MLtool: A Tool to Automate the Construction, Evaluation, and Selection of Machine Learning Models

C. Mendes, A. de Barcelos, and S. Rigo

***Abstract*—The use of Machine Learning has intensified in recent years, gaining notoriety in the most diverse applications. Thus, there is a growing demand for professionals in this area, which has favored the entry of countless inexperienced users in the labor market. The path from the initial step to the use of algorithms and Machine Learning techniques in production environment takes considerable time, even from experts, due to the realization of interactive and manual tasks required for building predictive models that provide the desired results. In view of this, this work presents a tool to automate the construction, evaluation and selection of predictive models considering Machine Learning algorithms and parameters values more appropriate for each situation. The stages of feature selection, standardization, resampling, and training in the construction of models were considered in the tool. The proposed tool deal with the treatment of the problem of data unbalancing, as needed, as well as the execution control of the steps involved in the process of creating predictive models. The results obtained demonstrate that the order and choice of the steps and the values chosen for the algorithm parameters affect the final results of the generated models.**

***Index Terms*—Machine Learning, Model Selection, Automatic Algorithm Selection, Automated Machine Learning.**

I. Introdução

CRIADO a partir de pesquisas relacionadas à Inteligência Artificial, o Aprendizado de Máquina (ou *Machine Learning*, em inglês) é definido como um método de análise dos dados que automatiza o desenvolvimento de modelos analíticos. Por meio de algoritmos que aprendem a partir de dados, de forma interativa, torna-se possível a sua autossuficiência, minimizando assim a intervenção humana [1]. Neste sentido, o crescente volume e variedade de dados disponíveis aliado ao processamento computacional mais acessível e poderoso tem contribuído para o consequente aumento de interesse, tanto de empresas como de usuários, nessa área [2].

Muitas das atuais atividades do dia-a-dia são alimentadas por algoritmos de aprendizado de máquina, incluindo: detecção de fraudes, resultados de pesquisa na web, análise de sentimento baseado em texto, previsão de falhas em equipamento, entre outras [1]. A ampla variedade de técnicas que podem ser aplicadas durante este processo apresenta como consequência uma grande complexidade, mesmo para usuários especialistas em aprendizagem de máquina [3]. Tanto a escolha de algoritmos quanto de seus parâmetros de configuração impactam diretamente nos resultados. Deste modo, pode ser difícil para os usuários definirem a escolha correta quando são confrontados com esses graus de liberdade. Além disso, muitos dos processos da aprendizagem acabam por serem repetitivos, demandando horas de especialistas no envolvimento destes projetos e ainda, em muitas vezes, inviabilizando os mesmos. Diante deste cenário, destacam-se, na literatura, os trabalhos [4], [5] e [6], os quais buscam criar soluções para automatizar a seleção de algoritmos e valores de parâmetros. De forma geral, os trabalhos procuram tornar o uso de aprendizado de máquina mais acessível a usuários não-especialistas.

O uso de técnicas de aprendizado de máquina tem possibilitado a construção de modelos preditivos de forma rápida e automatizada. Como resultado, previsões de alto valor são geradas, podendo levar a melhores decisões e ações inteligentes em tempo real sem a intervenção humana. Segundo [7], existe grande complexidade a ser enfrentada quando tenta-se aplicar algoritmos de aprendizado de máquina para resolver problemas do mundo real. Em paralelo a isso, existem inúmeras implementações de modelos de aprendizado de máquina em produção onde nem todas são efetivas. Outra motivação, desta vez relacionada à arquitetura de software, vem da utilização crescente da abordagem de microserviços na construção de aplicações escaláveis. O uso dessa abordagem proporciona o desenvolvimento de serviços interoperáveis que podem facilmente ser reutilizados e compartilhados entre aplicações e empresas [8].

Neste contexto, o presente trabalho objetiva desenvolver uma ferramenta que, baseando-se nas etapas do KDD [9], realize a automatização da construção, avaliação e seleção de modelos de aprendizado de máquina. A ferramenta busca auxiliar tanto usuários leigos quanto especialistas no processo de escolha do algoritmo e valores de parâmetros mais adequados para cada situação. Como diferenciais, destaca-se o tratameto do desbalanceamento de dados durante o processo de construção de modelos e a possibilidade de configuração, por parte do usuário, do encadeamento das etapas do KDD. Sendo assim, o usuário será capaz de realizar experimentos alterando a ordem de execução das etapas envolvidas. Além disso, o presente trabalho elaborou e aplicou um questionário à um grupo de especialistas em aprendizado de máquina com o intuito de tomar conhecimento sobre a demanda por soluções que automatizam a construção, avaliação e seleção de modelos, bem como avaliar os os pontos considerados como diferenciais

C. Mendes, Universidade do Vale do Rio dos Sinos (Unisinos), São Leopoldo, Rio Grande do Sul, casmendes@edu.unisinos.br.

A. B. Silva, Universidade do Vale do Rio dos Sinos (Unisinos), São Leopoldo, Rio Grande do Sul, allanbs@unisinos.br.

S. J. Rigo, Universidade do Vale do Rio dos Sinos (Unisinos), São Leopoldo, Rio Grande do Sul, rigo@unisinos.br.

da solução proposta.

O presente trabalho está dividido da seguinte forma: a Seção II apresenta os trabalhos relacionados. Já na Seção III é descrito o modelo proposto e seu funcionamento, seguido pelo detalhamento de sua implementação na Seção IV. Na Seção V, a avaliação dos resultados obtidos em experimentos é apresentada. Por fim, as conclusões deste estudo são descritas na Seção VI.

II. Trabalhos Relacionados

Nesta seção são descritos os principais trabalhos que objetivam automatizar a seleção de modelos para um determinado problema de aprendizado de máquina. A revisão do estado da literatura buscou identificar os principais trabalhos na área de *Automated Machine Learning*,

Os trabalhos [4], [5] e [6] apresentam sistemas que visam automatizar a seleção de algoritmos e valores de parâmetros, cada um com suas particularidades. O recente estudo apresentado em [4] descreve um conjunto de técnicas utilizadas a fim de solucionar de uma melhor maneira o problema em questão. Já o trabalho [6] se destaca por ser um dos pioneiros na área de seleção automática de algoritmos e valores de parâmetros para um determinado problema de aprendizado de máquina. Por fim, o trabalho [5] apresenta um sistema distribuído, o qual propõe também uma linguagem declarativa simplificada com o objetivo de tornar o uso de aprendizado de máquina mais acessível à usuários não-especialistas. A seguir, nas subseções seguintes, os trabalhos mencionados são descritos em maiores detalhes.

O trabalho [4] descreve o PredicT-ML, um sistema que visa automatizar a construção de modelos de aprendizado de máquina com grandes dados clínicos. Para o autor, há dois grandes desafios existentes nesta área: (1) seleção eficiente e automática de algoritmos e valores de hiperparâmetros; e (2) automatização eficiente da agregação temporal de atributos clínicos. Para ambos os casos, o autor propõe soluções ao longo do trabalho.

PredicT-ML utiliza a lingugem Spark[1] para obter melhor desempenho na execução de algoritmos de aprendizado de máquina. A aplicação integra funções de aprendizado de máquina do WEKA e MLlib sendo que, para tal, faz uso tanto da interface de programação de aplicativo Java quanto de modificações no próprio código-fonte das soluções mencionadas. Ainda, no que se refere a fontes de dados, a abordagem utilizada por [4] é generalizável, permitindo a integração de dados entre diferentes sistemas de saúde.

A aplicação realiza o descarte de atributos através de técnicas de seleção para encontrar os menos promissores. Além disso, o Predict-ML possui como diferenciais a construção automática de um modelo de conjunto a partir dos demais já construídos durante o processo de pesquisa, a paralelização de testes nas primeiras rodadas do processo de busca e remoção de testes excessivamente demorados.

Em [5], os autores propõem um sistema distribuído, denominado MLbase, o qual fornece uma forma declarativa simplificada para usuários especificarem tarefas de ML. A aplicação possuí um otimizador que objetiva selecionar e dinamicamente adaptar a escolha do algoritmo de aprendizagem; um conjunto de operadores de alto nível que possibilitam a implementação de métodos de ML, por parte de usuários, sem que necessitem de conhecimento aprofundado sobre o mesmo; e um *runtime* otimizado para os padrões de acesso a dados considerando os operadores previamente definidos.

O MLbase possibilita gerar novas combinações de parâmetros e modelos em segundo plano, de modo a refinar o modelo objetivando melhores resultados e consequentemente tornando o processo mais interativo. Além disso, a ferramenta é extensível à novos algoritmos de ML, onde a plataforma oferece um conjunto de primitivas de alto nível com o intuito de simplificar a construção de algoritmos de aprendizado de máquina de forma distribuída. Apesar de possuir processos bem definidos e apresentarem alguns benefícios tanto para usuários comuns como para usuários especialistas, segundo os autores, o sistema ainda não foi completamente construído.

O trabalho de [6] foi considerado o pioneiro no âmbito de seleção automática de algoritmos e valores de hiperparâmetro para um determinado problema de aprendizado de máquina, o Auto-WEKA [6] busca trazer esse recurso de automatização para WEKA, uma ferramenta amplamente utilizada por apresentar um conjunto considerável de algoritmos de aprendizado de máquina para realizar tarefas de mineração de dados.

O Auto-WEKA utiliza os algoritmos já cadastrados na ferramenta WEKA para solucionar os problemas de seleção de algoritmos de aprendizado de máquina e valores de hiperparâmetros [6], onde cada algoritmo é tratado como um hiperparâmetro no nível raiz. Sendo assim, o Auto-WEKA mapeia o problema de seleção tanto algoritmos quanto valores de hiperparâmetros para apenas uma seleção de hiperparâmetros. Para tanto, a ferramenta utiliza dois parâmetros de alto-nível para controle dos métodos de classificação (determinando quais classificadores serão utilizados) e avaliação da aplicação dos métodos de seleção de atributos.

Auto-WEKA usa Configuração do Algoritmo baseado em Modelo Sequencial [10] e um modelo *Random Forest* para aproximar a dependência de precisão de um modelo nos valores de algoritmo e hiperparâmetro. Para o Auto-WEKA, a escolha da técnica de seleção de atributos também é considerada como um hiperparâmetro. Dessa forma, a ferramenta pode escolher automaticamente técnicas de seleção de atributos durante o processo de construção do modelo.

A. Comparação de Trabalhos Relacionados e Considerações

Para análise dos trabalhos citados, alguns critérios foram elencados, tais como (1) Computação distribuída: avalia se o sistema permite um processamento distribuído na execução dos algoritmos de classificação e das demais técnicas envolvidas no processo; (2) Extensível a novos algoritmos: este critério indica se o trabalho avaliado possibilita a inclusão de novos algoritmos de aprendizado de máquina pelo usuário no sistema proposto; (3) Trata desbalanceamento: indica se as bases de dados desproporcionais, em relação ao atributo classe, são tratadas anteriormente ao processo de classificação; e (4) Melhoria contínua do modelo: avalia se a arquitetura proposta no trabalho avaliado contempla o constante refinamento

[1]Disponível em https://spark.apache.org/.

do modelo a partir de uma primeira execução. Neste caso, considera-se que o processo de melhoria contínua do modelo seja executado em segundo plano (*background*), não havendo necessidade de interação do usuário.

Na Tabela I podemos ver a relação dos trabalhos encontrados frente aos critérios elencados.

TABELA I
COMPARATIVO ENTRE OS TRABALHOS RELACIONADOS

Trabalho	Computação distribuída	Extensível a novos algoritmos	Trata desbalanceamento	Melhoria contínua do modelo
PredicT-ML [4]	✓	-	Parcialmente	✓
MLbase [5]	✓	✓	-	✓
Auto-WEKA [6]	-	-	-	-
MLtool	✓	✓	✓	✓

Conforme pode ser observado na Tabela I, tanto o PredicT-ML quanto o MLbase fazem uso de computação distribuída, de forma a otimizar a execução dos algoritmos. Ambos também buscam refinar o modelo ao longo do tempo. Já o Auto-WEKA, considerado uma extensão para a ferramenta WEKA, não possui nenhuma destas características. No que se refere a extensibilidade, o MLbase fica à frente de seus adversários, pois permite aos usuários especialistas adicionarem novos algoritmos à plataforma para que estes sejam utilizados na construção de novos modelos. Ainda na Tabela I, observa-se que, ao contrário do presente trabalho, nenhum dos estudos relacionados considera relevante o tratamento de conjuntos de dados desbalanceados, desconsiderando os argumentos apresentados em [11]. No trabalho de [4], esse critério foi classificado como parcialmente atendido pelo fato de considerarem o uso de técnicas que tratam deste problema somente para a etapa de Seleção de Atributos, não sendo considerado a sua utilização durante as demais etapas do processo.

O comparativo apresentado na Tabela I demonstra que ao contrário dos trabalhos relacionados, o MLTool é o único que realiza tratamentos para o problema de desbalanceamento de dados. Segundo [12], o desbalanceamento dificulta o aprendizado de técnicas supervisionadas de classificação, neste caso, impactando diretamente a seleção de modelos. Ainda, o tratamento deste problema torna-se relevante por aparecer de forma frequente em cenários do mundo real [13]. Deste modo, o presente trabalho propõe uma ferramenta que trate o desbalanceamento de dados durante o processo de construção de modelos, diferenciando-se assim dos demais trabalhos. Além disso, outro diferencial da abordagem proposta é possibilitar a configuração, por parte do usuário, do encadeamento das etapas do KDD [9], de forma que o usuário seja capaz de realizar experimentos alterando-se a ordem de execução das etapas envolvidas. Assim como os trabalhos de [4] e [5], a ferramenta proposta no trabalho atual foi projetada de tal forma que seja possível escalar conforme necessidade, sendo possível aumentar seu poder computacional executando-a em diferentes máquinas, com recursos de computação distribuída.

III. MODELO PROPOSTO

Esta seção apresenta o modelo proposto neste trabalho, descrevendo inicialmente uma visão geral e, em seguida, o detalhamento da arquitetura do modelo.

A. Visão Geral

A visão geral do modelo proposto, ilustrada na Fig. 1, consiste na interação do usuário com a aplicação. Neste contexto, o usuário pode definir parâmetros/configurações para a construção dos modelos preditivos (serão vistos a seguir) bem como enviar conjuntos de dados para que sejam classificados.

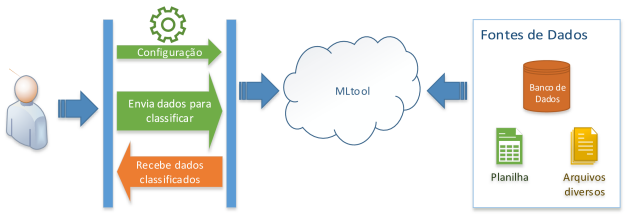

Fig. 1. Visão geral do modelo proposto.

Após o usuário dar início a operação de construção dos modelos preditivos, a aplicação inicia o fluxo de execução a fim de encontrar o melhor modelo preditivo para o conjunto de dados informado de acordo com as métricas selecionadas. Em seguida, ou até mesmo durante o processo de seleção de modelo, o usuário poderá enviar as amostras de forma a solicitar que estas sejam classificadas, desde que ao menos um modelo já tenha sido treinado. Como retorno, o usuário receberá as amostras classificadas.

B. Arquitetura

A arquitetura do modelo, ilustrada através da Fig. 2, contempla quatro elementos principais, sendo eles: Aplicação Web, Banco de Dados, Gerenciador de Execução e Executores. Nas seções a seguir, cada um desses elementos envolvidos será descrito.

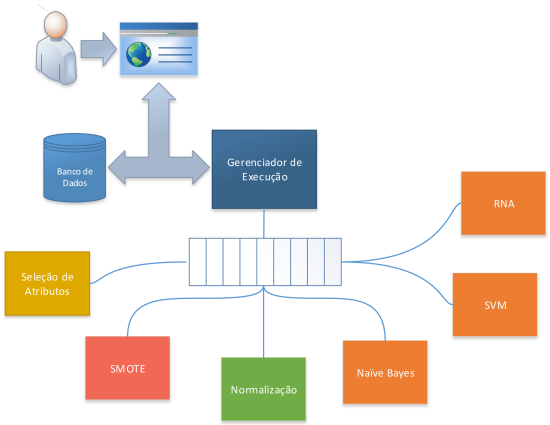

Fig. 2. Arquitetura do modelo proposto.

1) Aplicação Web: A comunicação com o usuário ocorre através de uma aplicação web, sendo o local onde este será capaz de definir novos modelos, configurar e importar os conjuntos de dados, analisar as métricas dos modelos preditivos criados (de forma automática), classificar novas amostras, entre outras funcionalidades. A aplicação web objetiva auxiliar

o usuário leigo ou especialista no uso da ferramenta como um todo. A definição de parâmetros considerados pertinentes por um usuário especialista, por exemplo, também poderá ser feita pela interface. Inclusive um dos pontos mais relevantes, que vão de encontro com a proposta deste trabalho, é o usuário ser capaz de definir o fluxo de execução do processo. Em outras palavras, o modelo proposto permite que o usuário defina a ordem e quais serão as etapas consideradas na execução. Entendem-se por etapas a Seleção de Atributos, Reamostragem, Normalização e Treino. Com isso, tornam-se possíveis rápidas experimentações sobre os dados, podendo optar por usar ou não a Seleção de Atributos, aplicar a Reamostragem antes da etapa de Normalização ou vice-versa, ou ainda considerar somente a etapa de Treino, não aplicando as etapas anteriores.

2) Banco de Dados: Para persistência, tanto de dados advindos do usuário como de informações pertinentes aos modelos preditivos treinados, foi utilizado um banco de dados, também representado na Fig. 2. Além disso, demais informações são mantidas, tais como: configurações gerais da aplicação; informações de acesso às fontes de dados; configurações específicas de cada conjunto de dados do usuário (atributo classe, atributos ignorados, entre outros); definições de modelos; configurações de etapas de execução (ordem e quais serão envolvidas em um determinado treino); configuração de algoritmos que serão aplicados em cada etapa do treino; parâmetros dos algoritmos; e os resultados de execução, onde armazenam-se modelos treinados, histórico de execução de cada um dos executores, assim como as métricas dos algoritmos de classificação.

3) Gerenciador de Execução: Sendo o principal elemento entre os demais, o Gerenciador de Execução é responsável por orquestrar todo o fluxo de execução do processo de geração de modelos preditivos. Além de publicar mensagens na fila, o Gerenciador de Execução também consome mensagens tanto da aplicação web como dos próprios executores através desta. Na Fig. 3 pode-se observar o fluxo do Gerenciador de Execução.

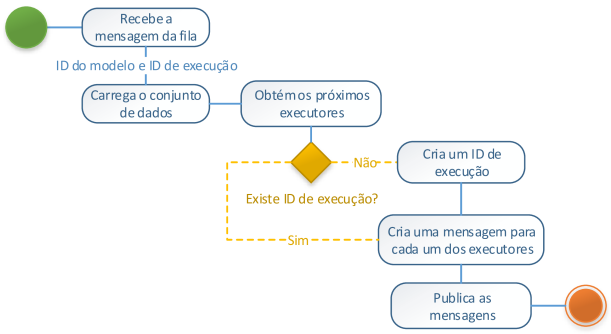

Fig. 3. Fluxo do Gerenciador de Execução.

Como já mencionado, o Gerenciador de Execução pode receber dois tipos de mensagens, as que indicam o início do processo de construção dos modelos preditivos — advindas da aplicação web quando o usuário dá início a operação — e as mensagens dos executores que indicam o término de sua respectiva execução. Sendo assim, quando o Gerenciador de Execução recebe uma mensagem de um executor ele identifica à qual etapa este pertence e dispara as novas mensagens para os executores da etapa seguinte, levando em consideração a ordem determinada pelo usuário. Esse processo ocorre até que se atinja a última etapa, normalmente de Treino.

Um dos grandes benefícios da forma como foi projetada esta arquitetura é a escalabilidade, já que caso sejam necessários mais executores de um determinado algoritmo/etapa pode-se simplesmente iniciar novas instâncias em diferentes máquinas para processar a demanda de mensagens contidas na fila.

4) Executores: O objetivo dos executores é processar as mensagens da fila sob demanda. Cada executor representa um ou mais algoritmos (poderia ter mais de um tipo de normalização, por exemplo, e ainda assim estaria dentro do mesmo executor). Para o protótipo foram implementados seis executores (vide Fig. 2). Além disso, cada um destes possui sua própria fila de mensagens. Dessa forma é possível distinguir e mensurar como está a demanda para cada um dos executores. Toda execução inicia-se a partir de uma mensagem recebida. Basicamente, o fluxo de um executor funciona da seguinte forma: 1) Obtém as informações da mensagem recebida (identificador de execução e parâmetros do algoritmo); 2) Carrega o conjunto de dados; 3) Executa o algoritmo utilizando os parâmetros recebidos; 4) Cria um novo conjunto de dados contendo os dados transformados pelo algoritmo executado, se necessário; 5) Salva os resultados da execução (parâmetros aplicados, tempos de execução, métricas, entre outros); 6) Publica uma mensagem sinalizando o término de execução para o Gerenciador de Execução.

Para a etapa de reamostragem há um executor responsável por tratar o desbalanceamento dos dados, o SMOTE [11]. Este, por sua vez, é responsável por gerar amostras sintéticas a partir das amostras existentes (técnica *over-sampling*) de forma a equilibrar a quantidade de amostras em ambas as classes.

Em especial na etapa de treino, onde ocorre o treinamento dos algoritmos, serão persistidos os modelos de aprendizagem de máquina obtidos. Desta forma, é possível o uso destes recursos posteriormente para classificar novas amostras.

C. Avaliação

Com o objetivo de validar a execução do protótipo para diferentes contextos de dados, foram utilizados dois conjuntos de dados disponíveis na plataforma UCI Machine Learning Repository[2]. Para esta análise, foi utilizado o Iris Data Set e então definidos quatro cenários, de forma a contemplar diferentes ordens de execução das etapas. Além disso, em alguns casos foram desconsideradas algumas etapas. O objetivo dessa avaliação foi promover a experimentação de diferentes cenários de uso do protótipo desenvolvido. Os cenários considerados foram:

- Cenário A: Seleção de Atributos, Normalização, Reamostragem e Treino
- Cenário B: Seleção de Atributos, Reamostragem, Normalização e Treino

[2]Disponível em http://archive.ics.uci.edu/ml/.

- Cenário C: Seleção de Atributos, Normalização e Treino
- Cenário D: Treino

O presente trabalho também buscou avaliar os cenários de utilização da ferramenta. Para tanto, como púlico-alvo foi foram considerados dois perfis de usuários: leigo e especialista em aprendizado de máquina. Conforme apresentado em [6], o número considerável de combinações de algoritmos e valores de seus respectivos parâmetros exige uma interação intensa e manual na construção de modelos até mesmo para especialistas da área. Neste quesito, a solução aqui proposta fornece para ambos os perfis de usuários a automatização da execução dessas combinações. Além disso, para os usuários leigos, o presente trabalho torna-se ainda mais interessante por não demandar dos mesmos um conhecimento aprofundado sobre as técnicas e algoritmos envolvidos no processo de mineração de dados, possibilitando que estes também façam uso destas tecnologias que hoje estão cada vez mais sendo utilizadas nas mais variadas aplicações.

Com o objetivo de se aproximarem da área, usuários leigos geralmente têm seu primeiro contato com técnicas de mineração de dados através de ferramentas como o WEKA, por exemplo. Entretanto, em ambos casos este perfil de usuários acaba por ter que aprender diversos conceitos envolvidos para então dar início aos seus experimentos. No caso do WEKA, em um fluxo simples, o usuário terá que: 1) selecionar o conjunto de dados; 2) importar o conjunto de dados na ferramenta; 3) realizar manualmente as etapas de pré-processamento, fazendo a escolha de técnicas, parâmetros e seleção de atributos quando necessário; 4) realizar transformações de dados quando necessário; 5) escolher o algoritmo classificador, assim como seus parâmetros; e 6) escolher o método de validação. Caso o usuário decida fazer experimentos com diferentes algoritmos e parâmetros com o objetivo de melhorar seus resultados terá que retomar o processo a partir da etapa 3, visto que este é um processo cíclico onde, normalmente, são feitas inúmeras experimentações para cada conjunto de dados.

Já na abordagem proposta, além do usuário não precisar, necessariamente, conhecer os algoritmos e técnicas envolvidas, ele poderá facilmente obter os modelos preditivos para o conjunto de dados especificado. Para isso, somente será necessário importar os dados históricos para a ferramenta, escolher a ordem de execução das etapas (caso deseje customizar) e iniciar o processo de construção de modelos. Com isso, os modelos preditivos, com diferentes combinações de algoritmos e parâmetros, serão gerados automaticamente. Cada um dos modelos será também avaliado de forma automática pela ferramenta e os resultados obtidos serão apresentados ao usuário. A partir disso, este poderá efetuar as classificações de novas amostras.

Para definição do cenário de usuários especialistas elaborou-se um questionário, composto por oito questões, o qual foi aplicado à um grupo de especialistas em aprendizado de máquina. Tais questões visam tomar conhecimento sobre a demanda por soluções que automatizam a construção, avaliação e seleção de modelos, bem como avaliar alguns pontos considerados como diferenciais do presente trabalho. Na Seção V são tecidos comentários sobre cada uma das questões aos participantes e avaliadas as suas respostas.

D. Parâmetros dos Algoritmos

Para o algoritmo *Information Gain* (etapa de Seleção de Atributos), foi especificado um parâmetro que consiste em determinar o número de atributos que serão considerados para um determinado conjunto de dados. Neste sentido, foram definidos quatro valores distintos para este parâmetro (5, 10, 15, 20).

Quanto à etapa de Normalização, considerou-se apenas o tipo de normalização Min-Max, onde seus parâmetros são o valor mínimo e máximo usado para normalização dos atributos numéricos do conjunto de dados, os quais foram 0 e 1, respectivamente. No caso do SMOTE (etapa de reamostragem), todos os experimentos contaram com as definições padrão do algoritmo para a biblioteca utilizada[3] e considerando as variações de tipo (*regular*, *borderline1* e *borderline2*).

Em relação aos algoritmos classificadores *Artificial Neural Network* (ANN) e *Support Vector Machines* (SVM), os parâmetros que foram utilizados pelos mesmos podem ser visualizados através das tabelas II e III, respectivamente. Já para o algoritmo Naïve Bayes foi utilizada a implementação Gaussian Naïve Bayes, para a qual nenhum parâmetro foi considerado.

IV. Aspectos de Implementação

Para o desenvolvimento dos componentes apresentados foram utilizados um conjunto de ferramentas, tecnologias e bibliotecas. Conforme dito anteriormente, a aplicação foi dividida em 4 componentes, dos quais abordaremos 3 deles nesta seção: banco de dados, gerenciador de execução e executores.

De forma a persistir os dados gerados pelo MLtool, assim como as informações cadastradas pelo usuário, em sua forma estruturada, optamos pelo uso do banco de dados SQL Server.

Antes de adentrarmos nos aspectos de implementação dos demais elementos, precisamos falar sobre o mecanismo central da arquitetura do MLtool, o servidor de mensageria (também conhecido por *Message-Oriented Middleware*). Para a construção deste projeto, optamos pelo uso do RabbitMQ, o qual possuí código aberto e foi escrito na linguagem de programação Erlang para melhorias de performance. Assim como outras ferramentas similares, o RabbitMQ implementa o protocolo de troca de mensagens AMQP. Tal aspecto possibilita a integração com diferentes linguagens de programação, como Python, Java, C#, entre outras. Um dos recursos mais comuns dos servidores de mensageria são as filas de mensagens (ou *queues*, em inglês). No MLtool, utilizamos uma fila para cada um dos executores criados, de forma a distinguir e mensurar a demanda de cada um deles.

Tanto para o desenvolvimento do gerenciador de execução bem como dos executores, utilizamos Python como linguagem de programação. O gerenciador de execução é um dos elementos mais complexos da arquitetura do MLtool, pois é

[3]Disponível em: https://imbalanced-learn.readthedocs.io/en/stable/index.html.

TABELA II
PARÂMETROS DO ALGORITMO ANN

#	Hidden Layer Sizes	Activation	Solver	Alpha	Batch Size	LR	LR Init	Power T	Max Iter	Shuffle	Random State	Tol	Momentum
1	(100,)	relu	adam	0,0001	0	constant	0,001	0,5	200	1	1	0,0001	0,9
2	(100,)	relu	adam	0,0001	0	constant	0,001	0,5	200	1	1	0,0001	0,7
3	(100,)	relu	adam	0,0001	0	constant	0,01	0,5	300	1	1	0,0001	0,7
4	(10,10,10,10,10,)	relu	adam	0,0001	0	constant	0,01	0,5	300	1	1	0,0001	0,7
5	(10,10,10,10,10,)	logistic	adam	0,0001	0	constant	0,01	0,5	300	1	1	0,0001	0,7
6	(10,10,10,10,10,)	logistic	sgd	0,0001	0	constant	0,01	0,5	300	1	1	0,0001	0,7
7	(10,10,10,10,10,)	logistic	lbfgs	0,0001	0	constant	0,01	0,5	300	1	1	0,0001	0,7
8	(100,)	relu	adam	0,0001	0	adaptive	0,001	0,5	200	1	1	0,0001	0,9
9	(100,)	relu	adam	0,0001	0	constant	0,01	0,5	400	1	1	0,0001	0,7

TABELA III
PARÂMETROS DO ALGORITMO SVM

#	C	Kernel	Degree	Gamma	Coef0	Probability	Shrinking	Tol	Cache Size	Max Iter	Decision Function Shape	Random State
1	10	rbf	3	0	0	0	1	0,001	200	-1	ovr	1
2	10	linear	3	0	0	0	1	0,001	200	-1	ovr	1
3	10	poly	3	0	0	0	1	0,001	200	-1	ovr	1
4	10	sigmoid	3	0	0	0	1	0,001	200	-1	ovr	1

nele que estão contidas todas as regras necessárias para que o fluxo de geração, avaliação e seleção dos modelos preditivos seja realizado. O processo de orquestração do fluxo acaba por ser bastante intenso, pois é o gerenciador quem criará mensagens na fila do servidor de mensageria, de modo que os executores tenham conhecimento de qual informação e quando processar. Com isso, vale ressaltar que até mesmo o gerenciador de execução pode ser escalado, podendo haver inúmeros gerenciadores executando em paralelo de forma a suprir a demanda.

Por fim, o desenvolvimento dos executores se torna mais acessível, visto que estes não conhecem o fluxo de execução e são independentes entre si. Sendo assim, caso seja necessário adicionar novos algoritmos de *Machine Learning*, basta que um novo executor seja criado e considerado pelo fluxo do MLtool. No caso destes executores, utilizamos a biblioteca *scikit-learn* por apresentar um vasto conjunto de algoritmos já implementados, porém a implementação possibilita o uso de outras bibliotecas e algoritmos fornecidos por estas.

V. RESULTADOS

Os resultados obtidos nos diferentes cenários propostos na subseção III-C são apresentados na Fig. 4, a qual possui no eixo X o número de modelos preditivos criados considerando diferentes combinações (etapas e parâmetros dos algoritmos) e no eixo Y o valor da métrica F1 para cada um dos modelos treinados nos diferentes cenários. Analisando a Fig. 4 pode-se notar que, para o conjunto de dados testado, o Cenário D apresenta alguns resultados que superam até mesmo os resultados dos demais cenários, mesmo que não se tenham aplicadas as etapas de seleção e transformação de dados.

Alguns dos modelos preditivos gerados obtiveram resultados aquém do desejado, com *Mean Squared Error* acima de 1,8, enquanto outros alcançaram valores abaixo de 0,1. A variação nos resultados, com base na parametrização dos modelos, reforça ainda mais a importância de soluções que automatizem a construção de modelos para o auxílio de usuários especialistas. Conforme dito, estes, despendem horas de experimentações sobre os dados e acabam por ter que

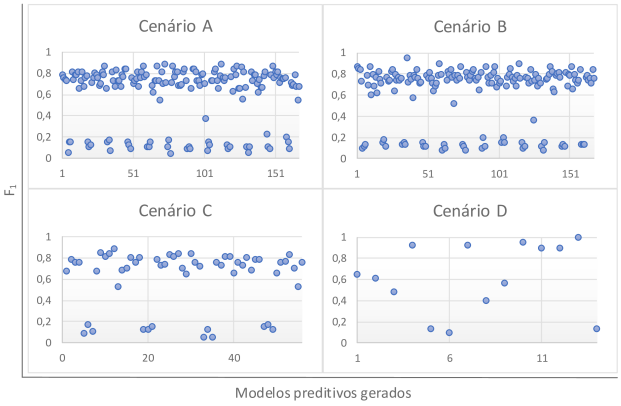

Fig. 4. Comparativo de cenários considerando a métrica F_1.

executar as etapas com diferentes combinações de parâmetros, estando sujeitos a esse tipo de variação [3], [6]. Neste tipo de solução é possível executar um número maior de combinações sem que haja necessidade de intervenção humana. Sendo assim, as combinações tanto de parâmetros como de etapas pouco promissoras são rapidamente descartadas.

Outro ponto que pode ser observado, ainda na Fig. 4, é a diferença do número de modelos preditivos criados para cada cenário, sendo que isso ocorre por conta da ordem e escolha das etapas que farão parte do processo de geração de modelos preditivos. Para a execução dos diferentes cenários não foram modificados os parâmetros dos algoritmos, pois o objetivo era justamente verificar o impacto causado pela alteração do fluxo do processo.

Na Fig. 5 é apresentado um comparativo entre os classificadores nos cenários estabelecidos. O eixo X representa os cenários e o eixo Y representa a métrica F_1 para os melhores modelos preditivos gerados em cada um dos cenários. Apesar das alterações de ordem e remoção de algumas etapas, pode-se observar que os algoritmos ANN e SVM tiveram resultados similares nos cenários B e C, enquanto que no cenário A o algoritmo ANN teve destaque, assim como no cenário D o

TABELA IV
MELHORES RESULTADOS, POR CLASSIFICADOR, PARA CADA UM DOS CENÁRIOS, CONSIDERANDO A MÉTRICA *Mean Squared Error* (MSE)

Worker	Parameters*	F1 Score	Accuracy	Precision Score	Recall Score	Kappa	Mean Squared Error
				Cenário A			
ANN	4	0,92105	0,92105	0,92279	0,92105	0,88113	0,07895
SVM	1	0,81840	0,81579	0,82824	0,81579	0,71762	0,18421
NB	-	0,72591	0,73684	0,72704	0,73684	0,59746	0,34211
				Cenário B			
Worker	**Parameters***	**F1 Score**	**Accuracy**	**Precision Score**	**Recall Score**	**Kappa**	**Mean Squared Error**
SVM	1	0,91887	0,92105	0,93060	0,92105	0,88199	0,07895
ANN	4	0,89090	0,89474	0,91049	0,89474	0,83460	0,10526
NB	-	0,87009	0,86842	0,87336	0,86842	0,79787	0,13158
				Cenário C			
Worker	**Parameters***	**F1 Score**	**Accuracy**	**Precision Score**	**Recall Score**	**Kappa**	**Mean Squared Error**
SVM	1	0,92334	0,92105	0,93797	0,92105	0,88162	0,15789
ANN	3	0,82971	0,81579	0,90175	0,81579	0,72234	0,18421
NB	-	0,76246	0,76316	0,78057	0,76316	0,64338	0,39474
				Cenário D			
Worker	**Parameters***	**F1 Score**	**Accuracy**	**Precision Score**	**Recall Score**	**Kappa**	**Mean Squared Error**
SVM	2	0,97392	0,97368	0,97632	0,97368	0,95992	0,02632
ANN	4	0,91845	0,92105	0,93233	0,92105	0,87248	0,07895
NB	-	0,83941	0,84211	0,84398	0,84211	0,75745	0,15789

* Referencia os parâmetros descritos nas Tabelas II e III (Seção III-D).

SVM alcançou um melhor resultado. Já o algoritmo Naïve Bayes (NB) apresentou resultados inferiores aos demais em todos os cenários, tendo seu melhor resultado no cenário B, onde ficou mais próximo de seus concorrentes. Ainda, observa-se que, nesse contexto, as etapas anteriores ao treino impactaram negativamente nos resultados do SVM, visto que no cenário D obteve o melhor resultado.

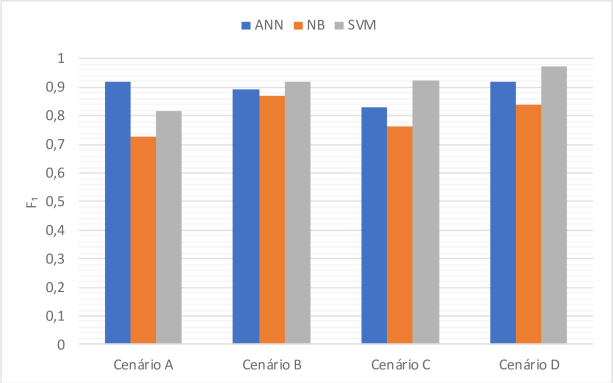

Fig. 5. Comparativo dos resultados obtidos por diferentes classificadores.

Na Tabela IV são apresentadas as métricas obtidas em cada um dos cenários descritos anteriormente. Além destas, O MLtool também captura as métricas *AUC*, *ROC*, *Log Loss*, *Sensitivity*, *Specificity*, *Positive Predictive Value* e *Negative Predictive Value*, porém estas somente são consideradas para conjuntos de dados que possuem classe binária e não multi-classe, como é o caso do conjunto de dados Iris.

No que tange a avaliação dos cenários de uso da ferramenta, a primeira e segunda questão objetivam compreender a real demanda por ferramentas que auxiliem usuários especialistas na construção de modelos preditivos. Neste quesito, segundo respostas, a maioria dos especialistas utiliza ou ao menos já utilizou esse tipo de ferramenta. Além disso, todos os especialistas atribuíram a pontuação máxima em relação ao seu interesse por essas soluções. Já a terceira e quarta questões remetiam ao tempo investido por estes usuários na parametrização dos algoritmos. Segundo respostas, em média 15% do tempo total empenhado na construção dos modelos preditivos é destinado às parametrizações dos algoritmos.

Na questão 5 buscou-se entender o impacto causado pela ordem e escolha das etapas nos resultados considerando a visão dos especialistas em aprendizado de máquina. Neste ponto, o resultado obtido foi unanime em afirmar que consideram relevante o fluxo de execução das etapas quando se refere a obtenção de melhores resultados. Já a sexta questão se atém a relevância do tratamento do desbalanceamento de dados quanto a obtenção de melhores resultados. De forma geral, os especialistas consideram que a falta de tratamento pode realmente impactar nos resultados, sendo importante tratar na maioria dos casos os conjuntos de dados que possuam tal característica. A sétima questão tenta evidenciar o esforço dedicado por especialistas nas etapas de Seleção de Atributos, Normalização, Reamostragem e Treino. Observa-se que o relato do maior percentual de esforço investido concentra-se nas etapas de Seleção de Atributos e Treino. Por fim, a última questão se refere a sugestões dos especialistas em relação a algoritmos que consideram relevantes para serem incorporados à ferramenta proposta. Estes, por sua vez, estão compreendidos nas diferentes etapas, sendo que os algoritmos *Random Forest* e *Decision Tree* fazem parte da etapa de treino e o algoritmo *Random Undersampling* da etapa de

reamostragem, por exemplo.

Os resultados apresentados neste trabalho demonstram que o controle de execução das etapas do processo KDD se torna um ponto relevante durante a criação de modelos preditivos, visto que impactam os seus resultados finais. Ainda, as opiniões coletadas de especialistas, por meio do questionário aplicado, reforçam esta análise.

VI. Conclusão

Neste trabalho foi apresentada uma ferramenta para automatizar a construção, avaliação e seleção de modelos de aprendizado de máquina, a qual considera algoritmos que tratam do problema de desbalanceamento de dados, quando necessário. Outro aspecto importante é a possibilidade do controle e customização de execução das etapas de seleção de atributos, normalização, reamostragem e treino. Além disso, foram apresentados os conceitos envolvidos no processo KDD, os quais foram essenciais para a elaboração deste trabalho. O tópico KDD é muito amplo, pois contempla inúmeras técnicas e algoritmos para realização de seu processo [9], [14]. Este fato, justifica ainda mais a demanda existente por ferramentas que auxiliem na escolha do algoritmo e parâmetros mais adequados para cada situação, visto que a execução interativa e manual das etapas do processo do KDD é considerada uma tarefa árdua até mesmo para especialistas em aprendizado de máquina [3].

Como trabalhos futuros, destacam-se algumas propostas. A primeira delas é incorporar novos algoritmos à ferramenta atual, atentando-se às sugestões de usuários especialistas em aprendizado de máquina. Otimizar a escolha de parâmetros dos algoritmos objetivando reduzir o tempo de execução do processo como um todo, assim como melhorar os resultados dos modelos preditivos gerados. Considerar o uso de outras técnicas, como a filtragem por correlação, para a etapa de Seleção de Atributos, visto que alguns trabalhos abordam o assunto como uma alternativa plausível em relação aos demais algoritmos [15], [16]. Além de avaliar o uso de algoritmos para remoção de *outliers*, pois tais registros podem distorcer predições consequentemente afetando os resultados [17].

Referências

[1] I. Bose and R. K. Mahapatra, "Business data mining—a machine learning perspective," *Information & management*, vol. 39, no. 3, pp. 211–225, 2001.

[2] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2016.

[3] E. R. Sparks, A. Talwalkar, D. Haas, M. J. Franklin, M. I. Jordan, and T. Kraska, "Automating Model Search for Large Scale Machine Learning," in *Proceedings of the Sixth ACM Symposium on Cloud Computing*. ACM, 2015, pp. 368–380.

[4] G. Luo, "PredicT-ML: a tool for automating machine learning model building with big clinical data," *Health Information Science and Systems*, vol. 4, p. 5, 2016.

[5] T. Kraska, A. Talwalkar, J. C. Duchi, R. Griffith, M. J. Franklin, and M. I. Jordan, "MLbase: A Distributed Machine-learning System," in *CIDR*, vol. 1, 2013, pp. 2–1.

[6] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2013, pp. 847–855.

[7] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison, "Hidden Technical Debt in Machine Learning Systems," in *Advances in Neural Information Processing Systems*, 2015, pp. 2503–2511.

[8] M. Fowler and J. Lewis, "Microservices," mar 2014, acessado em: 04/06/2017.

[9] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From Data Mining to Knowledge Discovery in Databases," *AI Magazine*, vol. 17, no. 3, p. 37, 1996.

[10] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential Model-Based Optimization for General Algorithm Configuration," in *International Conference on Learning and Intelligent Optimization*. Springer, 2011, pp. 507–523.

[11] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[12] J. Van Hulse, T. M. Khoshgoftaar, and A. Napolitano, "Experimental perspectives on learning from imbalanced data," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 935–942.

[13] N. V. Chawla, *Data Mining for Imbalanced Datasets: An Overview*. Boston, MA: Springer US, 2005, pp. 853–867.

[14] C. Raquel Woszezenki, A. Goncalves, and J. Souza, "Knowledge discovery model based on semantic and temporal associations between textual elements," *IEEE Latin America Transactions*, vol. 16, pp. 1243–1249, 04 2018.

[15] L. Yu and H. Liu, "Feature Selection for High-Dimensional Data: A Fast Correlation-Based Filter Solution," in *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, T. Fawcett and N. Mishra, Eds., 2003, pp. 856–863.

[16] M. A. Hall, "Correlation-based Feature Selection for Discrete and Numeric Class Machine Learning," in *Proceedings of the Seventeenth International Conference on Machine Learning*, ser. ICML '00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 359–366.

[17] C. C. Aggarwal, "Outlier analysis," in *Data Mining*. Springer, 2015, pp. 237–263.

Cassiano Mendes é graduado em Ciência da Computação pela Universidade do Vale do Rio dos Sinos (Unisinos), São Leopoldo, Rio Grande do Sul, Brasil, em 2017. Faz parte de uma equipe de inovação, onde busca criar soluções inovadoras para o setor educacional. Possui experiência em algoritmos e técnicas de Aprendizado de Máquina, também em Arquitetura de Software de aplicações. Suas áreas de interesse são: Inteligência Artificial, Arquitetura de Software e Mineração de Dados Educacionais.

Allan de Barcelos Silva BSc in Information Systems at Universidade do Vale do Rio dos Sinos - UNISINOS (2015); MSc in Applied Computing at Universidade do Vale do Rio dos Sinos - UNISINOS (2017); Professor and Researcher at UNISINOS University. Work on Data Mining, Natural Language Processing (NLP), Machine Learning and Artificial Intelligence areas, since 2012. Innovation Consultant on knowledge transfer projects with the Industry.

Sandro José Rigo Professor/Researcher in the Applied Computing Graduate Program Program – UNISINOS (since 2011); Bsc in Computer Science at Pontifícia Universidade Católica do Rio Grande do Sul PUCRS (1990); MsC in Computer Science at Universidade Federal do Rio Grande do Sul UFRGS (1993); PhD in Computer Science at Universidade Federal do Rio Grande do Sul UFRGS (2008). Professor and Researcher at UNISINOS University, since 1995.