Desenvolvimento de um Framework Integrador de Mineração de Dados Educacionais

Development of an Integrated Framework for Educational Data Mining

Italo Yoshito Fujisawa[1] orcid.org/0000-0002-5026-1722
Alexandre Magno de Andrade Maciel[1] orcid.org/0000-0003-4348-9291

[1] Escola Politécnica de Pernambuco, Universidade de Pernambuco, Recife, Brasil,

E-mail do autor principal: iyf@ecomp.poli.br

Resumo

Com o avanço da tecnologia nos últimos tempos, a área de educação a distância (EAD) vem ganhando espaço na sociedade moderna. Professores e alunos não precisam, necessariamente, estar fisicamente próximos no processo de aprendizagem, *utilizando sistemas gerenciadores de cursos que são categorizados como "Ambiente Virtual de Aprendizagem" (AVA) para realizar a comunicação* que, também geram grandes volumes de dados. Os dados gerados não são muito exploradas devido a grande dificuldade existente em realizá-lo pela sua diversidade de informações e seus formatos. Em meio a esse problema, o presente trabalho tem como objetivo desenvolver um *framework* integrador de mineração de dados, inicialmente testado com o AVA chamado Moodle, para permitirem que desenvolvedores construam com facilidade aplicações para as pessoas realizarem análises de maneira facilitada utilizando técnicas de mineração de dados no intuito de obter informações mais aprimoradas. Para Atingir tais objetivos foram construídas: uma aplicação *Web* para demonstração do *framework* integrador utilizando o *framework* chamado React criada pela Facebook, e a proposta deste trabalho foi construída em forma de *WebService* utilizando a linguagem *python*, juntamente com a biblioteca *Flask.*

Palavras-Chave*:* Ambiente Virtual de Aprendizagem. Mineração de Dados; *Framework*; Web Service. *Moodle*;

Abstract

With the advancement of technology in recent times, the field of distance learning has been gaining space in modern society. Teachers and students do not necessarily need to be physically close in the learning process, using course management systems that are categorized as "Virtual Learning Environment" (AVA) to perform communication that also generate large volumes of data. The data generated is not much explored due to the great difficulty in performing it for its diversity of information and formats. In the midst of this problem, the present work aims to develop a data mining integrator framework, initially tested with AVA called Moodle, to allow developers to easily build applications for people to perform analyzes in an easy way using datamining techniques. To achieve these objectives, a Web application for demonstration of the integrator framework was created using the framework called React created by Facebook, and the proposal of this work was built as a Web Service using the python language, together with the Flask library.

Key-words:*Virtual Learning Enviroment; Data Mining; Framework; Web Service; Moodle; React; Python; Flask;*

 http://dx.doi.org/10.25286/repa.v3i3.976

1 Introdução

No processo de ensino da sociedade moderna observa se que cada vez mais a presença da utilização de ferramentas eletrônicas de apoio ao ensino. Isso se dá pelo motivo das informações estarem presentes eletronicamente e de fácil acesso para todos e de maneira instantânea através de dispositivos eletrônicos conectados à *internet*, o mesmo sendo considerado como principal fonte de informação e estudo da atualidade. **[1]** E com a grande disseminação da informação pela *internet*, fez com que ocorresse grandes mudanças no setor da educação; o aprimoramento e formalização da educação a distância (EAD) **[2]**.

Com o surgimento dessa modalidade de ensino e com o avanço tecnológico, foram desenvolvidas sistemas categorizados como "Ambientes Virtuais de Aprendizagem" (AVA) para distribuição e gerenciamento de disciplinas ministradas pelas instituições de ensino, com o objetivo de aproveitar a facilidade que o meio eletrônico e a *internet* permite de ter acesso aos dados em diversos tipos de dispositivos em diversos formatos **[3]**.

O Moodle é um exemplo de uma ferramenta dessa categoria. Ela consiste em ser uma plataforma de ensino *open-source*, ou seja, é disponibilizado gratuitamente sendo de fácil customização **[4]**. Além disso, a plataforma possui uma grande maturidade, ampla comunidade de desenvolvedores, grande quantidade de documentação.

O Brasil, segundo o próprio Moodle **[15]**, é o quarto país que possui mais sites registrados que estão em funcionamento no momento, usados principalmente para o gerenciamento de cursos e disciplinas em universidades.

Esses tipos de sistemas estão gerando grandes volumes de dados que são poucos analisados e explorados pela sua grande dificuldade existente para realizar esse processo. O motivo disso está no fato por dados possuírem diversos formatos sem nenhuma padronização. Assim permitindo que possíveis informações úteis, que poderiam ser usados para o melhoramento dos cursos prestados pela instituição educacional, sejam despercebidas **[6]**.

Uma forma de contornar esse problema seria utilizar ferramentas que permitissem a automação do processo de análise que resume ou agrega os dados de maneira a extrair informações relevantes de forma visualmente notável para auxiliar no acompanhamento ou até na tomada de decisão do instrutor. Decisões estas que poderiam vir a ser utilizada na melhoria da qualidade dos cursos.

Gonçalves **[10]** propôs uma arquitetura de *framework* de mineração de dados integrado ao Moodle em forma de aplicação Web, com o objetivo de permitir que desenvolvedores construam ferramentas de visualização e análise de dados que usem técnicas de mineração de dados de maneira facilitada.

A estrutura proposta no trabalho de Gonçalves utiliza-se da linguagem R em conjunto com a biblioteca *Shiny*, aproveitando-se da facilidade de criação e utilização de técnicas de mineração de dados em que a tecnologia permite ter. Cada visualização ou análise consiste em ser uma aplicação Web construída utilizando o *Shiny* que é integrado ao Moodle em forma de extensão, que muitas vezes são popularmente chamados de *plugins*. Além disso, a arquitetura apresentou ser pouco flexível por ser especificamente integrado ao Moodle, não permitindo integrar em outros tipos de sistemas.

Dito isto, este trabalho tem como objetivo propor um *framework* de mineração de dados no contexto educacional que seja flexível e que possa permitir ser usado em diversos outros sistemas, independente da tecnologia que ela esteja usando. Permitindo que desenvolvedores usem em seus sistemas AVA, para analisar e explorar os dados gerados por elas sem dificuldades.

2 Fundamentação Teórica

2.1 Mineração de Dados Educacional

O uso de sistemas na forma de aplicações Web vem ganhando muita popularidade nos últimos tempos que como consequência causou diversas mudanças em diversos setores que vem utilizando esse tipo de tecnologia. como por exemplo: Nos setores da Educação, Saúde, Negócios e etc. Na educação, é muito comum observar que, está cada vez mais presente, instituições que se utilizam de plataformas de ambientes virtuais de

aprendizagem (AVA) para ministrar cursos, presenciais ou semipresenciais **[15]**.

Esses sistemas geram diversos tipos de dados oriundas das interações entre os elementos envolvidos que são alunos e professores. Informações como: quantidades de vezes que o aluno entrou, atividades submetidas por elas, desempenho de alunos em disciplinas, trocas de mensagens entre alunos e professores e etc, são todos armazenados no banco de dados do sistema.

Devido a sua grande quantidade de dados, surge novas possibilidades e ambientes que poderiam ser analisados por intermédio do uso de técnicas de mineração de dados no contexto educacional que é denominada Mineração de dados Educacional **[6]**.

Ela consiste em ser um sub-ramo da mineração de dados que está preocupada em aplicar suas técnicas para detectar padrões em grandes conjuntos de dados educacionais, que de outra forma seriam difíceis ou impossíveis de ser percebidas ou analisadas devido ao enorme volume de dados e variedade de formatos.

2.2 Framework

De acordo com Sommerville **[8]**, a engenharia de software trata-se de um ramo da ciênciada computação ou engenharia da computação que é voltada à especificação, desenvolvimento e manutenção de sistemas de software em relação a todos os aspectos de produção.

Atualmente, é essencial a utilização de técnicas e conceitos dessa área no desenvolvimento de qualquer projeto de software para seguir boas práticas no intuito de evitar uma possível inviabilidade na manutenção, consequentemente, na descontinuidade de um projeto. A reutilização de componentes de software é uma das ideias propostas pela área que ajuda a mitigar esse tipo de problema.

Na literatura, existem diversas definições de *framework* e que podem ser diferentes dependendo do contexto em que esteja sendo aplicada:

- Segundo Fayad **[12]**, *framework* consiste em um conjunto de classes abstratas para a solução de uma família de problemas.
- Em Mattson **[13]**, é uma estrutura pré-definida com o objetivo de atingir a máxima reutilização de componentes de software.
- Para Johnson **[11]**, framework consiste em um conjunto de objetos que colaboram com o objetivo de atender a um conjunto de responsabilidades para uma aplicação específica ou um domínio de aplicação.

As definições de Fayad e Mattson estão no contexto de desenvolvimento de software. A definição de Johnson está bem genérica, podendo ser aplicada tanto no sentido de desenvolvimento como na área de gestão de projetos, onde cada componente do *framework* consiste em atividades ou ações bem definidas para atingir um determinado objetivo etapa a etapa.

Para o presente artigo, a definição de Mattson se aproximou mais do posicionamento do trabalho, portanto consideramos ela como definição do framework.

2.3 Web Service

Segundo o *Open soft* **[7]**, *Web Service* trata-se de uma solução ou aplicação que oferecem serviços ou informações que são consumidos por outras. As informações são solicitadas por requisições como POST, GET, UPDATE, DELETE e etc. Ela pode ser implementada usando diversas linguagens de programação existentes, no entanto há uma necessidade de padronização no formato de transmissão de informações para que a comunicação seja efetiva com diversos sistemas.

Então foram criadas protocolos para fazer esse intermédio entre o WS e as aplicações que irão consumir o serviço, como por exemplo: REST e SOAP. Apesar de existir outros protocolos, esses dois formatos são os mais conhecidos hoje na literatura.

O protocolo REST padroniza o formato dos dados enviados em JSON, que é uma estrutura bem similar a objetos da linguagem *javascript* ou dicionários em outras linguagens. No protocolo SOAP, as informações são passadas no formato de XML.

http://dx.doi.org/10.25286/repa.v3i3.976

3. *Framework* Proposto

No presente trabalho, foram construídas duas aplicações distintas: uma em forma de *Web Service* (WS), que consiste em ser uma aplicação Web que provê recursos a diversas outras aplicações através de requisições HTTP, não possuindo uma interface como em outras aplicações Web. A outra aplicação foi construída como uma aplicação Web comum que será consumidora dos recursos do WS.

Na aplicação WS, é onde está presente toda a estrutura do framework proposto, ela foi construída nesse tipo de tecnologia pelo motivo de permitir que ela seja usada por vários tipos de AVA's, independente da tecnologia usada na sua construção, que é a principal ideia por trás do WS.

A aplicação Web comum, foi construída somente como objetivo de demonstração do uso do *framework*. Ambas aplicações são descritas com detalhe nos tópicos seguintes.

3.1 Ferramentas Utilizadas

As tecnologias utilizadas para a elaboração do *framework* Python e React.

Python como linguagem no *back-end* por ser uma linguagem referência quando se trata de ciência dos dados, juntamente com a linguagem R **[16]**. Em conjunto, foi utilizada o *Scikit-learn* que consiste em ser uma biblioteca de *Machine Learning* que possui diversos classificadores já implementados e prontos para uso. Também foi usada a biblioteca *Flask* que é usada na criação do *Web Services*, fácil de ser utilizada, exigindo uma curva de aprendizado relativamente pequena. **[17]**

Para a confecção da aplicação Web que consumirá o WS, foi decidido utilizar o *framework front-end* chamado React, desenvolvido pelo Facebook. Essa ferramenta trouxe um novo paradigma de desenvolvimento *Web* que consiste em modelar a aplicação separados em componentes. Componentes estes que possuem "estados" armazenados, e com base nesses estados o componente pode se comportar e ser exibido de maneira diferente, dependendo de como foi programado para lidar com tal situação.

Essa forma de desenvolver uma aplicação *Web* faz com que o desenvolvedor deixe toda a aplicação separadas em peças independentes que podem ser facilmente aproveitados em outros projetos, ou seja, deixando altamente escalável permitindo adicionar, com facilidade, novas funcionalidades e recursos na aplicação. Além dessa vantagem, ela foi escolhida para desenvolver uma aplicação *single-page* ou de página única. Isso garante um bom desempenho porque a página é carregada apenas uma única vez.

3.2 Arquitetura

A arquitetura da ferramenta, em uma visão mais ampla, está ilustrada na Figura 1. Como foi mencionado anteriormente, foram construídas duas aplicações. A primeira sendo uma aplicação React fazendo a comunicação com o segundo que é um *Web Service*, seguindo o padrão REST. A segundo aplicação se comunica diretamente com a base de dados do Moodle extraindo informações e dados via conexão direta com o banco.

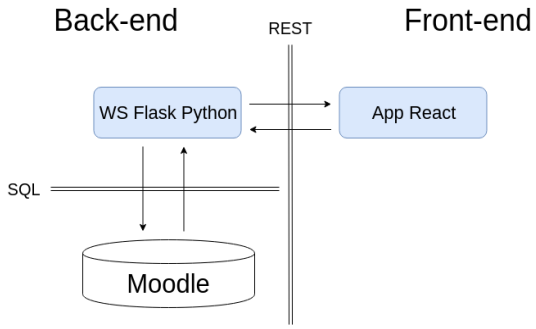


Figura 1 - Visão Geral do Projeto.
Fonte: Autor (2018).

3.2.1 Aplicação React

No desenvolvimento de uma aplicação React é comumente utilizado um script que automatiza a geração de uma aplicação básica com requisitos mínimos para começar o seu desenvolvimento, chamada create-react-app. Ela foi criada pela própria equipe de desenvolvimento do Facebook.

A aplicação proposta possui a seguinte estrutura de pastas, além das criadas pelo *scriptcreate-react-app*, sendo uma delas a pasta "*src*", e interna a elas foram criadas outras pastas:

- **Components:** Pasta com finalidade de guardar todos os componentes da interface do usuário.
- **Endpoints:** onde é guardada um arquivo javascript com variáveis constantes especificando todas as URLs de endpoints do Web Service.
- **Store:** Pasta que contém 2 outras pastas: actions e reducers que são relativos a utilização do Redux, um framework utilizado internamente ao React com o objetivo de permitir a centralização de estados dos componentes da aplicação em um local, assim fazendo com que a manipulação e comunicação entre os estados da aplicação se torne mais fácil.
- **Utils:** Local onde é armazenada arquivos javascript com funções auxiliares e de utilidades com diversas finalidades. Como por exemplo, funções de manipulação de listas, dados temporais, *Strings* e etc.
- **Views:** Arquivos javascript possuindo macro componentes React que, juntamente com os componentes localizados na pasta "components", forma uma página Web por completo.
- **Visualization:** Pasta onde está localizada os componentes referentes a cada visualização criada para cada Classe de treinamento criado pelos usuários. Não possuindo nenhum padrão, a forma de visualização dos dados está livre para ser definida pelo usuário do framework proposto neste artigo, são arquivos que vão consumir resultados vindos do WS.

Resultado da estruturação da aplicação mostrada na Figura 2:

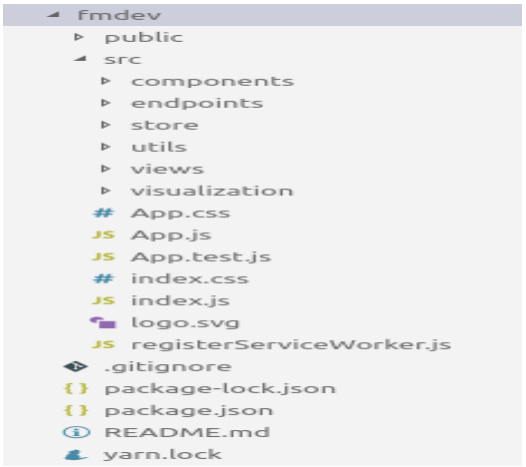


Figura 2 - Estrutura da aplicação React.
Fonte: Autor (2018).

3.2.2 *Web Service Flask*

A aplicação de *Web Service* está estruturada como mostra a Figura 3:

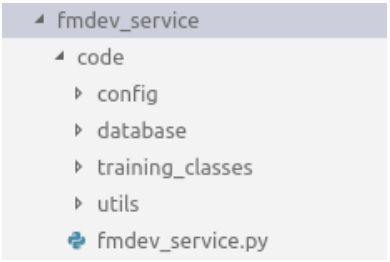


Figura 3 - Estrutura do *Web Service*.
Fonte: Autor (2018).

O arquivo **fmdev_service.py** é onde está localizado o código de inicialização do serviço, assim como as definições dos recursos e rotas oferecidos pelo mesmo.

A pasta **config** possui um arquivo de configuração onde são especificadas informações necessárias para abrir uma conexão com o banco, como por exemplo: tipo da base, tendo duas possibilidades, PostgreSQL ou MySQL, nome da base, usuário, senha, *host* e porta onde o serviço está disponível.

A pasta ***database*** possui dois arquivos *python*, um para PostgreSQL e outro para MySQL, classes estas responsáveis por iniciar uma conexão com a base de dados configurada e realizar consultas definidas por seus métodos, cada um com uma finalidade específica.

O ***training_classes*** é onde é criada arquivos com classes que são implementadas de acordo com a classe abstrata, localizada no mesmo diretório, chamada **ModelClass.py** que obriga o usuário do *framework* a fazer implementação de dois métodos, uma de tratamento de dados e outra de treinamento.

3.3 Utilização do *Framework*

Para adicionar uma nova classe de treinamento, o desenvolvedor deve levar os seguintes passos em consideração:

1. Criar uma Classe python na pasta **training_classes** no **fmdev_service** estendendo a classe abstrata **ModelClass** e

http://dx.doi.org/10.25286/repa.v3i3.976

implementando os dois métodos exigidos por ela que são: **tratar_dados** e **treinar**.

O primeiro método deve receber como parâmetro as seguintes informações:

- os dados em variáveis separadas contendo atributos e com os dados da coluna a ser classificada pelo classificador implementado.
- variável contendo o JSON com as opções de configuração recebida dos *inputs* de formulários do *front-end*, também contendo dados de configuração. vindas da aplicação que está utilizando o WS.

No segundo método é importante definir o formato de retorno dos resultados obtidos no treinamento que será usado no componente de visualização pela aplicação que está utilizando o WS. A Figura 4 ilustra a criação descrita anteriormente:

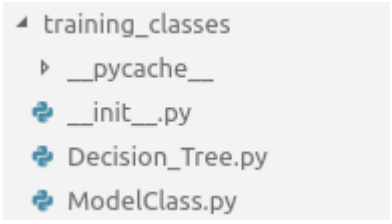


Figura 4 - Passo 1: criação do arquivo (exemplo usando árvore de decisão).
Fonte: Autor.

```
1  from abc import ABCMeta, abstractmethod
2
3  # Classe Modelo para ser seguido por outras classes
4  # Implementação Obigratória de alguns métodos da Classe
5
6  class ModelClass(object):
7
8      __metaclass__ = ABCMeta
9
10     def __init__(self, processing_op):
11         self.processing_op = processing_op
12
13     @abstractmethod
14     def tratar_dados(self): pass
15
16     @abstractmethod
17     def treinar(self): pass
```

Figura 5: Passo 1 - Classe Modelo que deve ser implementada.
Fonte: Autor.

```
13  def tratar_dados(self, X, Y, config):
14      # Conversão Numérica Categórica
15      Y['nota'] = np.where(Y['nota']>=50, 'aprovado', 'reprovado')
16
17      # Particionamento de Dados
18      part = config['part']['teste']
19      part = round(part/100, 2)
20      X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=part )
21
22      # Atribuição do resultado do tratamento como Atributos da Classe
23      self.X_train = X_train
24      self.X_test = X_test
25      self.Y_train = Y_train
26      self.Y_test = Y_test
```

Figura 6 - Passo 1: Exemplo de implementação de treinamento (Árvore de Decisão).
Fonte: Autor.

2. Após a implementação da nova Classe de treinamento, precisamos adicionar o código de instanciação em uma área específica no arquivo fmdev_service.py. Local onde existe um trecho de código condicional, que verifica o campo com o nome da classe que a aplicação consumidora deseja executar na variável contendo informações de configuração. Caso esse arquivo exista na para training_classes, o código cria uma nova instância atribuindo a variável training_class.

```
177     # Área de inserção de Classes Implementadas
178     # Inserir condição aqui para instanciar a classe solicitada
179     if (nome_arquivo == "Decision_Tree"):
180         training_class = Decision_Tree()
181
182     # Executa o tratamento dos dados implementados na Classe Selecionada
183     if (training_class != None):
184         training_class.tratar_dados(X, Y, config)
185         resultado_treino = training_class.treinar()
186         return {'result': resultado_treino}
187     else:
188         return {'result': "Class does not exist !"}
```

Figura 6 - Passo 2 :Inserir código de instanciação na condicional.
Fonte: Autor.

Os passos descritos anteriormente são suficientes para que outras aplicações externas comecem a utilizar o recurso criado pelo usuário do *framework*.

Os passos descritos logo em seguida são referentes aos passos que devem ser seguidas, especificamente, na aplicação que está utilizando o WS, que são específicas a aplicação React que foi criada para demonstrar o consumo do WS.

Esses passos podem variar dependendo de como o desenvolvedor da aplicação deseja consumir os resultados dos classificadores.

Criar novo componente de visualização para consumir os resultados do treinamento obtido. Para isso basta criar um novo componente dentro da pasta **visualization** na aplicação React. A estrutura interna do componente está livre para ser definida pelo usuário do *framework*, ou seja, pode adotar qualquer tipo de bibliotecas de visualizações de dados a ser exibido pelo componente criado.

Esta etapa é opcional, caso o usuário deseje acrescentar um novo tipo de tratamento de dados e configuração, é necessário criar um componente de formulários para o usuário inserir que tipo de tratamento deseja usar nos dados que está enviando para o WS. E implementar códigos no WS, que lide com a configuração desejada.

4 Testes e Resultados

O teste da ferramenta foi realizado em cima da base do Moodle do NEAD (Núcleo em Educação a Distância) da Universidade de Pernambuco, que foi disponibilizada por Maciel **[1]**. Os atributos que foram usados para os testes foram criados por Santana **[7]**, que consiste em consultas SQL que extraem as seguintes informações do banco de dados do Moodle do NEAD:

- Número de Postagens Realizados do Aluno.
- Número de Postagens de outros Alunos lidas pelo Aluno.
- Número de logins do Aluno.
- Número de Revisão de Postagens Anteriores.

Com base nestes atributos tentamos classificar se o aluno foi aprovado ou reprovado utilizando o algoritmo de árvore de decisão. O método que retorna o resultado foi implementado de maneira que ela apenas retorne como a estrutura da árvore. Não foi realizada nenhuma validação do resultado retornado pelo motivo de não estar no escopo do projeto, ficando apenas focado na construção das aplicações. Os resultados das construções são descritos detalhadamente no tópico seguinte.

A aplicação React trata-se de uma ferramenta onde o usuário pode realizar processos relacionados à mineração de dados sequencialmente: seleção de atributos, pré-processamento, treinamento, resultado. As imagens referentes às telas encontram-se anexo a este artigo e são descritas logo abaixo:

Tela de Autenticação: Onde o usuário precisa informar uma credencial válida cadastrada no Moodle. Após a autenticação o botão do próximo é habilitado para ir a próxima página. (anexo - tela 01).

Tela de seleção dos atributos: Nesta tela, o usuário pode selecionar atributos para passar pelo pré-processamento e posteriormente para a fase de treinamento. (anexo - tela 02).

Tela de edição, criação e pré-visualização: criar, editar consultas SQL para extrair os atributos do banco de dados. É, também, o local onde o pode executar SQL e pré-visualizar o resultado. (Anexo - tela 03)

Tela de Seleção e tratamento de Dados: Nesta etapa são listadas as colunas resultantes das consultas selecionadas na etapa anterior. É onde o usuário pode verificar o tipo dos dados, selecionar colunas que vão servir como entrada para o algoritmo de classificação, a coluna Classe, a coluna ID que vai servir para realizar o INNER JOIN das consultas selecionadas a partir da consulta com a coluna ID. É importante que na etapa anterior, as consultas possuírem colunas em comum para realizar esse processo, caso contrário serão desconsideradas para o envio na próxima etapa. (anexo - tela 04)

Tela de seleção de algoritmos de classificação: Tela de escolha do algoritmo a ser executado. As opções são disponibilizadas conforme o desenvolvedor adicione classes na pasta *training classes*. (anexo - tela 05)

Tela de Resultados: Após a escolha do algoritmo e sua execução, o resultado é apresentado conforme definido no componente de visualização criada pelo desenvolvedor. (anexo - tela 06).

5 Conclusões e trabalhos futuros

O *framework* proposto apresentou ser funcional e permitiu que novos classificadores sejam adicionados com poucos passos de

http://dx.doi.org/10.25286/repa.v3i3.976

codificação. Embora exista vários pontos que ainda devem ser analisados:

- Conexões diretas com a base do Moodle não é recomendada segundo a própria Moodle. Existem meios mais seguros para tal finalidade que consiste em realizar tais operações de consulta de informações através da API do próprio Moodle.
- Realizar experimentos mais elaborados com bases mais populosas. A base do NEAD que foi disponibilizada apresentou poucas informações referentes aos atributos selecionados para teste, apresentando 1500 alunos. No entanto, desses 1500 alunos, somente 97 apresentou ter tais atributos, e entre os 97 somente 8 apresentaram ser aprovadas, deixando o algoritmo com *Overfitting* de alunos aprovados.
- Mapear parâmetros dos classificadores. Existem diversos classificadores na biblioteca utilizada (*Scikit-learn*), cada um com suas propriedades e parâmetros específicos. Para melhor desempenho dos algoritmos classificadores, existe a necessidade de considerar tais características como parâmetro de configuração para serem processadas pelo *framework* criado.

6. Referências

[1] MACHADO, Liliana Dias; MACHADO, Elian de Castro. O papel da tutoria em ambientes de EAD. In: CONGRESSO INTERNACIONAL DE EDUCAÇÃO A DISTÂNCIA , 11., 2004, Salvador. **Anais**... Salvador: ABED, 2014. p. 1-3.

[2] LOPES, Maria Cristina et al. O processo histórico da educação à distância e suas implicações: desafios e possibilidades. IN: JORNADA DO HISTEDBR, 7., 2007, Campo Grande. **Anais**... Campo Grande, MS: UCDB. 2007.

[3] NUNES, E. R.; AQUINO, G. A. M.; FURTADO, A. M. A importância dos ambientes Virtuais de Aprendizagem na Busca de novos domínios do EAD. In: CONGRESSO INTERNACIONAL DE EDUCAÇÃO A DISTÂNCIA , 13., 2004, Salvador. **Anais**... Curitiba: ABED, 2007. p. 1-3.

[4] FRANCISCATO, Fábio Teixeira et al. Avaliação dos Ambientes Virtuais de Aprendizagem Moodle, TelEduc e Tidia-ae: um estudo comparativo. **Novas Tecnologias na Educação (RENOTE)**, v. 6, n. 2, 2008.

[5] HERMIDA, Jorge Fernando; BONFIM, Claudia Ramos de Souza. A educação à distância: história, concepções e perspectivas. **Revista HISTEDBR On-line**, Campinas, n. especial, p. 166-181, 2006.

[6] COSTA, Evandro et al. Mineração de dados educacionais: conceitos, técnicas, ferramentas e aplicações. In: Jornada de Atualização em Informática na Educação, 2., 2013, Campinas. **Anais**... Campinas:Sociedade Brasileira de Computação, 2013.

[7] SANTANA, L. C. D. Integração de um Mecanismo de Mineração de Dados educacionais ao Moodle. 2015.

[8] SOMMERVILLE, Ian. **Engenharia de software.** 6 ed. São Paulo: Addison Wesley, 2003.

[9] MACIEL, A. M. A.; RODRIGUES, R. L.; CARVALHO, E. C. B. Desenvolvimento de um Assistente Virtual Integrado ao Moodle para Suporte à Aprendizagem Online. In: Simpósio Brasileiro de Educação a Distância, 2., 2014, São Carlos. **Anais**... São Carlos, SP: SEAD, 2014.

[10] GONÇALVES A. F. D. Desenvolvimento de uma arquitetura integrada para a visualização de dados em ambientes de ensino a distância. 2017.

[11] JOHNSON, Ralph E. Components, frameworks, patterns. In: GEORGANS, Jhoan. **ACM SIGSOFT Software Engineering Notes**. v. 43, n.2. New York: ACM, 1997. p. 10-17.

[12] FAYAD M., SCHMIDT, D. JOHNSON, R. **Building Application Frameworks: Object-Oriented Foundations of Framework Design**. John Wiley & Sons, 1999.

[13] MATTSSON, Micheal. **Object-oriented Frameworks**: A survey of methodological issues. Licentiate thesis. Department of Computer Science, Lund University. Swend: Lund UNiverty, 1996.

[14] OPENSOFT. **Web Service**: o que é, como funciona e para que serve, 2016. Disponível em: <https://www.opensoft.pt/web-service/> Acesso em: 12 jul. 2018.

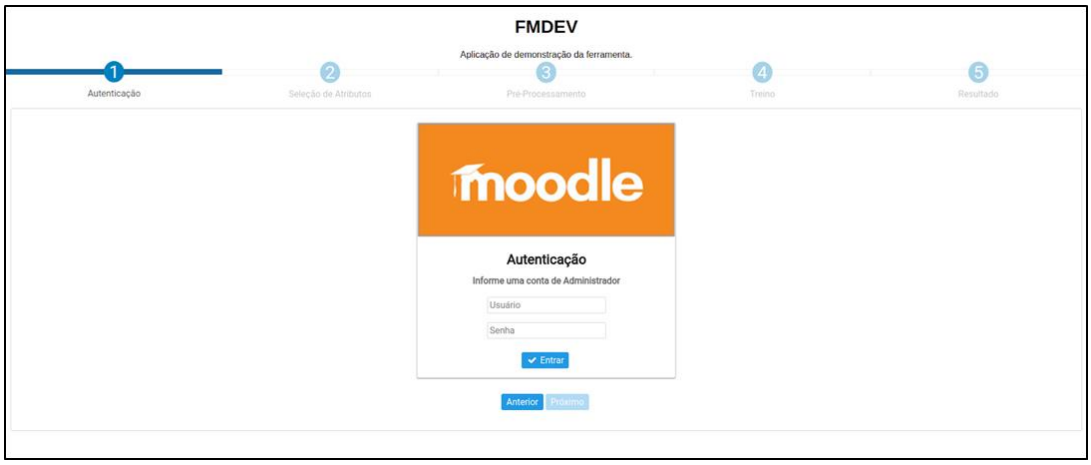
[15] MOODLE, **Moodle Statistics**, 2018, Disponível em: <https://moodle.net/stats/>. Acesso em: 12 jul.2018.

[16] CANTO, Carlos Eurico. As linguagens populares na ciência dos dados. **Propus Data Science**, 9 maio 2017. Disponível em: :<http://propus.science/as-linguagens-mais-populares-em-ciencia-de-dados/>. Acesso em: 08 ago. 2018.

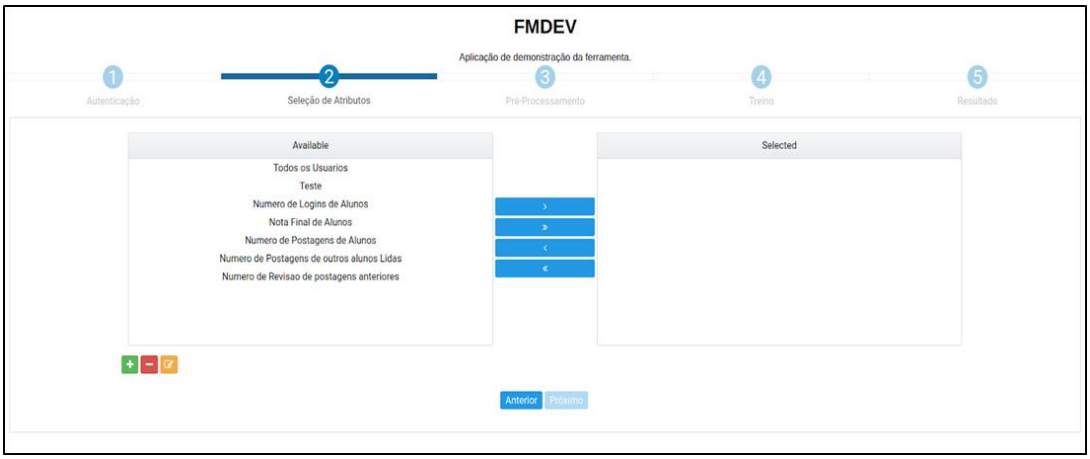
[17] FATIMA, H. Flask x Django: como escolher o framework concreto para seu aplicativo Web. **iMasters**, 10 abr. 2018. Disponível em: <https://imasters.com.br/back-end/flask-x-django-como-escolher-o-framework-correto-para-seu-aplicativo-web>. Acesso em: 08 ago. 2018.

http://dx.doi.org/10.25286/repa.v3i3.976

Anexos

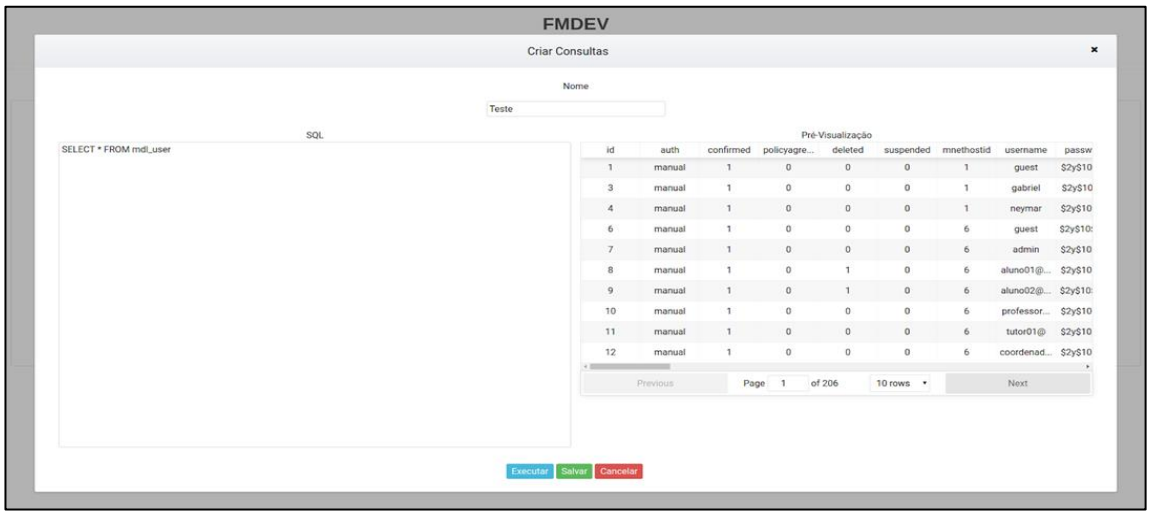


Tela 1: Tela de autenticação. Fonte: Autor.



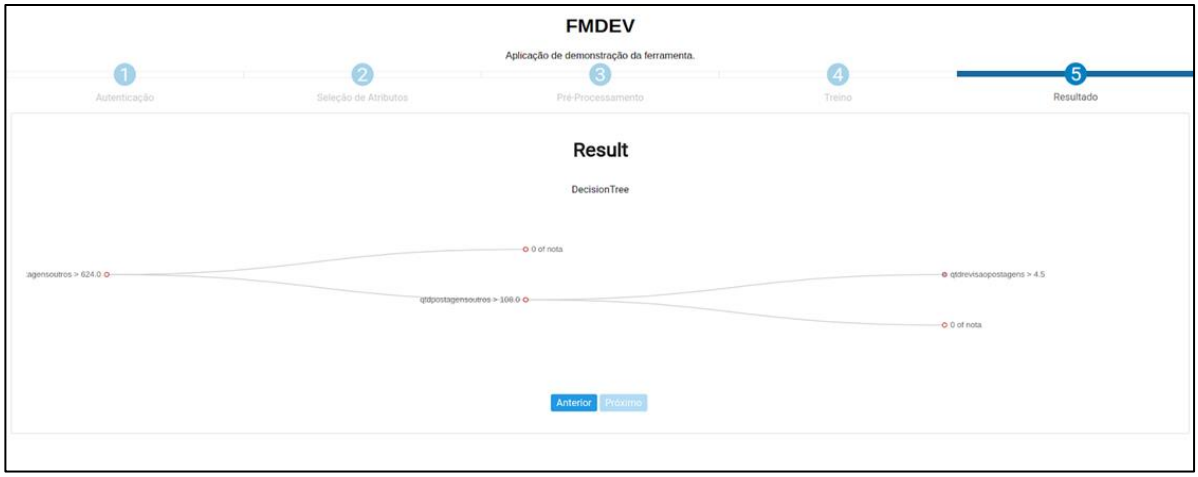
Tela 2: Tela de seleção de atributos. Fonte: Autor.

Tela 3: Tela de criação de atributos. Fonte: Autor.



Tela 4: Tela de pré-processamento. Fonte: Autor.

Tela 5: Tela de seleção de classificadores Fonte: Autor.



Tela 6: Tela de resultados Fonte: Autor.

http://dx.doi.org/10.25286/repa.v3i3.976