INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

Big Data Analytics para a classificação do risco de abandono escolar em cursos do ensino superior

Tiago Sanches Franco

Dissertação apresentada à Escola Superior de Tecnologia e Gestão para obtenção do Grau de Mestre em Sistemas de Informação

Trabalho realizado sob a orientação de:

Professor Doutor Paulo Alexandre Vara Alves

Professor Doutor André Koscianski

Bragança

Março de 2019

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Ponta Grossa

Diretoria de Graduação e Educação Profissional

TERMO DE APROVAÇÃO

Big Data Analytics para a classificação do risco de abandono escolar em cursos do ensino superior

por

Tiago Sanches Franco

Este Trabalho de Conclusão de Curso (TCC) ou esta Monografia ou esta Dissertação foi apresentado(a) em 27 de março de 2019 como requisito parcial para a obtenção do título de Bacharel em Ciência Da Computação. O(a) candidato(a) foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Doutor Paulo Alexandre Vara Alves
Prof. Orientador IPB

_____
Doutora Maria João Tinoco Varanda Pereira
Membro titular

_____
Doutor André Koscianski
Prof. Orientador UTFPR

_____
Doutor José Eduardo Moreira Fernandes
Membro titular

_____
Mestre Geraldo Ranthum
Responsável pelos Trabalhos
de Conclusão de Curso

_____
Mestre Saulo Jorge Beltrão de Queiroz
Coordenador do Curso
UTFPR - Campus Ponta Grossa

Agradecimentos

Primeiramente gostaria de agradecer aos meus pais e à minha irmã, pelas palavras de apoio e incentivo não só na trajetória deste trabalho, mas em todas etapas da minha vida, em especial por serem tão carinhosos e atenciosos mesmo distantes.

Aos meus orientadores, Professor Doutor Paulo Alexandre Vara Alves e Professor Doutor André Koscianski, por me auxiliarem com sugestões, comentários e conselhos que foram cruciais desde o começo até o final deste trabalho. Também os agradeço pela paciência e disposição que sempre demonstraram quando necessário.

À Universidade Tecnológica Federal do Paraná e ao Instituto Politécnico de Bragança, que viabilizaram e tornaram possível a realização deste mestrado.

Um segundo agradecimento especial ao Instituto Politécnico de Bragança, por fornecerem os dados que foram utilizados neste estudo e as máquinas virtuais para teste que proporcionaram um desenvolvimento mais rápido e melhorado.

Resumo

O abandono escolar é uma constante preocupação das instituições de ensino superior, com divergentes e complexos fatores relatados por diversos autores. Conseguimos notar que este problema é bastante abrangente e explorado, já sendo possível encontrar nas instituições setores especializados ou programas de auxílio como psicólogos, auxílio moradia, programa de monitoria, entre outros, buscando minimizar a quantidade de alunos desistentes. Entretanto estas propostas dependem do próprio aluno a buscar ajuda necessária, abrindo uma lacuna para aqueles que não se sentem confortáveis a procurar ou não possuem total conhecimento do próprio caso.

Este trabalho propõe um modelo para a identificação prévia dos alunos desistentes, com objetivo de tornar as instituições de ensino aptas a entender melhor os casos de abandono e se possível encaminhá-los a setores especializados. Para tal, utilizamos o Instituto Politécnico de Bragança como estudo de caso que nos forneceu mais 200 milhões de registros relacionados aos alunos matriculados entre 2008 a 2017.

Analisamos e processamos a Big Data fornecida com a finalidade de moldá-la como parâmetros de entrada de algoritmos de *machine learning*. Inicialmente testamos três algoritmos e descobrimos que o *random forest* demonstra ser o mais eficiente neste contexto. A partir disso, aproveitamos do volume de dados para identificar qual seria melhor ciclo de treino e obtemos que o período de 4 anos consegue atingir melhores resultados. No aprimoramento do modelo adicionamos mais 2 atributos buscando realçar a trajetória escolar do aluno.

Para implementação e visualização do modelo, desenvolvemos uma ferramenta de extração de dados e uma aplicação Web, que através de diferentes níveis de acesso, além de conseguir identificar os alunos em risco de abandono, também possibilita aos usuários efetuar análises comparativas entre escolas e cursos por meio de uma página personalizada com estatísticas transformadas em gráficos e tabelas.

O estudo se apresenta como uma boa solução para identificação prévia dos alunos em risco de abandono, possibilitando análises e encaminhamentos. O modelo ainda pode ser ampliado a mais parâmetros e tende a obter melhores resultados ao longo dos anos aperfeiçoando através do reforço os atributos criados.

Abstract

School dropout is a constant concern of higher education institutions, with divergent and complex factors reported by various authors. We can note that this problem is quite broad and exploited, since it is possible to find specialized sectors or assistance programs such as psychologists, housing assistance, monitoring programs, among others, in order to minimize the number of students dropping out. However, these proposals depend on the student himself to seek necessary help, opening a gap for those who do not feel comfortable to seek or do not have full knowledge of the case itself.

This work proposes a model for the prior identification of dropouts, with the aim of making higher education institutions able to understand cases of dropout and, if possible, refer them to specialized sectors. For that, we use as a case study the Instituto Politécnico de Bragança, which provided 200 million records related to students enrolled between 2008 and 2017.

We analyze and process the Big Data provided for molding it as input parameters of machine learning algorithms. We first tested three algorithms and found that random forest proves to be the most efficient in this context. From this, we take advantage of the data volume to identify which would be the best training cycle and obtain that the period of 4 years can achieve better results. In the improvement of the model we added two more attributes to highlight the trajectory of the student.

For the implementation and visualization of the model, we developed a data extraction tool and a Web application, which through different levels of access, besides being able to identify students at risk of abandonment, also allows users to make comparative analysis between schools and courses by of a custom page with statistics transformed into graphs and tables.

The study presents as a good solution for the prior identification of students at risk of dropout, enabling analysis and referrals. The model can still be extended to more parameters and tends to obtain better results over the years by improving by enhancing the attributes created.

Índice Geral

Lista de Siglas/Abreviaturas

CRUD - Create, Read, Update, Delete

CART - Classification And Regression Trees

EDM - Education Data Mining

ETL - Extract, Transform and Load

IHC - Interação Humano Computador

IPB - Instituto Politécnico de Bragança

KDD - Knowledge Discovery in Databases

LMS - Learning Management System

MVC - Model-View-Controller

NoSQL - Not Only SQL

ROC - Receiver Operating Characteristic

WSGI - Web Server Gateway Interface

Índice de Figuras

Índice de Tabelas

Capítulo 1 Introdução

Este capítulo apresenta de forma sucinta as informações introdutórias a esta pesquisa, iniciando com a descrição do problema abordado, lacunas que objetificam este estudo e os resultados esperados pelo mesmo.

1.1. Identificação do Problema

No contexto escolar, uma das principais métricas para medir o desempenho de uma instituição é a taxa de desistência dos alunos, também conhecida na literatura pelo seu termo em inglês *dropout*. A diminuição do número de discentes em uma universidade implica na redução dos principais recursos necessários para o funcionamento da mesma, como a estagnação de investimentos em infraestrutura, quantidade de professores, melhores tecnologias e equipamentos, entre outros meios que podem aumentar a qualidade do ensino.

Entretanto, diminuir a taxa de desistência dos alunos não é uma tarefa fácil, para resolver este problema as universidades buscam concentrar uma parte de seus recursos em setores especializados ou programas de auxílio, como psicólogos, auxílio moradia, auxílio alimentação, programa de monitoria, entre outros. Porém apesar de eficazes em suas propostas, estes serviços não são suficientes para rever todos os casos. Exceto quando há fatores externos à instituição, o problema se dá por depender do próprio aluno a buscar ajuda necessária, sendo que nem sempre está confortável a procurar ou não entende que precisa dela.

A partir desta lacuna, cria-se a necessidade de identificar individualmente os alunos que irão desistir antes que tal fato aconteça, possibilitando às universidades direcionar

recursos de maneira otimizada, encaminhando-os aos setores especializados a tempo de contornar a situação.

Dada a complexidade do problema, pela diversidade de fatores envolvidos, é fundamental que as instituições analisem a maior quantidade de parâmetros possíveis, buscando abranger todos os casos. Para isso, porém, é preciso que os dados sejam colhidos de forma a garantir a confiabilidade e evitar informações incorretas que afetariam os resultados de análise.

A princípio as universidades já produzem um certo volume de dados que retrata a vida dos acadêmicos; contudo, esses dados costumam ser divididos em diversas bases e nem sempre estão organizados de forma fácil para análise. Um exemplo disso são os registros acadêmicos de notas, normalmente parte de um sistema de secretaria e administração; e os sistemas de aprendizagem online, tipicamente implementados em plataformas como Moodle e Sakai. Além dos dados estarem em plataformas distintas, no caso de registros de atividades online há uma variedade de informações diferentes, como logins, mensagens enviadas, visualizações de recursos, que podem ser cruciais na identificação dos potenciais desistentes. Isso necessita um processamento inicial para extrair as informações relevantes.

1.2. Objetivos

Neste contexto, o presente estudo propõe uma solução para a identificação prévia dos alunos desistentes, extraindo semanalmente os dados gerados pela universidade e aplicando técnicas de *machine learning* com o objetivo de produzir uma classificação destes alunos. Esta classificação pretende prever a situação do aluno entre desistente ou não desistente baseando-se na coleta dos anos anteriores, com objetivo de tornar a instituição apta a tomar medidas preventivas atempadamente.

Como estudo de caso deste trabalho, foi fornecido pelo Instituto Politécnico de Bragança (IPB) dados de 3 conjuntos distintos, o primeiro referente às informações básicas dos alunos em relação a instituição como notas, quantidade de matérias aprovadas e reprovadas, quantidade de anos letivos e etc., o segundo referente às presenças em salas de aula e o último são os registros gerados pelo Sakai, sistema de aprendizagem online

utilizado pelo instituto. Os dados fornecidos são referentes aos anos de 2009 até 2017, que combinados ultrapassam 200 milhões de registros e ocupam aproximadamente 50 gigabytes de disco. Este trabalho propõe-se a apresentar uma solução para os seguintes objetivos específicos:

- Avaliar o desempenho dos algoritmos selecionados de machine learning em relação à adequação ao problema proposto;
- Refinar o modelo proposto para o algoritmo encontrado;
- Construção de uma ferramenta de extração de dados (ETL) capaz de tratar e transportar os dados semanalmente para aplicação Web;
- Construção de uma interface web para executar o modelo proposto e gerar relatórios com diferentes níveis de acesso;

A solução proposta também pretende apresentar relatórios com gráficos e tabelas que buscam traduzir de forma simples a situação atual da universidade em relação aos desistentes, como comparações entre escolas e cursos, evolução do número de desistências, classificação de risco, entre outras informações relevantes que podem auxiliar a instituição a entender as causas dos desistentes.

O presente documento encontra-se dividido em mais quatro capítulos que darão continuidade a explicação do estudo. O capítulo 2 é referente a revisão bibliográfica que viabilizou este estudo, abordando os principais conceitos, técnicas e trabalhos relacionados; o capítulo 3 retrata a trajetória de desenvolvimento deste projeto, incluindo o detalhamento da criação das ferramentas e as soluções encontradas para suas dificuldades; no capítulo 4 é apresentado e analisados os resultados obtidos pelo estudo; no último capítulo (capítulo 5) relata as principais conclusões e os trabalhos futuros.

Capítulo 2 Revisão da Literatura

Esta seção apresentará os conceitos necessários para o entendimento do estudo e termos comuns nas áreas de aprendizado de máquina, mineração de dados educacionais e na literatura sobre o abandono escolar que são mencionados ao longo do trabalho.

2.1. Big Data

Com sua origem incerta, o termo *Big Data* refere-se a um grande conjunto de informações, que diferente dos dados tradicionais, não se encontra necessariamente estruturado ou mesmo semiestruturado [1]. Apesar de possuir relatos na década de 90, o termo *Big Data* só foi amplamente difundido no ano de 2011 em diante, graças a grandes investimentos em tecnologias para análise de mercado de empresas mundialmente conhecidas como a IBM [2].

Existem três características que marcam a evolução atual dos dados e definem as dimensões do *Big Data*, são chamados “Os três Vs” [3] :

Volume: é referente a enorme quantidade de informação que conseguimos salvar. Hoje em dia multiplicamos nossa capacidade de armazenando diariamente, segundo Paul Zikopoulos e Chris Eaton [4] 800.000 *petabytes* seria o suficiente para armazenar toda informação que a humanidade havia produzido até o ano 2000, é esperado que esse número seja em torno de 35 *zettabytes* em 2020, ou seja, estamos prestes a produzir e armazenar 40 vezes mais informações em 20 anos do que a humanidade produziu em toda sua existência.

Variedade: são as diferentes estruturas que pertencem ao mesmo conjunto. Podemos definir a complexidade de trabalhar com um conjunto pelo seu nível de heterogeneidade: quanto maior, mais processos serão exigidos para tratamento antes da análise. Em relação a essa característica, os dados são divididas em três categorias [2]:

- Estruturado: dados tabulados em planilhas ou encontrados em banco de dados relacionais;
- Semiestruturado: são dados que não possuem como formato um padrão rígido, os exemplos mais comuns são os formatos XML e JSON;
- Não estruturado: imagens, áudios, vídeos e textos são exemplos de dados não estruturados, normalmente os mais difíceis de analisar;

Velocidade: é referente ao volume por unidade de tempo em que os dados são gerados e analisados. Com os avanços tecnológicos dos últimos anos, os smartphones e outros aparelhos digitais se tornaram mais atrativos e foram amplamente difundidos em todo o mundo, aumentando exponencialmente a criação de dados. Segundo estimativas de 2011, a rede social Facebook, unicamente, produziria 10 *terabytes* por dia e a rede social Twiiter 7 *terabytes* [4], grandes empresas de varejo como Wallmart produziram 2.5 *petabytes* por hora [1]. Todos os dados gerados por essas companhias são de extrema importância para análise de seus clientes, sendo possível extrair informações relevantes que mudam o curso econômico dessas empresas. Em razão disto, existe a grande preocupação pela velocidade da análise [2].

É possível achar na literatura mais recente, autores que defendem a adição de mais duas características para definir o termo *Big Data*, sendo elas: **Veracidade** referindo-se à confiabilidade dos dados e **Valor**, que seria a capacidade em transformar o conjunto dos dados em informações relevantes à pesquisa em questão. A junção dessas características foi nomeada como “Os cinco Vs” [3].

Não existe uma regra rígida ou valores específicos sobre essas características, devido à velocidade que as medidas mudam, a dimensão de “Grande” no termo *Big Data* se torna relativo ao tempo em que estamos vivendo [4], sendo assim, a classificação de um conjunto como *Big Data* tende a ser conceitual.

2.2. Aprendizado de Máquina

Também conhecido como *Machine Learning*, Aprendizado de máquina é um subcampo da Inteligência Artificial que concentra seus estudos em desenvolver e melhorar algoritmos capazes de reconhecer padrões e aprender de forma autônoma [5] [6]. A aprendizagem neste contexto não é uma atividade necessariamente provida da consciência humana, mas sim a habilidade de encontrar regularidades estatísticas ou padrões nos dados; sendo assim, aprendizado de máquina é sobre projetar algoritmos que possuem tal habilidade [7].

Podemos dividir os algoritmos de aprendizado de máquina em 6 áreas principais, definidas pela forma como "aprendem" e os resultados que se deseja obter [7]. Essas seis áreas são:

- Supervisionado: algoritmos que conseguem identificar padrões em um conjunto com entradas e saídas definidas; estas técnicas são normalmente utilizadas para classificação e regressão;
- Não supervisionado: algoritmos que trabalham com dados não rotulados, buscando encontrar parâmetros comuns e agrupá-los de modo a encontrar estruturas que não eram conhecidas;
- Semi-supervisionado: algoritmos que combinam dados rotulados e não rotulados no conjunto de treino com objetivo de gerar funções ou classificadores;
- Reforço: algoritmos que buscam desenvolver uma política de ações com maior ganho em um dado ambiente; para isso, utilizam as entradas fornecidas para explorar o ambiente e analisam os impactos nos possíveis caminhos para se orientar;
- Transdução: algoritmos que podem receber dados rotulados ou não rotulado, assim como os de aprendizagem semi-supervisionados, porém não produzem uma função explícita; ao invés disto, buscam prever valores com base nas entradas de treinamento;
- *Learning to learn*: algoritmos que utilizam em seu método de aprendizagem experiências anteriores para induzir valores;

As bases de dados utilizadas neste projeto não contêm dados não-estruturados (essa informação é mais detalhada na seção 3.2). Considerando que o objetivo central do trabalho é construir um modelo de classificação para identificação dos alunos com risco de abandono possuindo duas classes (desistentes e não desistente), a área que mais se encaixa é a de algoritmos supervisionados. Sendo assim como explicado na seção 2.4.1 deste documento, foram selecionados três algoritmos para comparação na montagem do modelo, descritos a seguir.

2.2.1 Árvore de Decisão

Pertencendo aos algoritmos de aprendizagem supervisionada, as árvores de decisão são modelos estatísticos que representam um conjunto de regras que seguem uma hierarquia de classes e valores, com finalidade de produzir uma função de classificação ou regressão em um determinado conjunto de dados [8].

O conjunto hierárquico produzido possui uma estrutura em árvore, semelhante a um fluxograma. Cada nó interno (não folha) representa um atribuído a ser testado e o resultado do teste são as possíveis ramificações a novos nós. Assim, o nó mais alto, denominado como raiz, dá início ao fluxograma sendo o primeiro a ser testado. Já cada um dos nós-folhas (nó-terminal) representa uma das classes conhecidas como objetivo do problema [9]. Na Figura 1 pode ser visto um exemplo de árvore de decisão, com o objetivo de prever se um consumidor irá ou não comprar um computador figuradamente.

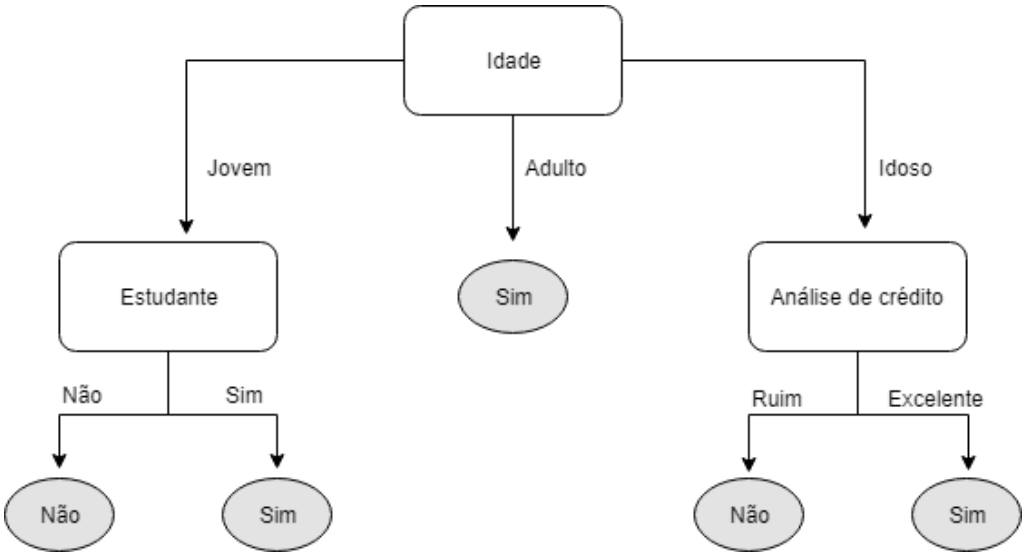

Figura 1: Exemplo de uma árvore de decisão (adaptado de Han et al. [10])

Como pode ser notado na Figura 1, os nós internos são denotados como retângulos e os nós-terminais são denotados como ovais. Nesta árvore há 3 atributos que influenciam na classificação: idade, estudante e análise de crédito; e duas possíveis classes como resposta: sim e não.

A divisão dos atributos é determinada a partir de uma heurística, que possui como objetivo selecionar o melhor critério de partição. O melhor critério para partição dos dados, é aquele que seleciona um atributo que após a divisão, obtém como resultado os dados de uma única classe em cada ramificação [10]. A partir disto, existem diversas heurísticas, cada uma possuindo características diferentes capazes de identificar um tipo padrão nos dados. As três heurísticas mais populares são: ganho de informação, taxa de ganho e índice de Gini.

Para a implementação das árvores de decisão existem diversos algoritmos. Os primeiros algoritmos de árvore de decisão foram desenvolvidos no final da década de 70 e início dos anos 80, popularizando-se na área de aprendizado de máquina e adquirindo novas versões e métricas que melhoraram seu desempenho [10]. Atualmente os algoritmos mais utilizados de árvores de decisão são ID3, C4.5 e CART.

Os algoritmos ID3 e C4.5 foram desenvolvidos pelo mesmo autor: Ross Quinlan, o primeiro a ser lançado foi o ID3 em 1986 [11], utilizando o ganho de informação como critério de partição, com ele era possível construir uma árvore de decisão através de um conjunto de dados categóricos, moldando sua estrutura de cima para baixo. O algoritmo C4.5 foi uma evolução do ID3, sendo lançado 6 anos mais tarde em 1993 [12], possuindo como principais diferenças a capacidade de manipular dados categóricos, ordinais e contínuos, lida com instâncias sem rótulo e apresenta uma função de poda moldando a estrutura de baixo para cima melhorando o desempenho.

O algoritmo CART foi apresentado pelos autores Breiman et al. em 1984 [13], método no qual consegue produzir árvores binárias (árvore que permite a cada cada nó interno possuir apenas duas ramificações), podendo manipular dados dados categóricos, ordinais e contínuos. O CART também apresenta uma função de poda, porém sua heurística de diverge do algoritmo C4.5, detectando quando não há mais ganho no ramo.

2.2.2 *Random Forest*

O algoritmo *Random Forest* é baseado no conceito de árvore de decisão explicado no subtópico anterior, porém ao invés de obter apenas uma árvore de decisão, o método produz múltiplas árvores. Este conjunto de árvores de decisão é denominado de 'floresta' e possui como objetivo melhorar a precisão. Para isto, o algoritmo combina o resultado das árvores através de votações, retornando um modelo de voto majoritário [8].

Como Han et at. [10] descreve, uma floresta por ser criada utilizando *bagging* em conjunto com a seleção de atributos aleatórios. Conceito introduzido por Breiman em 1996 [14], *bagging* é um método para gerar múltiplas versões de um preditor e os agrega no final, produzindo um preditor mais estável com precisão superior. O método pode ser aplicado da seguinte forma:

Considere-se D como o conjunto de treinamento, F como os atributos disponíveis em D, d e f uma instância de D e F, respectivamente, ambos sendo um subconjunto extraído de forma aleatória com substituição. Para cada iteração de D_i, sendo i $(i = 1, 2, \dots, k)$, deve-se construir uma árvore de decisão utilizando os atributos de F_i, selecionando-os aleatoriamente como candidatos a divisão do nó. Após cultivar as árvores até seu tamanho máximo a floresta está montada. Por fim, as árvores são combinadas utilizando uma votação, na qual é considerando o erro médio de todas as amostras, construindo um modelo mais estável a partir dos votos majoritários diminuindo o "sobre ajuste" (*overfitting*) [10].

2.2.3 Redes Neurais Artificiais

O campo de redes neurais foi originalmente estimulado por psicólogos e neurobiólogos que procuravam desenvolver e testar computação análoga a neurônios cerebrais [10]. Neste cenário, os pesquisadores não tinham como objetivo programar um computador digital para imitar um comportamento inteligente, como saber jogar xadrez, compreender e manter um diálogo, traduzir línguas estrangeiras, ou algo do gênero. O ideal teórico era construir um computador capaz de modelar os circuitos cerebrais e apresentar comportamentos inteligentes, aprendendo novas tarefas, errando, fazendo generalizações e descobertas [15].

Segundo Han et at. [10], podemos resumir uma rede neural como um conjunto de unidades de entradas e saídas conectadas com um peso associado em cada conexão. A partir disto, durante a fase de aprendizagem, a rede ajusta os pesos de cada ligação para fazer que a saída consiga prever os rótulos dos dados de entrada. Deixando de lado todo o detalhamento da parte matemática e computacional, resumidamente, cada unidade reconhece um padrão, e a combinação dessas unidades consegue identificar um conjunto de características que descrevem de forma genérica determinada classe. Entre as vantagens das redes neurais, uma delas é a capacidade de classificar padrões nos quais não foram treinados, aprendendo de forma autônoma através de suas generalizações.

Uma generalização do modelo de McCulloch e Pitts descrito por Barreto [15] sobre o modelo geral do neurônio, pode ser visto na Figura 2:

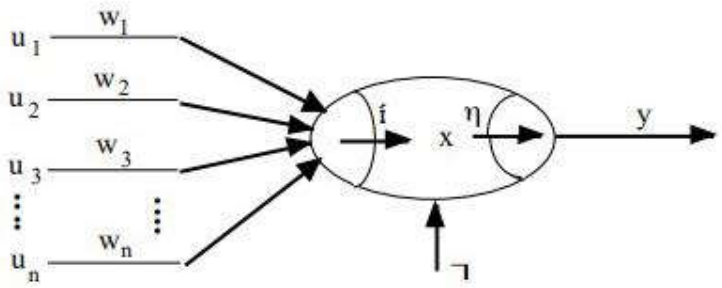

Figura 2: Neurônio artificial (reproduzido de Barreto [15])

Na Figura 2, u_n são as entradas e w_n seus respectivos pesos, que são combinados usando uma função Φ, para produzir um estado de ativação do neurônio que através da função η vai produzir a saída y do neurônio.

Durante um certo período, as redes neurais foram criticadas pelo seu difícil entendimento. Com a evolução do hardware houve um novo interesse e uma sequência em seu desenvolvimento, quando obtiveram sucesso no reconhecimento de padrões utilizando dados do mundo real, como o reconhecimento de caracteres manuscritos, patologia e medicina de laboratório, entre outros [10].

Redes neurais como a representada na Figura 2, são adequadas para entradas e saídas de valor contínuo [10], ao contrário da maioria dos algoritmos de árvore de decisão. Essa técnica é recomendada a aplicação que não necessita de respostas imediatas, pois geralmente demanda bastante tempo de treinamento.

2.2.4 Medidas de Avaliação

A área de aprendizado de máquina busca solucionar problemas de diversas naturezas, com múltiplos tipos de modelos e dados. Um elemento crítico desse cenário são os métodos de avaliação, que indicam a taxa de sucesso de uma dada solução.

No caso da classificação binária, os modelos utilizam um conjunto de dados rotulados em apenas duas classes, sendo possível interpretar como verdadeiro e falso. Uma forma de avaliação aplicável nesse caso é a análise de características de operação do receptor (ROC) da área de processamento de sinais [16]. Ela compara a taxa de resultados positivos verdadeiros com a taxa falsos positivos; as métricas mais conhecidas são acurácia, precisão e *recall* [16].

Acurácia é a porcentagem de registros classificados corretamente no conjunto de dados teste; precisão é a porcentagem de registros classificados como corretamente como verdadeiro sobre o conjunto de verdadeiros; *recall*, também conhecido como sensibilidade, é a porcentagem é de registros classificados corretamente como verdadeiro dentro do conjunto de classificados [17]. Tais medidas são calculadas a partir das seguintes fórmulas:

$$Acurácia = \frac{TP + TN}{P + N}$$

$$Precisão = \frac{TP}{P}$$

$$Recall = \frac{TP}{TP + FN}$$

, onde, P é o número de registros rotulados como verdadeiro; N é o número de registros rotulados como falsos; TP é o número de registros classificados corretamente como verdadeiro; TN é o número de registros classificados corretamente como falso; FN é o número de registros classificados incorretamente como falso.

2.3 Mineração de dados

Amplamente conhecido pelo seu termo em inglês *data mining*, mineração de dados é o processo de descobrimento de informação através dos padrões encontrados em um determinado conjunto de dados [18]. O processo de mineração é utilizado em grandes

volumes de dados, relacionando informações consistentes que tendem a ser padrões ou regras pertinentes do conjunto, entende-se que a mineração de dados não cria informações, apenas extrai conhecimentos que ali estavam ocultos [19].

Mineração de dados também pode ser conhecida pelo termo Extração de Dados, em inglês *Knowledge Discovery in Database* (KDD). A palavra minerar é mais utilizada para grande quantidade de dados, porém também se refere a extrair informações, devido a isto diversos autores utilizam o termo KDD como o processo ao todo e mineração como a técnica utilizada na etapa principal [20]. Segundo Sharma et at. [20] o processo de mineração de dados ou KDD possui 7 etapas sequenciais, como ilustra a Figura 3:

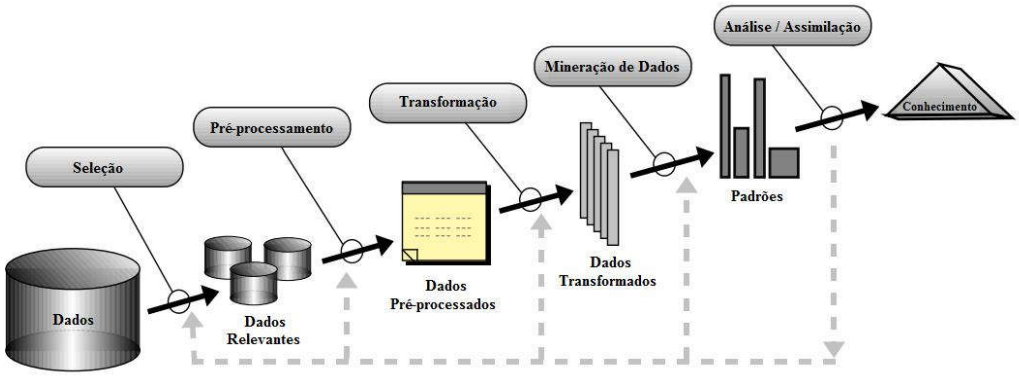

Figura 3: Etapas do processamento da mineração de dados (adaptado de Sharma et at. [20]).

1. Limpeza: Inicialmente é necessário remover os dados com ruído e dados que são irrelevantes para o estudo;
2. Integração: Reúnem-se dados de diferentes fontes em um único local;
3. Seleção: Nesta etapa, faz-se a triagem de dados relevantes para a tarefa atual;
4. Transformação: Convertem-se os dados em formato apropriado para a próxima etapa, geralmente aplicando normalizações e agregações;
5. Mineração: Nesta etapa são aplicadas diversas técnicas e ferramentas para extrair padrões e regras do conjunto;
6. Avaliação de padrões: Analisam-se padrões e regras que simbolizam informações relevantes;

7. Representação do conhecimento: Na última etapa, utilizam-se técnicas de visualização e representação de dados, com objetivo de ajudar aos usuários interpretarem as informações;

2.4 Mineração de Dados Educacionais

Segundo a *International Educational Data Mining Society* [21], podemos definir Mineração de Dados Educacionais (EDM) como uma área emergente que possui a finalidade de desenvolver e adaptar métodos para explorar exclusivamente os dados providos de ambientes educacionais, e utilizar estes métodos para entender melhor os alunos e as configurações em que eles aprendem.

A área de EDM teve seu início em workshops dentro de conferências sobre Inteligência Artificial Educacional, com foco em adaptar algoritmos já existentes de data mining sobre dados acadêmicos. Em 2008 foi criada a primeira conferência em EDM, evento que se estabeleceu e ganhou regularidade de realização e publicação de um periódico anual, ambos ativos até o presente momento (2019) [9].

Ao longo dos anos a área foi se expandindo, abrangendo e combinando mais campos, como modelagem computacional, estatística, pedagogia, IHC, simulação de dados, entre outros, não apenas *data mining*, alterando o foco da área de EDM, agora voltado ao estudo dos dados de ambiente educacionais em contexto geral [22]. Como exemplo, Shaun et at. [23] fez um estudo sobre as oportunidades na área de EDM a serem exploradas no Brasil retratando a carência dos estudos em comparando aos EUA, onde é possível encontrar diversas aplicações do EDM, até mesmo corporativas.

2.4.1 Métodos

Os métodos aplicados em EDM abrangem diversas áreas como aprendizado de máquina, psicométrica, estatística, visualização de informação e modelagem computacional [22].

A área de aprendizado de máquina, por exemplo, consegue, além de identificar padrões, aprender com os dados e produzir modelos mais completos, apresentando soluções a

problemas mais complexos, como a identificação de alunos em risco de desistência, retratado neste trabalho.

Existe assim um vasto conjunto de algoritmos que podem se encaixar no problema. Em razão disso foi considerado dentro dos objetivos do trabalho realizar em revisões bibliográficas específicas sobre o tema, quais algoritmos se apresentaram mais estáveis neste contexto e poderiam obter resultados melhores.

Segundo Shahiri et at. [24], o método mais utilizado para predição de performance dos alunos foi algoritmos de árvores de decisão, e em segundo lugar os algoritmos de redes neurais, porém com resultados ligeiramente melhores que os de árvore de decisão. Já Romero e Ventura [25] mostra os algoritmos mais utilizados em diferentes categorias de EDM, retratando que no caso de identificação do comportamento dos alunos, os algoritmos de árvore de decisão, *random forest, support vector machines*, rede neurais, foram os mais utilizados. Amal e Ahmad [26] realizaram um estudo em *Big Data* e *Learning Analytics* em escolas de ensino superior e recomenda o uso do algoritmo *random forest*, pela sua versatilidade e atingindo ótimos resultados.

Seguindo as indicações da literatura, os algoritmos selecionados foram árvore de decisão, *random forest* e rede neurais.

2.4.2 Sistemas de Gestão de Aprendizagem (LMS)

Sistemas informatizados como Moodle e Sakai começaram a surgir na década de 2000. Com a proliferação desse tipo de ferramenta surgiu a denominação de LMS, Learning Management Systems. Os requisitos se tornaram gradativamente mais complexos, podendo-se citar [27] [28]:

- Centralizar e automatizar a administração;
- Autoexplicativo e intuitivo;
- Montar e fornecer conteúdo de aprendizagem rapidamente;
- Facilitar a instalação em plataformas Web escaláveis;
- Suporte a portabilidade e padronizações;
- Personalizar o conteúdo e permitir a reutilização do conhecimento;

Os LMS têm um papel fundamental em EDM em virtude de poderem gerar uma grande quantidade de informações sobre o cotidiano dos estudantes. Isso inclui *logs* gerados a cada clique de um usuário, monitorando suas ações, os horários, entre outros aspectos que possibilitam análises mais finas sobre o comportamento, como exemplo, a rapidez que conseguimos tratar os dados, visto que os *logs* por serem são confiáveis, sem ruído e estruturados viabilizam a remoção de etapas de pré-processamento.

2.2.1.1. Sakai

Sakai é um conjunto de softwares de código aberto desenvolvidos pela comunidade e apresentado como uma solução LMS disponível gratuitamente no mercado, com diversos recursos para aprendizagem, ensino, pesquisa, e colaboração [29]. A primeira versão do software Sakai ficou disponível no ano de 2005. A partir disto, com o tempo foi se tornando mais flexível e enriqueceu-se de recursos alinhados às necessidades da comunidade, estando atualmente em mais de 350 instituições em todo o mundo [29], incluindo o Instituto Politécnico de Bragança sede deste trabalho.

Alguns dos recursos mais visíveis do software, incluem Interface do usuário responsiva, capaz de se adaptar em múltiplos ambientes, livro de notas, ferramentas para criação de lições, sistema de alerta com níveis de gravidade, avaliações aprimoradas, feedback inteligente e notações matemáticas. Além desses recursos importantes no aprendizado, a ferramenta também conta com mais de 250 tipos de *logs*, divididos por categorias que facilitam a extração e análise [29].

2.3. Abandono Escolar

Pesquisadores apontam que historicamente são requeridos níveis cada vez mais elevados de formação educacional [30]. O alto índice de profissionais qualificados tem se tornando um fator central para processos de criatividade, inovação e desenvolvimento social [31]. Reconhecendo a importância do tema, a União Europeia, em 2008, definiu como meta obter, pelo menos, 40% da população entre 30 a 34 anos ter uma formação superior de ensino ou equivalente até 2020, encorajando uma ampla camada da sociedade a ingressar no ensino superior e a reduzir as taxas de abandono nas universidades [32].

Uma dificuldade comumente reconhecida, apontada por Quinn [33] em 2013 é a falta de consistência no termo abandono escolar, possuindo múltiplas versões em diferentes países, tornando árduo o trabalho de comparação e análises entre países. Esta deficiência no termo é associada a diversidade da natureza dos diferentes sistemas de ensino superior entre os países e as diferentes naturezas das populações estudantis.

A evasão escolar é considerada de maneiras distintas por estudiosos do assunto; alguns exemplos são [34]: considerar evasão apenas quando de fato o estudante solicita o trancamento do curso; estar abaixo dos períodos médios para conclusão de curso; possuir mais faltas em aulas presenciais do que a instituição permite.

Para um aluno ser categorizado como desistente do Instituto Politécnico de Bragança, deve se encaixar simultaneamente em três situações:

1. não estar matriculado no ano letivo atual;
2. não constar na lista de formados;
3. estar matriculado no ano letivo anterior.

Dada a diversidade e irregularidade dos motivos de desistências dos alunos, não existe um consenso sobre quais fatores são mais significativos na desistência de um aluno, muitas vezes são combinações de fatores que levam a desistência, impossibilitando criar hierarquias absolutas sobre quais fatores são mais significativos [33]. Entretanto existem fatores que constantemente aparecem na literatura; entre eles:

1. Socioculturais: expectativas por parte das famílias, comunidades locais e funcionários da universidade que os estudantes não tradicionais não completam seus estudos;
2. Estruturais: o posicionamento desigual do aluno na sociedade por causa da pobreza, classe, raça, gênero provoca pressões que dificultam a persistência;
3. Políticos: decisões estratégicas sobre o ensino superior impactam negativamente na capacidade de conclusão dos alunos. Isso pode incluir aspectos como a remoção do apoio financeiro aos estudantes, o corte de recursos para as universidades, entre outros;

4. Institucionais: culturas e as práticas institucionais não apoiam os estudantes a concluírem o curso, como práticas inadequadas de avaliação, currículos e pedagogias que raramente são centrados no aluno, edifícios inacessíveis e instalações e falta de reconhecimento das necessidades dos alunos com deficiência;

5. Pessoais: doença, problemas de saúde mental, experiências traumáticas, desvio dos objetivos pessoais e ilusão de que se pode conjugar atividades extra com os estudos.

6. Aprendizagem: dificuldade em aprender por falta de base ou hábito de estudar regularmente, necessidade de habilidades prévias, qualificações ruins, baixa participação em aulas e seminários, ensino mal estruturado e não estimulante, entre outros.

Segundo documentos oficiais do Brasil [35] os fatores que influenciam a evasão dos alunos no ensino superior, podem ser descritos a partir de três categorias: (i) fatores externos a instituição, como reconhecimento social, mercado de trabalho, desvalorização da profissão, entre outros; (ii) fatores internos à instituição, como falta de clareza nos projetos, questões didático-pedagógicas, cultura institucional, entre outros; (iii) fatores individuais dos estudantes, como habilidade de estudo, escolha precoce da profissão, dificuldade de adaptação, entre outros.

No caso específico do IPB, um estudo [32] sobre o abandono escolar nos cursos do 1.º e 2.º ciclos de estudos encontrou que os principais motivos para abandono escolar foram a incompatibilidade profissional, incompatibilidade de horário, estar longe de casa, inserção no mercado de trabalho, transferência para outra instituição de ensino, dificuldades económicas e problemas pessoais, familiares.

2.4. Trabalhos relacionados

Nesta seção serão relatados artigos e dissertações relacionadas com o tema identificação dos estudantes em risco de abandono, abordando técnicas de machine learning, filtragens de dados utilizadas, ideias inovadoras para modelagem, revisões sistemáticas, entre outros estudos que serviram como base a esta dissertação.

O artigo [36] descreve uma abordagem de predição dos alunos que correm risco de abandono considerando 8 variáveis, sendo a primeira uma variável binária calculável sobre as notas médias e o desvio padrão (GPA), a segunda sobre porcentagem de presença dos alunos em salas de aula e as outras 6 são dados tratados em relação a utilização do LMS empregado pela universidade. No artigo é utilizado uma amostra de 202 alunos, nos quais produziram 200.979 registros, entre o período de 1 de abril até 5 de agosto de 2015. Foi utilizado o algoritmo *random forest* e obteve como resultado 40% de acerto.

Kew Si Na e Zaidatun Tasir [37], fizeram uma revisão sistemática sobre a modelagem computacional de identificação dos alunos em risco de abandono. Entre os levantamentos exibidos no artigo, destaca-se deste trabalho, a distribuição do uso das fontes de dados por categoria, sendo 37% dados comportamentais do LMS, 24% dados das notas dos alunos, 20% movimentações entre fóruns, publicações e comentários, 13% dados demográficos e 6% dados emocionais e psicológicos. Os algoritmos mais utilizados segundo a pesquisa, foram algoritmos de regressão e árvore de decisão, somando 33%, entretanto os autores declaram que houve grande variedade de algoritmos sendo utilizados para modelagem. O estudo não informa quais algoritmos obterão melhores resultados.

No artigo [38], os autores criaram um modelo para identificação de personalidade baseados na teoria Myers-Briggs Type Indicator (MBTI), utilizando os dados do LMS e de grupos no site Facebook, criados para compartilhamento de informação sobre as matérias. A teoria MBTI, descrita por Katharine Cook Briggs e sua filha Isabel Briggs Myers, é uma proposta de classificação de personalidades através de quatro dimensões binárias de personalidade, sendo elas, Extrovertidos ou Introvertidos, Sensoriais ou Intuitivos, Racionalistas ou Sentimentais e Julgadores ou Perceptivos. O modelo proposto foi mapear as 16 combinações das personalidades do MBTI, buscando padrões de comportamento, nos quais são descritos como 4 categorias de preferências dominantes. No estudo foi utilizado dados de 240 alunos, retirados dos logs do LMS, e obteve como resultado após executar um conjunto de 10 algoritmos de classificação, 97% e 94% de acerto com os algoritmos *OneR* e *random forest*.

Voltado para um cenário empresarial, Al-Radaideh e Al Nagi [39] utilizaram técnicas de *data mining* para previsão de performance dos funcionários de uma determinada empresa. Apesar de não ser o foco do estudo, é relatado diversas técnicas de manipulação de dados

que melhoram o desempenho dos algoritmos de classificação, como transformar atributos contínuos em categóricos, validações de desempenho, entre outros métodos que influenciaram este trabalho.

Shahiri et at. [24] produziu uma revisão literária sobre a predição da performance dos alunos utilizando técnicas de *data mining* e levantou os atributos e algoritmos mais relevantes na preparação dos modelos. Sendo em primeiro lugar o atribuído da média das notas de todas as matérias, também conhecido como coeficiente escolar, os autores sugerem que este atributo é o mais utilizado por possuir um valor tangível para futuro educacional; em segundo lugar os atributos de avaliação interna, nos quais possui características parecidas, segundo o autor, como questionários, trabalho de laboratório, teste de classe e frequência, entre outros, e em terceiro lugar os dados demográficos e avaliações externas, como idade, gênero, disponibilidade, histórico escolar, entre outros. Os algoritmos mais utilizados para a classificação foram respectivamente, árvore de decisão, redes neurais, *Naive Bayes*, *K-Nearest Neighbor* e *Support Vector Machine*, e os melhores resultados foram obtidos em ordem decrescente a partir dos algoritmos de redes neurais, árvore de decisão, *Support Vector Machine e K-Nearest Neighbor* e por último *Naive Bayes*.

Capítulo 3 Desenvolvimento

Este capítulo apresenta a trajetória do desenvolvimento deste estudo, incluindo as ferramentas utilizadas, o processamento e análise dos dados e o desenvolvimento do modelo e da ferramenta de aplicação e visualização do modelo.

3.1. Ferramentas utilizadas

3.1.1. Python

Python é uma linguagem de programação de alto nível, gratuita de código aberto, orientada a objetivos, interpretada e interativa, na qual através de sua sintaxe clara e de fácil entendimento, possibilita novos desenvolvedores se habituarem rapidamente, não exigindo refinamento de técnicas de linguagem programação ou formação no campo de tecnologia para utilizá-la. Esta linguagem também é frequentemente utilizada como uma linguagem de extensão para aplicações escritas em outras linguagem que precisam de scripts fáceis de usar ou para automação de interfaces [40] [8].

Como Raschka [41] descreve, Python se tornou a linguagem mais popular na área de *Data Science*, devido a sua capacidade de oferecer um ambiente no qual conseguimos rapidamente escrever nossas ideias e aplicar conceitos diretamente na ação, permitindo esquecermos as partes complexas e tediosas da programação. Neste trabalho foi utilizado a versão do Python 3.6, em conjunto com as bibliotecas e ferramentas descritas a seguir.

3.1.1.1. Scikit-learn

O Scikit-learn é uma biblioteca em Python, na qual contém uma ampla quantidade de algoritmos de aprendizagem de máquina de última geração, focados em solucionar problemas na área dos supervisionados e não supervisionados de média escala [42]. Esta biblioteca também acompanha um conjunto de funções e módulos para avaliar a performance dos algoritmos, facilitando a comparação entre eles.

Neste trabalho foi utilizada a implementação dos algoritmos *Random Forest*, Árvore de decisão e Redes Neurais presentes na biblioteca do Scikit-learn, juntamente com o módulo *metrics*, no qual providenciou as funções para medir precisão, acurácia e *recall* dos algoritmos.

3.1.1.2. Flask

Flask é uma framework Web minimal em Python, também conhecido por ser tão pequeno que é considerada uma micro-framework. A ideia principal deste tipo de framework é providenciar um núcleo sólido de serviços básicos, permitindo ao desenvolvedor escolher e adicionar os pacotes necessários em sua aplicação, tornando os softwares produzidos mais concisos, sem recursos irrelevantes que podem prejudicar a performance ou a segurança [43].

O Flask é baseado em duas bibliotecas principais, a primeira denominada de Werkzeug WSGI, agrega serviços http juntamente com recursos WSGI; a segunda é uma *engine* de templates moderna e amigável ao designer chamada Jinja2 [44]. Este micro-framework foi escolhido para este projeto devido a sua simplicidade e pelo fato de não precisarmos de recursos sofisticados, permitindo focar no modelo de classificação, objetivo principal deste projeto.

3.1.2. MongoDB

MongoDB é um banco de dados de documentos não relacional (NoSQL), com escalabilidade e flexibilidade. Os registros são armazenados em documentos semelhantes ao formato *JSON*, podendo variar os campos de documento para documento, com possibilidade de alterar sua estrutura ao longo do tempo. O seu núcleo é distribuído de

modo a oferecer uma alta disponibilidade, facilitando dimensionamento horizontal e a distribuição geográfica [45].

A escolha deste banco de dados baseou-se em dois motivos principais, sendo o primeiro, a facilidade de transportar os registros em formato CSV para MongoDB sem validação do domínio e, como alguns arquivos fornecidos pelo IPB não possuíam as chaves estrangeiras de suas entidades, precisaríamos entender os dados para gerar relacionamentos entre eles e passar por mais um pré-processamento, se fosse armazenar os dados em um banco de dados relacional, já que para o MongoDB nada disto foi necessário. O segundo fator desta escolha, é a indexação e agregação em tempo real que o software fornece. As operações de agregação conseguem processar uma grande quantidade de registros e retornar resultados computados em formato *JSON* modelados por nós. O MongoDB possui três maneiras de realizar agregações, *aggregation pipeline*, *map-reduce function* e single *purpose aggregation methods*, sendo possível a combinação entre elas.

3.2. Os dados

Os dados disponibilizados pelo IPB eram divididos em 14 arquivos em formato CSV, nos quais representavam diferentes entidades de bancos de dados relacionais de sistemas em atividade atualmente. Ao todo, foi fornecido dados de três sistemas, nos quais denominamos o primeiro como sistema do aluno, em que possuía dados das matrículas dos alunos, dados demográficos, quantidade de matérias, entre outros; o segundo de sistema de presenças, com as presenças dos alunos e informações das aulas e o terceiro de LMS, em que possuía os registros das atividades do Sakai. Para entendermos melhor a dimensão dos dados, construímos a Tabela 1 com as informações dos arquivos fornecidos.

Tabela 1: Listagem dos arquivos fornecidos pelo IPB.

	Entidade	Sistema	Quantidade	Volume
	Alunos	Alunos	39.510	2 MB
	Matrículas	Alunos	33.604	1,2 MB
	Anos letivos	Alunos	78.653	3 MB
	Inscrições	Alunos	699.924	31 MB
	Notas	Alunos	839.933	40 MB
	Diplomados	Alunos	27.032	1,2 MB
	Sumário	Presenças	576.852	21 MB
	Detalhes da Aula	Presenças	6.661.232	650 MB
	Presenças	Presenças	5.466.792	176 MB
	Mapeamento	-	28.414	1,6 MB
	Regras	LMS	49	1 KB
	Regras_usuários	LMS	2.031.275	95 MB
	Sessões	LMS	10.640.522	2,73 GB
	Eventos	LMS	175.931.890	42 GB
Total	14	3	203.055.682	45.752 MB

Como podemos notar na Tabela 1, apenas a entidade de mapeamento não pertence a nenhum sistema, pois ela é a ligação dos usuários do sistema Sakai com o sistema dos alunos e o sistema das presenças. Os dados fornecidos são referentes aos anos letivos de 2008 até 2017, porém identificamos que dados do ano de 2008 e do ano de 2017 não estão completos, sendo assim optamos por não os utilizar, restando apenas os dados de 2009 até 2016. A partir da Tabela 1, também é possível observar que o maior volume de dados é provido do sistema LMS, correspondendo a 97% do conjunto total, seguido do sistema de presenças com 1.8% e por último com 1.2% o sistema de informação académica.

3.3. Pré-processamento

O objetivo desta etapa era extrair de cada conjunto de dados, atributos que poderiam influenciar no modelo de classificação, para isto, os atributos deveriam ser consistentes, com a menor quantidade de dados com ruídos ou defeituosos possível e que sejam aplicáveis, nos anos anteriores e nos próximos, a todos os alunos. Apesar de possível para este modelo, não buscamos atributos mais complexos, como a média da quantidade de horas do usuário dentro do sistema Sakai, tempo médio para entrega de trabalhos, entre outros, devido ao tempo de extração e ao aumento da complexidade do trabalho, indicamos esta categoria de atributos como trabalhos futuros buscando possíveis melhorias para este modelo.

O primeiro conjunto analisado foi dos registros fornecidos do LMS, no qual buscamos entender inicialmente o domínio dos dados, para tal, produzimos o gráfico da distribuição dos dados por ano e o gráfico de crescimento de registros por semana, resultados que podem ser vistos na Figura 4.

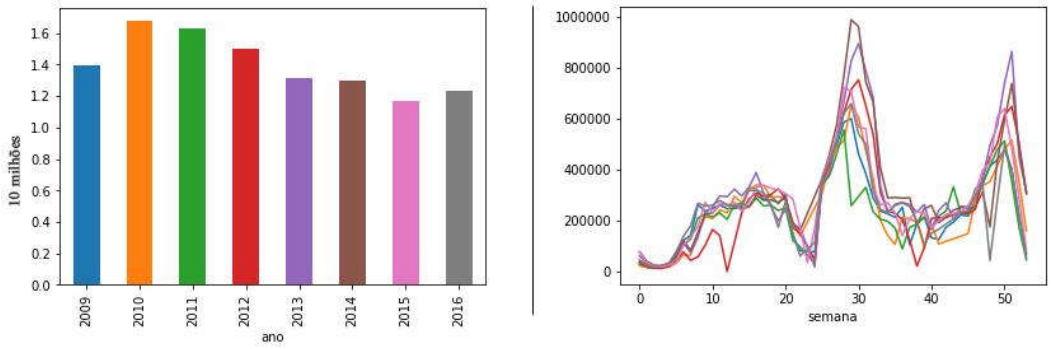

Figura 4: Gráficos da distribuição dos dados por ano e crescimento dos registros por semana do Sakai

Como podemos ver no gráfico da esquerda da Figura 4, não houve uma grande diferença entre anos consecutivos, porém se analisarmos o gráfico como um todo, é perceptível uma diminuição na utilização do Sakai. Já para o gráfico da direita, podemos notar que existem dois picos de utilização, sendo correspondentes aos finais do primeiro e segundo semestre, a maior depressão ao longo do ano letivo se encontra no período de natal e ano novo. A partir da análise dos dados, também encontramos uma diversidade de mais de 200 tipos de registros para uma média de 8600 usuários por ano.

Cada atividade realizada por um usuário dentro do Sakai produz um registro especificando o tipo de ação, como exemplo, efetuar login, abrir uma tarefa, utilizar o calendário, responder uma mensagem, entre outros, e o horário do ato. A partir disso, utilizando as operações de agregação do MongoDB, efetuamos a contagem das ações do mesmo tipo por aluno e agrupamos por semana, resultando em um novo conjunto de aproximadamente 13 milhões de dados. Com isto, computamos a quantidade de alunos que utilizam os mesmos recursos e calculou-se a média entre os anos, resultando em um *ranking*, como pode ser visto na Tabela 2.

Tabela 2: Ranking dos tipos de recursos do Sakai por média

	Tipo	Média
1º	press.begin[1]	8549
2º	content.read	8287
3º	messages.newfolder	6830
4º	messages.read	5684
5º	content.new	4605
6º	roster.view	3527
7º	webcontent.read	3496
8º	content.available	3172
9º	site.upd	3067
10º	profile.view.own	2957

A partir do *ranking* da Tabela 2, é possível notar que a quantidade média de alunos utilizando os recursos do Sakai, muda bastante a cada colocação, sendo assim, iremos utilizar apenas os tipos que foram utilizados por, acima de, 50% da quantidade média de usuários ativos anualmente, resultando em 5 atributos para nosso modelo, press.begin, content.read, messages.newfolder, messages.read, content.new, simbolizando as respectivas ações, entrada do usuário no site, download ou acesso a um documento, criação de uma mensagem, acesso a uma mensagem e criação de documento.

O segundo conjunto de dados a ser analisado foi o das matrículas, notas e dados demográficos. Para este conjunto percebemos que o intervalo deveria ser diferente, visto que alguns dados eram não mutáveis, como os dados demográficos, e outros apenas eram alterados após datas específicas, como quantidade de matérias aprovadas e reprovadas, média do aluno, etc., sendo assim, a extração precisou ser feita no período de um ano. A partir disso, relacionamos os múltiplos arquivos fornecidos através do código do aluno, curso matriculado e ano letivo, e transformamos em uma única entidade, na qual possuía os seguintes atributos: código do aluno, curso e escola, nacionalidade, sexo, data de nascimento, média de notas, quantidade de matérias aprovadas e reprovadas até aquele ano, quantidade de matérias que cursou no ano e o status, representando a situação no final do ano, sendo desistente ou não desistente.

Com este conjunto formado, buscamos extrair características que poderiam nos ajudar a montar e melhorar o modelo. Entre as particularidades descobertas, destaca-se a

[1] O Sakai também produz o registros de mais três tipos de ações semelhantes ao press.begin, simbolizando o momento de criação da sessão no navegador, término da sessão e o momento que o usuário saiu do site, porém como são relativamente parecidos e não iremos utilizar o cálculo entre eles, escolhemos apenas o press.begin para nosso modelo.

porcentagem de desistentes, possuindo uma média de 20% entre os anos para um conjunto de aproximadamente 7000 alunos ativos por ano. Para entender melhor esta distribuição produzimos o gráfico da Figura 5, comparando a quantidade de desistentes e não desistente entre os anos disponíveis.

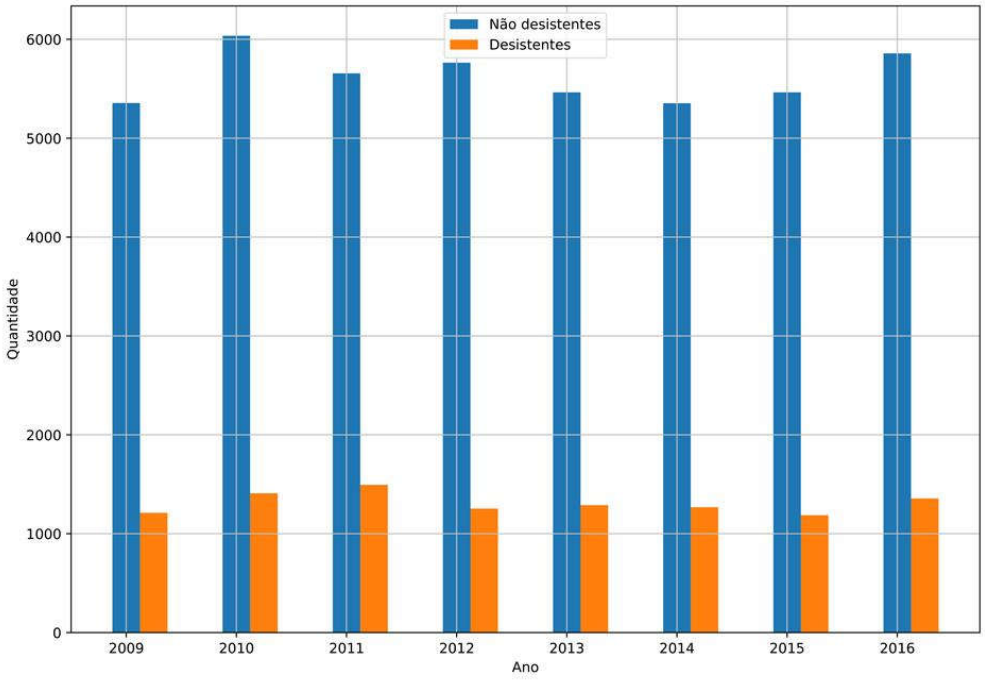

Figura 5: Quantidade de desistentes por ano letivo.

O terceiro conjunto, referente a presenças em sala de aula, foi o mais simples, no qual efetuamos uma contagem das aulas por semana de cada aluno e transformado em porcentagem, produzindo o último atributo que faltava.

A partir disto, os dados de cada conjunto processado foram agrupados em apenas uma tabela, resultando em um conjunto de 3.8 milhões de registros com 20 atributos, sendo eles, código do aluno, sexo, idade, nacionalidade, semana e ano da extração, código do curso, código da escola, ano letivo, presença em sala de aula, média de notas, quantidade de matérias aprovadas e reprovadas, quantidade de matérias cursando, press.begin, content.read, messages.newfolder, messages.read, content.new e o status.

3.4. Escolha do algoritmo

Como já mencionado, existe um vasto conjunto de algoritmos que poderiam se encaixar em nosso modelo, porém como já explicado, selecionamos apenas três deles para comparação (Árvore de decisão, *Random Forest* e Redes Neurais). Para avaliar o desempenho destes algoritmos e do modelo como um todo, utilizamos duas métricas principais, sendo elas, a precisão e o *recall* (sensibilidade). Apesar de já explicado seus conceitos, é prudente entendermos quais os significados destas métricas em nosso ambiente e qual aspecto do modelo cada uma pretende representar. Sendo assim, podemos traduzir precisão como a porcentagem de alunos classificados como corretamente como desistentes sobre o conjunto total de desistentes e *recall* como a porcentagem de alunos classificados corretamente como desistentes dentro do conjunto de classificados.

Ainda sobre as métricas, levando em consideração nosso problema, a medida de desempenho *recall* se torna a mais importante entre as duas utilizadas, visto que está diretamente ligada ao consumo de recursos da instituição, como exemplo, se o *recall* for 100% e a precisão 50%, significa que classificamos metade dos desistentes corretamente e caso não sejam devidamente atendidos irão realmente desistir, porém se a precisão for 100% e o *recall* 50%, significa que classificamos todos os desistentes corretamente, porém a mesma quantidade de pessoas que não irão desistir também foram classificados como desistentes, diminuindo a atenção de quem realmente precisa e tentando resolver problemas que não existem.

Inicialmente para testar qual o algoritmo que encaixa melhor no modelo, utilizamos os dados de 2009 até 2015 para montar dois conjuntos de treino, o primeiro sem nenhuma alteração e o segundo balanceando as duas classes em 50% cada uma, visto que estavam desproporcionais em 20% desistentes 80% não desistentes. Como teste foi utilizado os dados de 2016, os dados de 2017 foram descartados, pois não estavam completos com todas as semanas. Por fim, executamos este teste em quatro semanas diferentes (10, 30, 40 e 50), para verificar a estabilidade, os resultados podem ser vistos nas tabela 2 e 3.

Tabela 3: Teste com o conjunto de treino sem alteração.

Algoritmos	Semana 10	Semana 30	Semana 40	Semana 50
Árvore de Decisão	22% - 48%	34% - 49%	34% - 62%	37% - 59%
Redes Neurais	24% - 49%	21% - 55%	24% - 65%	29% - 62%
Random Forest	13% - 68%	22% - 73%	29% - 75%	35% - 75%

Tabela 4: Teste com o conjunto de treino balanceado.

Algoritmos	Semana 10	Semana 30	Semana 40	Semana 50
Árvore de Decisão	65% - 33%	61% - 41%	65% - 46%	73% - 42%
Redes Neurais	74% - 31%	68% - 37%	69% - 36%	62% - 48%
Random Forest	55% - 40%	64% - 47%	64% - 52%	66% - 52%

As porcentagens exibidas nas tabelas 2 e 3, representam respectivamente a precisão e o *recall* que foram obtidos em cada teste. Comparando as duas tabelas, é possível notar que o balanceamento no treino deu uma diferença significante nos resultados, conseguindo melhorar todas as precisões, porém diminuindo todas as sensibilidades, tornando o modelo menos eficaz em nosso problema. Em todos os casos o algoritmo *Random Forest* obteve as melhores sensibilidades, porém nenhuma vez atingiu a melhor precisão, alternando entre segundo e terceiro lugar, já os outros dois algoritmos conseguiram resultados semelhantes, trocando de posições sobre o mais preciso em cada semana.

Outros testes também foram realizados, como separar os dados de treino por ano letivo, escola e curso, porém não apresentaram melhores resultados, indicando que os comportamentos de desistência tendem a ser semelhantes em todos os anos. Sendo assim, o algoritmo *Random Forest*, foi escolhido como mais adequado a nosso problema, do que os outros algoritmos testados, conseguindo atingir uma sensibilidade maior sem perder muita precisão.

3.5. Aprimoramento do modelo

Os primeiros testes para tornar o modelo mais poderoso, foram relacionados aos atributos disponíveis. O algoritmo selecionado *Random Forest* consegue produzir uma lista ordenada dos atributos mais relevantes na classificação, sendo assim, verificamos se a remoção de atributos menos pertinentes, resultaria em uma classificação mais precisa ou mais sensível, porém os resultados mostraram que em apenas algumas semanas foi obtido números melhores, variando as métricas tanto para cima quanto para baixo. Sendo assim, como desejamos um conjunto não mutável de atributos para classificar todas as semanas, concluímos que a remoção de atributos menos pertinentes não é proveitosa para nosso modelo neste momento.

Como tínhamos um intervalo considerável de anos (2009 - 2016), aproveitamos este volume de dados para identificar qual seria o melhor ciclo de treino para o algoritmo, visto que os dados podem ser muito distintos ao longo dos anos, para isso, testamos todas as combinações de anos possíveis. Começando com o intervalo de um ano de treino, o primeiro passo foi calcular todas as semanas do ano de 2010 utilizando os dados de 2009, no final produzimos a precisão e o *recall* médio do ano, o mesmo foi executado no ano de 2011 utilizando os dados do seu ano anterior (2010), e assim por diante, até 2016. Após a execução, calculamos a média entre todos estes resultados, produzindo as métricas médias do uso de um ano de treinamento. Replicamos o teste para o intervalo de 2 a 7 anos, produzindo o resultado exibido na Tabela 4.

Tabela 5: Comparação entre os intervalos de anos de treinamento.

Métricas	**1 ano**	**2 anos**	**3 anos**	**4 anos**	**5 anos**	**6 anos**	**7 anos**
Precisão	21.08%	22.16%	22.89%	22.62%	22.92%	22.46%	22.29%
Recall	68.81%	70.53%	70.42%	71.46%	70.59%	69.76%	70.98%

O primeiro ponto a se considerar sobre este teste, é a possível diferença na replicação do mesmo com mais anos disponíveis, visto que, diferente de um ano de treinamento, no qual foi testado 6 vezes para calcular a média, o resultado para sete anos foi testado apenas uma, possibilitando resultados diferentes se testado mais vezes, tornando-o menos instável na comparação. A partir da tabela 4, podemos notar que os valores são bem semelhantes, porém replicamos o teste múltiplas vezes e obtemos os mesmos resultados. Sendo assim, considerando a instabilidade já descrita e a importância do *recall* em nosso cenário, escolhemos utilizar o intervalo de 4 anos para treino.

Outro problema que ainda persistia em nosso modelo, era a incapacidade de utilizar as informações de múltiplas semanas para treinamento, como ideia de trajetória do aluno no ano letivo. Apesar de serem acumulativos os atributos, não conseguimos mensurar a regularidade de crescimento. Para facilitar o entendimento deste caso, considere dois alunos ativos e suponha que estamos na semana 10 do ano letivo, o aluno A entra regularmente no LMS e produziu até o momento 50 logs de entrada no sistema, já o aluno B, nunca entrou no LMS até a semana 8, porém em duas semanas, entrou 50 vezes no sistema, produzindo a mesma quantidade de logs, sendo assim, para o modelo, na semana 10 os alunos A e B são iguais, ambos com 50 logs cada.

A partir disto, para tentar resolver ou minimizar este problema, tivemos a ideia de adicionar dois atributos em nosso modelo, o primeiro representando a classificação concedida ao aluno na semana anterior denominado de last_predicted e o segundo referente a quantidade de vezes que o algoritmo classificou o aluno como desistente até aquela semana, em porcentagem, sendo assim denominado de critical_rate. Porém para isto funcionar, precisávamos de um conjunto de treino de 4 anos com estes atributos, sendo assim, classificamos os anos de 2012, 2013, 2014 e 2015 utilizando seus respectivos 4 anos anteriores e após finalizado calculamos para cada aluno em cada semana os dois novos atributos. Por fim, para o ano de 2016 utilizamos o conjunto de treino produzido com estes atributos, e a cada semana de 2016, calculamos os atributos e aplicamos o modelo. Na Figura 6 é possível visualizar o impacto dos novos atributos em nosso modelo.

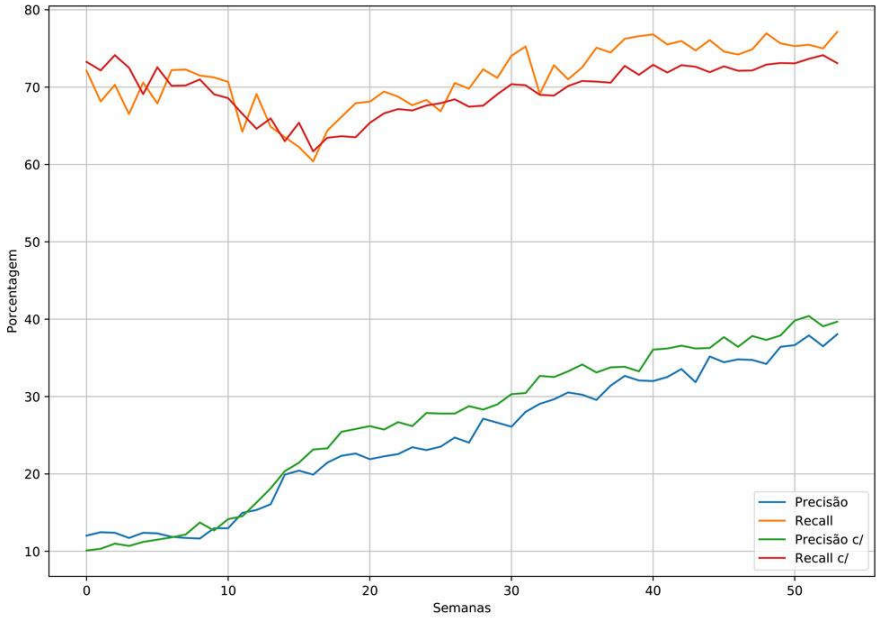

Figura 6: Comparação do uso dos novos atributos.

Como é possível visualizar na Figura 6, não houve uma diferença significante nas métricas de desempenho utilizando os novos atributos, conseguindo um pequeno aumento na precisão e perdendo sensibilidade em algumas semanas, porém ganhou linearidade ao longo das semanas. Como podemos notar, o *recall* sem os novos atributos (linha laranja) é bastante volátil, quando verificamos os alunos que estão sendo selecionados, a cada semana conseguimos identificar o surgimento de novos alunos que não tinham sido

selecionados anteriormente, já para o modelo com os atributos este número é menor, repetindo mais vezes os mesmos alunos, criando consistência na classificação ao longo do ano. Outro fato que suporta esse argumento é a lista ordenada da relevância dos atributos que a *Random Forest* pode produzir, exibida na Tabela 5.

Tabela 6: Lista dos atributos mais relevantes ao longo do ano.

Atributos	Semana 10	Semana 20	Semana 30	Semana 40	Semana 50
critical_rate	11.16%	14.14%	14.3%	15.01%	17.01%
presença	10.53%	10.62%	10.34%	11.42%	10.01%
press.begin	6.67%	10.52%	7.51%	9.37%	8.94%
content.read	7.14%	6.83%	9.29%	8.04%	8.06%
materias estudando	10.66%	7.85%	6.96%	6.41%	6.70%
materias aprovadas	10.02%	7.29%	5.71%	6.05%	6.81%
last_predicted	5.46%	7.84%	9.24%	8.03%	9.33%

Como podemos ver na Tabela 5, o atributo critical_rate, além de estar em primeiro lugar em todas as semanas, também aumenta sua relevância ao longo do ano. Esse aumento significa que o modelo está considerando cada vez mais seu próprio julgamento das semanas anteriores, mostrando o impacto que a trajetória do aluno ao longo do ano tem sobre a futura classificação.

3.6. Ferramenta de aplicação e visualização do modelo

Seguindo com os objetivos deste trabalho, além da produção do modelo de classificação, ainda nos faltava elaborar uma ferramenta para aplicação deste modelo. Contudo, como o modelo propõe, possuímos como requisito funcional a extração dos dados semanalmente, nos colocando o tempo como um fator importante na modelagem deste software, visto que, após a extração o modelo é necessário processar os dados, convertendo-os em formato único da aplicação do modelo, e ao final recalcular os atributos criados (critical_rate e last_predicted).

3.6.1. Estrutura

Para tal, arquitetamos uma estrutura que conseguisse suprir todos os requisitos, contendo uma ferramenta de extração de dados (ETL) transportando os dados dos sistemas ativos

no IPB em bancos de dados Oracle e transformando-os em documentos do MongoDB, e uma aplicação Web que conseguisse traduzir os resultados em uma interface amigável com níveis de acesso entre escolas e cursos. Para entender melhor a organização estrutural projetada para esta ferramenta, desenvolvemos o diagrama da Figura 7.

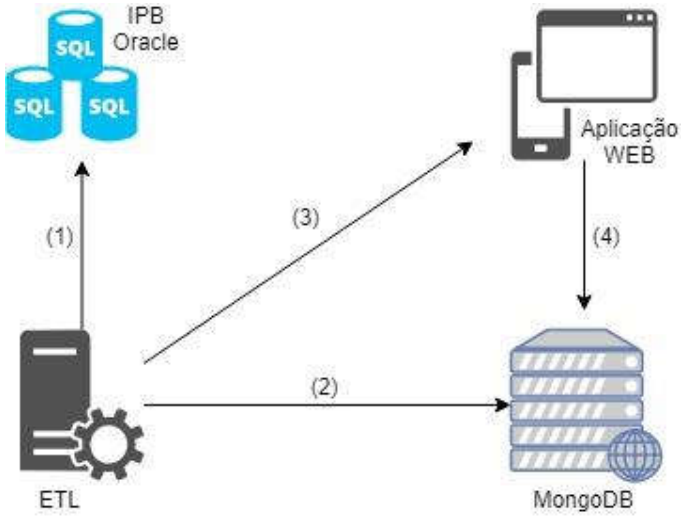

Figura 7: Diagrama da estrutura da ferramenta

Como podemos notar pela Figura 7, a ferramenta ETL é responsável por três das quatros ações principais desta estrutura (representadas pelas setas), sendo a primeira ação (1) efetuar a extração dos dados dos sistemas do IPB, a segunda (2) é transportar para o MongoDB, já com os dados em formato correto para classificação e por último, a terceira ação (3) é notificar a aplicação Web que os dados foram transportados e estão aptos a serem utilizados. Por sua vez a aplicação Web termina nosso ciclo de tarefas com a quarta ação (4), na qual deve-se verificar os dados fornecido pela ferramenta ETL e aplicar o modelo, calculando no final os novos atributos.

O principal motivo de não executarmos todas as ações na aplicação Web, é a segurança dos dados. Na estrutura proposta, conseguimos instalar nos servidores do IPB, a ferramenta ETL em uma máquina que não concede acesso por meios externos, com privilégios de leitura aos sistema ativos, sendo capaz de extrair e processar esses dados sem informações pessoais dos alunos e transportar para o MongoDB, no qual a aplicação Web já possui acesso, garantindo que agentes maliciosos não acessem os servidores ativos no IPB.

3.6.2. ETL

Ferramentas ETL são softwares responsáveis por efetuar extrações de fontes operacionais heterogêneas de dados, transformá-los, aplicando conversões, associações, limpeza, entre outras técnicas, e carregá-los em outro sistema, normalmente em *Data Warehouses*. As etapas deste software devem ser bem estruturadas e auditáveis, pois, são consideradas peças chaves para o sucesso da aplicação final [46].

Partindo do pressuposto que iremos aplicar inicialmente esta ferramenta apenas ao IPB, iremos integrar unicamente com banco de dados Oracle. Assim como os dados fornecidos, trabalharemos com três conjunto de dados providos de três sistemas distintos, o LMS, sistema de notas, dados demográficos e sistema de presenças. O código desenvolvido para a extração de dados foi baseado no estudo descrito como pré-processamento (capítulo 3.3), porém ao invés de agregações em MongoDB, tivemos que lidar com os relacionamentos do modelo relacional e efetuamos as contagens necessárias entre intervalos de datas.

Apesar de serem volumosos e de fontes distintas os dados que pretendemos extrair, como o intervalo de datas é pequeno, sempre de uma semana, o Oracle foi capaz de retornar todos os dados em apenas uma *query*, porém alguns atributos desejáveis ainda necessitaram de transformações antes de serem enviados ao MongoDB.

A aplicação ETL foi escrita como um script em Python, necessitando de apenas três bibliotecas como requisito, cx_Oracle, pymongo e requests, sendo respectivamente, a biblioteca responsável por fazer a ligação com o Oracle, a biblioteca responsável por fazer a ligação com o MongoDB e a biblioteca responsável por efetuar requisições online, como o método POST para a aplicação online, anunciando o fim da extração.

Instalamos a ferramenta em um servidor Linux e agendamos através do serviço *cron*, uma tarefa de executar nosso script todas as segundas feiras a 01:00 AM, período no qual tende a ter menos usuários ativos no sistema, caso o processo cause lentidão. Para a auditoria da ferramenta, a aplicação produz logs em um arquivo a cada início de atividade, como inicialização do script, acesso ao banco de dados e requisição da aplicação Web, além disso, é enviado ao MongoDB informações de cada coleta quando termina o processo.

3.6.3. Aplicação Web

Considerando a expansão dos múltiplos dispositivos nos últimos anos e os avanços tecnológicos dos sistemas operacionais móveis, a escolha de uma aplicação online, acessível por múltiplos ambientes se fez evidente. Apesar dos aplicativos móveis ganharem um volume considerável e serem recomendado para diversos casos, neste projeto optamos por desenvolver um web site, no qual também consegue ser acessível por smartphones e possui uma visualização dos elementos melhor em desktops, por exemplo, os gráficos e as tabelas, que perdem detalhes quando muito reduzidos.

A linguagem para o desenvolvimento da aplicação também foi o Python, devido a economia de tempo que conseguimos mantendo os códigos desenvolvidos para o modelo, visto que tínhamos a certeza que responderiam exatamente igual no mesmo ambiente. Além disto, Python também oferece ótimas soluções web, como o Framework Flask escolhido para este projeto. Inicialmente comparamos o Flask com o Framework Django, porém apesar de eficiente e de fácil aprendizagem, com diversos módulos pré-instalados que normalmente facilitam o desenvolvedor, para nós não tinha muita utilidade, aumentando a complexidade do projeto sem necessidade.

Entretanto, como já descrito, Flask é um microframework, no qual não requer um paradigma de programação específico ou uma determinada estrutura de arquivos, sendo possível desenvolver um servidor completo em apenas um arquivo de forma procedural, porém como desejamos produzir um código de fácil entendimento, habilitando outros desenvolvedores a aplicar manutenções e melhorias, é necessário a utilização de padrões de projeto e de arquitetura.

3.6.3.1. Estrutura de arquivos

Para a estrutura do projeto escolhemos o padrão Model-View-Controller (MVC), devido a sua capacidade de manter o código organizado e enxuto, permitindo de forma simples a reciclagem do mesmo sem dificuldade e com segurança. A arquitetura MVC tem o objetivo de separar a lógica do sistema e a interface do usuário, para isto, é sugerido a divisão do software em três componentes Modelo, Visão e Controle. Podemos definir a camada Modelo como a componente responsável de se comunicar com o armazenamento dos dados e trabalha na manipulação dos dados internos. A componente Visão, também

conhecida como apresentação, é responsável por capturar as ações do usuário e fornecer a interface da aplicação. Por fim, a camada Controle é responsável por controlar o fluxo entre as camadas, executar as funcionalidade que envolvem o comportamento da aplicação e gerar as respostas aos usuários [47].

A partir disso, iniciamos a aplicação organizando os arquivos com base na arquitetura MVC, dividindo-os em subpastas que representam diferentes componentes e criamos os arquivos de configuração e os arquivos básicos de cada camada. Para melhorar o entendimento desta hierarquia, desenvolvemos a Listagem 1.

Listagem 1: Estrutura básica da aplicação WEB.

```
1  config.py
2  index.py
3  app/
4  ├── __init__.py
5  ├── controllers/
6  |    ├── users.py
7  |    └── __init__.py
8  ├── forms/
9  |    └── user.py
10 ├── models/
11 |    ├── base.py
12 |    └── user.py
13 ├── static/
14 |    ├── css/
15 |    └── js/
16 |    └── imgs/
17 └── templates/
18      ├── error.html
19      ├── login.html
20      ├── layout.html
21      └── users/
```

Como podemos notar na Listagem 1, na linha 1 e 2 possui dois arquivos fora da pasta principal (app), estes arquivos são de inicialização da aplicação. O primeiro config.py é responsável por armazenar as configurações do banco de dados, diretório do projeto, chave secreta[2], informações da aplicação, e-mails e níveis de acesso. Já o segundo arquivo index.py, é responsável pela chamada de ligação com a pasta principal, normalmente separamos estes dois arquivos da pasta principal quando se possui mais de um serviço associado, conseguindo inicializar e configurá-los juntos, porém apesar de termos apenas um serviço, mantemos esta separação por questão de organização.

[2] A chave secreta neste projeto é uma string de 32 caracteres, na qual é utilizada para embaralhar as senhas dos usuários cadastrados.

Na pasta app, possui apenas um arquivo chamado __init___.py (linha 4), este documento é responsável por três questões fundamentais da aplicação. A primeira é importar as bibliotecas base para aplicação, sendo as duas principais, o próprio Framework Flask e a biblioteca de ligação com o MongoDB, chamada Mongo Engine. A segunda é instanciar e interligar estas bibliotecas com as configurações pré-definidas no arquivo config.py. Por último, este arquivo também é responsável por criar variáveis, objetos e funções globais, como variáveis da semana e do ano letivo, objeto do banco de dados, funções de rota para erros, entre outros. Podemos considerar que este é o arquivo base da aplicação, pois após sua chamada o servidor já está configurado e funcionando, porém, não possui nenhuma rota além das páginas de erro (403, 404, 405 e 500).

Para a camada Controle da arquitetura MVC, reservamos a pasta controllers (linha 5), na qual irá conter múltiplos arquivos divididos por conjunto de rotas que desejamos criar, a nomenclatura destes arquivos devem ser em plural, como exemplo, na Listagem 1 na linha 6, temos o arquivo users.py, nele são encontrados todas as rotas referente aos usuários do sistema, incluindo o CRUD, login, logout, recuperação de senha entre outros. Como todas as rotas precisam estar ativas na inicialização da aplicação, a pasta controllers é importada pelo arquivo base do projeto e através do arquivo de inicialização da pasta controllers (linha 7) os arquivos de controle são retornados.

A pasta models ficou determinada para o componente Modelo do MVC, nela deverá conter os arquivos separados por entidades, com sua nomenclatura em singular, como exemplo, na linha 12 da Listagem 1, o arquivo user.py é responsável por armazenar informações dos campos da entidade usuário. O arquivo base.py (linha 11) é uma classe abstrata, na qual fornece um escopo para as outras entidades. A pasta forms, é responsável por armazenar os arquivos referentes aos formulários, como exemplo, o arquivo user.py (linha 9), deverá conter os formulários de login, cadastro e edição do usuário.

Para a camada da interface do usuário (Visão), foi necessário um conjunto de pastas para nos organizarmos. Primeiramente na pasta principal do projeto, foram divididos os arquivos HTML dos arquivos CSS, Javascript e as imagens, respectivamente nas pastas templates e static, após este processo, subdividimos os arquivos HTML entre os pertencentes a uma determinada rota, com novas pastas utilizando a mesma nomenclatura do controle (como exemplo, a pasta users da linha 18) e os arquivos modelo de cada

layout da aplicação. Como esta camada possui uma complexidade maior em termos de organização e utiliza diversas bibliotecas exclusivas para layout, criamos a seção a seguir.

3.6.3.2. Interface do usuário

Dado que possuíamos o requisito de uma interface amigável aos usuários e facilmente podemos encontrar painéis administrativos com esta e outras características de código aberto, optamos por utilizar um painel já estruturado ao invés de desenvolvê-lo. O AdminLTE escolhido para este projeto, é um modelo de painel administrativo baseado em Bootstrap 3 de código aberto, no qual possui um vasto conjunto de elementos responsivos a múltiplos dispositivos, com uma ótima documentação que facilita a implementação, além de uma interface intuitiva de boa aparência.

A partir disto, como já mencionado na seção 3.6.3.1, nós dividimos a interface em dois conjuntos, os modelos e os núcleos das páginas. Os arquivos modelos (error.html, login.html, layout.html), contém as partes estáticas de cada layout, como exemplo, o arquivo layout.html contém o cabeçalho, menu lateral e rodapé do layout da aplicação, faltando unicamente o corpo, no qual deverá ser importado o conteúdo da rota em questão.

Para versão inicial da aplicação, desenvolvemos quatro conjuntos de rotas essenciais para o projeto, descritas a seguir:

- Estatísticas (Página inicial): Geral, escolas e cursos;
- Estudantes: Listagem dos estudantes e visualização individual.
- Coletas: Listagem das coletas;
- Usuários: CRUD, login e recuperação de senha;

Nesta aplicação, também foi utilizado algumas bibliotecas Javascript para melhorar a interação do usuário com o sistema, sendo as duas principais JQuery e AngularJS. A primeira biblioteca JQuery, é requerida pelo painel AdminLTE para gerenciamento de alguns elementos e foi utilizada nos conjuntos de rotas estudantes, coletas e usuários. A segunda biblioteca AngularJS, foi utilizada exclusivamente na página inicial, visto que, precisaríamos de uma dinâmica maior para gerenciar múltiplas requisições, além de diversos gráficos e tabelas, esta biblioteca se fez necessária, conseguindo manipular de forma simples os dados e os elementos após ao retorno das requisições.

3.6.3.3. Níveis de acesso

Para solucionar a questão dos níveis de acesso com base na hierarquia do IPB, no qual são necessários no mínimo três regras com limitação de conteúdo, criamos uma função decoradora e um atributo chamado role na entidade usuário. Uma função decoradora ou decoradores de funções, são funções de extensão ou de aprimoramento de uma outra função. Seu objetivo principal é executar algum procedimento antes ou depois da função principal ser executada sem modificá-la. Neste projeto, a função decoradora é chamada antes de cada rota, na qual deverá especificar qual o mínimo usuário de acesso para validação.

O atributo role é do tipo inteiro, no qual apenas pode assumir os valores determinados por uma lista enumerada encontrada no arquivo config.py. A ideia da lista enumerada é criar uma hierarquia nos níveis de acessos, em nossa aplicação, como padrão definimos três níveis, no qual denominados de *administrator*, *coordinator* e *master*, e possuem a numeração 0, 1 e 2 respectivamente, com isto, através da função decoradora será validado o usuário com o mínimo de permissão necessária para utilizar a rota, restringindo ou liberando o acesso a partir do resultado. A partir disso, seguindo esta lógica de hierarquias, é possível adicionar outros níveis de acesso apenas adicionando mais um rótulo na lista enumerada. Para entendermos melhor o que cada nível terá acesso, desenvolvemos o diagrama de caso de uso da Figura 8.

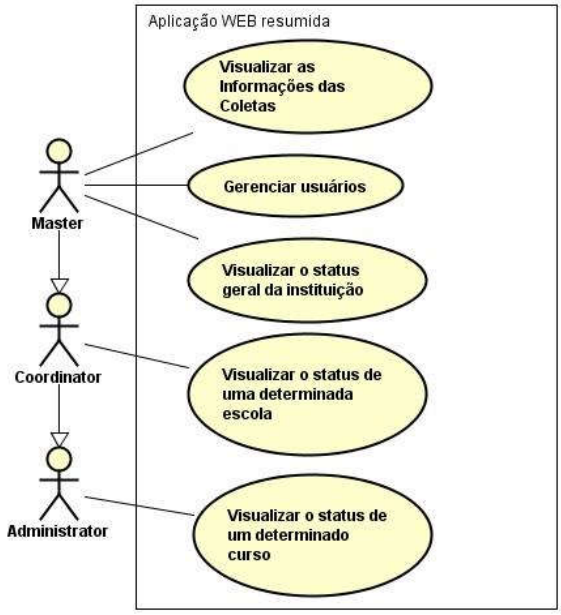

Figura 8: Diagrama de caso de uso da aplicação Web resumida.

O diagrama de caso de uso da Figura 8 é uma versão simplificada da aplicação com os principais casos de uso. Como é possível visualizar os atores possuem uma hierarquia representada pela herança, sendo o *master* topo da hierarquia, conseguindo acessar todos os casos e o *administrator* o último, apenas com um caso de uso.

Cada usuário com nível de acesso abaixo do *master*, possui um código associado. Se caso for *administrator*, o código de um curso e se caso for *coordinator*, o código de uma escola. Com isto, a aplicação só irá mostrar as informações atreladas a este código específico, como exemplo, um usuário *coordinator*, conseguirá acesso a todas as informações da escola, como comparações entre cursos, levantamentos, entre outros e o acesso a todos os cursos pertencentes aquela escola, já o *administrator* só terá acesso a um curso específico.

3.6.3.4. Integração com a ferramenta ETL e hospedagem

Com as interfaces desenvolvidas e a aplicação já estruturada com os níveis de acesso, só nos faltava desenvolver a integração com a ferramenta ETL e instalar a aplicação em um servidor. Sendo assim, relembrando do diagrama da estrutura da Figura 7, após os dados serem extraídos e enviados para o MongoDB, a própria ferramenta ETL enviará uma requisição notificando os dados extraídos, nos restando a tarefa de calcular os dois atributos criados e aplicar o modelo de classificação.

Para resolver este problema, criamos duas funções, sendo elas, o pré-processamento e a própria requisição que deverá aplicar o modelo. A função pré-processamento é capaz de aplicar as funções de agregação do MongoDB e retornar os novos atributos já calculados, restando apenas a tarefa de associá-los aos devidos alunos na semana extraída. Já a função para aplicação do modelo, primeiramente faz a chamada do pré-processamento e após gera a classificação utilizando os dados de treino dos últimos 4 anos relativos a semana. Por fim os dados são editados no MongoDB com a predição e os novos atributos atualizados, liberando a plataforma exibi-los na interface.

A aplicação Web e o banco de dados MongoDB foram instalados no mesmo servidor Linux com acesso público. Entretanto para o Framework Flask funcionar corretamente no servidor, foi necessário instalarmos três outros serviços. O primeiro chamado Gunicorn, é um servidor Python WSGI, no qual foi instalado a aplicação; O segundo denominado Supervisor, é um serviço que permite monitorar e controlar os processos do Gunicorn; Por último, instalamos o serviço Nginx como servidor de proxy reverso.

Capítulo 4 Análise e Discussão de Resultados

Este capítulo apresenta uma análise sobre as características do modelo proposto e um compilado das páginas da aplicação Web, com seus diferentes níveis de acesso, gráficos e tabelas de comparações entre escolas e cursos.

4.1. Análise do modelo proposto

Após concluirmos o desenvolvimento do modelo, buscamos explorar a competência do estudo analisando formas de representar o conhecimento produzido, no qual não se resumia apenas a classificação, mas também as informações que conseguimos obter e relacionar a partir dos dados extraídos. Sendo assim, elaboramos um conjunto de gráficos que nos proporciona uma investigação mais profunda dos resultados do modelo proposto, permitindo mensurar de forma mais nítida a eficiência do modelo. Continuando com a utilização do ano de 2016 como conjunto de teste e seus respectivos 4 anos anteriores como treino, visto que, até o momento esta combinação é a mais completa que possuímos, inicialmente buscamos entender quais são os impactos na prática das duas medidas de avaliação utilizadas (precisão e *recall*). Para tal, construímos o gráfico da Figura 9, no qual apresenta a comparação entre a quantidade de alunos selecionados, selecionados corretamente e o total de alunos desistentes no ano de 2016 e valores utilizados para construção das métricas.

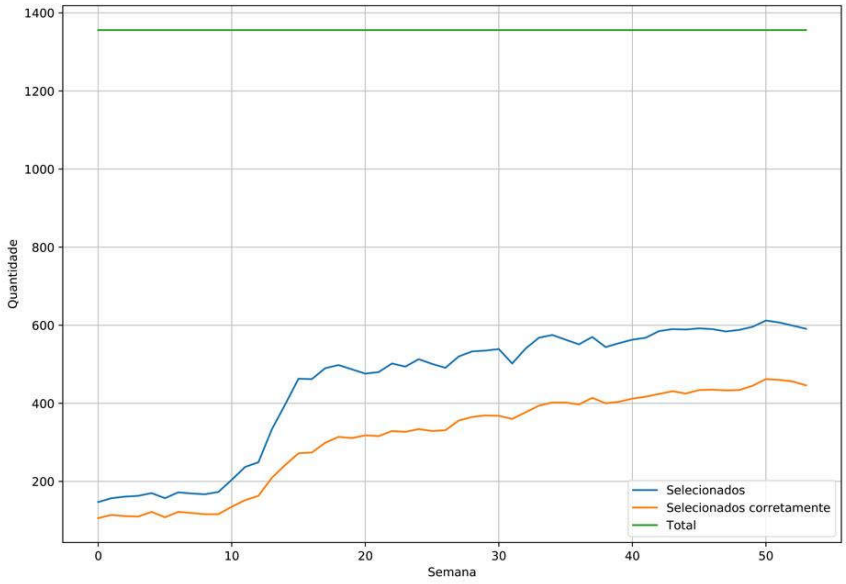

Figura 9: Gráfico de comparação entre a quantidade de selecionados e selecionados corretamente.

O encontro das linhas azul e laranja da simboliza a máxima da métrica *recall* (100%), representando que todos os alunos selecionados estão corretos. Já para a precisão o equivalente seria o encontro da linha laranja com a reta verde, representando que todos os alunos desistentes foram selecionados corretamente. A partir disso, o modelo ideal seria aquele que conseguisse igualar as linhas azul e laranja à reta verde, simbolizando que todos os alunos selecionados foram selecionados corretamente e que não existe nenhum aluno desistente fora do conjunto de selecionados.

Analisando o gráfico, podemos observar que nosso modelo está um pouco distante do ideal, iniciando com um pequeno número de alunos selecionados, devido à baixa variedade de informações acumuladas, e ao decorrer do ano conseguindo identificar com mais facilidade os alunos com risco de abandono, aumentando gradativamente a quantidade de selecionados e selecionados corretamente a cada semana. Entretanto mesmo nas melhores semanas ainda permanece com uma boa diferença entre as linhas azul e laranja com a reta verde.

Para minimizar esse déficit, percebemos que além de ser essencial para a classificação, o atributo criado critical_rate também se apresenta como um índice de ordenação por prioridade, mostrando que os alunos que são constantemente selecionados como desistentes realmente possuem uma tendência maior a abandonar o curso. Para entender

melhor este índice, inicialmente analisaremos as métricas de avaliação de diferentes setores do atributo critical_rate.

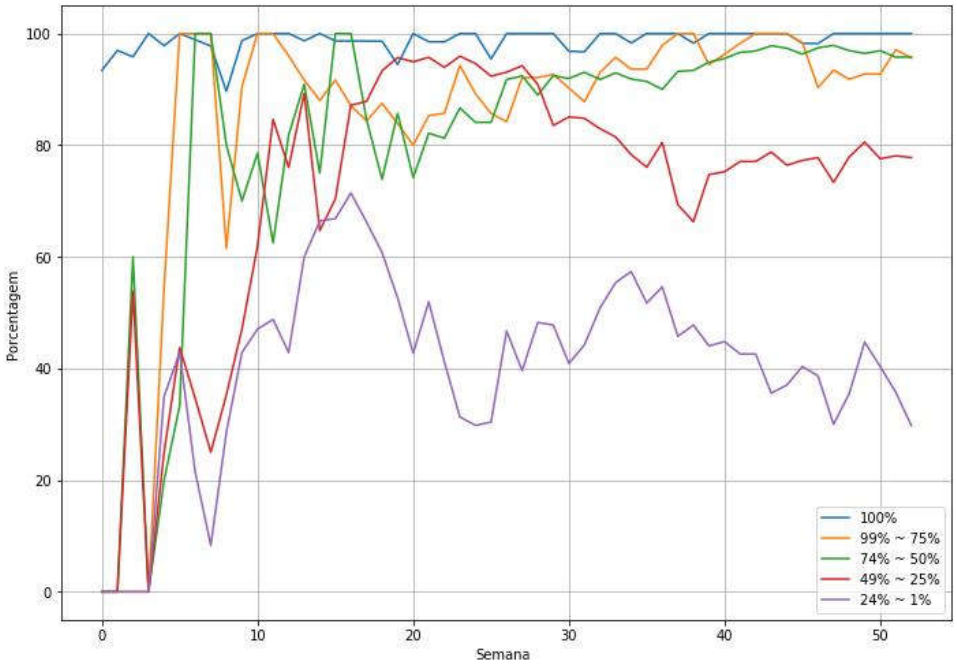

Figura 10: Precisão por setores do atributo critical_rate

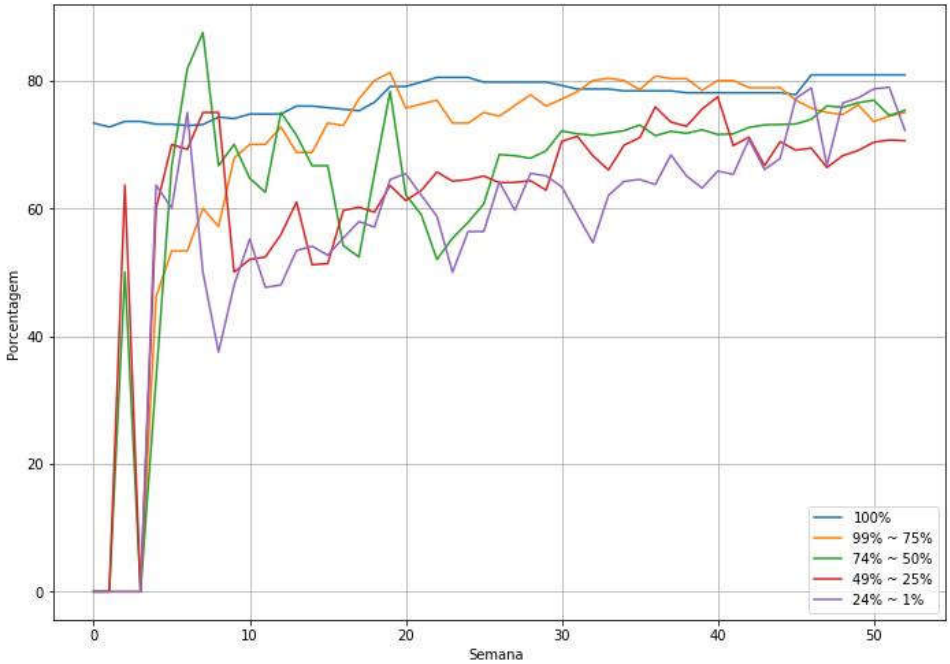

Figura 11: *Recall* por setores do atributo critical_rate

Como podemos visualizar no capítulo anterior na Figura 6, a precisão e o *recall* do modelo inteiro, está entre, respectivamente $[10, 40]$ e $[60, 80]$, porém como podemos ver na Figura 10 e na Figura 11, quando distribuímos os alunos pelo atributo critical_rate e recalculamos as métricas, os valores são consideravelmente melhores. A precisão média para alunos classificados todas as vezes como desistentes é de 98% e seu *recall* de 77%; sendo assim, a chance, nesta categoria, de escolher um aluno ao acaso e ele realmente desistir no final do ano é aproximadamente 8 a cada 10 alunos. Entretanto para os alunos classificados entre 24% a 1%, a precisão é bem menor e bastante volátil, representando o grupo de alunos classificados pelas primeiras vezes como desistentes.

Para obter uma comparação mais fiel aos valores quantitativos da Figura 9, reconstruímos o mesmo gráfico, porém com a adição de duas novas linhas, nas quais representam a quantidade de alunos selecionados e selecionados corretamente semanalmente acima de 50% no atributo critical_rate.

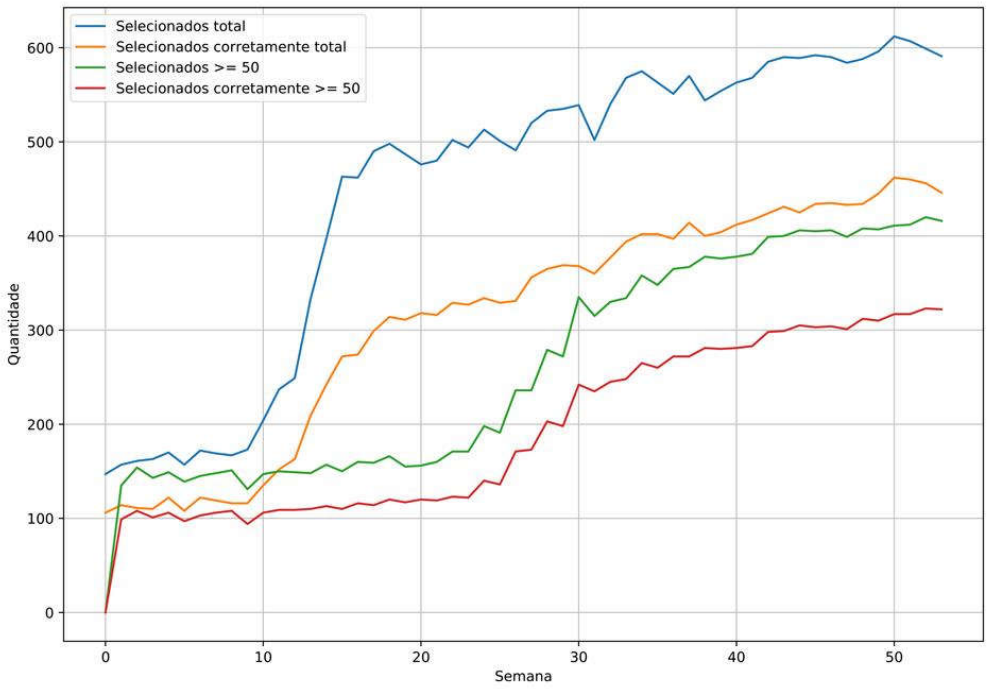

Figura 12: Comparação entre selecionados total e selecionados acima de 50% no critical_rate.

Como podemos perceber na Figura 12, as linhas adicionadas (verde e vermelha) possuem uma distância entre si notadamente inferior em comparação as linhas azul e laranja, conseguindo atingir em seu pior intervalo menos de 100 alunos classificados

erroneamente, contra aproximadamente o dobro no modelo sem a filtragem, confirmando os resultados encontrados nas Figura 10 e Figura 11 com métricas de avaliação. A partir disso, somos capazes de utilizar o modelo de forma mais eficiente, iniciando o tratamento em ordem crítica minimizando a possibilidade de encontrar um aluno que não necessita de ajuda classificado como desistente.

Outra comparação interessante que nos permite analisar melhor o funcionamento do modelo, em especial o atributo critical_rate, é a evolução da distribuição crítica dos alunos selecionados, como ilustra a Figura 13.

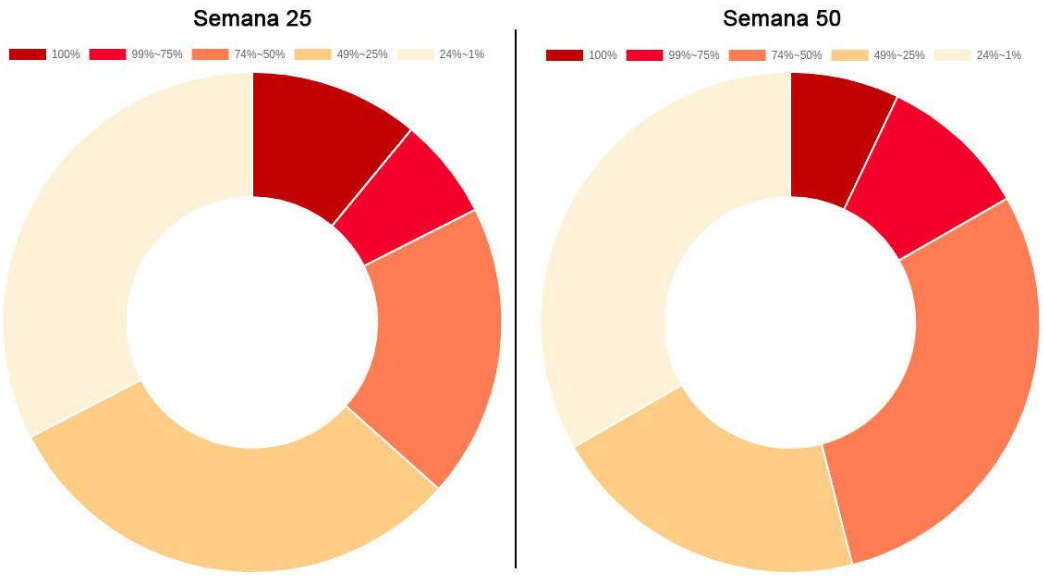

Figura 13: Evolução da distribuição crítica dos alunos selecionados.

Como podemos visualizar na Figura 13, próximo do meio do ano letivo, na semana 25, existia uma porcentagem maior de alunos classificados como 100%, e ao decorrer do ano, alguns destes alunos melhoraram sua performance e migraram de faixa para 99% a 75%, entretanto, já para os alunos na faixa de 49% a 25%, pioraram sua performance, sendo classificados mais vezes como desistentes, resultando no aumento da faixa de 74% a 50%. Este tipo de gráfico também pode ser proveitoso em comparações entre escolas ou cursos, visto que conseguem trazer de forma unificada o status da instituição, análise que é possível explorar melhor com a aplicação Web, descrita a seguir.

4.2. Aplicação Web

Buscando proporcionar aos usuários da plataforma uma visão geral dos conhecimentos produzidos pelo modelo de forma ágil, personalizamos a página inicial, após o login no sistema, de acordo com cada nível de acesso, apresentando elementos não só a respeito da classificação da semana atual, mas também sobre a evolução da classificação, com gráficos e tabelas expondo a trajetória do modelo até aquele ponto.

Inicialmente considerando o nível mais alto de acesso (*master*), na página inicial, serão apresentados gráficos de comparações entre as escolas, informações sobre a semana atual e dados técnicos do modelo, sendo ilustrado pelas imagens a seguir.

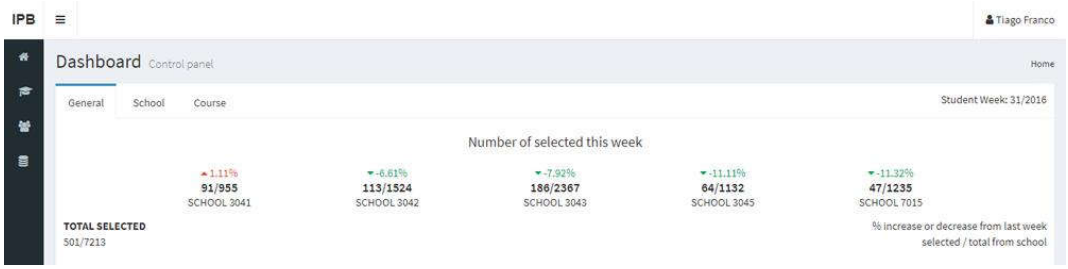

Figura 14: Página inicial para o nível *master* parte 1.

Como conseguimos perceber na Figura 14, no canto direito superior é exibido o nome do usuário que entrou no sistema e clicando nele é possível sair do sistema ou trocar a senha do usuário. Já na lateral esquerda é possível visualizar os ícones que levam as outras páginas que serão descritas mais à frente. Dentro do container principal da página no canto direito superior é mostrado a semana corrente do ano letivo e ao lado esquerdo as abas disponíveis para este nível de acesso. O primeiro elemento de comparação é relacionado a quantidade de alunos selecionados por escola, juntamente com o total de alunos matriculados na mesma e a variação dos selecionados em relação à semana anterior em forma de porcentagem.

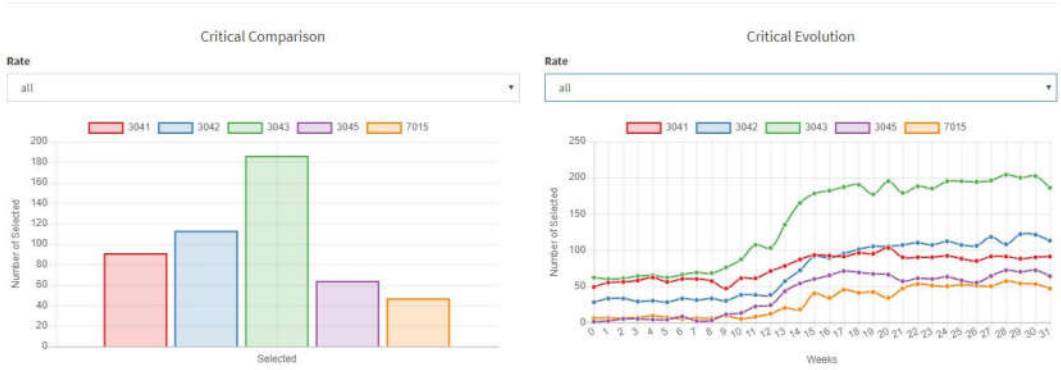

Figura 15: Página inicial para nível *master* parte 2.

Seguindo com o conteúdo da página inicial, são apresentados dois gráficos para comparação entre as escolas sobre o volume de alunos selecionados. Como podemos visualizar na Figura 15, o gráfico do lado esquerdo é referente ao volume de alunos selecionados na semana atual por escola e o segundo é referente a evolução semanal destes selecionados por escola.

Como também podemos notar na Figura 15, ambos os gráficos possuem um campo de seleção chamado *rate*, no qual consegue modificar o gráfico filtrando os alunos por setores do atributo critical_rate, sendo possível selecionar os seguintes setores em porcentagem [100, 1], [100], [99, 75], [74, 50], [49, 25], [24, 1]. Este campo de seleção também é presente em outros elementos que serão apresentados a seguir exatamente com a mesma função.

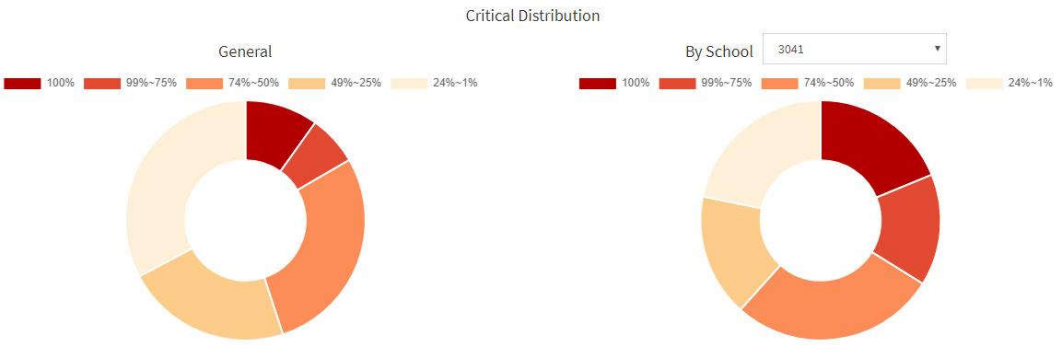

Figura 16: Página inicial para nível *master* parte 3.

Semelhantes aos gráficos da Figura 13 da seção anterior, os gráficos da Figura 16 dão continuidade a página inicial, buscando proporcionar uma comparação entre as

distribuições críticas dos alunos, o gráfico do lado esquerdo é estático e traz a distribuição crítica geral da universidade na semana atual, já o gráfico do lado direito exibe a distribuição crítica de uma escola específica e pode ser trocado para outra escola a partir do campo de seleção acima.

Levando em consideração que os usuários da aplicação Web não possuem como pré-requisito conhecimentos na área de *machine learning*, não faz sentido aprofundar os detalhes técnicos do modelo; sendo assim, os últimos elementos da página inicial são os únicos com este viés, apresentando uma comparação entre os selecionados e selecionados corretamente do ano anterior com os selecionados do ano atual e um gráfico da evolução das métricas de avaliação do ano anterior no mesmo período. Como não possuímos a informação de quais alunos desistiram ou não, não conseguimos calcular as métricas de avaliação para o ano atual. A Figura 17 ilustra os elementos discutidos acima e encerra a página inicial para o nível de acesso *master*.

Figura 17: Página inicial para nível *master* parte 4.

O próximo nível de acesso a ser descrito é o *coordinator*, nível no qual consegue visualizar as informações produzidas pelo modelo com restrição de apenas uma determinada escola associada a ele no cadastro do usuário. A página inicial para este nível de acesso possui menos componentes do que o nível *master*, são exibidos dois gráficos sobre a situação dos desistentes em contexto geral da escola e uma tabela com informações comparativas entre os cursos, como ilustra as figuras a seguir.

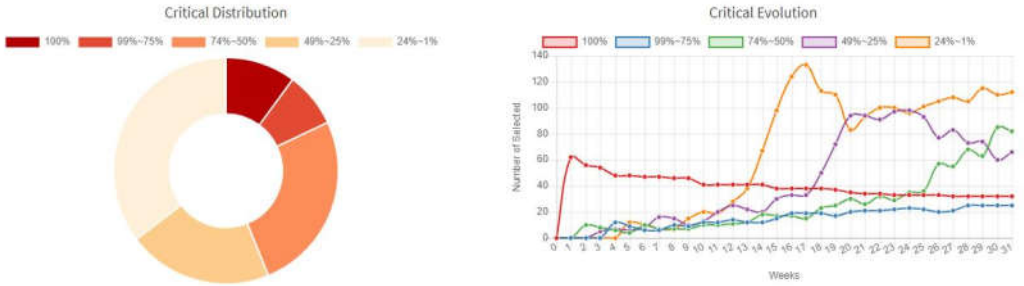

Figura 18: Página inicial para nível *coordinator* parte 1.

Como podemos notar na Figura 18, o gráfico da esquerda é o mesmo gráfico da direita da Figura 16, entretanto este gráfico é estático e contém apenas a distribuição da escola em questão na semana atual. Já do lado direito da Figura 18, podemos visualizar o gráfico da evolução da quantidade de selecionados distribuídos por setores do atributo critical_rate, semelhante ao campo de seleção *rate*, descrito anteriormente. Estes são os elementos de contexto geral sobre as escolas para o nível de acesso *coordinator*, com isso, espera-se que os usuários deste nível estejam aptos analisar e acompanhar as oscilações entre os setores que representam melhorias ou agravamento da situação.

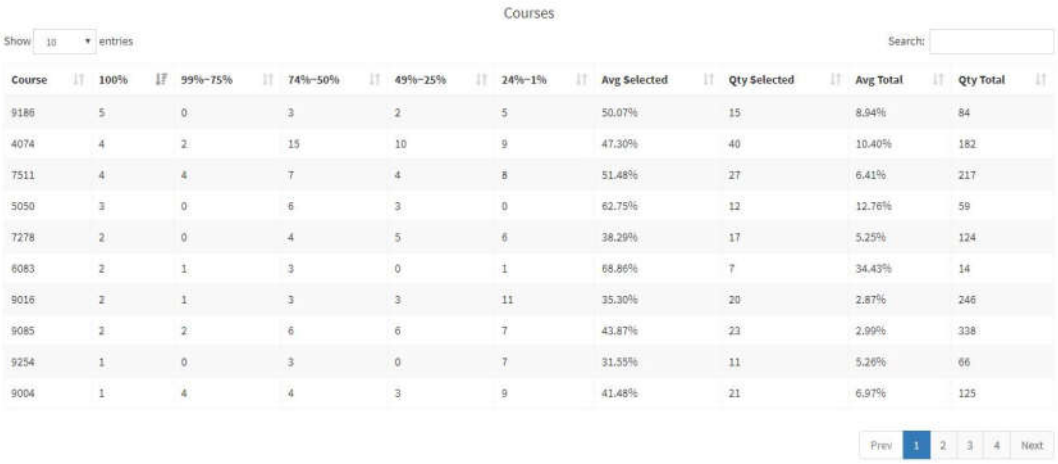

Figura 19: Página inicial para nível *coordinator* parte 2.

A Figura 19 ilustra o último elemento da página inicial para o nível *coordinator*, uma tabela de comparação entre os cursos de uma determinada escola. Esta tabela possui os seguintes campos: porcentagem média dos alunos selecionados, porcentagem média

considerando todos os alunos, quantidade de alunos selecionados, quantidade de alunos total no curso e mais 5 campos sobre a quantidade de alunos distribuídos pelos setores do atributo critical_rate. A tabela também conta com uma busca e um filtro por quantidade de cursos exibidos, além da possibilidade de ordenar os cursos por qualquer campo, possibilitando uma análise mais rápida.

Por fim, o nível de acesso *administrator*, no qual é permitido a visualização de informações particulares de apenas um curso associado ao usuário. A página inicial para este nível é bastante semelhante à página inicial para o nível *coordinator*, iniciando com os mesmos gráficos da Figura 18, porém com as informações do curso associado ao invés da escola como ilustrado na figura. Tais gráficos seguem a mesma ideia de tornar o usuário apto a identificar aumento ou diminuição de alunos selecionados em contexto geral do curso e analisar mudanças nos setores do atributo critital_rate.

Nesta página também possui uma tabela de comparação como último componente, porém os dados inseridos são os alunos matriculados no curso em questão e seus campos são apenas um resumo das informações que possuímos deles, sendo eles: média de notas, presenças em sala de aula, quantidade de matérias estudando no período atual, quantidade de matérias aprovadas e reprovadas, quantidade de anos cursados, critical_rate e last_predicted (predição da última semana). Esta tabela também conta com os mesmos elementos de suporte a análise da Figura 19 (busca, ordenação, etc..), porém com a adição de um botão a cada linha, no qual possui a função de abrir a página com as informações completas que a aplicação possui sobre um aluno específico. Para sintetizar melhor esta explicação a Figura 20 ilustra um exemplo de tal tabela.

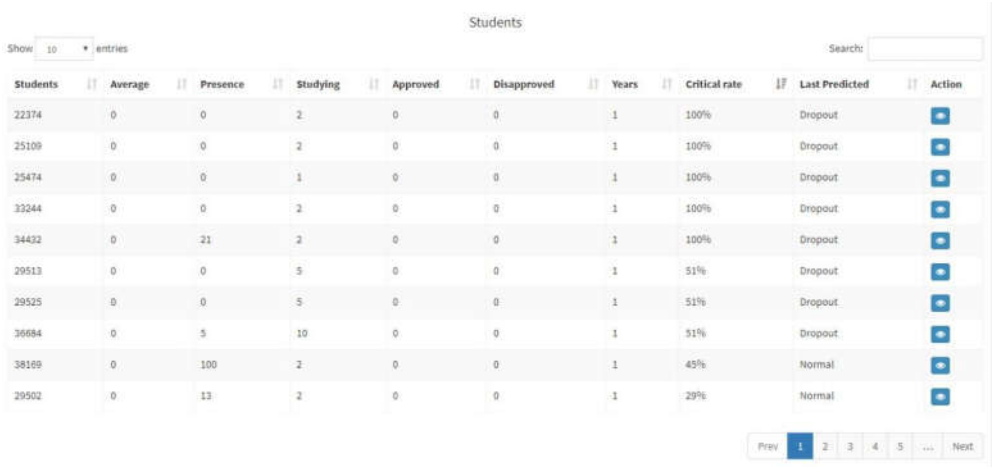

Figura 20: Página inicial para nível *administrator*.

A próxima página a ser exibida é referente a visualização individual do aluno, mencionada no parágrafo anterior. Todos os níveis de acesso conseguem entrar nesta página, entretanto só podem visualizar os alunos associados a ele; sendo assim, para nível *coordinator* todos os alunos de uma escola específica, para nível *administrator* todos os alunos de um curso específico e para *master* todos os alunos são liberados. Esta página foi elaborada para se assemelhar a um relatório, listando todas as informações colidas sobre o aluno em evidência através de elementos com diferentes cores e tamanhos, nos quais buscam enaltecer os dados mais relevantes, possibilitando uma análise mais ágil sem perder informação. As figuras a seguir ilustram estes elementos.

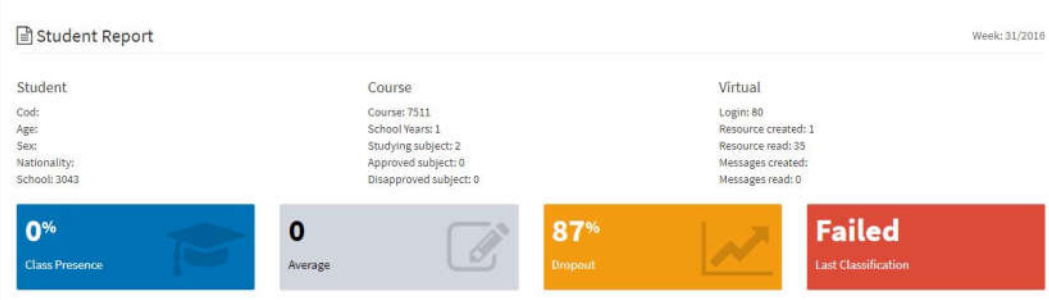

Figura 21: Página individual do aluno parte 1.

Como podemos perceber a Figura 21 ilustra a primeira parte da página individual do aluno e inicia dividindo as primeiras informações do *container* principal em três conjuntos, nos quais representam diferentes grupos de dados; sendo o primeiro responsável pelos dados demográficos, o segundo referente a informações sobre a situação do aluno no curso e o último referente a informações sobre o uso o Sakai. Os quatro elementos maiores e coloridos apresentam as informações que consideramos mais relevantes no contexto de abandono escolar, tratando-se da presença em sala de aula, média escolar e os dois atributos criados, critical_rate e a última classificação.

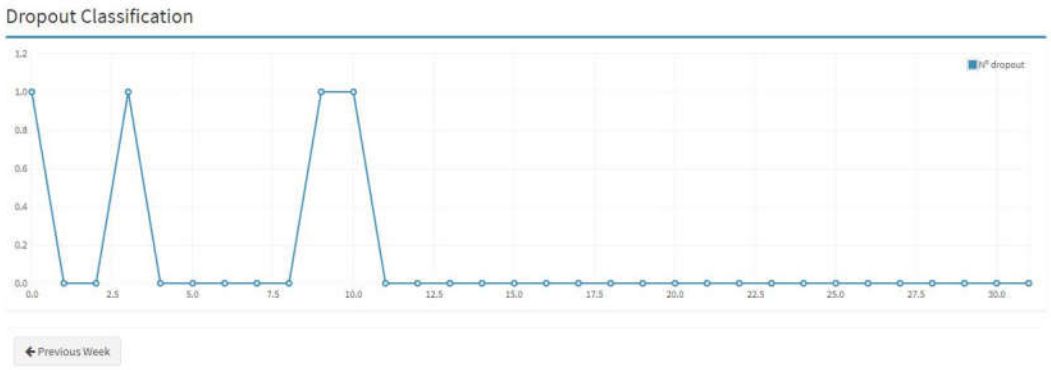

Figura 22: Página individual do aluno parte 2.

Na Figura 22 conseguimos visualizar o gráfico da trajetória das classificações ao longo das semanas de um aluno, como nossa classificação é binária, o gráfico permite apenas dois valores 0 e 1, representando respectivamente desistente e não desistente. A página individual do aluno também permite visualizar as informações das semanas anteriores, clicando no botão encontrado no canto esquerdo inferior da Figura 22.

Para auxiliar a auditoria da ferramenta ETL, desenvolvemos uma página com informações das coletas efetuadas, como ilustra na Figura 23.

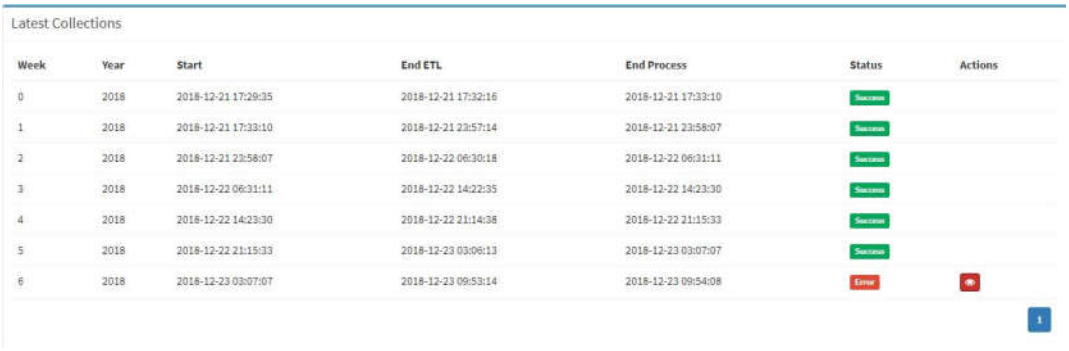

Figura 23: Página de coletas.

Como podemos notar na Figura 23 a página de coleta apenas possui um elemento, uma tabela da listagem das coletas efetuadas, contando com os seguintes campos: semana e ano da coleta, horário do início e término da extração, horário do término da classificação, status da coleta e ações. O último campo “ações” só é visível quando o status da coleta não teve sucesso, possibilitando visualizar a mensagem de erro retornada pela ferramenta

ETL. Procurando proporcionar uma possível análise sobre o desempenho da extração, os horários são armazenamos com seus segundos e milésimos.

A aplicação Web conta com mais duas páginas, sendo a primeira uma página de usuários, na qual possibilita cadastrar, editar e deletar os usuários e seus níveis de acesso, e segunda uma página de login, possibilitando o acesso a aplicação.

Capítulo 5 Conclusões

Neste capítulo será apresentada as conclusões do estudo realizado nos capítulos anteriores, mencionando os elementos que dão importância ao trabalho e por fim a descrição dos trabalhos futuros sugeridos.

5.1. Considerações finais

Neste trabalho foi apresentado um modelo para classificação prévia dos alunos em risco de abandono escolar, detalhando em especial a trajetória do desenvolvimento, que devido à complexidade gerada pelo volume e variedade de tipos de dados que a Big Data do IPB trouxe, foram exigidos mais passos até o modelo final, sendo analisado inicialmente cada banco de dados separadamente e depois do tratamento reagrupando-os como fonte de entrada para os algoritmos de machine learning. Cumprindo com os objetivos deste estudo, também foram apresentados uma aplicação Web para visualização das informações produzidas com uma interface amigável, que proporciona uma análise mais fina, e uma ferramenta de extração de dados que possibilita o funcionando deste conjunto.

Como é possível visualizar na análise apresentada no capítulo anterior, que o modelo proposto demonstra ser uma boa solução para as instituições identificarem os alunos em risco de abandono. A precisão do modelo não é muito alta, porém seu déficit é compensado pela alta sensibilidade, conseguindo evitar a classificação de alunos não desistentes como desistentes, minimizando o uso de recursos erroneamente e focando nos casos mais críticos.

O atributo criado critical_rate foi uma das peças principais para este trabalho, reduzindo o problema da trajetória do aluno, criando mais consistência no modelo e tornando-se um

índice de ordem crítica crucial na apresentação dos dados, além disso acreditamos que ainda pode ser mais explorado.

É verdade que foi fornecido uma boa quantidade de dados pelo IPB, entretanto se seguirmos a regra de 4 anos de treinamento, precisaríamos de no mínimo 13 anos para verificar de forma mais segura todos os impactos deste atributo. Neste trabalho tínhamos apenas 7 anos para treino, sendo assim, conseguimos criar este atributo e aplicá-lo ao ano de 2016, porém os anos que foram utilizados como treinamento, não foram treinados com os novos atributos, eles foram calculados após a classificação, deixando a dúvida de como o atributo critical_rate irá se comportar ao longo dos anos. A tendência deste atributo é ir se ajustando ao decorrer dos anos, pois ele não está ligado apenas a classificações anteriores dos alunos, mas também a todas as classificações já feitas por este modelo, aprendendo anualmente a ficar mais preciso.

O modelo também apresenta-se como uma boa solução devido ao seu baixo custo de implementação, como já descrito no referencial teórico deste trabalho, as instituições de ensino superior, normalmente já possuem setores e programas especializados a ajudar os alunos em sua vida acadêmica, porém com apenas um servidor para coleta e aplicação do modelo semanalmente, estes serviços podem ser impulsionados, como no caso de estudo, se tivesse sido aplicado em 2016 no IPB, conseguiríamos encaminhar mais de 350 alunos corretamente a partir da semana 30, como ilustra a Figura 12.

Os resultados expostos anteriormente a respeito da aplicação Web demonstram que é possível extrair e apresentar de forma eficiente informações do modelo com diferentes níveis de acesso, possibilitando não só a identificação prévia dos alunos desistentes, mas também o acompanhamento semanal da instituição como um todo, viabilizando diversas análises comparativas entre escolas e cursos através de uma interface de boa aparência com gráficos e tabelas interativos com o usuário.

O presente estudo descrito, em especial os fragmentos do desenvolvimento e resultados obtidos pelo modelo proposto, conquistaram uma vaga de apresentação oral com o artigo titulado de “Model for the identification of students at risk of dropout using big data analytics” na 13th annual International Technology, Education and Development Conference (INTED 2019) em Valência na Espanha, evento no qual possui grande prestígio internacional, sendo indexado desde 2010 pelo International Scientific Indexing (ISI) Web of Science e com importantes patrocinadores como Oracle Academy.

5.2. Trabalhos futuros

A respeito do modelo proposto, para trabalhos futuros, propomos a reformulação do modelo utilizando dados providos de outros sistemas ou como já mencionado no pré-processamento (seção 3.3), produzir atributos calculados com os dados já existentes, como a quantidade de horas dentro do sistema, tempo médio para entrega de trabalhos, entre outros. Outra sugestão de continuidade para este modelo, seria explorar outros algoritmos e técnicas de machine learning, assim como a utilização de mais métricas de desempenho ou combinações de algoritmos.

Neste trabalho não nos aprofundamos em construir elementos de auditoria da ferramenta de visualização do modelo e é necessário configurações manuais para o funcionamento completo de todo o estudo; sendo assim, sugerimos como trabalhos futuros, desenvolver uma interface que possibilite uma melhor auditoria, como estatísticas sobre as extrações, envio de e-mail quando algum erro acontece, ações na aplicação web que possibilite consertar pequenos problemas, como refazer uma coleta, excluir algum dado posto erroneamente, entre outros.

Um trabalho futuro mais ambicioso, seria verificar de forma mais realística os impactos que este estudo pode causar em uma universidade, sendo necessário aplicar este estudo durante alguns anos e efetuar análises comparativas entre o antes e depois do uso do modelo, justificando quais foram as diferenças para a instituição e a relevância sobre o estudo do abandono escolar.

Bibliografia

[1] A. Oussous, F. Benjelloun, A. Ait Lahcen e S. Belfkih, "Big Data technologies: A survey," *Journal of King Saud University - Computer and Information Sciences,* nº 30(p4), pp. 431-448, 2018.

[2] A. Gandomi e M. Haider, "Beyond the hype: Big data concepts, methods, and analytics," *International Journal of Information Management,* 2015.

[3] A. F. C. Santos, Í. P. Teles, O. M. P. Siqueira e A. A. de Oliveira, "Big data: A systematic review," em *Advances in Intelligent Systems and Computing*, 2018.

[4] P. C. Zikopoulos e C. Eaton, Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data, McGraw-Hill Education, 2011.

[5] T. Oladipupo Ayodele, "Introduction to Machine Learning 1 X Introduction to Machine Learning".

[6] M. Jordan, J. Kleinberg e B. Schölkopf, "Pattern Recognition and Machine Learning".

[7] T. Oladipupo, "Types of Machine Learning Algorithms," em *New Advances in Machine Learning*, 2010.

[8] I. Filipa e S. Martins, "MACHINE LEARNING ALGORITHMS TO PREDICT BLOOD-BRAIN BARRIER PERMEABILITY OF DRUG MOLECULES," 2011.

[9] E. Costa, R. S. Baker, L. Amorim, J. Magalhães e T. Marinho, "Mineração de Dados Educacionais: Conceitos, Técnicas, Ferramentas e Aplicações," pp. 1-29, 2012.

[10] J. Han, M. Kamber e J. Pei, "Data Mining. Concepts and Techniques, 3rd Edition (The Morgan Kaufmann Series in Data Management Systems)," 2011.

[11] J. R. Quinlan, "Induction of Decision Trees," *Machine Learning,* vol. 1, nº 1, pp. 81-106, 1986.

[12] S. L. Salzberg, "C4.5: Programs for Machine Learning by J. Ross Quinlan. Morgan Kaufmann Publishers, Inc., 1993," *Machine Learning,* vol. 16, nº 3, pp. 235-240, 1994.

[13] J. H. Friedman, R. A. Olshen, C. J. Stone e L. . Breiman, Classification and regression trees, ed., vol. , , : Wadsworth & Brooks/Cole Advanced Books & Software, 1984, p. .

[14] L. Breiman, "Random Forests," 2001.

[15] J. M. Barreto, "Indroduçàò as Redes Neurais Artificiais".

[16] D. Powers, "Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation," *Mach. Learn. Technol,* vol. 2.

[17] Y. Altujjar, W. Altamimi, I. Al-Turaiki e M. Al-Razgan, "Predicting Critical Courses Affecting Students Performance: A Case Study," em *Procedia Computer Science*, 2016.

[18] M. PhridviRaj e C. GuruRao, "Data Mining – Past, Present and Future – A Typical Survey on Data Streams," *Procedia Technology,* 2014.

[19] D. Asst Professor e A. Kamath Asst Professor, "IJERMT All Rights Reserved Page | 198," 2017.

[20] A. Sharma, R. Sharma, V. K. Sharma e V. Shrivatava, "Application of Data Mining-A Survey Paper".

[21] I. E. D. M. Society, "Educational Data Mining," [Online]. Available: http://educationaldatamining.org/. [Acesso em 08 Janeiro 2019].

[22] R. S. J. D. Baker e K. Yacef, "The State of Educational Data Mining in 2009: A Review and Future Visions".

[23] R. Shaun, J. De Baker, S. Isotani, A. Maria e J. Baker De Carvalho, "Mineração de Dados Educacionais: Oportunidades para o Brasil".

[24] A. M. Shahiri, W. Husain e N. A. Rashid, "A Review on Predicting Student's Performance Using Data Mining Techniques," em *Procedia Computer Science*, 2015.

[25] C. Romero e S. Ventura, *Educational data mining: A review of the state of the art,* 2010.

[26] A. Amal S e A. Ahmad A, "Big Data and Learning Analytics in Higher Education," *IEEE Conference on Big Data and Analytics ,* 2017.

[27] A. Sharma e S. Vatta, "Role of Learning Management Systems in Education," 2013.

[28] R. K. Ellis, "Learning Management Systems," 2009.

[29] A. Foundation, "Sakai Project," [Online]. Available: https://sakaiproject.org/. [Acesso em 08 Janeiro 2019].

[30] D. M. P. A. Ditutala, *ABANDONO ESCOLAR NO ENSINO SUPERIOR: Estudo de Caso do Instituto Superior Politécnico Metropolitano de Angola,* 2017.

[31] A. F. Costa e J. T. Lopes, "OS ESTUDANTES E OS SEUS TRAJECTOS NO ENSINO SUPERIOR: Sucesso e Insucesso, Factores e Processos,," 2008.

[32] S. M. G. Ferreira, *Estudo sobre o abandono escolar nos cursos do 1.º e 2.º ciclos de estudos do Instituto Politécnico de Bragança,* 2017.

[33] Q. J, "Drop-out and Completion in Higher Education in Europe: among students from under-represented groups," 2013.

[34] S. J. Rigo, S. C. Cazella e W. Cambruzzi, "Minerando Dados Educacionais com foco na evasão escolar: oportunidades, desafios e necessidades".

[35] MEC, ANDIFES, ABRUEM e SESu, *Diplomação, Retenção e Evasão nos Cursos de Graduação em Instituições de Ensino Superior Públicas,* 1996.

[36] N. Kondo, M. Okubo e T. Hatanaka, "Early Detection of At-Risk Students Using Machine Learning Based on LMS Log Data," em *Proceedings - 2017 6th IIAI International Congress on Advanced Applied Informatics, IIAI-AAI 2017*, 2017.

[37] K. S. Na e Z. Tasir, "Identifying at-risk students in online learning by analysing learning behaviour: A systematic review," em *2017 IEEE Conference on Big Data and Analytics*, 2017.

[38] M. S. Halawa, M. E. Shehab e E. M. Hamed, "Predicting student personality based on a data-driven model from student behavior on LMS and social networks," em *2015 5th International Conference on Digital Information Processing and Communications, ICDIPC 2015*, 2015.

[39] Q. A. Al-Radaideh e E. Al Nagi, "Using Data Mining Techniques to Build a Classification Model for Predicting Employees Performance," 2012.

[40] "The Python Wiki," [Online]. Available: https://wiki.python.org/moin. [Acesso em 22 Janeiro 2019].

[41] R. S., Python Machine Learning, 2015.

[42] F. Pedregosa, G. Varoquaux e a. et, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research,* 2011.

[43] G. M., Flask Web Development: Developing Web Applications with Python, 2014.

[44] R. A., "Flask (A Python Microframework)," [Online]. Available: http://flask.pocoo.org. [Acesso em 23 Janeiro 2019].

[45] I. MongoDB, "MongoDB," [Online]. Available: https://www.mongodb.com/what-is-mongodb. [Acesso em 23 Janeiro 2019].

[46] J. Trujillo, "A UML Based Approach for Modeling ETL Processes in Data Warehouses," *Conceptual Modeling - ER 2003,* pp. 307-320, 2003.

[47] V. Silva, "Revisão sistemática da evolução MVC na base ACM," em *15º Concurso de Trabajos Estudiantiles*, 2012.