenvolvida e ferramentas de AutoML

	Hyperopt	TPOT	AutoWeka	Solução Proposta
Iris	0,90	0,96	0,98	0,93
Wine	1,00	0,97	0,99	1,00
Abalone	0,27	0,28	0,33	0,23
Magic Gama	0,83	0,86	0,98	0,83
Adult	0,76	0,86	0,85	0,76

Para a base Iris, obteve acurácia somente de 0,047 inferior à melhor ferramenta, o AutoWeka. Para a base *Wine*, a solução proposta obteve performance superior ao AutoWeka e TPOT.

Análises adicionais mostram que tanto as bases Abalone quanto Adult, que tiveram resultados inferiores, possuem classes desbalanceadas. Para o caso da Abalone, como existe um alto número de classes, há situações de classes com apenas um registro. Essa situação gera uma dificuldade de treino do modelo, pois há a possibilidade de na base de treino não haver nenhum registro pertencente à essa classe. Buscando resolver essa limitação, podem ser realizados testes futuros com a inserção de técnicas de balanceamento dos dados. Japkowicz [49] mostra que em geral, bases com classes desbalanceadas atrapalham o desempenho dos classificadores. Portanto, com utilização de técnica de sobreamostragem da classe minoritária, por exemplo, a criação de dados artificiais para as classes minoritária poderia atenuar essa limitação e melhorar o resultado do modelo.

Com relação ao tempo para processamento da modelagem, para as bases menores (Iris e Wine), o tempo foi inferior a dois e três minutos, respectivamente. Porém, a medida que o tamanho da base aumentou, o tempo também cresceu consideravelmente. Para Abalone, houve uma duração de quinze minutos de processamento, enquanto para Magic Gama quarenta e dois minutos e para Adult, uma hora e doze minutos.

A comparação com as ferramentas TPOT, Hyperopt-Sklearn e AutoWeka mostrou que para as bases Iris, Wine, Abalone e Magic Gama, a solução proposta teve tempo de processamento consideravelmente inferior conforme apresentado na Tabela 3.6. Por sua vez, para a base Adult, teve tempo de processamento próximo da melhor ferramenta, o Hyperopt.

Porém, os comparativos de tempo não podem ser analisados sozinhos, pois a ferramenta desenvolvida possui somente um classificador implementado, enquanto as outras possuem volume de classificadores implementados consideravelmente superior. A medida que aumenta-se a quantidade de classificadores, o conjunto de configurações também aumenta e portanto, o tempo de processamento tende a aumentar. De qualquer forma, o tempo de processamento se mostra como um ponto de alerta para bases maiores. Esse

Tabela 3.6: Tempo processamento (em minutos) ferramenta desenvolvida e ferramentas de AutoML

	Hyperopt	TPOT	AutoWeka	Solução Proposta
Iris	2,9	8,4	10,0	2,0
Wine	3,4	9,7	10,0	2,7
Abalone	15,5	165,2	90,0	15,2
Magic Gama	43,2	400,4	90,0	42,5
Adult	75,2	1087,9	90,0	77,2

fato pode limitar a utilização da ferramenta e uma possível melhoria seria a implantação do processamento paralelo durante a etapa de modelagem, a qual demanda maior esforço computacional.

3.6 Análise da Ferramenta

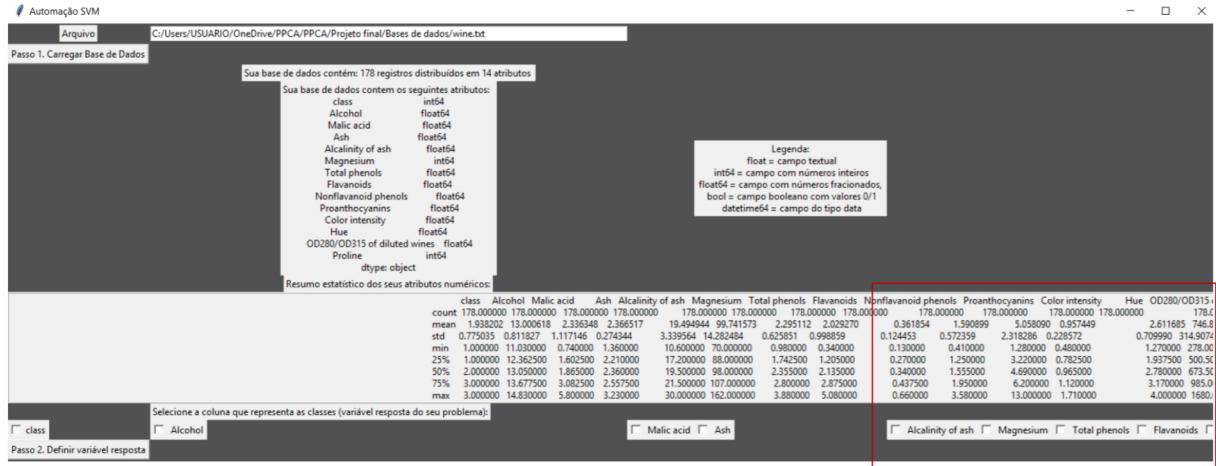
A ferramenta foi analisada levando em consideração os seguintes fatores: performance dos modelos construídos, tempo de processamento, funcionalidades desenvolvidas e facilidade de manipulação e utilização.

Conforme visto na seção anterior, em geral a ferramenta performou bem, gerando os modelos consistentes mesmo para bases mais complexas (maior quantidade de registros, atributos e classes). Os resultados foram similares a outras ferramentas de AutoML e para bases menores, a modelagem foi concluída de forma muito rápida, com tempos inferiores a três minutos, para alguns casos. Porém, a medida que a base aumenta de tamanho, o tempo de processamento é elevado consideravelmente.

Durante os testes, foram identificados *gaps* nas etapas de entendimento dos dados e avaliação, pois forneciam poucos insumos para que o usuário pudesse analisar e tomar decisões corretas. O desenvolvimento foi complementado considerando novas informações a serem disponibilizadas em tela e os testes foram refeitos.

No quesito utilização, houve uma dificuldade por conta da limitação de tamanho de tela imposta pelo pacote utilizado para desenvolvimento da interface gráfica. Considerando que o Tkinter não habilita a inclusão de barra de rolagem tanto para o eixo vertical quanto para o eixo horizontal, em algumas situações houve extração da quantidade de informações a ser exibida em tela, conforme pode ser visualizado na Figura 3.9.

Figura 3.9: Limitação Interface Gráfica Ferramenta



Por conta da lógica de organização do Tkinter disponibilizado em forma de *grid*, em situações onde as bases de dados possuem um volume maior de atributos, como a base Wine representada na Figura 3.9, as informações extrapolam o limite da tela, dificultando a utilização e entendimento dos dados.

Adicionalmente, durante a etapa de Entendimento dos Dados, foi percebido para algumas bases, a existência de outliers, porém na versão disponibilizada como protótipo da ferramenta ainda não foi implementada uma funcionalidade para tratamento desses *outliers*. Para a próxima versão espera-se que essa funcionalidade esteja presente, bem como a migração da ferramenta para Django, buscando resolver a limitação da quantidade de informações exibida em tela.

3.7 Testes e Resultados Preliminares - Parte 2

Caso tivéssemos a quantidade de dados ideal, a forma como foi realizada na etapa anterior, separando a base em treino e teste seria a mais recomendada para realização da modelagem. Porém, como em geral os dados são escassos, Campbell e Ying [1] sugerem que para dados limitados, deve-se utilizar a validação cruzada, pois ela tende a reduzir o viés de validação. A validação cruzada utilizando o método *k-fold cross-validation* consiste em separar a base de dados em *k* grupos, onde o modelo é treinado com *k-1* grupos e testado no remanescente e assim calculado o erro de previsão. Isso é feito para todas as combinações e combina-se as *k* estimativas para se obter o erro médio de previsão. Friedman et. al [38] recomendam que as escolhas mais comuns de *k* são 5 ou 10.

Diante do exposto, buscando reduzir o viés de validação e potencial overfitting do modelo, foi implementada a validação cruzada na ferramenta utilizando *k* igual a 5. Em

seguida, foi realizado a teste utilizando as mesas cinco bases anteriores. Os resultados obtidos com a validação cruzada estão apresentados na Tabela 3.7.

Tabela 3.7: Indicadores de performance utilizando validação cruzada

Base de dados	Acurácia	Precision	Recall	F1 score	Curva ROC
Iris	0,97	0,97	0,97	0,97	0,98
Wine	0,94	0,94	0,94	0,94	0,96
Abalone	0,27	0,28	0,27	0,27	0,55
Magic Gama	0,57	0,57	0,57	0,64	0,62
Adult	0,76	0,76	0,76	0,86	0,50

Para fins de comparação, a Tabela 3.8 mostra a acurácia para as bases com e sem validação cruzada.

Tabela 3.8: Comparação acurácia com e sem validação cruzada

Base de dados	Sem cross-validation	Com cross-validation
Iris	0,93	0,97
Wine	1	0,94
Abalone	0,23	0,27
Magic Gama	0,83	0,57
Adult	0,76	0,76

Em geral, não houve grandes modificações de performance quando comparadas as propostas com e sem validação cruzada, exceto para a base Magic Gama, a qual reduziu em aproximadamente 25p.p. Por fim, foi realizada a comparação do tempo de processamento utilizando a solução com validação cruzada e sem validação cruzada. Os resultados estão apresentados na Tabela 3.9

Tabela 3.9: Comparação tempo de processamento (em minutos) com e sem validação cruzada

Base de dados	Sem cross-validation	Com cross-validation
Iris	2,0	13,6
Wine	2,7	15,4
Abalone	15,2	93,8
Magic Gama	42,5	277,4
Adult	77,2	437,1

Um fato importante a se mencionar é que quando da implantação da validação cruzada, gerou um aumento considerável do tempo de processamento (aproximadamente 6 a 7 vezes o tempo de processamento sem a validação cruzada). Porém, apesar de não haver grandes variações de desempenho, os resultados obtidos transmitem maior segurança para o consultor, pois tende a ter um viés menor e portanto, insumos mais para tomada de decisão. Outro ponto a se observar é que mesmo com o aumento do tempo, o processamento da solução proposta com validação cruzada ainda é inferior ao do TPOT para as bases maiores. Chega-se portanto, em um *trade-off*, onde de uma lado está a velocidade de processamento, de outro, a confiabilidade do modelo gerado. Acredita-se que dada a complexidade dos projetos de consultoria e a criticidade e potencial impacto das soluções propostas, é mais recomendado que seja implementada a solução com validação cruzada.

Capítulo 4

Conclusões

Apesar dos grandes benefícios que a Mineração de Dados e o Aprendizado de Máquinas podem trazer, esse conhecimento ainda está restrito a uma quantidade limitada de profissionais. Somado a isso, a demanda por análise em larga escala de informações vem crescendo rapidamente. Bases de dados são realidade em projetos de consultoria e a utilização da mineração de dados pode otimizar os resultados gerados por esses projetos e apesar do interesse desse tema em diversos colaboradores da empresa, boa parte dos consultores ainda não domina a temática e encontra dificuldades para manipular bases de dados e extrair informações. Diante desse cenário, trabalhar com grandes bases de dados vem se tornando um gargalo dentro da organização.

Neste Capítulo são apresentadas as conclusões gerais e propostas de trabalhos futuros. O objetivo principal do trabalho definido no Capítulo 1 era a construção de um protótipo de ferramenta que automatizasse um classificador SVM para utilização em projetos de consultoria em gestão. Para tal, foi necessário programar o algoritmo que automatiza as etapas do processo de mineração de dados, desde o tratamento dos dados, até a definição dos hiperparâmetros do SVM; o desenvolvimento da interface que viabiliza a utilização pelos consultores e o teste da ferramenta desenvolvida.

Portanto, o presente trabalho propôs a criação de um protótipo de ferramenta que automatizasse um classificador SVM permitindo sua utilização mesmo por pessoas que desconhecem do assunto. Buscou-se atender uma carência da consultoria e do mercado em geral, que é a falta de disponibilidade de cientistas de dados.

Por meio da revisão de literatura de Mineração de Dados, *Machine Learning*, algoritmos de classificação e *Automated Machine Learning*, foi possível identificar potenciais funcionalidades a serem implementadas na ferramenta. Complementar à revisão de literatura, foram realizados testes práticos com as ferramentas de AutoML encontradas. Com a análise comparativa e a revisão de literatura, foi possível construir a ferramenta e com isso, permitiu sua evolução a partir ideias que melhoraram a performance e os resultados.

Diferentemente de outras ferramentas *open source* de AutoML, como por exemplo, Hyperopt-Sklearn e TPOT, testadas no Capítulo 3, a solução desenvolvida permite a construção de um modelo de mineração de dados sem a necessidade de uma linha de código sequer e sem conhecimentos aprofundados de *Machine Learning* e Mineração de Dados por parte do consultor. Foram implementadas funcionalidades para auxiliar o usuário no entendimento dos dados, tratamento dos dados, modelagem e avaliação.

A validação e análise da viabilidade da ferramenta foi testada utilizando bases de dados públicas com características distintas e obtendo boa performance. Foram testados também a ferramenta utilizando validação cruzada e sem validação cruzada. Acredita-se que a solução proposta com *cross-validation* apesar do maior tempo de processamento, se mostra mais recomendada para a situação em estudo, pois minimiza o risco de um modelo com *overfitting* e consequentemente, tomada de decisões equivocadas. Quando comparada com outras ferramentas disponíveis no mercado, possui como grande diferencial, o fato de não demandar necessidade de programação nem conhecimento prévio de Mineração de Dados e *Machine Learning*. Pelo fato do protótipo possuir somente um algoritmo de Mineração de Dados implementado, também tem a vantagem de ter menor tempo de processamento quando comparado às demais ferramentas (para a proposta sem validação cruzada).

Pode-se também pensar nos benefícios da ferramenta do ponto de vista de redução do tempo de execução do projeto. Considerando que um projeto de melhoria tipicamente executado na consultoria tem duração média dez semanas, sendo aproximadamente metade desse tempo destinado para a etapa de diagnóstico do cliente e a outra metade para proposição de melhorias, com a adoção da ferramenta, espera-se que haja uma redução da primeira fase do projeto. Isto posto, acredita-se que a duração do projeto possa ter uma redução de aproximadamente 10 a 20%, pois a modelagem dos dados tende a reduzir o esforço de manipulação de bases e auxiliar na extração de informações do cliente de forma mais rápida.

Diante do exposto, pode-se concluir que a ferramenta apresenta contribuição para o contexto de consultoria, auxiliando no processo de entendimento da situação do cliente, tomada de decisão e subsídio a geração de soluções para projetos de consultoria de gestão. A ferramenta trás uma proposta de permitir que seja trabalhado com Inteligência Artificial sem a necessidade de contratação extra de pessoal, ou seja, com o mesmo time, permite entregar valor aos seus clientes utilizando soluções de Mineração de Dados. Adicionalmente, democratiza a Mineração de Dados e permite que o consultor foque no entendimento do negócio e dos dados do seu cliente, contribuindo para a otimização dos resultados gerados pelos projetos.

4.1 Trabalhos Futuros

Durante o desenvolvimento do projeto, um dos pontos identificados na fase de testes é que ainda existe potencial de evolução da ferramenta, onde trabalhos futuros poderão aperfeiçoar as etapas de entendimento e tratamento de dados, com a implementação de novas funcionalidades que permitam uma compreensão mais aprofundada da base de dados do consultor e mais alternativas para limpeza e tratamento da base antes da modelagem, em busca de melhor desempenho dos modelos construídos.

Como sugestão, podem ser adicionadas técnicas de remoção de *outliers* ou técnicas de redução de dimensionalidade para tratamento dos dados. Durante a etapa de entendimento dos dados, pode ser implementada a geração de gráficos de dispersão ou boxplot dos dados, de forma a tornar a análise mais simples e visual. Adicionalmente, poderão ser incluídos outros algoritmos de classificação, como Árvores de Decisão, Regressão Logística, Naive Bayes de forma a tornar a ferramenta mais versátil e potencializar seus resultados.

Por fim, a medida que outros algoritmos forem implementados, há uma tendência natural que o tempo para processamento aumente, pois o conjunto de configurações de parâmetros se torna maior. Por isso, buscando tornar o processamento mais rápido, pode-se implantar a funcionalidade de processamento em paralelo durante a etapa de modelagem.

Para aperfeiçoar a usabilidade da ferramenta para o usuário final, sugere-se a migração da interface gráfica para Django. O Django se mostra mais completo, flexível e moderno quando comparado ao Tkinter que foi utilizado e tem a oportunidade de minimizar potenciais impactos de naveabilidade identificados. Dessa forma, com as sugestões de melhorias propostas, a ferramenta se tornará mais robusta e poderá se tornar uma solução aplicável em outros contextos além da consultoria.

Apêndice A

Código Etapa Entendimento dos Dados

```
1 import tkinter as tk
2 from tkinter import *
3 from tkinter.filedialog import askopenfilename
4 import pandas as pd
5 import numpy as np
6
7 #carregar a base de dados para dentro da ferramenta
8     csv_file_path = askopenfilename()
9     print(csv_file_path)
10    v.set(csv_file_path)
11    df = pd.read_csv(csv_file_path)
12
13 #resumo estatistico da base de dados
14
15    summary = df.describe()
16    print(summary)
17
18 #numero de colunas da base de dados
19    n = len(df.columns)
20
21 #numero de linhas da base de dados
22    m = len(df)
23
24 #nome das colunas da base de dados
25    headers = list(df.columns.values)
26
27 #tipo de cada atributo da base de dados
28    print(df.dtypes)
29
```

```
30 #registros iniciais da base de dados  
31     head = df.head()
```

Listing A.1: Código fonte em Python para etapa de Entendimento dos Dados

Apêndice B

Código Etapa Preparação dos Dados

```
1 import tkinter as tk
2 from tkinter import *
3
4 #apresentar todos os atributos para o usuario selecionar a variavel
5 #classe
6 cb_intvar = []
7 for this_row, text in enumerate(headers):
8     cb_intvar.append(tk.IntVar())
9     tk.Checkbutton(root, text=text, variable=cb_intvar[-1])
10        .grid(row=20,
11               column=column, sticky='w')
12     column +=1
13
14 #captar e armazenar a informacao fornecida da variavel classe
15 headers_final = list();
16 for ctr, int_var in enumerate(cb_intvar):
17     if int_var.get():    ## IntVar not zero==checked
18         headers_final.append(headers[ctr])
19
20 #separar o atributo classe dos demais atributos
21 headers_final = list();
22 for ctr, int_var in enumerate(cb_intvar):
23     if int_var.get():    ## IntVar not zero==checked
24         headers_final.append(headers[ctr])
25
26 #apresentar os atributos para o usuario definir aqueles relevantes para
27 #modelagem
28 headers_x = list(x.columns.values)
29
30 row = 2
31 column = 0
```

```

30
31     global cb_intvar2
32     cb_intvar2 = []
33     for this_row, text in enumerate(headers_x):
34         cb_intvar2.append(tk.IntVar())
35         tk.Checkbutton(root, text=text, variable=cb_intvar2[-1]
36                         ).grid(row=23,
37                                 column=column, sticky='w')
38         column +=1
39
40
41 #excluir os atributos que o usuario nao deseja manter
42 headers_final = list();
43 for ctr, int_var in enumerate(cb_intvar):
44     if int_var.get():    ## IntVar not zero==checked
45         headers_final.append(headers[ctr])
46
47 x_final = df[headers_x_final]
48
49 #transformar variaveis categoricas em dummy
50 x_final = pd.get_dummies(x_final)
51
52 #juntar x e y
53 df1 = pd.concat([x_final,y], axis = 1, sort = False)
54
55 #excluir linhas com registros faltantes
56 df1 = df1.dropna()
57
58 #calcular quantidade de registros excluidos
59 linhas_excluidas = m - len(df1)
60
61 #separar base de treino e teste
62 n = len(df1.columns)
63 x = df1.iloc[:, 0:n-1]
64 x = x.values
65
66 y = df1.iloc[:, n-1:n]
67 y = y.values.flatten()
68
69 #x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2)

```

Listing B.1: Código fonte em Python para etapa de Preparação dos Dados

Apêndice C

Código Etapa Modelagem

```
1  from hpsklearn import HyperoptEstimator, svc
2  from hyperopt import tpe, hp, fmin
3  from sklearn.model_selection import StratifiedKFold
4
5  skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=0)
6  folds_score = []
7  folds_recall = []
8  folds_precision = []
9  folds_f1 = []
10  folds_auc = []
11
12  for trn_idx, tst_idx in skf.split(x, y):
13      x_train, y_train, x_test, y_test = x[trn_idx], y[trn_idx], x[tst_idx], y[tst_idx]
14      estim = HyperoptEstimator(classifier=svc('my_svc'),
15                                 algo=tpe.suggest,
16                                 max_evals=100,
17                                 trial_timeout=60,
18                                 preprocessing=''))
19
20      estim.fit(x_train, y_train, valid_size=.2, cv_shuffle=False)
21
22      prediction=estim.predict(x_test)
23
24
25      folds_score.append(accuracy_score(y_test, prediction))
26      folds_recall.append(recall_score(y_test, prediction, average='weighted'))
27      folds_precision.append(precision_score(y_test, prediction,
28                                             average='weighted', labels=np.unique(prediction)))
```

```

28     folds_f1.append (f1_score (y_test , prediction , average = 'weighted'
29                         , labels=np.unique (prediction)))
30 def multiclass_roc_auc_score(y_test , y_pred , average="weighted"):
31     lb = LabelBinarizer()
32     lb . fit (y_test)
33     y_test = lb . transform (y_test)
34     y_pred = lb . transform (y_pred)
35     return roc_auc_score (y_test , y_pred)
36     folds_auc.append (multiclass_roc_auc_score (y_test , prediction))
37
38 accuracy = np.mean (folds_score)
39 recall = np.mean (folds_recall)
40 precision = np.mean (folds_precision)
41 f1= np.mean (folds_f1)
42 roc = np.mean (folds_auc)

```

Listing C.1: Código fonte em Python para etapa de Modelagem

Apêndice D

Código Etapa Avaliação

```
1  folds_score = []
2  folds_recall = []
3  folds_precision = []
4  folds_f1 = []
5  folds_auc = []
6  for trn_idx, tst_idx in skf.split(x, y):
7      x_train, y_train, x_test, y_test = x[trn_idx], y[trn_idx], x[tst_idx], y[tst_idx]
8      estim = HyperoptEstimator(classifier=svc('my_svc'),
9                                  algo=tpe.suggest,
10                                 max_evals=100,
11                                 trial_timeout=60,
12                                 preprocessing=''))
13
14      estim.fit(x_train, y_train, valid_size=.2, cv_shuffle=False)
15
16      prediction= estim.predict(x_test)
17
18
19      folds_score.append(accuracy_score(y_test, prediction))
20      folds_recall.append(recall_score(y_test, prediction, average = 'weighted'))
21      folds_precision.append(precision_score(y_test, prediction,
22                                              average = 'weighted', labels=np.unique(prediction)))
23      folds_f1.append(f1_score(y_test, prediction, average = 'weighted',
24                                labels=np.unique(prediction)))
24  def multiclass_roc_auc_score(y_test, y_pred, average="weighted"):
25      lb = LabelBinarizer()
26      lb.fit(y_test)
27      y_test = lb.transform(y_test)
```

```
28     y_pred = lb.transform(y_pred)
29     return roc_auc_score(y_test, y_pred)
30     folds_auc.append(multiclass_roc_auc_score(y_test, prediction))
31
32 accuracy = np.mean(folds_score)
33 recall = np.mean(folds_recall)
34 precision = np.mean(folds_precision)
35 f1 = np.mean(folds_f1)
36 roc = np.mean(folds_auc)
37
38
39 #informacoes dos hiperparametros do melhor modelo
40 model = estim.best_model()
```

Listing D.1: Código fonte em Python para etapa de Avaliação dos Resultados

Apêndice E

Código Interface Gráfica - Configuração de Tela

```
1 #criacao da tela
2 root = tk.Tk()
3
4 #definicao do titulo da pagina
5 root.title( "Automacao SVM" )
6
7 #definicao do tamanho da pagina
8 root.geometry( "%dx%d+0+0" % (root.winfo_screenwidth() ,root.winfo_
9 screenheight() ) )
10
11 #configuracao da cor de fundo da tela
12 root.configure (background = "#515153")
```

Listing E.1: Código fonte em Python para criação e configuração inicial da tela da interface gráfica

Apêndice F

Código Interface Gráfica - Botões e Funcionalidades

```
1 import tkinter as tk
2 from tkinter import *
3
4     #botao para encerrar a aplicacao
5     tk.Button(root , text='Fechar' ,command=root .destroy) .grid (row=0, column
6         =11)
7
8     #botao para carregar a base de dados
9     tk.Button(root , text='Passo 1. Carregar Base de Dados' ,command=import_
10         csv_data) .grid (row=1, column=0)
11
12     #caixa de texto para indicar onde o usuario fara o upload do arquivo
13     tk.Label(root , text='Arquivo') .grid (row=0, column=0)
14     v = tk.StringVar()
15
16     #entrada de dados com o caminho do arquivo que sera feito upload
17     entry = tk.Entry(root , textvariable=v , width = 100) .grid (row=0, column
18         =1)
19
20     #botao para atribuir variavel resposta do problema
21     tk.Button(root , text='Passo 2. Definir variavel resposta' ,command=
22         selecionar_colunas) .grid (row=21, column=0)
23
24     #botao para remover atributos indesejados da base de dados
25     tk.Button(root , text='Passo 3. Remover atributos' ,command= head) .grid (
26         row=24, column=0)
27
28     #botao para finalizar a preparacao dos dados
```

```
24     tk.Button(root , text='Passo 4. Finalizar Pre-processamento' , command =
25             get_dummies).grid(row=30, column=0)
26
27 #botao para iniciar a modelagem dos dados
28 tk.Button(root , text='Passo 5. Criar modelo' , command = SVM).grid(row
29 =41, column=0)
```

Listing F.1: Código fonte em Python para inserção dos demais botões que habilitam as funcionalidades via interface gráfica

Apêndice G

Código Completo da Ferramenta

```
1 import tkinter as tk
2 from tkinter import *
3 from tkinter.filedialog import askopenfilename
4 import pandas as pd
5 import numpy as np
6 import pickle
7 from sklearn import datasets
8 from sklearn.svm import SVC
9 from sklearn.model_selection import train_test_split
10 from hyperopt import tpe, hp, fmin
11 from sklearn.metrics import accuracy_score, recall_score, precision_score,
12     f1_score, roc_auc_score
12 from hpsklearn import HyperoptEstimator, svc
13 from sklearn.preprocessing import LabelBinarizer
14 from sklearn.model_selection import StratifiedKFold
15
16 pd.set_option('display.max_rows', 500)
17 pd.set_option('display.max_columns', 500)
18 pd.set_option('display.width', 1000)
19 from PIL import ImageTk, Image
20 import os
21
22
23 def import_csv_data():
24     global v
25     global df
26     csv_file_path = askopenfilename()
27     print(csv_file_path)
28     v.set(csv_file_path)
29     df = pd.read_csv(csv_file_path)
30     head = df.head()
```

```

31     summary = df.describe()
32     global n
33     global m
34     n = len(df.columns)
35     m = len(df)
36     print(summary)
37     global headers
38     headers = list(df.columns.values)
39     print(df.dtypes)

40
41     Label(root, text="Resumo estat stico dos seus atributos num ricos:
42         ").grid(row=5, column=1)
43     label = Label(root, text=summary, justify=LEFT, relief=RIDGE)
44     label.grid(row=6, columnspan=n+1, ipadx=10, sticky=W+E+N+S)
45     Label(root, text="Sua base de dados cont m: "+str(m)+" registros
46         distribu dos em "+str(n)+" atributos ").grid(row=3, column=
47         1)
48     Label(root, text="Sua base de dados contem os seguintes atributos: "
49         +'\n'+str(df.dtypes)).grid(row=4, column=1)
50     Label(root, text="Legenda:"+'\n'+" float = campo textual"+'\n'+
51         "int64 = campo com n meros inteiros "+'\n'+" float64 = campo com
52         n meros fracionados, "+'\n'+" bool = campo booleano com valores
53         0/1"+'\n'+" datetime64 = campo do tipo data").grid(row=4,
54         column=3)
55     row = 2
56     column = 0
57     global cb_intvar
58     cb_intvar = []
59     for this_row, text in enumerate(headers):
60         cb_intvar.append(tk.IntVar())
61         tk.Checkbutton(root, text=text, variable=cb_intvar[-1]
62                         ).grid(row=20,
63                         column=column, sticky='w')
64         column +=1

65     Label(root, text="Selecione a coluna que representa as classes (
66         vari vel resposta do seu problema):").grid(row=10, column=1, sticky
67         =W)
68     tk.Button(root, text='Passo 2. Definir vari vel resposta', command=
69         selecionar_colunas).grid(row=21, column=0)

70
71 def selecionar_colunas():
72     headers_final = list();
73     for ctr, int_var in enumerate(cb_intvar):
74         if int_var.get():    ## IntVar not zero==checked

```

```

65     headers_final.append(headers[ctr])
66
67
68 #Separar x e y. Definidas como variáveis globais para poder utilizar
69     nas outras funções
70 global y
71 global x
72 y = df[headers_final]
73 x = df.drop(headers_final, axis = 1)
74
75 #listar atributos em x
76 global headers_x
77 headers_x = list(x.columns.values)
78
79 row = 2
80 column = 0
81
82 global cb_intvar2
83 cb_intvar2 = []
84 for this_row, text in enumerate(headers_x):
85     cb_intvar2.append(tk.IntVar())
86     tk.Checkbutton(root, text=text, variable=cb_intvar2[-1]
87                     ).grid(row=23,
88                             column=column, sticky='w')
89     column +=1
90
91 Label(root, text="Algum atributo não ser relevante para a
92       construção do modelo e deseja excluir? Selecione somente as
93       colunas que deseja manter:").grid(row=22, column=1, sticky=W)
94 #botão para remover atributos indesejados da base de dados
95 tk.Button(root, text='Passo 3. Remover atributos', command= head).grid(
96     row=24, column=0)
97
98 def head():
99
100    headers_x_final = list();
101    for ctr, int_var in enumerate(cb_intvar2):
102        if int_var.get():    ## IntVar not zero==checked
103            headers_x_final.append(headers_x[ctr])
104
105    print(headers_x_final)
106    global x_final
107    x_final = df[headers_x_final]

```

```

106     print (x_final)
107
108     #botao para finalizar a preparacao dos dados
109     tk.Button(root , text='Passo 4. Finalizar Pr -processamento' , command =
110             get_dummies).grid(row=30, column=0)
111
112
113
114 def get_dummies():
115     global df1
116     #transformar vari veis categ ricas em dummy
117     global x_final
118     x_final = pd.get_dummies(x_final)
119
120     #juntar x e y
121     df1 = pd.concat([x_final,y] , axis = 1, sort = False)
122     print (df1)
123
124     #excluir linhas com registros faltantes
125     df1 = df1.dropna()
126
127     linhas_excluidas = m - len(df1)
128
129     Label (root , text = "Foram encontrados " + str(linhas_excluidas) + "
130             registros incompletos. Esses registros foram excluidos da base para
131             n o influenciarem negativamente na modelagem dos dados").grid(row
132             =31, column = 1)
133
134     print(df1)
135     #botao para iniciar a modelagem dos dados
136     tk.Button(root , text='Passo 5. Criar modelo' , command = SVM).grid(row
137             =41, column=0)
138
139
140 def SVM():
141     global df1
142     global n
143     n = len(df1.columns)
144     x = df1.iloc[:, 0:n-1]
145     x = x.values
146
147     y = df1.iloc[:, n-1:n]
148     y = y.values.flatten()
149     global y_test

```

```

146 x_train ,x_test ,y_train ,y_test = train_test_split (x,y,test_size=0.2)
147
148
149 skf = StratifiedKFold (n_splits=5, shuffle=True, random_state=0)
150 folds_score = []
151 folds_recall = []
152 folds_precision = []
153 folds_f1 = []
154 folds_auc = []
155 for trn_idx, tst_idx in skf.split (x, y):
156     x_train, y_train, x_test, y_test = x[trn_idx], y[trn_idx], x[tst_
157         idx], y[tst_idx]
158     estim = HyperoptEstimator(classifier=svc('my_svc'),
159                               algo=tpe.suggest,
160                               max_evals=100,
161                               trial_timeout=60,
162                               preprocessing=''))
163
164     estim.fit(x_train, y_train, valid_size=.2, cv_shuffle=False)
165
166
167
168     folds_score.append(accuracy_score(y_test, prediction))
169     folds_recall.append(recall_score(y_test, prediction, average =
170         'weighted'))
170     folds_precision.append(precision_score(y_test, prediction,
171         average = 'weighted', labels=np.unique(prediction)))
171     folds_f1.append(f1_score(y_test, prediction, average = 'weighted',
172         labels=np.unique(prediction)))
173 def multiclass_roc_auc_score(y_test, y_pred, average="weighted"):
174     lb = LabelBinarizer()
175     lb.fit(y_test)
176     y_test = lb.transform(y_test)
177     y_pred = lb.transform(y_pred)
178     return roc_auc_score(y_test, y_pred)
179
180     folds_auc.append(multiclass_roc_auc_score(y_test, prediction))
181
182 accuracy = np.mean(folds_score)
183 recall = np.mean(folds_recall)
184 precision = np.mean(folds_precision)
185 f1= np.mean(folds_f1)
186 roc = np.mean(folds_auc)
187
188 #informacoes dos hiperparametros do melhor modelo

```

```

187 model = estim.best_model()
188 print (model)
189 Label (root , text="Acuracia do melhor modelo: " + str (score)).grid(
190     row = 500, column = 0)
191 Label (root , text="Precision do melhor modelo: "+ str(precision)).grid(
192     row = 501, column = 0)
193 Label (root , text="Recall do melhor modelo: "+ str(recall)).grid(row =
194     502, column = 0)
195 Label (root , text="F1_Score do melhor modelo: "+ str(f1)).grid(row =
196     503, column = 0)
197 Label (root , text=" rea sob a curva ROC do melhor modelo: "+ str(roc))
198     .grid(row = 504, column = 0)
199 Label (root , text = "Interpretando os resultados").grid(row = 500,
200     column = 1)
201 Label (root , text = " Se o valor da acuracia alto (>0,8) e da F-
202     measure alto (>0,8) e a curva ROC alta (>0,8),"+ '\n' +
203     "significa que voc possui um modelo com boa capacidade preditiva.")
204     .grid(row = 501, column = 1)
205 Label (root , text = " Se o valor da acuracia alto (>0,8) e da F-
206     measure baixo (<0,4) e a curva ROC alta (>0,8), significa que
207     seu modelo possui overfitting"+ '\n' +
208     "(seu modelo se ajustou demais aos seus dados e tem pouco poder de previsibilidade). ->
209     Buscar uma base de dados maior e refazer o experimento").grid(row =
210     502, column = 1)
211 Label (root , text = " Se o valor da acuracia alto (>0,8) e da F-
212     measure baixo (<0,4) e a curva ROC baixa (<0,4), significa que
213     sua base de dados est desbalanceada"+ '\n' +
214     "(uma classe tem uma frequencia bem maior que as demais) -> Buscar um cientista de
215     dados para realizar o tratamento dos dados desbalanceados").grid(row =
216     503, column = 1)
217 Label (root , text = " Se o valor da acuracia baixo (< 0,4), seu
218     modelo tem baixo poder de previsibilidade. Provavelmente as
219     informa es que est o na base de"+ '\n' +
220     "dados n o tem muito
221     poder de previs o para a sua classe-> Rever os atributos presentes
222     na base de dados e refazer o experimento").grid(row = 504, column =
223     1)
224 Label (root , text = "Dados do melhor modelo").grid(row = 505, column =
225     0)
226 Label (root , text = model).grid(row = 505, column = 1)
227
228
229
230
231
232
233
234
235 #criacao da tela
236 root = tk.Tk()

```

```

207 #definicao do titulo da pagina
208 root.title("Automacao SVM")
209 #definicao do tamanho da pagina
210 root.geometry("%dx%d+0+0" % (root.winfo_screenwidth(),root.winfo_
211 screenheight())))
211 #configuracao da cor de fundo da tela
212 root.configure(background = "#515153")
213
214 #caixa de texto para indicar onde o usuario fara o upload do arquivo
215 tk.Label(root, text='Arquivo').grid(row=0, column=0)
216 v = tk.StringVar()
217 #entrada de dados com o caminho do arquivo que sera feito upload
218 entry = tk.Entry(root, textvariable=v, width = 100).grid(row=0, column=1)
219 #botao para carregar a base de dados
220 tk.Button(root, text='Passo 1. Carregar Base de Dados',command=import_csv_
221 data).grid(row=1, column=0)
221 #botao para encerrar a aplicacao
222 tk.Button(root, text='Fechar',command=root.destroy).grid(row=0, column=11)

```

Listing G.1: Código completo da Ferramenta em Python

Referências

- [1] Campbell, Colin e Yiming Ying: *Learning with support vector machines*. Synthesis lectures on artificial intelligence and machine learning, 5(1):1–95, 2011. x, 13, 14, 16, 40
- [2] Witten, Ian H, Eibe Frank, Mark A Hall e Christopher J Pal: *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016. 1, 7, 20
- [3] Provost, Foster e Tom Fawcett: *Data Science for Business: What you need to know about data mining and data-analytic thinking*. " O'Reilly Media, Inc.", 2013. 1
- [4] Campos, Vicente Falconi: *O verdadeiro poder*. INDG-Instituto de Desenvolvimento Gerencia, 2009. 1
- [5] Gil, Antonio Carlos: *Como elaborar projetos de pesquisa*. São Paulo, 5(61):16–17, 2002. 4
- [6] Silveira, Denise Tolfo, Fernanda Peixoto Córdova *et al.*: *Métodos de pesquisa*. Porto Alegre: Editora da UFRGS, 1, 2009. 4
- [7] Mitchell, T.M.: *Machine Learning*. McGraw-Hill international editions - computer science series. McGraw-Hill Education, 1997, ISBN 9780070428072. <https://books.google.com.br/books?id=x0GAngEACAAJ>. 7
- [8] Marsland, S.: *Machine Learning: An Algorithmic Perspective*. CRC Press, 2011, ISBN 9781420067194. 7
- [9] Hong, Tzung Pei, Chan Sheng Kuo e Sheng Chai Chi: *Trade-off between computation time and number of rules for fuzzy mining from quantitative data*. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 09(05):587–604, 2001. 7
- [10] Wu, Xindong, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, S Yu Philip *et al.*: *Top 10 algorithms in data mining*. Knowledge and information systems, 14(1):1–37, 2008. 7
- [11] Feurer, Matthias, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum e Frank Hutter: *Efficient and robust automated machine learning*. Em *Advances in Neural Information Processing Systems*, páginas 2962–2970, 2015. 8, 9, 11

- [12] Olson, Randal S e Jason H Moore: *Tpot: A tree-based pipeline optimization tool for automating machine learning*. Em *Workshop on Automatic Machine Learning*, páginas 66–74, 2016. 8, 11, 12
- [13] Chung, Che Min, Cai Cing Chen, Wei Ping Shih, Ting En Lin, Rui Jun Yeh e Iru Wang: *Automated machine learning for internet of things*. Em *Consumer Electronics-Taiwan (ICCE-TW), 2017 IEEE International Conference on*, páginas 295–296. IEEE, 2017. 8
- [14] Bergstra, James, Brent Komer, Chris Eliasmith, Dan Yamins e David D Cox: *Hyperopt: a python library for model selection and hyperparameter optimization*. Computational Science & Discovery, 8(1):014008, 2015. 8, 11
- [15] Kotthoff, Lars, Chris Thornton, Holger H Hoos, Frank Hutter e Kevin Leyton-Brown: *Auto-weka 2.0: Automatic model selection and hyperparameter optimization in weka*. The Journal of Machine Learning Research, 18(1):826–830, 2017. 8, 10
- [16] Brochu, Eric, Vlad M Cora e Nando De Freitas: *A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning*. arXiv preprint arXiv:1012.2599, 2010. 9
- [17] Xuesi, Li, Yang Hongqiao, Sun Jing, Bi Yangang e Wu Yuanli: *An algorithm of model selection for support vector regression*. Em *2012 Proceedings of the International Conference on Computer Applications and System Modeling*, 2012. 9
- [18] Ayat, Nedjem Eddine, Mohamed Cheriet e Ching Y Suen: *Automatic model selection for the optimization of svm kernels*. Pattern Recognition, 38(10):1733–1745, 2005. 9, 13
- [19] Wahba, Grace, Xiwu Lin, Fangyu Gao, Dong Xiang, Ronald Klein e Barbara Klein: *The bias-variance tradeoff and the randomized gacv*. Em *Advances in Neural Information Processing Systems*, páginas 620–626, 1999. 9
- [20] Vapnik, Vladimir N e A Ya Chervonenkis: *On the uniform convergence of relative frequencies of events to their probabilities*. Em *Measures of complexity*, páginas 11–30. Springer, 2015. 9
- [21] Hsu, Chih Wei, Chih Chung Chang, Chih Jen Lin *et al.*: *A practical guide to support vector classification*. 2003. 9, 14, 15, 16
- [22] Brazdil, Pavel, Christophe Giraud-Carrier, Carlos Soares e Ricardo Vilalta: *Metalearning: Applications to Data Mining*. Springer Publishing Company, Incorporated, 1^a edição, 2008, ISBN 3540732624, 9783540732624. 10
- [23] Kuba, Petr, Pavel Brazdil, Carlos Soares, Adam Woznica *et al.*: *Exploiting sampling and meta-learning for parameter setting support vector machines*. Em *Proceedings of the IBERAMIA*, volume 2002, páginas 217–225, 2002. 10, 14

- [24] Gomes, Taciana AF, Ricardo BC Prudêncio, Carlos Soares, André LD Rossi e André Carvalho: *Combining meta-learning and search techniques to svm parameter selection*. Em *Neural Networks (SBRN), 2010 Eleventh Brazilian Symposium on*, páginas 79–84. IEEE, 2010. 10, 14
- [25] Thornton, Chris, Frank Hutter, Holger H Hoos e Kevin Leyton-Brown: *Auto-weka: Combined selection and hyperparameter optimization of classification algorithms*. Em *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, páginas 847–855. ACM, 2013. 10
- [26] Hall, Mark, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann e Ian H Witten: *The weka data mining software: an update*. ACM SIGKDD explorations newsletter, 11(1):10–18, 2009. 10
- [27] Komer, Brent, James Bergstra e Chris Eliasmith: *Hyperopt-sklearn: automatic hyperparameter configuration for scikit-learn*. Em *ICML workshop on AutoML*, 2014. 11, 29
- [28] Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg *et al.*: *Scikit-learn: Machine learning in python*. Journal of machine learning research, 12(Oct):2825–2830, 2011. 11
- [29] Olson, Randal S, Nathan Bartley, Ryan J Urbanowicz e Jason H Moore: *Evaluation of a tree-based pipeline optimization tool for automating data science*. Em *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, páginas 485–492. ACM, 2016. 11
- [30] Swearingen, Thomas, Will Drevo, Bennett Cyphers, Alfredo Cuesta-Infante, Arun Ross e Kalyan Veeramachaneni: *Atm: A distributed, collaborative, scalable system for automated machine learning*. Em *IEEE International Conference on Big Data*, 2017. 12, 15
- [31] Jin, Haifeng, Qingquan Song e Xia Hu: *Efficient neural architecture search with network morphism*. arXiv preprint arXiv:1806.10282, 2018. 12
- [32] Achin, Jeremy, Thomas DeGodoy, Timothy Owen e Xavier Conort: *Systems and techniques for predictive data analytics*, 2017. US Patent 9,652,714. 13
- [33] Vapnik, Vladimir: *The nature of statistical learning theory*. Springer science & business media, 2013. 13
- [34] Ali, Shawkat e Kate A Smith-Miles: *A meta-learning approach to automatic kernel selection for support vector machines*. Neurocomputing, 70(1-3):173–186, 2006. 13, 14, 16
- [35] Cristianini, Nello e John Shawe-Taylor: *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000. 13, 14
- [36] Bennett, Kristin P e Colin Campbell: *Support vector machines: hype or hallelujah?* Acm Sigkdd Explorations Newsletter, 2(2):1–13, 2000. 14

- [37] Moguerza, Javier M e Alberto Muñoz: *Support vector machines with applications*. Statistical Science, páginas 322–336, 2006. 14
- [38] Friedman, Jerome, Trevor Hastie e Robert Tibshirani: *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001. 15, 18, 40
- [39] Lin, Hsuan Tien e Chih Jen Lin: *A study on sigmoid kernels for svm and the training of non-psd kernels by smo-type methods*. submitted to Neural Computation, 3:1–32, 2003. 16
- [40] Liu, Huan e Hiroshi Motoda: *Feature selection for knowledge discovery and data mining*, volume 454. Springer Science & Business Media, 2012. 17
- [41] Chizi, Barak, Lior Rokach e Oded Maimon: *A survey of feature selection techniques*. Em *Encyclopedia of Data Warehousing and Mining, Second Edition*, páginas 1888–1895. IGI Global, 2009. 17
- [42] Kohavi, Ron e George H John: *Wrappers for feature subset selection*. Artificial intelligence, 97(1-2):273–324, 1997. 17
- [43] Accock, Alan C: *Working with missing values*. Journal of Marriage and family, 67(4):1012–1028, 2005. 18, 28
- [44] Allison, Paul D: *Missing data*, volume 136. Sage publications, 2001. 18
- [45] Agresti, Alan: *An introduction to categorical data analysis*. Wiley, 2018. 19
- [46] Kelleher, John D, Brian Mac Namee e Aoife D’arcy: *Fundamentals of machine learning for predictive data analytics: algorithms, worked examples, and case studies*. MIT Press, 2015. 20
- [47] Azevedo, Ana Isabel Rojão Lourenço e Manuel Filipe Santos: *Kdd, semma and crisp-dm: a parallel overview*. IADS-DM, 2008. 26
- [48] Welch, Brent B, Ken Jones e Jeffrey Hobbs: *Practical Programming in Tcl/Tk*. Prentice Hall Professional, 2003. 30
- [49] Japkowicz, Nathalie *et al.*: *Learning from imbalanced data sets: a comparison of various strategies*. Em *AAAI workshop on learning from imbalanced data sets*, volume 68, páginas 10–15. Menlo Park, CA, 2000. 38