UNIVERSIDADE DO ESTADO DA BAHIA (UNEB)
DEPARTAMENTO DE CIÊNCIAS EXATAS E DA TERRA, CAMPUS I
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

JOILSON DE JESUS ARGOLO

AGSSA: SISTEMA WEB DE GENOTIPAGEM RÁPIDA DE CEPAS VIRAIS, BASEADO EM APRENDIZADO DE MÁQUINA

SALVADOR
2024

JOILSON DE JESUS ARGOLO

AGSSA: SISTEMA WEB DE GENOTIPAGEM RÁPIDA DE CEPAS VIRAIS, BASEADO EM APRENDIZADO DE MÁQUINA

Monografia apresentada ao Curso de Bacharelado em Sistemas de Informação do Departamento de Ciências Exatas e da Terra (DCET) - Campus I, da Universidade do Estado da Bahia (UNEB), como requisito à obtenção do grau de bacharel em Sistemas de Informação.
Área de concentração: Bioinformática

Aprovada em: 09/12/2024.

BANCA EXAMINADORA

Prof. PhD Diego Gervasio Frías Suárez
Orientador(DCET-I/UNEB)

Prof. Dra. Maria Inés Valderrama Restovic
Examinador interno (DCET-I/UNEB)

Prof. Dr. Vagner de Souza Fonseca
Examinador interno (DCET-I/UNEB)

JOILSON DE JESUS ARGOLO

AGSSA: SISTEMA WEB DE GENOTIPAGEM RÁPIDA DE CEPAS VIRAIS, BASEADO EM APRENDIZADO DE MÁQUINA

Monografia apresentada ao Curso de Bacharelado em Sistemas de Informação do Departamento de Ciências Exatas e da Terra (DCET) - Campus I, da Universidade do Estado da Bahia (UNEB), como requisito à obtenção do grau de bacharel em Sistemas de Informação.
Área de concentração: Bioinformática

Orientador(DCET-I/UNEB): Prof. PhD Diego Gervasio Frías Suárez

SALVADOR
2024

AGRADECIMENTOS

É com profunda gratidão que dedico esta seção de agradecimentos do meu Trabalho de Conclusão de Curso. Esta jornada acadêmica não teria sido possível sem o apoio e incentivo de pessoas incríveis que estiveram ao meu lado.

Agradeço primeiramente à minha família, por ser a minha base. Vocês foram a força que impulsionou cada passo desta caminhada.

Ao meu orientador, **Prof. PhD Diego Gervasio Frias Suárez**, agradeço pela paciência, sabedoria e orientação ao longo de todo o processo de pesquisa. Seu apoio foi fundamental para o desenvolvimento deste trabalho.

Aos professores que contribuíram com seus conhecimentos e experiências, a minha sincera gratidão. Cada aula, conselho e feedback foram fundamentais para o meu crescimento acadêmico.

A **Maurício de Souza Menezes** pela amizade e apoio. A professora, **dr Marta Valeria**, pela humanidade, por acreditar no meu potencial e comprometimento, no inicio da minha jornada. Aos colegas de curso, pela troca de ideias, momentos de estudo e pelo apoio mútuo, meu muito obrigado. Compartilhamos desafios e conquistas que tornaram esta jornada ainda mais significativa.

Agradeço a todos que, de muitas forma, contribuíram para o sucesso deste trabalho. Aos amigos **André Mercês**, **João Gabriel**, **Marcelo Leonardo**, por cada palavra de encorajamento, cada gesto de apoio, foi fundamental para chegar até este momento.

Este TCC não é apenas um marco acadêmico, mas uma realização coletiva. A todos vocês, o meu mais sincero agradecimento.

"A educação é a arma mais poderosa que você pode usar para mudar o mundo."

(Nelson Mandela)

RESUMO

A genotipagem de cepas virais é uma das tarefas mais comuns e importantes nos sistemas de monitoramento de doenças ao redor do mundo, especialmente após o impacto global da pandemia de COVID-19. A maioria dessas análises é realizada com base no processamento bioinformático de genomas virais obtidos por técnicas de sequenciamento rápido. Este trabalho apresenta o desenvolvimento de uma ferramenta web inovadora, projetada para realizar a genotipagem de cepas virais de forma rápida e eficiente, utilizando aprendizado de máquina. A metodologia utilizada foi baseada no Design Science Research (DSR), que consiste em três etapas principais: (1) identificação do problema, (2) desenvolvimento do artefato, e (3) avaliação de sua eficácia. A nova ferramenta, denominada *AGSSA* (Advanced Genotyping with Semi-Supervised Algorithm), é uma versão avançada da ferramenta anterior AGUA (Advanced Genotyping with Unsupervised Algorithm), que já havia sido testada com o gene spike (S) do SARS-CoV-2. Nesta etapa, a ferramenta foi aplicada ao gene envelope (E) do vírus da Dengue. Os resultados demonstraram que a nova versão da ferramenta é capaz de processar grandes volumes de dados genômicos de maneira eficiente, com baixo custo computacional. Além disso, sua interface web torna a solução acessível e fácil de usar, mesmo por pesquisadores com pouca ou nenhuma experiência em computação avançada.

Palavras-chave: genotipagem, aprendizado de máquina, bioinformática, sistemas web, Dengue.

ABSTRACT

The genotyping of viral strains is one of the most common and crucial tasks in disease monitoring systems worldwide, especially in the aftermath of the COVID-19 pandemic. Most of these analyses rely on the bioinformatics processing of viral genomes obtained through rapid sequencing techniques. This work introduces an innovative web-based tool designed to perform viral strain genotyping quickly and efficiently using machine learning. The methodology followed was based on Design Science Research (DSR), which consists of three main steps: (1) problem identification, (2) artifact development, and (3) evaluation of its effectiveness. The new tool, named *AGSSA* (Advanced Genotyping with Semi-Supervised Algorithm), is an advanced version of the previous AGUA (Advanced Genotyping with Unsupervised Algorithm), which had already been tested with the spike (S) gene of SARS-CoV-2. In this phase, the tool was applied to the envelope (E) gene of the Dengue virus. The results showed that the new version of the tool can efficiently process large volumes of genomic data at a low computational cost. Additionally, its web-based interface makes the solution accessible and easy to use, even for researchers with little or no experience in advanced computing.

Keywords: genotyping, machine learning, bioinformatics, web system, Dengue.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

1.1 Contextualização

O surgimento do vírus SARS-CoV-2 (Coronavírus 2 da síndrome respiratória aguda grave), no final de 2019 (Zhu *et al.*, 2020) causador da epidemia de COVID-19 (Coronavirus Disease 2019), mergulhou o mundo em um estado de incerteza sem precedentes. Esse novo coronavírus, rapidamente se disseminou por todo o planeta, expondo a carência de conhecimento sobre sua origem, mecanismos de ação e epidemiologia (Organization, 2020).

A comunidade científica se viu diante de um patógeno inédito, com características e comportamentos ainda a serem desvendados (Petrosillo *et al.*, 2020). A escassez de informações precisas dificultava o desenvolvimento de estratégias eficazes de controle da pandemia, a criação de testes diagnósticos confiáveis e o direcionamento de esforços para o desenvolvimento de vacinas e terapias.

Diante da urgência em compreender o SARS-CoV-2 e seus mecanismos de ação, a análise genômica se tornou uma ferramenta crucial para desvendar os segredos do vírus. Através do sequenciamento do genoma viral, pesquisadores puderam identificar mutações, rastrear a origem e a disseminação do vírus, e entender os fatores que contribuem para sua virulência e transmissibilidade (Walls *et al.*, 2020); (Kucharski *et al.*, 2020). Com isso, gerou-se um grande volume de dados, somente em três anos de existência, segundo o *GSAID* (*Global Initiative on Sharing All Influenza Data*) a comunidade cientifica gerou quase 17 milhões de genomas de SARS-CoV-2. Para poder realmente entender o volume de dados gerado pelo sequenciamento genômico do SARS-CoV-2 durante a epidemia de COVID-19, é necessário consultar quantos genomas tem sido gerados de outros patógenos. Segundo o *GENBANK* (banco de dados de sequências de nucleotídeos publicamente disponíveis), que pertence ao *National Institute of Health* (*NIH*) dos Estados Unidos, do Zika vírus, que é sequenciado desde 1947, tem pouco mais de 2.500 genomas, enquanto do vírus da Dengue, que é sequenciado desde os anos 60, aproximadamente 17 mil genomas.

Neste cenário, marcado pelo crescente volume dos dados genômicos gerados com modernas e eficientes técnicas de sequenciamento (NGS - *Next Generation Sequencing*) (Metzker; L., 2010), a bioinformática se apresenta como uma ferramenta crucial para a análise e interpretação desses dados, impulsionando avanços em diversas áreas da ciência, especialmente no combate a pandemias (Rana, 2023). Esse método científico, que combina métodos computacionais e técnicas estatísticas, permite extrair informações valiosas de sequências genômicas e proteicas, abrindo caminho para uma compreensão mais profunda da evolução dos vírus e o desenvolvimento de estratégias eficazes contra doenças como a COVID-19 (causada pelo vírus SARS-CoV-2), a Febre da dengue (causado pelo arbovírus

DENV), a AIDS (Síndrome da Imunodeficiência Humana causada pelo vírus HIV), a Doença do Ebola (Doença de altíssima taxa de mortalidade causada pelo vírus ZEV) e a Febre Amarela (causada pelo arbovírus da Febre Amarela YFV), entre outras (Wilkinson *et al.*, 2023; Lespinet; Médigue; Lazarević, 2021).

No contexto da pandemia de COVID-19, a bioinformática foi crucial na análise das sequências do vírus SARS-CoV-2, permitindo o rastreamento da origem e evolução da doença, a identificação de novas variantes e o desenvolvimento de vacinas e medicamentos mais eficazes (Padmanabhan; Heinen; Chockalingam, 2022; Olson; Schaffer; Shoresh, 2023). Através da análise de grandes conjuntos de dados genômicos, foi possível mapear a disseminação do vírus em diferentes regiões do mundo, identificar mutações que afetam sua transmissibilidade e virulência, e desenvolver modelos preditivos para o futuro da pandemia.

Fruto da mobilização da comunidade internacional de bioinformatas, diversas ferramentas foram desenvolvidas em um curto período de tempo para analisar as novas cepas que eram sequenciadas a cada dia. Dentre essas ferramentas, destaca-se o *Genome Detective*, desenvolvida por um grupo internacional de pesquisadores da área de programação, liderado por um pesquisador brasileiro, egresso e atual docente do Colegiado de Sistemas de Informação da UNEB, o Prof. Dr. Vagner Fonseca.

O impacto causado pelo uso dessas ferramentas bioinformáticas no combate ao COVID-19 mostrou a necessidade de utilizar mais intensivamente o sequenciamento dos patógenos virais conhecidos, assim como a necessidade da melhoria continuada das técnicas bioinformáticas para o estudo da evolução viral.

1.2 Justificativa do Trabalho

O crescente volume de dados genômicos demanda o aprimoramento de ferramentas bioinformáticas mais eficientes e escaláveis (Meena; Kumar; Tomar, 2021). Contudo, a escalabilidade para considerar um grande número de sequências, na grande maioria das ferramentas atuais estudadas de uso público[1], estão limitadas por dois fatores principais:

1. Não utilizam Inteligência Artificial (IA) (OECD, 2024) para a coleta de padrões que possam ser armazenados e utilizados como referência para análise de novas cepas virais,

2. Derivado da limitação anterior, para analisar uma nova cepa, é preciso refazer o processo de clusterização, que é computacionalmente complexo (demorado e consumidor de memória), com todas as sequências de referência. acrescidas da nova sequência. Isto faz com que seja necessário estabelecer um balanceamento entre o número de sequências de referência consideradas e o tempo de análise de uma sequência nova. Como não

[1] Esta pesquisa não consultou ferramenta que não são de uso público

é viável, salvo em raras exceções (Buchfink *et al.*, 2023), incluir todas as sequências distintas disponíveis em um conjunto de referência, este acaba por não representar toda a diversidade genética já sequenciada. Essa limitação, na prática, impede a identificação plenamente confiável de novas cepas. Em outras palavras, não é possível afirmar com total certeza que uma cepa recém-sequenciada é radicalmente diferente de todas as outras previamente sequenciadas, caso o conjunto de referência não inclua todas as sequências distintas já disponíveis.

Neste cenário, se justifica a exploração de abordagens para o problema de genotipagem viral que utilizem a IA e, em particular, o AM. (Goodfellow; Bengio; Courville, 2016). Neste sentido, o presente projeto se alinha nos esforços do Grupo de Pesquisas em Bioinformática e Biologia Computacional (G2BC) da UNEB para desenvolver uma ferramenta web para genotipagem viral baseada em AM.

1.3 Objetivos do Trabalho

O objetivo geral foi desenvolver a primeira versão web da ferramenta *AGSSA* (**A**dvanced **G**enotyping with **S**emi-**S**upervised **A**lgorithm) para a genotipagem rápida de sequências virais, realizando o aprimoramento e validação do algoritmo de classificação de sequências baseado em AM.

Os objetivos específicos foram:

1. Modificar o algoritmo de treinamento para que encontre de forma autônoma o melhor parâmetro de repulsão, ou seja, aquele que permite obter o menor número de clusters sendo todos puros[2].

2. Projetar, desenvolver e validar um pipeline (Backend da aplicação web) para a classificação de novas sequências utilizando os modelos pré-treinados.

3. Projetar e desenvolver camadas frontend e intermediária da ferramenta web *AGSSA* para teste.

4. Aferir o desempenho, tanto na qualidade dos agrupamentos quanto no tempo de execução, da nova versão do algoritmo, utilizando sequências anotadas do vírus da Dengue, comparando com ferramenta clássica de filogenia do estado da arte.

5. Avaliação funcional da ferramenta desenvolvida e proposta de melhorias para a próxima fase de desenvolvimento.

[2] Um cluster puro contem apenas sequências de uma única variante viral

1.4 Contribuições Esperadas

Este trabalho teve várias contribuições significativas ao campo da bioinformática e à luta contra as pandemias virais ao facilitar a identificação rápida de grandes volumes de sequências de cepas virais. A seguir, destacam-se as principais contribuições deste projeto:

1. **Integração de Aprendizado de Máquina:**

   A aplicação de técnicas de aprendizado de máquina (AM), no processamento de dados genômicos oferece uma abordagem mais eficiente para a análise de grandes volumes de dados. A IA permite a identificação de padrões e a utilização de sequências de referência de maneira mais inteligente, reduzindo a necessidade de refazer processos complexos de clusterização.

2. **Desenvolvimento de Ferramenta Inovadora:**

   A criação da ferramenta AGSSA, que utiliza algoritmos de AM para genotipagem viral, representa um avanço significativo em relação às ferramentas tradicionais. A ferramenta *AGSSA* facilita a análise rápida e precisa de sequências virais, o que é crucial para o monitoramento e controle de pandemias.

3. **Melhoria de Algoritmos de Clusterização:**

   A otimização do algoritmo de clusterização CLOPE melhora significativamente o desempenho do backend. Esta otimização permite um processamento mais rápido e eficiente das sequências, possibilitando análises em tempo real e com menor consumo de recursos computacionais.

4. **Avaliação Comparativa com Ferramentas Clássicas:**

   A comparação dos resultados gerados pela *AGSSA* com ferramentas clássicas de filogenia e genotipagem estabelece um benchmark de desempenho. Esta avaliação ajuda a validar a eficácia da ferramenta e a identificar áreas de melhoria contínua.

5. **Validação com Dados Reais:**

   A validação da ferramenta *AGSSA* utilizando conjuntos de dados reais, como as sequências de SARS-CoV-2 (trabalho anterior) e do vírus da dengue (neste trabalho), assegura a aplicabilidade e a robustez da ferramenta em contextos práticos. Esta validação demonstra a capacidade de *AGSSA* de lidar com diferentes tipos de vírus e suas variantes.

6. **Facilitação do Acesso e Uso da Ferramenta:**

   O desenvolvimento das camadas frontend e intermediária disponibiliza a *AGSSA* na web, permitindo que pesquisadores e profissionais de saúde possam utilizar a ferramenta

de forma acessível e intuitiva. Esta fase piloto é fundamental para testar e refinar a usabilidade da ferramenta antes de sua implementação em larga escala.

7. **Propor Melhorias Futuras da Ferramenta:**

   A avaliação prática do uso da ferramenta permitirá identificar oportunidades de melhoria, assegurando que *AGSSA* seja capaz de atender plenamente às demandas dos pesquisadores em evolução viral.

O restante desta monografia está organizado da seguinte forma: o Capítulo 2 - Fundamentação Teórica apresenta os conceitos e bases teóricas indispensáveis para compreender os fundamentos do estudo e trabalhos correlatos, destacando as lacunas que este projeto busca preencher. No Capítulo 3 - Desenvolvimento do Projeto, são descritas a metodologia geral adotada, além dos processos e métodos empregados no desenvolvimento da ferramenta web e no aprimoramento do backend. O Capítulo 4 - Contribuições do Projeto ressalta as inovações e os avanços proporcionados pelo estudo. No Capítulo 5 - Discussão de Resultados, são apresentados, analisados e interpretados os dados obtidos ao longo do projeto. Finalmente, o Capítulo 6 - Considerações Finais reflete sobre os resultados alcançados, discute as limitações do estudo e sugere caminhos para futuras pesquisas.

Além desses sete capítulos, a monografia inclui apêndices que aprofundam aspectos técnicos e metodológicos específicos. Entre eles, estão descritos a transformação de sequências de nucleotídeos em vetores, o modelo matemático do método *CLOPE*, a estrutura e atualização do modelo de dados, o pipeline de classificação de sequências e as instruções de instalação da ferramenta *AGSSA*. Esses apêndices oferecem uma visão detalhada e prática sobre os componentes e ferramentas desenvolvidos no projeto, proporcionando um entendimento completo das suas implementações e aplicações.

2 FUNDAMENTAÇÃO TEÓRICA

A biologia molecular é uma área essencial para a compreensão dos processos biológicos que sustentam a vida e também tem implicações importantes para a tecnologia, especialmente no que se refere à análise de dados genéticos. No contexto do nosso trabalho, dois processos fundamentais que precisam ser bem compreendidos são a transcrição e a tradução, processos que convertem a informação genética armazenada no DNA em proteínas funcionais, que são essenciais para a célula. Esses processos são de especial interesse quando consideramos ferramentas computacionais para a análise de sequências genéticas, como o AGSSA(Advanced Genotyping with Semi-Supervised Algorithm), que têm um papel crescente em áreas como bioinformática e genotipagem viral.

O processo de transcrição ocorre no núcleo da célula e é o primeiro passo para a expressão de um gene. Durante a transcrição, a sequência de nucleotídeos no DNA é copiada para uma molécula de RNA mensageiro (mRNA). A enzima RNA polimerase desempenha um papel crucial nesse processo, lendo as fitas de DNA e gerando uma cadeia de mRNA complementar. Cada sequência de três nucleotídeos no mRNA, chamados de códon, codifica para um aminoácido específico. O mRNA então deixa o núcleo e vai para o citoplasma, onde ocorre o próximo passo: a tradução.

A tradução ocorre no citoplasma e é a etapa final para a produção de proteínas. O mRNA transcrito é lido pelos ribossomos, que são complexos moleculares que atuam como "fábricas de proteínas". O mRNA contém os códons que guiarão a construção da proteína, e os RNA de transferência (tRNA) têm a função de transportar os aminoácidos específicos para os ribossomos, de acordo com a sequência de códons. Esse processo resulta em uma cadeia de aminoácidos que, após sua formação, se dobra em uma estrutura tridimensional, tornando-se uma proteína funcional. Essa proteína pode desempenhar diversas funções celulares, desde enzimas até proteínas estruturais.

Além de ser um processo fundamental para as células, a transcrição e a tradução também são cruciais para os vírus, especialmente os vírus de RNA, como o HIV, o vírus da gripe e os coronavírus. Esses vírus invadem células hospedeiras e se aproveitam das maquinarias celulares para replicar seu material genético e produzir as proteínas necessárias para sua propagação.

No campo da tecnologia, o processo de genotipagem viral, que envolve a análise de sequências de nucleotídeos e a identificação de mutações, requer ferramentas avançadas de processamento de dados.

A transcrição e a tradução virais não são apenas conceitos biológicos; elas têm aplicações diretas no campo da bioinformática e na análise de dados genéticos. No funcio-

namento do AGSSA, por exemplo, esses conceitos são aplicados para tratar sequências genéticas e identificar variações genotípicas que possam indicar a emergência de novas cepas virais. No AGSSA, a variabilidade genética é analisada em nível de códons, ou seja, nas unidades básicas que codificam aminoácidos e definem o processo de tradução no citoplasma celular. Essa integração de processos biológicos com tecnologia computacional exige algoritmos avançados para realizar alinhamentos de sequências, construção de árvores filogenéticas e identificação de variantes em tempo hábil, especialmente dada a urgência de monitorar a evolução dos vírus e antecipar possíveis ameaças à saúde pública.

2.1 Os vírus em geral

Os vírus, que são agentes infecciosos submicroscópicos, apresentam uma estrutura fundamental composta por um material genético (DNA ou RNA) encapsulado por uma capa proteica ou envelope lipídico. Contudo, eles somente podem se replicar dentro das células de um hospedeiro, já que não possuem estruturas capazes de transcrever e traduzir essa informação genética em proteínas funcionais. Para isso, eles precisam do sistema de transcrição/tradução da célula hospedeira, ou seja, do mecanismo de síntese de proteínas e enzimas do tipo de célula que é infectado pelo vírus (Ramirez-Martínez, 2024).

A diversidade genética viral é vasta, com milhares de espécies conhecidas infectando uma ampla gama de organismos, incluindo humanos, animais e plantas (Koonin; Dolja; Krupovic, 2019). A compreensão da diversidade genética dos vírus é crucial para a epidemiologia e para o desenvolvimento de tratamentos e vacinas. A vasta gama de genomas virais, resultante de mutações e recombinações, permite que os vírus se adaptem rapidamente a novos hospedeiros e ambientes, tornando o controle de infecções virais um desafio contínuo (Domingo; Perales, 2019). A compreensão detalhada das mutações é essencial para o desenvolvimento de vacinas e terapias eficazes. Através da vigilância genômica contínua, os cientistas podem identificar rapidamente novas variantes que possam escapar da imunidade conferida pelas vacinas atuais e ajustar as estratégias de saúde pública conforme necessário (Grubaugh *et al.*, 2020).

A estrutura de um vírus é essencial para sua função e patogenicidade. O capsídeo proteico que envolve o material genético serve como uma proteção contra degradação e auxilia no reconhecimento e na infecção das células hospedeiras. Alguns vírus possuem um envelope lipídico derivado da membrana celular do hospedeiro, o que pode ajudar na evasão do sistema imunológico do hospedeiro (Howley; Knipe; Enquist, 2021).

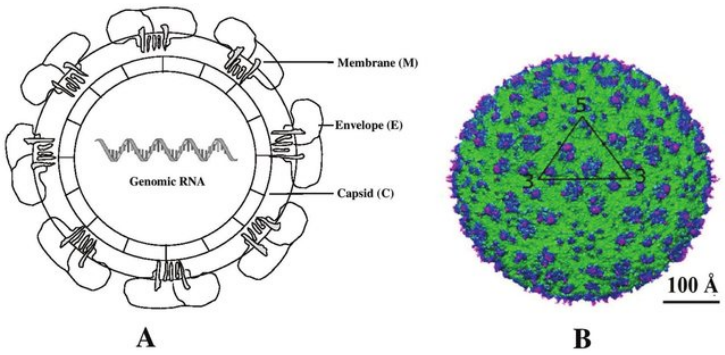
Além disso, os vírus possuem mecanismos sofisticados para entrar nas células hospedeiras. Eles se ligam a receptores específicos na superfície da célula, o que desencadeia a entrada do vírus ou do seu material genético na célula. Uma vez dentro, o vírus usa a maquinaria celular para replicar seu genoma e produzir novas partículas virais, que são

então liberadas para infectar novas células (Wagner *et al.*, 2021).

2.1.1 Vírus da Dengue

A Dengue, causada por um vírus flavivírus, é uma doença viral transmitida por mosquitos do gênero *Aedes*, principalmente *Aedes aegypti*. O vírus da dengue, representa um sério problema de saúde pública, afetando entre 390 e 500 milhões de infecções por dengue a cada ano, segundo a Organização Pan-Americana da Saúde (OPAS) (Saúde, 2023). Mais de 125 países em regiões tropicais e subtropicais estão em risco, com maior prevalência na Ásia, América Latina e Caribe. Aproximadamente 20.000 mortes anuais por dengue grave, principalmente entre crianças e adolescentes. A Dengue tem também um impacto econômico significativo, com custos diretos e indiretos relacionados à hospitalização, perda de produtividade e impacto no turismo.

Figura 1 – Estrutura do vírus da Dengue



Fonte: wikipedia

Na figura 1 mostramos a estrutura do vírus da Dengue. Um capsídeo esférico protege o RNA genômico. No capsídeo as proteínas E e M se juntam para formar estruturas que se fixam na membrana celular das células alvo, e são responsáveis pela resposta imune do organismo. As mutações no gene E estão especialmente associadas ao escape do sistema imunológico, o que o torna o gene mais frequentemente monitorado nos estudos sobre a evolução do vírus da Dengue (Gibson *et al.*, 2015). Por esta razão a genotipagem do gene E do vírus da Dengue é crucial para monitorar a circulação de diferentes sorotipos e para o desenvolvimento de vacinas e medicamentos mais eficazes (Saúde, 2023).

2.2 Pipeline de Sequenciamento Genético: Da Amostra ao Depósito da Sequência na Base de Dados

Nesta seção descrevemos o pipeline de sequenciamento, desde a extração do material genético até a anotação das sequências no GenBank (Metzker, 2010), que é fundamental para ter uma visão holística do contexto da pesquisa realizada. Segundo (Chen; Zhou, 2020)

o pipeline, consta dos processos In-vitro (Experimentos físicos e químicos realizados em laboratórios), In-silico (Análises e simulações computacionais baseadas em dados obtidos experimentalmente) seguindo as seguintes etapas: :

Laboratório Molhado (in-vitro):

1. **Obtenção da Amostra**:
   - Tipo de Amostra: A escolha da amostra depende do vírus em estudo, podendo ser sangue, tecido, secreções respiratórias, entre outras.
   - Coleta da Amostra: A coleta deve seguir protocolos rigorosos para garantir a integridade do material genético.
   - Armazenamento da Amostra: A amostra deve ser armazenada em condições adequadas para preservar o material genético.
2. **Extração de Ácido Nucleico**:
   - Método de Extração: A escolha do método depende do tipo de amostra e do vírus em estudo. Métodos comuns incluem extração por fenol-clorofórmio e kits comerciais.
   - Purificação do Ácido Nucleico: O ácido nucleico extraído deve ser purificado para remover contaminantes.
3. **Síntese de cDNA(Wacker *et al.*, 2002)**:
   - Transcrição Reversa: Se o material genético for RNA, a transcrição reversa é necessária para convertê-lo em cDNA (DNA complementar).
   - Síntese de cDNA: A síntese de cDNA é realizada usando a enzima transcriptase reversa e primers específicos.
4. **Amplificação por PCR (Mullis; Faloona, 1987)**:
   - Design de Primers: Primers específicos para o genoma viral são projetados para amplificar a região de interesse.
   - Reação de PCR: A reação de PCR amplifica o DNA ou cDNA viral, gerando múltiplas cópias da região de interesse.
5. **Purificação do Produto de PCR**:
   - Métodos de Purificação: O produto de PCR é purificado para remover primers, reagentes do PCR (dNTPs) e outros contaminantes.
   - Colunas de Separação: Colunas de centrifugação ou kits de purificação por sílica são comumente utilizados.

6. **Sequenciamento de DNA**:

   - Tecnologia de Sequenciamento: Diversas tecnologias de sequenciamento estão disponíveis, como o método tradicional de Sanger (Sanger; Nicklen; Coulson, 1977) e as tecnologias de NGS (*Next Generation Sequencing*), um termo geral que engloba métodos avançados como Illumina (Bentley et al., 2008), Roche 454, Ion Torrent, entre outros (Metzker; L., 2010).
   - Geração de Leitura: O sequenciamento gera leituras curtas ou longas da sequência de DNA.

Laboratório Seco (in-silico):

7. **Montagem e Análise de Sequências**:

   - Montagem de Contigs: As leituras curtas de sequenciamento são montadas em contigs (sequências sintéticas maiores formadas pela sobreposição de fragmentos menores usando técnicas de bioinformática), reconstruindo a sequência completa do genoma viral (Pop; Kosack; Salzberg, 2004).
   - Análise da Qualidade das Sequências: Softwares bioinformáticos são utilizados para analisar a qualidade do sequenciamento, podendo também identificar variantes genéticas, mutações e outras características (Cock *et al.*, 2009).

8. **Anotação e Depósito no GenBank**:

   - Anotação da Sequência: A sequência viral é anotada com informações sobre genes, proteínas e outras características (Boeckmann *et al.*, 2003).
   - Depósito no GenBank: A sequência anotada é submetida ao GenBank, um banco de dados público de sequências de DNA.

Vale a pena destacar que os primeiros 6 passos descritos aqui descrevem o pipeline no laboratório molhado (*in-vitro)* enquanto os outros passos ocorrem no laboratório seco (*in-silico*).

2.3 Técnicas Tradicionais de Bioinformática para o Estudo da Evolução de Sequências Virais

Na seção anterior, exploramos o trajeto que as amostras virais percorrem, desde a coleta até o depósito de suas sequências genômicas no GenBank. Agora, abordamos o método tradicional de estudar a evolução viral usando essas sequências, que é baseado na filogenia molecular.

A filogenia consiste no estudo das relações evolutivas entre organismos e é uma ferramenta fundamental na virologia para entender como os vírus evoluem e se diversificam. Técnicas filogenéticas são usadas para traçar a origem e a propagação de diferentes cepas virais, permitindo a identificação de linhagens emergentes e a previsão de possíveis surtos futuros (Felsenstein, 1985). No contexto do SARS-CoV-2, por exemplo, a análise filogenética tem sido usada para monitorar a emergência de novas variantes e suas implicações epidemiológicas (Rambaut *et al.*, 2020a).

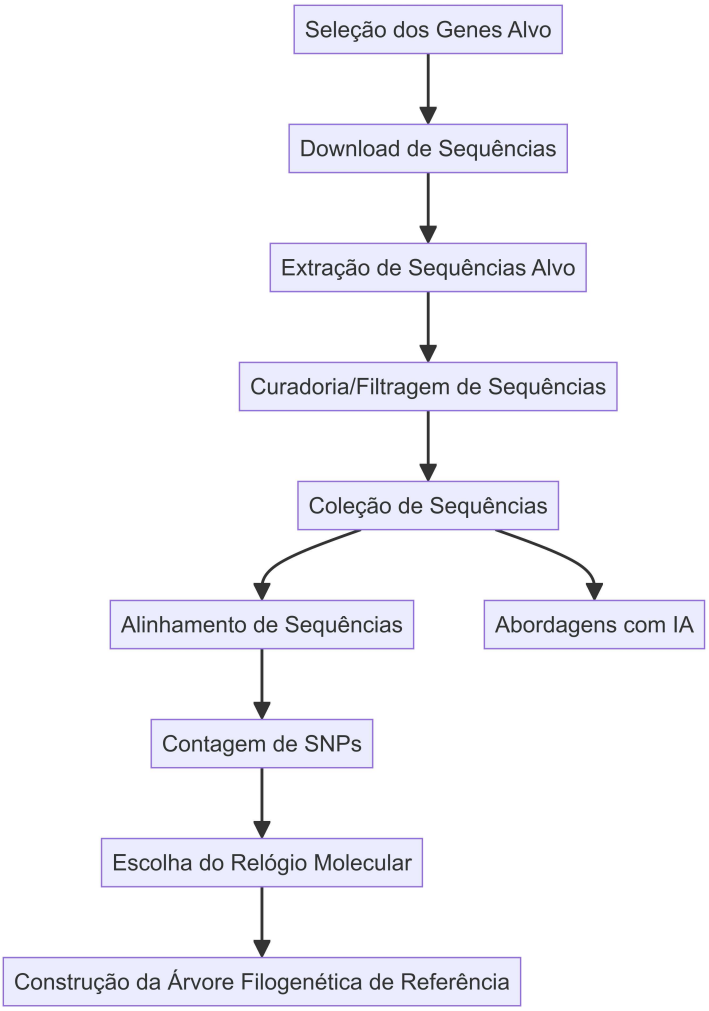
Com um vasto acervo de sequências à disposição, podemos realizar análises em larga escala, abrangendo diferentes regiões geográficas e um número expressivo de cepas virais. Essa abordagem poderosa nos permite:

1. **Identificar novos genótipos**: Através da seleção de genes específicos, extração de sequências, alinhamentos precisos, curadoria meticulosa e construção de árvores filogenéticas robustas baseadas na contagem de SNPs (Polimorfismo de Nucleotídeo Simples (Brookes, 1999)) e no relógio molecular (Kumar *et al.*, 2017), podemos identificar novas variantes virais, expandindo nosso conhecimento sobre a diversidade viral.
2. **Traçar redes de transmissão**: As árvores filogenéticas, quando combinadas com dados epidemiológicos e de geolocalização, revelam as rotas de transmissão viral, permitindo o rastreamento de surtos e a implementação de medidas de controle mais eficazes.

O estudo da evolução viral em grande escala exige um pipeline bioinformático robusto e bem estruturado, composto por etapas cruciais descritas a seguir:

1. **Seleção do(s) gene(s)**: A escolha do gene alvo depende do objetivo específico do estudo. Genes com alta taxa de mutação, como aqueles que codificam proteínas estruturais, são mais propensos a revelar eventos de diversificação viral. Contudo, genes que produzem proteínas expostas são mais relevantes desde o ponto de vista de infectividade e evasão do sistema imunológico,
2. **Extração de sequências**: As sequências de interesse são extraídas do GenBank ou de outras bases de dados relevantes, considerando critérios rigorosos de qualidade e confiabilidade.
3. **Alinhamento de sequências**: Alinhamento de sequências: Algoritmos de alinhamento precisos, como MAFFT (Katoh; Standley, 2013), Clustal Omega (Sievers *et al.*, 2011) e MINIMAP (Li, 2016), garantem a sobreposição correta das sequências, minimizando erros e maximizando a acurácia das análises subsequentes.
4. **Curadoria de sequências**: A curadoria manual ou automatizada remove sequências de baixa qualidade, duplicadas ou contaminadas, garantindo a confiabilidade do conjunto de dados.

Figura 2 – Pipeline Típico para a Construção da Árvore de Referência Filogenética para Genotipagem Viral



Fonte: Adaptada de (Menezes, 2023)

5. **Análise de SNPs e relógio molecular**: A contagem de SNPs e a aplicação do relógio molecular permitem estimar taxas de mutação e divergência viral, inferindo datas de eventos evolutivos e padrões de dispersão geográfica.

6. **Construção de árvores filogenéticas**: Diversos métodos, como *Maximum Likelihood* (ML) (Saitou; Nei, 2013) ou *Neighbor-Joining* (NJ) (Saitou; Nei, 1987), são utilizados para construir árvores filogenéticas robustas, revelando as relações evolutivas entre as cepas virais.

A identificação do genótipo de novas sequências exige a comparação com sequências previamente conhecidas. Para isso, é construída uma árvore filogenética de referência, conforme ilustrado na figura 2, e descrito na seção 2.3.1.

2.3.1 Árvores Filogenéticas de Referência

As árvores filogenéticas de referência, construídas a partir de um conjunto abrangente de sequências virais, servem como ferramentas valiosas para genotipar novas cepas sequenciadas. Ao posicionar uma nova sequência na árvore, podemos determinar seu genótipo e inferir sua relação evolutiva com outras cepas conhecidas. Essa abordagem é crucial para a vigilância viral, permitindo a rápida identificação e caracterização de novas variantes que podem representar riscos à saúde pública.

Além do GenBank, outras bases de dados como o ViPR (Pickett *et al.*, 2012), o BV-BRC (Olson *et al.*, 2023) e o Nextstrain (Hadfield *et al.*, 2018) fornecem acesso a sequências virais e ferramentas bioinformáticas para análise da evolução viral. Diversos softwares e plataformas online, como BEAST (Suchard *et al.*, 2018), PhyloSuite (Zhang *et al.*, 2020) e Genome Detective (Vilsker *et al.*, 2019), facilitam a construção e análise de árvores filogenéticas e a aplicação de métodos de relógio molecular.

Na figura 2 indicamos um fluxo alternativo depois da **Coleção de Sequências** sob o identificador **Abordagens com IA**. Nas próximas seções nos debruçamos na descrição desse ramo, começando na seção 2.4 com as Técnicas de AM, que são os métodos da IA aplicáveis para a solução do problema posto: genotipagem viral.

2.4 Técnicas de Aprendizado de Máquina na Bioinformática

A Aprendizagem de Máquina (AM), é um campo da Inteligência Artificial (IA) que foca no desenvolvimento de algoritmos capazes de aprender a partir de dados e melhorar seu desempenho em tarefas específicas sem a necessidade de serem explicitamente programados (Goodfellow; Bengio; Courville, 2016). Essa habilidade de aprendizado e adaptação torna a AM uma ferramenta poderosa em diversas áreas, como diagnóstico médico, análise financeira, reconhecimento de imagens e tradução automática, entre outras (Géron; Aurélien, 2019).

Os conjuntos de dados de treinamento consistem em coleções de atributos (*features*) extraídos de amostras do domínio em estudo. O vetor (ou matriz) de atributos que descreve uma amostra é chamado de padrão. Dependendo do objetivo do estudo, as amostras podem ser rotuladas (identificadas ou anotadas) ou não, sendo que, na maioria dos casos, elas são rotuladas.

Uma das tarefas mais comuns em AM é o agrupamento de padrões com base em seu grau de similaridade, problema conhecido como *clusterização*. O principal resultado desse processo é um conjunto de grupos (*clusters*) que podem ou não possuir rótulos.

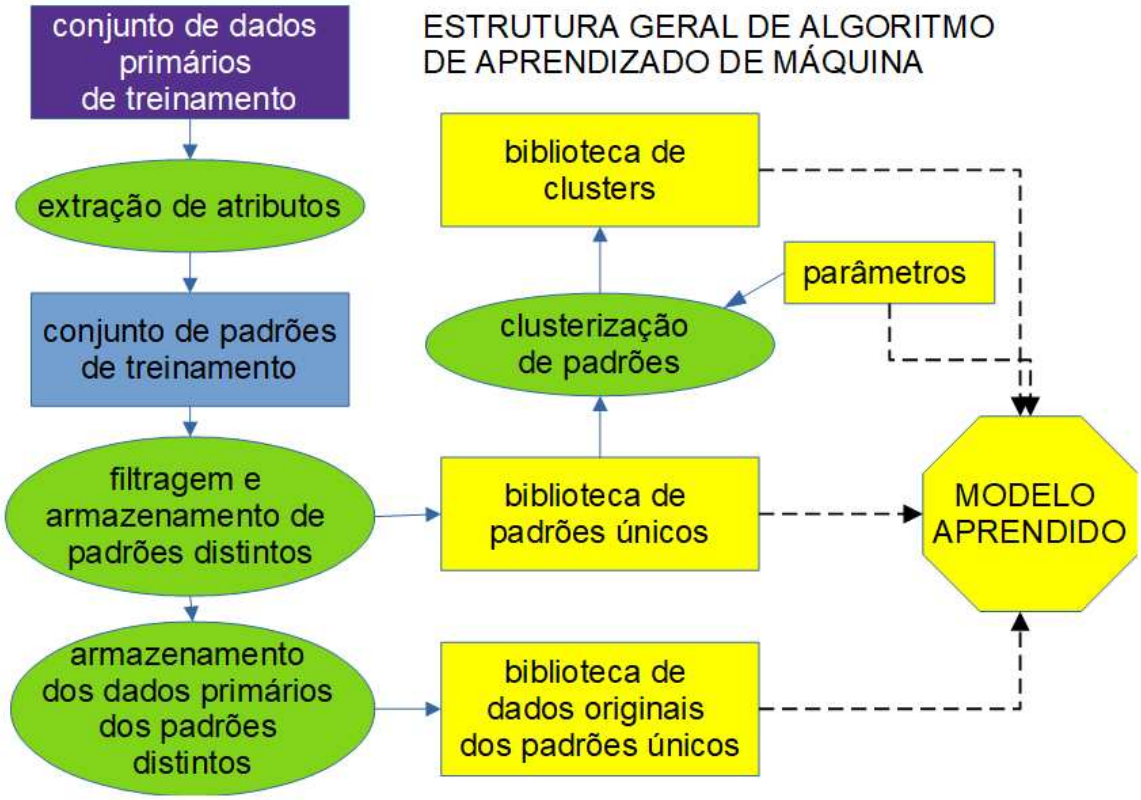
O que chamamos de Modelo Aprendido (MODAP) contém, além da estrutura de *clusters*, o mecanismo (algoritmo) que realiza o mapeamento de um padrão de entrada

a um *cluster* específico, bem como os parâmetros que regulam o funcionamento desse algoritmo.

Adicionalmente, caso o método utilizado inclua um mecanismo de otimização para ajustar os parâmetros do algoritmo de mapeamento, os hiperparâmetros do algoritmo otimizador também são incorporados ao MODAP. É importante ressaltar que, para otimizar os parâmetros do mapeador, é necessário definir métricas de qualidade para os agrupamentos, de modo que os parâmetros obtidos maximizem essas métricas.

Na figura 3 mostramos os componentes e as etapas de um algoritmo geral clássico de treinamento.

Figura 3 – Pipeline genérico de treinamento de um algoritmo de AM.



Fonte: Adaptada de (Menezes, 2023)

O diagrama é intuitivo, ilustrando como um conjunto de dados de treinamento com atributos primários é transformado em um conjunto de padrões formados por atributos específicos. A partir desses padrões, são identificados e coletados os padrões únicos, juntamente com os dados primários associados. Em seguida, o conjunto de padrões únicos é agrupado usando um algoritmo apropriado com um conjunto específico de parâmetros, resultando em um agrupamento registrado em uma biblioteca de *clusters*.

O octágono amarelo representa o MODAP. Os blocos amarelos que o alimentam com

linhas tracejadas são bibliotecas contendo dados originais das amostras únicas selecionadas, seus padrões, *clusters* e os parâmetros do algoritmo de agrupamento utilizado, além do próprio algoritmo (não representado).

O MODAP é posteriormente utilizado para classificar novas amostras com base no padrão identificado[1].

No Apêndice D, detalhamos as etapas de processamento e as bibliotecas que constituem o MODAP da ferramenta *AGUA*, desenvolvida por (Menezes, 2023). Esta ferramenta serve como ponto de partida para o nosso projeto.

2.4.1 Tipos de Aprendizado: Exemplos de Uso na Bioinformática

Segundo (Chapelle; Schölkopf; Zien, 2006) existem 4 tipos de treinamento:

1. **Treinamento Supervisionado:** No treinamento supervisionado, o modelo é construído com base em um conjunto de dados que inclui amostras e suas respectivas etiquetas. O objetivo é que o modelo aprenda a reproduzir essas etiquetas da forma mais fiel possível. Isso é alcançado ajustando os parâmetros do modelo de modo que ele possa prever corretamente a classe ou categoria de novas amostras, baseando-se nas características observadas nas amostras etiquetadas durante o treinamento.

   O treinamento supervisionado é amplamente utilizado em bioinformática quando há disponibilidade de dados anotados. Alguns exemplos incluem:

   - **Classificação de Doenças:**

     Aplicação: Identificação de subtipos de câncer a partir de perfis de expressão gênica.

     Descrição: Os modelos são treinados com dados de expressão gênica onde as amostras são etiquetadas com o subtipo de câncer correspondente. O modelo aprende a prever o subtipo de câncer para novas amostras com base em seus perfis de expressão gênica.

   - **Previsão de Estrutura Proteica:**

     Aplicação: Previsão de estruturas secundárias de proteínas a partir de sequências de aminoácidos.

     Descrição: Utiliza-se um conjunto de dados de proteínas com sequências conhecidas e suas estruturas secundárias anotadas para treinar o modelo a prever a estrutura de novas sequências.

[1] O leitor não familiarizado com as técnicas de AM pode consultar no apêndice A uma relação detalhada de termos relevantes nesta área do conhecimento.

Trabalhos correlatos: (Alpaydin, 2020) descrevem o uso de redes neurais convolucionais (CNNs) para a previsão de estruturas de proteínas, permitindo uma análise mais precisa em comparação com métodos tradicionais baseados em regras, enquanto (Bonetta; Valentino, 2020) apresentam o uso dos algoritmos SVM e *k-nearest neighbor* (k-NN) para classificar proteínas em diferentes famílias e prever suas funções biológicas.

- **Identificação de Genes:**

  Aplicação: Identificação de genes de interesse em estudos de associação genômica ampla (GWAS) (Gallagher; Chen-Plotkin, 2018).

  Descrição: Os modelos são treinados para prever a presença ou ausência de genes associados a doenças com base em variáveis genéticas conhecidas e associadas a essas doenças.

- **Identificação de Genes Associados a Doenças**

  Aplicação: Descoberta de genes provavelmente associados a certas doenças.

  Descrição: Os modelos são treinados para descobrir associações entre o padrão de expressão gênica em células/pacientes doentes e saudáveis, e assim identificar prováveis genes de interesse.

  Trabalho Correlato: (Li *et al.*, 2019) demonstraram como algoritmos de aprendizado supervisionado, como redes neurais e SVM, podem ser usados para identificar variantes genéticas associadas a doenças específicas, como câncer e doenças genéticas raras.

- **Análise de Variantes Genéticas (Filogenia)**:

  Aplicação: Classificação de variantes genéticas.

  Descrição: A classificação é realizada mediante a identificação de SNPs.

  Trabalho Correlato: (Wang *et al.*, 2023) exploram como as redes neurais podem ser aplicadas para inferir relações filogenéticas entre diferentes espécies a partir de sequências genéticas.

2. **Treinamento Não-Supervisionado:** No treinamento não-supervisionado, não são fornecidas etiquetas para as amostras. Em vez disso, o modelo busca identificar padrões, estruturas ou agrupamentos dentro dos dados com base em características intrínsecas das amostras. O objetivo é descobrir relações ou agrupamentos naturais dentro do conjunto de dados, sem qualquer orientação externa. Um exemplo comum de treinamento não-supervisionado é a análise de clusters, onde o modelo agrupa amostras semelhantes sem conhecer previamente suas categorias.

   O treinamento não-supervisionado é útil em bioinformática quando se lida com dados não anotados ou quando se deseja descobrir padrões desconhecidos. Exemplos incluem:

- **Clusterização de Dados de Expressão Gênica:**

  Aplicação: Descoberta de novos grupos de genes co-expressos ou de novos tipos de células.

  Descrição: Algoritmos de clusterização, como k-means (Ahmed; Seraj; Islam, 2020) ou análise de componentes principais (PCA) (Karamizadeh *et al.*, 2013), são usados para agrupar genes ou células com perfis de expressão similares sem usar etiquetas.

- **Análise de Variação Genética:**

  Aplicação: Identificação de subpopulações dentro de uma população com base em variações genéticas.

  Descrição: Técnicas como análise de componentes principais (PCA) (Karamizadeh *et al.*, 2013) ou t-SNE (Platzer, 2013) são usadas para reduzir a dimensionalidade dos dados e visualizar a estrutura da população sem etiquetas prévias.

  Trabalho Correlato: (Roman-Naranjo; Parra-Perez; Lopez-Escamez, 2023)) utilizaram algoritmos de agrupamento não-supervisionados para agrupar genes com base em características semelhantes, ajudando a identificar biomarcadores e padrões de expressão associados a doenças

- **Descoberta de Motivos de Sequência:**

  Aplicação: Identificação de motivos de DNA ou RNA que são conservados em um conjunto de sequências.

  Descrição: Algoritmos como MEME (Multiple EM for Motif Elicitation) (Bailey *et al.*, 2006) são usados para encontrar padrões de sequências conservadas sem a necessidade de etiquetas.

3. **Treinamento Auto-Supervisionado (Self-Supervised):** O treinamento auto-supervisionado é um tipo de treinamento não-supervisionado onde o modelo gera suas próprias etiquetas com base em partes do dado não etiquetado. Nesse tipo de treinamento, o modelo pode usar informações internas dos dados para criar pseudo-etiquetas, que são então usadas para treinar o modelo. Este método permite que o modelo aprenda de maneira supervisionada usando dados não etiquetados.

   O treinamento auto-supervisionado é vantajoso quando há uma grande quantidade de dados não anotados e se deseja aproveitar esses dados para treinar modelos. Exemplos incluem:

   - **Previsão de Interações Proteína-Proteína:**

     Aplicação: Prever interações entre proteínas a partir de sequências de aminoácidos.

Descrição: Os modelos podem ser treinados para prever partes faltantes de sequências de proteínas ou para gerar representações de proteínas que são usadas posteriormente para prever interações.

- **Análise de Sequências Genômicas:**

  Aplicação: Prever elementos funcionais no genoma, como *enhancers* ou *silencers*.

  Descrição: Os modelos são treinados com tarefas preditivas, como a predição de segmentos faltantes de sequências genômicas, que ajudam a modelar a estrutura e função do DNA sem etiquetas explícitas.

- **Representações de Dados Ômicos:**

  Aplicação: Aprender representações de dados multi-ômicos (genômica, transcriptômica, proteômica) que podem ser usadas em várias tarefas subsequentes.

  Descrição: Os modelos são usados para aprender *embeddings* que capturam informações relevantes dos dados ômicos, que podem então ser aplicados em tarefas supervisionadas ou não supervisionadas.

4. **Treinamento Semi-Supervisionado:** O treinamento semi-supervisionado ou não-supervisionado guiado, é uma abordagem híbrida. Embora as etiquetas das amostras não sejam usadas diretamente para treinar o modelo, elas são utilizadas para avaliar a qualidade dos agrupamentos ou clusters formados pelo modelo. A pureza dos clusters, que mede o grau de homogeneidade dos clusters em relação às etiquetas reais, é uma métrica importante nesse contexto. Essa avaliação é então usada para otimizar os parâmetros do modelo, orientando o processo de treinamento para melhorar a formação dos clusters. Esse tipo de treinamento busca um equilíbrio entre a descoberta de padrões intrínsecos nos dados e a utilização de informações de etiquetas para refinar esses padrões.

   O treinamento semi-supervisionado é útil quando há uma combinação de dados anotados e não anotados. Ele permite que o modelo aproveite as etiquetas disponíveis para guiar o aprendizado e, ao mesmo tempo, utilize o grande volume de dados não anotados para melhorar a performance. Alguns exemplos incluem:

   - **Anotação Funcional de Genes:**

     Aplicação: Anotação de funções gênicas em novos genomas.

     Descrição: Utiliza um pequeno conjunto de genes anotados com funções conhecidas e um grande conjunto de genes não anotados. O modelo aprende a partir dos genes anotados e generaliza esse conhecimento para prever as funções dos genes não anotados, utilizando informações adicionais dos dados não anotados para melhorar a precisão.

   - **Identificação de Regiões Funcionais no Genoma:**

Aplicação: Descoberta de *enhancers* e outras regiões regulatórias.

Descrição: Usa um conjunto de regiões genômicas com funções conhecidas (etiquetadas) e um grande conjunto de regiões não caracterizadas (não etiquetadas). O modelo aprende a partir das regiões conhecidas e utiliza esse conhecimento para prever as funções das regiões não caracterizadas, aproveitando os dados não anotados para melhorar a robustez das predições.

Essas distinções são fundamentais para entender como diferentes abordagens de treinamento podem ser aplicadas em cenários variados, cada uma com suas próprias vantagens e desafios. A abordagem adotada neste projeto é o Treinamento Semi-Supervisionado.

2.4.2 Desafios e Limitações

Embora o AM tenha trazido muitos avanços para a bioinformática, existem desafios significativos a serem superados. O primeiro é a necessidade de grandes volumes de dados rotulados para treinar modelos de aprendizado supervisionado, o que pode ser difícil de obter em algumas áreas. Além disso, a interpretabilidade dos modelos de AM é um problema importante, especialmente em bioinformática, onde os resultados podem ter implicações para a saúde e o tratamento de doenças. (Zhou *et al.*, 2019) discutem como técnicas de explicabilidade de modelos, como SHAP (Shapley Additive Explanations), podem ajudar a melhorar a transparência dos modelos de AM na bioinformática.

Podemos concluir que o AM desempenha um papel fundamental na bioinformática, oferecendo novas maneiras de analisar e interpretar grandes volumes de dados biológicos. No entanto, existem desafios técnicos e éticos que ainda precisam ser abordados para garantir que as ferramentas de AM sejam utilizadas de forma eficaz e responsável.

2.5 Aprendizado de máquina para Genotipagem

Na revisão da literatura, identificamos duas abordagens distintas de Aprendizagem de Máquina (AM) aplicadas à genotipagem viral: (1) aquelas baseadas em dados clínicos e exames laboratoriais de amostras contendo o vírus (sem sequenciamento) e (2) aquelas que utilizam dados obtidos por sequenciamento do material genético do vírus.

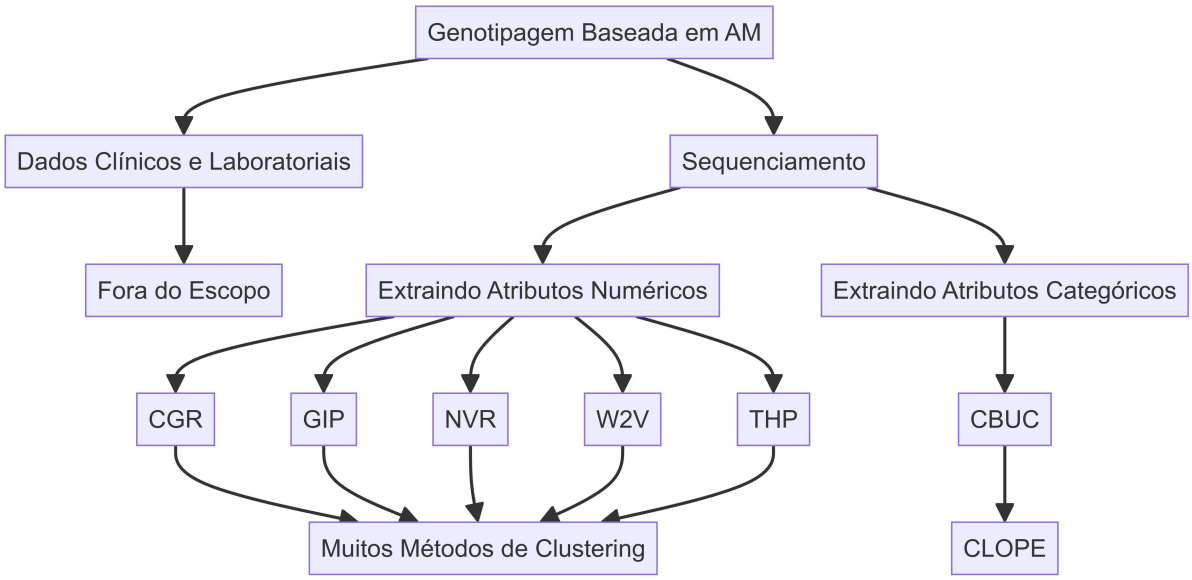
A primeira abordagem está fora do escopo deste trabalho, por isso o foco será na segunda, que utiliza a sequência genética do vírus. Dentro desta abordagem, identificamos duas estratégias principais: (1) baseadas em atributos numéricos e (2) baseadas em atributos categóricos.

Na abordagem de **atributos numéricos**, encontramos diversas propostas que são detalhadas no apêndice B, para consulta dos leitores interessados.

Por outro lado, para **atributos categóricos**, apenas identificamos a abordagem utilizada neste trabalho.

Na figura 4 representamos de forma esquemática os resultados da pesquisa da diversidade de abordagens para genotipagem. O fluxo situado à extrema direita nesse diagrama corresponde à abordagem desenvolvida neste estudo.

Figura 4 – Diferentes abordagens do uso de AM para genotipagem de vírus.



Fonte: O Autor

Na seção 2.5.1, apresentamos: (a) o método adotado para a extração de atributos categóricos a partir das sequências e (b) o algoritmo utilizado para a clusterização das sequências representadas como listas de atributos categóricos, baseado no método CLOPE.

2.5.1 Extração de Atributos Categóricos das Sequências de DNA

As sequências de DNA são representadas como cadeias de caracteres, onde cada posição corresponde a um nucleotídeo. Os nucleotídeos básicos são representados pelos caracteres A, G, C e T, que correspondem às bases nitrogenadas Adenina, Guanina, Citosina e Tiamina, respectivamente. No entanto, outros caracteres são introduzidos para representar deleções (doravante denominadas *gaps*) e combinações possíveis de nucleotídeos em sítios polimórficos. Esses caracteres estão especificados no código IUPAC (Johnson, 2010).

É importante ressaltar que as técnicas tradicionais de genotipagem, baseadas em árvores filogenéticas, utilizam essa representação básica das sequências de DNA. Em contrapartida, nossa abordagem, apresentada em (Menezes, 2023) e denominada *CBUC*

(Codon Usage Based Unsupervised Classification), adota como **atributos primários** os códons em cada posição das sequências codificantes, os quais codificam as proteínas virais de interesse para a genotipagem.

A abordagem utilizada baseia-se em duas considerações principais:

1. **A diversidade da realidade é melhor descrita quanto maior a variabilidade dos atributos independentes considerados para avaliar os sujeitos da população estudada:**

   Considere um atributo a_1 que possui $n_1 > 1$ variantes. As amostras serão classificadas em n_1 categorias ou classes.

   Agora, considere outro atributo a_2 que possui $n_2 > n_1$ variantes. É evidente que as amostras serão classificadas em um número maior de categorias ou classes, permitindo uma descrição mais detalhada da realidade se usarmos o atributo a_2 em vez do a_1.

   No contexto das sequências de DNA, existem apenas 4 nucleotídeos e 20 aminoácidos, mas há 61 códons codificantes (excluindo os 3 códons de parada). Portanto, os códons descrevem melhor a diversidade do que os nucleotídeos e os aminoácidos.

2. **As mutações em regiões não codificantes dos vírus não produzem nenhuma mudança fenotípica/funcional que tenha influência na genotipagem:**

   Por isso, na genotipagem viral, faz sentido considerar apenas as regiões codificantes (CDS/ORF) do genoma dos vírus estudados[2].

Além das considerações anteriores, o uso de códons para descrever as sequências tem as seguintes vantagens:

- **Maior compactação:** O número de atributos por amostra é reduzido em 3 vezes em relação ao número de nucleotídeos.
- **Reversibilidade:** A sequência original da amostra pode ser gerada sem ambiguidade a partir da sequência de atributos (códons) gerada na transformação de DNA para a sequência de atributos categóricos. Isso não seria possível representando a sequência pelos aminoácidos. Embora essa representação tenha o mesmo grau de compactação que a de códons, ela não é reversível devido à redundância do código genético.

Desvantagem:

Para trabalhar com códons como atributos, todas as sequências codificantes alvo (não necessariamente completas) precisam estar alinhadas no **quadro de leitura** $5' - 3'$ para permitir a identificação correta dos códons.

[2] Isso não significa que mutações nas regiões não codificantes não sejam úteis no estudo de vírus. De fato, elas são relevantes para a análise de descendência e no rastreamento crono-geográfico de variantes.

As técnicas filogenéticas também precisam ter as sequências alinhadas, mas não necessariamente no quadro de leitura, o que significa que o uso de códons introduz mais uma exigência durante o pré-processamento das sequências.

O alinhamento múltiplo de sequências tem um alto custo computacional, por isso diversos métodos têm sido desenvolvidos nos últimos anos (Chao; Tang; Xu, 2022). Convenientemente, para a tarefa de genotipagem, pode ser usado o alinhamento contra uma sequência de referência, o que transforma a complexidade computacional em linear.

Construção do Padrão: O *CBUC* utiliza apenas os códons presentes nos sítios polimórficos das sequências de entrada para construir o padrão, ou seja, nos sítios onde aparecem mais de um códon no alinhamento. Por isso, a lista de sítios polimórficos é armazenada no modelo de dados, permitindo que um padrão consistente seja construído a partir de uma nova sequência.

No Apêndice C, detalhamos os algoritmos utilizados para transformar sequências de nucleotídeos em sequências de códons, identificar os sítios polimórficos e construir os padrões.

2.5.2 Método de Classificação Não-Supervisionado CLOPE

CLOPE (*Clustering with sLOPE*) é um algoritmo de clusterização projetado especificamente para dados categóricos. Ele se destaca por sua simplicidade e eficiência em termos de custo computacional e é particularmente útil quando se trabalha com grandes conjuntos de dados categóricos.

A entrada é um conjunto de transações, onde cada transação é composta por uma lista não ordenada e de tamanho variável de *itens* definidos nominalmente, por exemplo: $manga$, $coca-cola-2L$ e a quantidade de cada um: $transação$ =[5:manga,1:coca-cola-2L], que indica que essa transação teve 5 mangas e uma garrafa de coca-cola de 2 litros.

No nosso caso, os itens são códons, pelo que temos apenas 64 deles (mais os códons não identificados e os gaps - veja apêndice A), e substituímos a quantidade pela posição na sequência, ou seja, nossos atributos têm a forma ($posicão$: $códon$), onde a $posição$ está ordenada de menor a maior[3]. Ou seja nossas transações são da forma $t = [(p : A_p),\ p = 1, 2, ..., P] \in T$, representando uma sequência do conjunto de treinamento, onde A_p é o numeral do códon na posição polimórfica p nessa sequência[4].

O método possui um único parâmetro, chamado de repulsão, r, o qual controla o compromisso entre a compactação intra-cluster e a separação inter-cluster. É crucial para determinar a formação dos clusters. Valores baixos de r favorecem clusters mais compactos,

[3] por conveniência para a utilização posterior do modelo com sequências parciais - fora do escopo deste estudo

[4] A construção do conjunto de treinamento T foi descrita na seção 2.5.1

enquanto valores altos de r favorecem clusters mais separados.

Uma vez definida a repulsão r o método realiza agrupamentos aleatórios até que encontra um agrupamento que maximiza uma função de lucro que considera tanto a compactação dos clusters q anto a separação entre eles. Ou seja, a função de lucro aumenta quando os clusters são compactos e afastados dos outros.

Embora este método tenha sido implementado em (Menezes, 2023), nem o algoritmo de treinamento nem o modelo matemático foram descritos em detalhes. Por isso, fornecemos uma descrição mais detalhada do processo de treinamento no Apêndice E e do modelo matemático no Apêndice F.

2.6 Estrutura e Funcionamento de Ferramentas de Bioinformática na web

Embora a estrutura de grande parte das ferramentas web de bioinformática atualmente em uso não seja amplamente divulgada, é possível inferir, com base no modo como operam e nos serviços que oferecem, uma ideia aproximada de sua arquitetura e dos componentes utilizados em sua construção. Ao mencionar a arquitetura, nos referimos às diversas camadas que a compõem, como a camada de dados persistentes, *middle* e *front-end*, *backend*, gerenciamento de *jobs*, comunicação com o usuário, entre outros elementos essenciais.

Essas ferramentas geralmente utilizam uma arquitetura modular para garantir escalabilidade, desempenho e facilidade de manutenção. A camada de dados persistentes é responsável pelo armazenamento de informações, como bancos de dados relacionais ou não relacionais, onde são mantidos dados genômicos, resultados de análises e *logs* de processamento.

O *backend*, por sua vez, funciona como o núcleo do sistema, lidando com o processamento das requisições do usuário, execução de algoritmos e integração com bibliotecas ou ferramentas externas, como *pipelines* de bioinformática. Ele também gerencia a alocação de recursos computacionais para garantir que os *jobs* sejam executados de forma eficiente.

A camada *middle* atua como um intermediário entre o *backend* e o *front-end*, facilitando a comunicação, formatando dados e garantindo a segurança e autenticidade das interações. Já o *front-end* é a interface com o usuário, geralmente desenvolvida com tecnologias modernas para garantir usabilidade, acessibilidade e uma experiência fluida.

O gerenciamento de *jobs* é um componente crítico em ferramentas de bioinformática, especialmente quando múltiplos usuários estão enviando análises simultâneas. Essa funcionalidade garante a priorização, escalonamento e monitoramento das tarefas, além de fornecer *feedback* ao usuário sobre o progresso de seus *jobs*.

Finalmente, a comunicação com o usuário inclui notificações, *logs* de execução e relatórios detalhados, garantindo transparência e permitindo que o usuário interprete e utilize os resultados de maneira eficaz.

Essa arquitetura integrada reflete as melhores práticas no desenvolvimento de ferramentas web modernas voltadas para aplicações científicas. Baseado nisto fornecemos um resumo das funções de cada componente seguindo o padrão WEB 2.0 (Zhang; Cheung; Townsend, 2008):

- **Gerenciamento de Usuários e Filas**:
  - Controle de acesso.
  - Priorização de tarefas.
  - Notificações de status e conclusão.
- **Acesso a Dados**:
  - Integração com bancos de dados locais e remotos.
  - Garantia de segurança e eficiência no acesso aos dados.
- **Camadas de Processamento**:
  - Backend poderoso para cálculos pesados.
  - Camada intermediária para gerenciamento de filas e comunicação.
  - Frontend amigável para interação com o usuário.
- **Comunicação Robusta**:
  - Comunicação assíncrona entre frontend e backend para lidar com tarefas longas.
- **Notificação de Resultados**:
  - Envio de emails com links para resultados ou anexos com os dados finais.

2.6.1 Tecnologias Mais Utilizadas

- **Linguagens de Programação**:

  Backend: Python, R, Java, C++(Cock *et al.*, 2009)

  Frontend: HTML5, CSS, JavaScript, python, e Jinja(Python, 2023);(Mozilla Developer Network, 2023).

- Frameworks:

  Backend: Django (Python)(Django Software Foundation, 2023), Flask(Flask, 2023)

- **Gerenciamento de Filas**:

  Celery(Python)(Celery Project, 2024),RabbitMQ (mensagens)(RabbitMQ, 2024).

- **Softwares para Alinhamento de Sequências**:

  BLAST(Altschul *et al.*, 1997); Clustal Omega, SAMtools, GATK(Sievers *et al.*, 2011)

2.6.2 Sistemas WEB para Genotipagem Viral: Uma Revisão.

A genotipagem viral desempenha um papel fundamental na vigilância epidemiológica, no diagnóstico e no controle de doenças infecciosas. A rápida evolução de vírus como o SARS-CoV-2 demanda métodos ágeis e eficientes para identificar mutações genéticas e monitorar a disseminação de variantes. Com a crescente disponibilidade de dados genômicos virais e os avanços nas tecnologias de sequenciamento, tem-se investido no desenvolvimento de sistemas web para facilitar a genotipagem viral, tornando essas ferramentas mais acessíveis a pesquisadores e profissionais de saúde.

Plataformas como o Nextstrain (Rambaut *et al.*, 2020b) e o GISAID(Shu; McCauley, 2017) são exemplos notáveis de sistemas web que possibilitam o rastreamento de variantes virais, como o SARS-CoV-2, e a visualização de dados genômicos em tempo real. Estudos como (Rambaut *et al.*, 2020b) e (Vilsker *et al.*, 2018) destacam a relevância do Nextstrain e do Genome Detective no acompanhamento da evolução do SARS-CoV-2, permitindo a identificação e o rastreamento rápidos de mutações.

Adicionalmente, ferramentas como o COV-GLUE (Singer *et al.*, 2020) oferecem funcionalidades para integrar dados de sequenciamento viral com informações sobre disseminação geográfica e desfechos clínicos de pacientes. Isso auxilia na análise de padrões de mutação e evolução viral, proporcionando insights relevantes para a pesquisa e o controle epidemiológico.

Por fim, trabalhos como o de (Gonzalez *et al.*, 2021) discutem a importância da implementação de criptografia e protocolos de segurança, como o HTTPS, para assegurar a proteção dos dados compartilhados nessas plataformas. Essa preocupação é essencial para garantir a privacidade e a integridade dos dados utilizados em sistemas web de genotipagem viral.

Embora os sistemas web de genotipagem viral tenham desempenhado um papel fundamental na resposta a pandemias e epidemias, ao fornecer ferramentas acessíveis para a análise de sequências virais e o monitoramento da evolução de variantes, ainda enfrentam desafios significativos. Entre eles, destacam-se a necessidade de escalabilidade para suportar a transferência bidirecional de dados entre o cliente e o servidor web, bem como o processamento eficiente de grandes volumes de dados genômicos em tempo real. Essas limitações comprometem a disponibilidade prática desses sistemas e dificultam sua capacidade de atender à crescente demanda por esses serviços.

3 DESENVOLVIMENTO DO PROJETO

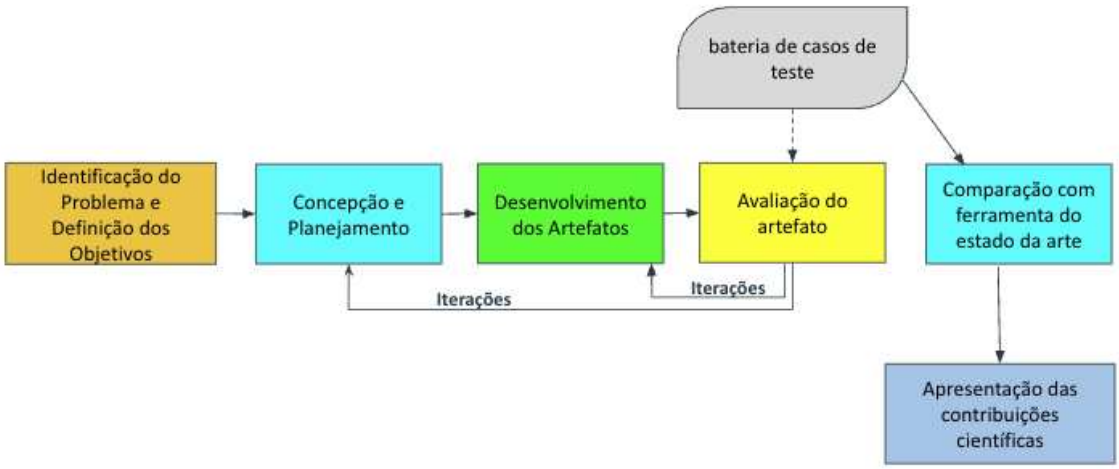
O sistema *AGSSA* (*Advanced Genotyping with Semi-Supervised Algorithm*) é uma plataforma desenvolvida a partir do protótipo da ferramenta *AGUA* (*Advanced Genotyping with Unsupervised Algorithm*) para melhorar o processo de treinamento e possibilitar o acesso à ferramenta via web. Por isso a evolução ocorreu tanto no kernel de cálculo quanto na interface com o usuário.

Metodologia:

A metodologia de pesquisa adotada neste trabalho é a pesquisa aplicada, que visa produzir conhecimento para aplicação prática no desenvolvimento de uma ferramenta para genotipagem viral. A pesquisa exploratória foi utilizada para proporcionar maior familiaridade com o problema e definir claramente os requisitos do sistema (Gil, 2008; Peffers *et al.*, 2007).

Mais especificamente, adotamos a metodologia DSR (*Design Science Research*) (Peffers *et al.*, 2007), que é um processo iterativo composto por uma série de etapas em cada iteração, conforme representado na Figura 5.

Figura 5 – Processo iterativo da metodologia DSR adaptado para o projeto



Fonte: O autor

O passos seguidos, conforme sugerido por (Gregor; Hevner, 2013) e (Peffers *et al.*, 2018) foram:

1. **Identificação do Problema e Definição dos Objetivos:** Nesta fase, o problema a ser resolvido é identificado e compreendido em detalhes, incluindo a compreensão

das limitações das abordagens existentes para a resolução do problema. Os resultados desta fase foram descritos na seções 1.3 e 1.4.

2. **Concepção e Planejamento:** Aqui, são definidos os objetivos do artefato a ser criado, suas características e funcionalidades. No projeto em questão, isso inclui, num primeiro momento, a definição das funcionalidades da ferramenta a ser desenvolvida.

   Para isso, foi feito um **levantamento dos requisitos**, listados a seguir:

   a) A ferramenta deve permitir a atualização/construção, via (re)treinamento, de modelos de classificação para uma ampla variedade de combinações vírus-gene. No entanto, a construção desses modelos deve ser realizada por profissionais experientes e certificados. Para isso, a ferramenta deve incluir um cadastro de usuários habilitados para essa função.

   b) Todos os usuários poderão utilizar o serviço de classificação de sequências sem necessidade de cadastro, sendo exigido apenas o fornecimento de um e-mail para o envio dos resultados. Esses e-mails serão excluídos automaticamente após a confirmação do envio. O usuário deverá selecionar o modelo desejado através da interface e carregar o arquivo contendo as sequências a serem classificadas (genotipadas).

   c) Usuários certificados terão acesso a três tipos de processamento: (1) Treinamento com pré-processamento do conjunto de treinamento, (2) Treinamento sem pré-processamento e (3) Análise de Sequências. Usuários não certificados terão acesso apenas à opção de Análise de Sequências.

   d) Para realizar o (re)treinamento de modelos, os usuários certificados deverão carregar um arquivo contendo as anotações das sequências incluídas no conjunto de treinamento, além do arquivo com as sequências.

   e) Na opção de Treinamento com Pré-processamento, o sistema deve ajustar automaticamente o conjunto de dados carregado pelo usuário para atender ao padrão de qualidade descrito na seção C.1. As sequências removidas por falta de qualidade deverão ser salvas em um arquivo separado e incluídas na pasta de resultados enviada ao usuário.

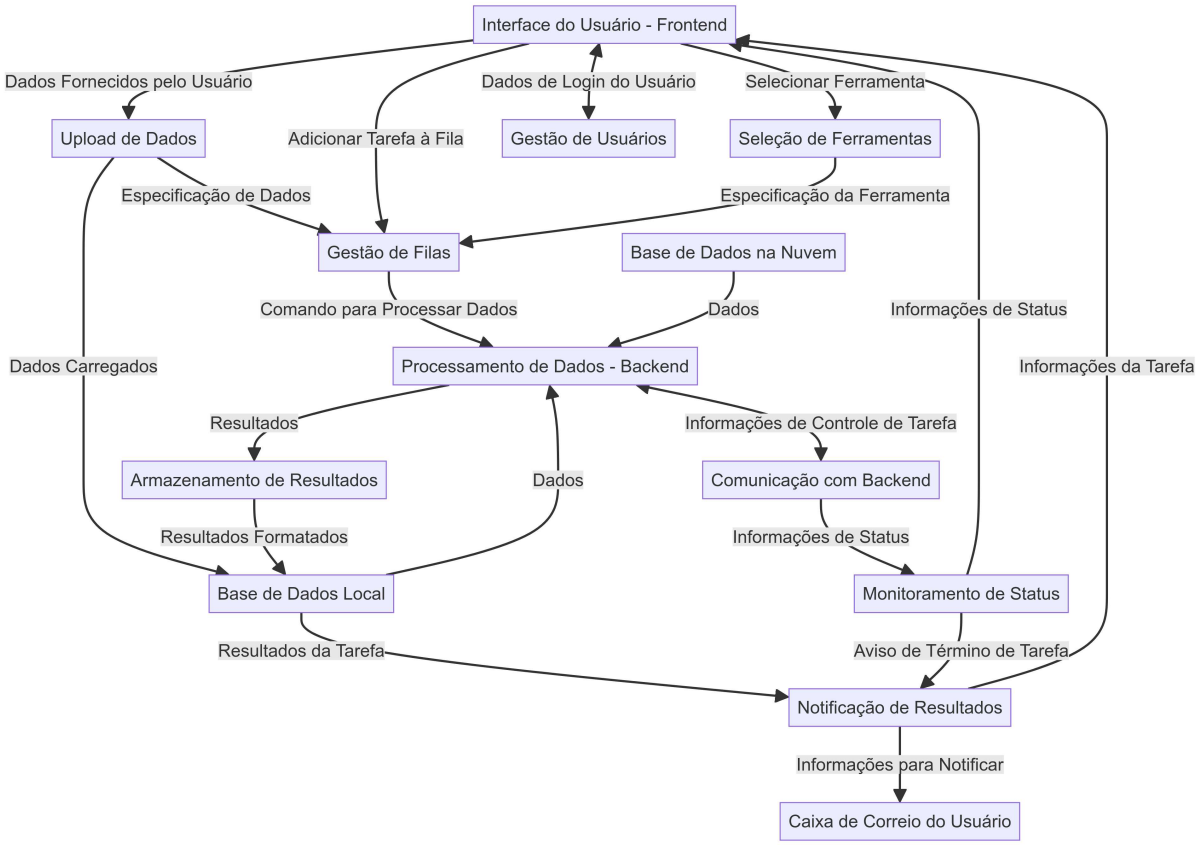
   f) Os resultados devem ser exportados em diversos formatos, permitindo uma visualização posterior mais prática e flexível.

   g) O sistema deve enviar notificações por e-mail, programadas para informar o status do processamento.

   h) A interface da ferramenta deve ser amigável, intuitiva e de fácil navegação.

   Uma vez definidos os requisitos, foi feita a concepção da arquitetura do sistema, mostrada na figura 6.

Figura 6 – Diagrama estrutural e funcional proposto para sistemas web de bioinformática.



Fonte: O autor

Em seguida, como parte do planejamento, foi decidido utilizar os seguintes **ambientes de desenvolvimento**:

- **GitHub**: Utilizado para controle de versionamento e gerenciamento colaborativo do código-fonte, facilitando o rastreamento de alterações e a colaboração entre os desenvolvedores.
- **Visual Studio Code (VS Code)**: Editor de código utilizado para o desenvolvimento do sistema, oferecendo recursos como depuração integrada, controle de versão Git embutido e extensões para várias linguagens de programação.

As **tecnologias escolhidas** para o desenvolvimento de APIs e do frontend do sistema *AGSSA*, foram:

- **HTML5**: Utilizado para a estruturação das páginas.
- **CSS**: Aplicado para estilizar as páginas e garantir uma interface responsiva.
- **Flask**: Framework para desenvolvimento web em Python, utilizado para a integração backend/frontend.
- **Python**: criação dos scripts de pré-processamento.

- **Jinja**: Motor de templates utilizado com Flask para renderização dinâmica das páginas, permitindo a injeção de dados do servidor para o cliente de forma eficiente.

A **interface de usuário** (frontend) do sistema *AGSSA* foi concebida com base em wireframes. Os wireframes foram utilizados para mapear a estrutura e o layout das páginas web, definindo a organização dos elementos na interface e o fluxo de navegação entre as diferentes seções do sistema. Com base na análise desses wireframes, foram desenvolvidas versões preliminares mais detalhadas da interface, na forma de protótipos.

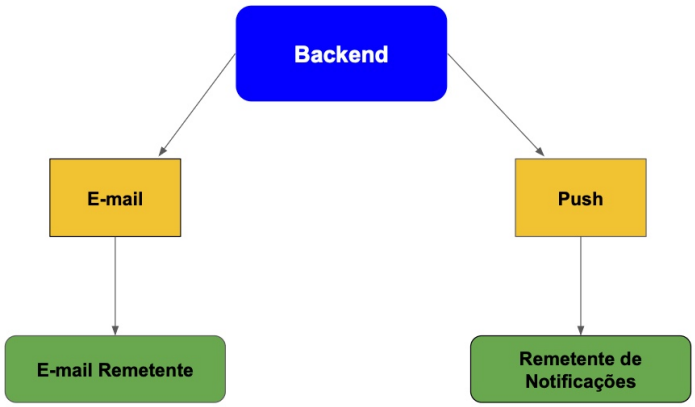
Esses protótipos interativos permitiram testar as funcionalidades do sistema, identificar pontos de melhoria e realizar ajustes necessários antes da implementação final. Essa abordagem garantiu um processo iterativo de refinamento, resultando em uma interface mais eficiente e adaptada às necessidades dos usuários.

Para a **integração frontend-backend** foi selecionado o **Flask**, um framework web para Python, para construir a API. O API em Flask gerencia as requisições HTTP, processa a lógica de negócios e interage com o banco de dados, permitindo uma comunicação eficaz entre o frontend e o backend do sistema.

Para o **processamento assíncrono de tarefas** escolhimos o **Celery**, conhecido por gerenciar a fila de tarefas e distribuir o processamento de dados de maneira eficiente. A arquitetura flexível do Celery suporta múltiplos paradigmas de execução, tornando-o adequado para operações que vão desde simples tarefas em background até complexas aplicações de processamento distribuído. Ele é amplamente adotado em ambientes de produção devido à sua robustez, facilidade de integração e vasta documentação (Celery Project, 2024).

Para a **comunicação assíncrona** entre componentes de software escolhemos o **RabbitMQ** (RabbitMQ, 2024), um broker de mensagens que implementa o protocolo AMQP (Advanced Message Queuing Protocol). O RabbitMQ é crucial para serviços de back-end que precisa enviar notificações aos usuários finais, como é o caso do sistema *AGSSA*. Existem dois canais de notificação: e-mails e notificações push para aplicativo móvel. O backend publica a notificação em duas filas, uma para cada canal. Programas que gerenciam emails e notificações push, como o RabbiMQ, se inscrevem na fila em que estão interessados e lidam com notificações assim que chegam, como mostrado na figura 7.

Figura 7 – Broker de Mensagens



Fonte: Retirada de RabbitMQ (2024)

Como parte da concepção do sistema, se definiu a estrutura do **envio de resultados por email**. Após o treinamento são gerados vários arquivos contendo: 1) o dendrograma gerado utilizando algum método selecionado de clusterização hierárquica no *AGSSA* (método default UPGMA), 2) a matrix de distâncias, 3) o MODAP com todos seus componentes descritos na seção 2.4, 4) um arquivo com a árvore filogenética construída pelo Iqtree2, em formato gráfico e, 5) o log do processamento contendo:

- Data e hora de início
- Quantidade de sequências enviadas
- Quantidade de sequências processadas
- Data e hora de Término

Por último, como parte do planejamento foi estabelecida a **configuração do ambiente** necessário para o desenvolvimento e validação de *AGSSA*. Os componentes necessários foram instalados e configurados para garantir a correta funcionalidade da aplicação. O ambiente é descrito no apêndice I.

3. **Desenvolvimento dos Artefatos:** Nesta etapa, foram desenvolvidos todos os componentes do sistema web, incluindo os modelos computacionais, os pipelines, as interfaces e os conjuntos de dados de treinamento utilizados para testes e validação.

   A versão atual da interface do sistema *AGSSA*, desenvolvida ao longo do projeto, é apresentada na seção 4.1. Já na seção 4.2, detalhamos o pipeline criado para a classificação de sequências, enquanto a seção 4.3 aborda o novo método de treinamento desenvolvido especificamente para a ferramenta *AGSSA*.

4. **Avaliação do artefato:** O artefato, a ferramenta web *AGSSA*, foi testado e avaliado quanto à sua eficácia na resolução do problema. Isto incluiu, testes de funcionalidade da

interface, de qualidade dos resultados e de desempenho temporal, usando indicadores medidos numa bateria de experimentos devidamente configurada. A avaliação incluiu comparações com métodos existentes. Mais especificamente, a avaliação do artefato consistiu em:

- **Teste Funcional**: Teste preliminar das funcionalidades para avaliar a forma como as tarefas de processamento previstas são configuradas e executadas desde a interface. Usamos o método de revisão por terceiros, onde outros desenvolvedores e especialistas próximos avaliam a funcionalidade da aplicação para garantir a qualidade dos resultados e identificar possíveis melhorias. As avaliações foram verbais e não registradas, por serem parte da avaliação preliminar.
- **Processamento através da interface com API flask**: Avaliação do correto processamento dos arquivos enviados pelo usuário, ou seja, a verificação do envio dos arquivos para o backend, o processamento deles e a geração dos resultados esperados. Isto foi feito tanto na fase de treinamento quanto na de classificação.
- **Teste comparativo de performance**: Teste comparativo de tempo de processamento com ferramenta já existente. Neste caso, foi utilizado o Genome Detective, um aplicativo web que processa e classifica genomas virais com rapidez e precisão (Vilsker *et al.*, 2018).

Os resultados da avaliação são apresentados no capítulo 5.

5. **Apresentação das contribuições científicas:** Para garantir a disponibilização de forma completa dos resultados e contribuições alcançados, será utilizado o repositório do projeto no GitHub do Grupo de Bioinformática Computacional (G2BC), e na plataforma da Universidade do Estado da Bahia (UNEB).

   As contribuições específicas deste projeto são apresentadas no capítulo 4.

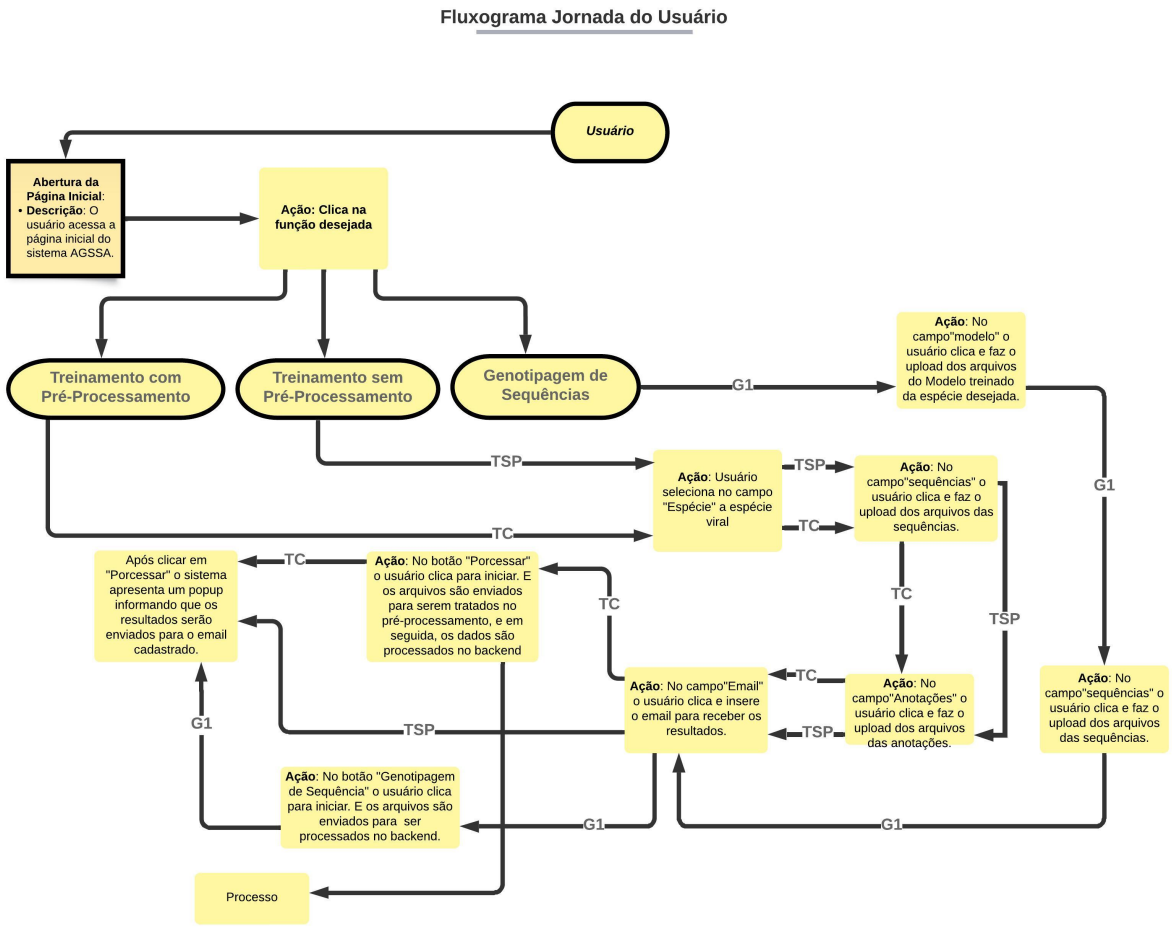
6. **Iterações:** Três iterações de refinamento do projeto foram realizadas à medida que novos problemas e ideias surgiam. Em particular, houve um aprimoramento gradual da interface do usuário, simplificando o design para torná-la mais intuitiva e fácil de usar. Foram adicionadas mensagens de confirmação para ações do usuário e o desempenho foi otimizado, tanto pela redução do tempo de carregamento quanto pela melhoria da eficiência do código do backend, através da refatoração do código. Em cada iteração, foram corrigidos os bugs identificados na iteração anterior.

4 CONTRIBUIÇÕES DO PROJETO

4.1 Desenvolvimento da Ferramenta Web AGSSA

Na figura 8 mostramos a estrutura da ferramenta a partir da Jornada do Usuário. O software *AGSSA* é uma plataforma desenvolvida para a classificação de sequências de DNA viral usando técnicas de AM. Por se tratar de uma ferramenta de aprendizado, é necessário implementar 2 processos bem diferenciados: (1) Treinamento, (2) Análise (Classificação).

Figura 8 – Jornada do Usuário da Ferramenta web AGSSA



Fonte: O Autor

A versão anterior da ferramenta, que era não baseada na web, realizava apenas o treinamento dos modelos. Contudo, os modelos treinados não eram utilizados para classificar novas sequências, que é o objetivo principal da versão atual desta ferramenta. Nesta versão foi incluída a funcionalidade de genotipagem de nova sequências.

Além disso, a versão anterior estava limitada no sentido de não estar pronta para

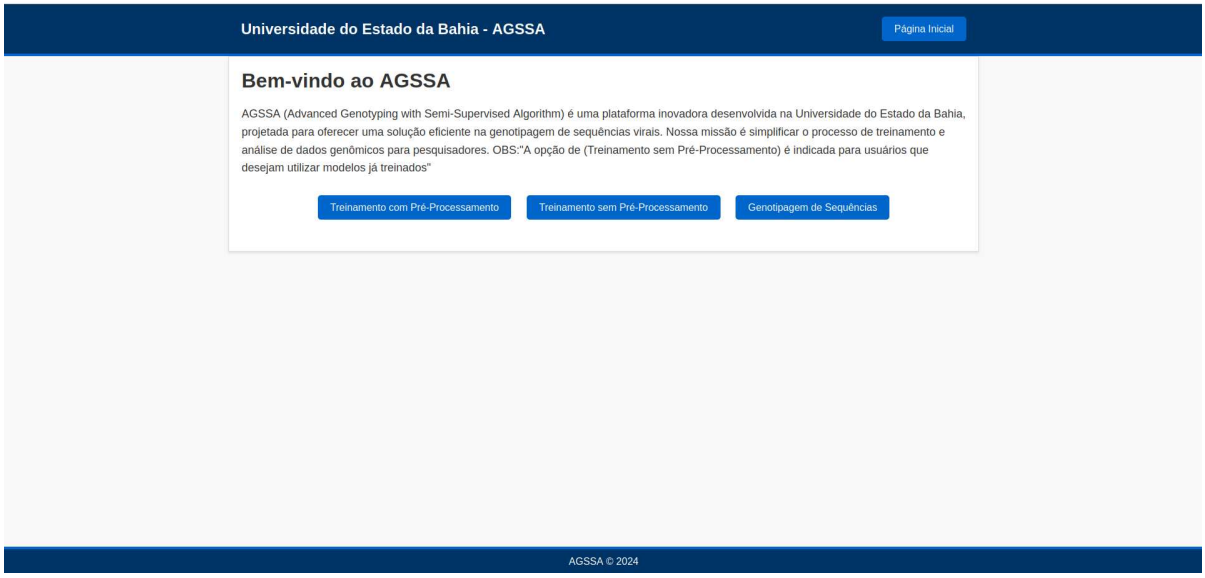
receber um dataset de treinamento que tivesse sido construído e validado pelo usuário, ou seja, previamente pré-processado, em cujo caso as fases de pré-processamento antes do treinamento implementadas na versão original não são mais necessárias. Devido a isto, foi criada uma nova opção de "Treinamento Sem Pré-Processamento", no qual *AGSSA* não realiza as etapas de pré-processamento habituais do dataset de treinamento, passando diretamente para a fase de treinamento em si. Nesta seção detalhamos o funcionamento da ferramenta, separando claramente os fluxos de **Treinamento sem Pré-processamento**, **Treinamento com Pré-processamento** e **Genotipagem de Sequências**.

4.1.1 Funcionamento da interface do sistema AGSSA

Abertura da Página Inicial

- O usuário acessa a página inicial do sistema AGSSA.
- Ação: Clica na Função Desejada O usuário escolhe entre as opções "Treinamento com Pré-Processamento", "Treinamento sem Pré-Processamento"ou "Genotipagem de Sequências".

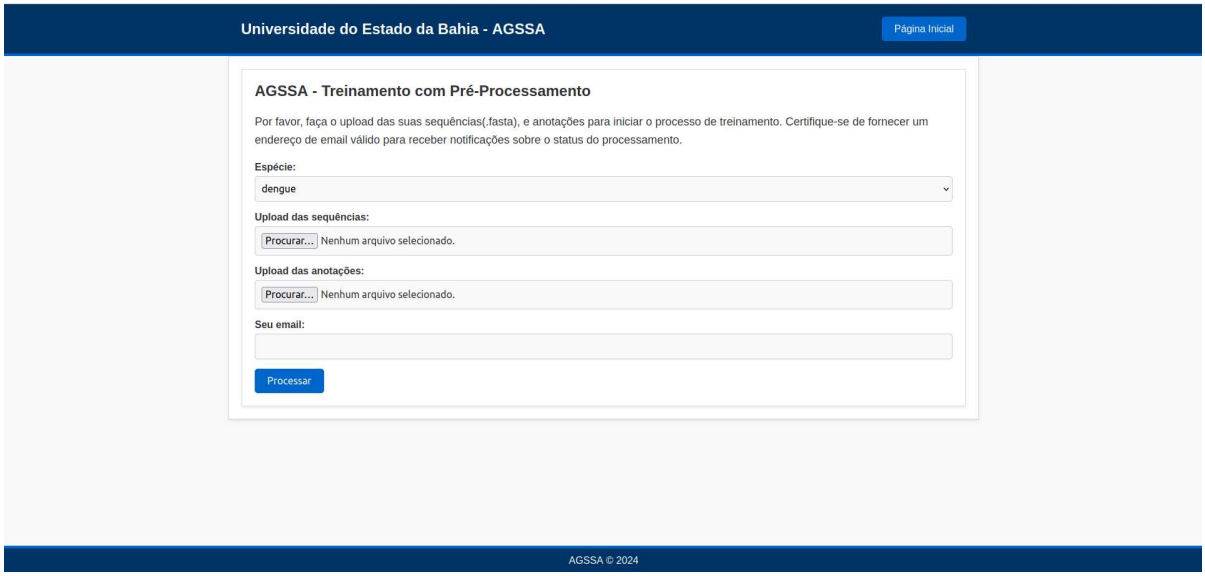
Figura 9 – Página Inicial do Sistema *AGSSA*



Fonte: O Autor

Fluxo de Treinamento:

Figura 10 – Treinamento **com** Pré-Processamento



Fonte: O Autor

- **Etapas**:
  1. **Seleção da Espécie Viral**: O usuário escolhe a espécie de vírus.
  2. **Upload de Sequências**: O usuário faz upload dos arquivos de sequências.
  3. **Upload de Anotações**: O usuário faz upload dos arquivos de anotações.
  4. **Inserção de E-mail**: O usuário insere o e-mail para receber os resultados.
  5. **Pré-Processamento**: O usuário clica em "Processar", e os arquivos são enviados para o Pré-Processamento e em seguida para o processamento no backend.

Processamento:

- Os arquivos são enviados e processados no backend. Um pop-up informa que os resultados serão enviados para o e-mail cadastrado.

Figura 11 – Upload de arquivos para treinamento **com** Pré-Processamento

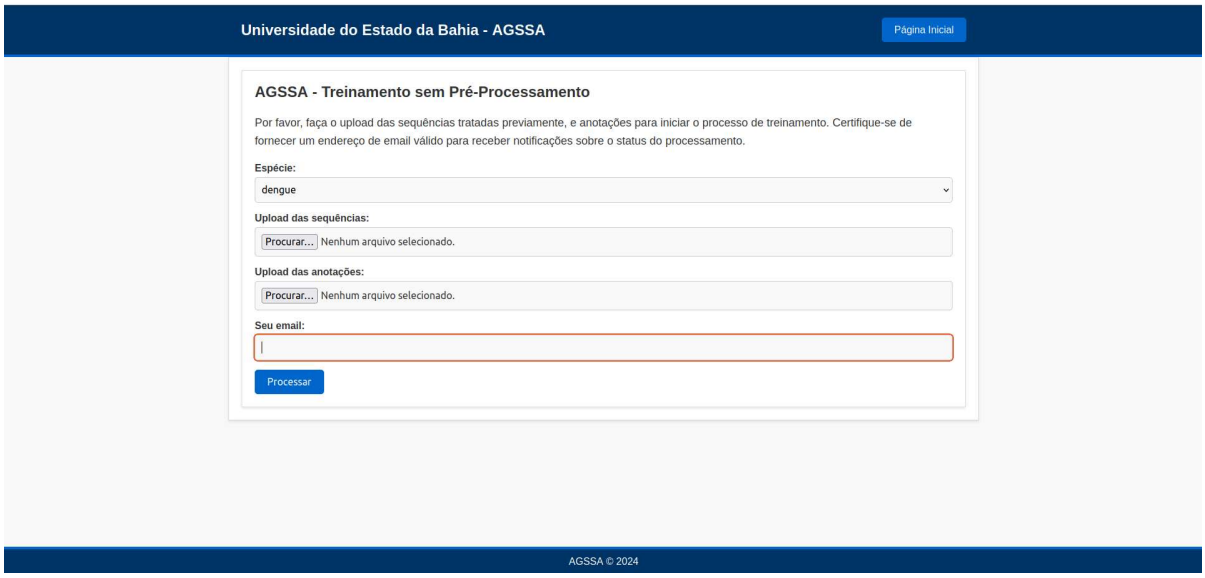
Fonte: O Autor

- **Notificação**: O sistema apresenta um pop-up informando que os resultados serão enviados para o e-mail cadastrado.

Fluxo de Treinamento sem Pré-Processamento:

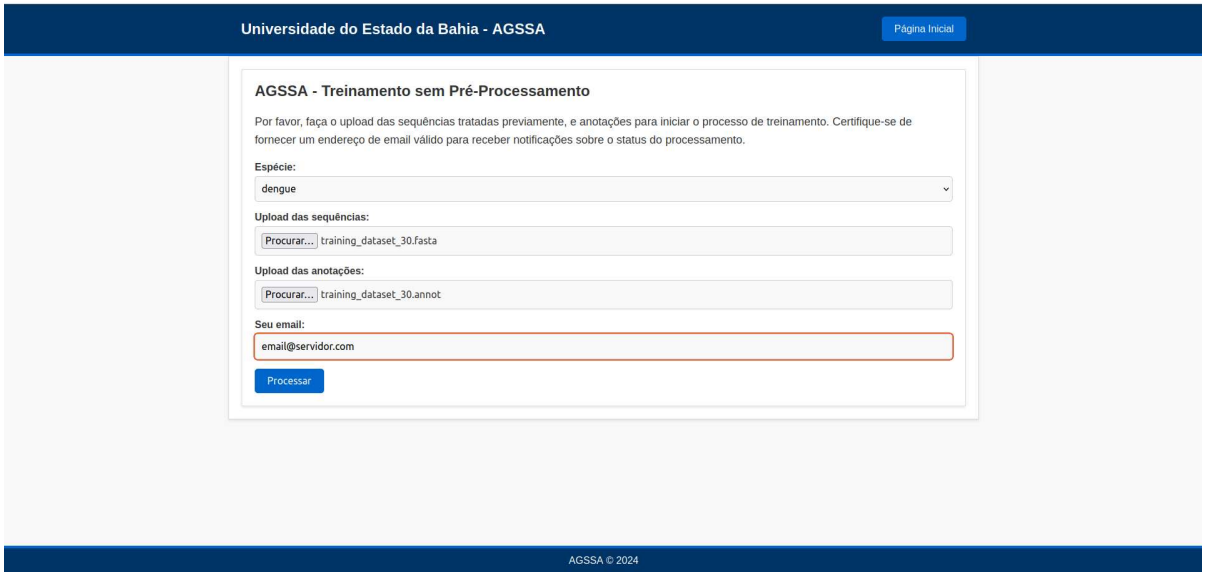
- **Etapas**:
  1. **Seleção da Espécie Viral**: O usuário escolhe a espécie de vírus.
  2. **Upload de Sequências**: O usuário faz upload dos arquivos de sequências já tratadas e alinhadas.
  3. **Upload de Anotações**: O usuário faz upload dos arquivos de anotações.
  4. **Inserção de Email**: O usuário insere o email para receber os resultados.

Figura 12 – Treinamento sem Pré-Processamento



Fonte: O Autor

Figura 13 – Upload de arquivos para treinamento sem pré-processamento

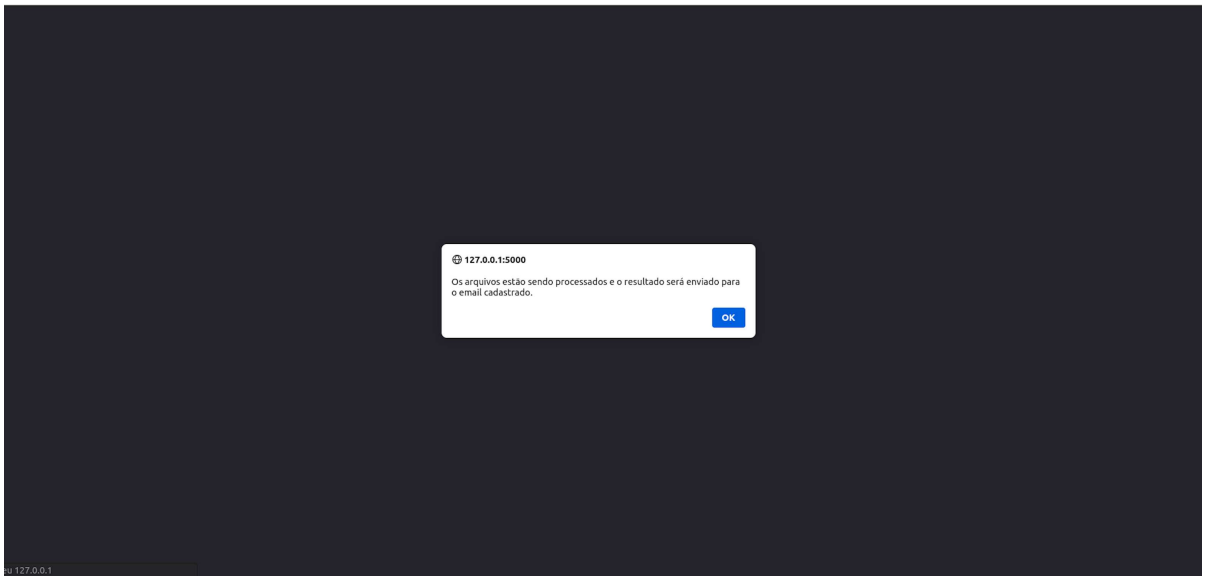


Fonte: O Autor

- **Processamento**: Os arquivos são enviados e processados no backend. Um pop-up informa que os resultados serão enviados para o e-mail cadastrado.

- **Notificação**: O sistema apresenta um pop-up informando que os resultados serão enviados para o email cadastrado.

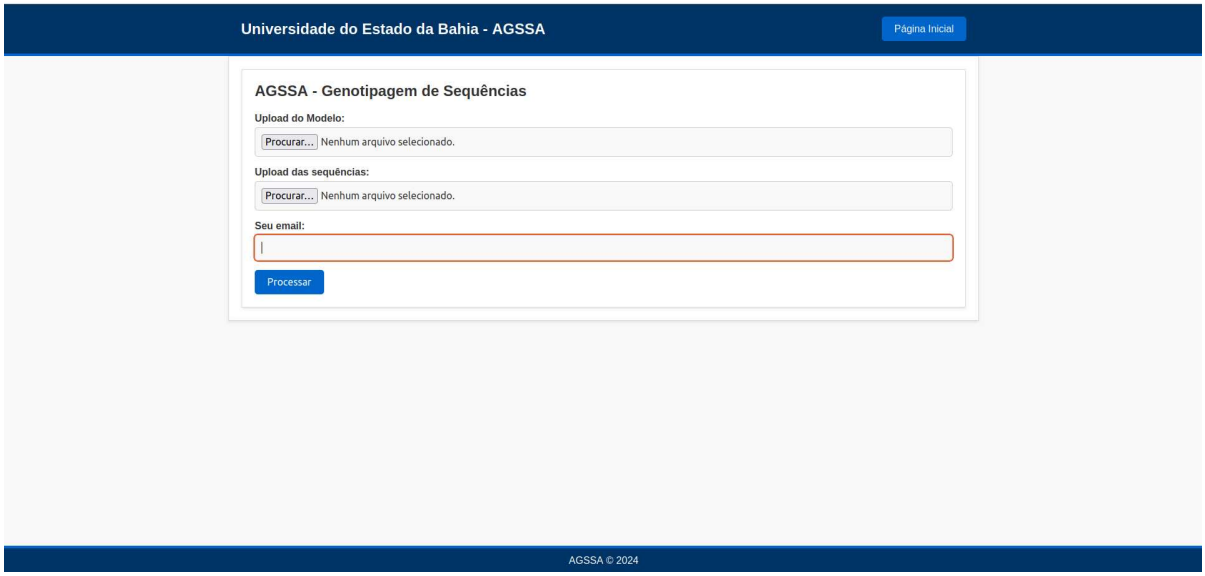
Figura 14 – Notificação de Envio



Fonte: O Autor

Fluxo de Genotipagem

Figura 15 – Tela de Genotipagem

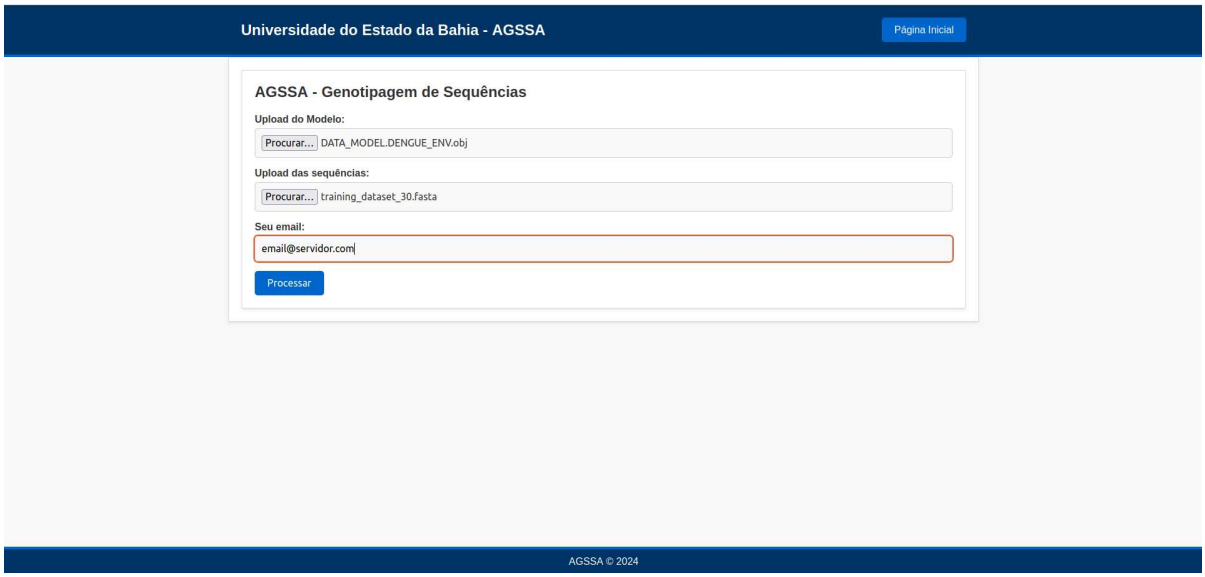


Fonte: O Autor

- **Etapas**:

  1. **Upload do Modelo**: O usuário faz upload dos arquivos do modelo treinado.
  2. **Upload de Sequências**: O usuário faz upload dos arquivos das sequências.

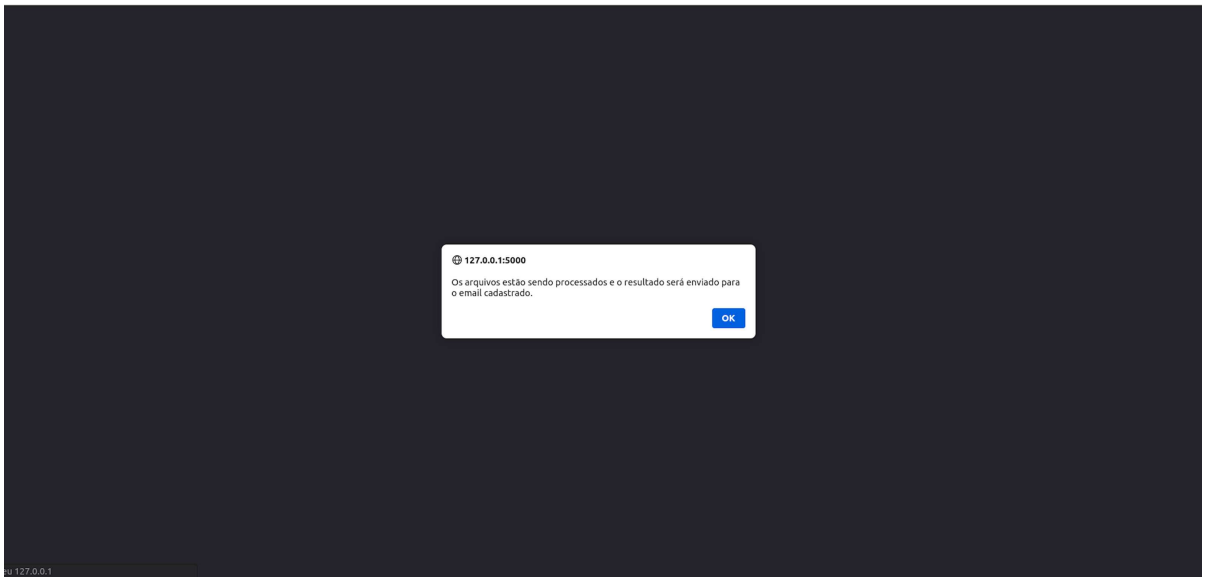
Figura 16 – Upload de arquivos para genotipagem



Fonte: O Autor

3. **Genotipagem**: O usuário clica em "Genotipar", e os arquivos são enviados para processamento no backend.

Figura 17 – Notificação de Envio



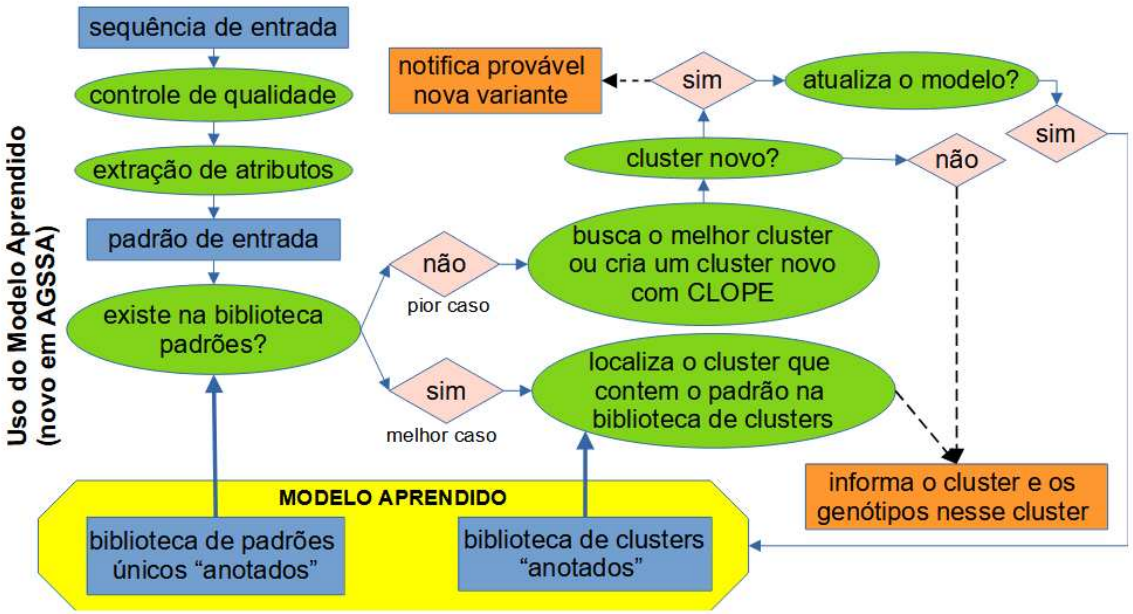
Fonte: O Autor

- **Notificação**: O sistema apresenta um pop-up informando que os resultados serão enviados para o email cadastrado.

4.2 Desenvolvimento do Pipeline de Genotipagem de Sequências

Uma representação esquemática do pipeline desenvolvido para a genotipagem de sequências, com o algoritmo de amprendizado de máquina embarcado na ferramenta *AGSSA*, é apresentado na figura 18.

Figura 18 – Pipeline de Genotipagem



Fonte: O autor

O pipeline inicia com a entrada de uma nova sequência e a realização de seu controle de qualidade. Após aprovação, o padrão da sequência é extraído, seguindo as definições estabelecidas no MODAP, e então submetido ao processo de classificação.

Os componentes principais do MODAP para essa tarefa de classificação estão representados no bloco amarelo na parte inferior da figura. Esses componentes são: (1) a biblioteca de padrões únicos, usada para verificar se o padrão gerado a partir da sequência de entrada já é conhecido; e (2) a biblioteca de clusters, utilizada para: (a) identificar o cluster que contém um padrão conhecido (melhor caso) e (b) localizar o cluster mais próximo de um padrão desconhecido (pior caso).

Além desses artefatos do MODAP, o pipeline utiliza o número de repetições e o valor do parâmetro de repulsão ótimo definidos durante o treinamento, no caso do pior cenário. Nesse caso, o pipeline precisa identificar o cluster com maior probabilidade de conter o novo padrão ou criar um novo cluster, caso o padrão não se ajuste adequadamente a nenhum dos clusters existentes.

Quando um novo cluster é criado, isso pode indicar que a sequência de entrada representa uma nova variante. Nesse caso, é fundamental informar tanto o usuário quanto a equipe de manutenção da ferramenta para que as devidas providências sejam tomadas. Após verificar que a sequência processada possui alta qualidade de sequenciamento, as ações a serem realizadas podem incluir: (1) a nomeação da nova variante, (2) a notificação aos órgãos competentes e (3) a atualização do MODAP, incorporando a nova sequência ao modelo.

Quando a sequência de entrada é alocada a um cluster existente, o pipeline retorna ao usuário a lista de variantes genotípicas presentes nesse cluster, juntamente com a porcentagem correspondente a cada variante.

Sempre que um novo padrão for identificado, seja ele correspondente ou não a uma possível nova variante, a equipe de manutenção deve ser notificada para decidir se o MODAP será atualizado com o novo padrão ou não.

Observações:

- A funcionalidade de notificação e consulta à equipe de manutenção, precisa ser implementada. Na versão atual a atualização acontece de forma automática.
- No apêndice H, descrevemos o algoritmo por trás do esquema apresentado na figura 18.

4.3 Aperfeiçoamento do Método de Aprendizado de Máquina(AM)

Outra importante contribuição deste projeto ao método de aprendizado de máquina com CLOPE, foi a implementação de um procedimento para a otimização do parâmetro de repulsão r usando para isto um conjunto de treinamento anotado.

Como descrito na seção 2.5.2 o método CLOPE é capaz de encontrar, fixado um valor de repulsão r, a melhor estrutura de *clusters* em um número prefixado R de repetições aleatórias, sem precisar saber o genótipo das amostras do conjunto de treinamento. A qualidade dos agrupamentos é aferida em cada tentativa a partir de uma métrica de valor dos *clusters* que leva em conta a coesão *intra-cluster* e o espaçamento *inter-clusters*.

Contudo, não há um critério *ad-hoc* para estimar nenhum dos dois parâmetros do método, nem r, T, salvo que sabemos que: (1) Quando r aumenta o número de *clusters* aumenta e o tamanho médio dos *clusters* diminui, e (2) Quando T aumenta, diminui a probabilidade de assumir como bom um agrupamento longe do ótimo, mas aumenta linearmente o custo computacional.

Visando reduzir a incerteza na definição dos hiper-parâmetros do método CLOPE, a qual é de fato uma limitação compartilhada por todos os métodos de AM, decidimos introduzir uma referência para guiar a escolha do melhor valor de repulsão, durante o treinamento.

Como na maioria dos casos práticos os conjuntos de treinamentos são construídos com amostras bem conhecidas (anotadas = genotipadas), decidimos usar essa informação, que já está disponível, como referência. Esta decisão leva a que o novo método não seja totalmente não supervisionado, passando a ser classificado como semi-supervisionado, segundo a classificação descrita na seção 2.4. De acordo com isto, o nome do método

foi modificado. Anteriormente era denominado **AGUA**, siglas de **Advanced Genotyping with Unsupervised Algorithm** e agora foi renomeado como **AGSSA**, siglas de **Advanced Genotyping with Semi-Supervised Algorithm**.

A meta central introduzida foi que: **"O agrupamento ideal não deve conter amostras de genótipos diferentes num mesmo *cluster*"**.

Uma vez definida a meta central, estabelecemos os dois estados extremos de qualquer agrupamento: (1) agrupamento onde um único *cluster* contem todas as amostras, (2) agrupamento com múltiplos *clusters* contendo apenas uma amostra cada. Por isso, se o conjunto de treinamento tiver n amostras, então o número de *clusters* possíveis variará entre 1 e n *clusters*. A partir desses dois extremos, apenas o de n *clusters* satisfaz nossa meta central, mas não fornece nenhuma informação de similaridade ou associação de amostras que é a informação buscada nos problemas de classificação. Por isso, ficou claro que nossa meta central permitiria encontrar o menor valor de r, diretamente relacionado ao número de *clusters* que a satisfaz.

A seguir descrevemos a métrica usada para avaliar a qualidade de um agrupamento no sentido de garantir a satisfação da meta central.

Para avaliar a qualidade de um agrupamento utilizamos uma métrica de **pureza** baseada nas etiquetas atribuídas a cada sequência. No nosso caso a etiqueta descreve a variante viral (genótipo) ao qual cada amostra do conjunto de treinamento pertence, ou seja, vem dada pela genotipagem feita com ferramentas certificadas, como o Genome Detective (Vilsker *et al.*, 2018), entre outras.

Um *cluster* totalmente **puro** contém sequências com um único conjunto de etiquetas, ou seja, não mistura genótipos no nosso caso. Na nossa aplicação não é problema que o agrupamento contenha vários *clusters* puros de um mesmo genótipo, pois isto apenas iria refletir a diversidade existente dentre os genótipos, a qual é uma informação secundária importante que o classificador poderia vir a fornecer.

Para quantificar a pureza, considere que no conjunto de treinamento T as sequências são classificadas em G genótipos distintos e etiquetadas com $g = 1, 2, \ldots, G$. Considere que CLOPE identificou $K > 1$ *clusters* e que alocou em cada um deles um conjunto de N_k transações $T_k = \{t_{k,1}, t_{k,2}, \ldots, t_{k,N_k}\}$, nos *clusters* $k = 1, 2, \ldots, K$.

Agora, se denotamos por $d_{k,g}$ o número de transações no *cluster* k, ou seja, em T_k, que estão anotadas com o genótipo g, considerando os K *clusters* e os G genótipos, podemos preencher uma Matriz de Distribuição

$$D = \begin{bmatrix} d_{1,1} & d_{1,2} & \cdots & d_{1,G} \\ d_{2,1} & d_{2,2} & \cdots & d_{2,G} \\ \vdots & \vdots & \ddots & \vdots \\ d_{K,1} & d_{K,2} & \cdots & d_{K,G} \end{bmatrix} \tag{4.1}$$

onde k indica uma linha (cluster) e g uma coluna (genótipo) de D.

De acordo com isto, a pureza do *cluster* k pode ser avaliada como:

$$p_k = \begin{cases} \dfrac{\max_g(d_{k,g})}{N_k} \in [0,1] & \text{para } N_k > 0 \\ 0 & \text{em caso de } \textit{cluster} \text{ vazio} \end{cases} \tag{4.2}$$

Note, que no caso que o *cluster* k seja puro, ou seja, que tenha apenas transações de um único genótipo $g*$, se cumpre que $\max_g(d_{k,g}) = d_{k,g*} = N_k$ pelo que $p_k = 1$.

Podemos medir a falta de pureza absoluta do *cluster* k como

$$h_k = N_k - \max_g(d_{k,g}) \geq 0 \tag{4.3}$$

e a falta total de pureza do agrupamento como

$$H = \sum_k h_k \geq 0 \tag{4.4}$$

pelo que a pureza do agrupamento pode ser estimada como

$$\mathcal{P} = 1 - \frac{H}{N} \in [0,1] \tag{4.5}$$

onde $N = N_1 + N_2 + \cdots + N_K$ é o número total de transações em T.

Baseado nas métricas definidas em 4.2 e 4.5, introduzimos uma medida de qualidade do agrupamento da forma:

$$Q = \left[p_k^{min}\ p_k^{avg}\ \mathcal{P}\right]^{1/3} \tag{4.6}$$

onde p_k^{min}, p_k^{avg} são a pureza mínima e média dos *clusters*.

A métrica combinada Q mede a extensão em que os *clusters* contêm uma única classe de amostras. Alta Q indica que os *clusters* são compostos predominantemente por uma única classe, tornando-os mais homogêneos.

O novo algoritmo de treinamento varia o parâmetro de repulsão r a ser otimizado, começando num valor mínimo $r_{min} \geq 1$ e aumentando com passo $\Delta_r \in [0.1, 0.9]$ até atingir um valor máximo $r_{max} >> r_{min}$, prefixados.

Com cada valor de repulsão r o algoritmo de treinamento realiza o número prefixado R de repetições do agrupamento, calculando em cada repetição a métrica de qualidade Q, armazenando a maior métrica Q_{best} e o melhor agrupamento até o momento.

A experiência demonstrou que a máxima qualidade dos agrupamentos com cada valor de r, $Q_{best}(r)$, começa a aumentar gradativamente com o aumento de r, até que começa a diminuir de forma permanente. Por este motivo, o algoritmo de busca do r ótimo é interrompido a primeira vez que Q_{best} decresce, retornando como valor ótimo o valor de repulsão anterior à descida, ou seja, para o qual foi obtido o máximo $Q_{best}(r)$.

Como resultado do treinamento o algoritmo retorna o valor ótimo r_{otim} e também o agrupamento de maior qualidade Q_{best}. O agrupamento consiste numa lista que contêm o número k do *cluster* ao qual cada transação (sequência) no conjunto de treinamento T foi alocado. Denotemos o agrupamento por

$$\mathcal{G} = [k_1, k_2, \ldots, k_N] \tag{4.7}$$

onde $k_i \in [1, K]$ representa o *cluster* ao qual a transação (sequência) i foi alocada.

Junto com o agrupamento definido em 4.7, o método retorna a matriz de distribuição D desse agrupamento definida em 4.1. Ou seja, no final do treinamento $\mathcal{G}$ e D são adicionados ao modelo de referência descrito no apêndice G.1.

4.3.1 Estrutura do Modelo AGSSA

A estrutura do novo modelo $AGSSA$ utilizado para a classificação das sequências codificantes é similar à do modelo anterior $AGUA$, descrito na seção D.1, adicionando os seguintes campos:

- **Repetitions:** Número de repetições do treinamento.
  - *Tipo:* Número (inteiro)
  - *Descrição:* Número de repetições do agrupamento sorteando as amostras do conjunto de treinamento para cada valor de repulsão prefixado.
- **Resolution:** Resolução geral do modelo.
  - *Tipo:* Número (float ou inteiro)
  - *Descrição:* Resolução utilizada no modelo.
- **CLOPE_repulsion:** Parâmetro de repulsão do algoritmo CLOPE.
  - *Tipo:* Número (float ou inteiro)
  - *Descrição:* Valor do parâmetro de repulsão utilizado no algoritmo CLOPE.

5 DISCUSSÃO DE RESULTADOS

Como definido nos objetivos, este projeto teve como metas realizar um desenvolvimento evolutivo de uma ferramenta de bioinformática denominada *AGUA*, desenvolvida pelo grupo de pesquisa nos dois anos anteriores, criando uma nova versão, denominada *AGSSA*, com melhorias funcionais, procedimentais e de interface web. Por isso, a análise de resultados do projeto abordou três direções principais, resumidas a seguir:

1. Avaliar a versão web da nova ferramenta *AGSSA* do ponto de vista funcional, com foco no funcionamento da interface (os resultados são apresentados na seção 5.1).
2. Avaliar a performance computacional da ferramenta *AGSSA* (os resultados são apresentados e analisados na seção 5.2).
3. Considerando que a versão anterior, *AGUA*, foi testada apenas com o vírus SARS-CoV-2, avaliar a qualidade da classificação de *AGSSA* com outra espécie viral, neste caso, o vírus da Dengue (resultados analisados na seção 5.2.2).

5.1 Avaliação Funcional da Interface

O desempenho da interface desenvolvida, composta pelas telas mostradas nas Figuras 9 a 17 na Seção 4.1.1, foi avaliado por três membros do grupo.

A avaliação incluiu aspectos como clareza e simplicidade da interface, destacando o design limpo e intuitivo, com rótulos claros e concisos, evitando sobrecarregar o usuário com informações desnecessárias.

Também foi analisada a consistência da interface, considerando o uso uniforme de padrões, como cores, fontes e a disposição dos elementos. Essa consistência é fundamental para facilitar a familiarização do usuário com o sistema.

Outro ponto de avaliação foi o feedback visual, ou seja, a orientação fornecida ao usuário após cada interação, bem como a prevenção e o tratamento de erros para garantir uma experiência eficiente e sem frustrações.

Por fim, foi dado um foco especial ao desempenho técnico da interface, avaliando tempos de carregamento rápidos e a fluidez geral da experiência do usuário.

Os resultados das avaliações foram considerados satisfatórios, embora tenha sido identificado que melhorias no feedback ao usuário e no tratamento de erros podem contribuir para uma experiência ainda mais eficiente e robusta.

5.2 Avaliação da Performance Computacional de *AGSSA*

5.2.1 Experimentos Realizados

Baseado no anterior, o back-end de *AGSSA* foi testado com conjuntos de treinamento formados por sequências genotipadas (anotadas) de alta qualidade do gene E do vírus da Dengue extraídos do ABVDB (Restovic *et al.*, 2018) e genotipadas com *Genome Detective* (Vilsker *et al.*, 2018), com distintos números de sequências: 49, 97, 196, 391, 1.080, 1.701 e 3.068. Os conjuntos de treinamento menores foram construídos a partir do conjunto de treinamento maior, com 3.068 sequências, selecionando percentuais iguais dos distintos genótipos presentes no conjunto de treinamento maior, de forma a preservar a representatividade de cada variante viral, mesmo nos conjuntos de treinamento menores. Este processo foi realizado de forma tal que nos conjuntos de treinamento menores se garante pelo mínimo uma sequência de cada variante viral no conjunto de treinamento maior, ou seja mantendo constante a diversidade genética nos datasets independentemente do tamanho deles.

A ferramenta *AGSSA* (o back-end) foi avaliada nas fases de treinamento e de análise com os mesmos conjuntos de dados. Na fase de treinamento foi utilizada a nova variante de treinamento sem pré-processamento, na qual o usuário já fornece um conjunto de treinamento validado, ou seja: (1) não contem sequências codificantes similares mas não repetidas, (2) as sequências tem o mesmo tamanho e estão alinhadas no quadro de leitura, (3) as sequências não contém caráteres estranhos além de um limiar estabelecido e (4) as sequências estão devidamente anotadas com o genótipo constante na etiqueta das sequências no arquivo de entrada no formato FASTA.

Com fins de comparação do tempo de processamento, foram criados conjuntos com 135, 405, 948, 1.356 e 1.628 sequências que foram submetidos para classificação/análise pelo *Genome Detective*, medindo o tempo aproximado entre a submissão e o recebimento do e-mail de finalização da tarefa que o *Genome Detective* envia a seus usuários. Todas as sequencias foram extraídas de forma aleatória do maior conjunto de treinamento de *AGSSA* contendo 3.068 sequências. Devido, principalmente, a uma limitação de até 2.000 sequências de entrada na versão utilizada do *Genome Detective*, o estudo se limitou a 1.628 sequências, enquanto *AGSSA* foi testado com mais de 3.000 sequências. Os experimentos com o *Genome Detective* foram realizados utilizando o servidor disponibilizado em: https://gd-test.genomedetective.com/app/typingtool/dengue/.

Cabe destacar que devido à abordagem do *Genome Detective*, não existe uma fase de treinamento inicial, seguida de fases de análise usando o modelo treinado como é o caso de *AGSSA*. O *Genome Detective*, após um pré-processamento inicial de controle da qualidade das sequências de entrada, precisa realizar todo o processo desde o alinhamento da sequência de entrada com o dataset da árvore de referência, passando pela construção da

árvore filogenética seguida da identificação da espécie mais próxima na árvore. Neste cenário, comparamos os tempos de treinamento e os de análise com os tempos de classificação do *Genome Detective*, em função do número de sequências do conjunto de treinamento.

Os experimentos realizados com *AGSSA* foram conduzidos após a configuração do ambiente descrito no apêndice I. Para a execução dos testes de performance todos os componentes do back-end foram instalados e configurados para garantir a correta funcionalidade da aplicação.

5.2.2 Execução dos Experimentos

Com arquivos de treinamento de tamanhos crescentes contendo sequências do gene E do vírus da Dengue, foram realizados os seguintes passos:

1. **Pré-processamento das Sequências**: Utilizando o BLAST com a referência do gene viral alvo, traduzida para aminoácidos, foi feito o alinhamento uma-a-uma das sequências do conjunto de treinamento bruto.

2. **Filtragem das Sequências**: Antes das sequências serem carregadas no sistema *AGSSA* elas passaram por um pré-processamento para remover sequências de baixa qualidade (com mais de 40 caráteres distintos de $\{A, G, c, T\}$ e com comprimento menor que o da sequência de referência depois de alinhadas).

3. **Treinamento**: Modelos de AM foram criados usando os conjuntos de sequências pré-processados, para agrupar as sequências em clusters de alta pureza, ou seja, compostos, na sua maioria, por sequências de uma mesma variante viral.

4. **Teste de Classificação**: As sequências do conjunto de treinamento foram classificadas com o modelo criado para verificar a correção do modelo. Nestes testes, além de avaliar a qualidade dos resultados, foi medido o tempo empregado para classificar as sequências. Esses tempos são comparados com os do *Genome Detective* na seção 5.3.

5. **Geração de Relatório do Treinamento**: Após a análise, foram gerados relatórios detalhados com:

   a) A métrica da performance classificatória, que qualifica o modelo aprendido, a partir da pureza dos clusters. A qualidade é maior na medida que a métrica Q tende a 1.0.

   b) A Matriz de Distribuição que contem o número de amostras de cada variante genotípica agrupadas em cada cluster. Nas figuras 19 e 20 mostramos em duas partes a Matriz de distribuição obtida do agrupamento de 3.067 amostras do vírus da dengue distribuídas em 19 variantes genotípicas, com o método $CLOPE$.

```
DISTRIBUTION MATRIX (parte 1)

genotypes: ['3.II','2.VI','1.IV','1.I','3.I','1.II','1.III','3.IV','2.IV','2.I','1.V','4.I'
,'4.III','2.III','2.II','2.V','3.III','4.II','4.IV']

0   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 144, 0, 0, 0]   SINGLE SPECIES: TYPE 2.V
1    [92, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]  SINGLE SPECIES: TYPE 3.II
2   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 163, 0, 0, 0, 0] SINGLE SPECIES: TYPE 2.III
3    [0, 0, 0, 0, 0, 0, 0, 0, 14, 0, 0, 0, 0, 0, 0, 0, 0, 0]  SINGLE SPECIES: TYPE 2.IV
4   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 182, 0, 0] SINGLE SPECIES: TYPE 3.III
5   [0, 0, 0, 249, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]   SINGLE SPECIES: TYPE 1.I
6    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 28, 0, 0, 0, 0, 0, 0]   SINGLE SPECIES: TYPE 4.I
7   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 143, 0, 0, 0, 0, 0, 0]   SINGLE SPECIES: TYPE 1.V
8    [0, 0, 0, 0, 25, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]   SINGLE SPECIES: TYPE 3.I
9    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 63, 0]  SINGLE SPECIES: TYPE 4.II
10   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 22, 0, 0, 0]  SINGLE SPECIES: TYPE 2.II
11   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 23, 0]  SINGLE SPECIES: TYPE 4.II
12   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 77, 0, 0, 0, 0, 0, 0, 0]   SINGLE SPECIES: TYPE 1.V
13   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 23, 0, 0] SINGLE SPECIES: TYPE 3.III
14  [0, 0, 0, 160, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]   SINGLE SPECIES: TYPE 1.I
15   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 86, 0, 0, 0]  SINGLE SPECIES: TYPE 2.II
16   [0, 0, 0, 0, 0, 0, 0, 0, 0, 14, 0, 0, 0, 0, 0, 0, 0, 0]   SINGLE SPECIES: TYPE 2.I
17    [0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]  SINGLE SPECIES: TYPE 2.VI
18    [0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]   SINGLE SPECIES: TYPE 1.I
19   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 21, 0, 0, 0, 0, 0, 0, 0]   SINGLE SPECIES: TYPE 1.V
20   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 70, 0, 0, 0]   SINGLE SPECIES: TYPE 2.V
21   [0, 0, 22, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] MULTIPLE:1.IV(0.96),1.I(0.04)
22   [0, 0, 0, 0, 0, 0, 42, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] SINGLE SPECIES: TYPE 1.III
23   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 79, 0, 0, 0, 0] SINGLE SPECIES: TYPE 2.III
24   [0, 0, 0, 0, 22, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]   SINGLE SPECIES: TYPE 3.I
25   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 98, 0, 0] SINGLE SPECIES: TYPE 3.III
26    [0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]  SINGLE SPECIES: TYPE 1.IV
27  [0, 0, 0, 145, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]   SINGLE SPECIES: TYPE 1.I
28   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 28, 0]  SINGLE SPECIES: TYPE 4.II
29   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 49, 0, 0] SINGLE SPECIES: TYPE 3.III
30    [0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]   SINGLE SPECIES: TYPE 1.I
31   [38, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]  SINGLE SPECIES: TYPE 3.II
32    [0, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]  SINGLE SPECIES: TYPE 3.IV
33   [0, 0, 23, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]  SINGLE SPECIES: TYPE 1.IV
34   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 35, 0, 0, 0]  SINGLE SPECIES: TYPE 2.II
35    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0]   SINGLE SPECIES: TYPE 4.I
36   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 53, 0, 0, 0, 0, 0, 0, 0]   SINGLE SPECIES: TYPE 1.V
37   [10, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]  SINGLE SPECIES: TYPE 3.II
38    [0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]   SINGLE SPECIES: TYPE 3.I
39    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0]   SINGLE SPECIES: TYPE 4.I
40   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 59, 0, 0, 0, 0, 0] SINGLE SPECIES: TYPE 2.III
41    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0]  SINGLE SPECIES: TYPE 4.II
42   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 11, 0] SINGLE SPECIES: TYPE 3.III
43   [0, 0, 0, 49, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]   SINGLE SPECIES: TYPE 1.I
44    [1, 0, 0, 0, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] MULTIPLE:3.II(0.10),3.I(0.90)
45    [0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]  SINGLE SPECIES: TYPE 1.IV
46    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0] SINGLE SPECIES: TYPE 4.III
47    [0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] SINGLE SPECIES: TYPE 1.III
48   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 36, 0, 0, 0]   SINGLE SPECIES: TYPE 2.V
```

Figura 19 – Primeira parte da Matriz de Distribuição de 3.067 amostras do vírus da Dengue usadas para construir o modelo de classificação em *AGSSA*

Fonte: O Autor

```
DISTRIBUTION MATRIX (parte 2)
species:
['3.II','2.VI','1.IV','1.I','3.I','1.II','1.III','3.IV','2.IV','2.I','1.V','4.I','4.III',
'2.III','2.II','2.V','3.III','4.II','4.IV']

49    [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]   SINGLE SPECIES: TYPE 2.I
50   [0, 0, 0, 81, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]   SINGLE SPECIES: TYPE 1.I
51   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 20, 0, 0, 0, 0, 0] SINGLE SPECIES: TYPE 2.III
52   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 15, 0, 0, 0, 0]  SINGLE SPECIES: TYPE 2.II
53    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0]  SINGLE SPECIES: TYPE 2.II
54    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0]  SINGLE SPECIES: TYPE 4.II
55   [0, 0, 0, 91, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]   SINGLE SPECIES: TYPE 1.I
56   [20, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]  SINGLE SPECIES: TYPE 3.II
57    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]  SINGLE SPECIES: TYPE 4.IV
58    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0]  SINGLE SPECIES: TYPE 4.II
59   [0, 0, 0, 34, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]   SINGLE SPECIES: TYPE 1.I
60   [0, 0, 0, 12, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]   SINGLE SPECIES: TYPE 1.I
61    [0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]  SINGLE SPECIES: TYPE 1.II
62   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 36, 0, 0] SINGLE SPECIES: TYPE 3.III
63    [0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] SINGLE SPECIES: TYPE 1.III
64    [0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] SINGLE SPECIES: TYPE 1.III
65    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0]   SINGLE SPECIES: TYPE 4.I
66    [0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]  SINGLE SPECIES: TYPE 2.VI
67   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 17, 0, 0, 0, 0, 0, 0, 0, 0]   SINGLE SPECIES: TYPE 1.V
68    [0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]  SINGLE SPECIES: TYPE 1.IV
69    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0]   SINGLE SPECIES: TYPE 4.I
70   [0, 0, 0, 23, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]   SINGLE SPECIES: TYPE 1.I
71   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 15, 0, 0, 0]   SINGLE SPECIES: TYPE 2.V
72    [0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0]   SINGLE SPECIES: TYPE 2.I
73   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 24, 0, 0, 0, 0, 0] SINGLE SPECIES: TYPE 2.III
74    [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]  SINGLE SPECIES: TYPE 2.IV
75    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0]  SINGLE SPECIES: TYPE 4.II
76    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0] SINGLE SPECIES: TYPE 3.III
77    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]   SINGLE SPECIES: TYPE 4.I
78   [0, 0, 0, 22, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]   SINGLE SPECIES: TYPE 1.I
79   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 24, 0, 0] SINGLE SPECIES: TYPE 3.III
80    [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]  SINGLE SPECIES: TYPE 2.VI
81    [7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]  SINGLE SPECIES: TYPE 3.II
82   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 14, 0, 0, 0, 0, 0] SINGLE SPECIES: TYPE 2.III
83   [0, 0, 0, 25, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]   SINGLE SPECIES: TYPE 1.I
84    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0] SINGLE SPECIES: TYPE 3.III
85    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0]   SINGLE SPECIES: TYPE 2.V
86   [0, 0, 0, 19, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]   SINGLE SPECIES: TYPE 1.I
87    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0]   SINGLE SPECIES: TYPE 1.V
88    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0]  SINGLE SPECIES: TYPE 4.II
89    [0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]   SINGLE SPECIES: TYPE 1.I
90   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 11, 0, 0, 0, 0]  SINGLE SPECIES: TYPE 2.II
91   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 10, 0, 0] SINGLE SPECIES: TYPE 3.III
92    [0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]   SINGLE SPECIES: TYPE 1.I
93    [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]   SINGLE SPECIES: TYPE 3.I
94   [0, 0, 0, 12, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]   SINGLE SPECIES: TYPE 1.I
95    [2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]  SINGLE SPECIES: TYPE 3.II
96    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0] SINGLE SPECIES: TYPE 2.III
97    [0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]   SINGLE SPECIES: TYPE 1.I
```

Figura 20 – Segunda parte da Matriz de Distribuição de 3.067 amostras do vírus da Dengue usadas para construir o modelo de classificação em *AGSSA*

Fonte: O Autor

Discussão: O agrupamento possui 98 clusters, sendo 96 (97.9%) deles clusters puros, ou seja, que contém sequências de uma única variante genotípica.

Apenas os clusters numerados 21 e 44 não são puros, pois contém sequências de duas variantes cada. O cluster 21 tem 22 sequências da variante $1.IV$ e 1 sequência da variante $1.I$. O cluster 44 possui 9 sequências da variante $3.I$ e 1 sequência da variante $3.II$. Estas raras exceções, ocorrem em clusters pequenos, o grau de mistura é mínimo, aparecendo contaminados com apenas uma sequência de outra variante, mas essa outra variante pertence ao mesmo sorótipo da variante maioritária no cluster. Como discutido no artigo(Fonseca *et al.*, 2019) em casos como esse, é possível

que essas 2 sequências podem conter segmentos genéticos de ambos os genótipos, indicando que ocorreu um evento de recombinação entre eles. A recombinação viral é um processo pelo qual duas ou mais sequências genéticas se combinam, resultando em uma nova sequência que contém segmentos de DNA ou RNA de diferentes origens. Esse fenômeno é comum em vírus que possuem genomas de RNA, como o Dengue, e pode levar à formação de novas cepas virais com características genéticas distintas. Além disso, discute-se a possibilidade de haver algum erro de genotipagem. Porém, para chegar a respostas conclusivas, seria necessário a realização de um estudo específico e aprofundado sobre o tema.

c) A matriz de distâncias, calculada como a distância de Jaccard (Jaccard, 1901) entre os IDs das classes primárias identificadas. A matriz de distâncias é usada para construir dendrogramas. Os dendrogramas são uma representação visual do agrupamento realizado por alguma técnica escolhida de clusterização hierárquica. $AGSSA$ utiliza o método UPGMA (Sokal; Michener, 1958) como default, mas possui uma ampla variedade de opções de clusterização hierárquica implementada. Na figura 21 mostramos o dendrograma gerado utilizando o método de ligação 'average' (UPGMA), exibindo as relações hierárquicas entre 391 sequências do gene E do vírus da Dengue dos sorótipos DENGUE-1, DENGUE-2, DENGUE-3 e DENGUE-4.

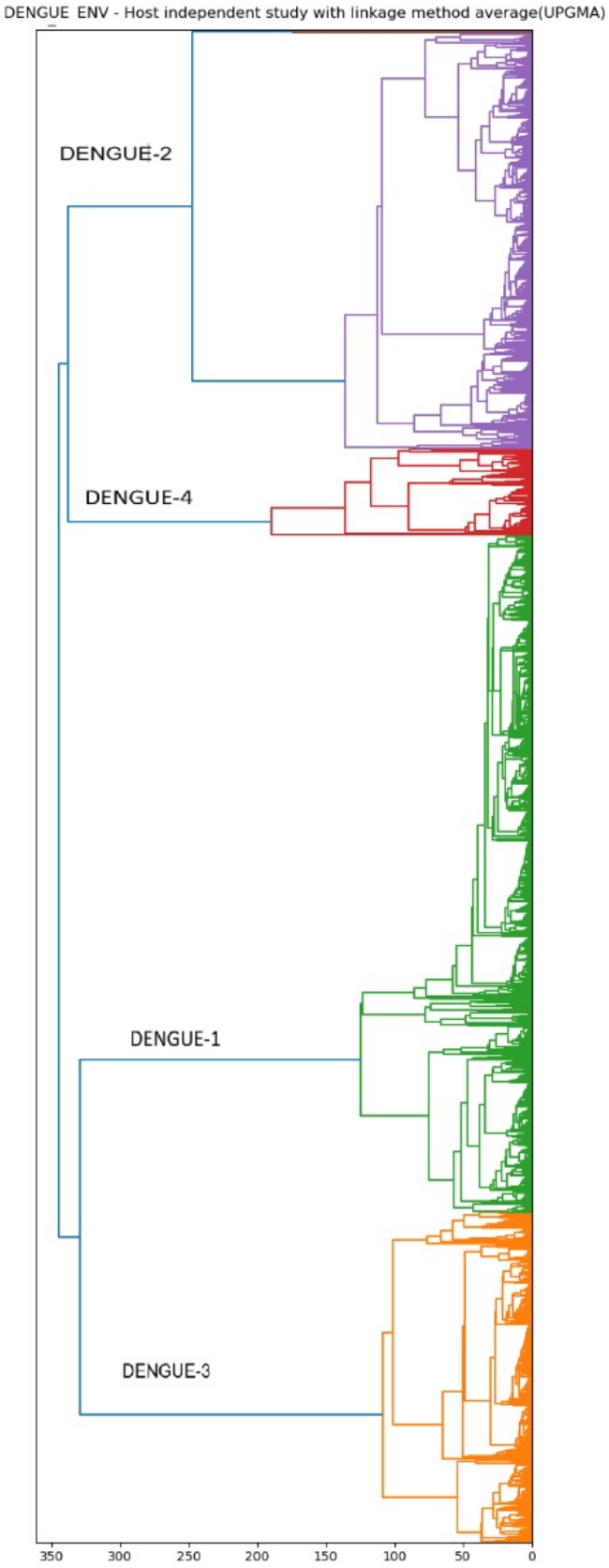


Figura 21 – Dendrograma do vírus da Dengue gerado pelo *AGSSA*

Fonte: O Autor

d) O tempo do treinamento, medido desde a submissão do dataset até o retorno do relatório do treinamento. Estes tempos dependem do tamanho do conjunto de treinamento e são comparados com os tempos de processamento do *Genome Detective* na seção 5.3.

5.3 Comparação do Tempo de Execução de $AGSSA$ e do Genome Detective

5.3.1 Tempo de Treinamento de $AGSSA$ vs Tempo de Genotipagem do Genome Detective

Comparando o tempo de genotipagem do *Genome Detective* (Genome Detective, 2024) com o tempo de treinamento de $AGSSA$ com conjuntos de treinamento de distintos tamanhos observamos que ambas ferramentas possuem complexidade quadrática no tempo, mas que $AGSSA$ é muito mais eficiente que o Genome Detective.

Nas tabelas 1 e 2 listamos os tempos medidos nos experimentos descritos.

Tabela 1 – Tempos de treinamento do $AGSSA$ em função do tamanho do conjunto de treinamento

número de sequências	tempo (segundos)
49	3,4
97	7,1
196	32,7
391	67,8
1080	284,7
1701	1399,2
3068	4938,9

Tabela 2 – Tempos de genotipagem do *Genome Detective* em função do tamanho do conjunto de treinamento

número de sequências	tempo (segundo)
50	120
100	180
200	240
414	420
600	780
648	840
1000	3045
1085	3120
1179	3300

Na figura 22 plotamos os tempos tabelados e as funções de ajuste quadrático das duas ferramentas. Comparando os coeficientes quadráticos do $AGSSA$ que é aproximadamente $0,00063$ com o do *Genome Detective* que é aproximadamente $0,003$, vemos que o

AGSSA é 4.76 vezes mais rápido no treinamento que o *Genome Detective* na genotipagem. É importante destacar que a diferença entre os tempos se faz mais pronunciada na medida que o número de sequências aumenta. Mais especificamente, o excesso de tempo do *Genome Detective* vem dado por: $0,00237n^2 - 0,26n + 128$ segundos onde n e o número de amostras distintas no conjunto de treinamento. Note (por exemplo) que para $n = 1.085$ sequências o *Genome Detective* precisou 3.120 segundos (52 minutos), enquanto *AGSSA* precisou apenas 285 segundos (4.75 minutos)para treinar o modelo com $n = 1.080$ sequências, ou seja, com apenas 5 sequências a menos.

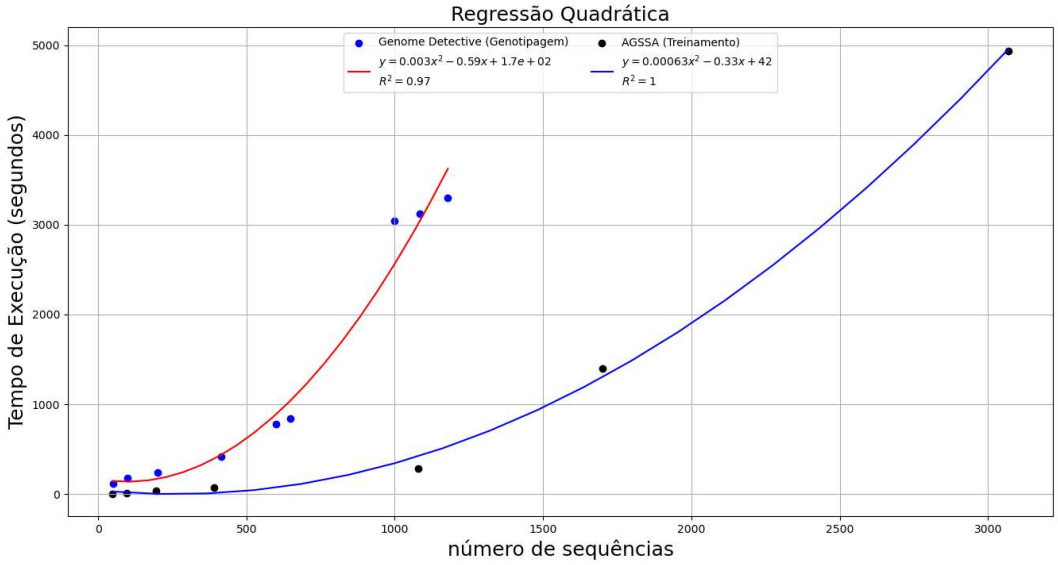


Figura 22 – Comparação dos tempos de treinamento de *AGSSA* e de genotipagem do *Genome Detective* com ajuste quadrático do tempo em função do número de sequências.

Fonte: O Autor

Este resultado demonstra a alta eficiência do algoritmo de AM utilizados no *AGSSA*, o que permite processar grandes volumes de dados genômicos de forma mais rápida e precisa.

5.3.2 Tempo de Classificação de AGSSA

Na seção 5.3.1 vimos que o tempo de genotipagem do *Genome Detective* era quadrático em função do número de sequências no dataset a ser genotipado com um elevado custo computacional dado por $t(n) = 0.003n^2 - 0.59n + 170$. De acordo com isto podemos estimar o número de sequências do gene E do vírus da Dengue que podem ser genotipadas com o *Genome Detective* durante determinado tempo t em segundos usando a solução da equação quadrática, ou seja:

$$n = \frac{0.59 + \sqrt{0.3481 - 0.012(170 - t}}{0.006}$$

Por exemplo, em 1 hora (3.600 segundos), processaria 1.172 sequências, e em 2 horas (7.200 segundos), processaria 1.632 sequências.

A classificação no AGSSA pode ser realizada de forma direta, verificando apenas se já existe no modelo de dados alguma amostra com o mesmo padrão que a amostra de entrada. Essa situação é denominada como melhor caso, devido à sua eficiência. No entanto, quando a amostra não é encontrada no modelo de dados (ou seja, é uma amostra desconhecida), a classificação ocorre utilizando o método CLOPE.

Nesse cenário, a amostra poderá ser incluída em um dos clusters existentes no modelo de classificação ou, em casos raros, formar um novo cluster, caso apresente características significativamente diferentes das amostras do conjunto de treinamento. Esse processo, por exigir maior esforço computacional, é chamado de pior caso.

O AGSSA foi testado em ambos os cenários, avaliando seu desempenho tanto no melhor quanto no pior caso.

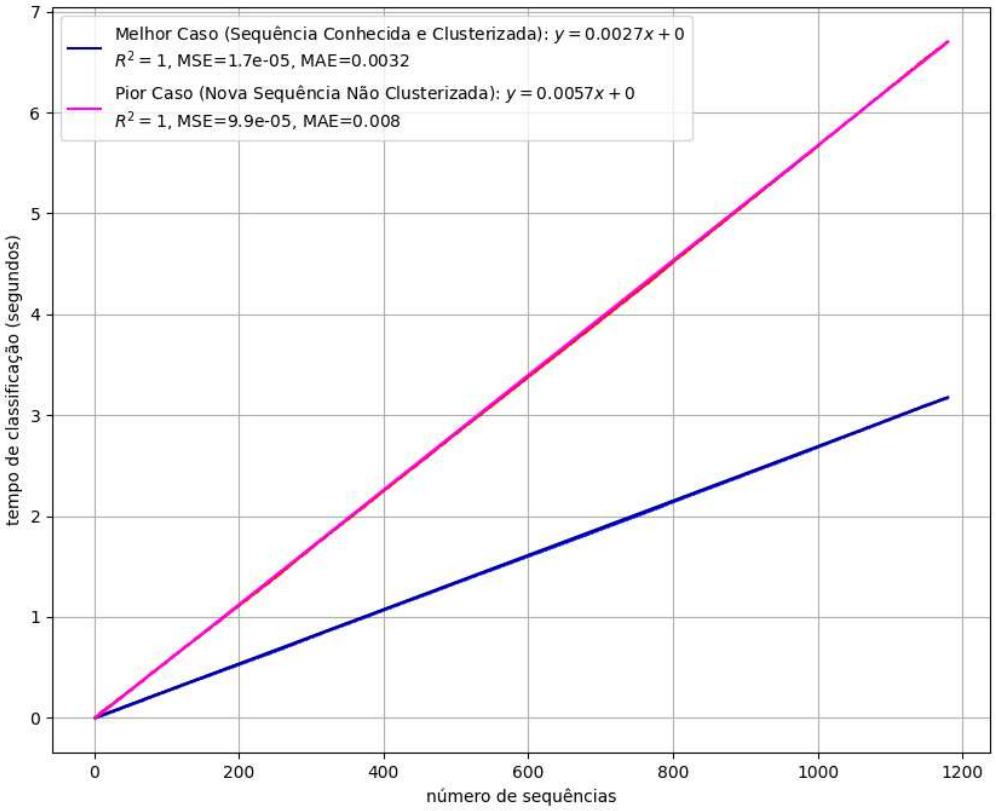


Figura 23 – Variação do melhor (azul) e o pior (vermelho) tempo de classificação de *AGSSA* em função do número de sequências com um modelo de classificação formado por 3.067 amostras distintas do gene E do vírus da Dengue.

Fonte: O Autor

Observamos que a dependência é linear o que significa que há um tempo constante por sequência, neste exemplo 0.0057 segundos no pior caso e 0.0027 segundos no melhor caso[1]. De acordo com isto $AGSSA$ poderia classificar aproximadamente 631.579 sequências numa hora, mesmo no pior caso.

Contudo, como tanto o pior quanto o melhor tempo dependem do tamanho do modelo de dados e do modelo de classificação, realizamos o estudo de como varia o tempo de classificação por sequência do $AGSSA$ em função do tamanho do conjunto de treinamento.

Na figura 24 mostramos o tempo de classificação (genotipagem com AM) por sequência de $AGSSA$ em função do tamanho do modelo de dados (conjunto de treinamento), no eixo horizontal.

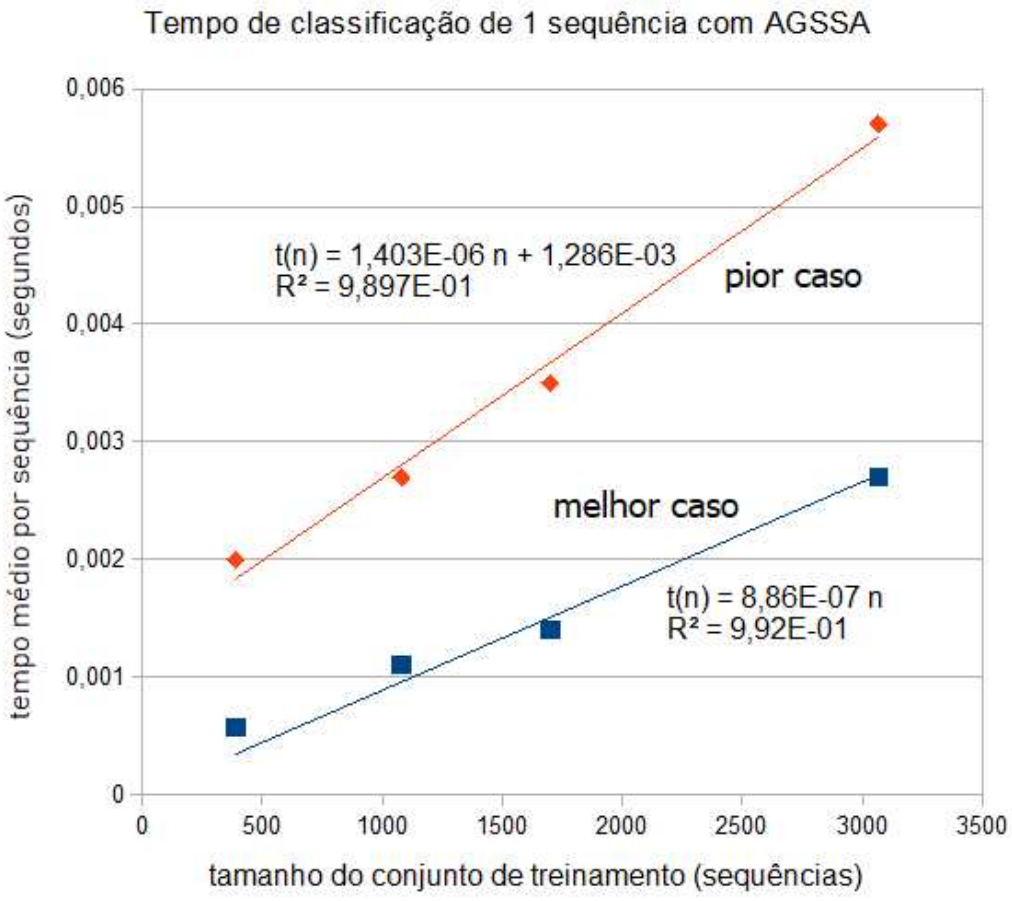


Figura 24 – Variação do melhor e o pior tempo de classificação por sequência de $AGSSA$ em função do número de sequências distintas no conjunto de treinamento.

Fonte: O Autor

Os resultados dos experimentos mostram que o tempo de classificação por sequência de $AGSSA$ varia de forma linear e suavemente (coeficientes angulares da ordem 10^{-7} - 10^{-6} segundos no pior e melhor caso, respectivamente) com o tamanho do conjunto de

[1] Veja os coeficientes angulares na função de ajuste linear no gráfico

treinamento o que é uma característica muito favorável já que permite utilizar grandes modelos sem comprometer a performance de tempo da ferramenta. A vantagem de tolerar modelos grandes fornece a possibilidade de ir construindo modelos de dados com todas as diversas variantes que tenham sido sequenciadas e atualizado com cada nova variante que for aparecendo no tempo.

Os resultados são altamente positivos o que sugere que *AGSSA* é uma ferramenta capaz de processar grandes volumes de dados em curto prazo de tempo e poderá ser uma ferramenta altamente útil no futuro.

6 CONSIDERAÇÕES FINAIS

Os aperfeiçoamentos significativos realizados na ferramenta *AGUA* conduziram à criação de uma nova ferramenta denominada *AGSSA*, que se diferencia substancialmente tanto no seu kernel de processamento (back-end) quanto na sua interface (front-end), que agora é baseada na web. A versão anterior do classificador não supervisionado de sequências codificantes de DNA, utilizando códons, *AGUA*, consistia de um pipeline para ser executado por linha de comando por especialista experiente no uso do sistema. A nova versão desenvolvida, *AGSSA*, integra tecnologias web modernas, que inclui pré-processamento dos dados, gestão de usuários e filas, utilização de broker de mensagens Rabbittmq que lida com as notificações para o email dos usuários, uma api como intermediário na comunicação do front-end com o back-end.

A nova ferramenta foi testada exaustivamente tanto na qualidade dos resultados quanto na sua performance computacional, sendo possível caracterizar a dependência do tempo de treinamento e de classificação em função do tamanho do conjunto de treinamento. Os resultados confirmam que a ferramenta permite lidar com grandes conjuntos de treinamento, formado por dezenas de milhares de amostras, construindo o modelo de aprendizado num tempo inferior a 1 dia num computador com a configuração de hardware como a usada no estudo. Em particular, no caso do gene E da Dengue, estimamos que *AGSSA* poderia treinar um modelo com aproximadamente 12.000 sequências distintas num dia de trabalho.

O fato que o tempo de classificação de *AGSSA* dependa de forma fraca e linear com o tamanho do conjunto de treinamento, permitindo processar uma sequência por segundo com um modelo composto por aproximadamente 713.000 sequências, sugere que *AGSSA* pode ser utilizado com a doble finalidade de Base de Dados da diversidade genotípica das espécies e como ferramenta de genotipagem rápida ao mesmo tempo.

Todos os objetivos traçados e descritos na seção 1.3 foram alcançados. Consideramos relevante fazer aqui uma rápida análise dos resultados alcançados:

1. O algoritmo de treinamento da ferramenta *AGUA* foi modificada de forma que a versão atual do algoritmo de treinamento na ferramenta *AGSSA* é capaz de encontrar de forma autônoma o melhor parâmetro de repulsão, usando a anotação do conjunto de treinamento.

2. Foi feito o desenho, desenvolvido e validado um pipeline do backend para a classificação de novas sequências utilizando os modelos pré-treinados. Esta funcionalidade essencial para o uso da ferramenta foi adicionada à nova ferramenta *AGSSA*.

3. Foi feito o desenho, desenvolvidas e testadas as camadas frontend e intermediária para disponibilizar o uso da ferramenta *AGSSA* na web. Foram utilizadas tecnologias como HTML5, CSS, JavaScript, python, Flask, Celery e RabbitMQ, proporcionando uma interface interativa que facilita a manipulação dos dados genômicos e visualização dos dados gerados recebidos pelo usuário no email. Uma versão preliminar está em fase de testes pré-publicação pela equipe do projeto.

4. Foi aferido o desempenho, tanto na qualidade dos agrupamentos quanto no tempo de execução, da nova versão do algoritmo utilizando sequências anotadas do gene E do vírus da Dengue, e comparado com o *Genome Detective*, ferramenta do estado-da-arte para genotipagem de sequências. *AGSSA* é consideravelmente mais rápido e mais robusto (suporta maiores volumes de dados) do que o *Genome Detective*. Para este estudo foi construído um dataset com 3.067 sequências completas do gene E do vírus da Dengue, altamente representativo contendo as 19 variantes genotípicas conhecidas.

5. Foi realizada uma avaliação da ferramenta desenvolvida identificando-se melhorias para a próxima fase de desenvolvimento. Em particular, sugerimos como trabalho futuro, a extensão do método desenvolvido, para classificar fragmentos de genes já que a versão atual somente permite tratar sequências completas.

Em resumo, os resultados alcançados pelo trabalho representam um avanço significativo na genotipagem viral. A integração de tecnologias web com técnicas avançadas de AM não só aumentou a eficiência e rapidez das análises, mas também permitiu processar grandes volumes de dados, o que possibilita um melhor mapeamento da diversidade genética, não apenas de vírus epidêmicos, mas de bactérias e fungos patógenos.

REFERÊNCIAS

AHMED, M.; SERAJ, R.; ISLAM, S. M. S. The k-means algorithm: A comprehensive survey and performance evaluation. **Electronics**, v. 9, n. 8, 2020. ISSN 2079-9292. Disponível em: https://www.mdpi.com/2079-9292/9/8/1295. Citado na página 28.

ALPAYDIN, E. **Introduction to Machine Learning**. [S.l.]: MIT Press, 2020. Citado na página 27.

ALTSCHUL, S. F. *et al.* Gapped blast and psi-blast: a new generation of protein database search programs. **Nucleic Acids Research**, Oxford University Press, v. 25, n. 17, p. 3389–3402, 1997. Ferramentas como Python podem ser usadas para automatizar análises baseadas em BLAST, armazenando resultados em bancos de dados como PostgreSQL ou MongoDB. Disponível em: https://doi.org/10.1093/nar/25.17.3389. Citado na página 36.

BAILEY, T. L. *et al.* Meme: discovering and analyzing dna and protein sequence motifs. **Nucleic Acids Research**, v. 34, n. $\text{suppl}_2, p.W369 - -W373, 072006.ISSN0305 - 1048.Disponívelem$: . Citado na página 28.

BOECKMANN, B. *et al.* The swiss-prot protein knowledgebase and its supplement trembl in 2003. **Nucleic Acids Research**, Oxford University Press, v. 31, n. 1, p. 365–370, 2003. Disponível em: https://doi.org/10.1093/nar/gkg095. Citado na página 21.

BONETTA, R.; VALENTINO, G. Machine learning techniques for protein function prediction. **Proteins: Structure, Function, and Bioinformatics**, Wiley Online Library, v. 88, n. 3, p. 397–413, 2020. Citado na página 27.

BRODERSEN, K. H. *et al.* The balanced accuracy and its posterior distribution. In: IEEE. **2010 20th International Conference on Pattern Recognition**. [S.l.], 2010. p. 3121–3124. Citado na página 82.

BROOKES, A. J. The essence of snps. **Gene**, Elsevier, v. 234, n. 2, p. 177–186, 1999. Disponível em: https://doi.org/10.1016/S0378-1119(99)00219-X. Citado na página 22.

BUCHFINK *et al.* Sensitive clustering of protein sequences at tree-of-life scale using diamond deepclust. **bioRxiv**, 2023. Citado na página 14.

Celery Project. **Celery Documentation**. 2024. Acesso em: 17 jun. 2024. Disponível em: https://docs.celeryproject.org/. Citado nas páginas 36 e 40.

CHAO, J.; TANG, F.; XU, L. Developments in algorithms for sequence alignment: A review. **Biomolecules**, v. 12, n. 4, p. 546, 2022. Citado na página 33.

CHAPELLE, O.; SCHÖLKOPF, B.; ZIEN, A. **Semi-Supervised Learning**. Cambridge, MA, USA: MIT Press, 2006. (Adaptive Computation and Machine Learning). ISBN 978-0-262-03358-9. Citado na página 26.

CHEN, S.; ZHOU, Y. Genomic analysis of sars-cov-2: current progress and future directions. **Clinical Science**, v. 134, n. 8, p. 1022–1039, 2020. Citado na página 19.

COCK, P. J. *et al.* Biopython: freely available python tools for computational molecular biology and bioinformatics. **Bioinformatics**, Oxford University Press, v. 25, n. 11, p. 1422–1423, 2009. Disponível em: https://doi.org/10.1093/bioinformatics/btp163. Citado nas páginas 21 e 35.

DAVIES; L, D.; BOULDIN, D. W. A cluster separation measure. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, n. 2, p. 224–227, 1979. Citado na página 81.

DELIBAS, E.; ARSLAN, A. Dna sequence similarity analysis using image texture analysis based on first-order statistics. **Journal of Molecular Graphics and Modelling**, v. 99, p. 107603, 2020. ISSN 1093-3263. Disponível em: https://www.sciencedirect.com/science/article/pii/S1093326319305510. Citado na página 83.

Django Software Foundation. **Django - The Web Framework for Perfectionists with Deadlines**. 2023. https://www.djangoproject.com/. Accessed: 26/07/2024. Citado na página 35.

DOMINGO, E.; PERALES, C. Viral quasispecies. **PLoS Pathogens**, Public Library of Science, v. 15, n. 11, p. e1008296, 2019. Disponível em: https://journals.plos.org/plospathogens/article?id=10.1371/journal.ppat.1008296. Citado na página 18.

DUNN, J. C. Well-separated clusters and optimal fuzzy partitions. **Journal of Cybernetics**, Taylor & Francis, v. 4, n. 1, p. 95–104, 1974. Citado na página 82.

FELSENSTEIN, J. Phylogeny as a problem of maximum parsimony. **Systematic biology**, v. 34, n. 4, p. 333–345, 1985. Citado na página 22.

Flask. **Flask Documentation**. 2023. https://flask.palletsprojects.com/en/3.0.x/. Accessed: 30/05/2023. Citado na página 35.

FONSECA, V. *et al.* A computational method for the identification of dengue, zika and chikungunya virus species and genotypes. **PLOS Neglected Tropical Diseases**, Public Library of Science, v. 13, n. 5, p. 1–15, 05 2019. Disponível em: https://doi.org/10.1371/journal.pntd.0007231. Citado na página 60.

GALLAGHER, M. D.; CHEN-PLOTKIN, A. S. The post-gwas era: From association to function. **The American Journal of Human Genetics**, Elsevier, v. 102, n. 5, p. 717–730, 2018. Disponível em: https://doi.org/10.1016/j.ajhg.2018.04.002. Citado na página 27.

Genome Detective. **Genome Detective**. 2024. https://www.genomedetective.com/. Acesso em: 03 jul. 2024. Citado na página 63.

GIBSON, W. *et al.* Genetic recombination between human and animal parasites creates novel strains of human pathogen. **PLoS Neglected Tropical Diseases**, v. 9, n. 3, p. e0003665, Mar 2015. Citado na página 19.

GIL, A. C. **Métodos e técnicas de pesquisa social**. 6. ed. São Paulo: Atlas, 2008. Citado na página 37.

GONZALEZ, A. *et al.* Security in web-based bioinformatics platforms: Challenges and solutions. **Journal of Biomedical Informatics**, 2021. Citado na página 36.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep learning**. [S.l.]: MIT press, 2016. Citado nas páginas 14 e 24.

GREGOR, S.; HEVNER, A. R. Positioning and presenting design science research for maximum impact. **MIS Quarterly**, v. 37, n. 2, p. 337–355, 2013. Disponível em: https://misq.umn.edu/positioning-and-presenting-design-science-research-for-maximum-impact.html. Citado na página 37.

GRUBAUGH *et al.* Making sense of mutation: what d614g means for the covid-19 pandemic remains unclear. **Cell**, v. 182, n. 4, p. 794–795, 2020. Disponível em: https://www.cell.com/cell/fulltext/S0092-8674(20)30812-6. Citado na página 18.

GéRON; AURéLIEN. **Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow**. O'Reilly Media, 2019. Disponível em: https://www.oreilly.com/library/view/hands-on-machine-learning-with-scikit-learn-keras-and-tensorflow/9781492038262/. Citado nas páginas 24, 80, 81 e 82.

HADFIELD, J. *et al.* Nextstrain: real-time tracking of pathogen evolution. **Bioinformatics**, v. 34, n. 23, p. 4121–4123, 2018. Disponível em: https://pubmed.ncbi.nlm.nih.gov/29790939/. Citado na página 24.

HOWLEY, P. M.; KNIPE, D. M.; ENQUIST, L. W. **Fields Virology**. 7th. ed. Lippincott Williams & Wilkins, 2021. 2-6 p. ISBN 9781975112547. Disponível em: https://shop.lww.com/Fields-Virology--7th-Edition/p/9781975112547. Citado na página 18.

HUBERT; LAWRENCE; ARABIE, P. Comparing partitions. **Journal of Classification**, Springer, v. 2, n. 1, p. 193–218, 1985. Citado na página 82.

JACCARD, P. The index of similarity and its use in the classification of binary data. **Bulletin of the New York Academy of Sciences**, v. 7, p. 203–210, 1901. Citado na página 61.

JAMES, G. *et al.* **An introduction to statistical learning**. [S.l.]: Springer, 2013. Citado nas páginas 80, 81 e 82.

JOHNSON, A. D. An extended iupac nomenclature code for polymorphic nucleic acids. **Bioinformatics**, Oxford University Press, England, v. 26, n. 10, p. 1386–1389, May 2010. Research Support, N.I.H., Extramural; Research Support, Non-U.S. Gov't. Disponível em: https://doi.org/10.1093/bioinformatics/btq098. Citado nas páginas 31 e 86.

KARAMIZADEH, S. *et al.* An overview of principal component analysis. **Journal of Signal and Information Processing**, v. 4, n. 3B, p. 173–175, 2013. Disponível em: https://doi.org/10.4236/jsip.2013.43B031. Citado na página 28.

KATOH, K.; STANDLEY, D. M. Mafft multiple sequence alignment software version 7: improvements in performance and usability. **Molecular Biology and Evolution**, Oxford University Press, v. 30, n. 4, p. 772–780, 2013. Disponível em: https://doi.org/10.1093/molbev/mst010. Citado na página 22.

KOONIN, E.; DOLJA, V.; KRUPOVIC, M. Origins and evolution of viruses of eukaryotes: The ultimate modularity. **Virology**, Elsevier, v. 529, p. 161–171, 2019. Disponível em: https://www.sciencedirect.com/science/article/pii/S0042682219300080. Citado na página 18.

KUCHARSKI, A. J. *et al.* Delaying the spread of covid-19 by public health interventions: a modelling study. **The Lancet**, Elsevier, v. 395, n. 10235, p. e012244, 2020. Disponível em: https://www.thelancet.com/journals/lancet/article/PIIS0140-6736(20)31064-5/fulltext. Citado na página 12.

KUMAR, S. *et al.* Timetree: A resource for timelines, timetrees, and divergence times. **Molecular Biology and Evolution**, Oxford University Press, v. 34, n. 7, p. 1812–1819, 2017. Disponível em: https://doi.org/10.1093/molbev/msx116. Citado na página 22.

LESPINET, O.; MéDIGUE, C.; LAZAREVIć, V. The role of bioinformatics in microbial ecology. **Microbiological Research**, Elsevier, v. 257, p. 106818, 2021. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S0960852414016526. Citado na página 13.

LI, H. Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences. **Bioinformatics**, Oxford University Press, v. 32, n. 14, p. 2103–2110, 2016. Disponível em: https://doi.org/10.1093/bioinformatics/btw152. Citado na página 22.

LI, Y. *et al.* Deep learning in bioinformatics: Introduction, application, and perspective in the big data era. **Methods**, v. 166, p. 4–21, 2019. ISSN 1046-2023. Deep Learning in Bioinformatics. Disponível em: https://www.sciencedirect.com/science/article/pii/S1046202318303256. Citado na página 27.

LöCHEL, H.; HEIDER, D. Chaos game representation and its applications in bioinformatics. **Computational Structural Biotechnology Journal**, v. 19, n. 1, p. 6263–6271, 2021. Citado na página 83.

MEENA, N.; KUMAR, P.; TOMAR, V. S. Big data and bioinformatics in the era of covid-19: A comprehensive review. **Briefings in Bioinformatics**, p. bbab278, 2021. Disponível em: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10568291/. Citado na página 13.

MENEZES, M. S. **Ferramenta Computacional para Estudo da Evolução de Espécies Virais Baseado no Uso de Códons**. Dissertação (Mestrado) — Universidade do Estado da Bahia, Salvador, Bahia, Brasil, December 2023. Monografia apresentada ao curso de Sistemas de Informação do Departamento de Ciências Exatas e da Terra da Universidade do Estado da Bahia - UNEB, como requisito parcial à obtenção do grau de bacharel em Sistemas de Informação. Área de Concentração: Ciência da Computação. Citado nas páginas 23, 25, 26, 31, 34, 84 e 88.

METZKER; L., M. Sequencing technologies—the next generation. **Nature Reviews Genetics**, v. 11, n. 1, p. 31–46, 2010. Disponível em: https://doi.org/10.1038/nrg2626. Citado na página 12.

METZKER, M. L. A comprehensive overview of next-generation sequencing: Applications, methods, and challenges. **Genome Research**, Cold Spring Harbor Laboratory Press, v. 20, n. 11, p. 1531–1544, 2010. Disponível em: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2910785/. Citado na página 19.

Mozilla Developer Network. **JavaScript Guide for Web Development**. 2023. https://developer.mozilla.org/en-US/docs/Web/JavaScript. Accessed: 27/07/2024. Citado na página 35.

MULLIS, K. B.; FALOONA, F. A. Specific enzymatic amplification of dna in vitro: the polymerase chain reaction. **Cold Spring Harbor Symposia on Quantitative Biology**, Cold Spring Harbor Laboratory Press, v. 51, p. 263–273, 1987. Disponível em: https://doi.org/10.1101/SQB.1986.051.01.032. Citado na página 20.

OECD. Artificial intelligence, data and competition. **OECD Artificial Intelligence Papers**, OECD Publishing, Paris, n. 18, 2024. Citado na página 13.

OLSON, M. D.; SCHAFFER, M. P.; SHORESH, N. The future of bioinformatics infrastructure. **Nature Genetics**, Nature Publishing Group, v. 55, n. 1, p. 1–5, 2023. Disponível em: https://www.nature.com/articles/s41598-023-32555-y. Citado na página 13.

OLSON, R. D. *et al.* Introducing the bacterial and viral bioinformatics resource center (bv-brc): a resource combining patric, ird and vipr. **Nucleic Acids Research**, v. 51, n. D1, p. D678–D689, 2023. Published 2023-01-06. Disponível em: https://doi.org/10.1093/nar/gkac1003. Citado na página 24.

ORGANIZATION, W. H. Covid-19. World Health Organization, 2020. Disponível em: https://www.who.int/health-topics/covid-19. Citado na página 12.

PADMANABHAN, V.; HEINEN, C.; CHOCKALINGAM, S. P. Bioinformatic approaches to combat covid-19: A comprehensive review. **Genomics**, Elsevier, v. 124, n. 1, p. 101329, 2022. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S0197245605800194. Citado na página 13.

PEFFERS, K. *et al.* Design science research: Advancing a paradigm. In: **Proceedings of the 2018 International Conference on Information Systems (ICIS)**. [s.n.], 2018. Disponível em: https://aisel.aisnet.org/icis2018/. Citado na página 37.

PEFFERS, K. *et al.* A design science research methodology for information systems research. **Journal of management information systems**, Taylor & Francis, v. 24, n. 3, p. 45–77, 2007. Citado na página 37.

PETROSILLO, G. *et al.* Covid-19, sars-cov-2 and the first 50 days. **Journal of Infection**, Elsevier, v. 80, n. 2, p. 252–255, 2020. Disponível em: https://www.sciencedirect.com/science/article/pii/S0163445X20301474. Citado na página 12.

PICKETT, B. E. *et al.* ViPR: an open bioinformatics database and analysis resource for virology research. **Nucleic Acids Research**, v. 40, n. Database issue, p. D593–D598, 2012. Published 2012-01-01. Disponível em: https://doi.org/10.1093/nar/gkr859. Citado na página 24.

PLATZER, A. Visualization of snps with t-sne. **PLOS ONE**, Public Library of Science, v. 8, n. 2, p. 1–6, 02 2013. Disponível em: https://doi.org/10.1371/journal.pone.0056883. Citado na página 28.

POP, M.; KOSACK, D. S.; SALZBERG, S. L. Hierarchical scaffolding with bambus. **Genome Research**, Cold Spring Harbor Lab, v. 14, n. 1, p. 149–159, 2004. Disponível em: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC314292/. Citado na página 21.

Python. **Python Programming Language**. 2023. https://www.python.org/. Accessed: 30/05/2023. Citado na página 35.

RabbitMQ. **RabbitMQ Documentation**. 2024. Acesso em: 17 jun. 2024. Disponível em: https://www.rabbitmq.com/documentation.html. Citado nas páginas 36, 40 e 41.

RAMBAUT *et al.* Genomic epidemiology of covid-19: the first 100,000 genomes. **Nature**, v. 581, n. 7808, p. 286–291, 2020. Citado na página 22.

RAMBAUT, A. *et al.* Nextstrain: Real-time tracking of pathogen evolution. **Virus Evolution**, 2020. Citado na página 36.

RAMIREZ-MARTíNEZ, G. Editorial: Reviews in virus and host. **Frontiers in Cellular and Infection Microbiology**, v. 14, 2024. ISSN 2235-2988. Disponível em: https://www.frontiersin.org/journals/cellular-and-infection-microbiology/articles/10.3389/fcimb.2024.1445721 Citado na página 18.

RANA, T. Bioinformatic approaches for the analysis of complex biological data. **Advances in Computational Biology and Bioinformatics**, Springer, Singapore, 2023. Disponível em: https://link.springer.com/chapter/10.1007/978-1-0716-3461-5_11. Citado na página 12.

RESTOVIC, M. I. *et al.* **Arthropod Borne Virus database - ABVdb**. 2018. http://abvdb.uneb.br. Banco de dados contendo informações filogenéticas, epidemiológicas e clínicas de sequências de arbovírus, incluindo os vírus da Dengue, Zika e Chikungunya. Desenvolvido pela FIOCRUZ, Salvador-Bahia, Brasil. Disponível em: http://abvdb.uneb.br. Citado na página 57.

ROMAN-NARANJO, P.; PARRA-PEREZ, A.; LOPEZ-ESCAMEZ, J. A systematic review on machine learning approaches in the diagnosis and prognosis of rare genetic diseases. **Journal of Biomedical Informatics**, v. 143, p. 104429, 2023. ISSN 1532-0464. Disponível em: https://www.sciencedirect.com/science/article/pii/S1532046423001508. Citado na página 28.

ROUSSEEUW, P. J. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. **Journal of Computational and Applied Mathematics**, Elsevier, v. 20, p. 53–65, 1987. Citado na página 81.

SAITOU, N.; NEI, M. The neighbor-joining method: a new method for reconstructing phylogenetic trees. **Molecular Biology and Evolution**, Oxford University Press, v. 4, n. 4, p. 406–425, 1987. Disponível em: https://doi.org/10.1093/oxfordjournals.molbev.a040454. Citado na página 23.

SAITOU, N.; NEI, M. Inferência filogenética: conceitos básicos e suas aplicações em virologia e epidemiologia molecular. **Acta Veterinaria Scandinavica**, v. 44, n. 4, p. 406–425, 2013. Disponível em: https://www.ufrgs.br/actavet/44/PUB%201392.pdf. Citado na página 23.

SANTALUCIA, J.; HICKS, D. The thermodynamics of dna structural motifs. **Annual Review of Biophysics**, Annual Reviews, v. 33, n. Volume 33, 2004, p. 415–440, 2004. ISSN 1936-1238. Disponível em: https://www.annualreviews.org/content/journals/10.1146/annurev.biophys.32.110601.141800. Citado na página 83.

SAúDE, O. M. da. **Dengue**. 2023. 1-14 p. Disponível em: ORGANIZAÃĞÃČOMUNDIALDASAÃŽDE. https://www.who.int/health-topics/dengue-and-severe-dengue.Acessoem:17mai.2024. Citado na página 19.

SHU, Y.; MCCAULEY, J. Gisaid: Global initiative on sharing all influenza data – from vision to reality. **Euro Surveillance**, v. 22, n. 13, p. 30494, 2017. Disponível em: https://www.eurosurveillance.org/content/10.2807/1560-7917.ES.2017.22.13.30494. Citado na página 36.

SIEVERS, F. *et al.* Fast, scalable generation of high-quality protein multiple sequence alignments using clustal omega. **Molecular Systems Biology**, Wiley Online Library, v. 7, n. 1, p. 539, 2011. Disponível em: https://doi.org/10.1038/msb.2011.75. Citado nas páginas 22 e 36.

SINGER, J. *et al.* Cov-glue: A web application for tracking sars-cov-2 genomic variation. **Preprints**, Preprints, June 2020. Disponível em: https://doi.org/10.20944/preprints202006.0225.v1. Citado na página 36.

SOKAL, R. R.; MICHENER, C. D. A statistical method for evaluating systematic relationships. **University of Kansas Science Bulletin**, v. 38, p. 1409–1438, 1958. Citado na página 61.

SUCHARD, M. A. *et al.* Bayesian phylogenetic and phylodynamic data integration using beast 1.10. **Virus Evolution**, Oxford University Press, v. 4, n. 1, p. vey016, 2018. Disponível em: https://doi.org/10.1093/ve/vey016. Citado na página 24.

VILSKER, M. *et al.* Genome Detective: an automated system for virus identification from high-throughput sequencing data. **Bioinformatics**, v. 35, n. 5, p. 871–873, 08 2018. ISSN 1367-4803. Disponível em: https://doi.org/10.1093/bioinformatics/bty695. Citado nas páginas 36, 42, 53 e 57.

VILSKER, M. *et al.* Genome detective: an automated system for virus identification from high-throughput sequencing data. **Bioinformatics**, Oxford University Press, v. 35, n. 5,

p. 871–873, 2019. Disponível em: https://doi.org/10.1093/bioinformatics/bty695. Citado na página 24.

WACKER *et al.* High-throughput analysis of gene expression by quantitative reverse transcription-pcr. **Biotechniques**, Future Science Ltd London, UK, v. 32, n. 2, p. 137–144, 2002. Disponível em: https://pubmed.ncbi.nlm.nih.gov/33765310/. Citado na página 20.

WAGNER, E. *et al.* **Basic Virology**. 4th. ed. Wiley-Blackwell, 2021. 8-12 p. ISBN 9781119580539. Disponível em: https://www.wiley.com/en-us/Basic+Virology%2C+4th+Edition-p-9781119580539. Citado na página 19.

WAHAB, A. *et al.* Dna sequences performs as natural language processing by exploiting deep learning algorithm for the identification of n4-methylcytosine. **Scientific Reports**, v. 11, n. 1, p. 212, 2021. Citado na página 83.

WALLS, A. C. *et al.* Structure and function of the sars-cov-2 spike protein. **Science**, American Association for the Advancement of Science, v. 369, n. 6505, p. eabb4553, 2020. Disponível em: https://science.sciencemag.org/content/369/6505/eabb4553. Citado na página 12.

WANG, Z. *et al.* Fusang: a framework for phylogenetic tree inference via deep learning. **Nucleic Acids Research**, v. 51, n. 20, p. 10909–10923, 10 2023. ISSN 0305-1048. Disponível em: https://doi.org/10.1093/nar/gkad805. Citado na página 27.

WILKINSON, M. C. *et al.* Bioinformatics for viral evolution. **Molecular Biology and Evolution**, Oxford University Press, v. 40, n. 1, p. msac101, 2023. Disponível em: https://academic.oup.com/mbe/article/40/1/msac101. Citado na página 13.

YANG, Y.; GUAN, X.; YOU, J. Clope: a fast and effective clustering algorithm for transactional data. In: **Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining**. [S.l.: s.n.], 2002. p. 682–687. Citado na página 87.

YU, H.; YAU, S. The optimal metric for viral genome space. **Computational Structural Biotechnology Journal**, v. 23, n. 1, p. 2083–2096, 2024. Citado na página 83.

ZHANG, D. *et al.* Phylosuite: An integrated and scalable desktop platform for streamlined molecular sequence data management and evolutionary phylogenetics studies. **Molecular Ecology Resources**, Wiley Online Library, v. 20, n. 1, p. 348–355, 2020. Disponível em: https://doi.org/10.1111/1755-0998.13096. Citado na página 24.

ZHANG, Z.; CHEUNG, K.-H.; TOWNSEND, J. P. Bringing web 2.0 to bioinformatics. **Briefings in Bioinformatics**, v. 10, n. 1, p. 1–10, 10 2008. ISSN 1467-5463. Disponível em: https://doi.org/10.1093/bib/bbn041. Citado na página 35.

ZHOU, X. *et al.* Explainable machine learning in bioinformatics. **Nature Biotechnology**, 2019. Citado na página 30.

ZHU, N. *et al.* A novel coronavirus from patients with pneumonia in china, 2019. **The New England Journal of Medicine**, Massachusetts Medical Society, v. 382, n. 13, p. 1089–1099, 2020. Disponível em: https://www.nejm.org/doi/full/10.1056/NEJMoa2001316. Citado na página 12.

APÊNDICE A – TERMOS E CONCEITOS RELEVANTES NO AMPRENDIZADO DE MÁQUINA(AM)

Nesta seção descrevemos brevemente termos e conceitos que são altamente relevantes na área de Inteligência Artificial e em particular no Aprendizado de Máquina para a Classificação de Objetos:

- Exemplo (padrão, instância, amostra): Uma unidade de informação a partir da qual um modelo será aprendido ou utilizado. Na maioria dos casos, exemplos são descritos por vetores de características (Géron; Aurélien, 2019).
- Característica (atributo, variável): Uma propriedade que descreve um exemplo. Cada atributo possui um domínio definido por seu tipo, que determina os valores que ele pode assumir. Além disso, os atributos podem ser numéricos (representados por variáveis inteiras ou reais) ou categóricos (representados por caracteres ou cadeias de caracteres - *strings*) (James *et al.*, 2013).
- Vetor de características: Uma lista de características que descreve um exemplo (Géron; Aurélien, 2019).
- Classe: No aprendizado supervisionado, cada exemplo possui um atributo especial chamado rótulo ou classe, que indica a categoria à qual ele pertence (James *et al.*, 2013).
- Conjunto de exemplos (conjunto de dados, conjunto de treinamento): Um conjunto de exemplos com seus respectivos valores de atributos. No aprendizado supervisionado, cada exemplo também possui um rótulo associado (James *et al.*, 2013).
- Conjunto de treinamento desbalanceado: Quando o número de exemplos de algumas classes é muito maior que o de outras classes (James *et al.*, 2013).
- Ruído: Imperfeições nos dados, como erros de medição ou rotulagem incorreta (Géron; Aurélien, 2019).
- *Overfitting* (superajuste): Ocorre quando o modelo se adapta excessivamente aos dados de treinamento, apresentando baixo desempenho em novos dados (Géron; Aurélien, 2019).
- Classificação Incorreta: Um exemplo de determinada classe A foi classificado como de outra classe qualquer(James *et al.*, 2013).

- Exemplo Não Classificado: Um exemplo de determinada classe A que não foi classificado como nenhuma das classes existentes, nem mesmo da classe à qual aparentemente pertence (James *et al.*, 2013). Geralmente representa um caso de ruído por anotação errada ou por atributos incorretos.

- Métricas de Desempenho para Classificadores Supervisionados Binários (índices de qualidade)

  - Falso positivo: Um exemplo da classe B (negativa) classificado como da classe A (positiva) (James *et al.*, 2013).
  - Falso negativo: Um exemplo da classe A (positiva) classificado como da classe B (negativa) (James *et al.*, 2013).
  - Acurácia: A proporção de previsões corretas feitas pelo modelo em um conjunto de dados (Géron; Aurélien, 2019).
  - Precisão (*Precision*): A proporção de verdadeiros positivos entre os exemplos classificados como positivos (James *et al.*, 2013).
  - Revocação (*Recall*): A proporção de verdadeiros positivos entre os exemplos que são realmente positivos (James *et al.*, 2013).
  - F1-Score: A média harmônica entre precisão e revocação, proporcionando um equilíbrio entre os dois (James *et al.*, 2013).
  - Curva ROC e AUC: A curva ROC (*Receiver Operating Characteristic*) representa a relação entre a taxa de verdadeiros positivos e a taxa de falsos positivos. A área sob a curva (AUC) é uma medida do desempenho do modelo (Géron; Aurélien, 2019).

- Métricas de Desempenho para Aprendizado Não Supervisionado: As métricas de desempenho para aprendizado não supervisionado, como a análise de clusters, são usadas para avaliar a qualidade dos agrupamentos sem rótulos de classe pré-definidos.

  - Índice de Silhueta: Mede o quão semelhante um exemplo é ao seu próprio cluster (coesão) em comparação com outros clusters (separação). Varia de -1 a 1, onde valores altos indicam que os exemplos estão bem correspondidos ao seu próprio cluster e mal correspondidos a clusters vizinhos (Rousseeuw, 1987).
  - Índice de Davies-Bouldin: Calcula a média das razões das distâncias dentro do cluster para a distância entre os clusters. Valores mais baixos indicam melhores separações entre os clusters (Davies; L; Bouldin, 1979).
  - Coeficiente de Variação Intracluster (WCV): Mede a dispersão dentro dos clusters. Valores menores indicam clusters mais compactos.

  - Índice de Rand Ajustado (ARI): Avalia a similaridade entre a clusterização obtida e uma clusterização de referência, corrigida pelo acaso. Valores próximos de 1 indicam alta similaridade (Hubert; Lawrence; Arabie, 1985).
  - Índice de Dunn: Mede a menor distância entre os pontos de diferentes clusters dividida pela maior distância intracluster. Valores mais altos indicam melhor clusterização (Dunn, 1974).

- Métricas de Desempenho para Aprendizado Supervisionado com Múltiplas Classes: As métricas de desempenho para aprendizado supervisionado com múltiplas classes são usadas quando os dados de treinamento possuem rótulos de múltiplas classes.

  - Acurácia Global: A proporção de previsões corretas feitas pelo modelo em um conjunto de dados com múltiplas classes (Géron; Aurélien, 2019).
  - Precisão por Classe: A proporção de verdadeiros positivos entre os exemplos classificados como positivos para cada classe individualmente (James *et al.*, 2013).
  - Revocação por Classe: A proporção de verdadeiros positivos entre os exemplos que são realmente positivos para cada classe individualmente (James *et al.*, 2013).
  - F1-Score por Classe: A média harmônica entre precisão e revocação para cada classe (James *et al.*, 2013).
  - Matriz de Confusão: Uma tabela que descreve o desempenho do modelo mostrando as previsões corretas e incorretas de cada classe (Géron; Aurélien, 2019).
  - Acurácia Balanceada: A média da acurácia para cada classe, útil para conjuntos de dados desbalanceados (Brodersen *et al.*, 2010).
  - Micro/macro/média ponderada: Diferentes formas de calcular a média de precisão, revocação e F1-Score. A média micro agrega todas as classes como uma, a média macro calcula a média simples entre as classes, e a média ponderada leva em consideração o número de exemplos em cada classe (Géron; Aurélien, 2019).

APÊNDICE B – MÉTODOS DE EXTRAÇÃO DE ATRIBUTOS NUMÉRICOS DE SEQUÊNCIAS DE DNA

A representação de sequências de DNA é um passo fundamental para diversos estudos em bioinformática, permitindo a análise, comparação e interpretação de dados genéticos. Superando a limitação da sequência básica de bases nitrogenadas (A, C, G e T), diversas abordagens alternativas surgem para converter as sequências em formatos mais adequados para técnicas de AM e análise estatística.

Na revisão bibliográfica identificamos cinco abordagens principais:

1. **Genome Image Pattern (GIP)** (Delibas; Arslan, 2020): Transformao DNA em Imagens Unidimensionais, convertendo cada base nitrogenada em um pixel com cor específica. A imagem é posteriormente analisada com redes convolucionais unidimensionais (1D CNNs).

2. **Chaos Game Representation (CGR)** (Löchel; Heider, 2021): O CGR utiliza a geometria fractal do Chaos Game para gerar uma representação de sequências de DNA como uma nuvem de pontos num plano ou num cubo de projeção. A sequência é percorrida e cada base nitrogenada é mapeada para um ponto nessa área ou volume de projeção, criando padrões fractais visíveis que podem ser armazenados para comparação futura com outras sequências.

3. **Natural Vector Representation (NVR)** (Yu; Yau, 2024): A NVR transforma sequências de DNA em vetores numéricos contando a frequência de bases e k-mers (subsequências de tamanho k) em cada posição da sequência. Essa representação captura informações sobre a composição local e regional da sequência, permitindo a comparação e análise usando técnicas de AM e estatística.

4. **Word2vec e Doc2vec (W2V)** (Wahab *et al.*, 2021): Inspiradas no processamento de linguagem natural, técnicas como Word2vec e Doc2vec convertem sequências de DNA em vetores numéricos. Essas técnicas consideram a ordem e a proximidade das bases nitrogenadas, capturando relações semânticas entre elas e permitindo a identificação de similaridades funcionais entre sequências.

5. **Propriedades Termodinâmicas (THP)** (SantaLucia; Hicks, 2004): A THP, baseada em propriedades termodinâmicas, considera as características energéticas das ligações entre bases nitrogenadas. Essas propriedades, como energia livre de Gibbs e entropia, fornecem informações sobre a estabilidade da molécula de DNA e a probabilidade de certos padrões de sequências ocorrerem.

A escolha da melhor representação de sequências de DNA depende do contexto específico da análise e dos objetivos do estudo. É importante considerar fatores como:

- A natureza da tarefa (classificação, detecção de genes, análise de expressão gênica etc.)
- O tamanho e a complexidade dos dados de sequenciamento
- A disponibilidade de recursos computacionais

Além disso, diferentes representações podem ser combinadas para capturar informações complementares sobre as sequências de DNA. A pesquisa em bioinformática segue explorando novas técnicas para representação de sequências, ampliando o arsenal de ferramentas disponíveis para desvendar os segredos do código genético.

Como dito anteriormente, neste trabalho não extraímos atributos numéricos senão categóricos (simbólicos) utilizando a abordagem CBUC (Codon-usage Based Unsupervised Classification) (Menezes, 2023), que é descrita na seção 2.5.1.

APÊNDICE C – TRANSFORMAÇÃO DE SEQUÊNCIAS DE NUCLEOTÍDEOS EM VETORES DE ATRIBUTOS BASEADOS EM CÓDONS

C.1 Conversão das Sequências de Nucleotídeos a Sequências de Códons - $CBUC$

Na figura 4, que descreve a diversidade de pipelines e abordagens para genotipagem, incluímos um passo chamado $CBUC$, no ramo correspondente aos atributos categóricos. Nesta seção detalhamos o que ocorre nesse passo de processamento.

O requisito para iniciar o processo é um conjunto de sequências: 1) codificantes, 2) de alta qualidade, 3) alinhadas, 4) no quadro de leitura +1 e, 5) do mesmo tamanho.

Para simplificar a notação dos códons, atribuímos a cada triplete de nucleotídeos de cada sequência do dataset de treinamento, alinhada com a sequência de referência no quadro de leitura, um número inteiro de 1 a 64, mas que é usado como atributo categórico (numeral), não como número em si.

Isto é feito em duas etapas: Primeiro fazemos uma numeração dos nucleotídeos $N = \{A, G, C, T\}$ da forma: $n[A] = 1, n[G] = 2, n[C] = 3$ a $n[T] = 4$.

Um códon vem dado pelas bases nas três posições: $C = N_1\ N_2\ N_3$, onde $N_i \in \{A, G, C, T\}$ é a base na posição $i = 1, 2, 3$, do códon.

O segundo passo é transformar cada nucleotídeo do códon para sua versão numerada, ou seja $n[C] = [n(N_1), n(N_2), n(N_3)]$.

Por último, com $n[C]$ dado atribuímos ao códon um numeral $c \in [1, 64]$ usando a fórmula seguinte:

$$c = n(N_3) + 4(n(N_2) - 1) + 16(n(N_1) - 1) \tag{C.1}$$

Exemplos:

- Dado o códon $C = AAA$. Então $n[C] = [n(A), n(A), n(A)] = [1, 1, 1]$, pelo que o numeral atribuído é $c = 1 + 4(1 - 1) + 16(1 - 1) = 1$
- Dado o códon $C = TTT$. Então $n[C] = [n(T), n(T), n(T)] = [4, 4, 4]$, pelo que o numeral atribuído é $c = 4 + 4(4 - 1) + 16(4 - 1) = 64$

Usando a fórmula C.1 transformamos qualquer sequência codificante de L nucleotídeos numa sequência de $L/3$ códons numerados de 1 a 64.

Incluir sequências redundantes no conjunto de treinamento apenas aumenta o volume de dados e o tempo de processamento, sem introduzir nenhuma informação relevante, pelo que sequências idênticas devem ser filtradas.

'Por outro lado, como as sequências codificantes não podem possuir códons de parada no quadro de leitura, se for encontrado algum desses códons (TAA, TAG ou TGA, que correspondem aos numerais $49, 50$ e 53, respectivamente) a sequência é excluída.

Outro aspecto importante na transformação de uma sequência de DNA a uma lista de features é o tratamento adequado dos "gaps". É comum que existam inserções e deleções de códons nas sequências alinhadas. Quando um códon é inserido em uma das sequências, isso causa o surgimento de gaps, representados com [—] nas outras sequências alinhadas. Da mesma forma, quando um códon é deletado de uma sequência, aparece um gap [—] na posição do códon deletado nessa sequência. Neste trabalho atribuímos o numeral 65 aos gaps detectados.

Da mesma forma é necessário tratar nucleotídeos indeterminados. Chamamos nucleotídeos indeterminados a qualquer caráter na sequência de DNA que não seja A, G, C ou T(U). O mais comum é o carácter N que indica qualquer nucleotídeo e ocorre quando a base não foi identificada corretamente durante o sequenciamento. Outros caráteres representam indeterminações menos fortes, como por exemplo R significa purina, que pode ser A ou G, e Y representa uma pirimidina, ou seja, C ou T. A lista de caracteres distintos de A, G, C, T, é definida pela notação IUPAC (Johnson, 2010).

Em qualquer situação, é necessário definir uma política de aceitação/rejeição de sequências com esses caráteres. Incluir sequências com caracteres indeterminados no conjunto de treinamento é factível, mas introduz ruído diminuindo a qualidade classificatória do modelo treinado.

A regra deve ser apenas considerar sequências sem caracteres indeterminados na hora de formar o conjunto de treinamento. Contudo, se por causa de falta de sequências for decidido considerar sequências com caráteres indeterminados, os códons que contem bases indeterminadas devem ser eliminados de todo o alinhamento, deixando apenas sítios onde todos os códons estão determinados. Isto logicamente complica tanto a atualização/ampliação do conjunto de treinamento quanto o pre-processamento das sequências que serão classificadas usando o modelo treinado.

Na seção C.2 descrevemos o procedimento para extração de atributos a partir das sequências traduzidas para códons que formam o conjunto de treinamento.

C.2 Extração de Atributos das Sequências de Códons

Denotemos por $\mathcal{S}_n$ o conjunto de treinamento formado por N sequências com L nucleotídeos alinhados no quadro de leitura, e por $\mathcal{S}_c$ a versão do conjunto de treinamento formado por N sequências com $M = L/3$ códons, tal que $\mathcal{S}_c[i, j]$ denota o códon na sequência $i = 1, 2, \ldots, N$ na posição $j = 1, 2, \ldots, M$.

Lembrando que os códons são numerais de 1 a 65 (excepto códons de parada: 49, 50 e 53), ou seja, $\mathcal{S}_c[i, j] \in \{1, 2, \ldots, 48, 51, 52, 54, \ldots, 65\}$ para toda sequência i e posição j.

O processo de extração é bem simples, consistindo na varredura de todas as posições de esquerda a direita, identificando posições polimórficas, ou seja, onde há códons distintos.

Os atributos tem o formato $[posição : valor]$ que é compatível com o algoritmo de classificação utilizado CLOPE (Yang; Guan; You, 2002). O algoritmo Python utilizado é descrito a seguir:

Listing C.1 – Finding polymorphic positions and generating the training set for CLOPE

```
# Finding polymorphic positions
pos = [] # Initializing list of polymorphic positions
for j in range(M): # Loop through positions
    if len(set(Sc[:,j])) > 1: # if there is more than one codon type
        pos.append(j) # add position j to the list

# Generating the training set for CLOPE
T = [] # Initializing the training dataset (list of lists of attributes)
for i in range(N): # Loop through sequences
    t = [] # Initializing the attribute list for sequence i
    for p in pos: # loop over polymorphic positions
        t.append(str(p) + ":" + str(Sc[i,p])) # Updating the attribute
            list of sequence i
    T.append(t) # Updating the training set
```

APÊNDICE D – CONSTRUÇÃO DO MODELO DE APRENDIZADO DE MÁQUINA *AGUA* (***ADVANCED GENOTYPING WITH UNSUPERVISED ALGORITHM***)

Nesta seção descrevemos o modelo desenvolvido em (Menezes, 2023), que foi o ponto de partida deste projeto. O modelo consta de duas partes: (1) modelo de dados e (2) modelo de classificação.

O modelo de dados consiste numa base de amostras diferenciadas com atributos relevantes, extraídos de conjuntos de treinamento, o qual pode(deve) ir sendo atualizado(ampliado) durante o uso da ferramenta. O modelo de dados pode ser visto como um inventário da diversidade das espécies estudadas, pelo que pode(deve) ser indo atualizado na medida que novas amostras são coletadas. Conceitualmente o modelo de dados é uma base de conhecimento específica (*KDB - Knowledge Data Base*) para cada espécie estudada. Esta base pode(deve) ser usada como referência para busca de similaridades de amostras futuras.

Na pesquisa da evolução viral, geralmente trabalhamos com genes alvo que definem o genótipo das cepas das espécies virais em estudo, como por exemplo, o gene que codifica a proteína spike do vírus SARS-CoV-2, o gene que codifica a proteína envelope do vírus da Dengue e os genes HA e NA do vírus da Influenza A.

A construção do modelo de dados tem como base um conjunto de treinamento bruto composto por sequências do gene alvo em formato FASTA, preferencialmente já anotadas (genotipadas).

Esse dataset é submetido a uma análise de qualidade, considerando critérios como o comprimento das sequências, o percentual de sítios definidos univocamente com nucleotídeos A, G, C ou T, e a cobertura no alinhamento com a(s) sequência(s) de referência do(s) gene(s) alvo(s). Durante esse processo, as sequências de baixa qualidade são filtradas. Define-se como de baixa qualidade qualquer sequência com comprimento inferior ao da referência ou com um percentual elevado (acima de um limiar definido pelo usuário) de sítios contendo caracteres diferentes de A, G, C ou T.

As sequências alinhadas que passam pelo filtro compõem o conjunto básico de treinamento do modelo. Posteriormente, as sequências de nucleotídeos alinhadas e no quadro de leitura são convertidas em sequências de códons da forma descrita no apêndice C. As sequências de códons resultantes passam por uma filtragem, removendo sequências que contenham códons de parada (exceto no final).

Posteriormente, são identificados e listados os sítios polimórficos (onde aparece

mais de um códon na amostra), listando e contando os distintos códons em cada sítio polimórfico.

Padrão Identificador Único das Amostras (ID):

Para cada sequência da amostra filtrada se constrói um padrão formado por uma lista dos códons nos sítios polimórficos ordenados de menor a maior. Transformando a lista em string se gera um identificador (IDs) para cada sequência.

As sequências com IDs únicos são salvas no conjunto de padrões de treinamento. A transformação para string reduz consideravelmente o tempo de busca por um padrão específico no conjunto de treinamento. Cada padrão (ID) único é considerada uma classe primária do modelo de classificação.

Posteriormente, as IDs são convertidas em transações e agrupadas utilizando o algoritmo não supervisionado CLOPE, com parâmetros definidos pelo usuário. Esses parâmetros incluem um valor de repulsão, que determina a granularidade dos clusters, e o número de tentativas aleatórias para selecionar o melhor agrupamento. O modelo de classificação resultante contém uma lista de clusters, cada um com suas respectivas amostras (classes) primárias agrupadas.

Por fim, o código utiliza a biblioteca SciPy para realizar a ligação hierárquica com diferentes métodos de ligação, gerando dendrogramas que são plotados a partir dos resultados da análise. Ao final do processo, a biblioteca Dill é utilizada para serializar o objeto e salvar o modelo.

Ao fim do processo o método AGUA tem como suas saídas: (1) uma matriz de distâncias, que pode ser utilizada para construir dendrogramas, (2) o modelo de dados e (3) o modelo de classificação treinado.

A seguir, detalhamos as estruturas de dados para armazenar tanto o modelo de dados como o de classificação. O modelo desenvolvido para a análise de genomas virais foi projetado como parte fundamental de uma ferramenta para a investigação da evolução de espécies virais, especificamente utilizando sequências genômicas do SARS-COV-2 e Dengue. A seguir, detalhamos a estrutura e os componentes deste modelo: O arquivo `modelo.obj` contém uma instância de um modelo.

D.1 Estruturas de Dados do Modelo

- **ListOfVarSites**: Lista de posições dos códons variáveis nas sequências genéticas.
  - *Tipo:* Lista de inteiros
  - *Descrição:* Posições dos códons variáveis nas sequências genéticas.
- **NmbOfClasses**: Número de classes identificadas no modelo.

  - *Tipo:* Inteiro
  - *Descrição:* Número total de classes identificadas no modelo.

- **CodeOfClass**: Códons associados a cada classe, representando características genéticas específicas.
  - *Tipo:* Lista de listas de strings
  - *Descrição:* Códons de cada classe, representando características genéticas específicas.
  - *Subcampos:* Cada sublista contém uma sequência de códons para uma classe específica.

- **GroundTruth**: Anotações de cada classe, associando sequências a suas respectivas classes.
  - *Tipo:* Lista de strings ou números
  - *Descrição:* Anotações de cada classe, associando sequências a suas respectivas classes.

- **Accession**: Identificadores das sequências ou classes, utilizados para referência cruzada.
  - *Tipo:* Lista de strings ou números
  - *Descrição:* Identificadores únicos (IDs) das sequências ou classes, utilizados para referência cruzada.

- **NmbOfClusters**: Número total de clusters formados após o agrupamento das sequências.
  - *Tipo:* Inteiro
  - *Descrição:* Número total de clusters formados após o agrupamento das sequências.

- **TheSpeciesOfCluster**: Espécies associadas a cada cluster identificado.
  - *Tipo:* Lista de listas de strings
  - *Descrição:* Espécies associadas a cada cluster identificado.
  - *Subcampos:* Cada sublista contém os nomes das espécies presentes em um cluster específico.

- **DistributionMatrix**: Matriz de distribuição das sequências nos clusters, representando a variação e a densidade de cada cluster.

  - *Tipo:* Matriz (lista de listas ou array)
  - *Descrição:* Matriz de distribuição das sequências nos clusters, representando a variação e a densidade de cada cluster.
  - *Subcampos:* Cada elemento da matriz representa uma distribuição específica.

- **TheClusterOfClass**: Classe associada a cada cluster formado.
  - *Tipo:* Lista de inteiros ou strings
  - *Descrição:* Classe associada a cada cluster formado.

APÊNDICE E – TREINAMENTO COM *CLOPE*

Dado um conjunto T de transações e definido um valor de repulsão r, se realiza um número indeterminado de iterações, da forma descrita a seguir:

1. Inicialização: Começa com uma distribuição inicial aleatória das transações no conjunto T.

2. Alocação Inicial: As transações são lidas na ordem sendo atribuídas a algum dos clusters existentes ou a um cluster novo, dependendo de qual opção agrega mais lucro ao agrupamento.

3. Iterações: As transações são iterativamente movidas entre clusters para maximizar a função de lucro. Em cada iteração todas as transações, sorteadas de forma aleatória, são testadas. O teste consiste em que cada transação é avaliada sendo transferida para todos os outros clusters, assim como criando um cluster novo com ela. As transações são movimentadas para a alternativa que agrega mais valor, podendo ficar no mesmo cluster se for a melhor opção. A função de lucro é projetada para equilibrar a densidade dos clusters e a diversidade dentro de cada cluster. As iterações finalizam quando nenhuma transação é transferida de um cluster para outro.

Para poder executar os passos 2 e 3 do algoritmo acima, se utilizam três funções de avaliação de valor: (1) Função de Valor do Cluster, (2) Função de Mudança de Valor do Cluster com a Incorporação de uma Transação e (3) Função de Valor da Criação de um Cluster. Estas funções são descritas no apêndice F.

Como o *CLOPE* agrupa as sequências na ordem em que aparecem no arquivo de entrada, cada vez que o arquivo é reordenado aleatoriamente, obtém-se um agrupamento ligeiramente diferente para o mesmo valor de repulsão. Por isso, os autores sugerem repetir a clusterização um número predefinido de vezes, embaralhando a entrada para escolher o melhor agrupamento.

Este procedimento permite evitar selecionar um agrupamento pseudo-ótimo por ter sido rodado apenas uma vez, mas ao mesmo tempo requer definir quantas vezes deve ser repetido o agrupamento, para ter certeza de ter escolhido um agrupamento suficientemente perto do agrupamento ótimo.

Em outras palavras, o pesquisador precisa definir além do valor de repulsão r, o número de vezes que o agrupamento será realizado com ordenações aleatórias das sequências do conjunto de treinamento.

Neste trabalho, implementamos um algoritmo de otimização automática da repulsão r utilizando a anotação do conjunto de treinamento e introduzindo uma métrica de qualidade do agrupamento. Esta contribuição é descrita na seção 4.3.

APÊNDICE F – MODELO MATEMÁTICO DO MÉTODO CLOPE

Nesta seção descrevemos a função de valor de um cluster assim como as fórmulas que permitem calcular a variação de valor de um cluster devido à adição e remoção de uma transação (sequência). Também descrevemos os critérios de decisão de movimentação de transações entre clusters.

F.1 Função de Valor do Cluster

Dado um cluster C_k que contém N_k transações $[t_1, t_2, \ldots, t_{N_k}]$, a função de valor do mesmo é definida como o produto do gradiente $\nabla(C_k)$ pelo peso w_k do cluster:

$$V(C_k) = \text{grad}(C_k)\ w_k \tag{F.1}$$

onde

$$\text{grad}(C_k) = \frac{S_k}{W_k^r}$$

$$w_k = \frac{N_k}{N}$$

sendo:

- $N = \sum_{k=1}^{K} N_k$: o numero total de transações no grupamento com K clusters
- $S_k = \sum_{n=1}^{N_k} |t_n|$: a soma dos tamanhos das transações no cluster C_k. Como no nosso caso todas as transações tem o mesmo tamanho P (número de sítios polimórficos), então $S_k = PN_k$. Ou seja, S_k é uma medida do tamanho do cluster e como S_k está no numerador da equação do valor, significa que clusters maiores tem mais valor que clusters menores.
- W_k: o número de itens distintos no cluster. No nosso caso W_k é o número de pares $(p : Ap)$ distintos nas N_k transações alocadas no cluster C_k. Lembrando que um par $(p : A_p)$ representa uma posição polimórfica e um códon, W_k é uma medida da variabilidade genética intra-cluster. Como W_k está no denominador da equação do valor, significa que clusters com menor variabilidade tem maior valor que clusters com maior variabilidade genética.
- $r \geq 1$: o parâmetro de repulsão. É o parâmetro a ser otimizado utilizando as etiquetas para avaliar critérios quantitativos de qualidade do agrupamento. Ou seja,

as etiquetas não influenciam o agrupamento em si, por isso que nosso método é classificado como semi-supervisionado. As etiquetas são utilizadas para otimizar o parâmetro de repulsão.

Fazendo as substituições indicadas, e eliminando a constante P/N, a função de valor do cluster C_k vem dado de forma simplificada por:

$$V(C_k) = \frac{N_k^2}{W_k^r} \tag{F.2}$$

F.2 Mudança no valor do cluster com a adição de uma transação

Para adicionar uma transação t_i a um cluster existente C_k, a mudança no valor do cluster é calculada como:

$$\Delta V_{\text{add}}(t_i, C_k) = \frac{(N_k + 1)^2}{(W_k + \omega_{i,k})^r} - V(C_k) \tag{F.3}$$

onde $\omega_{i,k}$ é o número de novos itens que a transação t_i trás com ela para o cluster k e $V(C_k)$ é o valor atual do cluster C_k definida pela equação F.2.

F.3 Mudança no valor do cluster com a exclusão de uma transação

Ao excluir uma transação t_i de um cluster existente C_k, a mudança no valor do cluster é calculada como:

$$\Delta V_{\text{remove}}(t_i, C_k) = \frac{(N_k - 1)^2}{(W_k - \delta_{i,k})^r} - V(C_k) \tag{F.4}$$

onde $\delta_{i,k}$ é o número de itens únicos da transação t_i no cluster k e $V(C_k)$ é o valor atual do cluster C_k definida pela equação F.2.

F.4 Valor da criação de um novo cluster com uma transação nova

Durante a inicialização do agrupamento, as novas sequências são adicionadas a clusters existentes ou a um cluster novo. Neste contexto é necessário definir o valor da criação de um cluster novo, para comparar com o valor da adição da transação aos clusters existentes.

Ao criar um novo cluster C_{new} com uma nova transação t_i, estamos criando um cluster com uma única transação, pelo que $N_{new} = 1$ e qualquer transação no nosso caso tem P itens distintos, pelo que o valor da criação do novo cluster é:

$$\Delta V(C_{new}(t_i)) = \frac{N_{new}}{|t_i|^r} = \frac{1}{P^r} \tag{F.5}$$

F.5 Decidindo a movimentação de uma transação de um cluster para outro

Considere uma transação t_i num cluster C_{k1} que possa ser movimentada para outro cluster C_{k2}. Neste caso precisamos calcular a mudança total de valor derivada da remoção de um e da adição no outro, ou seja:

$$\Delta V_{\text{move}}(t_i, C_{k1}, C_{k2}) = \Delta V_{\text{remove}}(t_i, C_{k1}) + \Delta V_{\text{add}}(t_i, C_{k2}) \tag{F.6}$$

Fazendo as substituições indicadas obtemos:

$$\Delta V_{\text{move}}(t_i, C_{k1}, C_{k2}) = \frac{(N_{k1}-1)^2}{(W_{k1}-\delta_{i,k1})^r} - V(C_{k1}) + \frac{(N_{k2}+1)^2}{(W_{k2}+\omega_{i,k2})^r} - V(C_{k2}) \tag{F.7}$$

Como $\Delta V_{\text{remove}}(t_i, C_{k1})$ é definido para a sequência t_i a ser movimentada, a busca tem que ser feita calculando apenas o ganho $\Delta V_{\text{add}}(t_i, C_{k2})$ nos outros clusters, para achar o maior ganho possível.

Contudo a decisão requer que $\Delta V_{\text{move}}(t_i, C_{k1}, C_{k2})$ seja positiva, pelo que se

$$\Delta V_{\text{remove}}(t_i, C_{k1}) > 0$$

então a transferência é realizada, mas, se $\Delta V_{\text{remove}}(t_i, C_{k1}) < 0$ então somente se realiza a transferência caso o máximo ganho com a adição a outro cluster for superior à perda com a remoção do cluster atual. Em caso contrário a transação t_i permanece no cluster $k1$.

F.6 Decidindo a remoção de uma transação de um cluster para criar um novo

Considere uma transação t_i num cluster C_k que possa ser removida para criar um novo cluster C_{new}. Neste caso precisamos calcular o ganho total de forma

$$\Delta V_{\text{rem-new}}(t_i, C_k, C_{new}) = \Delta V_{\text{remove}}(t_i, C_k) + \Delta V(C_{new}(t_i)) \tag{F.8}$$

Fazendo as substituições indicadas obtemos:

$$\Delta V_{\text{rem-new}}(t_i, C_k, C_{new}) = \frac{(N_{k1}-1)^2}{(W_{k1}-\delta_{i,k1})^r} - V(C_{k1}) + \frac{1}{P^r} \tag{F.9}$$

Fazendo os cálculos se

$$\Delta V_{\text{rem-new}}(t_i, C_k, C_{new}) \leq 0$$

a transação t_i permanece no cluster C_k. Em caso contrário, t_i é removida do cluster C_k e um novo cluster é criado com ela.

APÊNDICE G – MODELO DE DADOS: ESTRUTURA, CRIAÇÃO E ATUALIZAÇÃO

G.1 Estrutura e Papéis dos Dados do Modelo de Referência

O Modelo de Referência/Treinamento é uma estrutura de dados heterogênea que contem:

1. A sequência de referência, $S_{n,ref}$, utilizado para alinhar novas sequências. Estas novas sequências podem ser candidatas a serem incorporadas no conjunto de treinamento ou enviadas para classificação.

2. O conjunto de sequências de treinamento em nucleotídeos, $\mathcal{S}_n$, mantido como backup para futuras análises se for preciso.

3. O conjunto de sequências de treinamento em códons, $\mathcal{S}_c$, utilizada para a busca de posições polimórficas toda vez que uma nova sequência é adicionada ao conjunto de treinamento.

4. O vetor de posições polimórficas, *pos*, utilizado para a extração de atributos das sequências enviadas para classificação.

5. A matriz de atributos, T, é um inventário dinâmico da diversidade da espécie viral correspondente formatada da forma que o método de clusterização CLOPE lê o conjunto de treinamento. Contem apenas informação das posições polimórficas. Devido a isto, toda vez que o vetor de posições polimórficas *pos* for atualizado, a matriz T precisa ser atualizada.

   Por se tratar de um inventário, ela por si só serve para pesquisar se uma sequência de entrada é nova ou é idêntica a alguma outra já incluída no modelo. Em outras palavras, T é projetada para ser uma base de conhecimento da bio-diversidade genética viral a nível de códons.

   Devido a sua importância, a seção G.2 é dedicada a explicar em detalhe o processo de atualização.

G.2 Atualização do Conjunto de Treinamento

A atualização do modelo com uma nova sequência de nucleotídeos, S_{new}, é feito em dois passos:

1. **Adição da nova sequência:**

a) Verificar se S_{new} não possui caráteres indeterminados e se tem comprimento compatível com o dataset. Em caso positivo, continue.

b) Alinhar S_{new} com $S_{n,ref}$ e validar o alinhamento (se foi completo, ou seja, se o início e o fim de $S_{new}^{aligned}$ coincidem com o de $S_{n,ref}$).

c) Verificar se $S_{new}^{aligned}$ é uma nova sequência, ou seja, se já não existe uma idêntica em $\mathcal{S}_n$. Se for redundante, finalizar o processo. Em caso contrário, adicionar $S_{new}^{aligned}$ ao dataset $\mathcal{S}_n$ e continuar o pipeline.

d) Traduzir $S_{new}^{aligned}$ para sequência de códons: $S_{c,new} = [c_1, c_2, \ldots, c_M]$ e adicioná-la ao dataset $\mathcal{S}_c$:

Listing G.1 – Adding new sequence to the dataset

```
Sc.append(Sc_new)
N = len(Sc)  # update dataset size
```

e) Construir o vetor de atributos extraindo os códons das posições polimórficas, da forma descrita acima.

Listing G.2 – Updating training set for CLOPE

```
t_new = [] # Init the attribute list of the new sequence
for p in pos: # loop over polymorphic positions
    t_new.append(str(p) + ":" + str(Sc[-1,p])) # Updating the
       attribute list
```

f) Adicionar o vetor de atributos à matriz de treinamento T:

Listing G.3 – Add new feature vector to the training set

```
T.append(t_new) # Updating the training set
```

2. **Atualização da Estrutura do Dataset:**

a) Verificar se a nova sequência criou novas posições polimórficas: Para isto usamos o algoritmo a seguir:

Listing G.4 – Finding new polymorphic positions

```
pos_new = [] # Initializing list of polymorphic positions
for j in range(M): # Loop through positions
    if len(set(Sc[:,j])) > 1: # if there is more than one codon
       type
        pos_new.append(j) # add position j to the list
# Finding positions in pos_new that are not in pos
new_positions = [p for p in pos_new if p not in pos]
```

Se a lista *new_positions* está vazia finalize o pipeline. Em caso contrário, continue

b) Atualização do vetor de posições polimórficas:

Listing G.5 – Updating list of polymorphic positions

```
# Merging the lists while maintaining order
pos = sorted(pos + new_positions)
```

c) Recriar o dataset de treinamento de CLOPE incluindo as novas posições polimórficas:

Listing G.6 – Updating training set for CLOPE

```
T = [] # Initializing the training dataset (list of lists of
    attributes)
for i in range(N): # Loop through sequences
    t = [] # Initializing the attribute list for sequence i
    for p in pos: # loop over polymorphic positions
        t.append(str(p) + ":" + str(Sc[i,p])) # Updating the
            attribute list of sequence i
    T.append(t) # Updating the training set
```

3. **Salvar o novo modelo**: Em caso de atualização do modelo, salvar e registrar a data da atualização.

APÊNDICE H – PIPELINE DE GENOTIPAGEM DE SEQUÊNCIAS

Nesta seção descrevemos brevemente o pipeline para classificar uma sequência de entrada usando o modelo CLOPE treinado.

Na seção G.2 foi descrito o processo para adicionar uma nova sequência ao conjunto de treinamento de CLOPE. O processamento inicial para classificar uma sequência de entrada S_{input} é muito similar. A seguir os passos do processamento:

1. Verificar se S_{input} não possui caracteres indeterminados e se tem comprimento compatível com o dataset. Em caso positivo, continue.

2. Alinhar S_{input} com $S_{n,ref}$ e validar o alinhamento (se foi completo, ou seja, se o início e o fim de $S_{input}^{aligned}$ coincidem com o de $S_{n,ref}$).

3. Classificação Direta: Verificar se no conjunto de treinamento $\mathcal{S}_n$ existe uma sequência idêntica a $S_{input}^{aligned}$. Se existir, retornar a distribuição de probabilidade genotípica do cluster CLOPE ao qual a sequência idêntica do dataset foi alocada. Em caso contrário, continue.

   A distribuição de probabilidade genotípica vem dada pelo percentual de cada genótipo no cluster identificado. Sendo k o cluster ao qual pertence a sequência do dataset que é idêntica à sequência de entrada, a probabilidade da sequência de entrada pertencer a qualquer genótipo $g = 1, 2, \ldots, G$ vem dada por $d_{k,g}/N_k$, onde $N_k = \sum_g d_{k,g}$.

4. Traduzir $S_{input}^{aligned}$ para sequência de códons: $S_{c,input} = [c_1, c_2, \ldots, c_M]$.

5. Construir o vetor de atributos extraindo os códons das posições polimórficas:

   Listing H.1 – Getting feature vector for CLOPE

```
t_input = [] # Init the attribute list of the new sequence
for p in pos: # loop over polymorphic positions
    t_input.append(str(p) + ":" + str(Sc_input[-1,p])) # Updating
        the attribute list
```

6. Buscar o cluster CLOPE mais adequado para a sequência de entrada:

   a) Inicialização do CLOPE: Distribuir as transações nos clusters segundo o melhor agrupamento do treinamento $\mathcal{G}$.

   b) Alocação: Atribuir a transação de entrada a algum dos clusters identificados no treinamento ou a um cluster novo, dependendo de qual opção agrega mais lucro ao agrupamento. Diferenciamos dois casos:

> *A sequência de entrada é atribuída a um cluster existente:* Retornar a distribuição de probabilidade genotípica do cluster CLOPE atribuído à sequência de entrada.

> *A sequência de entrada é atribuída a um novo cluster:* Neste caso se retorna que a sequência parece ser de um novo genótipo. Este caso, que evidencia a capacidade natural de descoberta de novas cepas virais do método descrito, precisa ser analisado por especialistas, para determinar se a sequência lida é confiável ou pode ser de baixa qualidade. Em caso de ser considerada de alta qualidade (confiável) pelos especialistas, o processo padrão recomendado, consiste em:

- Notificar os órgãos de vigilância sanitária sobre a descoberta
- Nomear o novo genótipo
- Adicionar a nova sequência ao conjunto de treinamento como descrito na seção G.2
- Atualizar a Matriz de Distribuição D e o vetor de agrupamento $\mathcal{G}$ com o novo cluster e novo genótipo.

APÊNDICE I – INSTALAÇÃO DA FERRAMENTA AGSSA

I.1 Descrição do Repositório

Repositório para o desenvolvimento do Trabalho de Conclusão de Curso do curso de Análise e Desenvolvimento de Sistemas da Universidade do Estado da Bahia.

I.2 Começando

Instruções para instalação e execução do projeto **AGSSA (Advanced Genotyping with Semi-Supervised Algorithm)**.

I.2.1 Pré-requisitos

O que você precisa para instalar o software e como instalá-lo:

- Python 3.8
- RabbitMQ (`sudo apt install rabbitmq-server`)
- Alinhamento das Sequências
- Golang (https://go.dev/doc/install)
- Minimap2 (https://github.com/lh3/minimap2?tab=readme-ov-file#installation)
- Gofasta (https://github.com/virus-evolution/gofasta?tab=readme-ov-file#installation)

I.2.2 Executando o Projeto

I.2.2.1 Criando o ambiente virtual:

```
python3 -m venv venv
```

I.2.2.2 Ativando o ambiente virtual:

```
source venv/bin/activate
```

I.2.2.3 Instalando as dependências:

```
pip install -r aplicacao/requirements.txt
```

I.2.2.4 Configuração:

Criar o arquivo `.env` com as informações disponíveis no arquivo `.env.example`.

I.2.2.5 Executando o Celery:

```
celery -A app.celery worker --loglevel=info
```

I.2.2.6 Executando a aplicação para desenvolvimento:

```
flask --app app run --debug
```

I.3 Executando o Sistema

Após a configuração, e inicialização do projeto, o sistema estará disponível para acesso no endereço: http://localhost:5000. Certifique-se de que o ambiente esteja corretamente configurado para garantir o funcionamento adequado.