Universidade Federal do Estado do Rio de Janeiro
Centro de Ciências Exatas e Tecnologia
Escola de Informática Aplicada

UM ESTUDO SOBRE TROPISMO DO HIV COM *MACHINE LEARNING*

RITA DE CÁSSIA MENEZES SOARES

Orientadora
GEIZA MARIA HAMAZAKI DA SILVA

Coorientadora
LETÍCIA MARTINS RAPOSO

Rio de Janeiro, RJ – Brasil
Junho de 2021

Catalogação informatizada pelo autor

S676

Soares, Rita de Cássia Menezes
Um estudo sobre tropismo do HIV com machine learning / Rita de Cássia Menezes Soares. -- Rio de Janeiro, 2021.
86 f.

Orientadora: Geiza Maria Hamazaki da Silva.
Coorientadora: Letícia Martins Raposo.
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal do Estado do Rio de Janeiro, Graduação em Sistemas de Informação, 2021.

1. HIV. 2. Machine Learning. 3. Stacking. 4. Tropismo Viral. I. Silva, Geiza Maria Hamazaki da, orient. II. Raposo, Letícia Martins, coorient. III. Título.

UM ESTUDO SOBRE TROPISMO DO HIV COM *MACHINE LEARNING*

RITA DE CÁSSIA MENEZES SOARES

Projeto de Graduação apresentado à Escola de Informática Aplicada da Universidade Federal do Estado do Rio de Janeiro (UNIRIO) para obtenção do título de Bacharel em Sistemas de Informação.

Aprovado por:

_____

Profª. GEIZA MARIA HAMAZAKI DA SILVA, D. Sc (UNIRIO)

_____

Profª. LETÍCIA MARTINS RAPOSO, D. Sc (UNIRIO)

_____

Prof. PEDRO NUNO DE SOUZA MOURA, D. Sc (UNIRIO)

RIO DE JANEIRO, RJ – BRASIL.

JUNHO DE 2021

Agradecimentos

Este TCC foi elaborado a partir de um trabalho de Iniciação Científica (IC), e portanto primeiramente agradeço à Letícia Raposo, minha orientadora, membro da banca e uma grande inspiração como pessoa.

Agradeço à Geiza Maria Hamazaki pela orientação também, compreensão e conversas ao longo da elaboração deste trabalho, que mudou de temática durante a execução.

Agradeço ao professor Pedro Moura pela participação em banca.

Gratidão também pela contribuição direta ou indireta, porém positiva, dos excelentes professores que conheci na minha jornada pela UNIRIO, que foi longa:

Do DMQ: Alexandre Sousa, Bruno Simões, Felipe Rafael, Luciane Velasque, Maria Beatriz Cunha, Maria Tereza Serrano, Steven Dutt Ross.

Do DMat: Beatriz Malajovich, Dante Machado e Silva, Fábio Simas, Luiz Amancio Junior, Luzia Tonon.

Da EIA: Adriana Alvim, Alexandre Andreatta, Bruna Diirr, Gleison Santos, Jefferson Simões, Jobson Massollar, Kate Revoredo, Leonardo Guerreiro, Leonardo Rocha, Márcio Barros, Mariano Pimentel, Morganna Carmem Diniz, Sidney Lucena, Simone Bacellar, Vânia Félix Dias.

Do IB/IBIO e da Medicina: Alcides Guarino, Agostinho Alves, Angelo Malaquias, Claudia Netto, Cleonice Bento, Edwin Rojas, Jaime Villas da Rocha, Jefferson Oliveira da Silva, Joelma Freire De Mesquita, Laura Jane, Rafael Fortes, Renato Geraldo Filho, Ricardo Campos-da-Paz, Rosa Haido, Tânia Valente, Terezinha Agra Belmonte.

Devo muito às instituições que me auxiliaram, à todo o corpo docente e discente da UNIRIO e funcionários técnico administrativos sempre muito prestativos e simpáticos (cito aqui Douglas Brito, Juliano Braz e Ciara Sampaio, com quem tive maior contato); além da FAPERJ e CNPQ, cujo auxílio financeiro foi crucial no desenvolvimento de pesquisas por mim e colegas.

Agradeço aos meus amigos por terem feito parte dessa jornada e cito nominalmente Juliana Loureiro, Gisele Hottz, Mariana Souza e Marcos Júnior, obrigado por tudo. Gratidão em especial aos amigos e colegas que dividiram momentos importantes comigo durante o bacharelado em Sistemas de Informação.

Minha gratidão maior vai para minha mãe, Maria Luiza, pelo apoio e amor incondicional. Ao restante da minha família, meu pai José Pacheco, e a todos os meus irmãos, meu muito obrigado também pela presença na minha vida, amor e ajuda constante.

E por último, mas não menos importante, agradeço a Thiago Severo pela presença.

RESUMO

O HIV é o vírus da imunodeficiência humana, e seu tropismo indica com qual tipo de célula ele se liga. Alguns testes para determinar o tropismo focam no genoma (genotípicos) e no geral são criados com base no subtipo viral B e têm desempenho variado para outros subtipos. O objetivo foi medir o desempenho de algoritmos genotípicos, voto majoritário e *stacking* no subtipo C. Testes genotípicos e medidas de desempenho foram selecionados por revisão de literatura. A base de dados é a de Los Alamos para o subtipo C. T-CUP apresentou o melhor desempenho, com coeficiente de correlação de Matthews (MCC), acurácia e Kappa maior, porém Geno2Pheno com ponto de corte 0,20 mostrou maior sensibilidade. *Stacking* melhorou o desempenho com relação a eles, e SIMCA (*Soft Independent Modelling of Class Analogies*) atingiu médias de sensibilidade de 94% e especificidade de 72%.

Palavras-chave: HIV, Machine Learning, Stacking, Tropismo Viral.

ABSTRACT

HIV is the human immunodeficiency virus and its tropism indicates which type of cell it binds to. Some tests to determine tropism are genome based (genotypic), and created for viral subtype B, having varying performance for other subtypes. The objective was to measure the performance of genotypic algorithms, majority voting and stacking for subtype C. Genotypic tests and performance measures were selected from literature review. Subtype C database was obtained from Los Alamos. T-CUP had better performance, with higher Matthew's Correlation Coefficient (MCC), accuracy and Kappa, but Geno2Pheno with 0.20 cutoff showed higher sensitivity. Stacking improved upon their results and SIMCA (Soft Independent Modelling of Class Analogies) algorithm had mean sensitivity of 94% and mean specificity of 72%.

Keywords: HIV, Machine Learning, Stacking, Viral Tropism.

Lista de Abreviaturas e Siglas

Ac	Acurácia
BDT	*Boosted Decision Tree*
Esp	Especificidade
HIV	Vírus da imunodeficiência humana
IA	Inteligência artificial
KNN	*k-Nearest Neighbors*
MCC	Coeficiente de correlação de Matthews
ML	*Machine Learning*
NN	*Neural Network*
RF	*Random Forest*
Sen	Sensibilidade
SVM	*Support Vector Machine*

Índice

Índice de Tabelas

Índice de Figuras

Índice de Equações

Apêndices

1 Introdução

1.1 Motivação

A partir de meados da década de 1990, é possível notar o aumento do poder computacional e da disponibilidade de informações (*Big Data*), que possibilitaram a aplicação de algoritmos mais complexos aos mais diversos tipos de problemas (MARR, 2016). Entre esses algoritmos, estão os de *Machine Learning* (ML). ML se refere a uma série de técnicas que podem ser aplicadas na geração de novas informações, classificação, predição de valores numéricos (regressão) ou para gerar *insights* diversos sobre um conjunto de dados.

Dada a importância da Saúde para uma vida de qualidade, decidiu-se estudar esse tema com o uso de ML relacionando-o com uma síndrome que impacta muitas pessoas mundialmente: a aids.

O vírus da imunodeficiência humana, HIV, é responsável pelo desenvolvimento da aids, síndrome que afetava cerca de 38 milhões de pessoas mundialmente em 2019, das quais eram estimadas 920 mil no Brasil (GHO, 2019a; GHO, 2019b). O vírus infecta células do sistema imune que expressam a glicoproteína CD4, comumente linfócitos T CD4+, o que causa a evolução dos sintomas característicos da síndrome: surgimento de infecções oportunistas, tuberculose, sarcoma de Kaposi etc (KUMARI et al., 2017).

Um vírus possui material genético que pode ser DNA ou RNA, e utiliza o maquinário celular de organismos vivos para se replicar (WU, 2020). O HIV, vírus da imunodeficiência humana, é um vírus de RNA e atualmente possui algumas variantes de importância: HIV-1 e HIV-2, com seus grupos e subtipos. De interesse para este trabalho há a variante HIV-1, grupo M e seus subtipos, em especial B e C (KUIKEN et al., 1999).

O processo de entrada do HIV nas células ocorre da seguinte maneira (ilustrado na Figura 1): a proteína gp120, do envelope viral, se liga ao CD4 da célula alvo (*Host Cell* na imagem). Essa ligação causa uma mudança na configuração da gp120, expondo suas partes internas para ligação com um receptor de quimiocina da célula, que pode ser

CCR5 ou CXCR4. Esses passos estão ilustrados em destaque na imagem. Receptores de quimiocina são proteínas que se ligam a outras em circulação no sangue (CABRAL, 2014).

Após essa segunda ligação, uma outra proteína do envelope viral, gp41, promove a fusão da cápsula viral com a membrana celular (passo 2 no diagrama), o que permite a injeção do conteúdo do vírus no citoplasma, interior da célula (CABRAL, 2014). Assim sendo, chama-se de R5 o vírus que tem tropismo e infecta apenas células que expressam o correceptor CCR5; chama-se de X4 o vírus que tem tropismo e infecta apenas células que expressam o correceptor CXCR4 e chama-se de R5X4 o vírus capaz de infectar células com quaisquer desses correceptores. O tropismo viral é a capacidade de um vírus de se ligar apenas a células específicas (RAYMOND et al., 2012).

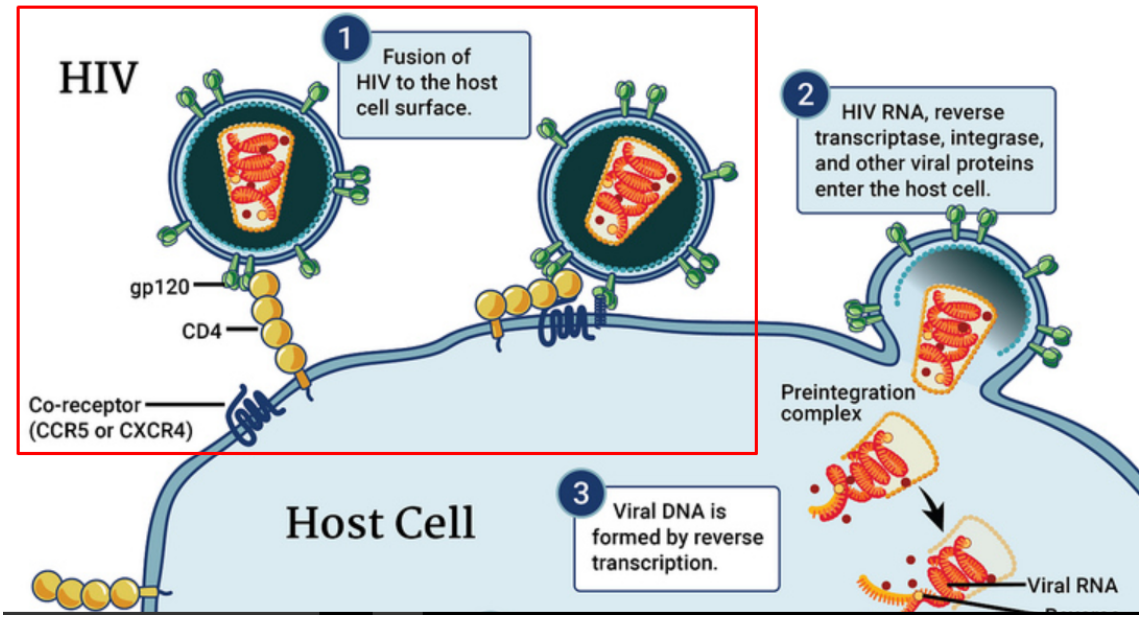

Figura 1. Ilustração da entrada do HIV na célula alvo.

Fonte: NIH, 2021 (adaptado).

Recentemente foram aprovados medicamentos inibidores de ligação entre o vírus e o receptor CCR5, como Maraviroque (IWAMOTO et al., 2010). De acordo com notas técnicas do Ministério da Saúde (2012a, 2012b), o tratamento com esse tipo de medicamento tem como uma de suas exigências a identificação e confirmação do tropismo da população viral do paciente como sendo R5, uma vez que a presença de X4

ou R5X4 em proporção maior que 2% leva a uma seleção positiva desta variante. A seleção positiva ocorre devido à eliminação dos vírus R5, porém não dos demais. Vírus com tropismo a CXCR4 estão associados a progressão mais rápida da doença e pior prognóstico clínico e, por isso, essa possibilidade deve ser evitada (CABRAL, 2014; SWENSON et al., 2012).

Para identificação do tropismo da carga viral do paciente existem diversos métodos, que são classificados como fenotípicos ou genotípicos. Os métodos fenotípicos normalmente identificam o tropismo por meio do comportamento do vírus em determinados tipos de cultura celular. Em geral, possuem alta especificidade e sensibilidade para detecção de X4 ou R5X4, porém, devido ao alto custo e restrições logísticas, sua ampla aplicação na prática clínica é limitada (LIN e KURITZKES, 2009). Os métodos genotípicos são uma alternativa mais barata e rápida e que, em muitos casos, ainda mantém alta sensibilidade para detectar vírus X4 ou R5X4. Eles são feitos usando técnicas estatísticas e de bioinformática no genoma de vírus obtidos de pacientes ou com tropismo já conhecido. A parte específica do genoma utilizada costuma ser a terceira região hipervariável (alça V3) da gp120, identificada como o fator mais importante para a interação com o correceptor de quimiocina e, consequentemente, para o tropismo (CABRAL, 2014; LIN e KURITZKES, 2009). Porém, a maioria dos testes genotípicos foi desenvolvida com base no genoma do subtipo B do HIV-1, que é um dos mais comuns na América do Norte e Europa (TEBIT e ARTS, 2011), e sendo assim podem não ser ideais para outros tipos.

Este trabalho faz portanto um panorama dos principais testes genotípicos existentes e sua metodologia, focando em analisar a concordância entre eles e seus desempenhos no subtipo C, que tem alta prevalência no Brasil (GRÄF e PINTO, 2013; GRÄF et al., 2016), além de buscar implementar técnicas novas para identificar o tropismo viral. Os testes genotípicos são combinados com voto majoritário e *stacking*, abordados ao final da seção 2.1, para verificar o impacto no desempenho.

Nas técnicas de ML, buscou-se também aumentar a sensibilidade sempre que possível. Quanto maior a sensibilidade, menor a existência de falsos negativos, ou seja, neste caso, populações virais classificadas com tropismo R5 e que na verdade tenham tropismo X4 ou R5X4. Testes genotípicos já existentes têm boa especificidade, porém, conforme resultados obtidos por Gupta et al (2015) para este subtipo, apenas em poucos casos sua sensibilidade atinge mais do que 90%. Portanto, além de atualizar o resultado

com novas sequências, este estudo busca aumentar este parâmetro para possibilitar o uso de medicamentos como Maraviroque com maior segurança.

Além do problema do tropismo do HIV, ML possui aplicações em grandes áreas técnico-científicas e artísticas da sociedade, como por exemplo na Indústria (MAYR et al. 2019), na Educação (KUCAK et al., 2018) e na Música (DANNENBERG et al., 2004), bem como em outros temas na área da Saúde, como: classificação de risco para diversas doenças, análise de usos de drogas em testes clínicos aleatorizados, prognóstico de cânceres a partir de análise genômica etc (BEAM e KOHANE, 2018).

Este estudo é resultado de projeto de Iniciação Científica e seus resultados parciais foram apresentados no XX Simpósio Brasileiro de Computação Aplicada à Saúde - SBCAS (MENEZES e RAPOSO, 2020a) e na 19º Jornada de Iniciação Científica da UNIRIO - JIC, modalidade assíncrona, sendo premiado nesta última como o melhor trabalho da categoria Matemática e Estatística (UNIRIO, 2020).

1.2 Objetivos

Este trabalho tem os seguintes objetivos:

Avaliar o desempenho dos algoritmos genotípicos na predição do uso de correceptores do subtipo C do HIV-1 com fenótipos biológicos conhecidos e verificar se voto majoritário e *stacking* melhoram o desempenho da classificação do tropismo, especialmente para sensibilidade.

Aprofundamento no tema de ML na área da Saúde.

Gerar valor para sociedade com desenvolvimento de novas ferramentas, metodologias e interfaces.

1.3 Organização do texto

O presente trabalho está estruturado em capítulos e, além desta introdução, será desenvolvido da seguinte forma:

Capítulo II: Revisão bibliográfica - este capítulo esclarece o estado da arte atual para o problema de tropismo do HIV, descrevendo os testes genotípicos e técnicas referenciadas neste trabalho.

Capítulo III: Ferramentas e tecnologias - esclarece quais foram as ferramentas, bases de dados e metodologia utilizados na elaboração deste estudo.

Capítulo IV: Resultados e discussão - nesta seção são apresentados gráficos e tabelas e é realizada uma discussão dos resultados obtidos juntamente com o que existe na literatura atual.

Capítulo V: Conclusões - Reúne as considerações finais, assinala as contribuições da pesquisa e sugere possibilidades para aprofundamento posterior.

2 Revisão bibliográfica

Este capítulo aborda as principais técnicas de *Machine Learning* (ML) utilizadas atualmente; como é feito o pré-processamento dos dados e o treinamento dos algoritmos de ML; os algoritmos genotípicos existentes na literatura para estimar o tropismo e as medidas de desempenho e resumo que podem ser utilizadas para avaliar resultados de algoritmos de ML.

2.1 Técnicas de *Machine Learning*

Existe uma gama variada de algoritmos de *Machine Learning* (Aprendizagem de Máquina). Nesta subseção são apresentadas as ideias por detrás das principais abordagens de algoritmos para aprendizagem supervisionada.

Aprendizagem de Máquina se refere a um conjunto de algoritmos que podem ser utilizados para prever algum resultado, seja ele numérico ou uma classificação, separar os dados em grupos ou obter algum tipo de conhecimento sobre a população.

Por exemplo, supõe-se que se deseja prever a altura de uma pessoa (variável desfecho, resultado de interesse) por meio de informações de outros dados ou variáveis, como altura dos pais, sexo biológico e idade. Este exemplifica um problema de regressão, no qual o desfecho é numérico ou quantitativo. Algoritmos para problemas de regressão podem ser aplicados sobre as variáveis disponíveis a fim de se prever o desfecho (altura), que neste caso é conhecido para as observações da amostra, mas não para novas.

Um outro exemplo, para desfecho qualitativo: busca-se prever se num determinado dia irá chover ou não. As variáveis de entrada são os históricos de pressão atmosférica, umidade, velocidade do vento, concentração de diferentes gases e se choveu ou não no dia. Algoritmos para problemas de classificação podem ser utilizados para prever se ocorrerá chuva. Este exemplo também demonstra um desfecho binário ou dicotômico, no qual a variável de interesse possui apenas dois valores possíveis.

Ambos os exemplos mencionados são de aprendizagem supervisionada. A aprendizagem supervisionada é feita quando se conhece o valor da variável de desfecho para as observações. Assim, busca-se obter um modelo que seja eficiente em prever o valor do desfecho. Na aprendizagem não supervisionada, busca-se obter uma separação dos dados em grupos e categorias, ou descrição dos mesmos. Não há variável de desfecho, pois se trata de uma análise exploratória (FACELI et al., 2011).

Nas seções a seguir serão apresentadas as técnicas de aprendizagem supervisionada mais utilizadas.

I. Classificadores Bayesianos

Os Classificadores Bayesianos são algoritmos probabilísticos, baseados no Teorema de Bayes (Equação 1), e entre os mais populares destaca-se o Naive Bayes. Estes calculam a distribuição da variável desfecho a partir das distribuições de probabilidade e frequência das demais variáveis, e em seguida utilizam as probabilidades condicionais para definir a classe de uma nova observação.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Equação 1. Teorema de Bayes.

Fonte: FACELI et al., 2011.

Na Equação 1 tem-se que:

- P(A|B): a probabilidade de um evento A ocorrer, dado que ocorreu um evento B.
- P(B|A): probabilidade de B ocorrer, dado que A já ocorreu.
- P(A): probabilidade de A ocorrer.
- P(B): probabilidade de B ocorrer.

Em termos mais simples, para eventos independentes o teorema usa as probabilidades já conhecidas (variáveis de entrada; na Equação 1 é apenas a variável B) para calcular a probabilidade condicional de interesse (desfecho, variável A).

O Naive Bayes, em particular, é um algoritmo simples, mas é ideal para dados naturalmente tendenciosos, que contém mais elementos em uma classe do que nas demais. Por exemplo, levando em consideração a presença de determinada doença em uma população, é natural que haja mais indivíduos saudáveis do que doentes. Porém, caso os dados não sejam verdadeiramente enviesados (ou seja, houve erro de coleta), pode ter baixo desempenho.

Como não se aproveita da inteligência nos dados, reduzindo-se a sua distribuição, pode não ser ideal para problemas mais complexos ou dos quais se deseja obter *insights* mais sofisticados, porém a sua simplicidade promove sua eficiência computacional. No cenário ideal, as variáveis componentes do problema são independentes entre si. Caso não sejam independentes, a eficácia preditiva do algoritmo é prejudicada (FACELI et al., 2011).

II. KNN - *k-Nearest Neighbors*

Esta técnica é baseada no conceito de vizinhança, no qual dados similares tendem a se concentrar e a estarem próximos de alguma maneira. Ou seja, assume que uma observação terá classificações ou valores similares aos seus vizinhos, e utiliza a informação deles para obter o desfecho.

Inicialmente, este algoritmo posiciona cada observação da base de dados num espaço *n*-dimensional a partir do valor de suas variáveis. Após, para cada nova observação que se deseja prever o valor do desfecho, é necessário posicioná-la nesse espaço e tomar a decisão com base nos valores de seus *k* vizinhos mais próximos. A distância pode ser calculada de várias maneiras, mas usualmente é a distância euclidiana entre os pontos. Para problemas de classificação, a classe de uma nova observação será a presente em maior quantidade nos vizinhos. Se for um problema de regressão, o valor esperado da nova observação será a média ou mediana dos vizinhos (FACELI et al., 2011).

Este algoritmo é interessante para dados que já possuem alguma relação posicional. Porém, necessita de grande poder de processamento para calcular a distância de novas observações, além de consumir muita memória, visto que não gera fórmulas ou inteligência sobre os dados, trabalhando apenas com as observações diretamente. Uma outra vantagem é poder utilizar dados quantitativos ou qualitativos. Entretanto,

este algoritmo é afetado por atributos não normalizados. Não sendo ideal para problemas com muitas variáveis, tanto pelo custo dos cálculos, como pelo fato da distância entre vizinhos mais próximos e mais distantes poder ser ínfima (FACELI et al., 2011).

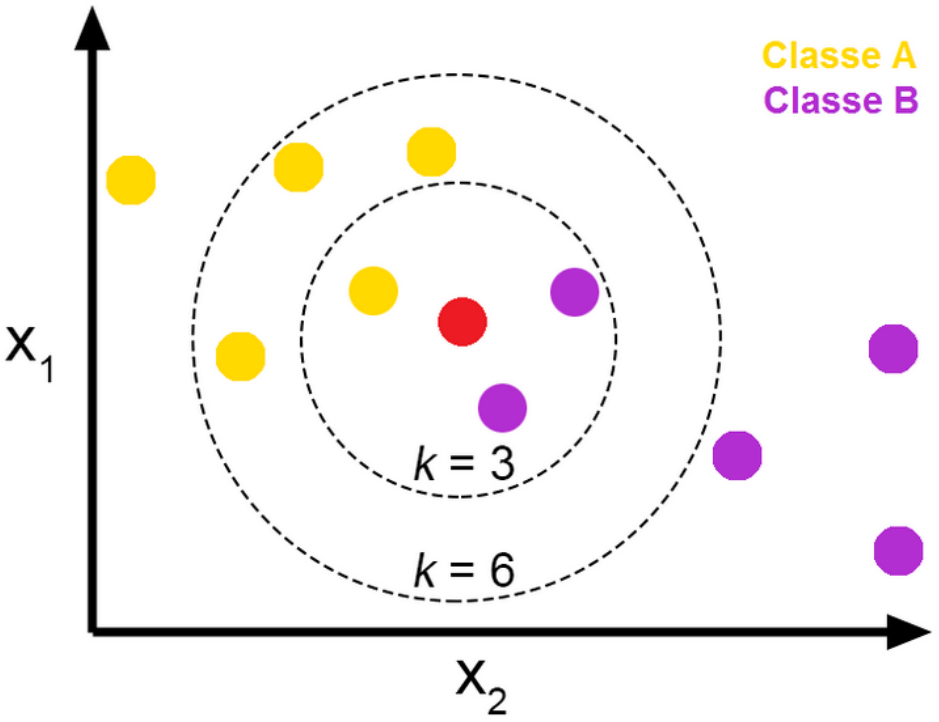

Figura 2. Ilustração do funcionamento do KNN.

Fonte: JOSÉ, 2018.

Na Figura 2, um exemplo de aplicação do algoritmo para um problema de classificação. Ao ser adicionada uma nova observação (em vermelho, no centro), ela poderá ser classificada como a classe A, amarela, ou como a classe B, roxa. Ao se definir uma vizinhança de tamanho 3 ($k = 3$), existem 2 vizinhos roxos e 1 amarelo e, portanto, a nova observação será classificada como roxa. Ao se aumentar a vizinhança, no entanto, existirão mais vizinhos amarelos, e a nova observação será classificada como A.

III. Árvores de decisão e regressão

Estes algoritmos se baseiam na divisão dos dados originais em níveis utilizando a estrutura de dados denominada árvore, que é um grafo acíclico com nós de divisão e folhas (nós sem sucessores). Com essa estratégia de dividir para conquistar, em cada nível é aplicado o mesmo conjunto de passos.

No primeiro nível encontram-se todos os dados; no nível seguinte é feita uma divisão (teste condicional) com base em alguma variável independente de modo a maximizar ou minimizar alguma métrica, como a entropia, para assim uniformizar os dados sucessivamente até que nas folhas, últimas camadas da árvore, restem apenas os dados divididos pela variável de desfecho. Não necessariamente todas as variáveis são utilizadas, apenas as consideradas de maior importância pelo algoritmo.

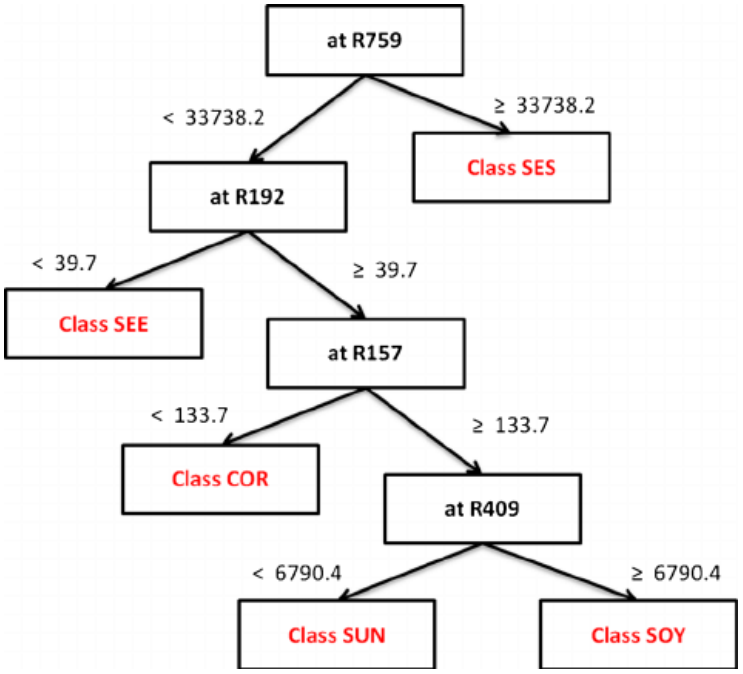

Figura 3. Ilustração do funcionamento de uma árvore de decisão simples.
Fonte: RUIZ-SAMBLÁS et al., 2014.

A decisão final é tomada como uma série de decisões anteriores. Na Figura 3 exemplo do seu funcionamento e também da estrutura de uma árvore. Na raiz é escolhida uma variável significativa (R759) e com base em seu valor é feito um teste condicional. Se o valor for menor que 33738,2, a observação de entrada segue pelo ramo da esquerda. Se for maior ou igual, segue pelo ramo da direita. No ramo da direita

há apenas uma folha (classe sem nós filhos), e, se a observação chegou a esse ponto, é definida como a classe da folha (nesse caso, SES).

É um método visual, de simples entendimento e altamente flexível, pois não pressupõe nenhuma distribuição nos dados. Este seleciona os atributos de maior relevância e é robusto a variáveis irrelevantes e a *outliers*, que são observações cujos valores são muito destoantes dos demais. Como desvantagens estão a replicação de decisões em ramos diferentes - o que dificulta a visualização de variáveis de importância - e a dificuldade em lidar com valores inexistentes e em como estes devem ser classificados (FACELI et al., 2011).

IV. Regressão linear

A Regressão linear utiliza uma equação de primeiro grau e pesa as variáveis independentes para prever o valor numérico de uma variável dependente. É uma técnica simples e de fácil compreensão, visto que fornece, ao final, o peso de cada variável para o resultado. Pode ser usada quando a variável de desfecho é numérica. A desvantagem é a dificuldade de prever relações não lineares entres as variáveis (YAN e GANG SU, 2009).

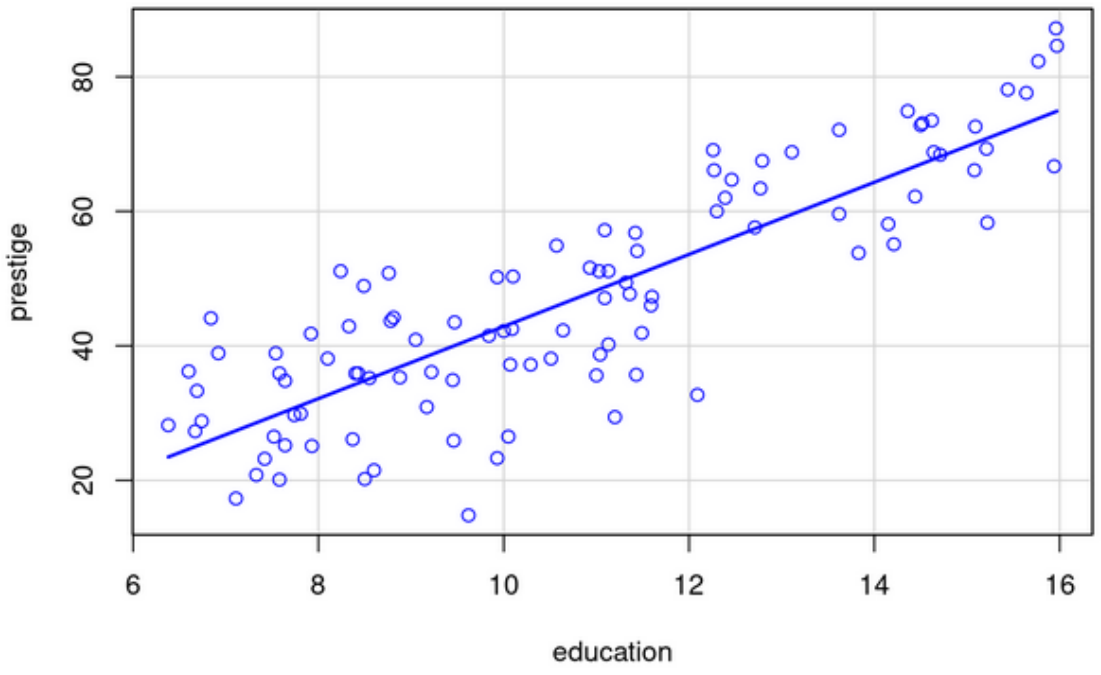

Figura 4. Reta de regressão.

Fonte: FREIRE, 2021.

Na Figura 4 um exemplo de reta de regressão. Novas observações, a partir de seus valores em *education* (variável independente) terão a variável de interesse, *prestige*, calculada como um ponto na reta azul.

Essa reta é montada a partir de uma fórmula no modelo da Equação 2, calculada de modo a minimizar a distância entre todos os pontos e a reta de regressão.

$$Y_i = \beta_1 X_i + \beta_0 + \epsilon_i$$

Equação 2. Equação do modelo de regressão.

Fonte: FREIRE, 2021.

Na equação:

- Y_i é o valor da observação *i* no eixo y (no exemplo, seria a variável dependente *prestige*).
- β_1 é o coeficiente angular da reta, ou sua inclinação, e indica também o peso ou grau de influência das variáveis uma na outra.
- X_i é o valor da observação *i* no eixo x (na imagem, *education*).
- β_0 é a interseção da reta com o eixo y (não representada na imagem, porém a extensão da reta ficaria próxima ao valor de y = 20).
- Єi representa o erro da observação *i*, ou sua distância até a reta de regressão. É estimado que os erros seguem uma distribuição de probabilidade com média 0 e variância = σ^2 (FREIRE, 2021).

V. Regressão logística

Esta técnica é similar à regressão linear, porém a variável de desfecho é um classificador com dois resultados possíveis. É muito utilizada na área da saúde para prever desfechos como “Doente” e “Não doente”, ou com tratamento e sem tratamento, especialmente por mostrar a ordem de importância das variáveis de entrada para a

classificação final (por exemplo, o quanto idade é importante para o desfecho de doença cardíaca).

Além disso, é capaz de controlar e eliminar, até certo ponto, efeitos de confusão e dependência entre as variáveis. Este algoritmo possui as mesmas restrições da regressão linear, não se adaptando bem a problemas cujas relações entre as variáveis e o desfecho não são lineares (TOLLES e MEURER, 2016).

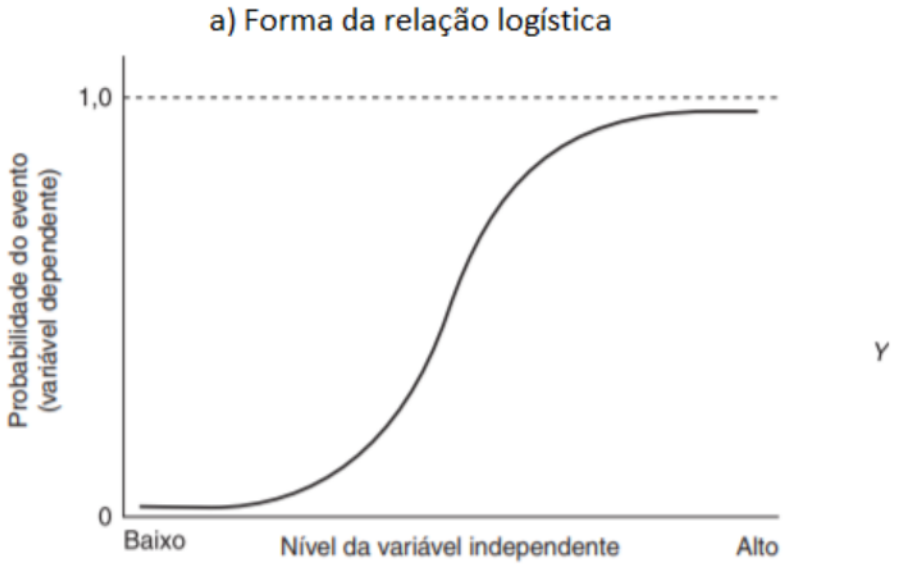

Figura 5. Representação da regressão logística.

Fonte: BATTISTI e SMOLSKI, 2021.

Na Figura 5, no eixo Y está a variável dependente, qualitativa dicotômica, e seus valores possíveis nas observações são apenas 0 ou 1, com 1 representando ocorrência do evento de interesse, e 0 a não ocorrência.

A variável independente se encontra no eixo X. Após resultado da regressão, é delineada a curva logística, que representa a relação de probabilidade entre as variáveis. Com o valor de uma nova observação no eixo X, o valor no eixo Y é estimado a partir do ponto correspondente na curva. O valor estimado será sempre entre 0 e 1, e indicará a probabilidade do evento Y ocorrer (valores acima de 0,50 são considerados ocorrência do evento de interesse) (BATTISTI e SMOLSKI, 2021).

Uma outra maneira de visualizar a relação entre as variáveis a partir da regressão logística é com as razões de chance (*odds ratio*). Ao se calcular o modelo da regressão logística, as variáveis independentes terão coeficientes, similares ao da regressão linear, que expressam o quanto cada variável independente interfere na dependente.

Então, por exemplo, se o coeficiente para uma variável X no modelo é 0,2, a sua razão de chance para a variável Y é $e^{0,2} = 1,221$. Como esse valor é maior do que 1, isso indica que a variável X impacta para aumentar o valor da variável Y. Ou seja, para cada variação unitária em X, a chance de Y ocorrer aumenta em 1,221 vezes (ou um aumento de 22,1%).

Valores negativos nos coeficientes como, por exemplo $e^{-0,1} = 0,905$ irão ocasionar resultados menores que 1, o que indica que a variável X contribui para diminuir o valor da variável Y. Novamente, para cada variação unitária em X, Y tem uma chance de ocorrer 0,905 menor, ou uma diminuição de 9,5% (100% - 90,5%) (BATTISTI e SMOLSKI, 2021).

VI. Redes neurais

As redes neurais são modelos baseados em otimização, ou seja, que buscam minimizar ou maximizar alguma função objetivo. Em geral, seguem a mesma abordagem: existem estruturas denominadas neurônios artificiais, inspirados nos biológicos, que recebem informação de uma camada de entrada com os dados, ou de outros neurônios.

Os neurônios processam individualmente a informação de entradas com variados pesos, e a saída no final da rede fornece um resultado de predição para variáveis quantitativas ou qualitativas.

A rede pode possuir uma quantidade variada de neurônios e camadas, bem como processar várias vezes os dados etc., a depender da customização do algoritmo em questão.

As redes neurais são úteis para diversos tipos de problemas, escaláveis, fazem boa generalização e toleram dados ruidosos, porém são difíceis de compreender em seus resultados e exigem maior poder de processamento, além de haver uma infinidade de parâmetros que podem ser ajustados, tornando sua otimização por vezes obscura (FACELI et al., 2011).

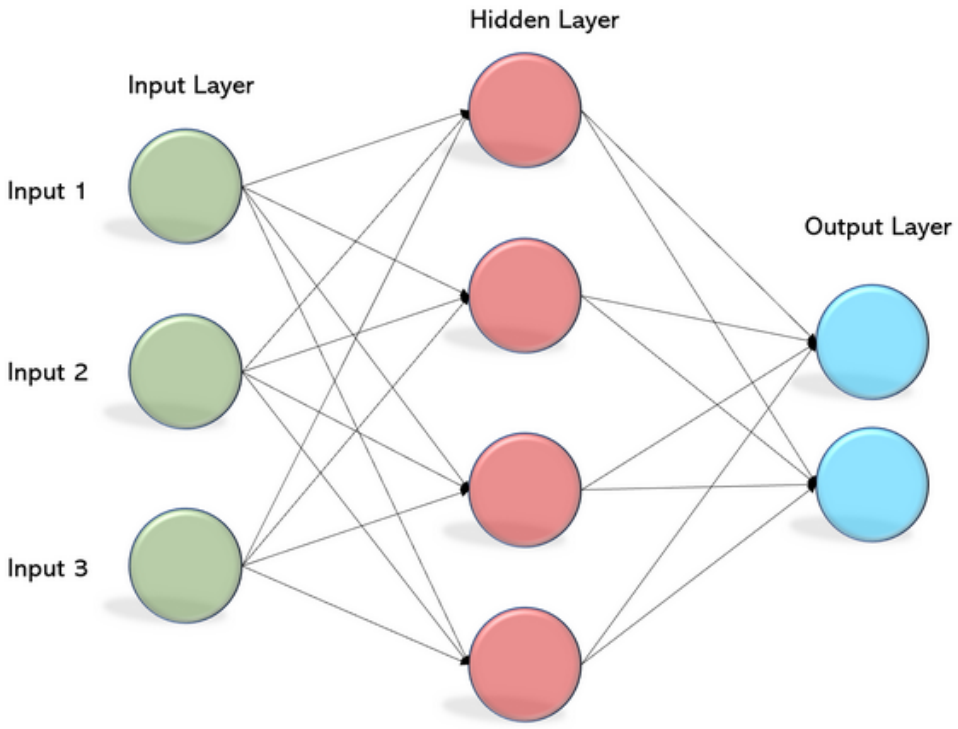

Figura 6. Ilustração do funcionamento do *Multi-Layer Perceptron*.
Fonte: MOHANTY, 2019.

Na Figura 6 está ilustrado *Multi-Layer Perceptron* (MLP), um algoritmo com uma rede neural completamente conectada (ou seja, todos os neurônios estão ligados uns nos outros) e *feedforward* (ou seja, a informação segue apenas um caminho na rede, ela não é recorrente) (MOHANTY, 2019).

VII. Máquina de vetores de suporte

Em inglês, as *Support Vector Machine* (SVM) são, assim como as redes neurais, algoritmos de otimização baseados na Teoria de Aprendizado Estatístico (TAE). Esta teoria estabelece fórmulas para a escolha de um melhor classificador a partir de um conjunto de classificadores, buscando maximizar a margem de separação entre observações de diferentes classes.

Podem ser utilizadas em problemas linearmente separáveis, ou que possam ser transformados neles por meio de uma função *kernel* (que pode fazer com que as observações, em maiores dimensionalidades, sejam separáveis por um hiperplano). A

margem de separação entre as classes pode ser rígida ou suave (esta última com variações no algoritmo).

Esta técnica possui boa generalização e tolera dados ruidosos e de alta dimensionalidade. Contudo, assim como as redes neurais, é de difícil interpretação, e pouca variação nos parâmetros pode gerar grande mudança nos resultados (FACELI et al., 2011).

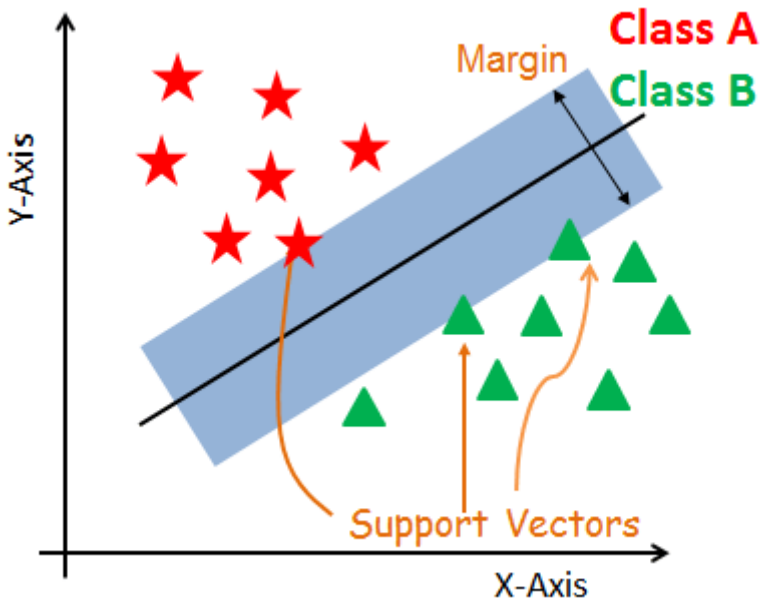

Figura 7. Ilustração do funcionamento do SVM.
Fonte: AVINASH, 2019.

Na Figura 7 uma ilustração com duas classes (*Class A* e *Class B*) que são linearmente separáveis a partir de uma reta/hiperplano. A reta e suas margens (área azul) são definidas com o auxílio de *Support Vectors* (vetores de suporte), que são as observações mais próximas do limite entre as classes (AVINASH, 2019).

VIII. PCA - análise de componentes principais

A análise de componentes principais é uma técnica estatística na qual as variáveis são combinadas a partir de sua correlação ou covariância e transformadas em outras, denominadas componentes principais. Os componentes principais possuem um peso para cada variável participante, são não correlacionados e buscam reter o máximo

possível de informação das observações e variáveis originais (VARELLA, 2008 e SILVA, 2020).

Esta abordagem é útil para resumir conjuntos de dados multivariados, e é também utilizada em muitas técnicas de ML. Dentre as técnicas que a utilizam, cita-se *Rotation Forest* e SIMCA, ambas utilizadas nesse trabalho.

Rotation Forest é uma técnica na qual são geradas múltiplas PCAs a partir de divisões do conjunto de dados original. Após, são montadas árvores de decisão com as componentes principais, e é feito floresta aleatória para resumir essas inúmeras árvores, sendo assim obtido o modelo final. Os primeiros passos estão demonstrados na Figura 8, na qual primeiro é feita a PCA e, após, são geradas árvores de decisão com os atributos (ROKACH, 2016).

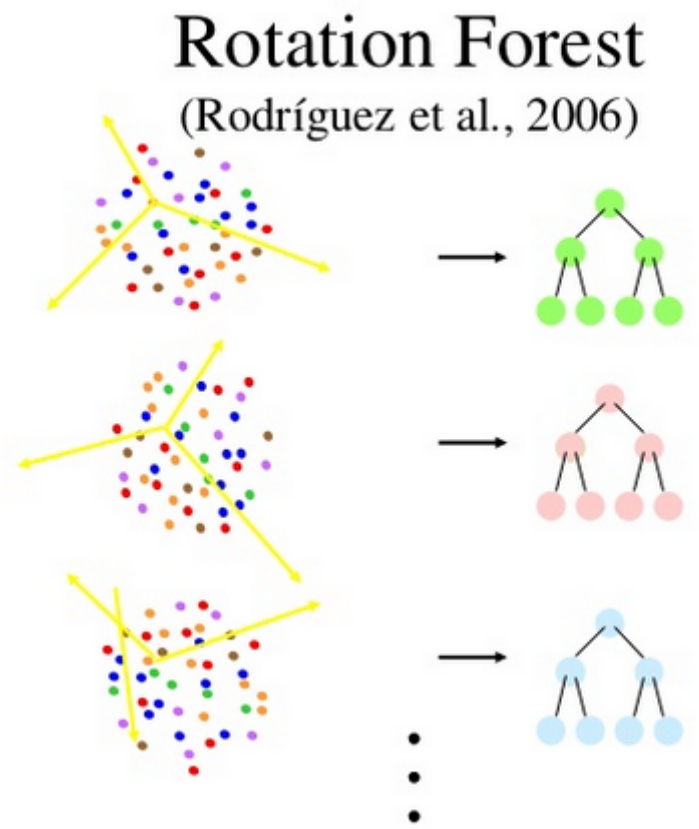

Figura 8. Ilustração do funcionamento do algoritmo *Rotation Forest*.
Fonte: ROKACH, 2015 (slide 19).

Já SIMCA (*Soft Independent Modelling of Class Analogies*) é um modelo de ML que realiza PCA várias vezes nos dados originais e classifica novas observações de acordo com a distância destas para cada componente principal gerada. Na Figura 9 estão

demonstradas as projeções das componentes principais (*Class 1, Class 2, Class 3*), e uma nova observação, o quadrado preto, que será classificada em *Class 2*, uma vez que está mais próxima desse plano (SIRVEN et al., 2007).

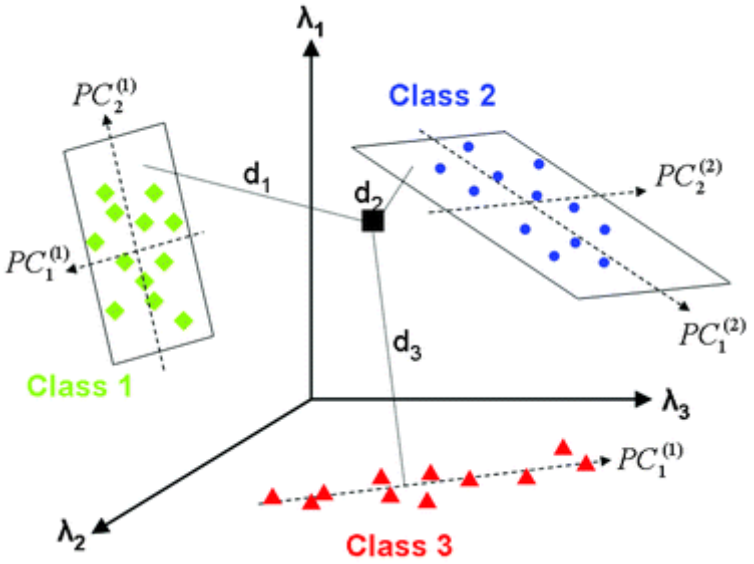

Figura 9. Ilustração do funcionamento do algoritmo SIMCA.

Fonte: SIRVEN et al., 2007.

IX. *Ensemble learning*

E*nsemble learning* (aprendizado em fila, em tradução livre) é uma técnica utilizada para melhorar algoritmos de ML que consiste na criação de um algoritmo de ML com base no resultado de outras técnicas, ou por meio da manipulação do conjunto de dados e amostras antes de treinar o algoritmo. O resultado frequentemente tem melhor desempenho e/ou maior capacidade de generalização do que sem o uso dessa abordagem. Técnicas comuns para *ensemble learning* incluem: *bagging*, *boosting*, floresta aleatória (*random forest*), voto majoritário e *stacking* (POLIKAR, 2006). Dentre as utilizadas mais diretamente neste trabalho, estão o voto majoritário e o *stacking*, abordados a seguir.

No voto majoritário, cada classificador componente possui um resultado específico para a classe de desfecho, sendo este denominado seu voto. Os votos em cada classe são somados por observação e por fim a classe mais votada (e por isso majoritária) é escolhida como a classificação resultante. O voto majoritário reduz a

variância do resultado e melhora o classificador tanto quanto mais os classificadores forem não correlacionados, uma vez que classificadores não correlacionados “votarão” de maneiras distintas (OZA e TUMER, 2008).

No *stacking* é criado um meta-classificador aplicando um algoritmo de *Machine Learning* sobre o resultado de classificadores base. Esse "empilhamento" de testes é especialmente útil quando eles têm diferentes vieses de decisão, ou seja, classificam as observações de formas distintas. Desta forma, o erro é minimizado (OZA e TUMER, 2008).

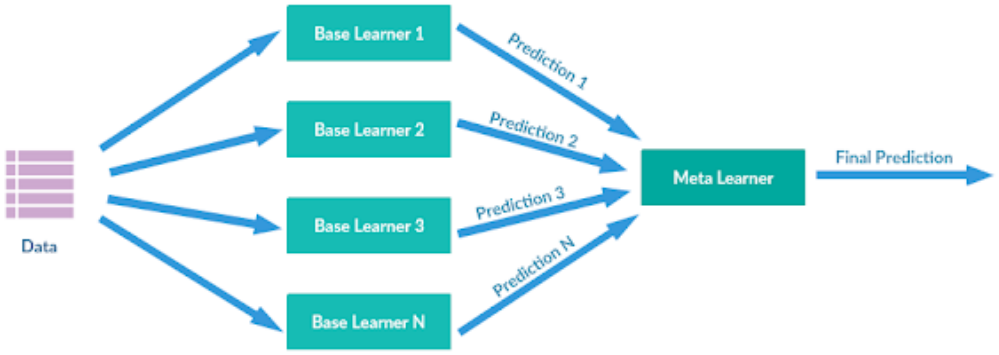

Figura 10. Ilustração do funcionamento do *stacking*.

Fonte: SETUNGA, 2016.

Na Figura 10 é ilustrado o procedimento do *stacking*. O primeiro passo é a obtenção dos dados e, após, da classificação ou regressão a partir dos classificadores base (na figura, os *Base Learner* no segundo passo do fluxo). Cada classificador base gera uma predição (*Prediction*). O conjunto de predições, bem como o valor real do desfecho, são utilizados então para treinar um algoritmo de ML de livre escolha, sendo este então chamado de metaclassificador.

Vale ressaltar que existe uma infinidade de algoritmos pertencentes a cada uma das classes apresentadas, porém estas concentram as principais estratégias de ML utilizadas atualmente.

Na seção seguinte são apresentados métodos de pré-processamento dos dados, bem como estratégias de treinamento que podem aumentar a eficiência de algoritmos de ML.

2.2 Pré-processamento e treinamento

Além da escolha das técnicas de *Machine Learning* apropriadas para determinado problema, também é necessário escolher como pré-processar os dados para obter melhor eficiência preditiva.

No pré-processamento, podem ser aplicadas algumas estratégias de modo a preparar as variáveis, as quais são descritas a seguir.

I. Eliminação de atributos irrelevantes ou redundantes

Atributos que não têm relação com o desfecho desejado, que têm o mesmo valor para todos ou praticamente todos os objetos (baixa variância), ou atributos que sejam calculados a partir de outros, por exemplo, podem ser eliminados (FACELI et al., 2011).

II. Amostragem de dados

Quanto maior a quantidade de informações, maior a eficiência preditiva, entretanto pode ser necessário reduzir o conjunto de dados por razões de desempenho ou outras restrições. Para tanto, pode-se utilizar técnicas de amostragem da estatística clássica, como amostragem aleatória simples (com ou sem reposição), amostragem estratificada e amostragem progressiva (FACELI et al., 2011).

III. Dados desbalanceados

Dados desbalanceados ocorrem quando, no desfecho, há uma quantidade maior de observações de uma classe (majoritária) do que da(s) outra(s) (minoritária(s)). Soluções para esse problema incluem: nova coleta dos dados, balanceamento artificial do conjunto e usar algoritmos nos quais possam ser adicionados pesos diferentes para cada classe, dando mais peso à(s) classe(s) minoritária(s). Dentre algumas técnicas de balanceamento artificial, destacam-se (elas são executadas apenas no conjunto de treino, abordado no final da seção 2.2) (FACELI et al., 2011):

Oversampling: seleciona observações da classe minoritária várias vezes por meio de amostragem com reposição, até igualar sua proporção à da classe majoritária. Essa técnica pode causar *overfitting*, que é quando o algoritmo prevê muito bem o desfecho no conjunto de treinamento, mas não no conjunto de teste (FACELI et al., 2011).

Undersampling: faz amostragem sem reposição da classe majoritária de modo que a sua proporção se equipare à da classe minoritária. Pode ser uma boa estratégia quando existem dados suficientes em ambas as classes, porém de qualquer modo ocorre perda de informação e há risco de ocorrer *underfitting*, que é quando o algoritmo não prevê bem resultados para o conjunto de teste e nem para o conjunto de treino (FACELI et al., 2011).

SMOTE: o SMOTE faz *undersampling* da classe majoritária, retirando aleatoriamente observações, e faz *oversampling* da classe minoritária por meio da criação de novas observações sintéticas (ou seja, não duplica a informação já existente, evitando *overfitting*). As observações originais da amostra são colocadas num espaço euclidiano com base em seu vetor de atributos. É selecionada uma observação da classe minoritária aleatoriamente e é calculado seu vizinho mais próximo. No vetor que liga esses dois vizinhos é então gerada uma observação sintética, possibilitando a expansão do espaço amostral e que algoritmos de *Machine Learning* sejam direcionados a tomar decisões mais generalistas (CHAWLA et al., 2002). Existem outras técnicas de balanceamento similares que trabalham com a geração de observações sintéticas, como ROSE (LUNARDON et al., 2014); cabe a cada pesquisador observar sua metodologia e definir a melhor para seu trabalho.

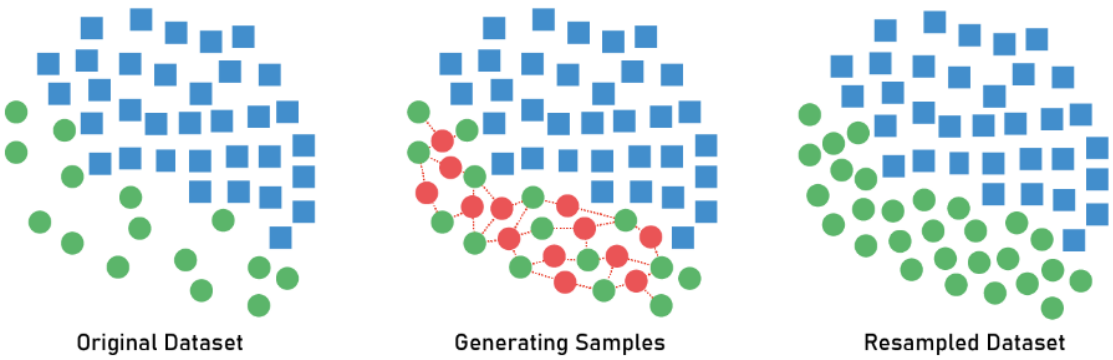

Figura 11. Ilustração do funcionamento do SMOTE.

Fonte: JEFFERSON, 2020.

Na Figura 11, após remoção de observações suficientes da classe majoritária (quadrados azuis), ou *undersampling*, tem-se o conjunto de dados denominado *Original Dataset*. Após, pares de observações da classe minoritária (círculos verdes) são escolhidos aleatoriamente e, de maneira iterativa, é gerada uma nova observação no vetor de ligação entre as observações, portanto contendo novos valores de variáveis (passo *Generating Samples*). Com isso, obtém-se um novo conjunto de dados (*Resampled Dataset*) com as classes balanceadas.

IV. Limpeza dos dados

Os dados brutos podem ter uma série de problemas, tais como:

- estarem incompletos, ou seja, com valores inexistentes;
- redundantes: com valores repetidos, sejam estes de atributos ou observações;
- inconsistentes: com valores que não fazem sentido ou que são conflitantes entre si como, por exemplo, idade de um humano superior a 200 anos, ou dois pacientes com a mesma informação, mas desfechos diferentes;
- ruidosos: com valores discrepantes, porém difíceis de detectar a olho nu, como número de filhos = 7 e idade = 12 anos.

Alguns algoritmos lidam bem com dados ruidosos, porém para os demais problemas pode ser necessária a eliminação manual de observações ou outros tratamentos específicos (FACELI et al., 2011).

V. Conversão simbólico-numérica

Para alguns algoritmos de ML é necessário que a entrada seja numérica. Caso haja variáveis qualitativas, estas deverão ser convertidas. Se a variável só tiver dois valores possíveis, estes podem ser convertidos em 0 e 1.

Se houver mais de dois valores possíveis, e não for uma variável ordinal, devem ser feitas conversões que preservem a informação de não ser ordinal, mantendo a distância entre as classes. Caso seja ordinal, uma escala numérica ordenada deverá ser utilizada, preservando a lógica da classe (FACELI et al., 2011).

VI. Conversão numérico-simbólica

Alguns algoritmos de ML também exigem a conversão oposta à anterior. Se o atributo original, quantitativo, é discreto, a conversão é direta de um valor para o outro (de 1 para "1", por exemplo). No entanto, para valores reais, é necessário definir intervalos, e cada intervalo ganhará uma classificação qualitativa. A divisão em intervalos pode ser realizada de diversas formas (FACELI et al., 2011).

Por exemplo, a variável peso é quantitativa contínua e portanto pode ser dividida em faixas de igual tamanho. Se os pesos variam de 40kg a 120kg na amostra, podem ser escolhidas faixas que os incrementem de 10kg em 10kg, como [40-50[, [50-60[, …, [100-110[, [110-120]. A classificação qualitativa a ser dada a cada faixa pode ser uma descrição simples dos valores que ela contém (por exemplo, "40-50kg") ou algo que faça sentido para a pesquisa em questão (por exemplo, "baixo peso").

VII. Transformação de valores numéricos

Pode ser necessário que valores numéricos sejam manipulados com funções, reescalados ou normalizados/padronizados para não gerarem vieses em algoritmos de ML. A normalização é uma boa alternativa para dados nos quais se espera encontrar muitos *outliers*, por lidar melhor com esse problema (FACELI et al., 2011).

Existe o conjunto de treinamento do algoritmo, e o conjunto de validação, ou teste. Esses dois conjuntos em geral vêm de uma divisão do conjunto original. Ao longo deste trabalho foram utilizadas divisões de 70% para treinamento e 30% para teste. Esta, assim como a divisão 80%/20%, é uma das mais comuns utilizadas, oferecendo bons resultados e bom aproveitamento dos dados (BREIMAN e SPECTOR, 1992).

Uma outra forma de dividir os dados é por *k-folds* (validação cruzada *k-fold* ou *k-fold cross validation*, em inglês), nos quais os dados são divididos em *k* partes de igual tamanho. Uma dessas partes é usada para teste e o restante é agregado e utilizado no treino. Este procedimento é repetido *k* vezes, sempre variando o *k-fold* escolhido para teste. No final, obtém-se médias e desvios-padrão das medidas de desempenho do algoritmo de ML, garantindo com isso que essas métricas foram bem estimadas (MARCOT e HANEA, 2020).

No conjunto de treinamento podem ser feitos balanceamentos e outras técnicas visando a melhorar os resultados, enquanto que no conjunto de validação não há alteração, excetuando-se pré-processamentos necessários.

Durante a aplicação dos algoritmos de ML também existe uma infinidade de customizações que podem ser realizadas por meio de hiperparâmetros, que são valores específicos de cada algoritmo.

2.3 Medidas de desempenho

Para avaliação do desempenho de algoritmos de ML, assim como para testes diagnósticos, podem ser utilizadas uma série de métricas, que serão abordadas nesta seção.

Avaliar o desempenho de algoritmos é uma necessidade pois tradicionalmente se testam vários algoritmos diferentes para o mesmo problema, e essa avaliação auxilia na escolha do melhor ou mais apropriado. Por exemplo, pode-se calcular o coeficiente de correlação de Matthews - MCC (abordado na seção 2.3 - V) de um algoritmo que use rede neural e de outro que use árvore de decisão para o mesmo problema e base de dados. O algoritmo melhor e mais equilibrado terá maior MCC.

Uma forma de visualizar o quão bom quantitativamente foi o desempenho de um teste para um desfecho qualitativo é com seus resultados colocados em uma matriz de confusão. A matriz de confusão possui nas colunas e nas linhas as classificações possíveis para o desfecho. Nas células pode ser observado qual foi a predição do teste (linha) e qual o valor real da observação (coluna). A matriz pode ser apresentada também com as colunas e linhas invertidas (com o valor previsto nas colunas). Com isso, a diagonal da matriz contém os resultados em que o teste acertou e, nas demais células, estão os resultados nos quais houve erro (FACELI et al., 2011). Na Tabela 1 é apresentado um exemplo para desfecho binário.

Tabela 1. Matriz de confusão 2x2.

Fonte: Elaboração própria.

		Referência	
		Positivo	Negativo
Teste	Positivo	VP	FP
	Negativo	FN	VN

Assim como na Tabela 1, as fórmulas das seções seguintes utilizam os valores de verdadeiros positivos (VP), verdadeiros negativos (VN), falsos positivos (FP) e falsos negativos (FN).

I. Acurácia

A acurácia é uma boa medida inicial para estimar o quão bom é um algoritmo, esta é frequentemente utilizada como métrica de otimização. Seu cálculo é simples, conforme Equação 3. Entretanto, como ponto negativo, tem-se que caso os dados estejam desbalanceados e o algoritmo acertar muitos exemplos da classe majoritária, a acurácia será alta, porém não representará bem a realidade (FACELI et al., 2011).

$$\text{Acurácia (Ac)} = \frac{VP + VN}{VP + VN + FP + FN}$$

Equação 3. Fórmula da acurácia.

Fonte: ŠIMUNDIĆ, 2009.

II. Sensibilidade ou *recall*

A sensibilidade indica o quão bem o algoritmo classifica exemplos da classe positiva, ou seja, a taxa de verdadeiros positivos. Caso haja mais de duas classes no problema, cada classe é considerada individualmente e comparada com as demais (a sensibilidade é calculada por classe). A sensibilidade é muito aplicada na área da saúde para saber a chance de acerto de um teste diagnóstico, por exemplo, que é uma informação

importante no tratamento de doenças. Quanto maior a sensibilidade, menor a existência de falsos negativos, conforme Equação 4 (FACELI et al., 2011).

$$\text{Sensibilidade (Sen)} = \frac{VP}{VP + FN}$$

Equação 4. Fórmula da sensibilidade.

Fonte: FACELI et al., 2011.

III. Especificidade

Indica a taxa de acerto das classes negativas. Quanto maior a especificidade, menor a existência de falsos positivos (FACELI et al., 2011). Sua fórmula está exposta na Equação 5.

$$\text{Especificidade (Esp)} = \frac{VN}{VN + FP}$$

Equação 5. Fórmula da especificidade.

Fonte: FACELI et al., 2011.

IV. Precisão

A precisão (Equação 6) é a proporção de observações positivas corretamente classificadas (FACELI et al., 2011).

$$\text{Precisão (Pre)} = \frac{VP}{VP + FP}$$

Equação 6. Fórmula da precisão.

Fonte: FACELI et al., 2011.

V. Coeficiente de correlação de Matthews (MCC)

O MCC varia entre -1 e 1, não sofre influência de dados desbalanceados e é uma das melhores medidas para avaliar desempenho de testes (POWERS, 2011). Sua fórmula está abaixo, na Equação 7.

$$\text{MCC} = \frac{(VP * VN) - (FP * FN)}{\sqrt{(VP+FP)\ (VP+FN)\ (VN+FP)\ (VN+FN)}}$$

Equação 7. Fórmula do MCC.

Fonte: POWERS, 2011.

VI. Kappa

O Kappa de Cohen é uma estatística que varia entre -1 e 1, muito utilizada para analisar concordância entre classificadores. Quanto mais próximo de 1, maior a concordância (MCHUGH, 2012). Na tabela seguinte os Kappas são apresentados quantitativamente e qualitativamente (Tabela 2).

Tabela 2. Classificação qualitativa parcial do Kappa de acordo com seu valor.

Fonte: MCHUGH, 2012.

Valor do Kappa	Concordância entre os testes
0,00 - 0,20	Nenhuma
0,21 - 0,39	Mínima
0,40 - 0,59	Fraca
0,60 - 0,79	Moderada
0,80 - 0,90	Forte
> 0,90	Quase perfeita

VII. *Diagnostic Odds Ratio* (DOR)

O DOR é uma medida de uso geral para estabelecer o desempenho de um teste, e também utilizada para comparar testes. Depende da sensibilidade e especificidade, conforme Equação 8. Seus valores variam de 0 a infinito, com valores maiores indicando testes melhores (ŠIMUNDIĆ, 2009 e GLAS et al., 2003).

$$\text{DOR} = \frac{\frac{VP}{FN}}{\frac{FP}{VN}} = \frac{\frac{Sen}{1-Sen}}{\frac{1-Esp}{Esp}}$$

Equação 8. Fórmula do DOR.

Fonte: GLAS et al., 2003.

VIII. Índice de Youden

Este índice é uma das medidas mais antigas para avaliar e comparar o desempenho de testes. Varia de 0 a 1, com 1 indicando o melhor teste possível (sem erros). Sua fórmula é simples (Equação 9).

$$\text{Índice de Youden} = \text{Sen} + \text{Esp} - 1$$

Equação 9. Fórmula do Índice de Youden.

Fonte: ŠIMUNDIĆ, 2009.

Como não é sensível a valores baixos de sensibilidade ou especificidade, utilizando apenas os valores absolutos, testes com valores distantes nessas medidas podem ter índice de Youden iguais, sendo essa sua maior desvantagem. Por exemplo, o teste A tem Sen = 0,9 e Esp = 0,4, e o teste B tem Sen = 0,6 e Esp = 0,7. Embora ambos tenham índice de Youden = 0,3, o teste B é mais equilibrado (ŠIMUNDIĆ, 2009).

IX. F-score

O *F-Score* é a média harmônica entre sensibilidade e especificidade, simplificada na Equação 10. Esta varia entre 0 e 1. Seu desempenho é melhor para avaliar desfechos com uma classe dicotômica. Entre outros problemas, sofre influência de dados desbalanceados (POWERS, 2015).

$$F\text{-}score = \frac{VP}{VP + 0{,}5\ (FP + FN)}$$

Equação 10. Fórmula do *F-score*.

Fonte: POWERS, 2015.

X. Métricas de regressão

As medidas apresentadas anteriormente são utilizadas para avaliar problemas de classificação, nos quais o desfecho é qualitativo. Para desfechos quantitativos, as métricas utilizadas costumam medir de alguma maneira a distância entre o valor real (y_i) e o predito (x_i). As mais populares são o erro quadrático médio (MSE - *mean squared error* - Equação 11) e a distância absoluta média (MAD - *mean absolute distance* - Equação 12).

Quanto menores os seus resultados numericamente, melhor o desempenho do modelo em prever o desfecho. Nas Equações 11 e 12 abaixo, *n* é a quantidade de observações (FACELI et al., 2011).

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - x_i)^2$$

Equação 11. Fórmula do MSE.

Fonte: FACELI et al., 2011.

$$\text{MAD} = \frac{1}{n}\sum_{i=1}^{n} |y_i - x_i|$$

Equação 12. Fórmula da MAD.

Fonte: FACELI et al., 2011.

XI. ROC e AUC

Os testes cujo desfecho é binário, ou seja, cuja variável de interesse possui apenas dois valores possíveis (como "Sim" e "Não", por exemplo) podem ser analisados por meio do espaço/curvas ROC (*Receiving Operating Characteristics*). Trata-se de um gráfico da sensibilidade (taxa de VP) pela especificidade (taxa de FP). Valores de testes podem estar dispostos como pontos (Figura 12), ou como linhas contínuas (Figura 13), no caso de testes em que é possível variar o ponto de corte (FACELI et al., 2011).

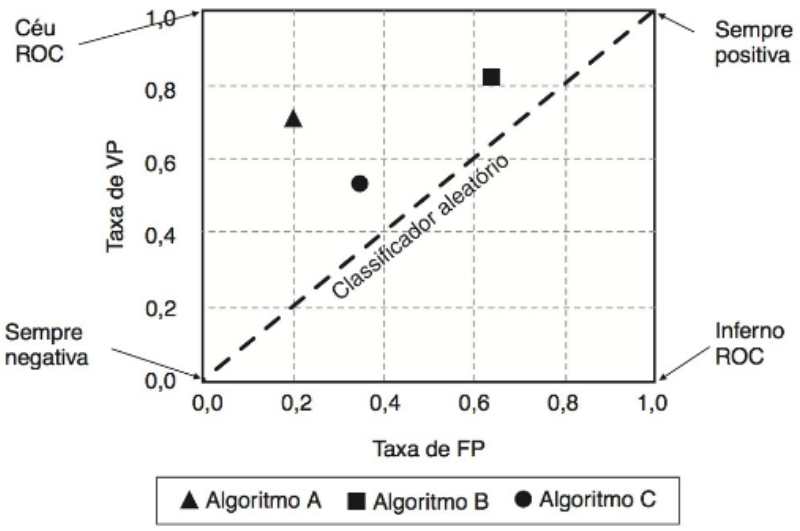

Figura 12. Espaço ROC com 3 classificadores exemplificados.

Fonte: FACELI et al., 2011.

Na Figura 12, pontos abaixo da diagonal descrita como "Classificador aleatório" indicam testes que são piores do que uma escolha aleatória das classes. Ou seja, eles são

tão bons quanto utilizar uma moeda não viciada para decidir a classificação de uma observação.

Quanto mais próximo do canto inferior direito do gráfico (chamado de “Inferno ROC”), pior é o teste, e o oposto é verdadeiro, ou seja, quanto mais próximo do canto superior esquerdo (“Céu ROC”), melhor é o teste (FACELI et al., 2011).

Para algoritmos que geram resultados contínuos, pode-se escolher uma série de pontos de corte e plotar o desempenho do teste como uma curva (Figura 13). Desta forma a então chamada curva ROC auxilia na escolha do melhor ponto. No caso das curvas, também é possível extrair uma medida única de resumo, a AUC (*Area under the ROC Curve*, área abaixo da curva ROC). Com valores entre 0 e 1, a AUC se resume ao cálculo da área gráfica abaixo da curva. Quanto maior a área, ou seja, quanto mais próximo de 1, melhor o teste (FACELI et al., 2011).

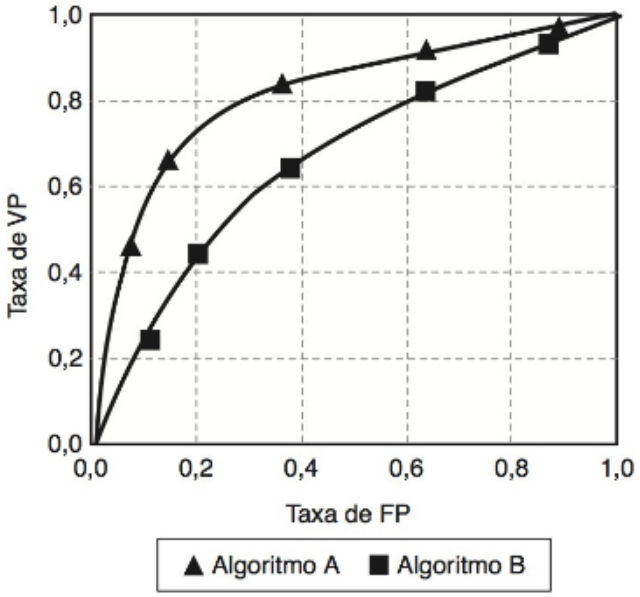

Figura 13. Exemplo de curvas ROC de dois diferentes algoritmos.

Fonte: FACELI et al., 2011.

Outras métricas que podem ser utilizadas, como auxiliares, para medir o desempenho de um teste são: os tempos de processamento e treinamento e a quantidade de memória utilizada, visto que algumas abordagens de ML podem ter um custo de execução proibitivo (FACELI et al., 2011).

As medidas Sen, Ac, Esp, MCC e o índice Kappa são abordadas ao longo dos resultados devido à sua importância na literatura relacionada (CABRAL, 2014; MASSO e VAISMAN, 2010; GUPTA et al., 2015; RIEMENSCHNEIDER et al., 2016; SOULIÉ et al., 2016; LÖCHEL et al., 2018).

2.4 Algoritmos genotípicos

Os algoritmos genotípicos, usados para classificação do tropismo do HIV, foram selecionados a partir de revisão bibliográfica e estão descritos em detalhes abaixo. Foram selecionados apenas aqueles que estavam disponíveis para utilização, seja *online*, por módulo de *software* instalável ou para implementação. São estes: Geno2Pheno (LENGAUER et al., 2007), Web PSSM (JENSEN et al., 2003; BRUMME et al., 2004; JENSEN et al., 2006), PhenoSeq (CASHIN et al., 2015), T-CUP 2.0 (HEIDER et al., 2014), AUTO-MUTE (MASSO e VAISMAN, 2010) e Regra de Raymond (RAYMOND et al., 2008).

Alguns destes algoritmos usam o genoma da alça V3, que será referenciado como V3, e outros utilizam a sequência de aminoácidos codificada. A origem dos dados e seu processamento no formato correto, bem como ferramentas utilizadas, serão elucidados no capítulo 3. Porém, nesta seção é explicado como cada algoritmo foi utilizado.

Todos, com exceção do T-CUP e Regra de Raymond, possuem uma interface *web* na qual se pode adicionar as sequências de DNA (Figura 14) ou as sequências de aminoácidos (Figura 15) para, após, obter os resultados para o tropismo.

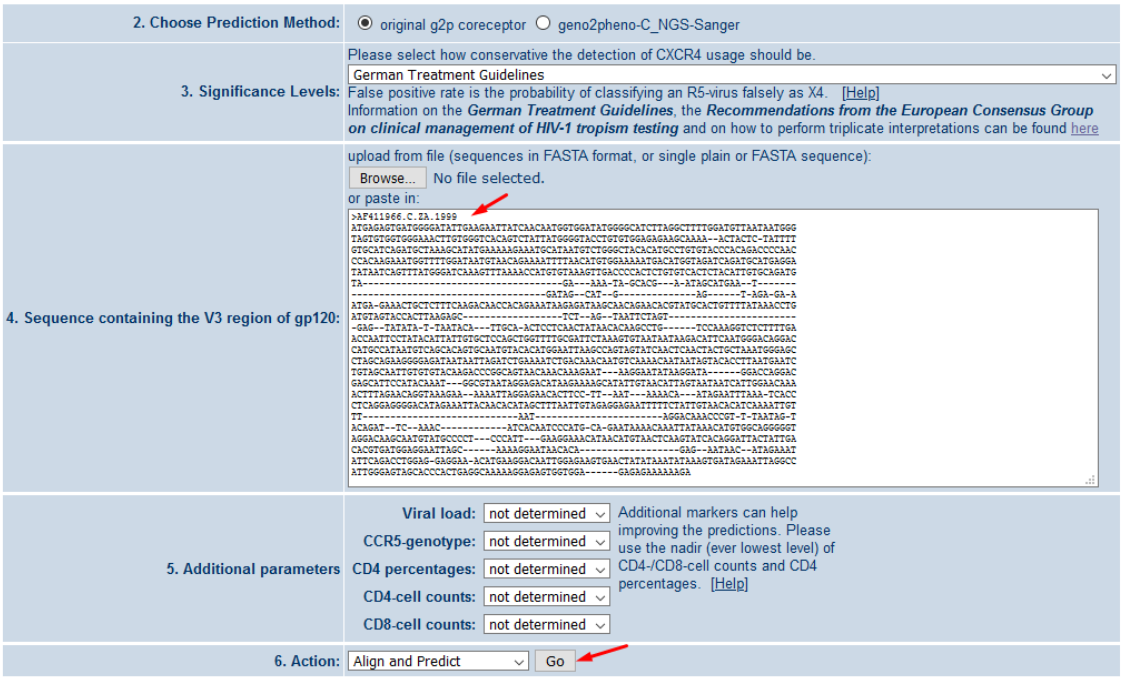

Figura 14. Site do Geno2Pheno com uma sequência de DNA inserida e botão para iniciar análise apontados.

Fonte: Elaboração própria.

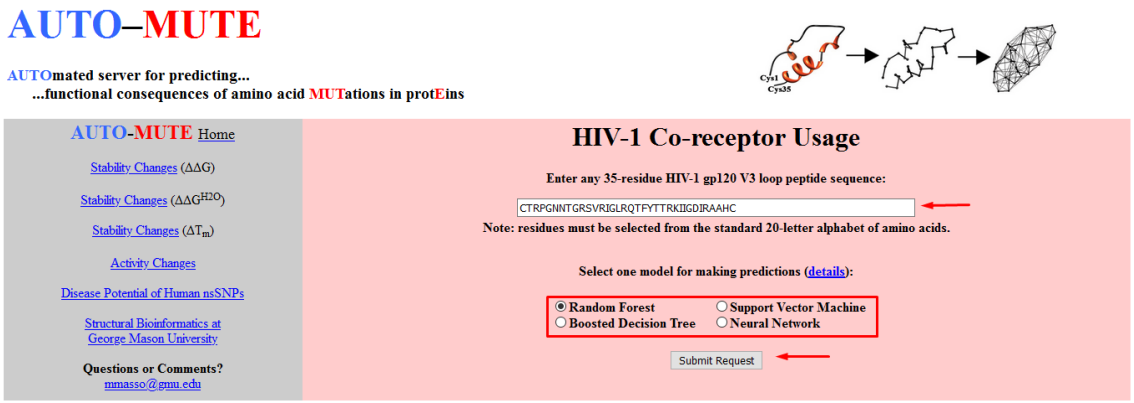

Figura 15. Site do AUTO-MUTE com uma sequência de aminoácidos inserida, opções de técnicas e botão para iniciar análise apontados.

Fonte: Elaboração própria.

No caso do T-CUP, conforme explicado em seguida, ele foi instalado no R e as sequências de DNA foram inseridas conforme exigido na documentação. Já para a regra de Raymond, as sequências de aminoácidos da V3 são utilizadas para obter os

resultados (com a soma das cargas, ou com a verificação da presença de determinados aminoácidos em algumas posições específicas).

A V3 contém 35 aminoácidos, ou seja, 105 (3 * 35) bases nitrogenadas de DNA (cada aminoácido é transcrito por 3 bases, ou também chamado um códon). Abaixo exemplo da mesma sequência como DNA e decodificada para aminoácidos (Figura 16). Existem apenas 20 aminoácidos naturais possíveis, e cada um deles é decodificado em uma letra, para facilitar a leitura.

DNA:	**ATG**	**AGA**	**GTG**	**AGG**	**GAG**	**ATA**	**CTG**	**AGG**	**AAT**
Aminoácidos:	**M**	**R**	**V**	**R**	**E**	**I**	**L**	**R**	**N**

Figura 16. Exemplo de decodificação do DNA em aminoácidos (escala NCBI).

Fonte: Elaboração própria.

A sequência de aminoácidos lida pelas células de um organismo vivo é montada em uma estrutura tridimensional, uma proteína ou polipeptídeo. Para esse estudo, a estrutura de interesse é a V3, uma parte da gp120, representada na Figura 17.

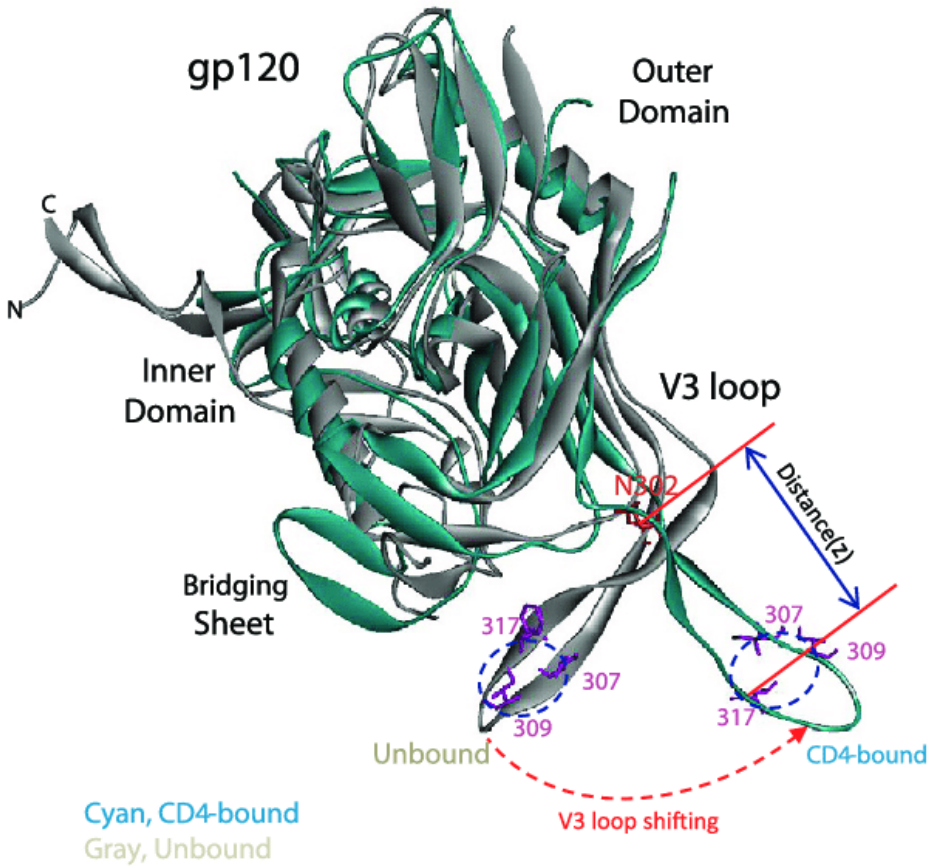

Figura 17. Estrutura da gp120 e, na região inferior, da alça V3 (*V3 loop*).

Fonte: BOWDER et al., 2018.

Exemplo de uma sequência da V3 (com 35 posições): CIRPDNNTRRSIRIGPGQVFYANG**D**IIGDIREASC

Observa-se que na posição 25 da V3, demonstrada em destaque, há o aminoácido representado por D, ácido aspártico ou aspartato (NCBI, 2021). Essa posição é de importância para definir o resultado de tropismo pela regra de Raymond, explicada a seguir junto com os demais algoritmos genotípicos.

Geno2Pheno[1] (G2P): uma das ferramentas mais consolidadas, baseada em SVM (Máquina de vetores de suporte) e disponível *online* desde o final de 2001. Foram submetidas as sequências de DNA com lacunas e as demais opções foram mantidas nos seus valores padrões. Os pontos de corte usados foram de 20% e 10%, ou seja, caso FPR (*false positive rate*), valor dado pelo algoritmo, fosse maior ou igual ao ponto de

[1] GENO2PHENO. Geno2Pheno coreceptor. Disponível em: <https://coreceptor.geno2pheno.org/index.php>. Acesso em: 21 abr. 2021.

corte, a sequência seria R5; caso contrário, seria X4. É denominado ao longo do trabalho G2P 0,10 o teste que utilizou o ponto de corte 0,10 e G2P 0,20 o que utilizou como ponto de corte 0,20.

Web PSSM[2]: disponível desde 2003 e atualizado desde então. Utiliza PSSM (*position-specific scoring matrices*), na qual vírus R5 devem ter uma pontuação menor do que os demais. Foram inseridas as sequências de DNA com intervalos e marcadas as opções de subtipo C. No caso de haver mais de uma sequência de aminoácidos prevista para o mesmo DNA, se uma delas fosse X4, era usada a predição X4. Pode haver mais de uma sequência de aminoácidos prevista pois as sequências de DNA são inseridas com intervalos. Por isso, o algoritmo supõe que pode haver diferentes bases nitrogenadas na posição vazia (A, C, G ou T) e, portanto, diferentes códons e aminoácidos podem ser resultantes.

PhenoSeq[3]: disponível desde 2015, analisa determinadas características da V3, como: tamanho dos aminoácidos, sua carga total, quantidade de locais de glicosilação tipo N, e frequência de alterações em posições específicas. Foi utilizado DNA sem *gaps* (intervalos) e escolhido C como subtipo. Se houvesse mais de uma sequência de aminoácidos para o mesmo DNA, se uma delas fosse X4, era usada a predição X4 de maneira a manter coerência com o que foi feito no Web PSSM.

T-CUP 2.0: disponível desde 2014, usa informações do potencial eletrostático e hidrofobicidade da V3 e floresta aleatória para classificar o tropismo. Instalado no *software* R, versão 3.6.0, descrito no capítulo seguinte (R CORE TEAM, 2019) junto com os pacotes RandomForest 4.5-30 e Interpol 1.3.1 (versões pedidas no README da ferramenta). Foram utilizadas sequências de aminoácidos. O algoritmo fornece a probabilidade de uma sequência ser X4. O ponto de corte escolhido foi de 0,20, ou seja, abaixo de 0,20 foi classificado como X4 e, acima disso, como R5.

AUTO-MUTE[4]: disponível desde 2010, usa 4 abordagens diferentes: floresta aleatória (*random forest* - RF), SVM, árvore de decisão melhorada (*boosted decision*

---

[2] WEB PSSM. Web PSSM. Disponível em: <https://indra.mullins.microbiol.washington.edu/webpssm/>. Acesso em: 21 abr. 2021.
[3] PHENOSEQ. PhenoSeq. Disponível em: <http://tools.burnet.edu.au/phenoseq/>. Acesso em: 21 abr. 2021.
[4] AUTO-MUTE. AUTO-MUTE: HIV-1 Co-receptor Usage. Disponível em: <http://binf.gmu.edu/automute/AUTO-MUTE_HIV-1_Co-receptor_Usage.html>. Acesso em: 21 abr. 2021.

tree - BDT) e redes neurais (*neural network* - NN). Foram utilizadas sequências de aminoácidos.

Regra de Raymond: define o tropismo com base em aminoácidos em determinadas posições e/ou com a carga de algum(ns) aminoácido(s) específico(s). É influenciada por outras regras, também descritas a seguir: Regra 11/25 (DE JONG et al., 1992), Regra 11/24/25 (CARDOZO et al., 2007) e Regra da Carga Total (BRIGGS et al., 2000). A Regra de Raymond foi implementada e testada por agregar as demais.

Nela, o vírus é X4 se sua região V3 atende a pelo menos um dos seguintes critérios: aminoácidos R ou K na posição 11; aminoácido K na 25; aminoácido R na 25 e carga total >= +5; carga total >= +6. Com as sequências de aminoácidos, foi calculada a carga total. As cargas dos aminoácidos individuais foram obtidas por meio do acesso à base de dados Aaindex (KAWASHIMA, OGATA e KANEHISA, 1999; TOMII e KANEHISA, 1996; NAKAI, KIDERA e KANEHISA, 1988), disponível no R e que possui diversas propriedades numéricas de aminoácidos. A característica utilizada foi a KLEP840101, descrita no capítulo 3, seção 3.1 - II (KLEIN, KANEHISA e DELISI, 1984).

A Regra de Raymond é baseada em outras que a precedem, a saber:

- Regra 11/25: a presença de aminoácidos com carga positiva nas posições 11 e/ou 25 do V3 caracteriza um vírus X4.
- Regra 11/24/25: a presença de aminoácidos com carga positiva nas posições 11, 24 e/ou 25 do V3 caracteriza um vírus X4.
- Regra da Carga Total: se a soma das cargas dos aminoácidos da região V3 é >= +5, isto caracteriza um vírus X4.

Para implementação e avaliação das técnicas abordadas neste capítulo são necessárias algumas ferramentas e dados, discorridos sobre no capítulo seguinte.

3 Ferramentas e tecnologias

Este capítulo disserta sobre as principais ferramentas e bases de dados utilizadas neste trabalho na sua primeira seção e, em seguida, aborda a metodologia de pesquisa e aplicação destas ferramentas.

3.1 Ferramentas e bases de dados

I. R *software* e bibliotecas

O R (R CORE TEAM, 2019) é uma linguagem de programação criada em 1993 com a finalidade de fornecer um ambiente completo para análises estatísticas diversas (IHAKA, 1998). Esta contém muitos pacotes, que são conjuntos de funcionalidades extras implementadas por diversos pesquisadores e que podem ser baixados de diretórios públicos ou privados, como por exemplo o CRAN (*Comprehensive R Archive Network*), diretório público (CRAN, 2021). O R é um *software* gratuito e de código aberto (R CORE TEAM, 2019), sendo esses os diferenciais para sua escolha. Além disso, possui pacotes com as funcionalidades necessárias a essa pesquisa implementadas, como os algoritmos de ML.

Neste trabalho foram utilizados os seguintes pacotes do R: SeqinR (CHARIF e LOBRY, 2007), Biostrings (PAGÈS et al., 2019), Stringr (WICKHAM, 2019), Caret (KUHN et al., 2019), Ggplot2 (WICKHAM, 2016), DMwR (TORGO, 2010), Mltools (GORMAN, 2018).

- O SeqinR e o Biostrings são utilizados para manipular sequências biológicas como as de DNA e aminoácidos;
- o Stringr é um pacote de manipulação de textos. Como as sequências biológicas são textuais, ele foi utilizado para auxiliar no pré-processamento;
- o Caret é utilizado para aprendizagem de máquina, fazendo a ligação com outros pacotes de modelos específicos, além de conter funções para calcular o desempenho, fazer treinamento e otimização de modelos, e para pré-processamento de dados;

- o Ggplot2 é um pacote para confecção de gráficos que foi utilizado para criar um gráfico com os Kappas entre testes, representando assim os resultados;
- o Mltools é um pacote que contém métodos para auxiliar no processamento de ML, e foi utilizado para o cálculo do MCC.

II. Bases de dados

AAindex (KAWASHIMA, OGATA e KANEHISA, 1999; TOMII e KANEHISA, 1996; NAKAI, KIDERA e KANEHISA, 1988), mencionada na seção 2.4, é uma base de dados que contém um compilado de 544 medidas dos 20 aminoácidos naturais, retiradas de artigos dos mais diversos temas. Esta base é disponibilizada no R por diversos pacotes, dentre os quais o SeqinR, que foi o utilizado. Por exemplo, a medida BIGC670101 indica o volume do aminoácido. Já a métrica FASG890101 faz relação com a hidrofobicidade do peptídeo.

Neste trabalho, ela foi utilizada para somar as cargas dos aminoácidos (KLEP840101) da cadeia V3 para cálculo da Regra de Raymond. Porém, sua ampla gama de informações fornece recursos para uso futuro nesta e em outras pesquisas que lidem com proteínas.

A característica KLEP840101 era a única presente no AAindex que se referia a carga total do aminoácido, sendo portanto a mais apropriada para calcular a Regra de Raymond, de acordo com descrição do artigo original (RAYMOND et al., 2008). Outras relacionadas incluem FAUJ880111 (carga positiva) e FAUJ880112 (carga negativa).

As sequências da região V3 da gp120 foram extraídas da base de dados de Los Alamos. O laboratório nacional de Los Alamos (LOS ALAMOS, 2021; KUIKEN et al., 2003) é um centro de pesquisa estadunidense que, dentre outros resultados e ferramentas, fornece sequências genéticas de HIV compiladas e com grande riqueza de metadados.

No *website* do laboratório de Los Alamos é possível exportar os dados de HIV em vários formatos, dentre os quais está o FASTA, que é um tipo de arquivo comumente utilizado para sequências de DNA ou de aminoácidos. O FASTA contém o identificador da sequência precedido por um “>” e, na linha seguinte, a sequência de interesse, com limite de no máximo 80 caracteres até a quebra de linha (Figura 18). Alguns algoritmos

genotípicos aceitam sequências apenas neste formato.

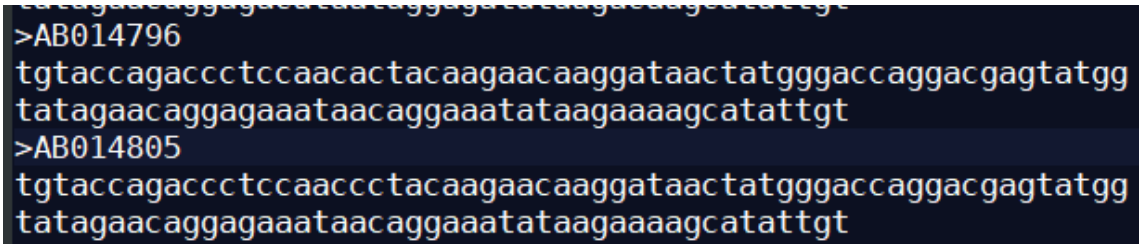

Figura 18. Arquivo FASTA com duas sequências de DNA (AB014796 e AB014805).
Fonte: LOS ALAMOS, 2021.

III. Github e Zenodo

O GitHub é um repositório para o compartilhamento de recursos, especialmente códigos, com as mais diversas licenças e funcionalidades. Baseado no Git, foi criado em 2008 (TECHCRUNCH, 2012) e adquirido pela Microsoft em 2018 (LARDINOIS e LUNDEN, 2018). Criado em 2005, o Git revolucionou o versionamento de código e compartilhamento e modificação de *software* livre (GIT, 2021a, GIT, 2021b).

O GitHub foi utilizado para compartilhar a base de dados montada e outros resultados a partir de uma outra ferramenta para compartilhamento de dados abertos, denominada Zenodo, criada em Maio de 2013 por meio de investimentos da União Europeia (ZENODO, 2021). Foi criada uma tag no GitHub (versão específica do diretório de códigos) e, para o Zenodo, foi gerado um DOI (*Digital Object Identifier*) (MENEZES e RAPOSO, 2020), necessário para citações e *links* em pesquisas.

IV. Python e mineração de dados

A linguagem Python (PYTHON, 2021) foi utilizada para automatizar a entrada de informações em algumas das ferramentas genotípicas que exigiam a inserção das sequências uma por uma para classificação do tropismo.

Para facilitar esse processo, utilizou-se um navegador automatizado a partir da biblioteca Selenium (SELENIUM, 2021). Bibliotecas no Python são semelhantes a pacotes no R, porém, como Python é uma linguagem de programação de uso mais

amplo, as bibliotecas têm muitas finalidades além de Estatística.

O Python possui o PyPI (PYPI, 2021), similar ao CRAN do R, no qual estão indexadas a maior parte das bibliotecas públicas para a linguagem.

Na próxima seção é discorrido sobre como essas ferramentas foram utilizadas.

3.2 Metodologia de pesquisa

I. Pré-processamento dos dados

No desenvolvimento do projeto, os dados dos vírus foram obtidos por meio da base de Los Alamos. Foram filtradas sequências de DNA de HIV-1 do subtipo C relativas à região V3 da gp120 viral em formato FASTA. Essas sequências foram então carregadas para o *software* R, versão 3.6.0, e preparadas com as bibliotecas: SeqinR, Biostrings e Stringr.

Em cada sequência, intervalos (-) foram removidos e o DNA foi convertido para aminoácidos (aa) usando a escala padrão NCBI (NCBI, 2021), configurada pelas bibliotecas do R. Códons (3 caracteres de DNA) que não correspondiam a nenhum aminoácido a princípio foram substituídos pelo caractere 'X'. Em seguida, as sequências foram cortadas de modo a começar e terminar, necessariamente, com o aminoácido cisteína (C), pois conforme descrito por Chiou et al. (1992), isto é o que caracteriza a região V3.

Além da cisteína na primeira e última posição da alça V3, foram admitidos na terceira posição os aminoácidos arginina (R), glicina (G) ou X e, na 33ª posição: alanina (A), treonina (T), prolina (P), serina (S) ou X; embora classicamente apenas R e A estejam presentes na 3ª e na 33ª, respectivamente (HEIDER et al., 2014; HUNG et al., 1999). Esses passos foram realizados para que 7 sequências fora do padrão não fossem descartadas.

Por fim, as cadeias de aminoácidos foram alinhadas com o consenso ancestral para a região V3 do subtipo C, obtido também de Los Alamos ("CTRPNNNTRKSIRIGPGQTFYATGDIIGDIRQAHC"), por meio do algoritmo Needleman-Wunsch (NEEDLEMAN e WUNSCH, 1970) com parâmetros padrões. O

alinhamento é realizado para que os aminoácidos fiquem nas posições corretas, mesmo com elementos faltantes.

Foram descartadas todas as cadeias que não continham exatamente 35 aminoácidos após alinhamento, ou que, antes do alinhamento, não possuíam os aminoácidos necessários no início e término ou menos de 28 aminoácidos no total (perda de 20% do total de 35 aminoácidos da V3, considerada muito significativa para que a sequência ainda fosse utilizada).

A sequência de passos até aqui foi feita com o código exemplificado no Apêndice 01. Na base de dados montada com esse processo, as sequências de aminoácidos duplicadas foram eliminadas, sendo mantida apenas a primeira ocorrência, mesmo que o DNA fosse diferente entre elas, para evitar super-representação, pois isso poderia enviesar os algoritmos de ML. Nas 562 sequências obtidas foram realizados os testes genotípicos conforme descritos no capítulo 2, seção 2.4, que estão entre os mais citados na literatura. Todos utilizam a região V3 para prever o tropismo e estavam disponíveis para teste e/ou implementação, sendo este um dos critérios de escolha.

No T-CUP 2.0, no AUTO-MUTE e na Regra de Raymond, se houvesse ‘X’ em qualquer posição do aminoácido, este era substituído pelo valor do consenso na mesma posição. Isso gerou mais algumas sequências de aminoácidos duplicadas, porém estas não foram descartadas, mas avaliadas como únicas.

Os vírus que estavam como R5X4 ou X4 no desfecho foram recategorizados para NR5 (não R5), uma vez que os testes genotípicos realizados possuíam apenas as categorias X4 e R5 e, nestes, X4 também foi recategorizado para NR5. Assim, um total de 55 sequências NR5 e 507 R5 foram estudadas.

Os passos metodológicos até aqui estão explicitados na Figura 19.

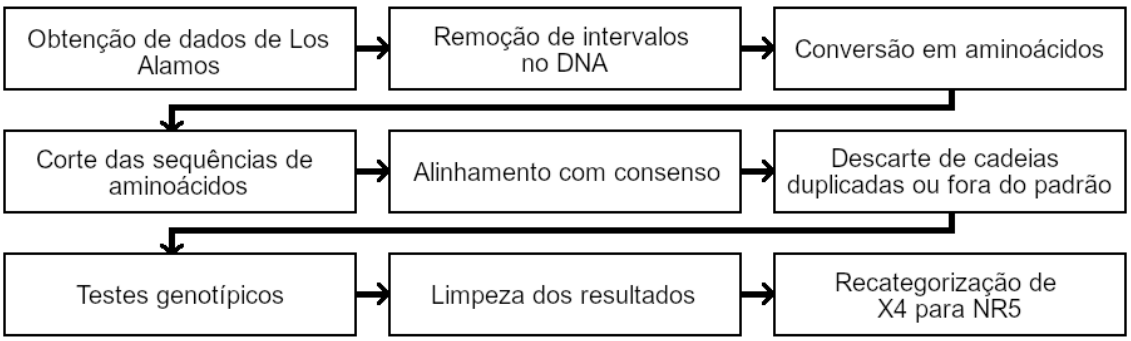

Figura 19. Passos metodológicos do pré-processamento de dados

Fonte: Elaboração própria.

II. Voto majoritário e *stacking*

O pacote Caret do R foi utilizado para gerar matrizes de confusão entre os testes e entre eles e o desfecho. Esse pacote também permitiu calcular o índice Kappa entre os testes e forneceu resultados para algumas das medidas descritas no capítulo 2, seção 2.3: sensibilidade, especificidade e acurácia. O MCC foi calculado com o pacote Mltools. A Figura 21 foi criada com pacote Ggplot2 a partir dos índices Kappa calculados sobre as matrizes de confusão dos testes.

Os resultados dos testes foram combinados por voto majoritário para verificar a melhora do desempenho. Foram feitas duas abordagens: combinar os 10 testes (VM10.R5 e VM10.NR5) ou combinar os 3 testes que apresentaram melhor sensibilidade (VM3), já que um dos objetivos era aumentar esse parâmetro.

O voto majoritário de 10 testes genotípicos foi realizado da seguinte maneira: em VM10.R5 foi favorecida a decisão R5 em caso de empate e, em VM10.NR5, a NR5. Este passo foi necessário uma vez que a quantidade de testes era par e situações de empate poderiam ocorrer.

Foram também criados metaclassificadores com *stacking* dos algoritmos disponíveis (código do R exibido no Apêndice 02). Para o *stacking*, primeiro os dados foram separados em dois conjuntos mutuamente exclusivos de 70% para treino e 30% para teste, retendo a proporção das classes originais (cerca de 10% de NR5).

Devido ao grande desbalanceamento dos dados, no conjunto de treino foi aplicada a estratégia de balanceamento SMOTE, o que permitiu ter cerca de 57% e 43% de R5 e NR5, respectivamente.

Após o balanceamento, métodos de treinamento disponíveis no pacote Caret para problemas de classificação foram aplicados e verificados nos dados de teste. Alguns métodos de treinamento, com os melhores desempenhos, que eram apropriados para as variáveis do problema em questão foram escolhidos, são eles (entre parêntesis o método do Caret utilizado): SIMCA (CSimca), *Rotation Forest* (rotationforest), CART (rpart2), KNN (knn) e *Multi-Layer Perceptron* (mlpWeightDecay).

O método SIMCA (*Soft Independent Modelling of Class Analogies*), explicado sobre no capítulo 2, seção 2.1 - VIII, é especialmente útil quando há alta dimensionalidade de atributos, uma vez que realiza a PCA (análise de componentes principais) várias vezes ao longo do seu processo. Novas observações são classificadas de acordo com as médias dos seus desvios/distâncias dos PCAs gerados (VANDEN e HUBERT, 2005).

Rotation Forest, também explorado no capítulo 2, seção 2.1 - VIII, é baseado em árvore de decisão e apropriado para *stacking*. Ele reduz os atributos recebidos com PCA e usa árvores de decisão sobre os atributos reduzidos (RODRÍGUEZ et al., 2006; BAGNALL et al., 2019).

CART (*Classification and Regression Tree*) é um algoritmo guloso que usa árvore de decisão binária simples. Nele, em cada nó é feita uma divisão dos dados de modo a otimizar algum critério. O parâmetro de parada do algoritmo disponibilizado pelo Caret é a profundidade máxima da árvore (KUHN et al., 2020).

Multi-Layer Perceptron, conforme mencionado no capítulo 2, seção 2.1 - VI, utiliza uma rede neural completamente conectada e *feedforward*. O treinamento é feito de modo que, cada vez que as observações passam pela rede, os pesos dos atributos são multiplicados por um número entre 0 e 1 (*decay*) e, com isso, reduzidos para minimizar o *overfitting* ao conjunto de treino (BAGNALL et al., 2019; KUHN et al., 2020; BERGMEIR e BENÍTEZ, 2012).

O KNN, conforme abordado no capítulo 2, seção 2.1 - II, é um algoritmo de *Machine Learning* baseado em distância, no qual as observações são classificadas com base no comportamento dos *k* vizinhos mais próximos (SRIVASTAVA, 2020).

O pacote Caret faz *bootstrap* 25 vezes por padrão em cada algoritmo ao treinar.

Nas 25 amostras, os hiperparâmetros dos algoritmos são modificados e, ao final, são selecionados os que obtiveram maior acurácia. Embora alguns modelos pudessem ser ajustados com base na sensibilidade, especificidade etc., optou-se por manter a acurácia como parâmetro de *tuning*, uma vez que os hiperparâmetros obtidos foram idênticos aos resultantes treinando com base na sensibilidade e, com essa ação, todos os algoritmos tiveram a mesma lógica no *tuning*.

Os hiperparâmetros encontrados foram:

- SIMCA: O Caret não faz *tuning* desse algoritmo. O número de componentes principais da PCA é determinado pelo pacote original com uma fórmula.
- *Rotation Forest*: sendo K o número de subconjuntos de atributos sobre o qual aplica PCA e L a quantidade de algoritmos genotípicos usados no *stacking*, foi utilizado K = 1 e L = 9.
- CART: a profundidade máxima da árvore = 3.
- *Multi-Layer Perceptron*: tamanho (quantidade de neurônios na camada oculta) = 5, *decay* = 0.
- *k-Nearest Neighbors*: k = 5.

Dada a natureza aleatória de alguns dos passos descritos (separação em treino e teste, SMOTE e técnica de *stacking*), para minimizar o viés do resultado final e obter reprodutibilidade, foram escolhidos 20 números inteiros aleatórios (sementes) e todos os passos do *stacking* foram repetidos para cada um deles. Na seção seguinte são apresentadas médias para acurácia, especificidade, sensibilidade, MCC e Kappa. As sementes foram: 9992, 9355, 6622, 9592, 9108, 3671, 4031, 3796, 7454, 7358, 7833, 859, 5279, 3134, 5186, 6957, 3813, 9045, 5714, 2711.

Um resumo dos passos executados nessa segunda etapa da metodologia está exposto na Figura 20.

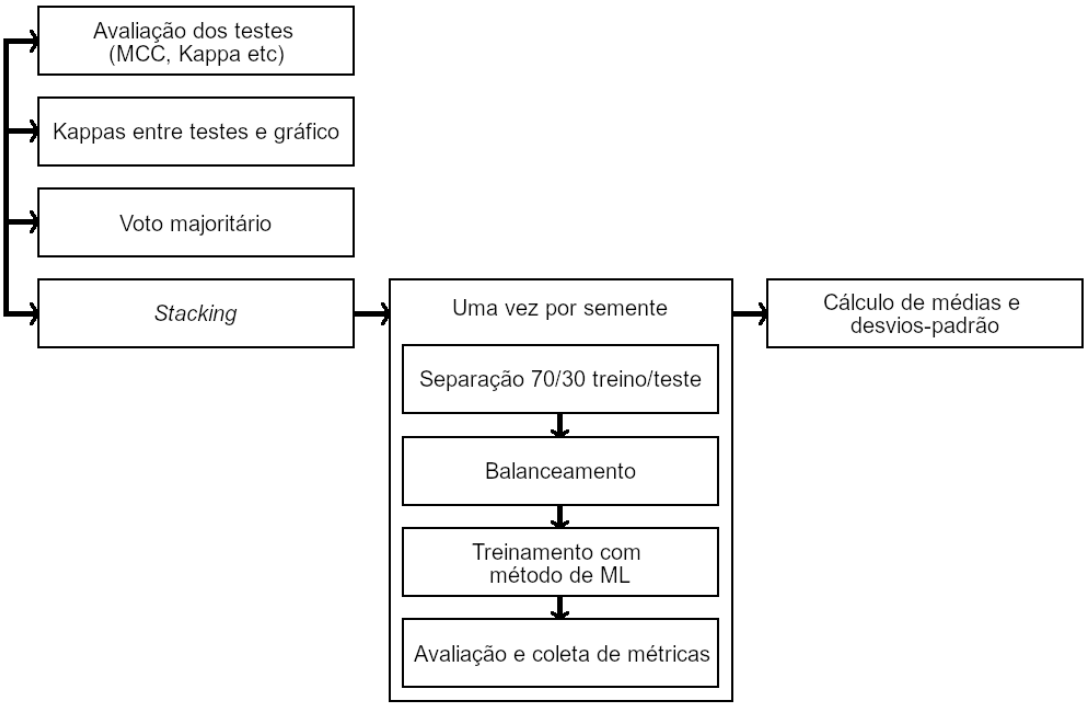

Figura 20. Passos metodológicos do voto majoritário e *stacking*.

Fonte: Elaboração própria.

4 Resultados e discussão

Este capítulo apresenta o resultado de desempenho dos algoritmos genotípicos testados, a comparação entre eles e o resultado da implementação de técnicas de ML, bem como a discussão sobre esses resultados.

Na Tabela 3, abaixo, está o primeiro resultado, o desempenho dos algoritmos genotípicos escolhidos da literatura para a base de dados completa obtida de Los Alamos.

Tabela 3. Desempenho dos algoritmos genotípicos em relação ao desfecho.

Ac: acurácia, Esp: especificidade, Sen: sensibilidade, MCC: coeficiente de correlação de Matthews. Maiores valores por coluna em destaque.

Fonte: Elaboração própria.

Teste	Ac	Esp	Sen	MCC	Kappa
Web PSSM	0,8363	0,8363	0,8364	0,4753	0,4205
PhenoSeq	0,8541	0,8560	0,8364	0,5036	0,4559
T-CUP 2.0	**0,9555**	0,9783	0,7455	**0,7422**	**0,7418**
G2P 0,20	0,8399	0,8343	**0,8909**	0,5070	0,4443
G2P 0,10	0,9288	0,9389	0,8364	0,6699	0,6579
AUTO-MUTE BDT	0,9288	**0,9901**	0,3636	0,5099	0,4674
AUTO-MUTE RF	0,9270	**0,9901**	0,3455	0,4932	0,4482
AUTO-MUTE NN	0,8861	0,9448	0,3455	0,3115	0,3104
AUTO-MUTE SVM	0,9181	0,9744	0,4000	0,4603	0,4468
Regra de Raymond	0,9520	0,9921	0,5818	0,6965	0,6784

Os métodos que apresentaram melhor desempenho, no geral, foram T-CUP e a Regra de Raymond, com maior Kappa (0,7418 e 0,6784, respectivamente), MCC (0,7422 e 0,6965) e acurácia (95,55% e 95,20%), o que é similar aos resultados obtidos

por Riemenschneider et al (2016). Como todas as medidas da tabela são numéricas, a comparação entre elas é feita quantitativamente.

Os algoritmos que usam floresta aleatória são T-CUP e AUTO-MUTE RF. O AUTO-MUTE RF teve razoável MCC e boa acurácia e especificidade (0,4932, 92,70% e 99,01%, respectivamente), embora sua sensibilidade seja menos do que a metade da resultante do T-CUP (34,55% *versus* 74,55%), o que leva a crer que uma abordagem de floresta aleatória pode ser aperfeiçoada para este problema, já que abordagens similares tiveram valores de sensibilidade tão distantes.

Quanto à regra de Raymond, que usa carga e verifica aminoácidos em posições específicas, o teste mais similar é o PhenoSeq, que por sua vez tem uma sensibilidade maior (58,18% *versus* 83,64%), o que indica que parâmetros como glicosilação ou outros usados apenas pelo PhenoSeq podem ser úteis para melhorar a sensibilidade.

Embora todos os valores de MCC tenham sido superiores aos de Kappa, estas medidas foram próximas em todos os testes, indicando que são razoavelmente equivalentes e possivelmente correlacionadas (ver colunas de MCC e Kappa na Tabela 3).

Nenhum teste apresentou Kappa forte (de 0,80 a 0,90) com o desfecho e apenas 3 demonstraram correlação moderada (de 0,60 a 0,79): T-CUP, G2P 0,20 e Regra de Raymond.

Todos os testes apresentaram acurácia razoável, acima de 80%, ainda que ela não seja uma medida ideal uma vez que os dados estão desbalanceados.

Na especificidade, AUTO-MUTE BDT e AUTO-MUTE RF tiveram um dos maiores valores (99,01% para ambos), indicando que abordagens com árvore de decisão podem ser úteis na detecção de vírus R5, apesar de esse não ser o problema de maior relevância (que é detectar vírus NR5, aumentando a sensibilidade).

À exceção do G2P, os testes com maior sensibilidade foram os específicos para o subtipo C. No entanto, nenhum deles conseguiu atingir uma sensibilidade maior que 90% nesta amostra. Uma alta sensibilidade, ou seja, capacidade de detectar verdadeiros positivos (NR5), é de grande importância para esses testes no contexto clínico, pois a medicação incorreta de um paciente que apresenta vírus com tropismo X4 pode piorar o seu quadro clínico e acelerar/desencadear a evolução da aids.

G2P 0,20 obteve maior sensibilidade do que G2P 0,10 (89,09% *versus* 83,64%), embora todos os outros parâmetros para ele tenham sido piores (valores cerca de 10% a 20% inferiores). No contexto clínico, portanto, pode ser melhor utilizar ponto de corte 0,20. Assim como nos resultados obtidos por Riemenschneider et al (2016) e Gupta et al (2015), G2P apresentou um dos melhores desempenhos para este subtipo.

Também se buscou verificar a concordância entre os algoritmos genotípicos por meio do índice Kappa. Os resultados estão demonstrados na Figura 21, abaixo.

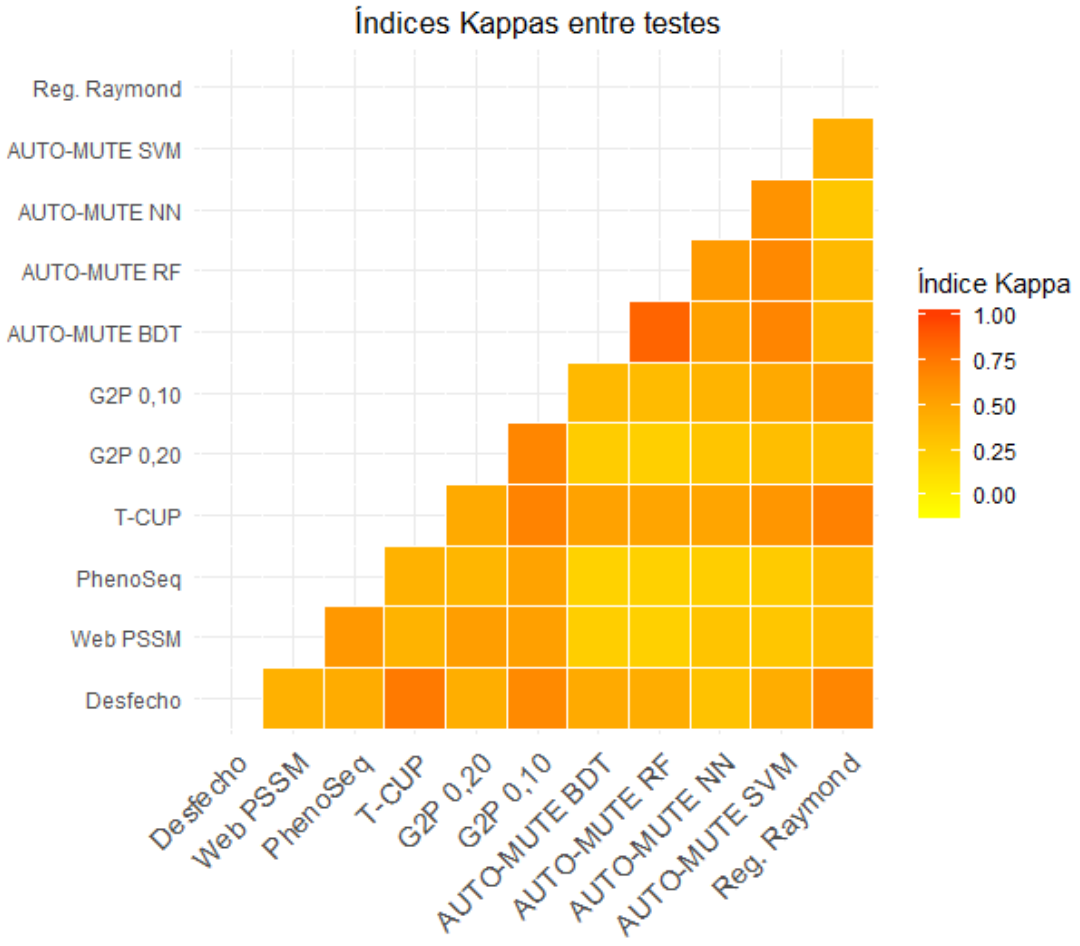

Figura 21. Kappa de Cohen entre os algoritmos genotípicos.

Fonte: Elaboração própria com o pacote Ggplot2.

As 4 metodologias diferentes do AUTO-MUTE tiveram Kappas de 0,5282 a 0,8506 entre si (este último, entre AUTO-MUTE BDT e AUTO-MUTE RF, um Kappa

considerado forte). Isto é em parte esperado, pois as 4 foram desenvolvidas pelos mesmos autores. Porém, com outros testes, os Kappas foram de 0,5164 (entre AUTO-MUTE BDT e T-CUP, correlação fraca) ou menos (correlação fraca, mínima ou nenhuma).

As duas abordagens específicas para o subtipo C, Web PSSM e PhenoSeq, apresentaram Kappa de 0,5759, que apesar de ser um valor considerado fraco, foi maior do que entre eles e o desfecho e um dos mais altos obtidos, mesmo esses testes tendo metodologias diferentes.

Outros Kappas de valores moderados (entre 0,60 e 0,79) foram: Entre T-CUP e Regra de Raymond (0,7049), G2P 0,10 e G2P 0,20 (0,6773), T-CUP e G2P 0,10 (0,695), o que é esperado pois são os testes de melhor desempenho (e, no caso do G2P, apenas alterou-se o ponto de corte).

Conforme explorado por Soulié et al (2016), não só o desempenho dos algoritmos genotípicos varia de acordo com o subtipo, mas também varia com a amostra, sugerindo que existem componentes genéticos pouco explorados na determinação do tropismo, que mudam de acordo com a população estudada.

A Tabela 4 apresenta os resultados dos votos majoritários obtidos a partir da combinação dos algoritmos genotípicos.

Tabela 4. Desempenho dos votos majoritários obtidos a partir da combinação dos algoritmos genotípicos.

Ac: acurácia, Esp: especificidade, Sen: sensibilidade, MCC: coeficiente de correlação de Matthews. Maiores valores por coluna em destaque.

Fonte: Elaboração própria.

Sigla	Ac	Esp	Sen	MCC	Kappa
VM10.R5	**0,9591**	**0,9882**	0,6909	**0,7511**	**0,7455**
VM10.NR5	0,9484	0,9724	0,7273	0,7054	0,7054
VM3	0,8737	0,8738	**0,8727**	0,5553	0,5106

O voto majoritário VM3 foi feito com os testes Web PSSM, PhenoSeq e G2P 0,20, os três de maior sensibilidade.

Nenhum voto majoritário foi efetivo para aumentar a sensibilidade.

VM10.R5 obteve pequena melhora sobre o T-CUP na maioria dos parâmetros, exceto na sensibilidade (69,09% *versus* 74,55%). Sendo assim, o uso de voto majoritário é contraindicado para este problema, uma vez que é uma técnica mais trabalhosa e que apresentou pouca ou nenhuma melhora relevante.

Um motivo pelo qual o voto majoritário pode ter apresentado pouca melhora é a baixa variabilidade nas técnicas de testes genotípicos. Embora sejam 10 testes no total, dois são baseados em SVM (Geno2Pheno e AUTO-MUTE SVM), três usam alguma abordagem baseada em árvore de decisão (T-CUP, AUTO-MUTE RF e AUTO-MUTE BDT), e três usam a carga/potencial eletrostático da cadeia V3 como parte do resultado (PhenoSeq, T-CUP e Regra de Raymond). Além disso, as quatro abordagens do AUTO-MUTE foram elaboradas pelos mesmos autores e, conforme apontado pelo Kappa, são correlacionadas. Assim sendo, é possível que os testes considerem as mesmas características em seus votos e por isso não levam a boas decisões quando considerados em conjunto.

Para o *stacking* (Tabela 5), as técnicas foram escolhidas com base nos seguintes critérios:

- A de maior sensibilidade (SIMCA);
- A de maior especificidade (KNN); e
- As 3 melhores que, em comparação com o G2P 0,20 ou G2P 0,10, mais aumentaram a sensibilidade ou especificidade. São elas: *Rotation Forest*, CART, *Multi-Layer Perceptron*.

Alguns métodos similares no Caret (por exemplo, “rpart2” e “rpart1SE”) ou que tiveram desempenhos próximos/empates foram eliminados e priorizou-se modelos de maior média de sensibilidade.

Tabela 5. Desempenho médio e desvio-padrão dos melhores métodos de *stacking* e dos melhores métodos genotípicos nos conjuntos de teste.

Ac: acurácia, Esp: especificidade, Sen: sensibilidade, MCC: coeficiente de correlação de Matthews, DP: desvio-padrão. Maiores valores por coluna em destaque.

Fonte: Elaboração própria.

Teste	Ac (DP)	Esp (DP)	Sen (DP)	MCC (DP)	Kappa (DP)
SIMCA	0,7432 (0,0626)	0,7224 (0,0724)	**0,9406** (0,0554)	0,4174 (0,0525)	0,3172 (0,0685)
KNN	0,9393 (0,0176)	0,9487 (0,0189)	0,8500 (0,0846)	0,7074 (0,0711)	0,6963 (0,0733)
Rotation Forest	0,9185 (0,0219)	0,9214 (0,0203)	0,8906 (0,0833)	0,6602 (0,0835)	0,6348 (0,0861)
CART	0,9190 (0,0282)	0,9224 (0,0315)	0,8875 (0,0826)	0,6649 (0,0831)	0,6397 (0,0922)
Multi-Layer Perceptron	0,9277 (0,0201)	0,9326 (0,0223)	0,8813 (0,0809)	0,6830 (0,0728)	0,6631 (0,0775)
T-CUP	**0,9545** (0,0113)	0,9753 (0,0080)	0,7563 (0,0905)	**0,7349** (0,0685)	**0,7334** (0,0686)
G2P 0,20	0,8360 (0,0228)	0,8303 (0,0230)	0,8906 (0,0857)	0,4979 (0,0643)	0,4324 (0,0606)
G2P 0,10	0,9265 (0,0189)	0,9349 (0,0186)	0,8469 (0,0823)	0,6655 (0,0776)	0,6493 (0,0796)
Regra de Raymond	0,9542 (0,0121)	**0,9924** (0,0058)	0,5906 (0,1043)	0,7023 (0,0899)	0,6831 (0,0970)

Para que a comparação fosse realizada de maneira justa, o desempenho dos melhores métodos genotípicos também está presente na Tabela 5 para os mesmos conjuntos de teste, ou seja, como uma média das 20 sementes executadas. Todos os valores de desempenho estão próximos daqueles obtidos com a amostra completa (desvio-padrão menor que 10% para a grande maioria) mesmo para os testes não apresentados (por exemplo, PhenoSeq), o que indica que, em média, os conjuntos de teste foram bem representativos.

SIMCA é um método interessante para confirmar a classificação de tropismo, visto que a sua sensibilidade é alta (94,06%), embora sua especificidade tenha sido a menor em comparação com os demais (72,24%). Suas sensibilidade e especificidade foram praticamente as do T-CUP invertidas, ou seja, obteve em especificidade o que o T-CUP obteve em sensibilidade, e vice-versa. Nota-se, no entanto, que os valores de MCC e Kappa foram bem menores que os do T-CUP (0,4174 *versus* 0,7349 no MCC e 0,3172 *versus* 0,7334 no Kappa). Embora a acurácia tenha sido menor que a do T-CUP (74,32% *versus* 95,45%), ela pode estar enviesada pelos dados desbalanceados, e isso

indica que Kappa e MCC são boas métricas de resumo, mas é necessário olhar o desempenho do teste como um todo para que sejam feitas boas escolhas.

Em relação ao desempenho geral, o melhor foi o KNN, com MCC, sensibilidade e especificidade maiores que os do G2P 0,10 (70,74%, 85,00% e 94,87% *versus* 66,55%, 84,69% e 93,49%). Mesmo em comparação com T-CUP, o melhor teste genotípico geral, seu uso é mais indicado, pois apresentou ganho de cerca de 10% em sensibilidade (85,00% *versus* 75,63%) e a especificidade decresceu em apenas cerca de 3% (94,87% *versus* 97,53%).

O *Rotation Forest*, o CART e o *Multi-Layer Perceptron* tiveram desempenhos similares e todos foram um pouco melhores do que o G2P 0,20, aumentando a especificidade em cerca de 10% (em relação aos 83,03% do G2P 0,20) e ainda mantendo a sensibilidade no mesmo patamar, de cerca de 89%.

O método CART é de fácil entendimento e visualização, e portanto decidiu-se demonstrar uma de suas árvores de decisão resultantes. Como cada uma das 20 iterações gerava um árvore diferente, tentou-se escolher a iteração que representasse bem a média das árvores obtidas. A grande maioria das árvores (17) começava com o T-CUP (10) ou G2P 0,10 (7). Além disso, muitas vezes PhenoSeq (6) ou Regra de Raymond (4) apareciam num terceiro ou quarto nível da árvore. Por isso, reproduz-se a árvore da semente 9992, que continha pelo menos três desses quatro testes mais frequentes, e estava na ordem mais comum, com o T-CUP na raiz (Figura 22).

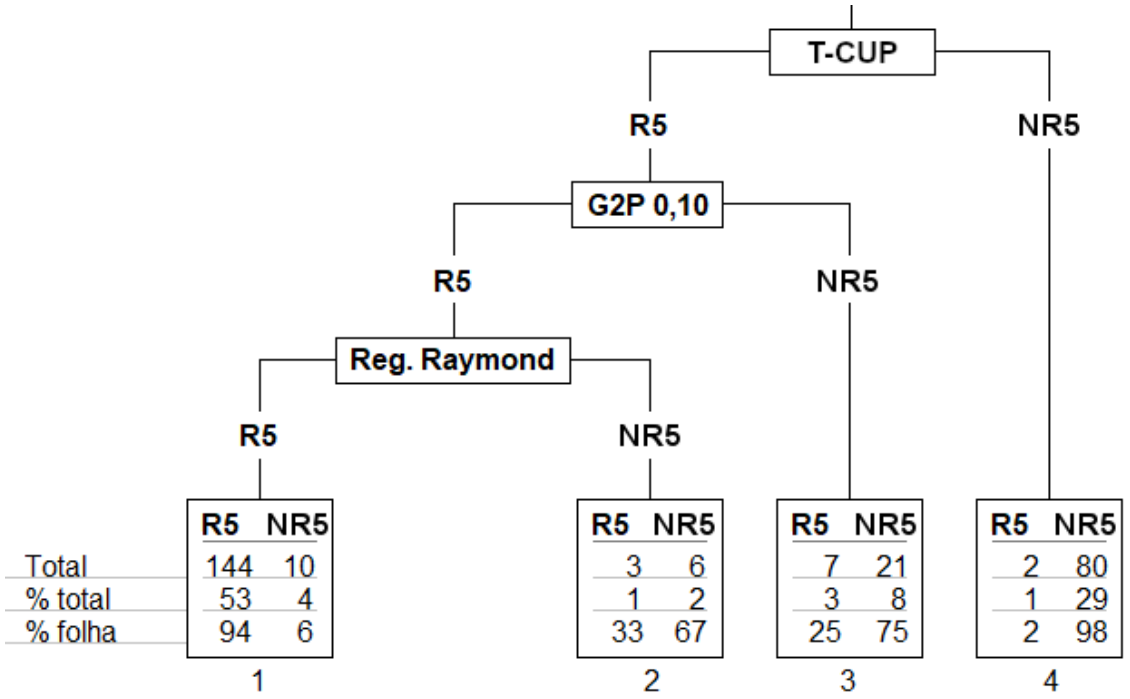

Figura 22. Gráfico da árvore obtida pelo algoritmo CART na semente 9992.

Fonte: Elaboração própria a partir de resultados do R.

Os dados apresentados na Figura 22 são referentes ao conjunto de treino. Assim sendo, há mais observações NR5 do que na amostra original em função do SMOTE. Na folha 1, por exemplo, na qual todas as observações foram classificadas como R5, 144 estavam corretas e 10 incorretas (94% e 6% naquela folha, respectivamente). Isto equivale a 53% dos R5 da amostra total e apenas 4% de NR5 (essa amostra continha 57% R5 e 43% NR5).

Cabe destacar que, devido ao desbalanceamento dos dados, ao utilizar a técnica de SMOTE, esta pode ter gerado observações indesejáveis quando, ao fazer a interpolação entre duas instâncias, uma delas é *outlier*. Isto pode ter influenciado os resultados do *stacking* e é uma das limitações metodológicas previstas.

Além disso, no presente trabalho, não foi aplicada nenhuma etapa de ajuste dos hiperparâmetros dos metaclassificadores, deixando que estes fossem ajustados pelo pacote Caret de modo padrão, a fim de que pudessem ser avaliados em grande quantidade.

Como ferramenta auxiliar, os algoritmos treinados (semente 9992) foram disponibilizados junto com guia de uso (MENEZES e RAPOSO, 2020b).

5 Conclusão

5.1 Considerações finais

O uso de testes genotípicos possui relevância clínica para a detecção da eficácia de um possível tratamento com inibidores de ligação ao CCR5.

Como pôde ser visto neste panorama dos algoritmos genotípicos mais comumente utilizados, seus desempenhos variam muito e podem não ser ideais para um subtipo não B.

É necessário desenvolver testes genotípicos com maior sensibilidade para subtipos presentes em países subdesenvolvidos, a exemplo do PhenoSeq e Web PSSM, focados no subtipo C. No entanto, o G2P 0,20 apresentou melhor sensibilidade para este subtipo.

As estratégias de voto majoritário não obtiveram desempenho relevante para melhorar as medidas estudadas.

O *Stacking* foi capaz de melhorar as medidas de desempenho. Os algoritmos *Rotation Forest*, CART e *Multi-Layer Perceptron* aumentaram a especificidade em cerca de 10% e ainda mantiveram a sensibilidade no mesmo patamar que o G2P 0,20, de cerca de 89%. KNN também se mostrou melhor do que G2P 0,10 e pôde aumentar a sensibilidade com relação ao T-CUP em cerca de 10% (85%), com perda de apenas 3% em especificidade (94%). SIMCA atingiu sensibilidade de 94% e especificidade de 72%, sendo um método interessante para o contexto clínico a fim de confirmar o tropismo.

A sequência de testes T-CUP, G2P 0,10 e regra de Raymond esteve presente em muitas das árvores de decisão obtidas no *stacking* com CART, não só sendo fácil de visualizar como obtendo bons resultados. Devido a facilidade de uso, recomenda-se, portanto, esses três testes para melhorar a previsão do subtipo C com as regras de decisão expostas na Figura 22 do capítulo 4.

5.2 Limitações do projeto

O projeto está limitado pela quantidade de sequências obtidas para o subtipo C, cerca de 550. Com a dependência de bases de dados públicas, como a de Los Alamos, não é possível resolver os problemas relacionados à quantidade de sequências.

Essa limitação pode levar a resultados sub-ótimos apesar de testarmos uma grande quantidade de técnicas de ML e balanceamentos. Devido a variabilidade genética do HIV, o ideal seria obter mais sequências.

Outra limitação é que a técnica de balanceamento escolhida, SMOTE, pode ter criado observações irreais e contribuído para piorar o resultado final. Outras foram testadas ao longo do desenvolvimento e não houve melhora nos resultados, porém é necessário, em próximas considerações metodológicas, realizar testes e comparar balanceamentos para levar a conclusões mais relevantes.

Os desempenhos não melhoraram muito mesmo com a aplicação do *stacking*, então embora esta seja uma abordagem relevante para o problema, dado que já existe uma gama de testes disponíveis na literatura, há que se buscar outras soluções.

5.3 Trabalhos futuros

Como trabalho futuro, uma melhor metodologia talvez seja focar nas diferenças entre os subtipos de modo a aumentar a sensibilidade, além de investigar as outras áreas do genoma que possam influenciar a grande variabilidade no tropismo viral em diferentes populações do mesmo subtipo.

Outra abordagem inclui fazer métodos próprios de classificação de tropismo, com maior foco no subtipo C e outros de países subdesenvolvidos, melhorando o entendimento biológico sobre o problema.

Além disso, como resultado futuro, pretende-se fazer interface gráfica para facilitar a utilização dos resultados por pesquisadores e profissionais de saúde, tanto da abordagem de *stacking* como de outras que forem desenvolvidas.

Este trabalho é resultante de projeto de Iniciação Científica ainda em andamento, tendo sido os resultados parciais apresentados no XX Simpósio Brasileiro de Computação Aplicada à Saúde - SBCAS e na 19º Jornada de Iniciação Científica da

UNIRIO - JIC, modalidade assíncrona, na categoria de Matemática e Estatística. A publicação dos resultados finais está sendo pleiteada em revistas internacionais relacionadas ao tema. As próximas etapas da Iniciação Científica levam em conta pontos abordados nesta seção de trabalhos futuros.

Referências Bibliográficas

AUTO-MUTE. **AUTO-MUTE:** HIV-1 Co-receptor Usage. Disponível em: <http://binf.gmu.edu/automute/AUTO-MUTE_HIV-1_Co-receptor_Usage.html>. Acesso em: 21 abr. 2021.

AVINASH, N. **(Tutorial) Support Vector Machines (SVM) in Scikit-learn**. DataCamp Community, 27 dez. 2019. Disponível em: <https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python>. Acesso em: 22 maio 2021.

BAGNALL, A. J.; BOSTROM, A.; CAWLEY, G. C.; et al. Is rotation forest the best classifier for problems with continuous features? **CoRR**, v. abs/1809.06705, 2018. Disponível em: <http://arxiv.org/abs/1809.06705>.

BEAM, A. L.; KOHANE, I. S. Big Data and Machine Learning in Health Care. **JAMA**, v. 319, n. 13, p. 1317, 2018. Disponível em: <doi:10.1001/jama.2017.18391>. Acesso em: 24 fev. 2021.

MINISTÉRIO DA SAÚDE. **Nota Técnica n° 30/2012 (atualizada em 3/12/2015)**. Brasil, 2012a. Disponível em: <http://www.saude.gov.br/images/pdf/2016/janeiro/12/maraviroque-3.12.2015TCGF.pdf> Acesso em: 4 abr. 2020.

MINISTÉRIO DA SAÚDE. **Maraviroque para pacientes em terapia antirretroviral**. CONITEC - Comissão Nacional de Incorporação de Tecnologias no Sistema Único de Saúde, Brasil, 2012b. Disponível em: <http://conitec.gov.br/images/Incorporados/Maraviroque-AIDS-final.pdf> Acesso em: 4 abr. 2020.

BATTISTI, I. D. E.; SMOLSKI, F. M. S. **Capítulo 7 Regressão Logística | Software R:** curso avançado. [s.l.: s.n., s.d.]. Disponível em: <https://smolski.github.io/livroavancado/reglog.html>. Acesso em: 24 maio 2021.

BERGMEIR, C.; BENÍTEZ, J. Neural Networks in R Using the Stuttgart Neural Network Simulator: RSNNS. **Journal of Statistical Software**, v. 46, n. 7, p. 1–26, 2012.

BOWDER, D.; HOLLINGSEAD, H.; DURST, K.; et al. Contribution of the gp120 V3 loop to envelope glycoprotein trimer stability in primate immunodeficiency viruses. **Virology**, v. 521, p. 158–168, 2018.

BREIMAN, L.; SPECTOR, P. Submodel Selection and Evaluation in Regression. The X-Random Case. **International Statistical Review / Revue Internationale de Statistique**, v. 60, n. 3, p. 291–319, 1992.

BRIGGS, D. R.; TUTTLE, D. L.; SLEASMAN, J. W.; et al. Envelope V3 amino acid sequence predicts HIV-1 phenotype (co-receptor usage and tropism for macrophages). **AIDS**, v. 14, n. 18, 2000. Disponível em: <https://journals.lww.com/aidsonline/Fulltext/2000/12220/Envelope_V3_amino_acid_sequence_predicts_HIV_1.16.aspx>.

BRUMME, Z.; DONG, W.; YIP, B.; et al. Clinical and immunological impact of HIV envelope V3 sequence variation after starting initial triple antiretroviral therapy. **AIDS**, v. 18, n. 4, 2004.

CABRAL, G. B. **Avaliação da resposta à terapia antirretroviral de resgate contendo antagonista do correceptor CCR5 em pessoas vivendo com HIV/AIDS**. 2014. 118 p. Programa de Pós-Graduação em Ciências (Tese de Mestrado). Coordenadoria de Controle de Doenças da Secretaria de Estado da Saúde de São Paulo, São Paulo, 2014.

CARDOZO, T.; KIMURA, T.; PHILPOTT, S.; et al. Structural Basis for Coreceptor Selectivity by The HIV Type 1 V3 Loop. **AIDS Research and Human Retroviruses**, v. 23, n. 3, p. 415–426, 2007.

CASHIN, K.; GRAY, L. R.; HARVEY, K. L.; et al. Reliable Genotypic Tropism Tests for the Major HIV-1 Subtypes. **Scientific Reports**, v. 5, n. 1, p. 1–8, 2015.

CHARIF, D.; LOBRY, J. R. **SeqinR 1.0-2**: a contributed package to the R project for statistical computing devoted to biological sequences retrieval and analysis. In: BASTOLLA, U.; PORTO, M.; ROMAN, H. E.; et al (Orgs.). Structural approaches to sequence evolution: Molecules, networks, populations. New York: Springer Verlag, 2007.

CHAWLA, N. V. et al. SMOTE: Synthetic Minority Over-sampling Technique. **Journal of Artificial Intelligence Research**, v. 16, p. 321–357, 2002.

CHIOU, S. H.; FREED, E. O.; PANGANIBAN, A. T.; et al. Studies on the role of the V3 loop in human immunodeficiency virus type 1 envelope glycoprotein function. **AIDS research and human retroviruses**, v. 8, n. 9, p. 1611–1618, 1992.

CRAN - The Comprehensive R Archive Network. **The Comprehensive R Archive Network**. Disponível em: <https://cran.r-project.org/>. Acesso em: 30 mar. 2021.

DANNENBERG, R. B.; THOM, B.; WATSON, D. **A Machine Learning Approach to Musical Style Recognition**. School of Computer Science, Carnegie Mellon University, 2004.

DE JONG, J. J.; DE RONDE, A.; KEULEN, W.; et al. Minimal requirements for the human immunodeficiency virus type 1 V3 domain to support the syncytium-inducing phenotype: analysis by single amino acid substitution. **Journal of Virology**, v. 66, n. 11, p. 6777–6780, 1992.

FACELI, K. et al. **Inteligência Artificial:** Uma abordagem de aprendizado de máquina. Rio de Janeiro: LTC, 2011.

FREIRE, S. M. **19 Regressão Linear | Bioestatística Básica**. [s.l.: s.n., s.d.]. Disponível em: <http://www.lampada.uerj.br/arquivosdb/_book/regress%C3%A3o-linear.html>. Acesso em: 22 maio 2021.

GENO2PHENO. **Geno2Pheno coreceptor**. Disponível em: <https://coreceptor.geno2pheno.org/index.php>. Acesso em: 21 abr. 2021.

GHO - GLOBAL HEALTH OBSERVATORY. **By category | Number of people (all ages) living with HIV - Estimates by WHO region**. GHO, 2019a. Disponível em: <http://apps.who.int/gho/data/view.main.22100WHO?lang=en>. Acesso em: 24 fev. 2021.

GHO - GLOBAL HEALTH OBSERVATORY. **By category | Number of people (all ages) living with HIV - Estimates by country**. GHO, 2019b. Disponível em: <http://apps.who.int/gho/data/view.main.22100?lang=en>. Acesso em: 24 fev. 2021.

GIT. **git --fast-version-control**. Disponível em: <https://git-scm.com/>. Acesso em: 2

abr. 2021a.

GIT. **A Short History of Git**. Disponível em: <https://git-scm.com/book/en/v2/Getting-Started-A-Short-History-of-Git>. Acesso em: 2 abr. 2021b.

GLAS, A. S.; LIJMER, J. G.; PRINS, M. H.; et al. The diagnostic odds ratio: a single indicator of test performance. **Journal of Clinical Epidemiology**, v. 56, n. 11, p. 1129–1135, 2003.

GORMAN, B. **mltools: Machine Learning Tools**. [s.l.: s.n.], 2018. Disponível em: <https://CRAN.R-project.org/package=mltools>.

GRÄF, T.; PINTO, A. R. The increasing prevalence of HIV-1 subtype C in Southern Brazil and its dispersion through the continent. **Virology**, v. 435, n. 1, p. 170–178, 2013.

GRÄF, T.; MACHADO FRITSCH, H.; DE MEDEIROS, R. M.; et al. Comprehensive Characterization of HIV-1 Molecular Epidemiology and Demographic History in the Brazilian Region Most Heavily Affected by AIDS. **Journal of Virology**, v. 90, n. 18, p. 8160–8168, 2016.

GUPTA, S.; NEOGI, U.; SRINIVASA, H.; et al. Performance of genotypic tools for prediction of tropism in HIV-1 subtype C V3 loop sequences. **Intervirology**, v. 58, n. 1, p. 1–5, 2015.

HEIDER, D.; DYBOWSKI, J. N.; WILMS, C.; et al. A simple structure-based model for the prediction of HIV-1 co-receptor tropism. **BioData Mining**, v. 7, p. 14, 2014.

HUNG, C.; VANDER HEYDEN, N.; RATNER, L. Analysis of the Critical Domain in the V3 Loop of Human Immunodeficiency Virus Type 1 gp120 Involved in CCR5 Utilization. **Journal of Virology**, v. 73, n. 10, p. 8216, 1999.

IHAKA, R. **R:** Past and Future History A Draft of a Paper for Interface '98, Statistics Department, The University of Auckland, Auckland, New Zealand, 1998. Disponível em: <https://www.stat.auckland.ac.nz/~ihaka/downloads/Interface98.pdf>.

IWAMOTO, A.; HOSOYA, N.; KAWANA-TACHIKAWA, A. HIV-1 tropism. **Protein & Cell**, v. 1, n. 6, p. 510–513, 2010.

JEFFERSON, Z. **Bank Data: SMOTE**. Medium, 31 ago. 2020. Disponível em: <https://medium.com/analytics-vidhya/bank-data-smote-b5cb01a5e0a2>. Acesso em: 22 maio 2021.

JENSEN, M. A.; LI, F.-S.; VAN 'T WOUT, A. B.; et al. Improved Coreceptor Usage Prediction and Genotypic Monitoring of R5-to-X4 Transition by Motif Analysis of Human Immunodeficiency Virus Type 1 env V3 Loop Sequences. **Journal of Virology**, v. 77, n. 24, p. 13376–13388, 2003.

JENSEN, M. A.; COETZER, M.; VAN 'T WOUT, A. B.; et al. A Reliable Phenotype Predictor for Human Immunodeficiency Virus Type 1 Subtype C Based on Envelope V3 Sequences. **Journal of Virology**, v. 80, n. 10, p. 4698–4704, 2006.

JOSÉ, I. **KNN (K-Nearest Neighbors) #1**. Medium, 1 jul. 2018. Disponível em: <https://medium.com/brasil-ai/knn-k-nearest-neighbors-1-e140c82e9c4e>. Acesso em: 22 maio 2021.

KAWASHIMA, S.; OGATA, H.; KANEHISA, M. AAindex: Amino Acid Index Database. **Nucleic Acids Research**, v. 27, n. 1, p. 368–369, 1999.

KLEIN, P.; KANEHISA, M.; DELISI, C. Prediction of protein function from sequence properties. Discriminant analysis of a data base. **Biochimica Et Biophysica Acta**, v. 787, n. 3, p. 221–226, 1984.

KUCAK, D.; JURICIC, V.; DAMBIC, G. **Machine Learning in Education - a Survey of Current Research Trends**. In: Proceedings of the 29th DAAAM International Symposium, Vienna, Austria, 2018.

KUHN, M.; WING, J.; WESTON, S.; et al. **caret:** Classification and Regression Training. [s.l.: s.n.], 2019. Disponível em: <https://CRAN.R-project.org/package=caret>.

KUHN, M.; WING, J.; WESTON, S.; et al. **Package 'caret' - Documentation Version 6.0-86**. [s.l.: s.n.], 2020. Disponível em: <https://cran.r-project.org/web/packages/caret/caret.pdf>. Acesso em: 13 fev. 2020.

KUIKEN, C. et al. **A compilation and analysis of nucleic acid and amino acid sequences**. In: Human Retroviruses and AIDS. Los Alamos, New Mexico: Theoretical Biology and Biophysics Group, Los Alamos National Laboratory, 1999.

KUIKEN, C.; KORBER, B.; SHAFER, R. W. HIV Sequence Databases. **AIDS reviews**, v. 5, n. 1, p. 52–61, 2003.

KUMARI, S.; CHOUHAN, U.; SURYAWANSHI, S. K. Machine learning approaches to study HIV/AIDS infection: A Review. **Bioscience Biotechnology Research Communications**, v. 10, n. 1, p. 34–43, 2017.

LARDINOIS, F.; LUNDEN, I. **Microsoft has acquired GitHub for $7.5B in stock**. TechCrunch, 4 jun. 2018. Disponível em: <https://social.techcrunch.com/2018/06/04/microsoft-has-acquired-github-for-7-5b-in-microsoft-stock/>. Acesso em: 2 abr. 2021.

LENGAUER, T.; SANDER, O.; SIERRA, S.; et al. Bioinformatics prediction of HIV coreceptor usage. **Nature Biotechnology**, v. 25, n. 12, p. 1407–1410, 2007.

LIN, N. H.; KURITZKES, D. R. Tropism testing in the clinical management of HIV-1 infection. **Current opinion in HIV and AIDS**, v. 4, n. 6, p. 481–487, 2009.

LÖCHEL, H. F.; RIEMENSCHNEIDER, M.; FRISHMAN, D.; et al. SCOTCH: subtype A coreceptor tropism classification in HIV-1. **Bioinformatics** (Oxford, England), v. 34, n. 15, p. 2575–2580, 2018.

LOS ALAMOS. **HIV Databases**. Disponível em: <https://www.hiv.lanl.gov/content/index>. Acesso em: 31 mar. 2021.

LUNARDON, N.; MENARDI, G.; TORELLI, N. ROSE: a Package for Binary Imbalanced Learning. **The R Journal**, v. 6, n. 1, p. 79, 2014.

MARCOT, B. G.; HANEA, A. M. What is an optimal value of k in k-fold cross-validation in discrete Bayesian network analysis? **Computational Statistics**, 2020. Disponível em: <https://doi.org/10.1007/s00180-020-00999-9>. Acesso em: 16 mar. 2021.

MARR, B. **A Short History of Machine Learning -- Every Manager Should Read**. Forbes, 19 fev. 2016. Disponível em: <https://www.forbes.com/sites/bernardmarr/2016/02/19/a-short-history-of-machine-learning-every-manager-should-read/>. Acesso em: 24 fev. 2021.

MASSO, M.; VAISMAN, I. I. Accurate and efficient gp120 V3 loop structure based models for the determination of HIV-1 co-receptor usage. **BMC Bioinformatics**, v. 11, p. 494, 2010.

MAYR, A.; KISSKALT, D.; MEINERS, M.; et al. Machine Learning in Production – Potentials, Challenges and Exemplary Applications. **Procedia CIRP**, v. 86, p. 49–54, 2019.

MCHUGH, M. L. Interrater reliability: the kappa statistic. **Biochemia Medica**, v. 22, n. 3, p. 276–282, 2012.

MENEZES, R.; RAPOSO, L. M. Desempenho de ferramentas genotípicas e stacking na predição de tropismo do subtipo C do HIV-1. *In*: **Anais Estendidos do Simpósio Brasileiro de Computação Aplicada à Saúde (SBCAS)**. [s.l.]: SBC, 2020a, p. 99–104. Disponível em: <https://sol.sbc.org.br/index.php/sbcas_estendido/article/view/11565>. Acesso em: 21 abr. 2021.

MENEZES, R.; RAPOSO, L. **HIV Tropism Ensemble Methods (Version v1.0) [Data set]**. Zenodo, 23 jun. 2020b. Disponível em: <http://doi.org/10.5281/zenodo.3905343>. Acesso em: 2 abr. 2021.

MOHANTY, A. **Multi layer Perceptron (MLP) Models on Real World Banking Data**. Medium, 15 maio 2019. Disponível em: <https://becominghuman.ai/multi-layer-perceptron-mlp-models-on-real-world-banking-data-f6dd3d7e998f>. Acesso em: 22 maio 2021.

NAKAI, K.; KIDERA, A.; KANEHISA, M. Cluster analysis of amino acid indices for prediction of protein structure and function. **Protein Engineering**, v. 2, n. 2, p. 93–100, 1988.

NCBI. **The Genetic Codes**. Disponível em: <https://www.ncbi.nlm.nih.gov/Taxonomy/Utils/wprintgc.cgi?chapter=tgencodes#SG1>. Acesso em: 6 abr. 2021.

NIH - National Institute of Allergy and Infectious Diseases. **HIV Replication Cycle**. Disponível em: <https://www.niaid.nih.gov/diseases-conditions/hiv-replication-cycle>. Acesso em: 25 maio 2021.

NEEDLEMAN, S. B.; WUNSCH, C. D. A general method applicable to the search for similarities in the amino acid sequence of two proteins. **Journal of Molecular Biology**, v. 48, n. 3, p. 443–453, 1970.

OZA, N. C.; TUMER, K. Classifier ensembles: Select real-world applications. **Information Fusion**, v. 9, n. 1, p. 4–20, 2008.

PAGÈS, H.; ABOYOUN, P.; GENTLEMAN, R.; et al. **Biostrings:** Efficient manipulation of biological strings. [s.l.: s.n.], 2019.

PHENOSEQ. **PhenoSeq**. Disponível em: <http://tools.burnet.edu.au/phenoseq/>. Acesso em: 21 abr. 2021.

POLIKAR, R. Ensemble based systems in decision making. **IEEE Circuits and Systems Magazine**, v. 6, n. 3, p. 21–45, 2006.

POWERS, D. M. W. Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation. **Journal of Machine Learning Technologies**, v. 2, n. 1, p. 37–63, 2011.

POWERS, D. M. W. What the F-measure doesn't measure: Features, Flaws, Fallacies and Fixes. **CoRR**, v. abs/1503.06410, 2015. Disponível em: <http://arxiv.org/abs/1503.06410>.

PYPI. **The Python Package Index**. PyPI. Disponível em: <https://pypi.org/>. Acesso em: 2 abr. 2021.

PYTHON. **Welcome to Python.org**. Python.org. Disponível em: <https://www.python.org/>. Acesso em: 2 abr. 2021.

RAYMOND, S.; DELOBEL, P.; IZOPET, J. Phenotyping methods for determining HIV tropism and applications in clinical settings. **Current opinion in HIV and AIDS**, v. 7, n. 5, p. 463–469, 2012.

RAYMOND, S.; DELOBEL, P.; MAVIGNER, M.; et al. Correlation between genotypic predictions based on V3 sequences and phenotypic determination of HIV-1 tropism. **AIDS** (London, England), v. 22, n. 14, p. F11-16, 2008.

R CORE TEAM. **R:** A Language and Environment for Statistical Computing. Vienna, Austria: R Foundation for Statistical Computing, 2019.

RIEMENSCHNEIDER, M.; CASHIN, K. Y.; BUDEUS, B.; et al. Genotypic Prediction of Co-receptor Tropism of HIV-1 Subtypes A and C. **Scientific Reports**, v. 6, n. 1, p. 1–9, 2016.

RODRÍGUEZ, J.; KUNCHEVA, L.; ALONSO, C. Rotation Forest: A New Classifier Ensemble Method. **IEEE transactions on pattern analysis and machine intelligence**, v. 28, p. 1619–30, 2006.

ROKACH, L. Decision forest: Twenty years of research. **Information Fusion**, v. 27, p. 111–125, 2016.

ROKACH, L. **Decision Forest:** Twenty Years of Research. SlideShare, 12 jul. 2015. Disponível em: <https://pt.slideshare.net/liorrokach/dmbi-talk>. Acesso em: 30 maio 2021.

RUIZ-SAMBLÁS, C.; CADENAS, J. M.; PELTA, D. A.; et al. Application of data mining methods for classification and prediction of olive oil blends with other vegetable oils. **Analytical and Bioanalytical Chemistry**, v. 406, n. 11, p. 2591–2601, 2014.

SELENIUM. **SeleniumHQ Browser Automation**. Disponível em: <https://www.selenium.dev/>. Acesso em: 2 abr. 2021.

SETUNGA, S. **Stacking in Machine Learning**. 10 jun. 2016. Disponível em: <http://supunsetunga.blogspot.com/2016/06/stacking-in-machine-learning.html>. Acesso em: 22 maio 2021.

SILVA, A. R. **Análise de Componentes Principais (PCA):** cálculo e aplicação no R. Oper, 17 dez. 2020. Disponível em: <https://operdata.com.br/blog/analise-de-componentes-principais-pca-calculo-e-aplicacao-no-r/>. Acesso em: 30 maio 2021.

ŠIMUNDIĆ, A. Measures of Diagnostic Accuracy: Basic Definitions. **EJIFCC**, v. 19, n. 4, p. 203–211, 2009.

SIRVEN, J.-B.; SALLÉ, B.; MAUCHIEN, P.; et al. Feasibility study of rock identification at the surface of Mars by remote laser-induced breakdown spectroscopy and three chemometric methods. **Journal of Analytical Atomic Spectrometry**, v. 22, n. 12, p. 1471–1480, 2007.

SOULIÉ, C.; FOFANA, D. B.; BOUKLI, N.; et al. Performance of genotypic algorithms for predicting tropism of HIV-1CRF02_AG subtype. **Journal of Clinical Virology: The Official Publication of the Pan American Society for Clinical Virology**, v. 76, p. 51–54, 2016.

SRIVASTAVA, T. **Introduction to KNN, K-Nearest Neighbors**: Simplified. Analytics Vidhya, 26 mar. 2018. Disponível em: <https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/>. Acesso em: 23 mar. 2020.

SWENSON, L. C.; DÄUMER, M.; PAREDES, R. Next-generation sequencing to assess HIV tropism. **Current opinion in HIV and AIDS**, v. 7, n. 5, p. 478–485, 2012.

TEBIT, D. M.; ARTS, E. J. Tracking a century of global expansion and evolution of HIV to drive understanding and to combat disease. **The Lancet**. Infectious Diseases, v. 11, n. 1, p. 45–56, 2011.

TECHCRUNCH. **GitHub Pours Energies into Enterprise – Raises $100 Million From Power VC Andreessen Horowitz**. TechCrunch, 9 jul. 2012. Disponível em: <https://social.techcrunch.com/2012/07/09/github-pours-energies-into-enterprise-raises-100-million-from-power-vc-andreesen-horowitz/>. Acesso em: 2 abr. 2021.

TOLLES, J.; MEURER, W. J. Logistic Regression: Relating Patient Characteristics to Outcomes. **JAMA**, v. 316, n. 5, p. 533–534, 2016.

TOMII, K.; KANEHISA, M. Analysis of amino acid indices and mutation matrices for sequence comparison and structure prediction of proteins. **Protein Engineering**, v. 9, n. 1, p. 27–36, 1996.

TORGO, L. **Data Mining with R, learning with case studies**. [s.l.]: Chapman and Hall/CRC, 2010.

UNIRIO - UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO. **19ª Jornada de Iniciação Científica da UNIRIO - Prêmio de Iniciação Científica:** Apresentação Assíncrona - Por área de conhecimento - Formato vídeo. UNIRIO, 2020. Disponível em: <http://www.unirio.br/jic/premio-de-iniciacao-cientifica/2020/premiacao-assincrona/>. Acesso em: 21 abr. 2021.

VANDEN, K. B.; HUBERT, M. Robust classification in high dimensions based on the SIMCA Method. **Chemometrics and Intelligent Laboratory Systems**, v. 79, n. 1, p. 10–21, 2005.

VARELLA, C. A. A. **Análise de Componentes Principais**. 11 dez. 2008. Disponível em: <http://www.ufrrj.br/institutos/it/deng/varella/Downloads/multivariada%20aplicada%20as%20ciencias%20agrarias/Aulas/analise%20de%20componentes%20principais.pdf>. Acesso em: 30 maio 2021.

WEB PSSM. **Web PSSM**. Disponível em: <https://indra.mullins.microbiol.washington.edu/webpssm/>. Acesso em: 21 abr. 2021.

WICKHAM, H. **ggplot2:** Elegant Graphics for Data Analysis. [s.l.]: Springer-Verlag New York, 2016. Disponível em: <https://ggplot2.tidyverse.org>.

WICKHAM, H. **stringr:** Simple, Consistent Wrappers for Common String Operations. [s.l.: s.n.], 2019. Disponível em: <https://CRAN.R-project.org/package=stringr>.

WU, K. J. **There are more viruses than stars in the universe. Why do only some infect us?** Science. National Geographic, 15 abr. 2020. Disponível em: <https://www.nationalgeographic.com/science/article/factors-allow-viruses-infect-humans-coronavirus>. Acesso em: 25 fev. 2021.

YAN, X. e GANG SU, X. **Linear Regression Analysis:** Theory and Computing. [s.l.]: World Scientific, 2009.

ZENODO. **Zenodo - Research. Shared.** Disponível em: <https://about.zenodo.org/>. Acesso em: 2 abr. 2021.

Apêndice 01 - Código do R para preparar as sequências genéticas.

```
# Carregamento de pacotes.
library(seqinr)
library(Biostrings)
library(stringr)


# Definição de constantes.
arquivo = "arquivo.fasta"
arquivoSaida = "saida.csv"

consenso_C = "CTRPNNNTRKSIRIGPGQTFYATGDIIGDIRQAHC"
consenso_B = "CTRPNNNTRKSIHIGPGRAFYTTGEIIGDIRQAHC"


# Funções.

# Deve iniciar com C.
# Localiza início da região V3 com base nos aminoácidos (aa) esperados.
localizaInicioV3 <- function(aa.str) {
    # X nos padrões abaixo é um aa desconhecido.
    inicioV3 <- str_locate_all(pattern = "C.R", aa.str)[[1]][1]
    if (is.na(inicioV3)) {
       inicioV3 <- str_locate_all(pattern = "C.G", aa.str)[[1]][1]
    }
    if (is.na(inicioV3)) {
       inicioV3 <- str_locate_all(pattern = "C.X", aa.str)[[1]][1]
    }
    return(inicioV3)
}

# Deve terminar com C.
# Localiza final da região V3 com base nos aa esperados.
localizaFimV3 <- function(aa.str) {
    fimV3 <- str_locate_all(pattern = "A.C", aa.str)[[1]][2]
    if (is.na(fimV3)) {
       fimV3 <- str_locate_all(pattern = "X.C", aa.str)[[1]][2]
    }
    if (is.na(fimV3)) {
       fimV3 <- str_locate_all(pattern = "T.C", aa.str)[[1]][2]
    }
    if (is.na(fimV3)) {
       fimV3 <- str_locate_all(pattern = "P.C", aa.str)[[1]][2]
    }
    if (is.na(fimV3)) {
       fimV3 <- str_locate_all(pattern = "S.C", aa.str)[[1]][2]
    }
    return(fimV3)
}


# Leitura do arquivo.
seqsDNA <- read.fasta(file = arquivo, seqtype = c("DNA"), strip.desc = TRUE)


dados <- data.frame(name=c(1:36))
cont <- 1
for (seq in seqsDNA) {
    # Remove intervalos/gaps ('-').
```

```
    dna.str <- paste(seq, collapse="")
    dna.str <- gsub("-", "", dna.str)
    dna.vector <- substring(dna.str, seq(1, nchar(dna.str), 1), seq(1,
nchar(dna.str), 1))


    # Conversão para aa.
    aa <- seqinr::translate(dna.vector, frame = 0, sens = "F", numcode = 1,
NAstring = "X", ambiguous = FALSE)


    # Corta a sequência.
    aa.str <- paste(aa, collapse = "")

    # Se houver menos de 20% do total de aa da V3 (35), não usa a sequência.
    if (nchar(aa.str) < 28) {
       next
    }
    inicioV3 <- localizaInicioV3(aa.str)
    fimV3 <- localizaFimV3(aa.str)
    seq_cort <- str_sub(aa.str, inicioV3, fimV3)

    if (is.na(inicioV3) || is.na(fimV3)) {
       print("Sequência fora do padrão:")
       print(aa.str)
       next
    }

    # Alinha com o consenso.
    # Deve ter exatamente 35 aa a partir daqui.
    descricao <- attr(seq, "name")
    if (startsWith(descricao, "C")) {
       consenso <- consenso_C
    } else {
       # Por segurança, se não houver dados, alinha com B.
       consenso <- consenso_B
    }

    alinhamento <- pairwiseAlignment(consenso, seq_cort)
    subject <- subject(alinhamento)
    score <- score(alinhamento)

    # Substitui gaps por X, que não é nenhum aa.
    seq_cort <- gsub("-", "X", subject)

    # Adiciona na base de dados.
    aa.vector = substring(seq_cort, seq(1, nchar(seq_cort), 1), seq(1,
nchar(seq_cort), 1))
    if (length(aa.vector) != 35) {
       next
    }
    dados <- cbind(dados, c(aa.vector, score))
    colnames(dados)[cont + 1] <- descricao # muda nome da coluna
    cont <- cont + 1

}

dados[36,1] <- "score_aligment"
dadosAA <- t(dados) # transpõe
write.table(dadosAA, file = arquivoSaida, col.names=FALSE, sep=";")
```

Apêndice 02 - Código do R para *stacking*.

```
# Carregamento de pacotes.
library(caret)
library(DMwR)
library(mltools)


# Funções:
# Recebe uma matriz de confusão do pacote Caret e calcula o Matthews correlation
coefficient (MCC).
calculaMCC <- function(matrizConfusao) {
    tp = matrizConfusao$table[2,2]
    fp = matrizConfusao$table[2,1]
    tn = matrizConfusao$table[1,1]
    fn = matrizConfusao$table[1,2]

    # Função mcc abaixo precisa de library(mltools)
    return(mcc(TP = tp, FP = fp, TN = tn, FN = fn))
}


dados <- read.table("arquivoComTestesGenotipicos.csv", header = TRUE,
      sep = ";", na.strings = "NA", dec = ".", strip.white = TRUE)

# Limita apenas para as variáveis de interesse (remove DNA etc).
dados <- dados[,6:16]


# Função que treina de acordo com um método e semente aleatória.
# Ex: modelar(3, "plr")
modelar <- function(semente, metodo) {

    # Separa os dados em treino e teste (70% - 30%).
    set.seed(semente)
    inTrain = createDataPartition(dados$desfecho.reduzido, p = 0.7)[[1]]
    training = data.frame(dados[inTrain,])
    testing = data.frame(dados[-inTrain,])

    # Balanceamento dos dados com SMOTE.
    set.seed(semente)
    dadosBalance <- SMOTE(desfecho.reduzido ~ ., data  = training)

    # Cria o modelo e vê seu desempenho nos dados de teste.
    # Impede output no console.
    invisible(capture.output(
       set.seed(semente)
       modelFit <- train(desfecho.reduzido ~ ., data = dadosBalance, method =
metodo)
    ))
    pred <- predict(modelFit, newdata = testing)
    cf <- confusionMatrix(pred, testing$desfecho.reduzido, positive="NR5")

    # Método | semente | Ac | Sen | Esp | MCC | Kappa
    cat(
       metodo, "|",
       semente, "|",
       cf$overall[[1]], "|",
       cf$byClass[[1]], "|",
       cf$byClass[[2]], "|",
```

```
        calculaMCC(cf), "|",
        cf$overall[[2]], "|",
        fill=TRUE
    )

}

# Roda com 20 sementes diferentes e captura os dados relevantes do modelo.
sementes = list(9992, 9355, 6622, 9592, 9108, 3671, 4031, 3796, 7454, 7358,
7833, 859, 5279, 3134, 5186, 6957, 3813, 9045, 5714, 2711)

metodos = list("CSimca", "rotationForest", "rpart2", "knn", "mlpWeightDecay")

for (metodo in metodos) {
    for (i in 1:20) {
        modelar(sementes[[i]], metodo)
    }
}
```