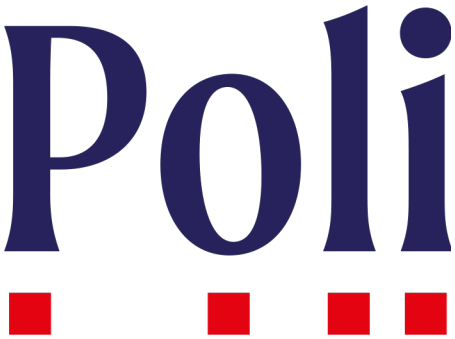

MÉTODOS DE INTELIGÊNCIA COMPUTACIONAL APLICADOS À ANÁLISE PREDITIVA DA PERFORMANCE DE ESTUDANTES DO SEGUNDO GRAU

Gabriel Ponte, Vinicius Silva

Trabalho desenvolvido na disciplina de Inteligência Computacional, COC361, na Universidade Federal do Rio de Janeiro.

Professor: Alexandre Evsukoff

Rio de Janeiro
Março de 2022

Lista de Figuras

Lista de Tabelas

Sumário

1 Introdução

Um dos principais fatores para um país ascender econômicamente e socialmente é a educação. Além de contribuir com a formação intelectual do individuo, ela também contribui para a formação coletiva dos estudantes e promove a transformação do meio social para o bem comum.

De acordo com [19], em 2016, a taxa de evasão escolar em Portugal era de cerca 40% para alunos com idade entre 18 e 24 anos, enquanto para o resto da Europa a taxa era de aproximadamente 15%. Reprovações em disciplinas básicas como Português e Matemática são um dos causadores para tamanha evasão, pois uma má absorção desse conteúdo prejudica o resto da jornada escolar que depende bastante dessa base.

Sendo assim, utilizaremos dados obtidos a partir de um formulário respondido voluntariamente por alunos de dois agrupamentos de escolas lusitanas [27] na disciplina de Português, para que, com o auxílio de ferramentas tecnológicas, possamos estimar a nota final do aluno a partir de suas informações pessoais.

2 Dataset e Tecnologia

Baseando-se em [8], optamos por utilizar o dataset disponibilizado em [7]. Em sala de aula foram aprendidas diversas técnicas de Inteligência Computacional. O objetivo deste trabalho é apresentar as técnicas estudadas e utilizá-las para tentar prever qual será a nota final obtida pelo aluno na disciplina, a partir de suas informações pessoais.

2.1 Descrição dos dados

O dataset une em 33 variáveis, informações sobre 649 alunos que responderam o formulário. Os pesquisadores selecionaram atributos que poderiam influenciar na vida acadêmica dos alunos, de forma direta ou indireta. Dentre as 33 variáveis, temos que 17 são do tipo objeto e 16 são do tipo inteiro de 64 bit. Apesar de serem de tipos diferentes, todas são variáveis discretas, isto é, armazenam valores do tipo inteiro (quantitativa discreta), categóricos (qualitativa nominal) ou ordinais (qualitativa ordinal).

Variáveis quantitativas discretas são aquelas que possuem valores distantes ao longo de uma escala, já as variáveis qualitativas nominais são atributos em que não é possível estabelecer uma ordem entre as categorias e as variáveis qualitativas ordinais são atributos no qual é possível estabelecer algum tipo de ordem ou grau. Como exemplo de cada uma dessas categorias, temos a idade para as variáveis quantitativas discretas, o sexo e profissão para qualitativas nominais e grau de escolaridade para variáveis qualitativas ordinais.

As informações de cada um dos atributos estão descritas abaixo.

1. **School** - Binário - Escola onde o aluno estudou, sendo *Gabriel Pereira (GP)* ou *Mousinho da Silveira (MS)*.

2. **Sex** - Binário - Sexo do aluno, sendo *Masculino (M)* ou *Feminino (F)*.

3. **Age** - Quantitativa discreta - Idade do aluno, podendo variar entre 15 e 22 anos.

4. **Address** - Binário - Tipo de região onde o aluno mora, podendo ser *Urbano (U)* ou *Rural (R)*.

5. **Famsize** - Binário - Quantidade de pessoas que moram em sua casa, sendo *LE3:* ≤ 3 ou *GT3:* > 3.

6. **Pstatus** - Binário - Indica se os pais do estudante moram *Juntos (T)* ou *Separados (A)*.

7. **Medu** - Qualitativa ordinal - Nível de educação da mãe do aluno. Este nível pode ser 0 para *Nenhuma educação*, 1 para *Ensino Fundamental 1*, 2 para *Ensino Fundamental 2*, 3 para *Ensino Médio* e 4 para *Ensino Superior*.

8. **Fedu** - Qualitativa ordinal - Nível de educação da pai do aluno. Este nível pode ser 0 para *Nenhuma educação*, 1 para *Ensino Fundamental 1*, 2 para *Ensino Fundamental 2*, 3 para *Ensino Médio* e 4 para *Ensino Superior*.

9. **Mjob** - Qualitativa nominal - Tipo de trabalho da mãe do aluno. Que apresenta as seguintes opções: *Professor*, *Área de Saúde*, *Serviço Civil (ex: administrativo ou policia)*, *"Do Lar"*, *Outros*.

10. **Fjob** - Qualitativa nominal - Tipo de trabalho do pai do aluno. Que apresenta as seguintes opções: *Professor*, *Área de Saúde*, *Serviço Civil (ex: administrativo ou policia)*, *"Do Lar"*, *Outros*.

11. **Reason** - Qualitativa nominal - Razão para estudar na escola. Dado por: *Perto de Casa*, *Reputação da Escola*, *Preferência de Curso*, *Outros*.

12. **Guardian** - Qualitativa nominal - Responsável do aluno, podendo ser a *Mãe*, o *Pai*, ou *Outro*.

13. **Traveltime** - Qualitativa Ordinal - Tempo de deslocamento até a escola, podendo ser 1 para menos de 15 min, 2 para $15 - 30$ min, 3 para $30 - 60$ min, 4 para mais que 60 min.

14. **Studytime** - Qualitativa Ordinal - Tempo de estudo semanal, podendo ser 1 para menos de 2 horas, 2 para $2-5$ horas, 3 para $5-10$ horas ou 4 para mais que 10 horas.

15. **Failures** - Qualitativa Ordinal - Número n de reprovações, sendo 0 para nenhuma reprovação, 1 para uma reprovação, 2 para duas reprovações ou 3 para mais de duas reprovações.

16. **Schoolsup** - Binário - Se recebe suporte financeiro escolar, sendo *Sim* ou *Não*.

17. **Famsup** - Binário - Se recebe suporte financeiro familiar, sendo *Sim* ou *Não*.

18. **Paid** - Binário - Se paga aulas particulares, sendo *Sim* ou *Não*.

19. **Activities** - Binário - Se realiza atividades extracurriculares, sendo *Sim* ou *Não*.

20. **Nursery** - Binário - Se já frequentou a enfermaria da escola, sendo *Sim* ou *Não*.

21. **Higher** - Binário - Se deseja realizar ensino superior, sendo *Sim* ou *Não*.

22. **Internet** - Binário - Se possui acesso à internet em seu domicilio, sendo *Sim* ou *Não*.

23. **Romantic** - Binário - Se está em um relacionamento amoroso, sendo *Sim* ou *Não*.

24. **Famrel** - Qualitativa Ordinal - Qualidade da relação familiar, sendo 1 muito ruim, 2 ruim, 3 razoável, 4 bom ou 5 excelente.

25. **Freetime** - Qualitativa Ordinal - Tempo livre depois da escola, sendo 1 muito ruim, 2 ruim, 3 razoável, 4 bom ou 5 excelente.

26. **Goout** - Qualitativa Ordinal - Vida social, saída com amigos, sendo 1 muito ruim, 2 ruim, 3 razoável, 4 bom ou 5 excelente.

27. **Dalc** - Qualitativa Ordinal - Consumo diário de álcool, sendo 1 nada ou muito pouco, 2 pouco, 3 razoável, 4 alto ou 5 muito alto.

28. **Walc** - Qualitativa Ordinal - Consumo de álcool no final de semana, sendo 1 nada ou muito pouco, 2 pouco, 3 razoável, 4 alto ou 5 muito alto.

29. **Health** - Qualitativa Ordinal - Estado de saúde atual, sendo 1 muito ruim, 2 ruim, 3 razoável, 4 bom ou 5 excelente.

30. **Abscences** - Quantitativa discreta - Número de faltas do aluno, sendo entre 0 e 93 faltas.

31. **G1** - Quantitativa discreta - Nota do primeiro período, podendo ser entre 0 e 20.

32. **G2** - Quantitativa discreta - Nota do segundo período, podendo ser entre 0 e 20.

33. **G3** - Quantitativa discreta - Nota final período, podendo ser entre 0 e 20.

Ao longo das próximas seções serão detalhados processos para o tratamento dos dados que serão utilizados, visando ter uma base de dados somente com valores numéricos.

2.2 Apresentação da tecnologia

Para o desenvolvimento deste trabalho foi utilizado a linguagem de programação de código aberto **Python 3.8**, amplamente utilizado na área de ciências de dados e inteligência computacional. Essa escolha facilitou a evolução do trabalho, já que a linguagem contém uma vasta quantidade de bibliotecas com foco na área de estudo atual e de forma gratuita. As bibliotecas utilizadas nesse trabalho foram:

- **Matplotlib** - Biblioteca para criação de gráficos e visualização de dados em geral.
- **NumPy** - Acrônimo de Numerical Python, possui diversas funções para se trabalhar com computação numérica, vetores e matrizes.
- **Pandas** - Ferramenta de análise e manipulação de dados.
- **Scipy** - Processos em diversas áreas como integração numérica, otimização e processamento de sinais. Possui rotinas de estatística.
- **Seaborn** - Trabalha em cima do Matplotlib e ajuda a melhorar a visualização dos gráficos.
- **Sklearn** - Voltado para o aprendizado de máquina.

Pela popularidade da linguagem existem diversas plataformas de desenvolvimento. Algumas que foram utilizadas e auxiliaram na construção do código foram o ambiente **Jupyter Notebook**, juntamente com a IDLE (ambiente de desenvolvimento integrado para Python) **Spyder 5.1**, ambos presentes no **Anaconda**, uma plataforma poderosa para ciências de dados e o **Google Colaboratory** que é um ambiente de desenvolvimento online.

3 Metodologia

3.1 Descrição Matemática da Metodologia

Neste tópico apresentaremos a teoria e a descrição matemática dos métodos que serão adotados no nosso trabalho.

3.1.1 Regressão Logística

A Regressão Logística é um modelo que permite classificar a saída em classes categóricas. A partir de [10], vimos que o modelo de Regressão Logística inicialmente foi dado como um modelo de classificação binária. Depois foi visto que seria possível estender este modelo para múltiplas classes, utilizando a função softmax, que realiza uma transformação exponencial normalizada da função logística.

Seja o número de classes igual a k, o vetor de entrada $x \in \mathbb{R}^{n\times 1}$, a matriz de parâmetros $W \in \mathbb{R}^{k\times n}$, o vetor de bias $b \in \mathbb{R}^{k\times 1}$ e o vetor de saída $y \in \mathbb{R}^{k\times 1}$. Segundo [12], caso a classe c for a correta, defina $y_c := 1$ e $y_j = 0\ \forall j \neq c$, ou seja, o vetor y possui todas componentes nulas com exceção do indíce da classe c que é igual a 1. O objetivo deste método é estimar y, a partir de um vetor $\hat{y}$.

3.1.1.1 Softmax

Seja $z \in \mathbb{R}^k$, a função softmax o põe numa distribuição de probabilidade, com cada valor do vetor pertencente ao intervalo $(0, 1)$, com o somatório de cada elemento deste vetor sendo igual a 1. A função softmax é definida como

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^{k} \exp(z_j)}, \quad i = 1, \ldots, k, \tag{1}$$

sendo o denominador $\sum_{j=1}^{k} \exp(z_j)$ utilizado para normalizar o vetor. Sendo assim,

$$\text{softmax}(z) = \left[\frac{\exp(z_1)}{\sum_{j=1}^{k} \exp(z_j)}, \frac{\exp(z_2)}{\sum_{j=1}^{k} \exp(z_j)}, \ldots, \frac{\exp(z_k)}{\sum_{j=1}^{k} \exp(z_j)}\right] \tag{2}$$

3.1.1.2 Regressão Softmax

Portanto, é possível calcular a Regressão Softmax. Seja w_i a coluna i de W, então a probabilidade de cada uma das saídas $\hat{y}_k$ será dada por:

$$p(y_i = 1|x) = \frac{\exp(w_i^\top x + b_i)}{\sum_{j=1}^{k} \exp(w_j^\top x + b_j)}. \tag{3}$$

A partir da Equação 2, é possível generalizar a Equação 3 da seguinte forma

$$\hat{y} = \text{softmax}\big(Wx + b\big) \tag{4}$$

3.1.1.3 Treinamento da Regressão Logística

Para o aprendizado dos pesos w e o bias b, é preciso de duas componentes, que serão explicadas a seguir. A primeira componente é uma métrica de quão próxima $\hat{y}$ esteja de y, essa distância

é denominada de função de perda denotada por L. Seja K o número de classes do problema, esta função de perda é comumente utilizada sendo a entropia cruzada, que é apresentada na Equação 5.

$$L_{EC}(\hat{y}, y) := -\sum_{i=1}^{K} y_i \log \hat{y}_i \ . \tag{5}$$

A segunda componente é o algoritmo de otimização, que atualiza iterativamente os pesos com o intuito de minimizar a função de perda L. O algoritmo padrão que é utilizado é o de gradiente descendente.

3.1.2 Classificação Bayesiana

O objetivo do classificador Naive Bayes é maximizar o cálculo da probabilidade a posteriori. Seja C o conjunto de classes, então para um conjunto de dados de entrada x, então o objetivo da saída $\hat{c}$ será dado por

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c|x) \ . \tag{6}$$

A partir de [4], temos que

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)} \ , \tag{7}$$

onde $P(c|x)$ é a probabilidade a posteriori, $P(x|c)$ é a distribuição de probabilidade condicional, $P(c)$ a probabilidade a priori. Substituindo a Equação 7 na Equação 6, temos que o objetivo será dado por

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} \frac{P(x|c)P(c)}{P(x)} \ . \tag{8}$$

Em [13], é dito que o denominador $P(x)$ pode ser retirado, tendo em vista que estamos calculando a probabilidade a posteriori para uma determinada classe, e $P(x)$ não varia de acordo com a classe, ela possui um valor fixo.

Caso supormos que os atributos são independentes, ou seja, $P(x_i|c)$ é independente para todo $i \in \{1, \ldots, n\}$, então

$$P(x|c) = P(x_1|c) \cdot P(x_2|c) \cdot \ldots \cdot P(x_n|c) \ . \tag{9}$$

Sendo assim, substituindo a Equação 9 na Equação 8 e removendo o denominador, temos que o objetivo do problema é dado como

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{i=1}^{n} P(x_i|c) \ . \tag{10}$$

De acordo com [10], o modelo mais utilizado para a probabilidade condicional $P(x_i|c)$ é estimado como sendo uma distribuição normal monovariável dada pela seguinte fórmula:

$$P(x_i|c) \approx \frac{1}{\sqrt{2\pi\hat{\sigma}_{ij}^2}} \exp\left(- \frac{(x_i - \hat{\mu}_{ij})^2}{2\hat{\sigma}_{ij}^2} \right), \tag{11}$$

onde os parâmetros $\hat{\mu}_{ij}$ e $\hat{\sigma}_{ij}^2$ são estimados como sendo a média e a variância, respectivamente, do conjunto de treinamento da variável x_i nos registros da classe c_j.

3.1.3 Árvore de decisão

Em [22], vimos que o objetivo de se utilizar as árvores de decisão é de realizar uma classificação mais intuitiva e popular, principalmente na área de medicina, já que o classificador poderia imitar a forma que o médico pensa.

3.1.3.1 Construção da árvore de decisão

A metodologia de construir a árvore é dada da seguinte forma:

- Comece da raíz
- Aumente a profundidade da árvore, dividindo os seus atributos um por um. A forma de escolher a divisão dos atributos é dada a partir da impureza do nó.
- Atribuir aos nós folha o voto majoritário na folha.
- Prunar a árvore para evitar overfitting. Existem diversas formas de prunar a árvore, como por exemplo não ter mais atributos para dividir, todos exemplos possuírem a mesma amostra ou atingir um limite de profundidade máximo.

3.1.3.2 Ganho de Informação

Para dividir os atributos, foi dito que é preciso informar uma divisão a partir da impureza do nó, isso é feito a partir da impureza que esse split apresentará. Seja I a impureza e p a probabilidade de um evento ocorrer, então a impureza é dada a partir das seguintes regras:

1. Todas probabilidades devem ser não-negativas e caso a probabilidade de um evento seja igual a um, então nenhuma informação será recebida, portanto $I(p) \geq 0$, $I(1) = 0$.
2. A probabilidade de dois eventos ocorrerem é igual a soma de suas probabilidades, ou seja, $I(p_1 \cdot p_2) = I(p_1) + I(p_2)$.
3. Pequenas mudanças na probabilidade implicam em pequenas mudanças na impureza.

Existem duas funções muito utilizadas que satisfazem essas propriedades:

1. Entropia: Dada por

$$\mathrm{H}(\mathbf{p}) := \sum_{j=1}^{J} p_j I(p_j) = -\sum_{j} p_j \log_2 p_j, \tag{12}$$

   veja que quanto menor a Entropia, melhor.

2. Índice de Gini: A impureza de Gini é dada por

$$\mathrm{I}_G(\mathbf{p}) := \sum_{j=1}^{J} \left(p_j \sum_{k \neq j} p_k \right) = 1 - \sum_{j=1}^{J} p_j^2 \tag{13}$$

Seja I uma impureza, tal que I possa pertencer a $\{\mathrm{H}, \mathrm{I}_G\}$, então o ganho de informação $G(T, x_i)$ de um nó T e com um atributo x_i é dado:

$$G(T, x_i) = I(T) - I(T|x_i) \ , \tag{14}$$

onde $I(T)$ é a impureza do nó pai T, e $I(T|x)$ é igual a soma das entropias dos filhos.

Porém, a Equação 14 do Ganho de Informação apresenta problemas, tendo em vista que ela faz com que o método tenda a fazer muitas partições e divisões. Sendo assim, definimos a informação do split $S_i(T, x_i)$ dada por

$$S_i(T, x_i) = -\sum_{j=1}^{J} \frac{|T_j|}{|T|} \log \left(\frac{|T_j|}{|T|} \right) \ , \tag{15}$$

onde $|S_j|$ é a cardinalidade da divisão j. Com isso, é possível trabalhar com o Razão do Ganho, dada por G_R, sendo

$$G_R(T, x_i) = \frac{G(T, x_i)}{S_i(T, x_i)} \ , \tag{16}$$

e a partir disso, é desejável possuir um ganho maior e uma informação do split menor.

3.1.4 Random Forest

Como forma de melhorar o algoritmo da árvore de decisões para um grande conjunto de dados, é apresentado o classificador Random Forest. A partir de [9], o Random Forest utiliza árvores para o seu aprendizado, porém, seleciona diversas árvores, com um número menor de atributos. Após isso, cada árvore apresentará uma previsão de classe e a classe que possuir mais votos se tornará a previsão do modelo.

Entrada: $\mathcal{D}, J, m, n, k$
Saída: $\hat{y}$

Defina $\tilde{Y} := \text{zeros}(n, J)$;
para j = *1,...,m* **faça**
    Escolha uma amostra $\mathcal{D}_j$ com k atributos;
    Construa uma árvore de decisão com a amostra $\mathcal{D}_j$;
    Receba o output $\tilde{y}$;
    para i = *1,...,n* **faça**
        Coloque $\tilde{y}_i$ na forma **one-hot vector** e adicione-o em $\tilde{Y}$;
Defina $\hat{y} := \text{zeros}(n, 1)$ **para** i = *1,...,n* **faça**
    $\hat{y}[i] := \text{argmax}\, Y[i, :]$;

Algoritmo 1: Random Forest

Seja J o número total de classes, $x_i \in \mathbb{R}^p$ a amostra i do conjunto de treino, y_i a resposta e um conjunto de dados $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, k o número de atributos a serem escolhidos e m o número de árvores da floresta a serem escolhidas. Então o procedimento para realizar esse classificador é apresentado no Algoritmo 1. Para facilitar a visualização do algoritmo, utilizamos uma imagem de [20], na Figura 1 que também apresenta o funcionamento do Random Forest.

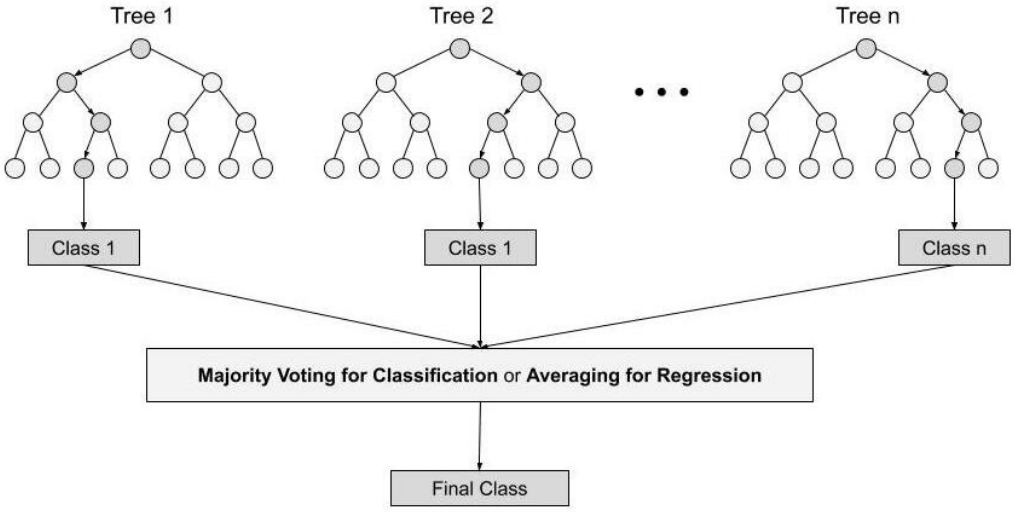

Figura 1: Exemplo de Random Forest para n árvores

3.1.5 Gradient Boosting

Segundo [21], o Boosting é um modelo que começou respondendo uma pergunta de Michael Kearns, que desejava saber se seria possível tornar um algoritmo fraco de aprendizado para um algoritmo com forte aprendizado, e esta será a metodologia aplicada ao Gradient Boosting.

Seja uma função de avaliação $L(y, F(x, \alpha_i, \rho_i))$, podendo ser dada como por exemplo erro quadrático e erro absoluto. Em [11] é mostrado um problema que, em geral, é muito díficil de ser resolvido, veja a Equação 17. O algoritmo gradient boosting, veio para solucionar este problema, com a seguinte ideia, ao invés tentar resolver um problema muito difícil, a metodologia decompõe o mesmo em subproblemas menores, fáceis de resolver, facilitando assim, a sua convergência. Seja h uma função de base (também chamada de *weak learner* ou *base learner*) e seja $F_m(x_i, \beta, \alpha_m) := \sum_{m=1}^{M} \beta_m h(x_i, \alpha_m)$, então a solução ótima do problema $F^*(x)$ é encontrada a partir do ajuste de parâmetros do seguinte problema de otimização

$$F^*(x) = \underset{\beta, \alpha_m}{\operatorname{argmin}} \sum_{i=1}^{N} L\Big(y_i, F_m(x_i, \beta, \alpha_m)\Big) \ . \tag{17}$$

Porém, como dito anteriormente, em geral é inviável resolver esse problema. Uma opção que podemos realizar é utilizar um algoritmo guloso em M etapas até atingir um critério de parada. Sendo assim, podemos utilizar um aproximador $F_{m-1}(x)$, e a função $\beta_m h(x; \alpha_m)$ como um passo guloso em direção a $F^*(x)$.

$$(\beta_m, \alpha_m) := \underset{\beta, \alpha}{\operatorname{argmin}} \sum_{i=1}^{N} L(y_i, F_{m-1}(x_i) + \beta h(x_i; a)) \ , \tag{18}$$

e

$$F_m(x) := F_{m-1}(x) + \beta_m h(x; \alpha_m) \ . \tag{19}$$

Seja $g_m(x_i)$ o gradiente em x_i na etapa m, então

$$-g_m(x_i) := -\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} \ , \tag{20}$$

porém, este gradiente é a menor direção de descida para x_i, e não para todo vetor x. Sendo assim, vamos generalizar, na Equação 21 e encontrar a parametrização $h(x_i, \alpha_m)$ que possua a inclinação mais paralela possível de $-g_m(x)$ sendo

$$\alpha_m := \underset{\alpha, \beta}{\operatorname{argmin}} \sum_{i=1}^{N} \big(- g_m(x_i) - \beta h(x_i, \alpha)\big)^2 \ . \tag{21}$$

Após encontrada α_m, substituiremos $-g_m(x)$ por $h(x, \alpha_m)$. Com isso, é performada uma busca na linha para obter o menor valor da função de perda, sendo

$$\rho_m := \underset{\rho}{\operatorname{argmin}} \sum_{i=1}^{N} L(y_i, F_{m-1}(x_i) + \rho h(x_i, \alpha_m)) \ , \tag{22}$$

e a partir do passo ρ, atualizamos F_m da seguinte forma

$$F_m(x) := F_{m-1}(x) + \rho_m h(x, \alpha_m) \ . \tag{23}$$

A partir disso, é apresentado o algoritmo para encontrar a solução do problema utilizando o Gradient Boost no Algoritmo 2.

Entrada: $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$
Saída: $F_m(x)$
$F_0(x) := \text{argmin}_\rho \sum_{i=1}^{N} L(y_i, \rho)$;
para $m = 1, \ldots, M$ **faça**
  para $i = 1, \ldots, N$ **faça**
    $\tilde{y}_i := -\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}$;
  $\alpha_m := \text{argmin}_{\alpha,\beta} \sum_{i=1}^{N} \left(\tilde{y}_i - \beta h(x_i, \alpha)\right)^2$;
  $\rho := \text{argmin}_\rho \sum_{i=1}^{N} L(y_i, F_{m-1}(x_i) + \rho h(x_i, \alpha_m))$;
  $F_m(x) := F_{m-1}(x) + \rho_m h(x, \alpha_m)$;

Algoritmo 2: Gradient Boosting

Finalizando, ao invés do Gradient Boosting obter a solução numa restrição suave do problema com o melhores parâmetros otimizados, na realidade é solucionado diversas vezes um problema de minimização a partir dos mínimos quadrados seguido de um parâmetro ρ para determinar o passo, esse algoritmo é feito em múltiplas etapas e utiliza uma função de base h que é um *weak learner*. Com relação à convergência do algoritmo, se a função de avaliação L for convexa, o que é geralmente satisfeita, então é garantida a convergência para a solução ótima em M etapas, veja em [26].

3.1.6 SVM

3.1.6.1 Intuição do Problema

Considere um problema com duas classes, e que se deseje criar uma margem para separar as duas classes. Em [10], podemos obter uma figura que apresenta interpretação geométrica da margem e a separação para duas classes, veja a Figura 2. Em [23], é dito que quanto maior o tamanho da margem obtida, melhor, tendo em vista que assim os pontos estariam mais distantes do limite da mudança de classe, deixando o modelo mais estável.

Nas próximas subseções para o caso de problemas separáveis e apresentação a explicação de como obter seus modelos matemáticos. Gostaríamos de ressaltar que toda a metodologia utilizada para explicá-lo foi utilizada a partir de [10] e [23].

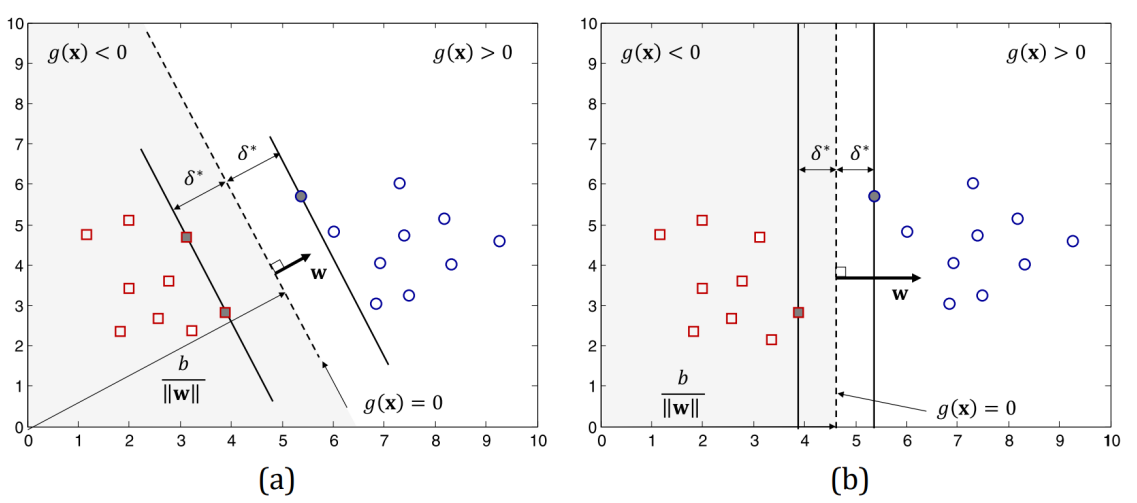

Figura 2: Interpretação geométrica da margem da separação

3.1.6.2 SVM Linear

Seja $\mathcal{D} = \{(x_1, v_1), \ldots, (x_n, v_n)\}$, sendo um conjunto de dados que possua duas classes, onde para todo $i = 1, \ldots, N$ temos

$$v_i = \begin{cases} +1, & \text{se } x_i \in C_0 \\ -1, & \text{se } x_i \in C_1 \end{cases} \tag{24}$$

Além disso, suponha que seja decidido utilizar separar as duas classes por um plano em $\mathbb{R}^n$ dado por $g(x)$ com sua normal sendo $w \in \mathbb{R}^n$ e o seu termo independente sendo $b \in \mathbb{R}$. Portanto, temos que $g(x)$ pode ser escrito como

$$g(x) := w^\top x + b \ . \tag{25}$$

Veja que caso

- $g(x) > 0$, então a região da classe estará em C_0
- $g(x) < 0$, então a região da classe estará em C_1.
- $g(x) = 0$, então será o plano que separa as duas classes.

Tendo em vista que w é normal ao plano, então o vetor unitário $\frac{w}{\|w\|_2}$ é ortogonal ao plano $g(x) = 0$. Em [10], é dito que a distância δ de um ponto qualquer em $\mathcal{D}$ até $g(x) = 0$ é dada por

$$\delta_i = \frac{v_i g(x_i)}{\|w\|_2} \ , \tag{26}$$

e como a margem é a menor distância entre todos os pontos de $\mathcal{D}$, então a solução do problema seria obter o mínimo dentre todos δ_i, o problema é que isso pode ter infinitas soluções se $g(x_i)$ for multiplicado por uma constante. Portanto, com o intuito de obter uma única solução, será utilizado o hiperplano canônico que é um hiperplano normalizado, com isso o valor da margem $\delta^\star$ é dado por

$$\delta^\star = \frac{1}{\|w\|_2} \ . \tag{27}$$

Como o objetivo de maximizar a margem $\delta^\star$, então desejamos minimizar $\|w\|_2$. Para facilitar o problema de otimização, vamos minimizar $\|w\|_2^2$. Além disso, os pontos $\tilde{x}_i$ contidos exatamente na margem de separação são chamados de vetores suporte e pela definição do hiperplano, temos que

$$g(\tilde{x}_i) = \begin{cases} +1, & \text{se } \tilde{x}_i \in C_0 \\ -1, & \text{se } \tilde{x}_i \in C_1 \end{cases} \tag{28}$$

sendo assim, podemos formular o problema de otimização da seguinte forma

$$\begin{array}{ll} \min & \frac{1}{2}\|w\|_2^2, \\ \text{s.a.} & v_i g(x_i) \geq 1, \quad i = 1, \ldots, N \end{array} \tag{29}$$

Note que $\|w\|^2 = w^\top w$, além disso esse problema é convexo, já que possui função objetivo convexa e as restrições são afins, porém o Problema 29 não está na forma de um problema de otimização convexa (veja [5]), portanto

$$\begin{array}{ll} \min & \frac{1}{2} w^\top w, \\ \text{s.a.} & 1 - v_i g(x_i) \leq 0, \quad i = 1, \ldots, N \end{array} \tag{30}$$

para cada restrição de desigualdade, associaremos à ela, multiplicadores de Lagrange $\lambda \succeq \mathbf{0}$ e a função Lagrangeana será dada da seguinte forma

$$\mathcal{L}(b, w, \lambda) = \frac{1}{2} w^\top w + \sum_{i=1}^{N} \lambda_i (1 - v_i g(x_i)) \ . \tag{31}$$

A partir de [5], o problema 29 tem solução primal e dual ótima (com gap igual zero) se satisfazer as condições KKT, apresentadas abaixo

1. Primal viável: $1 - v_i g(x_i) \leq 0, \ i = 1, \ldots, N.$
2. Dual viável: $\lambda \succeq \mathbf{0}$.
3. *Complementary slackness*: $\lambda_i (1 - v_i g(x_i)) = 0, \ i = 1, \ldots N$

4. $\nabla_{b,w}\mathcal{L}(x, w, \lambda) = 0$, portanto

$$\frac{\partial \mathcal{L}}{\partial b} = \sum_{i=1}^{N} \lambda_i v_i = 0 \ , \tag{32}$$

e

$$\frac{\partial \mathcal{L}}{\partial w} = w - \sum_{i=1}^{N} \lambda_i v_i x_i = \mathbf{0} \ , \tag{33}$$

sendo $\mathbf{0}$ um vetor nulo com N componentes.

A partir das condições KKT apresentadas acimas e da Equação 25, então podemos reescrever 31 da seguinte forma

$$\begin{aligned}
\mathcal{L}(b, w, \lambda) &= \frac{1}{2} w^\top w + \sum_{i=1}^{N} \lambda_i - \sum_{i=1}^{N} \lambda_i v_i g(x_i) \\
&= \frac{1}{2} w^\top w + \sum_{i=1}^{N} \lambda_i - w^\top \sum_{i=1}^{N} \lambda_i v_i x_i - \sum_{i=1}^{N} \lambda_i v_i b \\
&= \frac{1}{2} w^\top w + \sum_{i=1}^{N} \lambda_i - w^\top \underbrace{\sum_{i=1}^{N} \lambda_i v_i x_i}_{w} - b \underbrace{\sum_{i=1}^{N} \lambda_i v_i}_{0} \\
&= \frac{1}{2} w^\top w + \sum_{i=1}^{N} \lambda_i - w^\top w \\
&= -\frac{1}{2} w^\top w + \sum_{i=1}^{N} \lambda_i \\
&= -\frac{1}{2} \sum_{j=1}^{N} w_j^2 + \sum_{i=1}^{N} \lambda_i \\
&= -\frac{1}{2} \sum_{j=1}^{N} \left(\sum_{i=1}^{N} \lambda_i v_i x_i^{(j)} \right)^2 + \sum_{i=1}^{N} \lambda_i \\
&= -\frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \lambda_i \lambda_k v_i v_k x_i^\top x_k + \sum_{i=1}^{N} \lambda_i \ .
\end{aligned} \tag{34}$$

Segundo [5], o Lagrangeano sempre retorna um limite inferior ao problema primal, e o dual lagrangeano pode ser escrito da seguinte forma

$$\begin{aligned}
\max \quad & -\tfrac{1}{2} \textstyle\sum_{i=1}^{N} \sum_{k=1}^{N} \lambda_i \lambda_k v_i v_k x_i^\top x_k + \sum_{i=1}^{N} \lambda_i \ , \\
\text{s.a.} \quad & \textstyle\sum_{i=1}^{N} \lambda_i v_i = 0, \quad i = 1, \ldots, N \\
& \lambda \succeq \mathbf{0}.
\end{aligned} \tag{35}$$

É possível resolver o Problema 35 e encontrar $\lambda^\star$. A partir das condições KKT, temos que

$$w^\star = \sum_{i=1}^{N} \lambda_i^\star v_i x_i \ , \tag{36}$$

então a partir da solução dual, encontrar $\lambda^\star$, é fácil encontrar $w^\star$, porém, ainda é preciso saber o valor de $b^\star$. Pelas condições KKT de complementariedade, temos que se

$$\begin{cases} \lambda_i^\star > 0 \Rightarrow v_i g^\star(x_i) = 1 \ , \\ 1 - v_i g^\star(x_i) < 0 \Rightarrow \lambda_i^\star = 0 \ . \end{cases} \tag{37}$$

Segundo [23], para $\lambda_i^\star > 0$, a função $v_i g^\star(x_i)$ é denominada de vetores suporte e possuem margem escalada igual a 1. Defina $v_i = 1$, então o vetores suportes ativos são dados por

$$g(x_i) = 1 \Rightarrow \tag{38}$$

$$w^{\star^\top} x_i + b^\star = 1 \Rightarrow \tag{39}$$

$$b^\star = 1 - w^{\star^\top} x_i, \tag{40}$$

como os vetores suporte possuem as menores margens, então

$$b^\star = 1 - \min_{i:v_i=1} w^{\star^\top} x_i \ . \tag{41}$$

3.1.6.3 SVM Não Linear

Segundo [10], a formulação dual do SVM não linear pode ser facilmente encontrada a partir da formulação dual do SVM linear, utilizando um *truque* do núcleo. De acordo com o autor, é possível substituir a Equação 36 na Equação 25 e obter

$$g(x) = \sum_{i=1}^{N} \lambda_i^\star v_i x^\top x_i + b \ , \tag{42}$$

e substuir o produto interno $x^\top x_i$ por uma função não linear $k(x_i, x_j)$. Além disso, definindo $\theta := \left(\lambda_1 v_1, \ldots, \lambda_n v_n\right)$ e $V = (v_1, \ldots, v_n)$, temos que

$$\theta V^\top = \sum_{i=1}^{N} \lambda_i \ , \tag{43}$$

além disso, seja K a matriz de núcleo, então

$$\theta K \theta^\top = \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j v_i v_j k(x_i, x_j) \ . \tag{44}$$

Sendo assim, a partir de 35, podemos formular o dual do SVM não linear da seguinte forma

$$\begin{aligned} \max \quad & -\tfrac{1}{2}\theta K\theta^\top + \theta V^\top \ , \\ \text{s.a.} \quad & \mathbf{1}^\top\theta = \mathbf{0}, \\ & v_i\theta_i \geq 0, \ i = 1, \ldots, N \end{aligned} \tag{45}$$

sendo K a matriz de núcleo, $\mathbf{1}$ um vetor com todos elementos iguais a um e $\mathbf{0}$ um vetor com todos elementos iguais a zero.

3.1.6.4 Considerações finais sobre o SVM

Apresentamos o SVM original que é utilizado para problemas com duas classes. Porém, é possível generalizá-lo para múltiplas classes, veja em [2]. Seja m o número de classes, existem duas metodologias para fazer isso:

1. *One-to-One*: É feita uma combinação simples de 2 a 2 entre as m classes e o SVM é realizado $m(m-1)/2$ vezes.

2. *One-to-Rest*: Cada classe é separada de todas as outras. Ou seja, seja uma classe C_i, então o método SVM vai considerar somente duas classes, as que pertencem a C_i e as que não pertencem a C_i. Para isso, será necessário realizar o SVM m vezes.

Em [2], é apresentado um exemplo com três classes e é feita a metodologia *One-to-One* na Figura 3 e *One-to-Rest* na Figura 4.

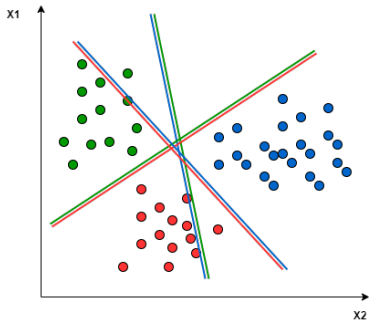

Figura 3: Interpretação gráfica utilizando One-to-One para três classes

Na Seção 3.1.6.3, é apresentada a função de núcleo dada por $k(\hat{x}, x)$. Segundo [3], existem algumas funções clássicas para k, dentre elas

1. Linear: Quando k é linear, então volta-se ao caso do SVM linear apresentado em 3.1.6.2.

$$k(\hat{x}, x) = \hat{x}x^\top \ . \tag{46}$$

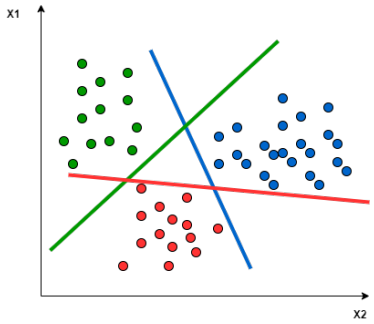

Figura 4: Interpretação gráfica utilizando One-to-Rest para três classes

2. Polinomial: Sendo uma função polinomial de grau r, dada por

$$k(\hat{x}, x) = \left(\hat{x}x^\top + c\right)^r \ , \tag{47}$$

onde c é uma constante.

3. RBF: Uma função radial, comumente utilizada quando não se tem um conhecimento prévio dos dados, com o parâmetro $\gamma > 0$

$$k(\hat{x}, x) = \exp\left(-\gamma\|\hat{x} - x\|^2\right) \ , \tag{48}$$

4. Sigmoide: Sendo uma função

$$k(\hat{x}, x) = \tanh\left(\gamma\hat{x}x^\top + c\right) \ , \tag{49}$$

com parâmetro $\gamma > 0$ e c uma constante.

A partir de [16], temos a Figura 5 onde apresenta a diferença entre as funções $k(\hat{x}, x)$.

3.1.7 Redes Neurais

3.1.7.1 Perceptron

Segundo [14], o modelo de Redes Neurais se chamava Perceptron e começou sendo utilizado com um simples neurônio, que recebia como entrada valores x_i com seus respectivos termos w_i, esses valores eram somados e ao final era somado um *bias* que é dado por $b \in \mathbb{R}$. Portanto, dado um vetor $x := (x_1, \ldots, x_n)$ e um vetor w com seus respectivos pesos tal que $w := (w_1, \ldots, w_n)$, então a soma $z \in \mathbb{R}$ será dada da seguinte forma

$$z := w^\top x + b \ , \tag{50}$$

e o valor da resposta y pode ser igual ao valor de z.

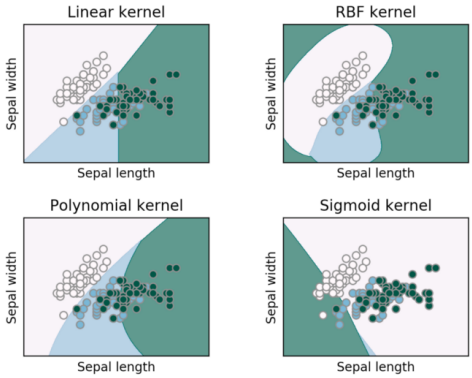

Figura 5: Diferença dos classificadores SVM

3.1.7.2 Funções de ativação

O valor da resposta y não precisa necessariamente ser igual a z. Na realidade, em muitos casos, é utilizada uma função não-linear, chamada de função de ativação $g(z)$, onde $y = g(z)$. De acordo com [14], existem diversos tipos de função de ativação, dentre eles:

1. Função sigmoide, ela mapeia todos valores de saída no intervalo de $[0, 1]$, ela é muito útil para colocar em outliers próximos de 0 ou 1.

$$g(z) = \frac{1}{1 + \exp(-z)} \tag{51}$$

2. Função Tanh, uma função similar da sigmoide, porém , de acordo com [14], geralmente apresenta resultados melhores que a sigmoide. Os valores dessa função variam pertencem ao intervalo $[-1, 1]$.

$$y = g(z) = \tanh(z) = \frac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)} \tag{52}$$

3. Função ReLU, uma função muito utilizada, e que se aproxima da função linear. Ela possui o valor da função linear, caso $z \geq 0$, e caso contrário seu valor é igual a zero.

$$y = g(z) = \text{ReLU}(z) = \max\{z, 0\} \tag{53}$$

A partir de [15], podemos observar a Figura 6, que apresenta as funções de ativação que foram citadas acima. Enquanto as curvas sigmoide e tanh são curvas mais suaves, a curva ReLU é mais brusca. Apresentar uma curvatura mais suave é bom por ter derivadas bem definidas em todos os pontos, porém valores de z muito altos deixam o valor próximo de 1, onde o gradiente é próximo de zero e isso atrapalha no aprendizado, já na curva ReLU, existe um ponto onde o gradiente não é definido, porém seu valor será diferente de zero para valores altos de z.

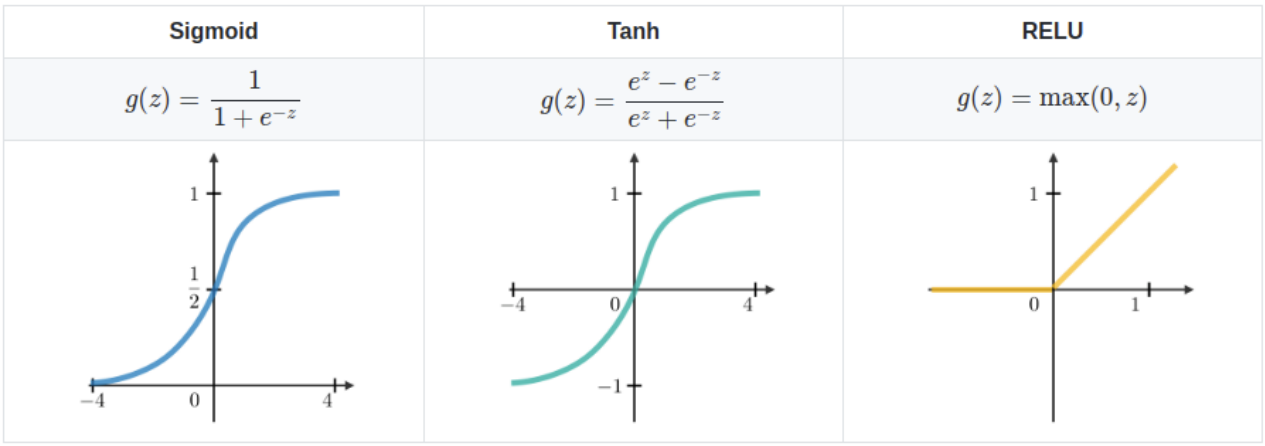

Figura 6: Tipos de funções não-lineares de ativação

3.1.7.3 Perceptron de Múltiplas Camadas (MLP)

Em [18], foi demonstrado que o Perceptron não conseguiria calcular uma função simples de XOR e assim foi visto que seria necessário possuir mais camadas no modelo. O modelo de perceptrons de múltiplas camadas é chamado de MLP e ele é constituído da seguinte forma: uma camada de entrada x, camadas escondidas h e a camada de saída y. Em sua arquitetura padrão, cada camada é totalmente conectada com a camada seguinte, veja a Figura 7 retirada de [14].

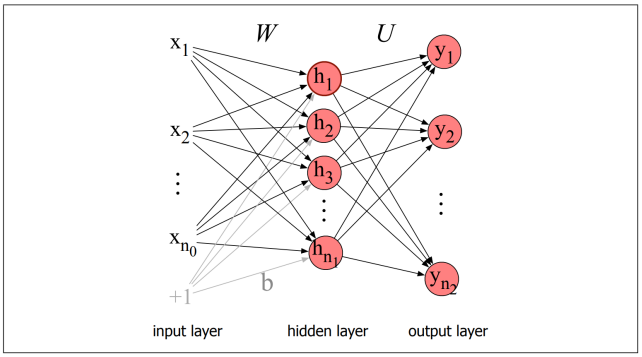

Figura 7: Exemplo de MLP com uma camada escondida

Seja uma rede com três camadas possuindo a camada de entrada sendo a camada 0 com n_0 nós, a camada 1 sendo a camada escondida com n_1 nós e a camada 2 sendo a camada de saída com n_2 nós e seja $g(z)$ uma função de ativação. Portanto, temos que $x \in \mathbb{R}^{n_0}$, $h \in \mathbb{R}^{n_1}$ e $y \in \mathbb{R}^{n_2}$. Portanto, podemos definir a matriz de pesos $W \in \mathbb{R}^{n_1 \times n_0}$

$$h = g(Wx + b) \ , \tag{54}$$

além disso, seja U uma matriz de pesos da camada última camada escondida (neste caso, a camada 1, portanto $U \in \mathbb{R}^{n_2 \times n_1}$). De acordo com [14], ocorrem casos na literatura que não se adiciona o bias neste caso, portanto para simplificar o modelo a saída intermediaria z será dada por

$$z = Uh \ . \tag{55}$$

Foi dito que z é uma saída intermediária, tendo em vista que z é um vetor com números reais e para classificação é preciso de um vetor de probabilidade. Sendo assim, para normalizar o vetor e colocá-lo numa distribuição de probabilidade, vamos utilizar a mesma técnica da Regressão Logística, apresentada na Seção 3.1.1, que utiliza a Equação 2 que apresenta a função softmax, e é por esse motivo que é dito que a rede neural é como uma regressão logística em diversas camadas.

Seja a^i a saída da camada i, W^i os pesos da camada i e z^i a combinação dos pesos e do bias na camada i, b^i o bias na camada i e g^i a função de ativação da camada i. Sendo assim, o Algoritmo 3 apresenta para resolver um problema com n camadas escondidas.

Entrada: x
Saída: $\hat{y}$
$a^0 := x$;
para $i = 1, \ldots, n$ **faça**
    $z^i := W^i a^{i-1} + b^i$;
    $a^i := g^i(z^i)$;
$\hat{y} := a^n$;

Algoritmo 3: MLP com n camadas

É preciso ressaltar que, ao possuir múltiplas camadas, é preciso utilizar uma função de ativação não linear, caso contrário o problema seria equivalente a possuir somente uma camada. **Prova:** Sejam as camadas $k-1$ e k com seus respectivos a, W, z, b e admita que g seja linear, ou seja $a^k := z^k$. Então $z^k := W^k(a^{k-1}) + b^k$, porém como $a^k = z^k$, então temos que $z^k := W^k(W^{k-1}a^{k-2} + b^{k-1}) + b^k$. Defina $\tilde{b} := W^k b^{k-1} + b^k$ e $\tilde{W} := W^k W^{k-1}$, então $z^k := \tilde{W}a^{k-2} + \tilde{b}$ e podemos realizar essa mesma operação k vezes e assim obtemos $z^k := \hat{W}a^0 + \hat{b}$, como $a^0 := x$, então $z^k := \hat{W}x + \hat{b}$, sendo assim equivalente a uma rede com somente uma camada.

3.1.7.4 Treinamento das Redes Neurais

O objetivo das redes neurais é ter um procedimento de treinamento para W^i e b^i em cada camada i, com o intuito de se aproximar ao valor real de y.

Primeiramente, para treinar as redes neurais, é preciso definir uma função de perda L que modele a distância entre a saída do sistema e a saída desejada. Seja K o número de classes do problema, segundo [14], assim como na Regressão Logística, normalmente se utiliza a função de perda sendo a entropia cruzada, apresentada na Equação 5.

Em seguida, são encontrados parâmetros que minimizem a função de perda. Isso geralmente é feito utilizando o algoritmo de otimização do gradiente descendente. Para fazer isso é preciso saber o gradiente da função de perda, porém em redes neurais com diversos parâmetros e camadas, é dificil calcular a derivada parcial de um peso em camadas iniciais, quando a perda ocorrer muitas camadas depois. De acordo com [14], para resolver isso, o algoritmo utiliza a retropropagação do erro e a retropropagação da diferenciação.

3.2 Descrição Experimental da Metodologia

3.2.1 Pré-processamento

Para conseguirmos prever a nota da última prova de um aluno a partir das variáveis, seguimos algumas etapas que envolvem a manipulação de dados e escolha de diferentes modelos de aprendizado de máquina supervisionado.

Como queremos que todas as variáveis sejam numéricas, a primeira coisa que precisamos fazer é analisar a base de dados e realizar as devidas manipulações de acordo com a sua classificação. Nesse trabalho as modificações foram feitas seguindo o critério:

- **Variáveis binárias** - Substituímos os rótulos das variáveis para 0 ou 1.
- **Variáveis qualitativas nominais** - Transformar em variáveis *dummy*.

As variáveis binárias são um caso especial de variáveis qualitativas nominais, na qual possuem apenas dois tipos de valores possíveis, como por exemplo se a pessoa tem ou não acesso a internet. Levando em consideração que a maioria dos modelos de classificação trabalham com valores numéricos, alteramos os atributos para serem 0 ou 1, possibilitando utilizar a mesma informação nos modelos, mas com uma codificação diferente.

As variáveis qualitativas nominais não possuem uma relação de ordem ou grau. Então, caso elas possuam mais de dois valores distintos, sua substituição não pode ser dada por um conjunto de número inteiros crescentes, por exemplo. Sendo assim, tem-se a necessidade de transformá-las em variáveis *dummy*, a partir da codificação one-hot.

A codificação one-hot realiza uma transformação das variáveis aumentando sua dimensão e binarizando-a. Seja a uma variável que possua n grupos e seja x_δ a variável x transformada pela codificação one-hot. Portanto, $x_\delta \in \{0,1\}^n$, e cada coluna de x_δ é denotada sendo uma variável

dummy. Nesta codificação, cada variável *dummy* representa um grupo, e caso o grupo seja ativo, então seu valor será dado por 1 e caso contrário seu valor será igual a 0, representando a negação do valor específico. Na Tabela 1, é apresentado um exemplo onde temos uma variável nominal com 4 grupos que depois de sua codificação one-hot, se tornam 4 variáveis *dummy*.

		Codificação One-hot			
Nome	Profissão	medico	professor	motorista	musico
João	Médico	1	0	0	0
Maria	Professor	0	1	0	0
Isabela	Motorista	0	0	1	0
José	Músico	0	0	0	1

Tabela 1: Exemplo de codificação one-hot

Após codificar as variáveis para que todas tenham atributos numéricos, passamos para a parte de análise individual. Observamos algumas estatísticas das variáveis quantitativas discretas e qualitativas ordinais, como a média, mediana, desvio padrão, visando identificar o comportamento e o tipo de distribuição probabilística que os dados obedecem. O estudo dessas estatísticas é importante pois auxiliará a encontrar valores aberrantes do dataset que pode deixar os modelos tendenciosos, prejudicando a predição.

Além da análise individual das variáveis é muito importante a parte da análise multivariável, na qual identificamos através da matriz de correlação e de coeficientes de correlação, qual é a influência das variáveis entre si. Uma variável com alto nível de correlação com outra pode ser descartada em alguns casos, como quando elas possui o mesmo tipo de informação, isso ajuda a diminuir a complexidade necessária. No entanto, escolher qual das variáveis descartar pode influenciar no resultado do modelo.

Padronizar o conjunto de dados também é muito importante, pois frequentemente os datasets contém variáveis em escalas diferentes, o que pode ser prejudicial para os modelos por ter atributos desproporcionalmente distantes uns dos outros. Padronização é indicado para variáveis que possuem uma distribuição simétrica e que não tenha outliers. No entanto para distribuições assimétricas é possível fazer uma transformação logarítmica antes da padronização para deixar a distribuição mais simétrica. No caso de outliers, é possível retira-los da base de dados antes ou então utilizar um tipo de padronização robusta a outliers.

Segundo [10], as análises estatísticas individuais são boas para encontrar prováveis *outliers*, entretanto a melhor forma para detectar uma amostra aberrante é através das abordagens baseadas em distribuições ou distâncias. Nós utilizamos a abordagem baseada em distâncias que utiliza função de distância ou de similaridade para determinar os registros mais afastados

do dataset, essa abordagem considera todas as variáveis, portanto identifica uma amostra atípica em relação ao resto.

Além disso, nos nossos experimentos realizamos a padrozinação pelo Z-score, que se refere a normalizar todos valores do dataset tal que a média seja igual a zero e o desvio-padrão igual a um. Seja x uma variável que desejamos analisar, $\bar{x}$ a média de x e σ_x o desvio-padrão de x, a fórmula para encontrar o novo valor padronizado pelo Z-score $\hat{x}$, é dada a partir da Equação 56.

$$\hat{x} := \frac{x - \bar{x}}{\sigma_x} \tag{56}$$

Por fim, vale ressaltar que como o nosso objetivo é de prever a nota final *'G3'* dos alunos baseado em suas informações pessoais, não queremos que outras notas influenciem o resultado. Sendo assim, nós criamos uma coluna alvo que representa a nota final *'G3'* e após isso, removemos as colunas *'G1', 'G2','G3'* da nossa matriz de dados.

3.2.2 Métricas de avaliação

Neste tópico, apresentaremos algumas das métricas de avaliação mais comuns no ramo de Inteligência Computacional. Com o objetivo de facilitar a compreensão das métricas para o leitor, vamos primeiro apresentar o caso binário, para depois mostrarmos o caso geral, multiclasse.

3.2.2.1 Caso binário

Suponha que um conjunto de dados $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ possua duas classes, com y podendo ser positivo ou negativo. Além disso, suponha que um modelo tenha sido treinado e sua solução encontrada seja dada por $\hat{y} \in \{-1, 1\}^n$. Portanto

1. Caso $y < 0$
   (a) Caso $\hat{y} = y$: Obtivemos um verdadeiro negativo (V_N)
   (b) Caso $\hat{y} \neq y$: Obtivemos um falso negativo (F_N)
2. Caso $y > 0$
   (a) Caso $\hat{y} = y$: Obtivemos um verdadeiro positivo (V_P)
   (b) Caso $\hat{y} \neq y$: Obtivemos um falso positivo (F_P)

A partir disso, é possível obter uma matriz que apresenta os erros e os acertos do modelo, comparando com o resultado esperado. Esta matriz é chamada de matriz de confusão e é apresentada na Tabela 2.

Após realizada a contagem de todos termos e a obtenção da matriz de confusão, é possível calcular métricas de avaliação do modelo [25].

		Detectada	
		$\hat{y} = -1$	$\hat{y} = +1$
Real	$\hat{y} = -1$	V_N	F_P
	$\hat{y} = +1$	F_N	V_P

Tabela 2: Matriz de confusão para classificação binária

1. Acurácia: Indica o percentual de classes classificadas corretamente pelo modelo. Ela é uma boa indicação geral de como o modelo performou, porém não deve ser utilizada com dados desbalanceados.
$$A := \frac{V_P + V_N}{V_P + V_N + F_P + F_N} \ . \tag{57}$$

2. Precisão: Apresenta o percentual de classes positivas classificadas corretamente. A precisão é comumente utilizada quando falsos positivos são muito mais prejudiciais que os falsos negativos.
$$P := \frac{V_P}{V_P + F_P} \ . \tag{58}$$
Caso o denominador seja igual a zero, definimos que $P := 0$.

3. Revocação: Calcula o percentual de positivos encontrados dentre todos os positivos. Normalmente utilizada em situações que falsos negativos são considerados muito mais prejudiciais do que falsos positivos.
$$R := \frac{V_P}{V_P + F_N} \ . \tag{59}$$
Caso o denominador seja igual a zero, definimos que $R := 0$.

4. F1-Score: Apresenta a média harmônica entre a precisão e o revocação. O F1-Score é comumente utilizado quando se deseja possuir tanto uma boa precisão quanto uma boa revocação. Esta técnica de avaliação também é uma ótima forma de analisar uma métrica ao invés de duas, caso esta métrica esteja com valor baixo, então isso é um indicativo de que ou a precisão ou a revocação está baixa.
$$F_1 := 2 \cdot \frac{P \cdot R}{P + R} \ . \tag{60}$$
Caso o denominador seja igual a zero, definimos que $F_1 := 0$.

3.2.2.2 Caso multiclasse

Suponha que um conjunto de dados $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ possua k classes, tal que

$y \in \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}^n$. Além disso, suponha que um modelo tenha sido treinado e sua solução encontrada seja dada por $\hat{y} \in \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}^n$. Assim como no caso binário, apresentado na Seção 3.2.2.1, segundo [24], também é possível realizar uma comparação entre a classe preditiva e a real para o caso multiclasse. Seja a Tabela 3, a matriz de confusão para k classes. Para

		Detectada ($\hat{y}$)				
		$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\ldots$	$\mathcal{C}_k$
Real (y)	$\mathcal{C}_1$	a_{11}	a_{12}	a_{13}	$\ldots$	a_{1k}
	$\mathcal{C}_2$	a_{21}	a_{22}	a_{23}	$\ldots$	a_{2k}
	$\mathcal{C}_3$	a_{31}	a_{32}	a_{33}	$\ldots$	a_{3k}
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
	$\mathcal{C}_k$	a_{k1}	a_{k2}	a_{k3}	$\ldots$	a_{kk}

Tabela 3: Matriz de confusão para k classes

cada classe $\mathcal{C}_i$, teremos suas respectivas $V_P^{(i)}$, $F_N^{(i)}$, $V_N^{(i)}$, $F_P^{(i)}$, que são encontradas a partir da Equação 61.

$$V_P^{(i)} := a_{ii} \ , \qquad F_P^{(i)} := \sum_{\substack{j=1 \\ j \neq i}}^{k} a_{ji} \ , \qquad F_N^{(i)} := \sum_{\substack{j=1 \\ j \neq i}}^{k} a_{ij} \ , \qquad V_N^{(i)} := \sum_{\substack{j=1 \\ j \neq i}}^{k} \sum_{\substack{l=1 \\ l \neq i}}^{k} a_{jl} \tag{61}$$

A partir disso, para cada classe $\mathcal{C}_i$, é possível calcular a precisão $P^{(i)}$ pela Equação 58, a revocação $R^{(i)}$ pela Equação 59 e o F1-score $F_1^{(i)}$ pela equação 60. Seja γ_i a porcentagem da classe $\mathcal{C}_i$ em comparação ao total de classes, então:

1. Acurária Multiclasse

$$A := \frac{\sum_i V_P^{(i)} + \sum_i V_N^{(i)}}{\sum_i V_P^{(i)} + \sum_i V_N^{(i)} + \sum_i F_P^{(i)} + \sum_i F_N^{(i)}} \tag{62}$$

2. Precisão Multiclasse

$$P := \sum_{i=1}^{k} \gamma_i P^{(i)} \ . \tag{63}$$

3. Revocação Multiclasse

$$R := \sum_{i=1}^{k} \gamma_i R^{(i)} \ . \tag{64}$$

4. F1-score Multiclasse

$$F_1 := \sum_{i=1}^{k} \gamma_i F_1^{(i)} \ . \tag{65}$$

3.2.3 Parâmetros dos modelos

Após analisados os modelos utilizados e as métricas de avaliação, decidimos determinar as funções e os parâmetros dos modelos. Variamos os parâmetros de todos modelos, com exceção da Classificação Bayesiana, e apresentaremos todos os resultados obtidos na Seção 4.

3.2.3.1 Regressão Logística

Para realizar a Regressão Logística que foi apresentada na Seção 3.1.1, nós utilizamos a função LogisticRegression do sklearn. Como parâmetros, selecionamos a tolerância do critério de parada tol $= 10^{-4}$, o número máximo de iterações para os solucionadores convergirem max_iter $= 100$.

Além disso, como forma de regularização, para obter uma menor variância do modelo, é adicionado à função objetivo um fator dado por $\frac{1}{C} \cdot R(W)$, sendo assim, nós variamos dois tipos de parâmetros diferentes, variando assim a força da regularização de cada modelo:

1. Penalty: Indica a norma da penalidade (veja [17])

   (a) L1: Nesta regularização $R(W) = \|W\|_1$, ou seja, o somatório da magnitude dos coeficientes, também conhecida como distância de Manhattan.

   (b) L2: Nesta regularização $R(W) = \frac{1}{2}\|W\|_2^2$, ou seja, é o somatório do quadrado dos coeficientes multiplicado por meio.

   (c) elasticnet: Esta é uma regularização mista, onde são adicionadas penalidades tanto em L1, quanto em L2. Portanto para um $\alpha \in [0, 1]$, a regularização elasticnet é dada por

   $$R(W) = \alpha\|W\|_1 + \frac{(1-\alpha)}{2}\|W\|_2^2 \,. \tag{66}$$

   Veja que se $\alpha = 0$, então a penalização seria igual a L2, e se $\alpha = 1$ então a penalização seria igual a L1. No sklearn, o termo α é dado por l1_ratio, sendo $0 \leq$ l1_ratio ≤ 1. A partir disso, definimos l1_ratio $= 0.5$ para que cada penalização possuísse o mesmo peso.

2. C: Indica o inverso da força de regularização. Valores maiores indicam uma regularização mais fraca. Testamos 5 amostras igualmente espaçadas entre $[1.00, 2.00]$.

3.2.3.2 Classificação Bayesiana

Para a Classificação Bayesiana, apresentada na Seção 3.1.2, utilizamos a função GaussianNB do sklearn, que utiliza o modelo Gaussian Naive Bayes, tendo em vista que os dados apresentam uma distribuição normal. Selecionamos o parâmetro que var_smoothing igual a 10^{-9}.

3.2.3.3 Árvore de decisão

Para a árvore de decisão, apresentada na Seção 3.1.3, utilizamos a função DecisionTreeClassifier do sklearn. Como parâmetros, escolhemos o splitter best, sendo sua estratégia escolher a melhor divisão em cada nó da árvore, o menor número de amostras necessário para dividir o nó interior sendo min_split_samples = 2 e variamos dois tipos de parâmetros diferentes:

1. Criterion: Sendo a função que calcula a qualidade da divisão, vista na Seção 3.1.3.2.
   (a) Gini: Utiliza a impureza de Gini.
   (b) Entropy: Utiliza a entropia.
2. Max_depth: A máxima profundidade da árvore. Testamos cinco amostras igualmente espaçadas entre $[3, 15]$ e uma amostra igual a *None*, que é o caso onde a árvore não tem limite de profundidade, ou seja, os nós são expandidos até todas folhas serem podadas ou conterem um número menor que min_samples_split.

3.2.3.4 Random Forest

Para a realização do Random Forest que foi apresentada na Seção 3.1.4, utilizamos a função RandomForestClassifier do sklearn. Seja n_A o número de atributos do modelo, então como parâmetros, escolhemos o número de árvores na floresta sendo n_estimators = 100, o número máximo de atributos para cada árvore sendo igual a $\sqrt{n_A}$ (a partir de max_features = auto), o menor número de amostras necessário para dividir o nó interior sendo min_split_samples = 2 e variamos dois tipos de parâmetros diferentes:

1. Criterion: Sendo a função que calcula a qualidade da divisão, vista na Seção 3.1.3.2.
   (a) Gini: Utiliza a impureza de Gini.
   (b) Entropy: Utiliza a entropia.
2. Max_depth: A máxima profundidade da árvore. Testamos cinco amostras igualmente espaçadas entre $[3, 15]$ e uma amostra igual a *None*, que é o caso onde a árvore não tem limite de profundidade, ou seja, os nós são expandidos até todas folhas serem podadas ou conterem um número menor que min_samples_split.

3.2.3.5 Gradient Boosting

Para o Gradient Boosting, apresentado na Seção 3.1.5, utilizamos a função GradientBoosting-Classifier do sklearn. Como parâmetros, escolhemos a função de perda a ser otimizada sendo

loss igual a deviance, que se refere à regressão logística para classificação com resultados probabilísticos, para medir a qualidade do split foi utilizado o erro quadrático médio, e variamos dois tipos de parâmetros diferentes que apresentam um trade-off entre eles:

1. Learning_rate: Fator que reduz a contribuição de cada árvore. Testamos quatro amostras igualmente espaçadas entre $[0.05, 0.20]$.

2. N_estimators: Indica o número de estágios de boosting a serem performados. Testamos quatro amostras igualmente espaçadaa entre $[50, 200]$.

3.2.3.6 SVM

Para o SVM, apresentado na Seção 3.1.6, utilizamos a função svc do sklearn. Como parâmetros, escolhemos a função a tolerância do critério de parada $tol = 10^{-3}$, a função da decisão da forma sendo decision_function_shape $= ovr$, que treina um classificador para cada classe ajustado contra todas as classes. Além disso, variamos dois tipos de parâmetros:

1. Kernel: Sendo a função de núcleo do modelo, tendo as seguintes opções: linear, poly, rbf e sigmoid.

   (a) linear: Sendo uma função linear, apresentada na Equação 46.

   (b) poly: Sendo uma função polinomial de grau r e constante c, vista na Equação 47. Definimos $r := 3$ e constante $c := 0$.

   (c) rbf: Uma função radial, comumente utilizada quando não se tem um conhecimento prévio dos dados, com o parâmetro $\gamma > 0$, apresentada na Equação 48. No nosso caso, definimos o valor de γ sendo o inverso do número de atributos dos dados, que é o default do sklearn (gamma := scaled).

   (d) sigmoid: Sendo uma função com parâmetro $\gamma > 0$ e constante c, que pode ser encontrada na Equação 49. Definimos o valor de γ sendo o inverso do número de atributos dos dados, que é o default do sklearn (gamma := scaled) e $c := 0$.

2. C: Indica o inverso da força de regularização. Valores maiores indicam uma regularização mais fraca. Testamos 5 amostras igualmente espaçadas entre $[1.00, 2.00]$.

3.2.3.7 Redes Neurais

Para as redes neurais, apresentado na Seção 3.1.7, escolhemos a função MLPClassifier do sklearn. Como parâmetros, escolhemos o solver *Adam* que é um otimizador default baseado em gradiente estocástico. Além disso, também variamos dois tipos de parâmetros:

1. Activation: Sendo a função de ativação $f(x)$ da camada escondida

   (a) Logistic: Sendo a função sigmoide logística, dada por

   $$f(x) = \frac{1}{1 + \exp(-x)} \ . \tag{67}$$

   (b) Tanh: Sendo a função hiperbólica tan, dada por

   $$f(x) = \tanh(x) \ . \tag{68}$$

   (c) Relu: Sendo a função de unidade linear retificada, dada por:

   $$f(x) = \max\{0, x\} \ . \tag{69}$$

2. Hidden_layers_sizes: Indicando o número de camadas escondidas, sendo o tamanho da tupla, e o cada valor x_i contido no índice i da tupla dado pelo número de neurônios da camada escondida i. Decidimos escolher utilizar duas camadas escondidas, com o número de neurônios podendo ser dado por $\{2, 5, 10, 20\}$.

4 Resultados

4.1 Visualização e Caracterização dos dados

Inicialmente, foram geradas algumas visualizações para compreendermos melhor os dados do problema. Para isso, utilizamos de gráficos, tabelas e imagens com o intuito de verificar as tendências, padrões e valores discrepantes nos dados.

4.1.1 Informações dos Atributos

O primeiro passo realizado foi remover as variáveis *G1* e *G2*, tendo em vista que não desejamos obter informações de notas anteriores. Em seguida, utilizamos a função *info* e a função *describe* da biblioteca *pandas*.

A função *info* indica a quantidade de dados não nulos para cada um dos atributos e o tipo desses dados, podendo ser objeto ou inteiro. A partir dela, vimos que o nosso dataset contém 31 atributos, sendo 17 do tipo objeto e 14 do tipo inteiro de 64 bit. Além disso, todos os atributos possuem 649 dados, ou seja, o nosso dataset não apresenta valores nulos.

A função *describe* apresenta um resumo estatístico dos atributos não-categóricos do dataset. Com ela podemos observar diversas informações, como quantidade de valores, média,

	age	Medu	Fedu	travel	study	fail	famrel	freetime	goout	Dalc	Walc	health	absences	G3
count	649	649	649	649	649	649	649	649	649	649	649	649	649	649
mean	16.744	2.515	2.307	1.569	1.931	0.222	3.931	3.180	3.185	1.502	2.280	3.536	3.659	11.906
std	1.218	1.135	1.100	0.749	0.830	0.593	0.956	1.051	1.176	0.925	1.284	1.446	4.641	3.231
min	15	0	0	1	1	0	1	1	1	1	1	1	0	0
25%	16	2	1	1	1	0	4	3	2	1	1	2	0	10
50%	17	2	2	1	2	0	4	3	3	1	2	4	2	12
75%	18	4	3	2	2	0	5	4	4	2	3	5	6	14
max	22	4	4	4	4	3	5	5	5	5	5	5	32	19

Tabela 4: Tabela retornada pela função describe

desvio-padrão, mínimos, máximos e também nos auxilia também a identificar distribuições assimétricas quando temos média e mediana (percentil 50) muito distantes uma da outra. Devido à sua importância, apresentamos o output retornado pela função na Tabela 4.

Após isso, decidimos verificar a distribuição das variavéis do dataset, que é apresentada na Figura 8. A partir dela, é possível identificar mais a fundo as informações pessoais dos estudantes presentes no dataset. Nos atributos numéricos, vemos que alguns deles apresentam uma distribuição próxima à distribuição normal, como por exemplo: *age*, *freetime*, *goout* e *G3*. Além disso, também percebemos que a maior parte dos alunos moram com mais de três pessoas em sua casa, com seus pais em união estável, não recebem suporte financeiro escolar, não pagam aulas particulares e desejam realizar ensino superior.

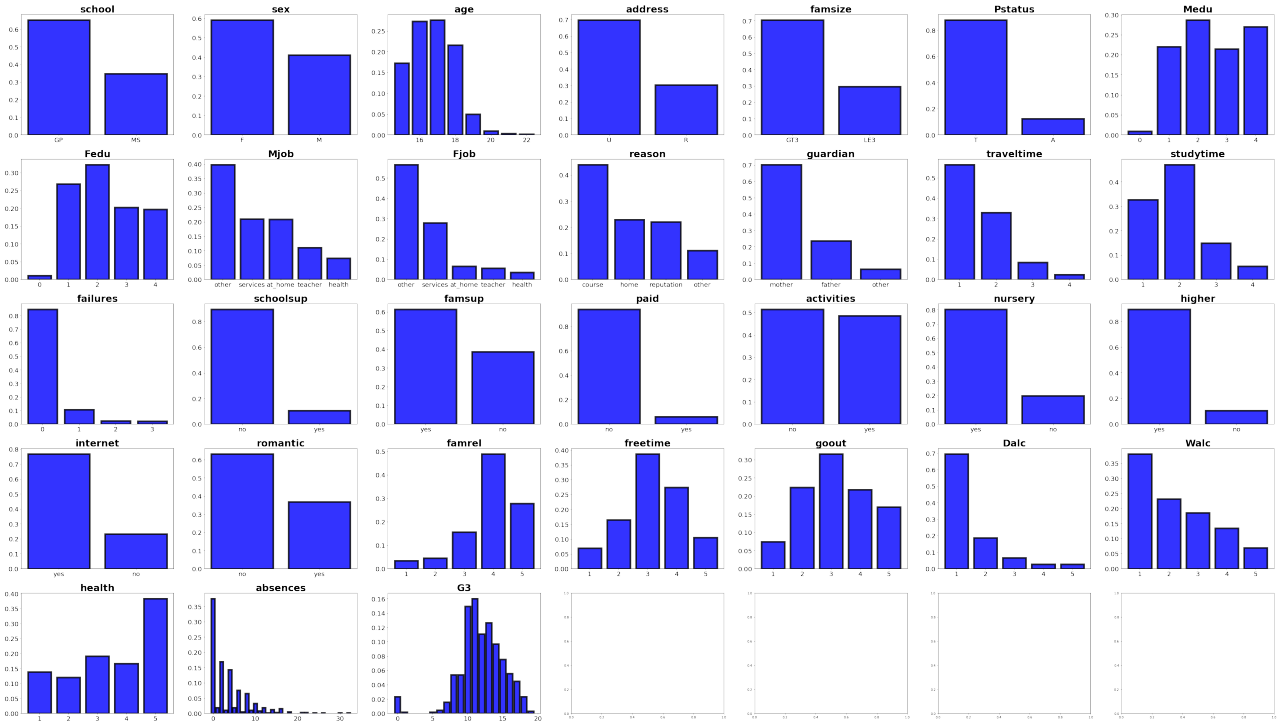

Figura 8: Histograma dos atributos

4.1.2 Binarização e Codificação One-Hot

Após a pré-visualização dos dados no histograma, se tornou mais fácil de identificar os próximos passos do projeto. Primeiramente, vimos que era preciso tratar as variáveis nominais de forma que elas se tornassem registros numéricos, através da binarização ou da codificação one-hot. O procedimento para cada uma das variáveis é apresentado abaixo:

- **Binarização:** 'school', 'sex', 'address', 'famsize', 'pstatus', 'schoolsup', 'famsup', 'paid', 'activities', 'nursery', 'higher', 'internet', 'romantic'.
- **Codificação one-hot:** 'Mjob', 'Fjob', 'reason', 'guardian'

Para a binarização, fizemos uma função no *Python* que recebe o dataframe *pandas*, a variável a ser binarizada e seus rótulos. Depois disso passamos por todas as linhas da coluna e com auxilio do método *.loc* do *pandas* substituimos os valores dos rótulos por 0 ou 1. Para realizar a codificação one-hot, utilizamos o método *.get_dummies* do *pandas* que recebe o dataset e a coluna a ser transformada como argumentos e devolver o dataframe já com as variáveis *dummy*. A Tabela 5 apresenta um exemplo das 4 primeiras linhas do dataset após as transformações de binarização e codificação one-hot.

index	schoolsup	guardian_father	guardian_mother	guardian_other
0	1	0	1	0
1	0	1	0	0
2	1	0	1	0
3	0	0	1	0

Tabela 5: Exemplo de transformação feita no dataset utilizado

4.1.3 Tratamento dos dados

Durante o tratamento de dados é importante analisar a distribuição dos dados, pois distribuições muito assimétricas, com média e desvio padrão muito distantes, podem enviesar o modelo.

Na Figura 9 mostramos os gráficos das frequências relativas das variáveis numéricas para conseguirmos visualizar o formato da distribuição respectiva. A partir dela, notamos que algumas variáveis possuíam distribuições assimétricas, portanto optamos por realizar a transformação logarítmica das variáveis numéricas com o intuito de deixar as distribuições mais simétricas. A Figura 10 ilustra como ficaram os histogramas após a transformação logarítmica.

Como nosso conjunto de dados possui dados com escalas diferentes, padronizamos as variáveis a partir do Z-Score, apresentado na Seção 3.2.1, tendo em vista que deixar as variáveis com escalas diferentes poderia interferir nos modelos.

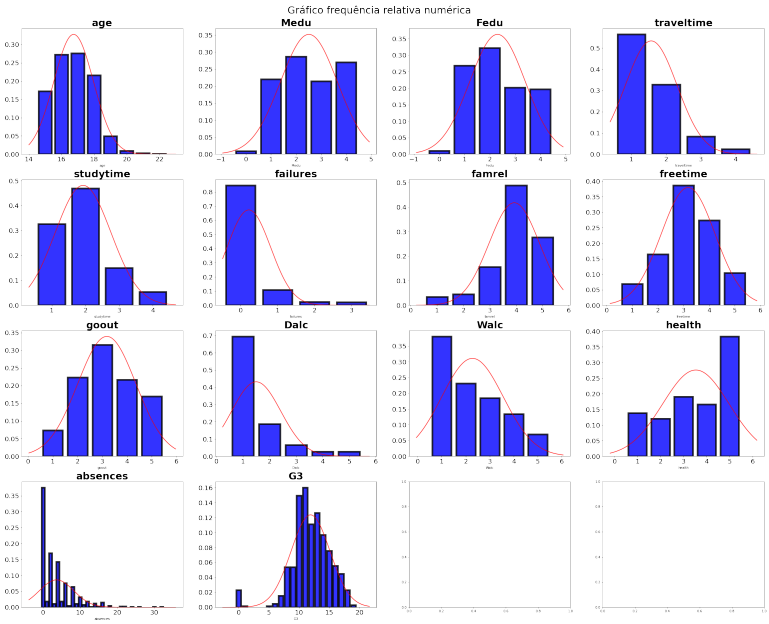

Figura 9: Frequência relativa das variáveis numéricas

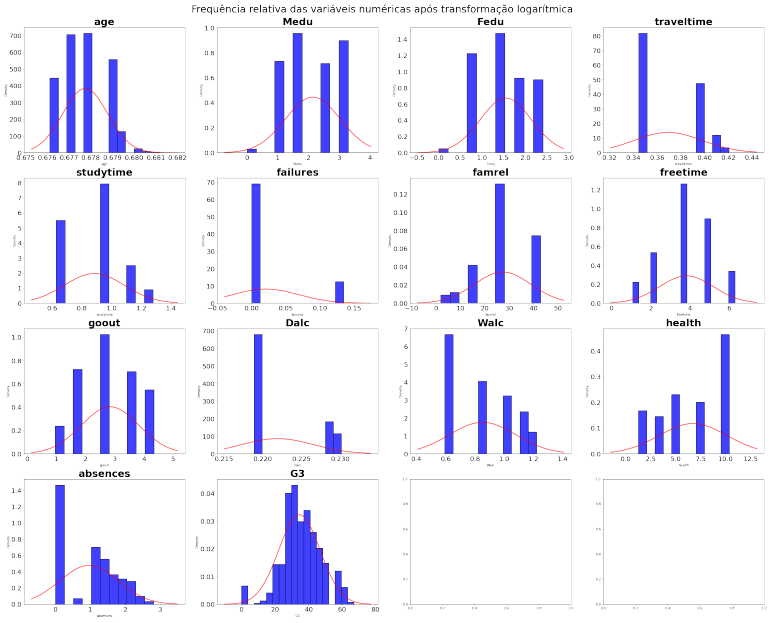

Figura 10: Frequência relativa das variáveis numéricas após transformação logarítmica

Após isso, com o objetivo de encontrar outliers analisamos a matriz de distâncias, que é apresentada na Figura 11 e retiramos as 10 amostras mais distantes. A matriz de distâncias após o tratamento é apresentada na Figura 12.

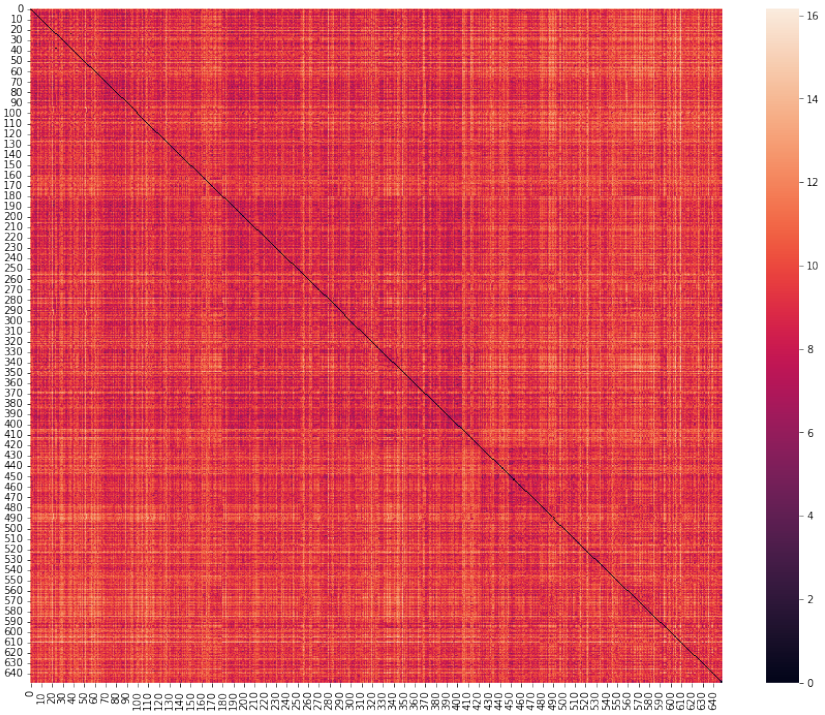

Figura 11: Matriz de distância original

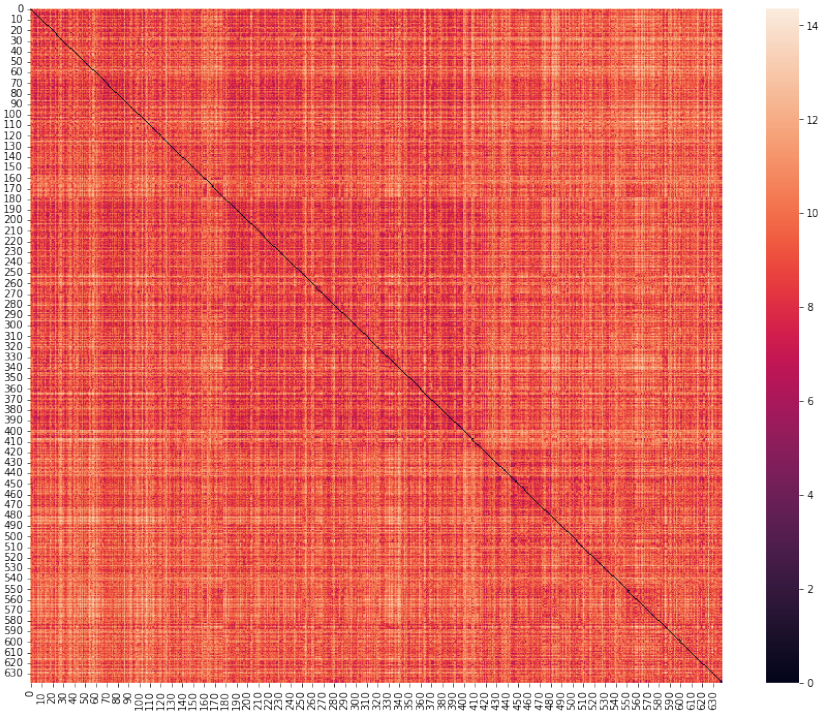

Figura 12: Matriz de distância após tratamento

4.1.4 Correlação entre os atributos

A correlação linear apresentada na Figura 13, mostra o quão forte é a relação entre um atributo à outro. Quando a correlação é mais próxima de 1, indica que os atributos são positivamente correlacionados e quando a correlação é mais próxima de -1, indica que os atributos são negativamente correlacionados. Caso a correlação entre eles seja 0, então os atributos não apresentam correlação entre si. A Figura 13 apresenta a correlação linear dos atributos, nela podemos perceber que a maior parte dos atributos apresentam uma baixa correlação, sendo em sua maioria entre -0.25 e $+0.25$.

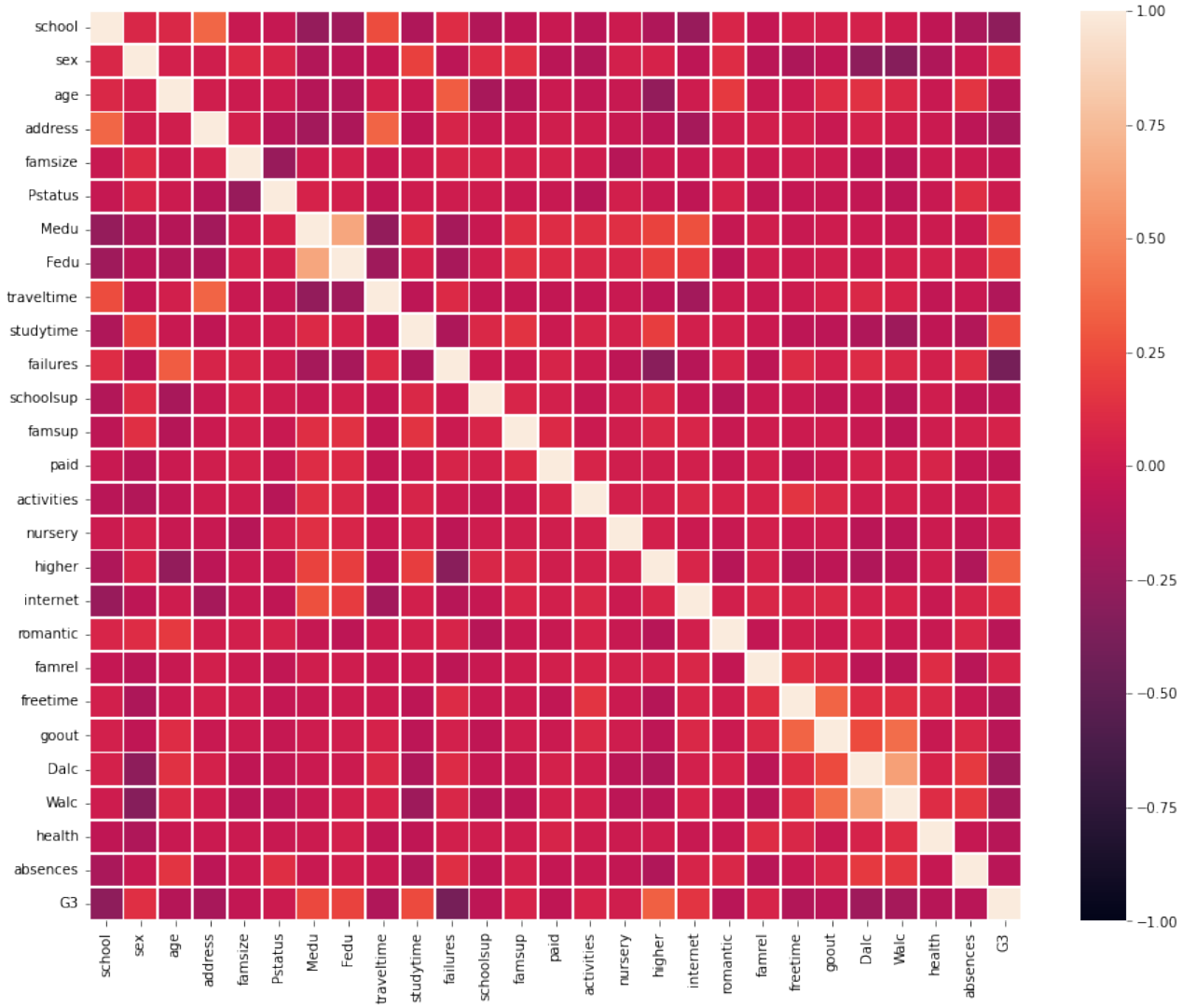

Figura 13: Matriz de Correlação

4.1.5 Distribuição de Classes

Os dados das notas finais da disciplina de Português pertenciam a um conjunto de $\{1, \ldots, 20\}$. Com o objetivo de agrupar essas notas em grupos menores, baseado em [8], vamos utilizar o sistema de classificação Erasmus, que classifica essas notas em cinco grupos, de *Reprovado* a *Excelente*. Sua classificação é mostrada na Tabela 6.

Grupo	Número da Classe	Descrição
Reprovado	0	$0 \leq G_3 \leq 9$
Suficiente	1	$10 \leq G_3 \leq 12$
Satisfatório	2	$12 \leq G_3 \leq 13$
Bom	3	$14 \leq G_3 \leq 15$
Excelente	4	$16 \leq G_3 \leq 20$

Tabela 6: Descrição das Classes

Após esta conversão analisamos a distribuição de classes para verificar se temos dados balanceados ou desbalanceados, veja a Figura 14. Nela, percebemos que o problema não é balanceado já que enquanto o grupo *Suficiente* possui 31.0% dos alunos, o grupo *Excelente* possui somente 12.6%, sendo apresentada uma grande variação entre essas duas classes.

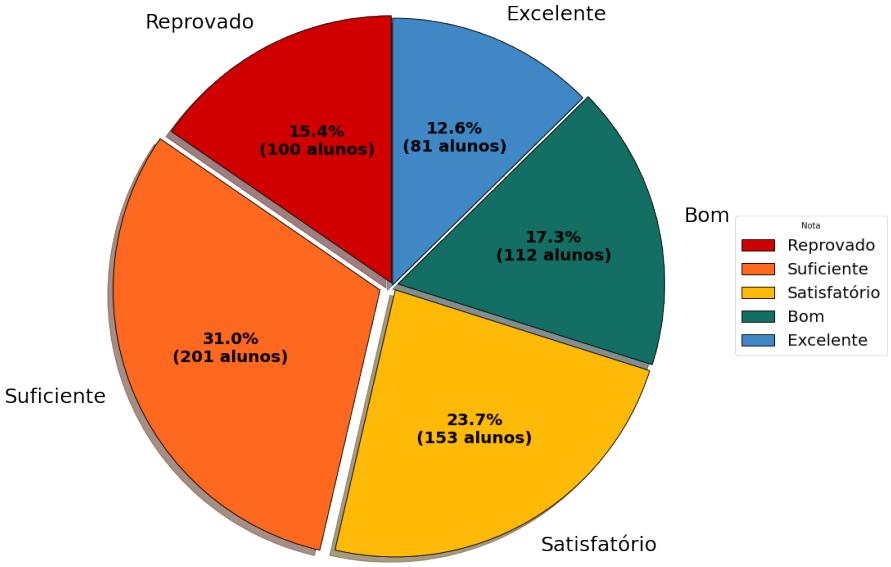

Figura 14: Distribuição de Classes

4.2 Descrição do precedimento de validação cruzada

Segundo [28], a validação cruzada é uma técnica comumente utilizada com o intuito de avaliar a capacidade de generalização do modelo. Seja $\mathcal{D}$ um conjunto de dados, tal que $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, a validação cruzada particiona $\mathcal{D}$ em dois subconjuntos mutualmente exclusivos e utiliza um deles como conjunto de treinamento e o outro como conjunto de teste. Existem diversas técnicas de validação cruzada, porém escolhemos utilizar a K-Fold.

O método K-Fold, divide $\mathcal{D}$ em k subconjuntos mutualmente exclusivos, que denotaremos por $\mathcal{C}_k$. A cada iteração i, utilizaremos $\mathcal{C}_i$ como conjunto de treinamento e $\mathcal{D} \setminus \mathcal{C}_i$ como conjunto de treinamento. Ao final de k iterações, é calculada a média e o desvio-padrão dos erros de cada iteração. A partir de [1] é apresentada a Figura 15 que mostra um exemplo de uma divisão de um conjunto $\mathcal{D}$ em 10 subconjuntos, sendo na iteração i, $\mathcal{C}_i$ o subconjunto de cor azul mais escura e $\mathcal{D} \setminus \mathcal{C}_i$ de cor mais clara.

Neste trabalho decidimos utilizar a validação cruzada em 10 subconjuntos. A partir da Figura 14, vimos que os nossos dados são desbalanceados, sendo assim utilizamos a função *StratifiedKFold* do *sklearn* com o intuito de manter a mesma porcentagem de classes em cada amostra em comparação a porcentagem original.

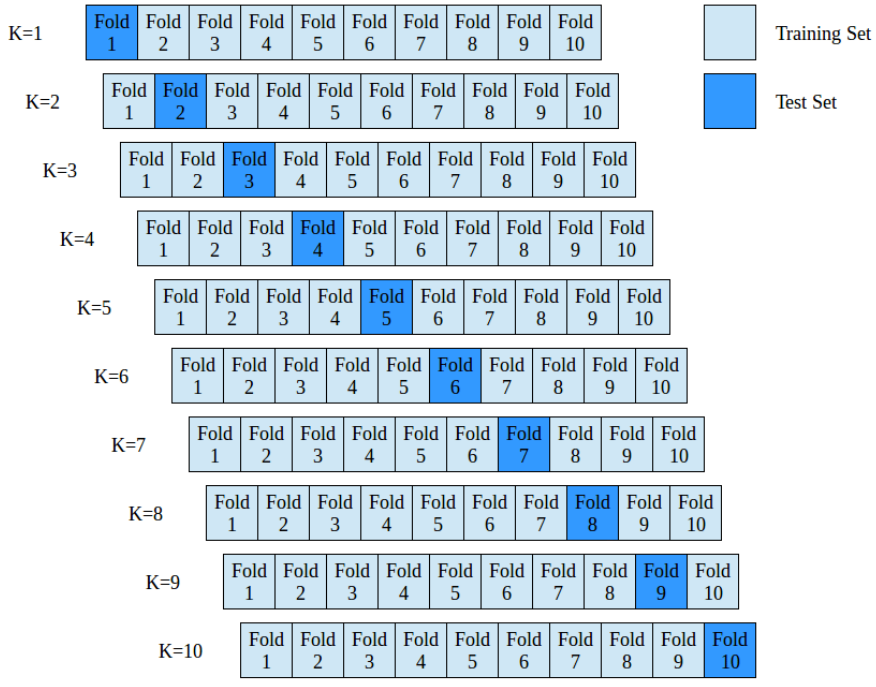

Figura 15: Validação Cruzada: K-Fold ($k = 10$)

4.3 Resultados dos Modelos

Nós apresentaremos os resultados encontrados por cada um dos modelos, testando os diferentes parâmetros descritos na Seção 3.2.3. Com base nos métodos de avaliação, explicados na Seção 3.2.2, analisaremos os resultados do F1-score, tendo em vista que desejamos obter boas precisões e revocações.

Para cada método apresentado na Seção 3.1, teremos três tabelas: F1-score, sendo a métrica mais importante da nossa análise, Precisão e Revocação. Não trabalharemos com a acurácia, já que os nossos dados são desbalanceados, não sendo aconselhável utilizá-la.

Em cada tabela, serão apresentados a média e o desvio-padrão da determinada métrica em porcentagem, para 10 ciclos de validação cruzada, da seguinte forma: (Média ± Desvio-Padrão). O F1-score advindo da combinação de parâmetros que apresentar a maior média, será apresentado em negrito e seus respectivos parâmetros serão determinados sendo os melhores parâmetros a serem utilizados. Nas tabelas de Precisão e Revocação, para efeitos de comparação, apresentaremos em negrito o que resultado relacionado com o melhor F1-score e em sublinhado o melhor resultado obtido pela métrica.

4.3.1 Regressão Logística

Na Regressão Logística, testamos diferentes parâmetros e os resultados obtidos pela métrica F1-Score podem ser vistos na Tabela 7. Os melhores parâmetros foram C = 2.00 juntamente com a penalização L2.

	penalty		
C	L1	L2	elasticnet
1.00	31.87 ± 6.58	32.08 ± 6.32	31.51 ± 6.63
1.25	31.52 ± 6.16	32.11 ± 6.36	31.59 ± 6.58
1.50	31.39 ± 6.27	32.11 ± 6.36	31.96 ± 6.79
1.75	31.30 ± 6.28	32.11 ± 6.36	31.89 ± 6.37
2.00	31.30 ± 6.28	$\mathbf{32.14 \pm 6.37}$	31.78 ± 6.50

Tabela 7: F1-score (%) para Regressão Logística

Para as métricas de precisão (Tabela 8) e revocação (Tabela 9), ambas tiveram os melhores parâmetros sendo C = 1.00 e a penalização L1. A primeira vista, pode parecer estranho a precisão e a revocação serem as mais altas, e mesmo assim, nesta combinação de parâmetros, F_1 não ser o melhor. Porém, isso representa que em média tanto a precisão e a revocação foram boas, contudo apresentaram grandes variações, isso pode ser visto pelo desvio-padrão

	penalty		
C	L1	L2	elasticnet
1.00	$\underline{35.28 \pm 8.10}$	34.93 ± 7.86	34.69 ± 8.46
1.25	34.89 ± 7.79	34.96 ± 7.92	34.13 ± 8.35
1.50	34.52 ± 8.29	34.96 ± 7.92	34.50 ± 8.47
1.75	34.43 ± 8.21	34.96 ± 7.92	34.40 ± 8.06
2.00	34.43 ± 8.21	$\mathbf{35.03 \pm 7.96}$	34.39 ± 8.18

Tabela 8: Precisão (%) para Regressão Logística

	penalty		
C	L1	L2	elasticnet
1.00	$\underline{33.79 \pm 5.97}$	33.64 ± 5.90	33.32 ± 6.05
1.25	33.48 ± 5.64	33.64 ± 5.90	33.32 ± 6.05
1.50	33.32 ± 5.76	33.64 ± 5.90	33.64 ± 6.17
1.75	33.17 ± 5.75	33.64 ± 5.90	33.64 ± 5.77
2.00	33.17 ± 5.75	$\mathbf{33.64 \pm 5.90}$	33.48 ± 5.94

Tabela 9: Revocação (%) para Regressão Logística

também, que é bem elevado. Ou seja, quando a precisão estava elevada, a revocação estava baixa e vice-versa, e como o F1-score apresenta a média harmônica entre elas, se uma delas está mais baixa, o F1-score tende a ser menor.

4.3.2 Classificação Bayesiana

Para a Classificação Bayesiana, não variamos os parâmetros. Neste método encontramos os resultados abaixo:

1. F1-score: 17.14 ± 5.42 (%)
2. Precisão: 26.90 ± 12.55 (%)
3. Revocação: 23.00 ± 4.30 (%)

Vemos que este método apresenta um F1-score muito baixo, além disso, obteve uma precisão um pouco mais elevada em média, porém com um desvio-padrão extremamente elevado. Portanto, não o consideremos como sendo um bom método a ser utilizado para este problema.

4.3.3 Árvore de decisão

Na árvore de decisão, obtemos os resultados do F1-score apresentados na Tabela 10 para os diferentes parâmetros testados, vimos que o melhor resultado foi dado utilizando o parâmetro critério sendo a Gini juntamente sem limite de profundidade na árvore. A precisão (Tabela 11) também foi a melhor neste critério, porém a Revocação (Tabela 12) foi aproximadamente 5% inferior a melhor revocação obtida.

	Criterion	
Max_depth	Gini	Entropy
3	22.45 ± 6.85	23.87 ± 6.97
6	21.56 ± 3.61	23.58 ± 5.98
9	25.09 ± 5.82	25.56 ± 4.28
12	25.57 ± 5.75	24.86 ± 6.03
15	24.52 ± 6.57	24.78 ± 7.24
Ilimitada	$\mathbf{28.11 \pm 6.33}$	26.37 ± 6.11

Tabela 10: F1-score (%) para Árvore de decisão

	Criterion	
Max_depth	Gini	Entropy
3	21.39 ± 6.35	24.86 ± 7.49
6	23.97 ± 6.01	26.19 ± 7.48
9	25.85 ± 6.48	28.49 ± 10.10
12	25.87 ± 6.21	26.17 ± 6.06
15	24.95 ± 6.59	24.96 ± 6.91
Ilimitada	$\underline{\mathbf{29.92 \pm 6.55}}$	26.96 ± 5.50

Tabela 11: Precisão (%) para Árvore de decisão

4.3.4 Random Forest

No modelo Random Forest, testamos diferentes parâmetros, que são apresentados na Tabela 13. Percebemos que o melhor resultado do F1-score foi dado utilizando o critério Entropy juntamente sem limite de profundidade. A precisão (Tabela 14) e a revocação (Tabela 15) com estes parâmetros foram próximas em comparação com o melhor resultado, tendo uma diferença de cerca de 1%.

	Criterion	
Max_depth	Gini	Entropy
3	29.73 ± 6.73	$\underline{33.33 \pm 6.27}$
6	25.35 ± 3.62	28.33 ± 5.25
9	27.23 ± 5.23	27.71 ± 4.35
12	26.59 ± 5.51	26.16 ± 7.28
15	25.35 ± 6.18	26.16 ± 8.40
Ilimitada	$\mathbf{28.64 \pm 5.98}$	27.41 ± 7.14

Tabela 12: Revocação (%) para Árvore de decisão

	Criterion	
Max_depth	Gini	Entropy
3	23.14 ± 5.32	22.84 ± 3.84
6	27.53 ± 5.61	28.68 ± 4.22
9	27.16 ± 5.22	28.48 ± 5.80
12	27.95 ± 4.46	29.04 ± 4.03
15	28.43 ± 5.54	29.35 ± 5.67
Ilimitada	28.65 ± 2.95	$\mathbf{30.20 \pm 5.09}$

Tabela 13: F1-score (%) para Random Forest

	Criterion	
Max_depth	Gini	Entropy
3	21.32 ± 8.47	22.96 ± 8.80
6	27.66 ± 6.40	$\underline{34.64 \pm 4.95}$
9	29.68 ± 6.36	31.60 ± 8.56
12	30.49 ± 7.28	31.50 ± 5.29
15	32.31 ± 6.68	32.36 ± 10.67
Ilimitada	29.81 ± 4.75	$\mathbf{33.98 \pm 5.60}$

Tabela 14: Precisão (%) para Random Forest

4.3.5 Gradient Boosting

Para o método Gradient Boosting, obtemos os resultados do F1-score que são apresentados na Tabela 16 para os diferentes parâmetros testados. Vimos que o melhor resultado foi dado utilizando o parâmetro learning_rate = 0.10 juntamente com n_estimators = 50. A precisão (Tabela 17) e a revocação (Tabela 18) apresentaram a melhor média com esse parâmetros,

	Criterion	
Max_depth	Gini	Entropy
3	34.42 ± 4.55	33.80 ± 3.50
6	$\underline{34.58 \pm 5.68}$	34.43 ± 3.94
9	30.83 ± 4.61	32.55 ± 5.47
12	31.29 ± 3.90	32.70 ± 4.01
15	31.92 ± 4.86	32.08 ± 5.55
Ilimitada	31.92 ± 3.03	$\mathbf{33.02 \pm 5.31}$

Tabela 15: Revocação (%) para Random Forest

porém, para a revocação, existiria um outro parâmetro a ser utilizado que também apresentaria a melhor média, porém com um desvio-padrão reduzido.

	n_estimators			
learning_rate	50	100	150	200
0.05	28.34 ± 4.56	29.88 ± 4.40	29.25 ± 3.37	28.49 ± 3.41
0.10	$\mathbf{30.11 \pm 4.89}$	29.48 ± 3.52	29.71 ± 4.88	29.56 ± 5.08
0.15	29.62 ± 3.40	29.33 ± 6.21	28.87 ± 6.52	28.90 ± 6.32
0.20	29.15 ± 3.98	28.86 ± 4.24	29.62 ± 4.52	28.58 ± 4.31

Tabela 16: F1-score (%) para Gradient Boosting

	n_estimators			
learning_rate	50	100	150	200
0.05	31.74 ± 7.54	32.01 ± 7.61	31.34 ± 5.60	31.12 ± 3.54
0.10	$\underline{\mathbf{33.13 \pm 7.91}}$	31.77 ± 4.16	31.94 ± 5.65	31.16 ± 6.30
0.15	31.95 ± 3.76	30.74 ± 6.27	30.42 ± 7.44	30.29 ± 6.96
0.20	30.93 ± 5.93	30.53 ± 5.71	31.60 ± 5.35	30.30 ± 5.70

Tabela 17: Precisão (%) para Gradient Boosting

4.3.6 SVM

Para o método SVM, a Tabela 19 apresenta os resultados para os diferentes parâmetros testados, vimos que o melhor resultado foi dado utilizando o parâmetro kernel sendo igual a sigmoid juntamente com C := 2.00. A precisão (Tabela 20) e a revocação (Tabela 21) também apresentaram a melhor média com esses parâmetros.

	n_estimators			
learning_rate	50	100	150	200
0.05	30.21 ± 4.82	31.61 ± 3.71	30.51 ± 2.08	29.26 ± 3.17
0.10	$\mathbf{31.61 \pm 4.53}$	30.67 ± 2.97	30.67 ± 5.25	30.51 ± 4.82
0.15	30.83 ± 2.85	30.36 ± 6.40	30.20 ± 6.10	30.20 ± 6.45
0.20	30.21 ± 4.17	30.04 ± 4.10	30.68 ± 4.72	30.04 ± 4.39

Tabela 18: Revocação (%) para Gradient Boosting

	Kernel			
C	linear	poly	rbf	sigmoid
1.00	29.71 ± 5.41	24.89 ± 3.92	27.79 ± 4.68	28.95 ± 6.94
1.25	29.88 ± 5.30	25.34 ± 3.36	29.00 ± 4.26	29.98 ± 6.01
1.50	30.15 ± 5.39	25.97 ± 4.00	28.16 ± 2.80	30.34 ± 6.17
1.75	30.20 ± 4.84	25.50 ± 3.80	29.22 ± 2.75	29.57 ± 5.54
2.00	30.43 ± 4.85	25.11 ± 3.65	28.95 ± 2.38	$\mathbf{31.75 \pm 6.08}$

Tabela 19: F1-score (%) para SVM

	Kernel			
C	linear	poly	rbf	sigmoid
1.00	32.02 ± 5.26	28.95 ± 6.81	30.90 ± 7.70	31.03 ± 7.76
1.25	32.35 ± 5.52	29.76 ± 5.20	32.10 ± 6.13	34.24 ± 6.33
1.50	32.66 ± 5.68	30.92 ± 5.57	30.49 ± 5.45	32.51 ± 6.97
1.75	32.70 ± 5.15	31.17 ± 7.04	31.32 ± 5.21	33.00 ± 5.96
2.00	33.12 ± 5.09	29.81 ± 6.07	30.49 ± 4.11	$\mathbf{34.25 \pm 6.19}$

Tabela 20: Precisão (%) para SVM

	Kernel			
C	linear	poly	rbf	sigmoid
1.00	31.44 ± 5.46	31.61 ± 4.90	31.93 ± 4.97	32.55 ± 6.42
1.25	31.44 ± 5.28	30.84 ± 4.68	31.78 ± 4.14	32.70 ± 6.29
1.50	31.75 ± 5.35	30.68 ± 5.33	30.37 ± 2.81	32.70 ± 5.83
1.75	31.75 ± 4.97	29.59 ± 5.07	30.99 ± 3.07	31.45 ± 5.80
2.00	31.91 ± 4.88	28.49 ± 4.46	30.52 ± 2.40	$\mathbf{33.96 \pm 6.50}$

Tabela 21: Revocação (%) para SVM

4.3.7 Redes Neurais

Nas redes neurais, testamos diferentes parâmetros e seus resultados são apresentados na Tabela 22. Percebemos que o melhor resultado foi dado utilizando o parâmetro activation sendo igual a tanh juntamente com hidden_layers_sizes := (5, 5). Para a precisão (Tabela 23) a melhor média também é dada com estes parâmetros, e a revocação (Tabela 24) é aproximadamente 2% menor que a melhor revocação, porém apresenta um valor consideravelmente alto relacionando com as revocações com outros parâmetros.

	Activation		
Hidden_layers_sizes	Logistic	Tanh	Relu
(2,2)	16.97 ± 3.03	24.19 ± 6.71	20.16 ± 7.06
(5,5)	24.82 ± 5.44	$\mathbf{33.25 \pm 5.89}$	25.86 ± 5.30
(10,10)	27.71 ± 4.08	30.71 ± 5.23	27.31 ± 4.18
(20,20)	30.88 ± 5.68	25.78 ± 3.66	31.98 ± 5.39

Tabela 22: F1-score (%) para Redes Neurais

	Activation		
Hidden_layers_sizes	Logistic	Tanh	Relu
(2,2)	12.74 ± 4.84	22.79 ± 8.40	18.81 ± 7.94
(5,5)	19.99 ± 4.49	$\underline{\mathbf{35.60 \pm 8.30}}$	27.08 ± 9.95
(10,10)	25.48 ± 6.21	33.79 ± 6.29	28.05 ± 4.60
(20,20)	34.04 ± 7.41	28.57 ± 4.59	33.94 ± 5.23

Tabela 23: Precisão (%) para Redes Neurais

	Activation		
Hidden_layers_sizes	Logistic	Tanh	Relu
(2,2)	31.77 ± 1.98	31.74 ± 7.38	29.25 ± 8.89
(5,5)	35.99 ± 6.68	$\mathbf{35.35 \pm 5.58}$	32.08 ± 4.40
(10,10)	$\underline{37.08 \pm 5.38}$	31.78 ± 5.23	29.26 ± 4.01
(20,20)	35.04 ± 6.21	26.92 ± 3.70	33.16 ± 5.20

Tabela 24: Revocação (%) para Redes Neurais

4.4 Comparação dos resultados

A partir da definição dos resultados obtidos com o melhores hiper-parâmetros, foi feita a Tabela 25 que apresenta o F1-score, Precisão e Revocação para cada modelo com os parâmetros definidos anteriormente. Também é possível visualizar graficamente o resultado com F1-Score na Figura 16.

Modelo	**F1-Score** (%)	**Precisão** (%)	**Revocação** (%)
Regressão Logística	32.14 ± 6.37	35.03 ± 7.96	33.64 ± 5.90
Classificação Bayesiana	17.14 ± 5.42	26.90 ± 12.55	23.00 ± 4.30
Árvore de Decisão	28.11 ± 6.33	29.92 ± 6.55	28.64 ± 5.98
Random Forest	30.20 ± 5.09	33.98 ± 5.60	33.02 ± 5.31
Gradient Boosting	30.11 ± 4.89	33.13 ± 7.91	31.61 ± 4.53
SVM	31.75 ± 6.08	34.25 ± 6.19	33.96 ± 6.50
Redes Neurais	$\mathbf{33.25 \pm 5.89}$	$\mathbf{35.60 \pm 8.30}$	$\mathbf{35.35 \pm 5.58}$

Tabela 25: F1-score (%) de todos modelos

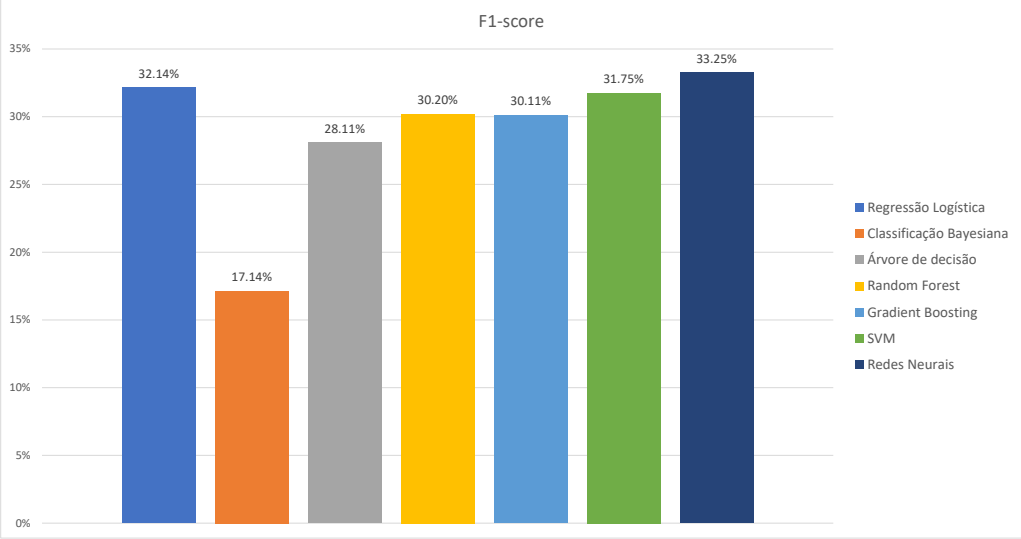

Figura 16: F1-score (%) dos melhores modelos

A partir dos resultados obtidos nos nossos experimentos, vimos que o melhor modelo foi o de Redes Neurais, que obteve a melhor média para o F1-score, a Precisão e a Revocação.

Nos nossos experimentos, encontramos desvios-padrões extremamente elevados, isso é algo muito negativo, tendo em vista que isso indica que os subconjuntos de validação cruzada dos

modelos apresentaram resultados muito variados. Na realidade, se fossem feitos outros experimentos numéricos com a validação cruzada escolhendo aleatoriamente outros subconjuntos, nós poderíamos obter um método distinto com a melhor média e as Redes Neurais podendo ser um dos piores métodos encontrados. O único método que realmente podemos garantir, é o da Classificação Bayesiana, que obteve a pior média do F1-score com uma discrepância muito elevada em comparação com os outros métodos.

5 Conclusão

Realizamos um estudo teórico e computacional dos modelos de Inteligência Computacional estudados em sala de aula. Este trabalho foi muito interessante para nos aprofundarmos nos métodos e vê-los sendo aplicados na prática.

Para essa aplicação, recomendamos a utilização do modelo de Redes Neurais utilização a função de ativação Tanh juntamente com duas camadas escondidas com cinco neurônios cada. Porém, vale ressaltar que, em geral, os resultados obtidos pelos métodos não foram satisfatórios, todos eles apresentaram F1-score abaixo de 50%, indicando erros de predição muito elevados, mesmo após o pré-processamento, remoção de outliers e a verificação de que os dados possuem uma distribuição próxima da normal. Isso mostra que as informações pessoais presentes no formulário não são muito relevantes para indicar se um aluno irá bem na disciplina ou não.

Na realidade, a partir de [8], vimos que o mais importante seria o modelo obter as notas prévias de cada aluno. Isso faz sentido, já que cada pessoa é diferente e possui dificuldades diferentes, o maior motivo de um aluno não ser bom na disciplina é muito mais complexo do que simplesmente saber se seus pais moram juntos ou separados, por exemplo. A alta correlação das notas anteriores apresentada em [8], indica que se um aluno for muito ruim em uma matéria, ele terá que fazer um esforço muito maior para melhorar, e em geral as notas finais se mantém semelhantes às iniciais, caso nada seja feito.

Segundo [6], existem técnicas para reduzir a variação dos resultados dos modelos, que no nosso caso foi muito elevada. Uma delas seria aumentar o conjunto de dados, pois assim seria obtido mais informação no dataset. Esta, poderia ser uma técnica que os pesquisadores poderiam utilizar, já que foram feitas muitas perguntas nos formulários, porém só foram obtidas 649 respostas, de somente duas escolas de Portugal, o que é um número baixo para um problema como esse. Como objetivo de trabalho futuro, desejamos diminuir a variação dos resultados. Como não é possível entrevistar mais alunos para aumentar o dataset, pretendemos utilizar a técnica de empilhar modelos, que a partir da literatura, mostrou ser muito efetiva para diminuir essa variação.

Referências

[1] Jonata Jefferson Andrade, Leonardo Goliatt da Fonseca, Michèle Farage, and Geraldo Luciano de Oliveira Marques. Prediction of the performance of bituminous mixes using adaptive neuro-fuzzy inference systems previsão do desempenho de misturas bituminosas usando sistemas de inferência adaptativa neuro-difusa. 2020.

[2] Baeldung. Multiclass classification using support vector machines, Aug 2021. URL: https://www.baeldung.com/cs/svm-multiclass-classification.

[3] Gulshan Baraik. Major kernel functions in support vector machine (svm), Feb 2022. URL: https://www.geeksforgeeks.org/major-kernel-functions-in-support-vector-machine-svm/.

[4] Thomas Bayes. Lii. an essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, frs communicated by mr. price, in a letter to john canton, amfr s. *Philosophical transactions of the Royal Society of London*, (53):370–418, 1763.

[5] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[6] Jason Brownlee. How to reduce variance in a final machine learning model, Apr 2021. URL: https://machinelearningmastery.com/how-to-reduce-model-variance/.

[7] Paulo Cortez and Alice Silva. Student performance data set, Nov 2014. URL: https://archive.ics.uci.edu/ml/datasets/Student+Performance.

[8] Paulo Cortez and Alice Maria Gonçalves Silva. Using data mining to predict secondary school student performance. 2008.

[9] Adele Cutler, D Richard Cutler, and John R Stevens. Random forests. In *Ensemble machine learning*, pages 157–175. Springer, 2012.

[10] Alexandre G Evsukoff. Inteligência computacional—fundamentos e aplicações. *E-Papers: Rio de Janeiro, RJ, Brazil*, 2020.

[11] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[12] Daniel Jurafsky and James H. Martin. Speech and language processing - logistic regression - stanford university, Dec 2021. URL: https://www.web.stanford.edu/~jurafsky/slp3/5.pdf.

[13] Daniel Jurafsky and James H. Martin. Speech and language processing - naive bayes and sentiment - stanford university, Dec 2021. URL: https://web.stanford.edu/~jurafsky/slp3/4.pdf.

[14] Daniel Jurafsky and James H. Martin. Speech and language processing (3rd ed. draft) dan jurafsky and james h. martin, Dec 2021. URL: https://web.stanford.edu/~jurafsky/slp3/.

[15] Bhavika Kanani. Activation functions in neural network, Oct 2019. URL: https://studymachinelearning.com/activation-functions-in-neural-network/.

[16] Hucker Marius. Multiclass classification with support vector machines (svm), kernel trick & kernel functions, Dec 2021. URL: https://towardsdatascience.com/multiclass-classification-with-support-vector-machines-svm-kernel-trick-kernel-functions-f9d5377d6f02.

[17] Kathrin Melcher. Regularization for logistic regression: L1, l2, gauss or laplace?, Mar 2018. URL: https://www.knime.com/blog/regularization-for-logistic-regression-l1-l2-gauss-or-laplace.

[18] Marvin Minsky and Seymour Papert. Perceptrons. 1969.

[19] Pascale Poulet-Coulibando. Early school-leavers in europe. *formations*, page 165.

[20] Sruthi E. R. Random forest: Introduction to random forest algorithm, Jun 2021. URL: https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/.

[21] Cynthia Rudin. Boosting mit 15.097 course notes cynthia rudin, 2012. URL: https://ocw.mit.edu/courses/sloan-school-of-management/15-097-prediction-machine-learning-and-statistics-spring-2012/lecture-notes/MIT15_097S12_lec10.pdf.

[22] Cynthia Rudin. Prediction: Machine learning and statistics - decision trees - mit opencourseware, 2012. URL: https://ocw.mit.edu/courses/sloan-school-of-management/15-097-prediction-machine-learning-and-statistics-spring-2012/lecture-notes/MIT15_097S12_lec08.pdf.

[23] Cynthia Rudin. Support vector machines, 2012. URL: https://ocw.mit.edu/courses/sloan-school-of-management/15-097-prediction-machine-learning-and-statistics-spring-2012/lecture-notes/MIT15_097S12_lec12.pdf.

[24] Boaz Shmueli. Multi-class metrics made simple, part ii: The f1-score, Jul 2020. URL: https://towardsdatascience.com/multi-class-metrics-made-simple-part-ii-the-f1-score-ebe8b2c2ca1.

[25] Koo Ping Shung. Accuracy, precision, recall or f1?, Apr 2020. URL: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9.

[26] Ryan Tibshirani. 6.1 gradient descent: Convergence analysis - cmu statistics, 2013. URL: https://www.stat.cmu.edu/~ryantibs/convexopt-F13/scribes/lec6.pdf.

[27] Wikipedia. Agrupamentos de escolas, Jul 2020. URL: https://pt.wikipedia.org/wiki/Agrupamentos_de_escolas.

[28] Wikipedia. Validação cruzada, Aug 2020. URL: https://pt.wikipedia.org/wiki/Valida%C3%A7%C3%A3o_cruzada.