INSTITUTO FEDERAL DE GOIÁS

ANDRÉ OLIVEIRA GOMES
JOSÉ PEDRO DE ASSIS A. JUNIOR

UMA ABORDAGEM DE APRENDIZADO DE MÁQUINA PARA A PREVISÃO DO DESEMPENHO ESCOLAR DE ALUNOS DO ENSINO MÉDIO

JATAÍ-GOIÁS
2017

ANDRÉ OLIVEIRA GOMES

JOSÉ PEDRO DE ASSIS A. JUNIOR

UMA ABORDAGEM DE APRENDIZADO DE MÁQUINA PARA A PREVISÃO DO DESEMPENHO ESCOLAR DE ALUNOS DO ENSINO MÉDIO

Trabalho de conclusão de Curso apresentado ao curso de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Goiás como requisito à obtenção do grau de Tecnólogo em Análise e Desenvolvimento de Sistemas

Orientador: Prof. Doutor Gustavo de Assis Costa

JATAI-GOIÁS

2017

Dados Internacionais de Catalogação na Publicação na (CIP)

GOM/abo	Gomes, André Oliveira. Uma abordagem de aprendizado de máquina para a previsão do desempenho escolar de alunos do ensino médio / André Oliveira Gomes; José Pedro de Assis A. Júnior. -- Jataí: IFG, Coordenação dos cursos de Informática ï Tecnologia em Análise e Desenvolvimento de Sistemas, 2018. 69; il. Orientador: Prof. Dr. Gustavo de Assis Costa. Bibliografias. Apêndices. 1. Inteligência artificial. 2. Aprendizado de máquina. 3. Regressão linear. 4. Evasão escolar. I. Arantes Júnior, José Pedro de Assis. II. Costa, Gustavo de Assis. III. IFG, Câmpus Jataí. IV. Título. CDD 006.3

Ficha catalográfica elaborada pela Seção Téc.: Aquisição e Tratamento da Informação.
Bibliotecária ï Rosy Cristina Oliveira Barbosa ï CRB 1/2380 ï Campus Jataí. Cód. F040/18.

André Oliveira Gomes

José Pedro de Assis Arantes Júnior

Uma Abordagem de Aprendizado de Máquina para o Auxílio no Diagnóstico de Evasão Escolar em Turmas do Ensino Médio

Esta Monografia foi julgada adequada para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas e aprovada em sua forma final pela Banca Examinadora.

Trabalho aprovado em 06 de dezembro de 2017.

BANCA EXAMINADORA

Prof. Dr. Gustavo de Assis Costa
(Presidente da Banca/Orientador)
Instituto Federal de Educação, Ciência e Tecnologia de Goiás

Prof. Esp. Danillo Vaz Borges de Assis
(Membro Interno)
Instituto Federal de Educação, Ciência e Tecnologia de Goiás

Prof. Me. Roney Lopes Lima
(Membro Interno)
Instituto Federal de Educação, Ciência e Tecnologia de Goiás

RESUMO

A evasão escolar é um dos grandes problemas enfrentados pelos sistemas educacionais de vários países mundo afora. Existem vários estudos que avaliam possíveis causas para esta situação, como perfil socioeconômico do aluno, características e estrutura da escola e também da região onde a mesma está inserida, perfil dos docentes, além de várias outras. Contudo, grande parte destas informações não é conhecida ou são imprecisas por uma série de motivos que extrapolam a capacidade dos sistemas de gestão educacional. Desta forma, a necessidade do uso de sistemas que automatizem a análise das possíveis causas é de extrema importância para o processo de tomada de decisões a serem tomadas pelos gestores. A proposta deste trabalho é aplicar técnicas de aprendizado de máquina para realizar este tipo de análise. A técnica de Regressão Linear foi a escolhida pois a mesma após analise previamente feita foi a que melhor atendeu ao propósito, apresentando índices satisfatórios como por exemplo o método avaliativo RSS (Soma dos Quadrados dos Resíduos) com valor de 1.22 e MSE (Erro quadrático médio) com valor de 239.19. Com o advento de técnicas do aprendizado de máquina é possível a exploração otimizada de bases de dados, o que permite o surgimento de novas perspectivas e informações que poderão auxiliar no processo de tomada de decisão. Desta forma, é possível prever com relativa exatidão a possibilidade de evasão de alunos do ensino médio, tendo como base fatores determinantes encontrados em dados dos anos acadêmicos anteriores como informação socioeconômica familiar. Para a realização dos experimentos utilizamos uma base de dados obtida a partir da realidade de turmas do ensino médio em escolas de Portugal.

Palavras chave (Inteligência Artificial, Aprendizado de Máquina, Regressão Linear, Evasão escolar)

ABASTRACT

School dropout is one of the major problems faced by education systems in many countries around the world. There are several studies that evaluate possible causes for this situation, such as socio-economic profile of the student, characteristics and structure of the school and also the region where it is inserted, profile of teachers, as well as several others. However, much of this information is imprecise or not known for a number of reasons that go beyond the capacity of educational management systems. Therefore, the need to use systems that automate the analysis of possible causes is extremely important for the decision-making process to be taken by managers. The purpose of this paper is to apply machine learning techniques to perform this type of analysis. The advent of machine learning techniques makes the optimal exploitation of databases possible, allowing the emergence of new perspectives and information that may aid in the decision-making process. Thus, it is possible to predict possible high school dropout with relative accuracy, based on determinant factors found in data from previous academic years as family socioeconomic information. For the accomplishment of the experiments we used a database obtained from the reality of high school classes of schools in Portugal.

LISTA DE ILUSTRAÇÕES

Figura 1 - Evasão escolar no Brasil 10
Figura 2 - Tipos de Aprendizado de Máquina 19
Figura 3 - Exemplo KNN 21
Figura 4 - Exemplo de Árvore de decisão (Empréstimo bancário) 23
Figura 5 - Exemplo Multi-Layer Perceptron 25
Figura 6 - Diagrama de dispersão 26
Figura 7 - Gráfico de dispersão (Previsão de preço de casas) 28
Figura 8 - Exemplo de Cluster 30
Figura 9 - Etapas K-means 32
Figura 10 - Formas para medição da distância de Clusters 33
Figura 11 - Exemplo de Self Organizing map 35
Figura 12 - por sexo e notas por sexo 39
Figura 13 - Distribuição das notas 40
Figura 14 - Aulas extras pagas e Atividades extracurriculares 40
Figura 15 - Acesso à internet - Relacionamento amoroso 41
Figura 16 - Apoio Educacional Familiar - Apoio Educacional Escolar extra 42
Figura 17 - Tamanho da Família - Responsável legal do aluno 42
Figura 18 - Interesse em fazer ensino superior 43
Figura 19 - Tipo de endereço (Urbano ou Rural) 44
Figura 20 - Trecho de código Regressão com Scikit-Learn 49
Figura 21 - Predição com apenas a "G2" 51
Figura 22 - Predição com "G1" e "G2" 56
Figura 23 - Predição com todos os atributos 57

1 INTRODUÇÃO 9
1.1 JUSTIFICATIVA 12
1.2 OBJETIVOS 14
1.2.1 Objetivo Geral 14
1.2.2 Objetivos específicos 14
2 TRABALHOS RELACIONADOS 15
3 FUNDAMENTAÇÃOTEÓRICA 18
3.1 O QUE É MACHINELEARNING? 18
3.2 MACHINE LEARNING E DATAMINING 18
3.3 APRENDIZADO SUPERVISIONADO E NÃO SUPERVISIONADA 19
3.3.1 Aprendizado Supervisionado 20
3.3.1.1 Classificação 20
3.3.1.2 KNN 20
3.3.1.3 Árvore de decisão 22
3.3.1.4 Naive bayes 23
3.3.1.5 Perceptron/Adaline 23
3.3.1.6 Multi-Player Perceptron 24
3.3.1.7 Regressão 25
3.3.1.8 Regressão Linear Simples 26
3.3.1.9 Regressão Linear Múltipla 29
3.4 APRENDIZADO NÃO SUPERVISIONADO 30
3.4.1.1 K-means 32
3.4.1.2 Algoritmos Hierárquicos 33
3.4.1.3 Self organizing maps 34
3.4.2 Aprendizado semi supervisionado 36
4 METODOLOGIA 37
4.1 BASE DE DADOS 37
4.2 ANÁLISE EXPLORATÓRIA 39
4.3 FERRAMENTAS E BIBLIOTECAS 44
5 DESENVOLVIMENTO 47
5.1 ESCOLHA DA TÉCNICA DE MACHINE LEARNING A SER UTILIZADA 47

5.2 IMPLEMENTAÇÃO 47
5.2.1 Etapas 47
5.2.2 Criação do Modelo 48
5.2.3 Testes e validações 50
5.2.3.1 Modelo com apenas o atributo “G2” 51
5.2.3.2 Modelo com os atributos “G1” e “G2” 56
5.2.3.3 Modelo utilizando todos os atributos 57
6 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS 58
7 APÊNDICE 59
7.1 APÊNDICE 1 – MODELO COM SCKIT-LEARN 59
7.2 APÊNDICE 2 - MODELOS SEM SICKIT-LEARN 61
8 REFERÊNCIAS 68

1 INTRODUÇÃO

Segundo dados publicados pelo Eurostat[1] referentes à pesquisa realizada no ano de 2016 para mapeamento do abandono escolar na União Europeia, apesar de estar em constante queda no percentual de evasão escolar, Portugal ainda apresenta índices alarmantes, sendo o quarto pior país na região no quesito abandono escolar. Cerca de 14% dos jovens entre 18 e 24 anos deixam prematuramente a educação, ficando atrás apenas de Malta (19,6), Espanha (19%) e Romênia (18,5%). Ainda se levarmos em consideração a média de abandono escolar da União Europeia, que ficou em torno de 10%, as taxas apresentadas por Portugal ainda destoam dos demais países da zona do euro.

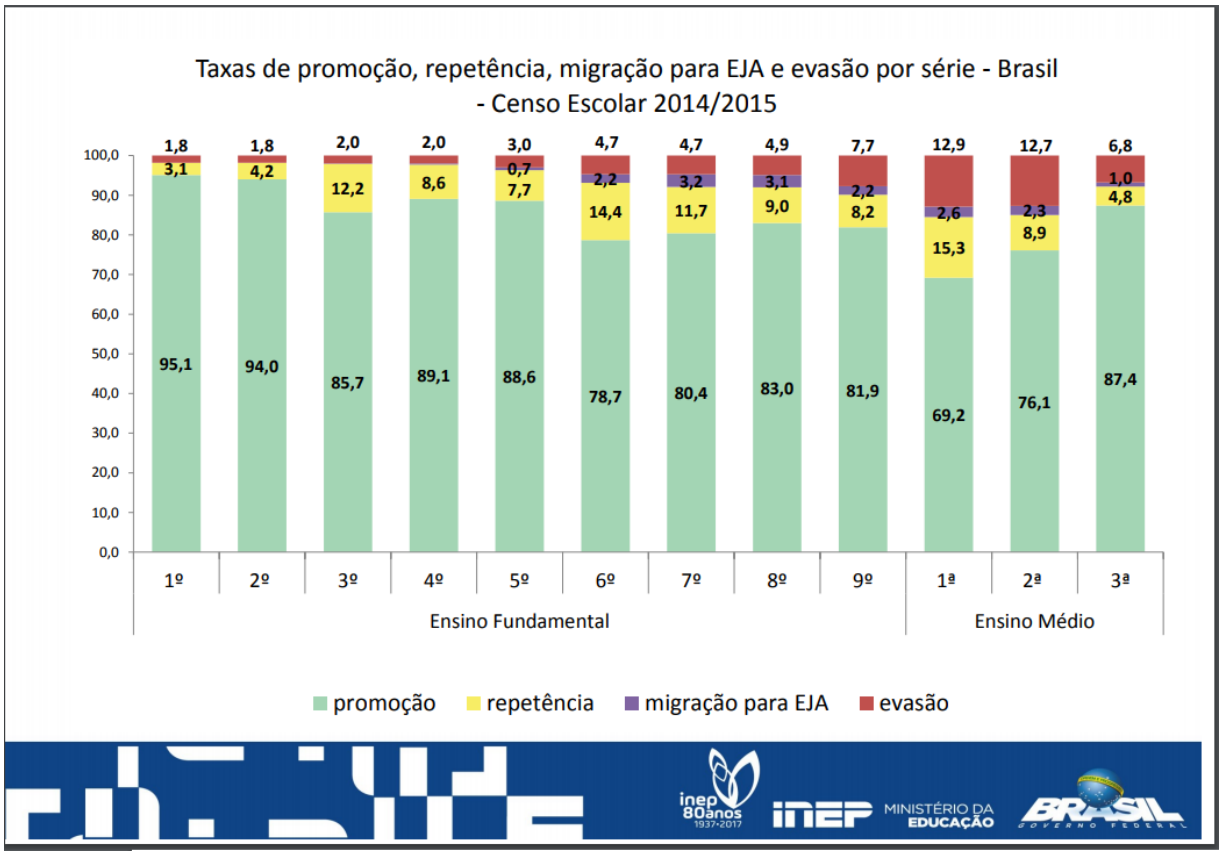
O problema da evasão escolar não é exclusividade de Portugal tornando-se uma preocupação mundial como podemos ver em pesquisa realizada pelo Inep[2] durante o período de 2007 e 2015 em todo território nacional brasileiro. Alguns dados sobre a evasão escolar foram coletados (fig. 1), nos quais fica evidente a situação preocupante sobre o assunto. Cerca de 12,9% e 12,7% dos alunos matriculados na 1ª e 2ª série do ensino médio respectivamente, evadiram da escola entre 2014 e 2015 sendo as maiores taxas de evasão durante o ciclo escolar. Ainda foi constatado que respectivamente na terceira e quarta colocação de maiores evasões estão o 9º ano do ensino fundamental e 3º ano do ensino médio respectivamente com 7,7% e6,8%.

Porém o que a pesquisa não mostra são os motivos para esta evasão e quais poderiam ser as causas que mais contribuem para tal situação. Utilizando técnicas de aprendizado de máquina é possível que se tenha uma melhor compreensão ao analisar dados de turmas anteriores. Quanto mais informações (atributos) houver sobre os alunos, como dados socioeconômicos da família e informações pessoais, bem como quanto mais registros existirem, melhor poderão ser os resultados e consequentemente a integridade e acurácia. Com isto é possível identificar os perfis dos alunos que evadem do ensino escolar, além de o sistema ter condições de prever se determinado aluno irá ou não evadir, com elevada taxa probabilística de acerto como apontado em (Nicolae-Bogdan, et. Al. 2015) e (J. V. Amorim, et. Al. 2008) auxiliando na prevenção deste problema

[1]URL: https://www.dn.pt/sociedade/interior/portugal-tem-a-quarta-taxa-de-abandono- escolar-mais-elevada-da-ue---eurostat-6244999.html

[2]URL: http://portal.inep.gov.br/artigo/-/asset_publisher/B4AQV9zFY7Bv/content/inep- divulga-dados-ineditos-sobre-fluxo-escolar-na-educacao-basica/21206

Figura 1 - Evasão escolar no Brasil



Fonte: Inep - Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira Legislação e Documentos (2016).

Com a crescente evolução do uso da tecnologia no cotidiano, novas ferramentas e soluções foram surgindo no intuito de facilitar a resolução dos problemas computacionais de pessoas e empresas, dentre estas soluções podemos destacar o aprendizado de máquina. Se durante um longo período os computadores apenas eram capazes de processar e efetuar o que lhes era instruído a fazer, novas técnicas possibilitaram que os computadores pudessem aprender a se auto programar, principal característica do campo de aprendizado de máquina (Machine Learning).

A diferença entre as máquinas e os seres humanos, está no fato de que nós gastamos parte do tempo da realização de uma tarefa tentando aperfeiçoar o processo da mesma, enquanto as máquinas somente executam o que lhes é programado, porém o aprendizado de máquina nos abre um novo leque de opções onde a máquina é capaz de se auto programar, fazendo um paralelo com o ser humano. Parte dos dados a serem processados são tratados como treinamento para que o sistema possa aprender e executar uma nova ação não

previamente programada.

O aprendizado de máquina teve seu surgimento na década de 1980 surgindo como uma das áreas da inteligência artificial (J. Russell S. and Norvig P. 1995). Porém, ao longo da década de 1980 e início da década de 1990 houve o destaque do aprendizado de máquina como disciplina independente da Inteligência artificial.

É possível fazer uso destas técnicas para a resolução de vários problemas os quais antes demandariam bastante tempo e que em alguns casos não seriam possíveis de serem realizados, devido à dificuldade em trabalhar com grande quantidade de dados.

Atualmente dispomos de poucos recursos alternativos para previsão de notas, pois ficamos reféns da análise manual onde mesmo podendo chegar a padrões, o tempo para que uma análise simples de apenas um estudante seja concluído é demasiadamente alto, tornando inviável a previsão em tempo hábil.

1.1 JUSTIFICATIVA

Em se tratando de educação, um dos problemas mais agravantes e onerosos é o da evasão escolar. Segundo levantamento[3] feito pelo ex-secretário estadual de educação do estado do Rio de Janeiro, Cláudio Mendonça Prejuízo com evasão escolar pode chegar a R$ 400 milhões por ano. Este valor seria suficiente para construir 266 escolas ou implantar 10 mil laboratórios de informática, vale salientar que estes números refletem uma taxa de 14% de evasão, em apenas um estado da união. Quando levado em consideração todo o âmbito nacional, estes números são ainda mais alarmantes. Pouquíssimos são os recursos utilizados na prevenção. Também não é difícil notar que soluções convencionais não são suficientes, pois somente atendem uma pequena quantidade de instituições, e dependendo exclusivamente do controle humano, que está sujeito a falhas, seja pela sobrecarga de funções, seja pela quantidade grande de alunos a serem observados.

A discussão em torno do impacto causado pela evasão escolar é de suma importância para a sociedade como um todo, pois as consequências deste fenômeno são sentidas em todo âmbito social, desde o prejuízo financeiro direto que é sentido em virtude do desperdício do dinheiro público investido, dificultando políticas de formação do cidadão, impactando indiretamente também no processo de promoção da qualificação profissional e também para o surgimento de problemas sociais. Portanto, o uso de soluções computacionais pode se tornar um grande aliado no combate à evasão, pois todo o processo de análise do perfil dos alunos é feito de forma automática pelo sistema, o que permite acompanhar todos os alunos e, com pequenas alterações, pode-se usar a mesma solução em mais de uma instituição. Com isto, governo, instituições de ensino e até mesmo a sociedade podem se beneficiar.

O uso de técnicas de aprendizado de máquina podem ofertar uma gama de ações proativas e reativas que podem ser tomadas após a análise de dados, possibilitando o melhor entendimento de quais medidas devem ser tomadas para melhoria individual do desempenho do aluno, desde atividade extraclasse até uma diferente abordagem na forma de ensino para aquele determinado aluno, pois o ajuste na metodologia de ensino para um aluno que se aproximou da média necessária é diferente do ajuste de um aluno que teve um desempenho extremamente baixo.

A luta contra a evasão e a pesquisa acadêmica de soluções e estudos nesta área é de grande benefício para o meio acadêmico, pois o número maior de produções de artigos sobre evasão pode representar o início de uma transformação que começa na academia e que pode se estender para a realidade social. Para o curso de Tecnologia em Análise e

[3] URL: https://oglobo.globo.com/sociedade/educacao/prejuizo-com-evasao-escolar- pode-chegar-r-400-milhoes-por-ano-4590772

Desenvolvimento de Sistemas, a solução proposta poderá agir como beneficiador em duas vertentes, primeiro por ser uma solução alternativa para a prevenção da evasão dos alunos, e no fato de propor uma nova abordagem de futuros trabalhos e estudos de aplicação geral.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

O objetivo deste trabalho é estudar e implementar a utilização de técnicas de aprendizado de máquina como mecanismo para análise automatizada do desempenho escolar de estudantes do ensino médio nas disciplinas de português e matemática. Identificando alunos que obterão notas abaixo da média, auxiliando o profissional da educação a adaptar a metodologia de ensino de modo unitário ao aluno além de otimizar os resultados das instituições de ensino. Vale salientar que vários são os fatores que contribuem para a evasão escolar, porém no escopo deste trabalho apenas a nota será considerada para combater a evasão.

1.2.2 Objetivos específicos

Sabe-se que certas condições e situações específicas dos alunos são determinantes para seu sucesso em termos de desempenho. Desta forma, propõe-se utilizar uma técnica específica do aprendizado de máquina, a regressão. A regressão será efetuada para a previsão com relativa exatidão o desempenho do aluno em sua nota final, bem como identificar fatores que mais influenciam nessa predição e relatar a eficiência e acurácia do modelo proposto.

2 TRABALHOS RELACIONADOS

Nesta sessão apresentaremos alguns trabalhos relacionados ao uso de aprendizado de máquina no auxílio do controle e previsão de evasão escolar, bem como as similaridades e diferenças com a solução proposta neste projeto.

Os primeiros trabalhos relacionados nesta seção apresentam soluções similares à nossa, contudo o fazem a partir de dados do ensino superior e, em sua maioria, por meio de técnicas de mineração de dados.

(Kantorski et. Al. 2016) apresentam em artigo submetido ao XXXVI Encontro nacional de engenharia de produção, uma solução genérica no intuito de previsão da evasão em cursos de graduação presenciais. A pesquisa realizada utilizou de análise dos dados dos alunos, gerando uma lista de alunos que não realizaram matrícula no período posterior ao que estão cursando. Os experimentos realizados obtiveram uma acurácia média de 96% e mais de 73% de sucesso nas suas previsões de alunos que abandonaram o curso. Para a análise destes dados os autores utilizaram da ferramenta WEKA, que é uma suíte que contém diversas técnicas de mineração de dados (Marcelo Damasceno. 2010). Os algoritmos utilizados para modelagem de dados foram J48, KNN, CART, Naive Bayes, Multilayer Perceptron e OneR. O trabalho efetuado pelos autores, diferentemente do que foi proposto por nós, teve um enfoque no ensino superior e utilizou técnicas específicas de mineração de dados.

(Mauricio et. Al. 2008) apresentaram em seu artigo “Técnicas de aprendizado de máquina aplicadas na previsão da evasão acadêmica” proposta de solução computacional para prevenção e controle da evasão acadêmica de cursos de graduação, diferentemente do trabalho aqui proposto que tem o enfoque na educação básica. Os dados foram obtidos em uma instituição de ensino superior particular do município de Campos de Goyatacazes, RJ. Novamente a aplicação escolhida para trabalhar foi o WEKA, utilizando os seguintes algoritmos: J48, Sequential Minimal Optimization(SMO) e Bayes Net. Para os testes realizados o algoritmo que apresentou maior eficácia foi o SMO. Porém, o artigo apresentado não trabalhou com os pesos dos atributos, ficando como sugestão para próximos trabalhos.

(Sandro et. Al. 2012) apresentaram em artigo focado no quarto desafio do SBC que trata do “Acesso participativo e universal do cidadão brasileiro ao conhecimento”, uma análise sobre possíveis melhorias na aplicação de mineração de dados educacionais visando à prevenção da evasão escolar. Os autores salientam a dificuldade da criação de uma aplicação genérica, pois as soluções dependem da disponibilidade de dados e estes nem sempre estão armazenados com as mesmas estruturas de atributos, fazendo com que uma única aplicação não seja capaz de ser trabalhada em todos os universos de dados. Outro ponto destacado é a parte do Pré-Processamento, pois como se trata de um artigo focado no ensino superior, existem várias possibilidades a serem consideradas como evasão, desde trancamento de

matrícula, desistência de determinada disciplina, até saída definitiva do curso. O trabalho apresentado pelos autores mostra pesquisa sobre a evasão escolar e o uso da mineração de dados como ferramenta para auxílio no combate à mesma, porém, não foi apresentado nenhuma solução desenvolvida, conforme apresentamos no presente trabalho.

Dentre os trabalhos existentes na literatura podemos destacar aqueles que mais se aproximam do que está sendo proposto. Todos estes trabalhos aplicaram diferentes técnicas para algumas das principais tarefas como predição e classificação e com exceção do primeiro a ser citado, os outros trabalhos foram propostos a partir da análise de dados de alunos do ensino médio, assim como o que está sendo proposto neste trabalho. Isto posto, podemos destacar a relevância das técnicas e à aplicação das mesmas.

(Lovenoor Auleket. Al. 2016) mostram uma perspectiva da evasão acadêmica nos Estados Unidos, onde cerca de 40% dos estudantes que começam curso de bacharelado não completam a formação dentro de 6 anos. 30% dos ingressantes não retornam após o primeiro ano, gerando um prejuízo de mais de 9 bilhões de dólares anualmente. Os dados coletados foram dos anos entre 1998 e 2006 da Universidade de Washington, no total um número de 69.116 registros foram disponibilizados. Os autores escolheram trabalhar com três algoritmos de aprendizado de máquina que foram: *regularized logistic regression, k-nearest neighbors* e *random forests*. Os três algoritmos trabalhados apresentaram resultados próximos entre si, com leve superioridade para *logistic regression*. Conforme mostram os resultados, ficou constatado que a idade do aluno, a época do ano que foi efetuada a matrícula e o curso foram os fatores de maior peso para evasão.

(Everaldo Aguiaret. Al. 2013) mostram um pouco do cenário da evasão no ensino médio nos Estados Unidos onde mais de 700 mil alunos por ano não se formam na escola secundária após 4 anos de seu ingresso na mesma. Vários índices são divulgados sobre dados de evasão escolar naquele país bem como alguns dos fatores agravantes são conhecidos, porém não existem métodos de amplo acesso para predição e prevenção. Foram utilizados dados de mais de 11 mil alunos do 6º ano até o ensino médio, alguns fatores foram identificados como responsáveis pela evasão, dentre eles o comportamento do aluno e metodologia do curso aplicado. Outro ponto abordado seria na previsão de alunos que não vão se evadir, porém precisam de apoio educacional ou social extra. Outro dado importante identificado é a quantidade de informação que é possível ter já na fase de análise, uma das informações identificadas nessa fase foi o fato de que quanto menos o aluno troca de escola ele está menos propenso a evasão. Os autores utilizaram dados desde a educação fundamental até o ensino médio.

(Nicolae-Bogdan et. Al. 2015) apresentam uma perspectiva da evasão escolar na

Dinamarca onde cerca de 14% dos alunos que iniciam no ensino médio acabam evadindo. É importante salientar que na Noruega existem 3 sistemas de ensino e a solução apresentada, focou no sistema mais utilizado, o sistema Lectio. Os algoritmos utilizados *foram Random Forest*, *Suport Vector Machine* (SVM), *Classification and Regression Tree* (CART) e *Naive Bayes*, pois foi identificado que seriam os mais eficientes perante a amostra de dados. Dentre as técnicas utilizadas a mais eficiente foi a de Random Forest com uma precisão de 93,5% seguida de respectivamente SVM, CART e por fim Naive Bayes. Os dados mais relevantes para previsão foram tamanhos da turma e da escola, ausência no último mês e a renda média por código postal.

3 FUNDAMENTAÇÃO TEÓRICA

3.1 O QUE É MACHINE LEARNING?

O aprendizado de máquinas é uma das áreas de mais rápido crescimento da Ciência da Computação (Shalev-Shwartz et. Al. 2014). Uma das definições encontradas no dicionário Aurélio para aprendizagem é: “tempo durante o qual se aprende”. Com esta definição podemos enxergar como nós humanos aprendemos durante a execução de alguma tarefa, pois faz parte do processo de efetivamente fazer algo, aprendermos como poderemos fazer de forma aprimorada e eficaz a mesma tarefa em futuras execuções. Porém as máquinas não apresentavam a mesma capacidade, pois, somente eram capazes de executar o que lhes eram programadas a fazer. Com o advento do Aprendizado de Máquina este panorama mudou, pois agora a possibilidade de os sistemas se auto programar abriu um novo leque de opções. Dentre as técnicas de Machine Learning algumas delas se baseiam justamente na capacidade biológica de aprendizagem, emulando para o campo computacional os métodos de aprendizagem biológico (Nils J. Nilsson.1998).

3.2 MACHINE LEARNING E DATAMINING

As áreas de Machine Learning (ML) e Data Mining (DM) são duas das áreas de maior crescimento no campo da tecnologia da informação nos dias atuais. Não é somente o fato de estarem em vago mercado de TI que aproxima estas duas áreas, os dois campos compartilham do uso de técnicas e conceitos como, por exemplo, a classificação e regressão, que são usadas por ambas (Cássio O. et. Al. 2009), tornando a sua separação muitas vezes uma tarefa difícil. Porém podemos destacar algumas diferenças básicas entre ML e DM. O data mining é direcionado para busca de conhecimento compreensível, ou seja, identificar informações e possibilitar inferências em quantidades massivas de dados que antes não eram possíveis de serem identificadas, enquanto que o Machine Learning visa a otimização de processos, ou seja, a partir de um treinamento do algoritmo, é possível que o próprio algoritmo consiga automatizar tarefas. Podemos afirmar que o treinamento de uma rede neural para a otimização de uma balança é parte apenas de ML e não de DM, porém existem casos que a extração de conhecimento através de uma rede neural é muito relevante para o DM. Geralmente DM é concentrado em enormes bancos de dados, pois depende de uma grande quantidade de dados para que tenha seu desempenho garantido, já o ML geralmente utiliza uma quantidade de dados menor. O DM é a parte do ML direcionada na procura de conhecimentos compreensíveis e antes não notada em grandes quantidades de exemplos

reais (Universidade estadual do Maringá, dep. De informática, algoritmos de mineração tecnologias relacionadas,1998).

3.3 APRENDIZADO SUPERVISIONADO E NÃO SUPERVISIONADA

Dentre as técnicas de aprendizado de máquina duas se destacam, o aprendizado supervisionado e não supervisionado, possuindo vários algoritmos baseados nelas, conforme pode-se ver na figura 2 e na sessão subsequente.

Figura 2 - Tipos de Aprendizado de Máquina

Fonte: Ricardo Lima Feitosa de Ávila (2014).

Tradicionalmente, os métodos de aprendizado de máquina são divididos em aprendizado supervisionado (preditivo) e não-supervisionado (descritivo) (Cássio Oliveira 2009). Porém, a divisa entre os dois métodos é extremamente sensível, pois alguns métodos preditivos podem ser descritivos e vice-versa.

3.3.1 Aprendizado Supervisionado

O aprendizado supervisionado é utilizado para fazer previsões com base em conjunto de exemplos pré-definidos, ou seja, dados anteriores são levados em consideração para novas predições, tanto os dados de entrada como de saídas são de conhecimento prévio, sendo comum o uso de algoritmos de aprendizado supervisionado para predições como, por exemplo, para sabermos se um determinado aluno irá ou não evadir de sua escola baseando-se em alunos de turmas anteriores com características parecidas. Existem várias técnicas de aprendizagem supervisionada dentre elas podemos destacas as seguintes:

3.3.1.1 Classificação

Uma das tarefas mais comuns, a Classificação, visa identificar a qual classe um determinado registro pertence. Nesta tarefa, o modelo analisa o conjunto de registros fornecidos, com cada registro já contendo a indicação à qual classe pertence, a fim de "aprender" como classificar um novo registro (Cássio Oliveira Camilo et. al. 2009).

3.3.1.2 KNN

KNN é uma das técnicas utilizadas para classificação onde o conjunto de dados é organizado em vetores n-dimensionais, cada um dos elementos deste vetor representa um ponto no espaço n-dimensões, portanto o algoritmo procura dentre os elementos de treinamento os registros mais próximos do que está sendo testado, ou seja, procura K elementos de menor distância, este K elementos são comumente chamados de K-vizinhos mais próximos, após isto o algoritmo verifica a classe mais frequente entre eles atribuindo-a ao elemento desconhecido. Algumas métricas são utilizadas para definição das distâncias entre elementos, dentre elas as mais famosas são a distância Euclidiana que é dada entre pontos x e y por:

$$d(x,y) = \sqrt{\sqrt{(\mathrm{x1} - \mathrm{y1})^2 + (\mathrm{x2} - \mathrm{y2})^2 + .. + (\mathrm{xn} - \mathrm{yn})^2}}$$

E a distância Manhattan que é dada entre os pontos x e ypor:

$$d(x,y)=|x_1-y_1|+|x_2-y_2|+..+|x_n-y_n|$$

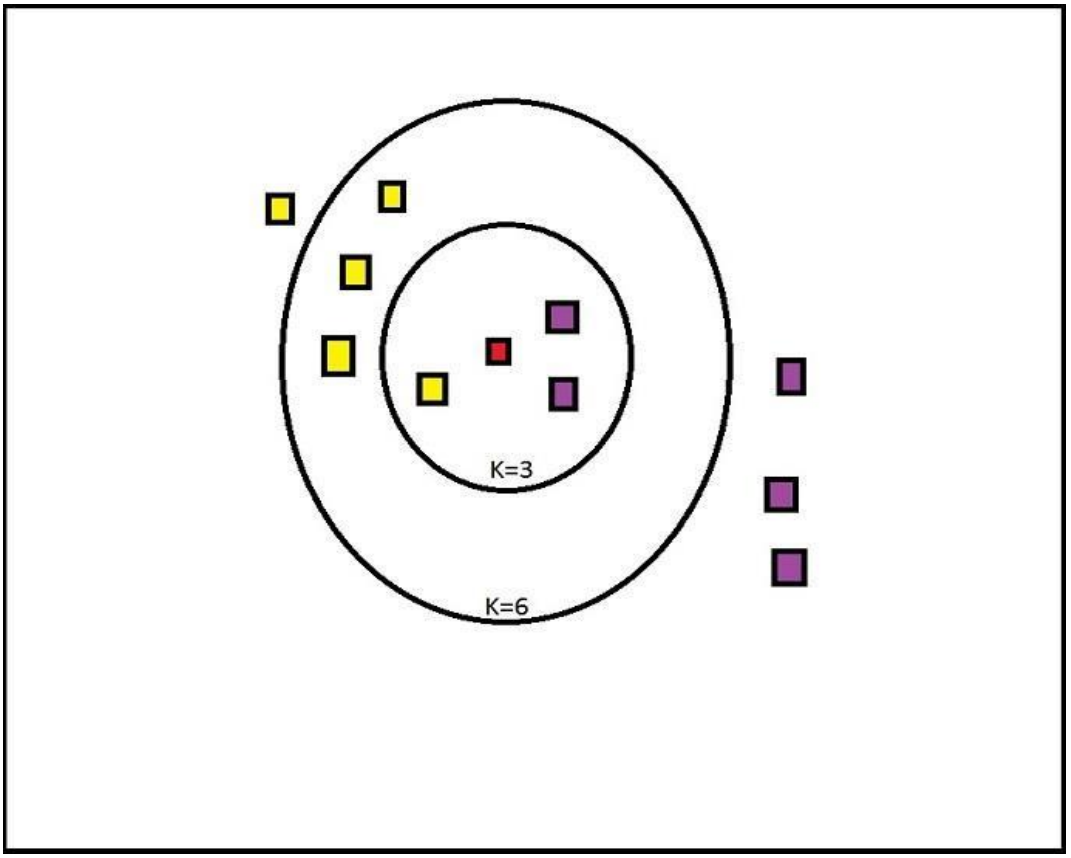
Também é possível fazer o cálculo dos vizinhos mais próximos utilizando a distância

Euclidiana ponderada, que é dada por:

$$d(x, y) = \sqrt{w_1(x_1 - y_1)^2 + w_2(x_2 - y_2)^2 + \ldots + w_n(x_n - y_n)^2}$$

Ao utilizarmos o KNN somente um dos parâmetros é livre para definirmos, o número de K-vizinhos, que é utilizado para obtenção de melhor classificação. Na figura abaixo podemos verificar um exemplo de classificação conforme número de K-vizinhos selecionados, ao selecionar 3 K-vizinhos é possível definir que o elemento pertence à classe “roxa”, porém ao selecionarmos 6 K-vizinhos mais próximos podemos definir o elemento como sendo da classe “amarela”.

Figura 3 - Exemplo KNN



Fonte: Arquivo pessoal.

Para determinadas bases de dados grandes o processo computacional de classificação se torna muito exaustivo, por este motivo existe uma variação mais rápida desse algoritmo. Nesta variação é selecionado pontos de dentro de uma hiper-esfera de raio R, sendo a classe predominante dentro dessa esfera à classe do elemento desconhecido, porém temos como desvantagem o fato de existirem algumas hiper-esferas que não tenham nenhum ponto.

3.3.1.3 Árvore de decisão

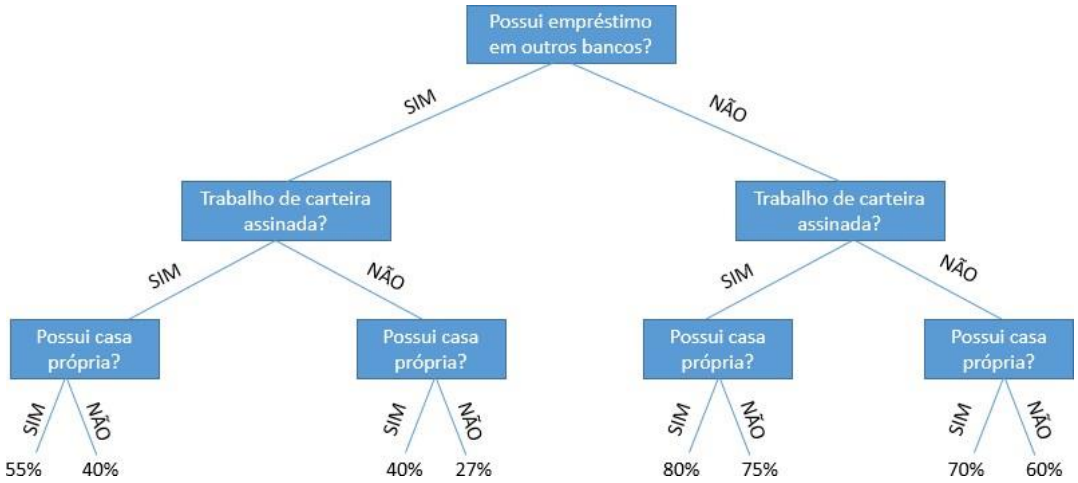
Dentre os métodos de aprendizagem supervisionada existe o de Árvores de Decisão, que são modelos estatísticos para classificação e previsão de dados (Maxwell. 2004). As árvores de decisão são compostas por nós representados por círculos e quadrados conectados por ramos que são representados por linhas, o nó quadrado é um nó de decisão, os ramos que emergem desses nós de decisão representam as alternativas de uma decisão particular. Além destes também estão presentes nós que representam os atributos e arcos que também são ramos, que recebem os valores possíveis para os atributos, ainda temos as folhas que representam as classes de um conjunto de treinamento (Paola G. C. 2010).

Uma das vantagens da árvore de decisão é que o elemento indefinido será testado hierarquicamente, portanto é possível definir a importância dos atributos que serão testados, colocando na parte superior os atributos mais importantes e o restante decrescentemente ao longo da árvore. A ordem natural da execução da árvore de decisão é da raiz para as folhas onde o fluxo natural representa uma regra de classificação.

Computacionalmente falando ao trabalhar com árvores de decisão a tarefa será dividido em três partes, a árvore de decisão em si, o algoritmo para criar a árvore e o algoritmo para aplicação da árvore nos dados, destas partes a criação da árvore é a mais difícil, pois necessita de grande análise para que seja definido os atributos mais relevantes no processamento (Asterio K. Tanaka. 2012).

Podemos verificar na imagem abaixo um exemplo de árvore de decisão no qual um banco fictício é utilizado uma árvore de decisão com alguns atributos onde é definido se o banco fará ou não negócio com o cliente, lembrando que somente porcentagens acima de 50% são consideradas como seguras para o dado exemplo.

Figura 4 - Exemplo de Árvore de decisão (Empréstimo bancário)



Fonte: https://estatsite.com/2016/06/11/1970/

3.3.1.4 Naive bayes

Naive Bayes é um algoritmo de classificação comumente usado para classificação de textos, auxiliando, por exemplo, na detecção de spam. Graficamente as redes Bayesianas são representadas por diagramas

3.3.1.5 Perceptron/Adaline

As duas técnicas foram criadas praticamente ao mesmo tempo, o Perceptron foi apresentado por Rosenblatt em uma revista de psicologia já o Adaline foi apresentado pelos pesquisadores Widrow e Hoffem uma conferência do IRE atual IEEE. Com enfoque inicial em processamento de sinais logo o Adaline passou a ser utilizado dentro da área de redes neurais artificiais (RNA) e seu nome passou a significar ADAptive Linear NEuron (A. C. G. Thomé e C. P. B. Marques. 2003).

O Adaline é idêntico ao Perceptron exceto pelo seu algoritmo de treinamento. O Adaline trabalha com ajuste de pesos visando minimizar uma função de erro quadrático para cada padrão de treinamento.

$$E_i = \frac{(d_i - net(X^i))^2}{2}$$

Onde X^1é o i-ésimo padrão de treinamento e d_ié o valor desejado que o neurônio deva gerar como saída quando estimulado com o vetor X^i(A. C. G. Thomé e C. P. B. Marques. 2003).

$$X^1 = [X_0^i,\ X_1^i, X_2^i, \dots, X_n^i]$$

Portanto a parte de treinamento consiste em aplicar a fórmula de atualização de pesos para cada padrão de treinamento até que o número máximo de iterações seja atingido ou que o valor E_1 fique abaixo de uma tolerância (erro máximo)(A.C.G.Thomée P. B. Marques.2003).

O algoritmo de treinamento das redes Perceptron segue uma determinada sequência, a obtenção do conjunto de amostras de entrada, iniciar os pesos iniciais com valores pequenos, iniciar os ciclos de treinamento, durante os ciclos corrigir os pesos a partir do cálculo da variação deste e atualização de seus valores pela equação dada, calcular o erro encontrado e identificar se for menor ou igual o erro aceitável, se o erro for maior que o aceitável reiniciar o processo a partir do passo três, senão simular a rede treinada para os dados de entrada.

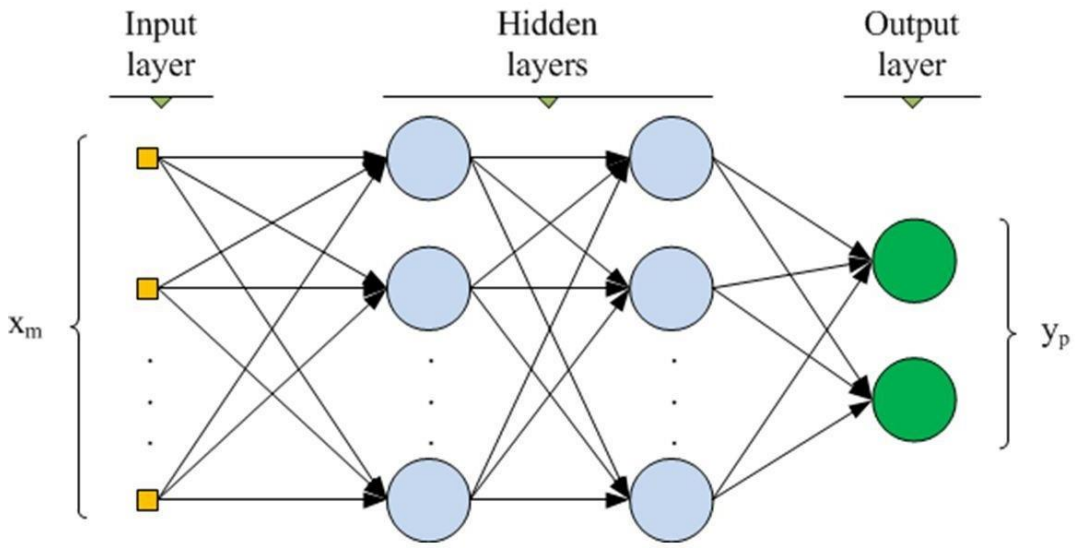
Os Perceptron são algoritmos que utilizam o reforço para aprendizado, ou seja, é utilizado após as interações de entrada e saída, atualizando os pesos com base nessa informação (R. Rojas, Neural Networks, Spring-Vertag, 1996).

3.3.1.6 Multi-Player Perceptron

Uma das técnicas de aprendizado por redes neurais, surgindo na década de 1980 na época em que o investimento em redes neurais voltou a crescer (A. C. G. Thomé. 2003). A MLP é estruturalmente composta por camadas, cada camada dessas contendo vários neurônios, cada um dos neurônios estimula todos os neurônios da camada seguinte, nenhum dos neurônios pode estimular outro da mesma camada ou de camada anterior. A estrutura do MLP consiste em três camadas, a de entrada que é a camada que possui os sinais de entrada, porém não possui neurônios. Camada de saída, contendo os neurônios que geram a saída, e por último as camadas escondidas ou intermediarias que estão situadas entre as camadas de entrada e saída, não existe limites de quantidade dessas camadas assim como a sua existência não é obrigatória.

O algoritmo mais famoso usado para treinamento de uma MLP é o back- progragation, por ser um algoritmo estático é necessário prévio conhecimento da arquitetura da rede e a mesma não pode ser alterada durante o treinamento.

Figura 5 - Exemplo Multi-Layer Perceptron



Fonte: https://www.intechopen.com/source/html/39071/media/f2.jpg

Conforme podemos ver na imagem acima temos a camada de entrada, a camada de saída e as intermediarias, é possível verificar que nas camadas intermediarias não há interação com camadas anteriores somente com camadas posteriores.

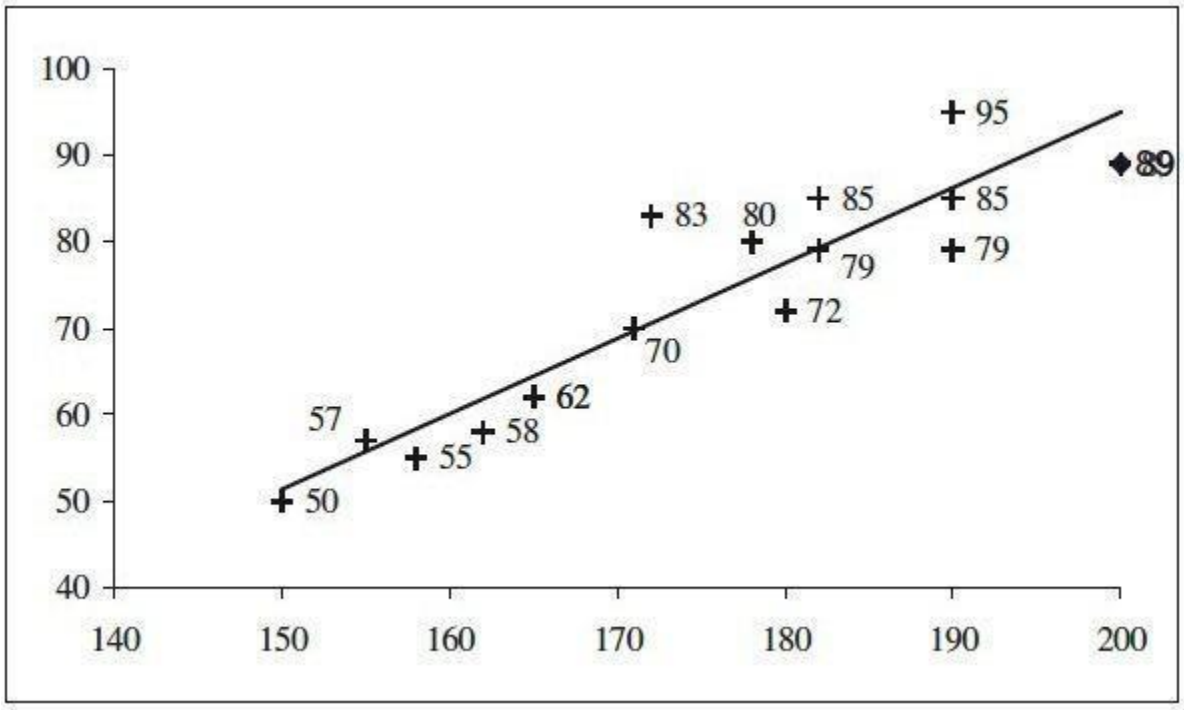
3.3.1.7 Regressão

Dentre as formas de aprendizado supervisionado encontra-se a Regressão, método onde podemos destacar algumas de suas subcategorias como, por exemplo, a regressão linear simples e múltipla. A regressão linear estuda a relação entre uma variável dependente e outra variável chamada variável independente (Ligia Henriques-Rodrigues. 2015). Já a regressão linear múltipla envolve três ou mais variáveis, ou seja, uma única variável dependente e duas ou mais variáveis independentes ou explanatórias ou co-variáveis ou regressoras (Luiz Alexandre Paternelli. 2004). Algumas das técnicas de classificação também podem ser utilizadas na regressão como, por exemplo: o KNN e Adaline.

3.3.1.8 Regressão Linear Simples

A regressão linear simples realiza uma análise estatística com o objetivo de verificar a existência de uma relação funcional entre uma variável dependente com uma ou mais variáveis independentes (Luiz Alexandre Paternelli. 2004). Para a melhor visualização e representação da equação pode-se fazer um gráfico chamado de diagrama de dispersão para que seja possível acompanhar como se comportam os valores da variável dependente (Y) em função da variável independente (X).

Figura 6 - Diagrama de dispersão



Fonte: https://qualescap.files.wordpress.com/2014/11/sem-tc3adtulo112.png

Como podemos ver na figura 6, no diagrama de dispersão existe uma reta a qual seria o “ajuste ideal”, porém vale ressaltar que existem outros modelos que podem representar a relação entre a variável X e Y, podendo ser, linear, quadrática, cúbica entre outras. Os pontos de diagrama de dispersão não se ajustam perfeitamente a curva do modelo matemático utilizado, existindo certa distância entre os pontos e a curva do modelo matemático, isso ocorre devido ao fato de não estarmos trabalhando com um fenômeno matemático e sim fenômeno a qual está sujeito a influencias que acontecem ao acaso. Assim, o objetivo da regressão é obter um modelo matemático que melhor se ajuste aos valores observados de Y em funções da variação dos níveis da variável X (Luiz Alexandre Paternelli. 2004). Alguns

critérios devem ser respeitados para termos melhor obtenção de resultados, como por exemplo, o modelo utilizado que deve ser condizente tanto com a curva quanto com o grau para representação gráfica do fenômeno estudado, além disso, o modelo deve conter somente as variáveis relevantes para explicar o fenômeno, tornando a fase de análise indispensável.

Para que os pontos do diagrama sejam o mais próximo possível do ponto da curva do modelo matemático é possível o uso de alguns métodos entre eles o Método dos Mínimos Quadrados, com o uso desse método a soma dos quadrados das distâncias entre os pontos do diagrama e os respectivos pontos da curva da equação são minimizados, com o mínimo possível de erros.

Estatisticamente o modelo da regressão linear simples consiste da seguinte fórmula:

$$Y_i = \beta_0 + \beta_1 X_1 + e_i$$

Onde o Y é o valor observado para a variável Y, β_0 representa a constante da regressão, β_1 o coeficiente da regressão, X_i o i-ésimo nível da variável independente X e por fim e_i que é erro associado a distância entre as variáveis Y e X.

Para o cálculo do Método dos Mínimos Quadrados (MMQ), devem-se seguir as seguintes etapas, primeiro temos:

$$e_1 = Y_i - \beta_0 - \beta_1 X_1$$

Elevamos todos os membros da equação ao quadrado:

$$e^2 = [Y_i - \beta_0 - \beta_1 X_1]^2$$

E aplicamos a somatória:

$$\sum_{i=1}^{N} \square^2 = \sum_{i=1}^{n} [y_i - \beta_0 - \beta_1 X_1]^2$$

Para que possamos ter uma visão um pouco mais prática sobre como é o funcionamento da regressão linear, usaremos como exemplo a previsão de preço de casas a venda, o primeiro passo é encontrar a base de dados onde teremos informações de casas vendidas anteriormente, quanto mais atributos como, por exemplo, tamanho em metro quadrado, quantidade de cômodos, a quanto tempo a casa foi construída, sua localização entre outros, vale ressaltar que a quantidade de registro na base de dados também pode influenciar na integridade do algoritmo, quanto mais registros tiver a base de dados melhor. Depois de selecionados os dados e normalizados, ou seja, fazendo o tratamento dos valores faltantes (missing values) e demais tratamentos conforme necessidade do projeto, como por exemplo, os valores discrepantes. Em seguida a base de dados é passada no algoritmo de regressão linear, é importante salientar que é necessária a separação de qual atributo é o atributo alvo ou pelo termo técnico classe, no caso do exemplo das casas o atributo classe é o preço de venda. O algoritmo irá treinar para “aprender” como os atributos influenciam e indicar o preço de novos exemplos que forem passados ao algoritmo.

Figura 7 - Gráfico de dispersão (Previsão de preço de casas)

Fonte: https://image.slidesharecdn.com/machinelearning101-speakerdeck-160918000037/95/machine-learning- 101-7-638.jpg?cb=1474157116

3.3.1.9 Regressão Linear Múltipla

A regressão linear múltipla envolve três ou mais variáveis, sendo uma variável dependente e duas ou mais variáveis independentes. A utilização de mais de uma variável independente implica diretamente na precisão da predição do algoritmo, portanto a regressão linear múltipla é uma variação da linear (Luiz Alexandre Paternelli. 2004). Porém com o uso de mais variáveis a técnica de cálculo é bastante complicada, porém é facilitada com o uso de álgebra de matrizes.

Um dos modelos utilizados para regressão linear múltipla é o modelo com K-variáveis regressoras, que se dá por:

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k + \epsilon$$

Onde os parâmetros β_1 são coeficientes de regressão parcial.

Existem alguns métodos para avaliarmos um modelo criado a partir de regressão, tanto simples quanto múltipla dentre elas podemos citar duas: primeiro o Método de mínimos quadrados: A soma dos quadrados representa uma medida de variação ou desvio da média. É calculada como uma soma dos quadrados das diferenças da média. O cálculo da soma total dos quadrados considera a soma dos quadrados proveniente dos fatores e da aleatoriedade ou do erro. A soma dos quadrados dos resíduos. Na regressão, a soma total dos quadrados ajuda a expressar a variação total dos y. A soma dos quadrados do erro residual é a variação atribuída ao erro.

RSS é definida e dada pela seguinte função:

$$MSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Outro método é o de erro quadrático médio que é definido como sendo a média da diferença entre o valor do estimador e do parâmetro ao quadrado. Ele mensura o deslocamento dos valores. Erro baixo, quanto menor for o MSE, melhor está sendo o preditor.

Em outras palavras, quer dizer que o preditor está prevendo valores bem próximos dos dados reais. Seu resultado mostra o quanto os valores preditos estão se distanciando. Os erros são sempre elevados ao quadrado para não retornar um resultado negativo, então o erro

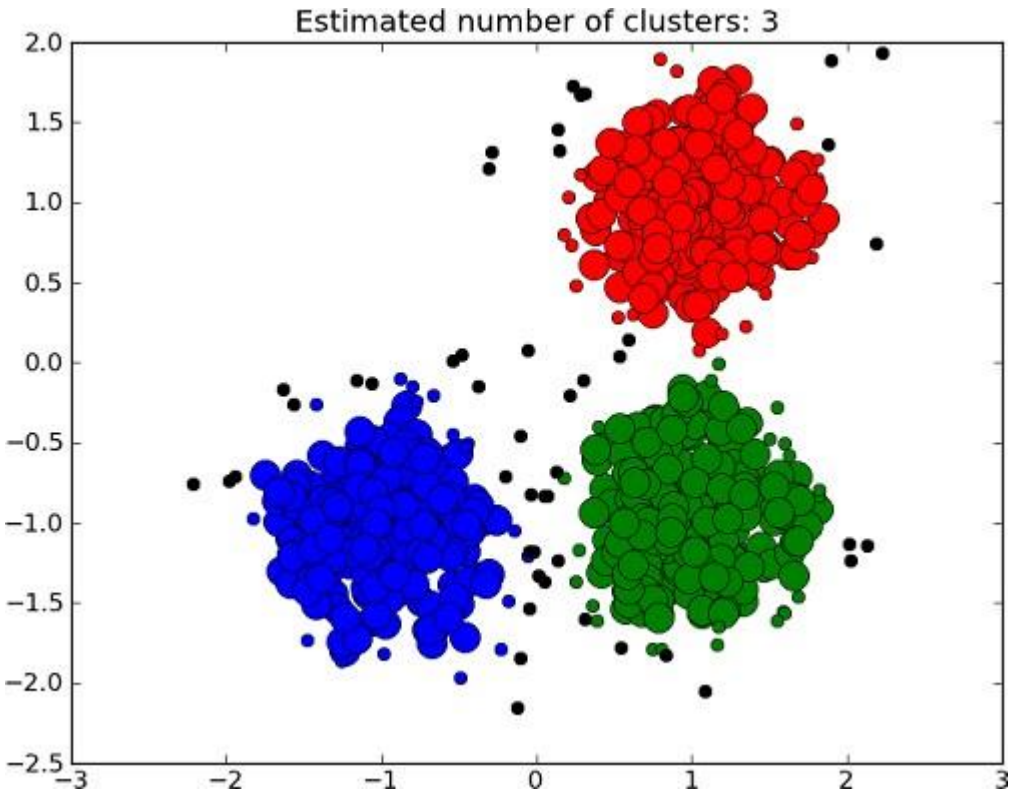
quadrático médio sempre estará maior ou igual a zero (0). Quanto menor, mais aderente, mais próximo é.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}i - yi)^2$$

3.4 APRENDIZADO NÃO SUPERVISIONADO

Ao contrário do aprendizado supervisionado no não supervisionado o algoritmo não observa a saída, somente a entrada dos dados, pois, a classe alvo não é previamente identificada, portanto é feito um agrupamento ou cluster (agrupamento) levando em consideração características parecidas (Heitor Medeiros. 2013).

Figura 8 - Exemplo de Cluster



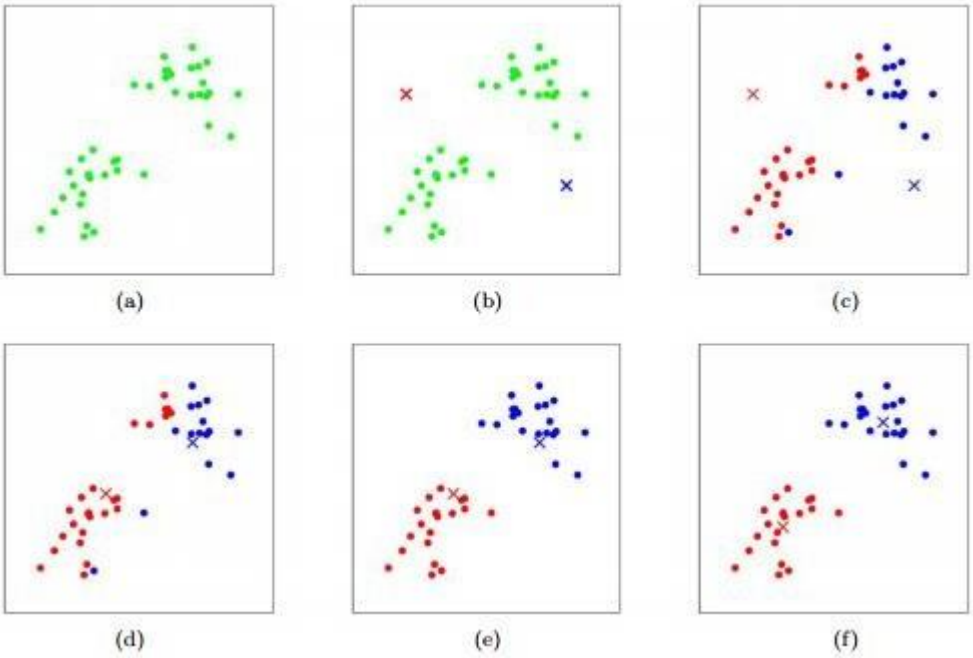
Fonte: https://image.slidesharecdn.com/machinelearning101-speakerdeck-160918000037/95/machine- learning-101-7-638.jpg?cb=1474157116

Como podemos observar na imagem acima, os algoritmos de aprendizado não supervisionado trabalham agrupando os objetos com características parecidas, na imagem podemos ver que após processados os dados se aproximam, tornando-os cluster, neste exemplo verificamos três clusters. Dentre os algoritmos de aprendizado não supervisionado podemos destacar os seguintes:

3.4.1.1 K-means

O algoritmo K-means e o método mais simples de análise de agrupamento (Ke Chen.2012). Definindo um centro para cada cluster. Para utilização do K- means é necessário primeiramente definir quantidade de clusters desejados, após isto é necessário atribuir cada objeto ao cluster de semente mais próxima, a métrica da distância deve ser especificada após os passos são repetidos até obter resultado satisfatório. Calcular novos pontos de semente com os centroids (centro) dos clusters da partição atual. Lembrando que este cálculo da distância é feito com distância Euclidiana, também é possível a utilização da distância Manhattan.

Figura 9 - Etapas K-means



Fonte: http://stanford.edu/~cpiech/cs221/img/kmeansViz.png

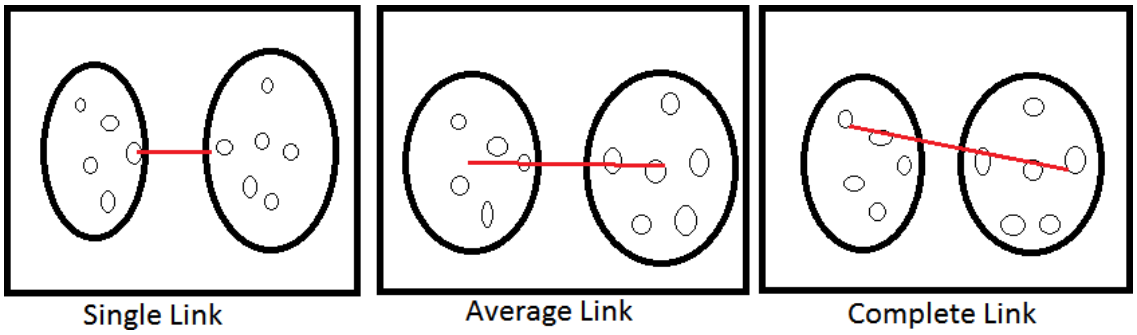
Como verificado no gráfico acima, no quadro A os elementos estão dispersos, já no quadro B foram definidos dois clusters onde ainda não estão associados nenhum elemento. No quadro C, já temos os elementos associados às sementes dos dois clusters definidos, porém estes ainda não estão devidamente centralizados, na sequencia até o último quadro

(f) os centros dos clusters são definidos respeitando melhor as distâncias entre os elementos.

3.4.1.2 Algoritmos Hierárquicos

Os algoritmos hierárquicos criam uma hierarquia de relacionamento entre os elementos. Os algoritmos hierárquicos existem em duas versões a aglomerativa que cria conjuntos a partir de elementos isolados, e a divisiva que começa com um grande conjunto e vai dividindo até chegar a elementos isolados (Ricardo Linden. 2009). Para que seja feito a aglomeração é necessário seguir alguns passos. Primeiro é feito um cluster para cada elemento, após isso é feito a busca para encontrar os pares de clusters mais similares, de acordo com uma medida de distância pré-definida, fundi-los em um cluster maior e recalcular a distância deste para todos os outros elementos, e por fim repetir as últimas etapas até que sobre apenas um cluster. Existem três formas de medir a distância entre dois clusters, single link, onde a distância é medida entre seus pontos mais próximos também conhecidos como agrupamento de vizinhos, priorizando os elementos mais próximos e deixando em segundo plano os mais distantes. Average link, a distância é dada pelos seus centroides e complete link onde a distância entre clusters é a distância entre seus pontos mais distantes.

Figura 10 - Formas para medição da distância de Clusters



Fonte: Arquivo pessoal

É importante salientar que a escolha destas medidas de distâncias é importante, pois cada uma delas pode gerar diferentes resultados de agrupamentos, não sendo equivalentes.

Existe ainda uma versão do algoritmo de aglomeração, porém divisível. Iniciando com um único Cluster que contenha todos os elementos e dividindo ele. Para isso em cada iteração é necessário à utilização de um algoritmo *flat* (particionador) como, por exemplo, o K-means que possa separar os clusters atuais em clusters menores repetindo até que tenha conjuntos compostos por apenas um elemento ou até que um critério de parada estabelecido seja atingido.

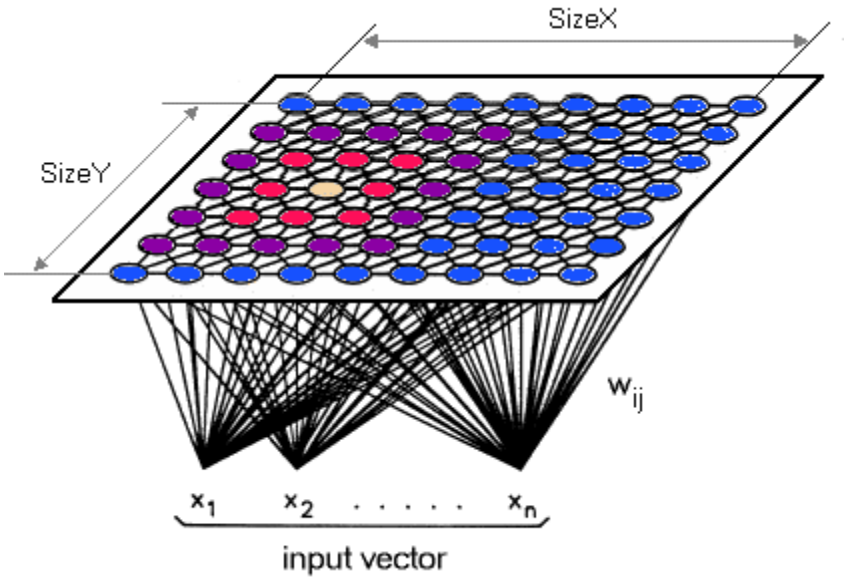
A versão divisiva é mais complexa que a função aglomerativa, pois necessita da utilização de um algoritmo flat, porém por outro lado apresenta algumas vantagens como, por exemplo: não é necessário recalcular todas as distâncias entre os clusters a cada passo e o procedimento pode ser interrompido de maneira programada antes que chegue ao último nível da árvore, podendo em alguns casos ser mais rápido e o fato de começar usando a informação do todo ao invés de tomar decisões baseando em informações locais sendo mais fieis em relação a real distribuição da informação nos elementos.

3.4.1.3 Self organizing maps

As SOM (do inglês, Self Organizing Maps) são redes neurais que implementam os conceitos de auto-organização cerebral, sendo um tipo de aprendizado não supervisionado que trabalha com a auto-organização, ou seja, quando o sistema evolui para uma forma organizada por conta própria sem interferência externa. As SOM são comparáveis a organização neural cerebral que onde insumos sensoriais, seja motor, visual, auditivo, etc, são mapeados em áreas correspondentes do córtex cerebral de forma ordenada, conhecido como mapa topográfico, tendo duas propriedades importantes: Em cada fase de representação cada informação recebida é mantida em seu próprio contexto e os neurônios que lidam com informações intimamente relacionadas são mantidos próximos para que possam interagir através de conexões sinápticas curtas (John A. Bullinaria. 2004).

A SOM é composta por duas camadas a camada de entrada e a camada competitiva (Ricardo Linden. 2009). Cada neurônio da camada de entrada representa uma dimensão do padrão de entrada e distribui seu componente vetorial para a camada competitiva.

Figura 11 - Exemplo de Self Organizing map



Fonte: http://www.decom.ufop.br/imobilis/wp-content/uploads/2017/05/kohonen1.gif

Como podemos ver no gráfico acima a parte inferior (input vector) representa a camada de entrada que estão conectados com a camada competitiva através de pesos enquanto a parte superior representa a camada competitiva. Cada um dos neurônios da camada competitiva recebe a soma ponderada de entrada e tem uma vizinhança de K- neurônios podendo ser em uma, duas ou mais dimensões. Ao receber uma entrada alguns dos neurônios serão excitados o suficiente para dispararem, sendo que cada um dos neurônios disparados pode ter efeito excitatório ou inibitório em sua vizinhança.

Logo após a inicialização dos pesos três processos básicos são realizados: Competição: onde o maior valor é selecionado. Para definição do neurônio vencedor utiliza-se a seguinte formula: $i(x) = \arg\min||x - w||, j = 1,2,3, \dots, |$ o vencedor determina a localização do centro da vizinhança dos neurônios a serem treinados. Cooperação: os vizinhos do neurônio vencedor são selecionados e excitados por uma função de vizinhança. Adaptação sináptica: os neurônios excitados ajustam seu peso sinápticos quando um neurônio vence uma competição, não só o neurônio vencedor tem seu peso ajustado, mas os de sua vizinhança também.

As SOM são eficientes para problemas de classificação, porém, não existe garantia de sua convergência para redes de dimensões maiores além do que o treinamento pode ser extremamente longo para problemas grandes de classificação. Porém apresenta a vantagem de não só descobrir os conjuntos como também mostrar uma ordenação entre eles graças à

noção de vizinhança entre neurônios (Ricardo Linden, Técnicas de Agrupamento, 2009).

3.4.2 Aprendizado semi supervisionado

O aprendizado supervisionado tem por característica a necessidade da utilização de uma grande quantidade de dados rotulados para treinamento fazendo com que em alguns casos seja inviável sua utilização. Para suprir este tipo de necessidade uma alternativa são os algoritmos de aprendizado semi supervisionado, que utilizam uma pequena quantidade de dados rotulados para treinamento e age rotulando o restante dos dados (Bruno Vicente A de Lima et. Al. 2014). O aprendizado semi supervisionado pode ser usado tanto para classificação quanto para agrupamento. Em casos de classificação o objetivo é rotular os dados não-rotulados para que posteriormente possam ser utilizados na fase de treinamento de um classificador, já nos casos de agrupamento os exemplos rotulados são utilizados para a formação dos grupos (Bruno Vicente A de Lima et. Al. 2014). Dentre os algoritmos que utilizam as técnicas de aprendizado semi supervisionado podemos destacar o algoritmo de *self-training* que utiliza uma pequena quantidade de dados iniciais, e parte para a parte prática que consiste em treinar, prever, treinar novamente usando suas próprias previsões e repetir (PiyushRai.2011).

4 METODOLOGIA

4.1 BASE DE DADOS

A escolha e preparação dos dados consistem em uma das partes mais importantes para o aprendizado de máquina. Para nossa proposta a base de dados escolhida foi a de dados escolares de duas instituições secundárias portuguesas, Escola Gabriel Pereira e Escola Mousinho da Silveira, das quais foram selecionadas apenas notas que representem o desempenho nas duas principais disciplinas, ou seja, português e matemática. Os dados utilizados estão disponíveis no UCI Machine Learning Repository Center for machine learning and Intelligent Systems[4] que consiste em um repositório online de bases de dados para a comunidade de aprendizado de máquina, tendo disponível mais de 390 bases de dados.

Antes de dissertar um pouco mais sobre os dados em si bem como sua estrutura, vale comentar sobre as particularidades do sistema de ensino português.

A variação de notas consiste no intervalo entre 0 e 20 sendo 0 a nota mínima e 20 a nota máxima, a média mínima para aprovação do aluno é 10. Em Portugal a educação primária consiste em 9 séries seguidas por 3 séries da educação secundária. Durante o ano letivo o aluno é avaliado em três períodos onde a terceira nota (G3) corresponde à nota final (Paulo Cortez et. Al. 2008).

Os dados foram coletados nos anos de 2005-2006 por Paulo Cortez e Alice Silva (Paulo Cortez et. Al. 2008), que foi o artigo que tomamos como base para nosso trabalho. Para coleta de dados foi usado o banco de dados das instituições para obtenção das notas e dados referentes a ausências escolares. Os demais dados foram coletados através de questionários com perguntas fechadas. Do total de 788 questionários respondidos, foram descartados 111, pois não tinham identificação. Por fim os dados foram organizados em dois conjuntos, o primeiro referente à matemática com 395 instancias e o de português com 649 instancias.

A tabela abaixo mostra a distribuição dos 33 atributos da base de dados bem como descrição e tipo de cada um.

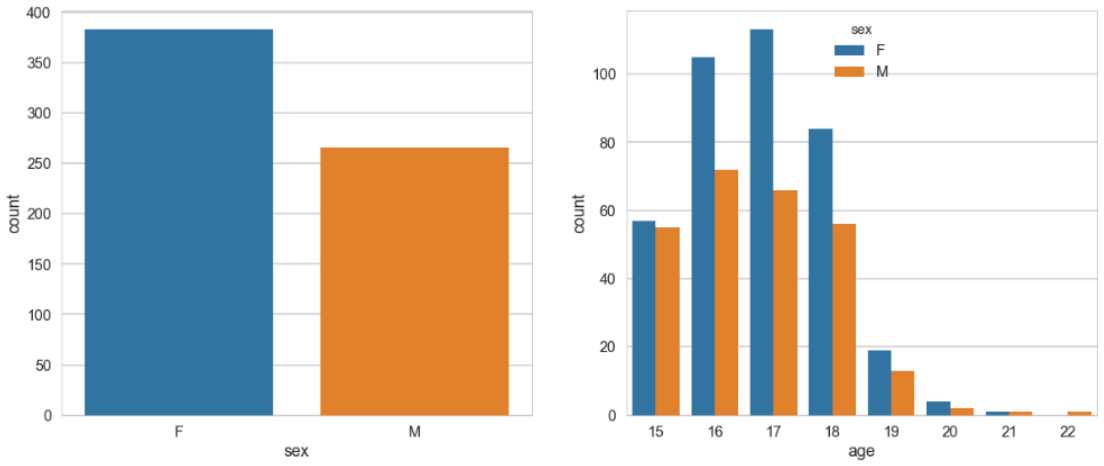
Atributo	Descrição
Sex	Sexo do estudante (binário: Masculino e feminino)
Age	Idade do estudante (numérico: entre 15 e 22 anos)
School	Escola do estudante (binário: Gabriel Pereira ou Mousinho da Silveira)
Address	Tipo de endereço do aluno (binário: urbano ou rural)
Pstatus	Relação de habitação com os pais (binário: vivem juntos ou não)
Medu	Grau de educação da mãe (numérico: de 0 a 4ª)
Mjob	Trabalho da mãe (nominal)
Fedu	Grau de educação do pai (numérico: de 0 a 4)
Fjob	Trabalho do pai (nominal)
Guardian	Responsável pelo aluno (nominal: mãe, pai ou outros)
Famsize	Quantidade de membros da família (binário: <= 3 ou >3)
Famrel	Qualidade da relação familiar (entre 1 – muito ruim e 5 – excelente)
Reason	Motivos para o qual escolheu esta escola (nominal: perto de casa, reputação da escola ou outro)
Traveltime	Tempo de viagem até a escola (numérico: 1 - <15 min., 2 – 15 a 30 min., 3 – 30 min. há uma hora, 4 - >1 hora)
Studytime	Tempo semanal de estudo (numérico: 1- <2 horas, 2- 2 a 5 horas, 3- 5 a 10 horas, 4- >10 horas)
Failures	Número de reprovações em classes passadas (numérico: entre 1 e 4)
Schoolsup	Apoio educacional extra (binário: sim ou não)
Famsup	Suporte educacional familiar (binário: sim ou não)
Activies	Atividades extracurriculares (binário: sim ou não)
Paidclass	Aulas extras pagas (binário: sim ou não)
Internet	Acesso a internet em casa (binário: sim ou não)
Nursery	Frequentava a escola maternal (binário: sim ou não)
Higher	Pretende fazer ensino superior (binário: sim ou não)
Romantic	O aluno tem relacionamento amoroso (binário: sim ou não)
Freetime	Tempo livre após a escola (numérico: de 1- pouquíssimo a 5 muito)
Goout	Sair com os amigos (numérico: de 1- pouquíssimo a 5- muito)
Walc	Consumo de álcool no fim de semana (numérico: 1- po pouquíssimo a 5- muito)
Dalc	Consumo de álcool no meio de semana (numérico: 1- pouquíssimo a 5- muito)
Health	Estado de saúde atual (numérico: 1- muito ruim a 5 muito bom)
Absences	Número de faltas (numérico: entre 0 e 93)

G1	Primeiro período (numérico: de 0 a 20)
G2	Segundo período (numérico: de 0 a 20)
G3	Nota final (numérico: de 0 a 20)

4.2 ANÁLISE EXPLORATÓRIA

É importante conhecer e entender os dados com os quais se está trabalhando. Para isto, algumas técnicas são de grande auxílio, dentre elas a plotagem dos dados em gráficos que auxiliam a ter uma visão melhor. Como, por exemplo, nos gráficos a seguir.

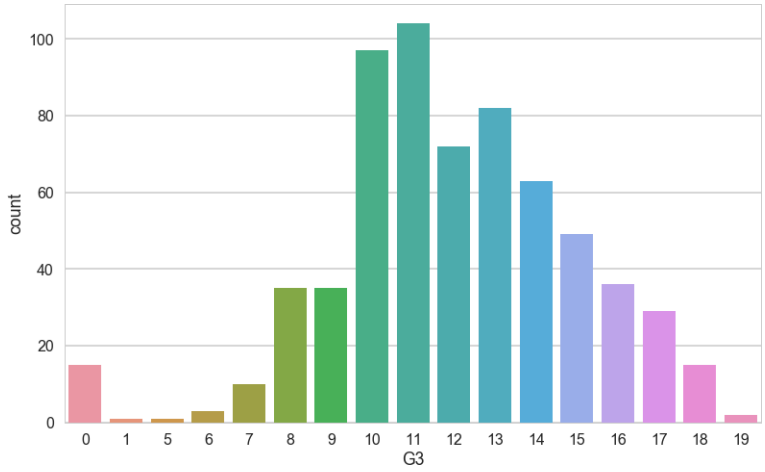
Figura 12 - por sexo e notas por sexo



Fonte: Arquivo pessoal

Nos gráficos acima podemos ver como está a distribuição dos dados em relação ao sexo do aluno, onde podemos observar que o sexo feminino possui uma quantidade de 383 registros enquanto o masculino possui 266 registros. Podemos ainda notar que o intervalo entre 16 e 18 anos representa a idade da maioria dos alunos, sendo que percentualmente pessoas do sexo feminino apresentam maior quantidade em relação ao masculino também dentro do intervalo entre 15 e 20 anos.

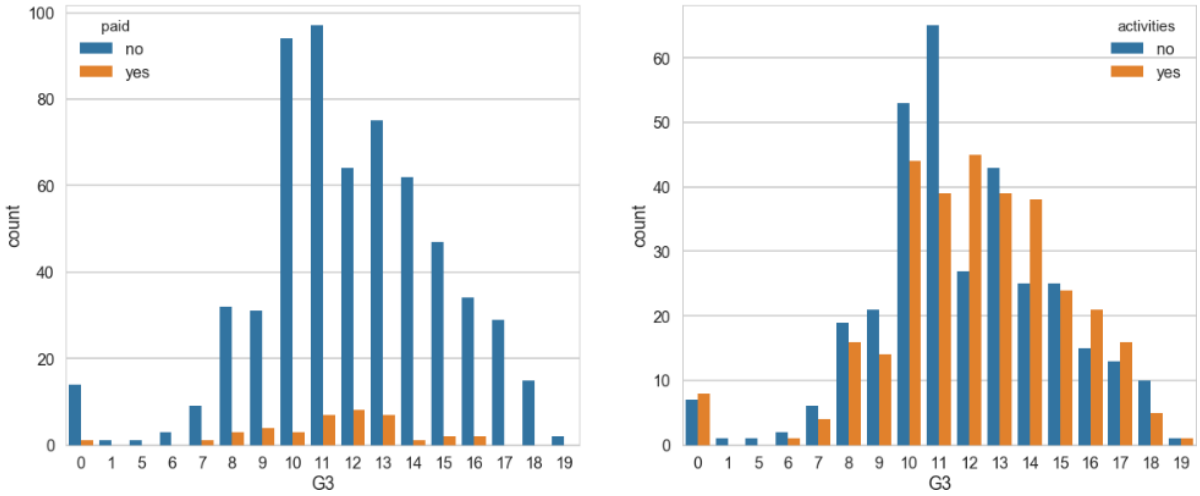
Figura 13 - Distribuição das notas



Fonte: Arquivo pessoal

O gráfico acima representa a distribuição de quantidade de alunos pela terceira nota (G3). Analisando podemos fazer algumas inferências, como por exemplo, que a maioria dos alunos consegue atingir a média necessária para aprovação, mas apesar disso, a maioria esmagadora dos alunos somente atinge a média, ou ficando próximo da nota necessária para aprovação, ficando a maioria dos alunos com notas entre 10 e 13. Portanto notamos que várias informações podem ser obtidas já na análise dos dados.

Figura 14 - Aulas extras pagas e Atividades extracurriculares

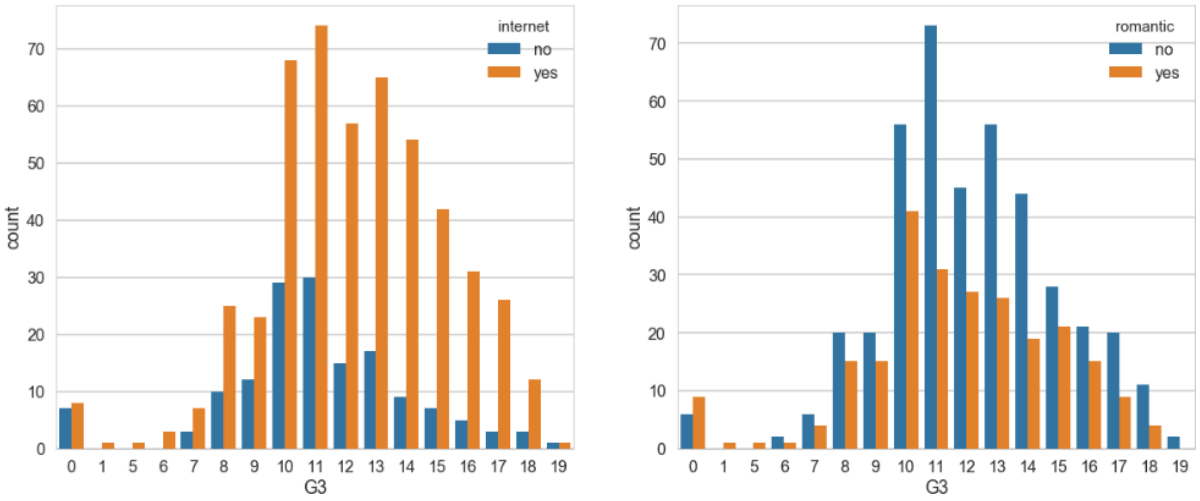


Fonte: arquivo pessoal

Como podemos observar em todo o intervalo de notas os alunos que não tiveram aulas

extras pagas apresentaram desempenho superior, alguns fatores possivelmente explicariam esta disparidade, entre eles está o fato de o número de alunos que tiveram aulas extras pagas ser bastante reduzido em relação aos que não fizeram, cerca de 40 alunos o fizeram enquanto cerca de 610 não fizeram aulas extras pagas. Enquanto isso vemos que alunos que fizeram atividades extracurriculares obtiveram vantagens significativas nas notas 12, 14, 16, 17. Notas consideradas altas, portanto poderíamos inferir que as realizações de atividades extras apresentam a possibilidade de serem benéficas e apresentar um rendimento maior escolar ao aluno que as praticam.

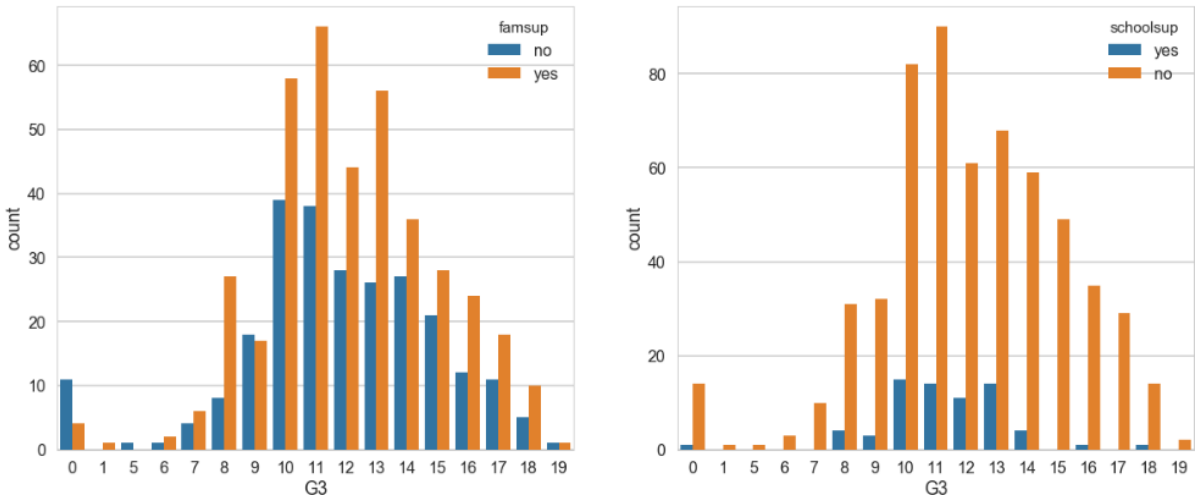
Figura 15 - Acesso à internet - Relacionamento amoroso



Fonte: Arquivo pessoal

No gráfico acima temos uma perspectiva da situação dos alunos em relação a acesso à internet em casa. A grande maioria de alunos em âmbito geral tem acesso à internet, dos 649 alunos na base de dados cerca de 498 possuem acesso à internet, também é possível notar que quanto mais alto a nota maior a diferença entre alunos que possuam acesso à internet em detrimento aos que não possuem. Quando o assunto é relacionamentos amorosos notamos que em alguns intervalos de notas os alunos que não possuem relacionamento tiveram desempenho muito superior aos demais, como por exemplo, a nota 11 e 13, além destas 2 notas que apresentam a maior diferença notamos que praticamente todas as notas acima da média os alunos sem relacionamento amoroso apresentaram desempenho superior.

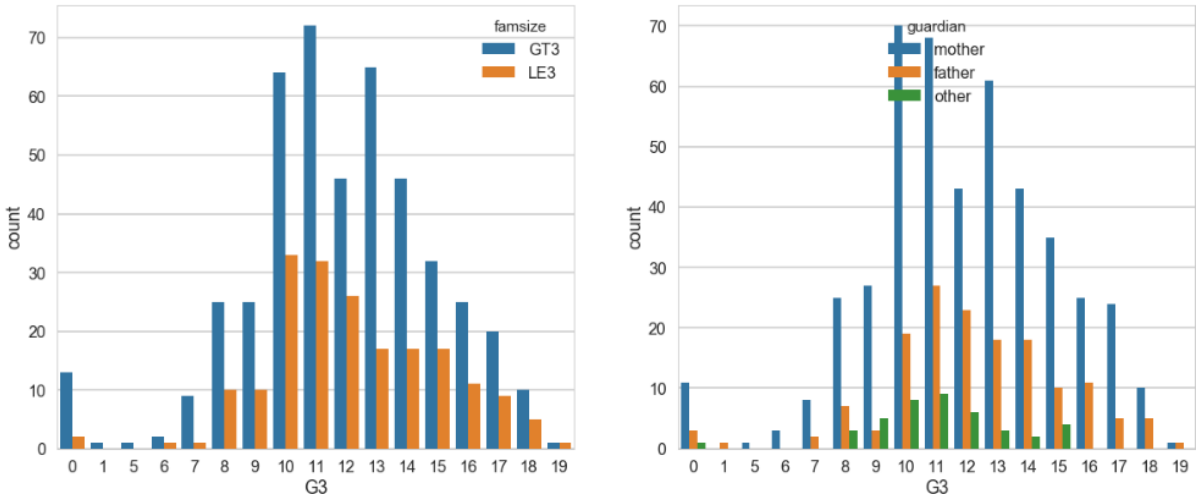
Figura 16 - Apoio Educacional Familiar - Apoio Educacional Escolar extra



Fonte: Arquivo pessoal

Quando levamos em conta suporte familiar a educação notamos um fato que pode trazer consequências negativas, o número de alunos sem esse tipo de suporte é elevado dos quais cerca de 250 não o possuem. Os alunos com suporte familiar tiveram desempenho consideravelmente melhor. Já em relação a apoio educacional extra por parte da instituição encontramos um quadro onde apenas 39 alunos tiveram este tipo de suporte, tornando graficamente pouco relevante.

Figura 17 - Tamanho da Família - Responsável legal do aluno

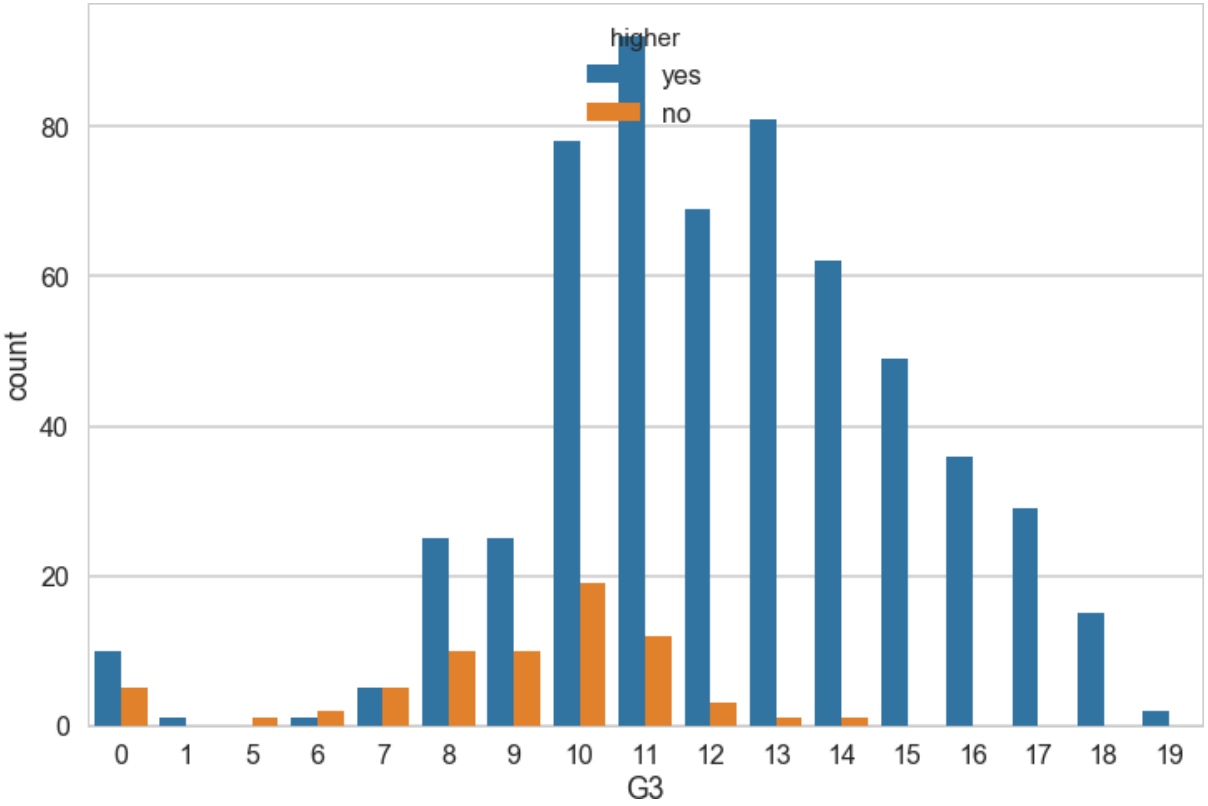


Fonte: Arquivo pessoal

No gráfico acima identificamos que alunos que possuem família com mais de 3 membros apresentam vantagem sobe os que possuem menos de 3 membros nas notas intermediarias, perdendo essa vantagem conforme a nota aumenta chegando a se equiparar na nota 19. Parte dessa disparidade pode ser explicada pela diferença de número de alunos com mais de 3 membros

na família, chegando a 457 do total de 649. Outro dado importante para análise é o responsável legal pelo aluno dos quais notamos que a grande maioria tem a mãe como responsável sendo 455, seguidos por 153 que tem como responsável o pai e 41outros, vale salientar que o desempenho dos alunos que possuem como responsável a mãe apresenta superioridade aos demais, porém em notas mais altas como 16 e 18 essa diferença diminui em favor aos alunos com pai como responsável.

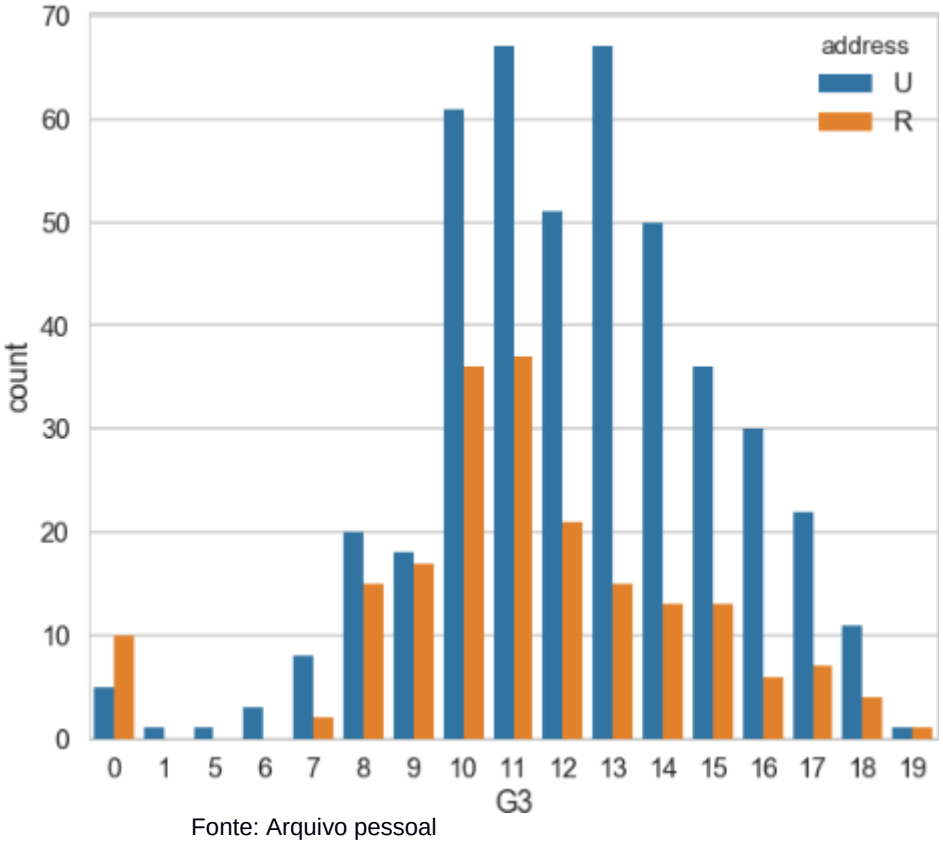
Figura 18 - Interesse em fazer ensino superior



Fonte: Arquivo pessoal

Outro atributo importante para análise é o interesse do aluno em fazer um curso de graduação, dos quais notamos que a grande maioria possui este interesse, cerca de 580 alunos, além disso, notamos que os alunos que não possuem interesse em seguir os estudos para ensino superior, obtiveram notas baixas, em sua grande maioria atingindo apenas a média (10) tendo como maior nota 14.

Figura 19 - Tipo de endereço (Urbano ou Rural)



Fonte: Arquivo pessoal

O número de alunos que moram em região urbana é bastante superior ao de alunos que moram em região rural, porém é interessante notar a distribuição das notas onde tanto de região urbano quanto rural, existe uma grande distribuição, principalmente para notas que ficam entre a média e acima, enquanto notas abaixo da média chegamos a ter um intervalo composto apenas por alunos que moram em região urbana (1 a 6).

4.3 FERRAMENTAS E BIBLIOTECAS

No processo do desenvolvimento do presente projeto algumas ferramentas, bibliotecas e frameworks são de suma importância. Ao iniciarmos a fase preparatória do desenvolvimento devemos escolher a linguagem de programação que utilizaremos no caso a linguagem escolhida foi Python, graças a grande quantidade de recursos e bibliotecas presentes para quem deseja trabalhar na área de Inteligência artificial (Wes Mckinney. 2013). Dentre estes recursos podemos destacar alguns que nos auxiliaram no desenvolvimento do projeto.

Jupyter Notebook[4]é um projeto de código aberto que nasceu em 2014 sendo um sistema exclusivo para programação em Python, porém com o passar do tempo e a evolução do software passou a suportar outras linguagens. O Jupyter Notebook é uma IDE, sendo um software cliente- servidor voltado a programação, a escolha de trabalhar com o Jupyter deve-se ao fato de que o software é interativo, os códigos são executados na mesma página, além da flexibilidade, pois possibilita misturar imagens, texto e código executável. Facilitando o acompanhamento do que está sendo feito.

Pandas[5]é uma biblioteca de código aberto para a linguagem de programação Python voltada a manipulação de bases de dados (DataFrame), é uma ótima ferramenta para se trabalhar com dados estruturados (Wes Mckinney, 2013) sendo possível a leitura de arquivos de extensões .csv, .xml entre outros, além de permitir a criação de novas colunas na base de dados, detalhar informações, criar novos arquivos csv e xml a partir dos dados tratados pelo código, agregar conjuntos de dados bem como separação em subconjuntos, além de várias outras funções de manipulação de arquivo. O Pandas combinado aos recursos de computação de matriz do NumPy se torna em uma das ferramentas mais poderosas para aprendizado de máquina comPython.

Matplotlib[6]Biblioteca Python mais popular para produção de gráficos e outros dados 2d foi criada por John D. Hunter neurobiologista americano, durante suas pesquisas para seu pós-doutorado, foi criada originalmente para visualizar dados de Eletrocorticografia de pacientes com epilepsia.

Numpy[7]Abreviação de Numeric Python, o *Numpy* é uma biblioteca criada para a computação científica, trabalhando com arrays e matrizes multidimensionais possuindo uma quantidade grande de funções matemáticas que facilitam as operações com array, dentre os cálculos possíveis de serem feitos estão, as operações básicas da matemática (soma, subtração, divisão e multiplicação) entre arrays, operações de álgebra linear, geração de números aleatórios entre outras. O *Numpy* é frequentemente comparado ao *MatLab*, ferramenta para mesma finalidade, porém paga. Para a computação científica utilizando a linguagem de programação Python, está é uma ferramenta imprescindível. Além de todas as funções presentes na biblioteca, seu uso em conjunto com outras ferramentas como o Pandas traz inamismo ao desenvolvimento, otimizando o desenvolvimento.

[4]Disponível em: <http://jupyter.org/>. Acesso em 06 out. 2017

[5]Disponível em: <http://pandas.pydata.org/>. Acesso em 06 out. 2017

[6]Disponível em: <https://matplotlib.org/>. Acesso em 06 out. 2017

[7]Disponível em: <http://www.numpy.org/>. Acesso em 05 out. 2017

Sklearn[8]uma das ferramentas mais importantes para quem deseja trabalhar com aprendizado de máquina, pois esta ferramenta de código livre acompanha várias funções já pré-programadas para o aprendizado de máquina. O Scikit-Learn da suporte a várias das principais técnicas de aprendizado de máquina seja para regressão linear, Lasso regresion, Classificação, Clustering dentre outras. A utilização de Sklearn possibilita a automatização no trabalho com aprendizado de máquina, pois ao invés do programador ter a necessidade de fazer todo o processo manual desde a criação de uma função para carregamento dos dados fazendo a separação de parte dos dados para treinamento e outra parte para validação, passando pela criação de outra função para treinamento do algoritmo com a base de teste, até a parte da validação onde e necessário criação de função que possa processar os dados que foram previamente separados para validação para testar a eficiência do algoritmo programado. Com o Scikit-learn todos estes processos são feitos automatizados pelas funções pré- compiladas, poupando tempo e otimizando o código.

[8]Disponível em: <http://scikit-learn.org/>. Acesso em 05 out. 2017

5 DESENVOLVIMENTO

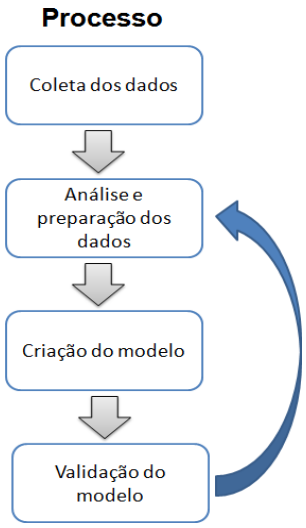
5.1 ESCOLHA DA TÉCNICA DE MACHINE LEARNING A SER UTILIZADA

Quando falamos em Machine Learning várias são as técnicas que podemos utilizar. Para que possamos saber a melhor técnica para cada caso, é necessária uma análise profunda, onde leva-se em conta, a base de dados a qual está trabalhando, qual a finalidade do trabalho e qual tipo de retorno queremos ter. Para este trabalho, foram testadas inicialmente algumas técnicas para realização de diferentes tarefas. Foram obtidos alguns resultados preliminares por meio de experimentos realizados com a tarefa de classificação, contudo, optou-se por se concentrar na tarefa de predição para se manter o escopo da proposta inicial do trabalho. Com o uso da regressão, foi possível obter uma estimativa da nota final do aluno. Com a possibilidade de estimara nota do aluno, o leque de opções preventivas a evasão é maior, pois as ações tomadas para que um aluno não se desmotive pelo baixo rendimento são muito particulares, portanto um aluno que está próximo da média, não necessariamente precisa da mesma medida de prevenção que um aluno que está longe de atingir a média.

5.2 IMPLEMENTAÇÃO

5.2.1 Etapas

Durante o processo da criação do modelo algumas etapas são necessárias para que possamos ter um melhor controle e eficiência. No desenvolvimento do nosso projeto quatro etapas foram respeitadas, como podemos ver na seguinte imagem:



A primeira etapa é a de coleta de dados que consiste em conseguir a base de dados a qual deseja trabalhar, para isso existe a possibilidade da obtenção junto a alguma instituição seja pública ou privada ou ainda nos repositórios online como, por exemplo, no UCI – Machine Learning Repository, onde conseguimos a base de dados que aqui trabalhamos. Em seguida, a próxima etapa consiste na análise e preparação dos dados, é nesta etapa onde é feito a conferência se existem dados faltantes, se sim qual será a melhor maneira de tratá-los, verificar se existem dados discrepantes e determinar se eles podem ou não interferir na predição do modelo. A base de dados que utilizamos já foi disponibilizada no repositório em partes normalizada, ou seja, sem dados faltantes. A próxima etapa é a da criação do modelo, a implementação do código é dada nesta etapa após a escolha do método de aprendizado mais adequado. Para implementação é possível fazer todo o procedimento de forma manual, criando todas as funções e procedimentos, porém, também é possível a utilização de bibliotecas que automatizam várias funções como, por exemplo, a Scikit-learn que possui vasta gama de códigos pré-programados que auxiliam o programador. A última etapa é a de validação do modelo, onde os dados que foram separados para teste são submetidos ao modelo e através de funções de validação é possível mensurar o quanto o modelo foi assertivo e qual o nível de acurácia do mesmo. É importante salientar que algumas destas etapas não são estáticas, ou seja, é necessário a repetição das três últimas etapas até que o modelo esteja satisfatório.

5.2.2 Criação do Modelo

Após todas as etapas de preparação dos dados concluída, é iniciada a fase de criação do modelo, conforme anteriormente mencionado. Existem bibliotecas que auxiliam na criação

do modelo de aprendizado, como por exemplo, o Scikit-Learn, porém, para melhor entendermos o funcionamento do modelo é importante a criação do mesmo de forma manual sem o auxílio desta biblioteca. Portanto criamos um modelo com o auxílio da biblioteca e outro sem o auxílio. Para o desenvolvimento de ambos os métodos utilizamos a ferramenta Jupyter Notebook para implementação. Primeiramente falaremos da criação com auxílio do Scikt-Learn.

Figura 20 - Trecho de código Regressão com Scikit-Learn

```
# import LinearRegression
from sklearn.linear_model import LinearRegression
# Create linear regression object
reg = LinearRegression()
# Train the model using the training sets
reg.fit(X_train, y_train)
# execute prediction
y_pred = reg.predict(X_test)
```

Fonte: Arquivo pessoal

Como visto na figura acima, a implementação da regressão linear utilizando Scikit-Learn é relativamente simples, sendo necessário como vemos na primeira linha a importação da biblioteca responsável por trabalhar com a regressão linear, na segunda linha é instanciado um objeto (reg) com as funções de regressão, na terceira linha é feita o treinamento da função com a base de dados previamente separada para treino e, na última linha é carregado na variável de nome y_pred os dados retornados pela função de predição do algoritmo (reg.predict()).

Porém, somente utilizando as funções do Scikit-Learn não é possível entender o funcionamento do código, já que as mesmas abstraem a complexidade do mesmo e, por isso também desenvolvemos o modelo sem a utilização do Scikit-Learn, programando manualmente todas as etapas antes mostradas. Vale ressaltar que os dois modelos estarão disponíveis nos apêndices 1 e 2.

Primeiramente é necessária a criação de uma função para computar o declive e a interceptação da regressão linear simples. Como já temos o declive e a interceptação, o próximo passo é a criação de uma função para a predição dos valores em si, onde é de suma importância o valor de interceptação e declive. Como já falado anteriormente é necessário fazer a validação do modelo. No nosso código utilizamos o método de mínimos quadrados, para entendermos se o modelo está nos trazendo resultados satisfatórios.

Apesar de existirem bibliotecas que automatizam este teste, implementamos manualmente a função para cálculo do método de mínimos quadrados.

5.2.3 Testes e validações

Para que possamos ter um ajuste melhor de nosso modelo e conseguirmos resultados de previsões mais próximos em relação aos dados reais é necessário o uso de técnicas de otimização. A técnica por nós utilizada é a de gradient descent, técnica esta que consiste em encontrar o mínimo de uma função, aproximando ao máximo o valor previsto do valor real, este algoritmo se destaca em encontrar os melhores pesos. Na otimização baseada em Gradient Descent é feita a tentativa de se encontrar a perda, ou seja, o quão ruim sua previsão foi com um determinado conjunto de pesos. O algoritmo trabalha para que em cada iteração, seja calculado o gradiente do termo de erro para descobrir a direção apropriada para mover nossos pesos. Podemos dizer que estamos calculando as mudanças a serem feitas em nossos pesos para reduzir o erro, ajustando nossa solução para a mais próxima possível do ideal. Para a utilização de gradient descent foi criada uma função contendo o algoritmo responsável pelo cálculo de melhor distância. Para a utilização desta técnica existem alguns valores de suma importância que devem ser definidos previamente para otimização do resultado, que são: tolerance e step size.

Tolerance é usado como condição de parada para o algoritmo, já que o mesmo é feito de forma iterativa e precisa de uma parada em seu laço de repetição, além disso, delimita o valor resultado da equação, portanto é necessário a definição do valor mais adequado, lembrando que para cada modelo é necessária utilização de valor diferente, pois deve-se analisar cada caso separadamente. Por estarmos trabalhando com algoritmo iterativo precisamos da definição do tamanho do próximo passo que será realizado, para isso precisamos definir o valor destepsize.

Step Size é a taxa de aprendizagem, é um fator na regra de atualização para os parâmetros que ajudam a determinar a rapidez com que o algoritmo converge para a solução ideal. A definição do tamanho do passo é um fator determinante o quanto deseja mover enquanto tenta descer na descida gradiente, ou seja, minimizar o máximo possível a perda. O sinal do tamanho da etapa deve ser sempre positivo. O algoritmo irá decidir a direção do movimento em que a descida irá se mover. O step size não pode ser muito pequeno, pois levará muito mais tempo para atingir os melhores pesos, ou seja, o tempo para execução do treinamento será relativamente maior. Tendo isso em mente devemos definir o step size com um valor maior, pois poderá resultar em uma grande redução no tempo de treinamento. No entanto, isso pode ser considerado um problema maior ainda. Se o tamanho for muito grande, o algoritmo pode superar a posição ideal e manter oscilando entre os dois lados.

O algoritmo de atualização de peso é representado pela seguinte fórmula:

$$p = p + (-stepe_size) * d$$

p = O conjunto atual de pesos

d = Gradiente de perda em relação ao peso

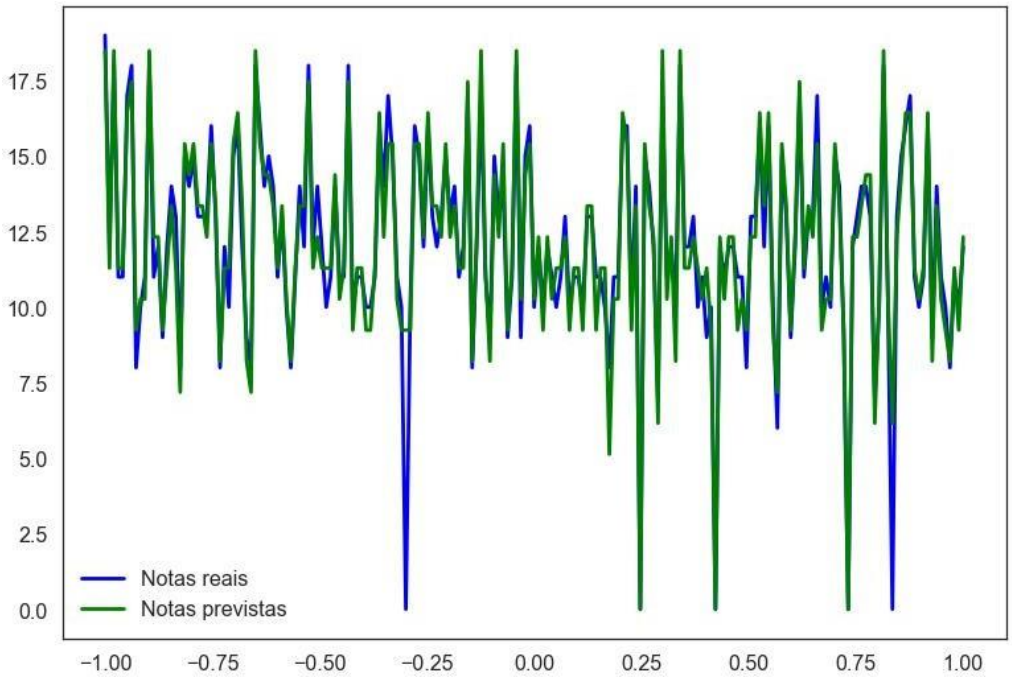
Todo modelo deve passar pela fase de testes e validação para que se possa analisar a eficiência do mesmo. Durante o desenvolvimento de nossos modelos utilizamos de várias técnicas para teste de eficiência, dentre elas a do método dos mínimos quadrados, erro quadrático médio e também pela plotagem de gráficos que nos dá a possibilidade de interpretação por meio da representação visual. A seguir mostraremos alguns dos testes realizados para validação de nossos modelos, acompanhando dados referentes às três técnicas citadas. Os gráficos apresentados mostrarão a diferença das notas previstas e das reais.

5.2.3.1 Modelo com apenas o atributo “G2”

O primeiro modelo apresenta resultados de predição levando em consideração apenas um atributo dos dados, no caso a segunda nota (G2). Este atributo foi escolhido de forma aleatória para que fosse possível uma primeira análise empírica do processo. Para este modelo utilizamos como valore para tolerance de 1.0e4 e stepsize 7e-6, pois foram os valores que trouxeram melhores resultados ao modelo. Podemos verificar no gráfico abaixo (figura 21) que o valor previsto pelo modelo se aproximou consideravelmente dos valores reais salvo exceções em alguns registros que apresentaram discrepância dos demais.

RSS = 239.195042723
Mean squared error = 1.22664124473

Figura 21 - Predição com apenas a "G2"



Fonte: Arquivo Pessoal

Tabela 1 - Demonstrativo entre a diferença da nota real e a nota prevista pelo modelo

NOTA REAL	NOTA PREVISTA	NOTA REAL	NOTA PREVISTA
18	18	11	12
11	12	9	9
13	11	9	9
12	11	8	7
8	7	18	18
14	13	11	12
11	12	18	18
18	18	14	13
19	18	12	11
18	18	11	10
13	11	19	18
14	13	12	12
12	12	9	9
13	11	12	11
9	9	12	11
12	12	9	9
9	9	8	9
14	13	18	18
9	9	15	15
12	11	18	18
11	12	19	18
12	12	9	9
8	7	14	13
10	10	13	11
12	12	8	7
8	7	11	12
11	12	18	18
11	10	11	12
14	13	13	11
9	9	11	10
12	11	12	11
11	10	11	12
11	12	18	18
15	15	10	10
11	10	11	12
14	13	18	18
13	11	12	12
18	18		

NOTA REAL	NOTA PREVISTA	NOTA REAL	NOTA PREVISTA
14	13	11	12
12	11	13	11
14	13	13	11
9	9	18	18
9	9	12	11
9	9	19	18
13	11	14	13
9	9	12	11
10	10	12	12
12	12	8	9
18	18	8	7
14	13	11	12
13	11	12	12
12	12	18	18
11	10	15	15
10	10	12	12
15	15	12	12
13	11	9	9
12	11	18	18
14	13	11	12
12	11	11	10
11	12	18	18
9	9	19	18
18	18	12	12
10	10	11	12
11	12	12	12
12	11	12	12
12	12	11	12
15	15	11	12
12	12	10	10
12	11	9	9
12	12	9	9
18	18	13	11
11	12	12	12
12	11	13	11
11	12	18	18
11	12	18	17
15	15	14	13
18	18	9	9
11	12	11	10
11	12	12	12

NOTA REAL	NOTA PREVISTA	NOTA REAL	NOTA PREVISTA
14	14	12	12
12	12	18	18
15	15	12	12
11	12	11	12
11	12	18	18
8	7	11	12
15	15	9	9
10	10	18	18
18	18	11	12
11	12	11	12
15	15	18	18
11	12	9	9
12	12	9	9
11	10	11	12
12	12	11	12
12	11	18	18
9	9	10	10
10	10	11	12

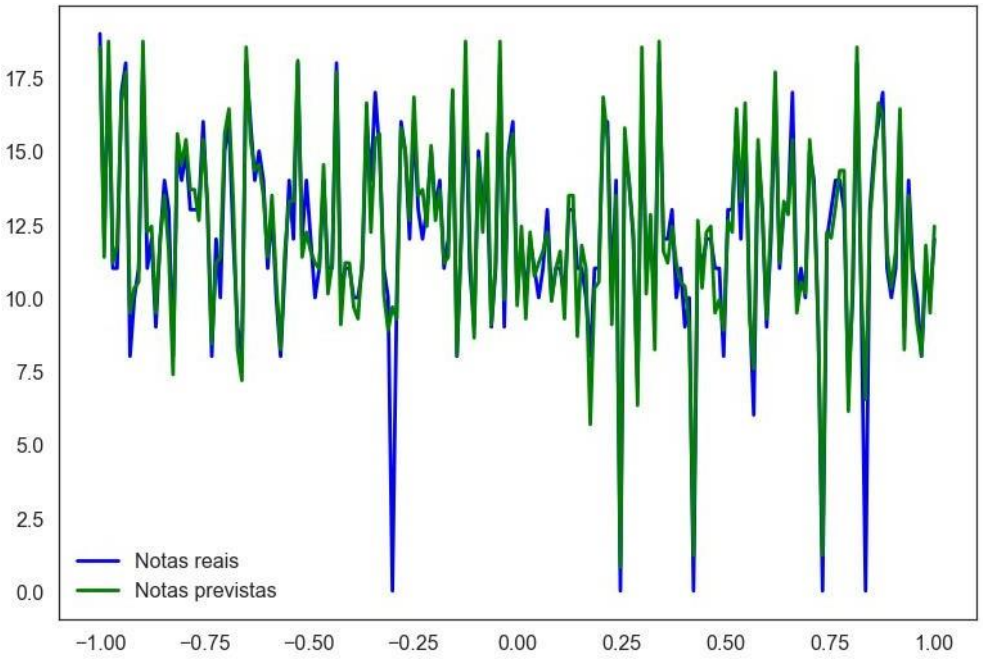
5.2.3.2 Modelo com os atributos “G1” e “G2”

No segundo modelo utilizamos dois atributos a primeira e segunda nota (G1 e G2). Neste modelo podemos ver o comportamento do mesmo quando inseridos mais atributos além de verificar se existe alguma alteração na eficiência do modelo com as alterações realizadas, para este segundo modelo os valores mais adequados para tolerance e stepsize foram respectivamente 1.0e1 e 7.0e-6.

RSS = 253.815927349

Mean squared error = 1.30162014025

Figura 22 - Predição com "G1" e "G2"



Fonte: Arquivo pessoal

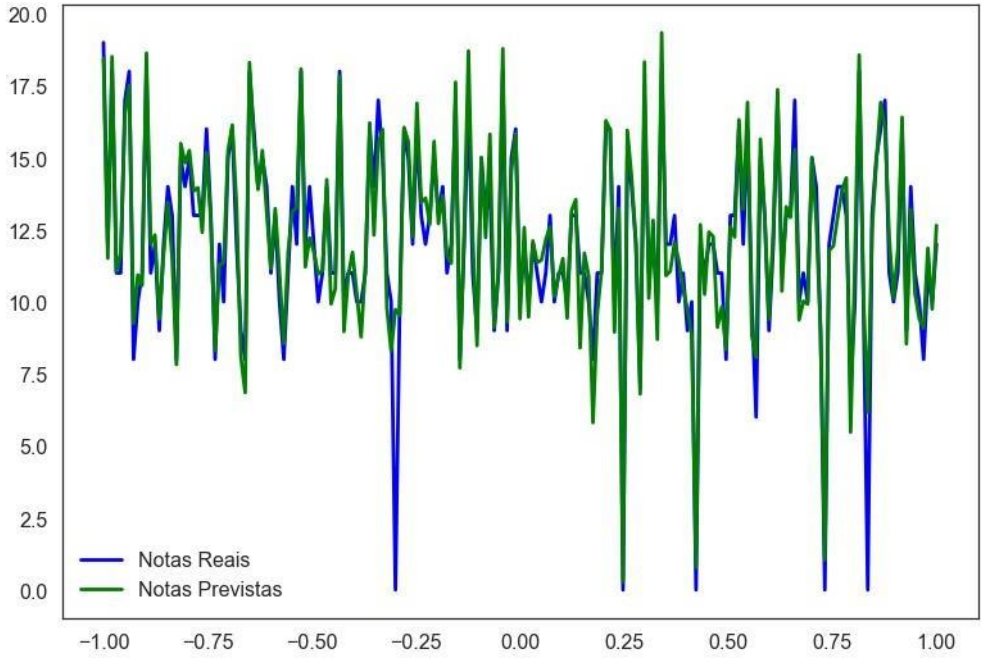
5.2.3.3 Modelo utilizando todos os atributos

Por último criamos um modelo com regressão linear múltipla utilizando todos os atributos da base de dados incluindo a primeira e segunda nota (G1 e G2), para este último modelo os valores de tolerance e stepsize foram respectivamente de 0.06e3 e 8e-7.

RSS = 271.268163116

Mean squared error = 1.39111878521

Figura 23 - Predição com todos os atributos



Fonte: Arquivo Pessoal

6 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

O objetivo deste trabalho foi realizar um estudo sobre a evasão escolar, que tem sido um mal que afeta todo o mundo, além disso, apresentar proposta de um auxílio na prevenção da mesma, pois existe uma grande dificuldade na definição dos causadores desta evasão, portanto deve-se considerar combater o problema de formas variadas, uma delas é conseguir melhorar o desempenho acadêmico do aluno consequentemente aumentando a motivação do mesmo, para isso o uso de aprendizado de máquina torna-se um grande aliado.

Com base nos dados coletados e dos resultados do presente trabalho, é possível apontar algumas considerações. Ainda na fase de análise dos dados é possível identificar padrões que auxiliem o profissional gestor em educação, sendo possível, por exemplo, entender que, quanto maior a média escolar maior é a diferença entre quantidade de pessoas do sexo masculino e do sexo feminino. Também foi possível notar que a grande maioria dos alunos obtiveram notas próximas à média. Este tipo de informação obtida pode gerar um poderoso retorno, pois na criação de programas de contenção da evasão, dados como estes possibilitam a melhor divisão de grupos de alunos para táticas diferentes de trabalho para cada um deles.

Conforme descrito nos capítulos anteriores, os modelos apresentados possibilitam que se alcance resultados relevantes na previsão da nota final dos alunos. Ao levarmos em consideração o uso de regressão linear com a base de dados, conseguimos uma taxa de aproximação entre nota prevista e nota real, utilizando a técnica de erro quadrático médio de 1.22, ou seja, um resultado de aproximação de valores consideravelmente alto.

Algumas perspectivas podem ser vislumbradas a partir deste estudo, como a possibilidade de futuros trabalhos, por exemplo: a investigação do peso de cada atributo em relação a nota final, ou seja, a obtenção da informação de quais atributos influenciam mais no desempenho escolar.

Com isso apresentamos uma solução alternativa para a prevenção da evasão escolar, ao mesmo tempo em que adquirimos conhecimento tanto teórico quanto prático, sobre as ferramentas e técnicas de Aprendizado de Máquina. Podendo confirmar que esta é realmente a área que escolhemos para nossa vida profissional.

7 APÊNDICE

7.1 APÊNDICE 1 – MODELO COM SCKIT-LEARN

In [ ]:

```
# Import libraries
%matplotlib inline

#import sys
#sys.path.append("../tools/")
#import plot

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from time import time

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score, f1
_score
from sklearn.linear_model import LinearRegression

sns.set_context('poster')
sns.set_style('whitegrid')
plt.rcParams['figure.figsize'] = [12, 8]
```

In [ ]:

```
# Read student data
df = pd.read_csv("student-por.csv", delimiter=';')
print("Dados carregados com sucesso!")
```

In [ ]:

```
columns_train = df.columns.difference(['G3'])
column_target = df.columns[-1]

X = df[columns_train]
y = df[column_target]
```

In [ ]:

```
label = LabelEncoder()
cat_Colums = X.dtypes.pipe(lambda X: X[X=='object']).index
for col in cat_Colums:
    X[col] = label.fit_transform(X[col])
```

In [ ]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7,
random_state=42)
```

In [ ]:

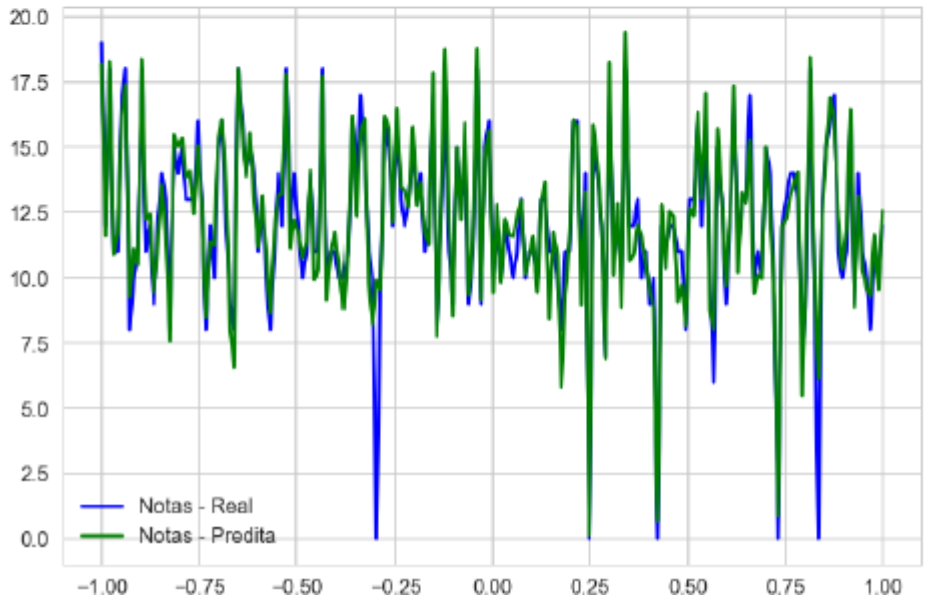
```
# Create linear regression object
reg = LinearRegression()
# Train the model using the training sets
reg.fit(X_train, y_train)

y_pred = reg.predict(X_test)
```

In [ ]:

```
# The coefficients
print('Coefficients: \n', reg.coef_)
# The mean squared error
print("Mean squared error: %.2f" % np.mean((reg.predict(X_test) - y_test) ** 2))
# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % reg.score(X_test, y_test))
```

In [10]:

```
plt.plot(np.linspace(-1, 1, len(y_test)), y_test,  label='Notas - Real', color='b')
plt.plot(np.linspace(-1, 1, len(y_pred)), y_pred,  label='Notas - Predita', color='g')
plt.legend()
plt.show()
```



7.2 APÊNDICE 2 - MODELOS SEM SICKIT-LEARN

In [ ]:

```
import pandas as pd
import numpy as np
from datetime import datetime
from sklearn.linear_model import LinearRegression
from math import sqrt, floor
from matplotlib import pyplot as plt
%matplotlib inline

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score, f1
_score
```

In [ ]:

```
# Read student data
df = pd.read_csv("student-por.csv", delimiter=';')
print("Dados carregados com sucesso!")
```

In [ ]:

```
x = df[df.columns.difference(['G3'])]
y = df['G3']
train_data, test_data, y_train, y_test = train_test_split(x, y, train_size=0.7, random_
state=42)
```

In [ ]:

```
def simple_linear_regression(input_feature, output):
    # compute the sum of input_feature and output
    input_feature_sum = float(input_feature.sum()) #10
    output_sum = float(output.sum()) #15

    #compute the number of datapoints
    N = input_feature.size #5

    # compute the product of the output and the input_feature and its sum
    product_input_output = (input_feature*output).sum() #40

    # compute the squared value of the input_feature and its sum
    squared_input_feature = (input_feature**2).sum() #30

    # use the formula for the slope
    numerator = product_input_output - (input_feature_sum * output_sum)/N #40-(150/5) =
 2,5
    denominator = squared_input_feature - (input_feature_sum * input_feature_sum)/N #30
-(150/5)
    slope = numerator/denominator

    # use the formula for the intercept
    intercept = output_sum/N - slope * input_feature_sum/N


    return (intercept, slope)
```

In [ ]:

```
grade_intercept, grade_slope = simple_linear_regression(train_data['G2'], y_train)

print("Intercept: " + str(grade_intercept))
print("Slope: " + str(grade_slope))
```

In [ ]:

```
def get_regression_predictions(input_feature, intercept, slope):
    # calculate the predicted values:
    predicted_values = intercept + slope*input_feature

    return predicted_values
```

In [ ]:

```
#Agora que podemos realizar predição de dados com o slope e o intercept,
#vamos fazer uma predição. Vamos tentar encontrar o valor estimado da nota G3
#a partir da nota G2, de acordo com o modelo de regressão estimado acima
my_G2_grade = 12
estimated_G3_grade = get_regression_predictions(my_G2_grade, grade_intercept, grade_slo
pe)
print( "A nota G3 estimada de um aluno com nota G2 = %d é %.2f" % (my_G2_grade, estimat
ed_G3_grade))
```

In [ ]:

```
#Método de mínimos quadrados
def get_residual_sum_of_squares(input_feature, output, intercept, slope):
    # First get the predictions
    predictions  = get_regression_predictions(input_feature, intercept, slope)

    # then compute the residuals (since we are squaring it doesn't matter which order y
ou subtract)
    predicted_values = output - (intercept + slope*input_feature)

    # square the residuals and add them up
    RSS = float((predicted_values**2).sum())

    return(RSS)
```

In [ ]:

```
#Agora vamos usar a função para calcular o RSS sobre os dados de treinamento
#do modelo calculado acima. Observe que para este modelo específico, o erro foi
#consideravelmente alto, o que é justificado primeiro pelo uso de somente um atributo
#e, segundo pelo fato de estarmos considerando valores inteiros, desprezando-se em sua
#maioria os valores decimais.
rss_grades = get_residual_sum_of_squares(train_data['G2'], y_train, grade_intercept, gr
ade_slope)
print('O erro RSS para predição de notas G3 baseadas na nota G2 é: ' + str(rss_grades))
```

In [ ]:

```
label = LabelEncoder()
feature_cols = df.dtypes.pipe(lambda df: df[df=='object']).index
for col in feature_cols:
    df[col] = label.fit_transform(df[col])

model_features = df.columns.difference(['G3'])
target_feature = df.columns[-1]

X = df[model_features]
y = df[target_feature]

train_data, test_data, y_train, y_test = train_test_split(X, y, train_size=0.7, random_
state=42)
```

In [ ]:

```
def get_numpy_data(data_frame, features, output):
    # select the columns of data_Frame given by the features list into the data_Frame f
eatures_frame (now including constant):
    feature_frame = data_frame[features]
    feature_frame.insert(0, 'constant', 1)

    # the following line will convert the features_Frame into a numpy matrix:
    feature_matrix = feature_frame.as_matrix()
    # assign the column of data_frame associated with the output to the Array output_ar
ray
    # the output parameter can be a pd.Series or even a literal corresponding to the co
lumn title
    if isinstance(output, pd.Series):
        output_array = output
    else:
        output_array = data_frame[output]

    # the following will convert the Array into a numpy array by first converting it to
 a list
    output_array = output_array.as_matrix()
    return(feature_matrix, output_array)
```

In [ ]:

```
feat = ['G2']
(example_features, example_output) = get_numpy_data(df, feat, 'G3') # the [] around 'sq
ft_living' makes it a list
print(example_features[0,:]) # this accesses the first row of the data the ':' indicate
s 'all columns'
print(example_output[0]) # and the corresponding output
```

In [ ]:

```
def predict_output(feature_matrix, weights):
    # assume feature_matrix is a numpy matrix containing the features as columns and we
ights is a corresponding numpy array
    # create the predictions vector by using np.dot()
    predictions = np.dot(feature_matrix, weights)

    return(predictions)
```

In [ ]:

```
def feature_derivative(errors, feature):
    # Assume that errors and feature are both numpy arrays of the same length (number o
f data points)
    # compute twice the dot product of these vectors as 'derivative' and return the val
ue
    derivative = 2*np.dot(feature, errors)

    return(derivative)
```

In [ ]:

```
def regression_gradient_descent(feature_matrix, output, initial_weights, step_size, tol
erance):
    converged = False
    weights = np.array(initial_weights) # make sure it's a numpy array
    while not converged:
        # compute the predictions based on feature_matrix and weights using your predic
t_output() function
        predictions = predict_output(feature_matrix, weights)

        # compute the errors as predictions - output
        errors = predictions-output

        gradient_sum_squares = 0 # initialize the gradient sum of squares
        # while we haven't reached the tolerance yet, update each feature's weight
        for i in range(len(weights)): # loop over each weight
            # Recall that feature_matrix[:, i] is the feature column associated with we
ights[i]
            # compute the derivative for weight[i]:
            derivative = feature_derivative(errors, feature_matrix[:, i])

            # add the squared value of the derivative to the gradient sum of squares (f
or assessing convergence)
            gradient_sum_squares += derivative*derivative

            # subtract the step size times the derivative from the current weight
            weights[i] = weights[i] - step_size*derivative

        # compute the square-root of the gradient sum of squares to get the gradient ma
gnitude:
        gradient_magnitude = sqrt(gradient_sum_squares)
        if gradient_magnitude < tolerance:
            converged = True
    return(weights)
```

In [ ]:

```
#Modelo de regressão tendo G2 como entrada única para a predição
# Let's test out the gradient descent
simple_features = ['G2']
(simple_feature_matrix, output) = get_numpy_data(train_data, simple_features, y_train)
initial_weights = np.array([-0.01, 1.])
step_size = 7e-6
tolerance = 1.0e4
```

In [ ]:

```
model1_weights = regression_gradient_descent(simple_feature_matrix, output, initial_wei
ghts, step_size, tolerance)
print(model1_weights)
```

In [ ]:

```
(test_simple_feature_matrix, test_output) = get_numpy_data(test_data, simple_features,
y_test)
```

In [ ]:

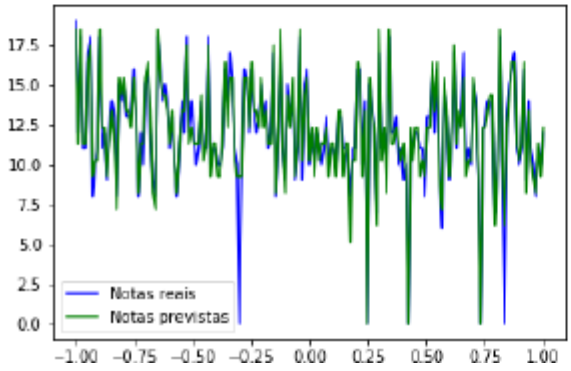
```
predictions_model1 = predict_output(test_simple_feature_matrix, model1_weights)
predictions_view = [int(floor(x)) for x in predictions_model1]
print(predictions_model1)
print(predictions_view)
print(test_output)
```

In [ ]:

```
#Cálculo de erro residual RSS para este primeiro modelo de regressão linear aonde temos
 como entrada a nota G2.
errors_model1 = predictions_model1 - test_output
rss_model1 = (errors_model1*errors_model1).sum()
print('RSS =', rss_model1)
print('Mean squared error =', np.mean(errors_model1*errors_model1))
```

In [21]:

```
plt.plot(np.linspace(-1, 1, len(test_output)), test_output,  label='Notas reais',
color='b')
plt.plot(np.linspace(-1, 1, len(predictions_model1)), predictions_model1,  label='Notas
 previstas', color='g')
plt.legend()
plt.show()
```



In [ ]:

```
#Modelo de regressão tendo G1 e G2 como entrada para a predição
model_features = ['G1', 'G2']
(feature_matrix, output) = get_numpy_data(train_data, model_features, y_train)
initial_weights = np.array([-0.01, 1., 1.])
step_size = 7.0e-6
tolerance = 1.0e1
```

In [ ]:

```
model2_weights = regression_gradient_descent(feature_matrix, output, initial_weights, s
tep_size, tolerance)
print(model2_weights)
```

In [ ]:

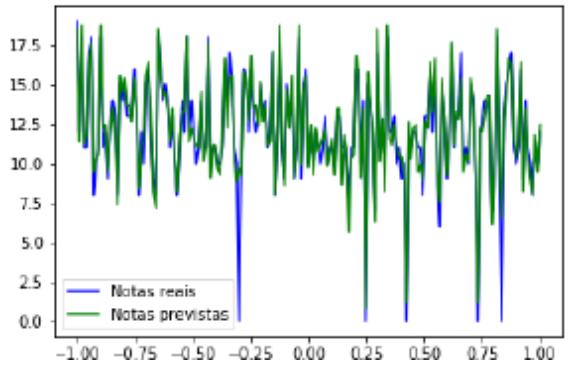
```
(multiple_feature_matrix, test_output) = get_numpy_data(test_data, model_features, y_te
st)
predictions_model2 = predict_output(multiple_feature_matrix, model2_weights)
print(predictions_model2)
print(test_output)
```

In [ ]:

```
errors_model2 = predictions_model2 - test_output
rss_model2 = (errors_model2*errors_model2).sum()
print('RSS =', rss_model2)
print('Mean squared error =', np.mean(errors_model2*errors_model2))
```

In [26]:

```
plt.plot(np.linspace(-1, 1, len(test_output)), test_output,  label='Notas reais',
color='b')
plt.plot(np.linspace(-1, 1, len(predictions_model2)), predictions_model2,  label='Notas
 previstas', color='g')
plt.legend()
plt.show()
```



In [ ]:

```
weights = regression_gradient_descent(feature_matrix, output, initial_weights, step_siz
e, tolerance)
print(weights)
```

In [ ]:

```
#Modelo de regressão múltipla (todos os atributos como entrada para a predição)
model_features = X.columns.difference(['G3'])
(feature_matrix, output) = get_numpy_data(train_data, model_features, y_train)
initial_weights = np.append(np.array([-.001]), [0.0001] * (train_data.shape[1]))
step_size = 8e-7
tolerance = 0.06e3
```

In [ ]:

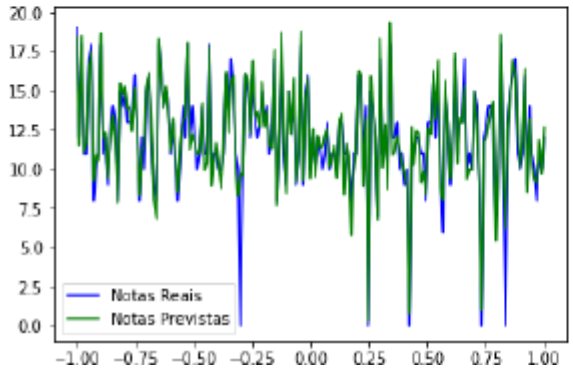
```
(mult_matrix, mult_output) = get_numpy_data(test_data, model_features, y_test)
predictions_model = predict_output(mult_matrix, weights)
print(predictions_model)
print(mult_output)
```

In [ ]:

```
errors = predictions_model - mult_output
rss = (errors*errors).sum()
print('RSS =', rss)
print('Mean squared error =', np.mean(errors*errors))
```

In [31]:

```
plt.plot(np.linspace(-1, 1, len(mult_output)), mult_output,  label='Notas Reais',
color='b')
plt.plot(np.linspace(-1, 1, len(predictions_model)), predictions_model,  label='Notas P
revistas', color='g')
plt.legend()
plt.show()
```



8 REFERÊNCIAS

Michalski, Ryszard, S.; Kodratoff, Yves. MACHINE LEARNING An Artificial Intelligence Approach Volume III. MORGAN KAUFMANN PUBLISHERS, INC. 1990, 814 p.

Shalev-Shwartz, Shai.; Ben-David, Shai. Understanding Machine Learning, From Theory to algorithms, Cambridge University Press.2014.449 p. Nilsson, Nils. J.: Introduction to machine learning. Stanford University, 1998.

Camilo, Cássio. O.; Da Silva. Joao Carlos: Mineração de dados: Conceitos, Tarefas, Métodos e Ferramentas. Instituto de Informática Universidade Federal de Goiás, 2009.

De Lima, Bruno Vicente Alves.; Machado, Vinicius Ponte.; Veras, Rodrigo de Melo Souza.: Abordagem semi-supervisionada para Rotulação de dados. Departamento de Computação Universidade Federal do Piauí, 2013.

Thomé, A. C. G.; Marques, Airam C. P. B.: Inteligência Computacional. Núcleo de Computação eletrônica Universidade Federal do Rio de Janeiro. 2003.

Shalev-Shwartz, Shai. Ben-David, Shai.; Understanding Machine Learning: From Theory to Algorithms, Cambridge, Cambridge University Press, 2014.

J. Nilsson, Nils.; Introduction to Machine Learning. Department of Computer Science Stanford.1998.

Oliveira Camilo, Cassio. Carlos da Silva, João.; Mineração de Dados: Conceitos, Tarefas, Métodos e Ferramentas, Instituto de Informática Universidade Federal de Goiás, 2009.

Tanaka, Asterio.; Banco de dados distribuídos e Datawarehousing – Introdução a Data Mining e arvores de decisão. Unirio,2012.

A. C. G. Thomé. C. P. B. Marques.; Inteligência artificial – Introdução ao MatLab, Universidade Federal do Rio de Janeiro, 2003.

Rojas, Raul.; Neural Networks A Systematic Introduction, Springer-Verlag, Berlin, Chen, Ken.; K-means Clustering, The university of Manchester, Manchester. 2012 1996.

Enos, Daniel. Medeiros, Heitor.; Inteligência Artificial Aplicada, Universidade Federal do Rio Grande do Norte, 2013.

Linden, Ricardo.; Técnicas de agrupamento, Revista de Sistemas de Informação da FSMA nº 4, 2009.

A. Bullinaria, John.; Introduction to Neural Networks, 2nd Year UG, MSc in Computer Science, 2004.

Cortez, Paulo. Silva, Alice.; Using Data Mining to predict secondary school student performance, Dep. Information Systems/Algoritmi R&D Centre University of Minho,2008.

Mckinney, Wes.; Python for data analysis, Published by O'Reilly Media, Inc,2013.