

**UNIVERSIDADE FEDERAL DE MINAS GERAIS**  
**Instituto de Ciências Exatas**  
**Programa de Pós-graduação em Estatística**

Alberto Thalison Silveira

**APLICAÇÃO DE TÉCNICAS DE APRENDIZADO DE MÁQUINA PARA A  
PREDIÇÃO DE MUTAGENICIDADE *IN VITRO***

Belo Horizonte  
2025

Alberto Thalison Silveira

**APLICAÇÃO DE TÉCNICAS DE APRENDIZADO DE MÁQUINA PARA A PREDIÇÃO DE MUTAGENICIDADE *IN VITRO***

Monografia de especialização apresentada ao Departamento de Estatística – Instituto de Ciências Exatas da Universidade Federal de Minas Gerais, como requisito parcial à obtenção do título de Especialista em Estatística Computacional Aplicada.

Orientador: Professor Dr. Guilherme Lopes de Oliveira

Belo Horizonte

2025

Silveira, Alberto Thalison.

S587a      Aplicação de técnicas de aprendizado de máquina para a  
predição de mutagenicidade in vitro [recurso eletrônico] /  
Alberto Thalison Silveira. – 2025.

1 recurso online (122 f. il., color.) : pdf.

Orientador: Guilherme Lopes de Oliveira

Monografia (especialização) - Universidade Federal de Minas  
Gerais, Instituto de Ciências Exatas, Departamento de  
Estatística.

Referências: f. 73-76.

1. Estatística. 2. Análise por conglomerados. 3. Aprendizado  
do computador. 4. Indústria farmacêutica - Toxicidade.  
5. Toxicologia – Modelos computacionais. I. Oliveira, Guilherme  
Lopes de. II. Universidade Federal de Minas Gerais, Instituto de  
Ciências Exatas, Departamento de Estatística. III. Título.

CDU 519.2(043)



**Universidade Federal de Minas Gerais**  
**Instituto de Ciências Exatas**  
**Departamento de Estatística**  
**Programa de Pós-Graduação**

Caixa Postal 702

31270-901 Belo Horizonte- MG – Brasil

Telefone (31) 3409-5923

Fax (31) 3499-5924

E-mail: [pgest@ufmg.br](mailto:pgest@ufmg.br)

WEB: <http://www.est.ufmg.br/posgrad/>

## **ATA DO 345ª. TRABALHO DE FIM DE CURSO DE ESPECIALIZAÇÃO EM ESTATÍSTICA DE ALBERTO THALISON SILVEIRA.**

Aos dez dias do mês de abril de 2025, às 10:00 horas, com utilização de recursos de videoconferência a distância, reuniram-se os professores abaixo relacionados, formando a Comissão Examinadora homologada pela Comissão do Curso de Especialização em Estatística Computacional Aplicada, para julgar a apresentação do trabalho de fim de curso do aluno **Alberto Thalison Silveira**, intitulado: “*Aplicação de técnicas de aprendizado de máquina para a predição de mutagenicidade in vitro*”, como requisito para obtenção do Grau de Especialista em Estatística. Abrindo a sessão, o Presidente da Comissão, Professor Guilherme Lopes de Oliveira – Orientador, após dar conhecimento aos presentes do teor das normas regulamentares, passou a palavra ao candidato para apresentação de seu trabalho. Seguiu-se a arguição pelos examinadores com a respectiva defesa do candidato. Após a defesa, os membros da banca examinadora reuniram-se sem a presença do candidato e do público, para julgamento e expedição do resultado final. Foi atribuída a seguinte indicação: o candidato foi considerado Aprovado condicional às modificações sugeridas pela banca examinadora no prazo de 30 dias a partir da data de hoje por unanimidade. O resultado final foi comunicado publicamente à candidata pelo Presidente da Comissão. Nada mais havendo a tratar, o Presidente encerrou a reunião e lavrou a presente Ata, que será assinada por todos os membros participantes da banca examinadora. Belo Horizonte, 10 de abril de 2025.

Documento assinado digitalmente  
 **GUILHERME LOPES DE OLIVEIRA**  
Data: 10/04/2025 11:54:01-0300  
Verifique em <https://validar.iti.gov.br>

**Prof. Dr. Guilherme Lopes de Oliveira (orientador)**  
**DECOM/CEFET-MG**

Documento assinado digitalmente  
 **MARCOS OLIVEIRA PRATES**  
Data: 10/04/2025 15:27:44-0300  
Verifique em <https://validar.iti.gov.br>

**Prof. Dr. Marcos Oliveira Prates**  
**DEST/UFMG**

## RESUMO

No contexto da indústria farmacêutica, a toxicidade é uma das principais causas de fracasso de novos medicamentos, tornando essencial o desenvolvimento de métodos alternativos mais eficientes. A toxicologia computacional surge como uma solução promissora, utilizando ferramentas *in silico*, como as Relações Quantitativas entre Estrutura e Atividade (QSAR) e as Relações Estrutura-Atividade (SAR), para prever propriedades toxicológicas com base na estrutura química dos compostos. O presente estudo tem como objetivo desenvolver um modelo preditivo de mutagenicidade *in vitro* utilizando aprendizado de máquina, explorando descritores moleculares e métricas de desempenho para aumentar a confiabilidade das previsões. Foram avaliados os modelos Random Forest (RF), XGBoost e CART em três abordagens distintas para a definição do espaço químico. Na primeira abordagem, que considerou todos os dados do banco (6.467 moléculas), o XGBoost apresentou desempenho equilibrado (ROC de 0,8518, sensibilidade de 0,7765 e F1-score de 0,7624), enquanto o CART teve o menor desempenho. Na segunda abordagem, baseada na técnica de agrupamento e definição do espaço químico como o *cluster* mais próximo, o RF obteve melhora na acurácia, mas com redução na sensibilidade. Já a terceira abordagem, que utilizou distância euclidiana entre moléculas, o RF atingiu a maior sensibilidade (0,8889) e ROC (0,8649), tornando-se a melhor opção para evitar falsos negativos. Os resultados indicam que os modelos RF da terceira abordagem e XGBoost da primeira abordagem são os mais adequados para a predição de mutagenicidade, reduzindo significativamente a ocorrência de falsos negativos e custo computacional. Além disso, durante aplicação a moléculas externas à modelagem, o modelo RF obteve 100% de acerto, enquanto o XGBoost, com apenas um erro, destaca-se como alternativa mais rápida. Como perspectiva futura, propõe-se uma combinação dos modelos XGBoost e RF para equilibrar tempo e desempenho, além da incorporação de um terceiro método, como o CatBoost, para fortalecer a base de evidências.

Palavras-chave: mutagenicidade; toxicologia computacional, análise de agrupamentos, análise de componentes principais; aprendizado de máquina.

## ABSTRACT

In the pharmaceutical industry, toxicity is one of the main causes of failure for new drugs, making the development of more efficient alternative methods essential. Computational toxicology emerges as a promising solution, utilizing *in silico* tools such as Quantitative Structure-Activity Relationships (QSAR) and Structure-Activity Relationships (SAR) to predict toxicological properties based on the chemical structure of compounds. This study aims to develop an *in vitro* mutagenicity predictive model using machine learning, exploring molecular descriptors and performance metrics to enhance the reliability of predictions. The models Random Forest (RF), XGBoost, and CART were evaluated in three distinct approaches for defining the chemical space. In the first approach, which considered all data from the database (6,467 molecules), XGBoost showed balanced performance (ROC of 0.8518, sensitivity of 0.7765, and F1-score of 0.7624), while CART had the lowest performance. In the second approach, based on clustering techniques and chemical space as the nearest cluster, RF improved accuracy but with a reduction in sensitivity. In the third approach, which used the Euclidean distance between molecules, RF achieved the highest sensitivity (0.8889) and ROC (0.8649), making it the best option for avoiding false negatives. The results indicate that the RF model from the third approach and the XGBoost model from the first approach are the most suitable for mutagenicity prediction, significantly reducing the occurrence of false negatives and low computational demand. Additionally, when applied to external molecules, the RF model achieved 100% accuracy, while XGBoost, with only one error, stands out as a faster alternative. As a future perspective, a combination of XGBoost and RF models is proposed to balance computational time and performance, along with the incorporation of a third method, as the CatBoost algorithm, to strengthen the weight of evidence.

**Keywords:** mutagenicity; computational toxicology; cluster analysis; principal component analysis; machine learning.

## SUMÁRIO

|       |                                                                          |     |
|-------|--------------------------------------------------------------------------|-----|
| 1     | <b>INTRODUÇÃO</b>                                                        | 8   |
| 2     | <b>REVISÃO DE LITERATURA</b>                                             | 10  |
| 2.1   | TOXICOLOGIA COMPUTACIONAL E AVALIAÇÃO TOXICOLÓGICA                       | 10  |
| 2.2   | APRENDIZADO DE MÁQUINA                                                   | 14  |
| 3     | <b>OBJETIVOS</b>                                                         | 19  |
| 3.1   | OBJETIVO GERAL                                                           | 19  |
| 3.2   | OBJETIVOS ESPECÍFICOS                                                    | 19  |
| 4     | <b>MATERIAIS E MÉTODOS</b>                                               | 20  |
| 5     | <b>RESULTADOS E DISCUSSÃO</b>                                            | 33  |
| 5.1   | ANÁLISE DESCRITIVA DOS DADOS                                             | 33  |
| 5.2   | MODELOS DE PREDIÇÃO DE MUTAGENICIDADE <i>IN VITRO</i>                    | 41  |
| 5.2.1 | Primeira abordagem: ajuste sem a definição de espaço químico             | 42  |
| 5.2.2 | Segunda abordagem: espaço químico otimizado por técnicas de agrupamento  | 48  |
| 5.2.3 | Terceira abordagem: espaço químico definido pelas distâncias euclidianas | 60  |
| 5.3   | APLICAÇÃO DOS MODELOS DE PREDIÇÃO <i>IN SILICO</i>                       | 66  |
| 6     | <b>CONSIDERAÇÕES FINAIS</b>                                              | 71  |
|       | <b>REFERÊNCIAS</b>                                                       | 73  |
|       | <b>APÊNDICE A</b>                                                        | 77  |
|       | <b>APÊNDICE B</b>                                                        | 79  |
|       | <b>APÊNDICE C</b>                                                        | 81  |
|       | <b>APÊNDICE D</b>                                                        | 86  |
|       | <b>APÊNDICE E</b>                                                        | 92  |
|       | <b>APÊNDICE F</b>                                                        | 100 |

|                         |            |
|-------------------------|------------|
| <b>APÊNDICE G .....</b> | <b>109</b> |
| <b>APÊNDICE H .....</b> | <b>115</b> |
| <b>APÊNDICE I .....</b> | <b>122</b> |

## 1 INTRODUÇÃO

Atualmente, mais de 80.000 substâncias químicas são amplamente utilizadas no mundo e centenas de novos compostos e misturas são introduzidos no mercado anualmente. No entanto, devido ao alto custo (material e ético) e à demora dos testes de toxicidade em animais, apenas uma pequena parte dessas substâncias foi devidamente analisada quanto aos seus riscos potenciais (WALKER, 2012).

A avaliação toxicológica é essencial para fármacos, pois a toxicidade é uma das principais causas de fracasso de candidatos a novos medicamentos em estágios avançados de desenvolvimento. A indústria farmacêutica enfrenta desafios como a grande quantidade de compostos que necessitam de estudos de avaliação da toxicidade *in vivo*, a escassez de ensaios *in vitro* eficientes e a dificuldade dos modelos *in vitro* e animais em prever com precisão certos desfechos toxicológicos em humanos (WALKER, 2012).

Para poluentes ambientais e fármacos, abordagens alternativas são necessárias para prever riscos e otimizar a priorização dos testes diante de recursos limitados. Uma dessas abordagens é a toxicologia computacional, que utiliza ferramentas computacionais para modelagem preditiva, análise de conjuntos de dados complexos e integração de evidências, oferecendo alternativas aos testes em animais (WALKER, 2012).

As relações quantitativas entre estrutura e atividade (QSAR) estabelecem uma correlação entre a atividade biológica de ligantes e suas estruturas químicas. Para isso, é necessário medir a atividade biológica, codificar a estrutura química por meio de descritores moleculares e derivar equações matemáticas que correlacionem esses descritores com a atividade biológica (TODESCHINI; CONSONNI, 2009).

O sucesso de abordagens *in silico*, como *Quantitative Structure-Activity Relationship* (QSAR) e *Structure-Activity Relationship* (SAR), depende da qualidade dos dados e da escolha de descritores que capturem as características moleculares responsáveis pelos efeitos tóxicos. O QSAR tornou-se essencial na pesquisa farmacêutica para a otimização de compostos, sendo utilizado na previsão de propriedades farmacocinéticas, como absorção, distribuição, metabolismo e eliminação (ADME),

além da toxicidade de candidatos a fármacos (KLEINSTREUER; TETKO; TONG, 2021).

O uso de sistemas de predição *in silico* tem se tornado comum na avaliação do potencial genotóxico de substâncias em submissões regulatórias. Como exemplo, temos as diretrizes do *International Council for Harmonisation of Technical Requirements for Pharmaceuticals for Human Use* (ICH), através do *Guideline on assessment and control of DNA reactive (mutagenic) impurities in pharmaceuticals to limit potential carcinogenic risk* (ICH-M7), que recomendam o uso de dois sistemas complementares de QSAR para avaliar o potencial mutagênico de impurezas farmacêuticas que reagem com o DNA (ICH, 2023b). O aumento da dependência dessas ferramentas intensificou o foco na avaliação de suas métricas de desempenho preditivo e precisão.

Diante da relevância da mutagenicidade na avaliação de risco toxicológico e da necessidade de abordagens cada vez mais robustas e confiáveis, este trabalho tem como objetivo desenvolver um modelo para prever a mutagenicidade a partir de dados de literatura e ferramentas estatísticas de aprendizado de máquina, explorando descritores moleculares e métricas de desempenho para aumentar a confiabilidade das previsões. Com isso, espera-se fornecer uma ferramenta eficiente que auxilie na identificação de potenciais riscos genotóxicos, em consonância com as diretrizes regulatórias e fortalecendo o uso de métodos alternativos ao uso de animais na avaliação da segurança química.

## 2 REVISÃO DE LITERATURA

### 2.1 TOXICOLOGIA COMPUTACIONAL E AVALIAÇÃO TOXICOLÓGICA

A avaliação de risco toxicológico consiste em processos sistemáticos e científicos utilizados para identificar, analisar e quantificar os riscos potenciais à saúde pública e ao meio ambiente, causados por agentes químicos e físicos. Essas práticas são essenciais para garantir a proteção da população e do ambiente, permitindo que medidas preventivas sejam adotadas com base em evidências científicas. Nesse contexto, as agências regulatórias têm a incumbência de revisar, quantificar e regulamentar os fatores de risco, com o objetivo de reduzir potenciais danos à saúde pública (ICH, 2023a).

Em 1983, o *National Research Council* (NRC) introduziu um modelo pioneiro de avaliação de risco composto por quatro etapas, o qual permanece como um referencial metodológico até os dias atuais. Este modelo envolve uma abordagem rigorosa e sistemática, englobando a identificação de perigos, a avaliação de dose-resposta, a avaliação de exposição e a caracterização de risco, integrando diferentes áreas da toxicologia para proporcionar uma análise abrangente dos riscos (NRC, 1983). Cada uma dessas etapas desempenha uma função específica:

- a) Identificação de perigos: busca identificar os efeitos adversos possíveis;
- b) Avaliação dose-resposta: estabelece a relação entre a intensidade da exposição e a gravidade dos efeitos;
- c) Avaliação de exposição: mede o nível e a duração da exposição;
- d) Caracterização de risco: combina todas essas informações para estimar a magnitude do risco à saúde.

Para as indústrias farmacêutica, cosmética e química, bem como para as agências reguladoras, a avaliação de risco toxicológico é crucial e inevitável. No entanto, os objetivos dessas avaliações são bastante diversos. Na indústria farmacêutica, a

avaliação de risco é realizada durante todo o processo de descoberta e desenvolvimento de medicamentos. Começando com a primeira avaliação de um composto, ainda antes da síntese, ela continua até os ensaios clínicos, em que a avaliação de risco é crucial para a saúde humana. Em decorrência da rigorosa avaliação de segurança de novos compostos, a taxa de rejeição durante o processo de desenvolvimento ainda é alta, muitas vezes devido a preocupações com a segurança. Dessa maneira, iniciativas público-privada tem levado à criação de projetos colaborativos para prever a toxicidade por meio de modelos *in silico* (HEMMERICH; ECKER, 2020).

Na indústria química, a avaliação de segurança também é realizada, mas muitas vezes com o objetivo principal de garantir a segurança dos trabalhadores e a avaliação de riscos ambientais. Por outro lado, as avaliações de risco na indústria cosmética concentram-se principalmente na segurança do consumidor. Os testes de segurança atuais em todas as indústrias seguem como tendência à implementação dos princípios dos 3Rs, que propõem a redução (*reduction*), refinamento (*refinement*) e substituição (*replacement*) dos estudos de segurança realizados com animais. Esses princípios são fundamentais para garantir uma abordagem ética e responsável no uso de animais em pesquisas científicas. Em suma, a *redução* busca minimizar o número de animais utilizados em experimentos por meio de métodos mais eficientes e técnicas que permitam obter dados válidos com menos amostras; o *refinamento* foca em melhorar o bem-estar dos animais durante os experimentos, utilizando práticas que reduzam o sofrimento e o estresse, como melhores condições de habitat e técnicas menos invasivas; e a *substituição* busca substituir os animais por métodos alternativos sempre que possível, como modelos *in vitro* ou simulações computacionais, evitando o uso de animais quando há outras opções viáveis (HEMMERICH; ECKER, 2020; MACARTHUR CLARK, 2018).

Historicamente, a toxicologia tem sido dependente de experimentos com animais para avaliar os efeitos adversos de medicamentos candidatos. No entanto, nos últimos anos, têm sido feitos esforços significativos para reduzir os testes em animais por Novas Metodologias de Abordagem (NAMs). Apesar desses avanços, as complexidades inerentes aos fenômenos em estudo ainda persistem tanto nos métodos experimentais quanto nos computacionais (*in silico*) dentro do âmbito das NAMs, tornando a avaliação da toxicidade dos produtos químicos em análise, um desafio. Nesse contexto, definindo complexidade como a característica de algo que envolve múltiplas

partes interconectadas, tornando difícil a compreensão total ou a busca por uma solução clara para um problema, o desafio é capturar a interação desses componentes nas avaliações de toxicidade, já que os fenômenos que emergem dessa interação (resultados finais) podem ser imprevisíveis caso o nível de representação não seja detalhado o suficiente (RODRÍGUEZ-BELENGUER *et al.*, 2023; SEWELL *et al.*, 2024).

Dentre os desfechos mais estudados na toxicologia, temos a mutagenicidade. Tal efeito deletério se refere à capacidade de uma substância causar danos genéticos transmissíveis, como mutações gênicas, aberrações cromossômicas e alterações no número de cromossomos. Esses danos podem resultar da interação com o DNA ou outros alvos celulares. A pesquisa sobre mutagenicidade e carcinogenicidade química está interligada, com a teoria de que substâncias eletrofílicas podem ser tanto mutagênicas quanto carcinogênicas, sendo a mutagenicidade um possível passo no processo carcinogênico. Inicialmente, a ideia de que mutágenos poderiam ser carcinogênicos era teórica, mas foi confirmada com estudos de Malling e Ames, os quais demonstraram que certos carcinógenos podiam se tornar mutagênicos. A genotoxicidade, que inclui a mutagenicidade, refere-se a danos ao material genético que podem resultar em mutações, mas nem sempre são transmitidos para as células filhas (BENIGNI; BOSSA, 2011).

A toxicologia computacional (*in silico*) visa prever riscos, como mutagenicidade e toxicidade em órgãos, por meio de modelos computacionais. Esses modelos podem ser baseados em regras desenvolvidas por especialistas ou utilizar abordagens estatísticas, como o aprendizado de máquina. Sua aplicabilidade é ampla, desde as fases iniciais do desenvolvimento de medicamentos, até avaliações de risco em situações urgentes onde testes tradicionais não são viáveis. Ademais, os métodos *in silico* ajudam a direcionar o desenvolvimento de medicamentos, selecionando compostos com melhores perfis de segurança, antes de serem submetidos a testes pré-clínicos. A toxicologia computacional também oferece suporte para identificar riscos raros e eventos idiossincráticos que não podem ser observados em testes convencionais, além de ser uma ferramenta valiosa para prever riscos tanto para usuários de produtos químicos quanto para o meio ambiente. Dessa forma, ela se torna crucial no aprimoramento da segurança e da eficácia de novos compostos (HEMMERICH; ECKER, 2020).

Outro conceito importante para as análises *in silico* remonta sobre a Relação Estrutura–Atividade Quantitativa (QSAR), a qual modela como as mudanças na estrutura química afetam as atividades biológicas ou as propriedades físico-químicas de uma molécula. Os tipos de atividades e propriedades que esses estudos tentam trabalhar são diversos, abrangendo desde atividades farmacológicas até efeitos toxicológicos e propriedades bioquímicas e físico-químicas. O objetivo dos métodos QSAR é criar um modelo generalista aplicável a um número variável de moléculas, que pode ser de poucas unidades ou até milhares, e utilizar esse modelo para situações específicas, como a identificação de riscos na avaliação de segurança química (CRONIN *et al.*, 2025).

As SARs (Relações Estrutura-Atividade) têm se mostrado amplamente aplicáveis na toxicologia, avaliação de risco e outras áreas regulatórias, com um aumento no interesse devido à necessidade de avaliar a segurança química sem recorrer ao uso de animais. O fator crucial para o uso de SAR como ferramentas *in silico* nessas aplicações é o desenvolvimento de regras estruturais codificáveis computacionalmente, permitindo sua aplicação na identificação de potenciais riscos em novas moléculas (CRONIN *et al.*, 2022).

Os estudos QSAR geralmente utilizam descritores moleculares para a geração dos modelos. Estes, por sua vez, são representações numéricas de estruturas químicas que transformam informações codificadas em números úteis para que os computadores possam processar as moléculas. Eles podem ser classificados conforme sua dimensionalidade em unidimensionais (1D), baseados em propriedades físico-químicas; bidimensionais (2D), que descrevem propriedades de uma representação 2D; e tridimensionais (3D), que dependem da conformação molecular. Além disso, existem descritores 4D, 5D e 6D, que envolvem simulações dinâmicas e múltiplos modelos de solvatação. A classificação também pode ser feita quanto à natureza, incluindo descritores constitucionais, topológicos, geométricos, eletrostáticos e mecânico-quânticos. Vários programas, como DRAGON®, CDK®, CODESSA® e RDKit®, estão disponíveis para calcular esses descritores (ALVES *et al.*, 2018).

## 2.2 APRENDIZADO DE MÁQUINA

Métodos de aprendizado de máquina nos conceitos da área de inteligência artificial permitem a sistemas computacionais identificar padrões e tomar decisões com base em grandes volumes de dados, sem a necessidade de programação explícita. Isso é possível, por exemplo, através do uso de técnicas estatísticas e algoritmos que ajustam modelos preditivos à medida que mais dados são coletados, permitindo que o sistema aprenda de forma autônoma. O aprendizado de máquina tem se mostrado extremamente valioso em diversas áreas, especialmente na saúde e pesquisa, em que a análise de grandes quantidades de dados pode levar a avanços significativos (SARKER, 2021).

Os algoritmos de aprendizado de máquina são principalmente divididos em quatro categorias (SARKER, 2021):

- a) **Aprendizado supervisionado:** Neste tipo de aprendizado, o objetivo é aprender uma função que mapeia entradas (*input*) para saídas (*output*), utilizando dados rotulados. Entre as tarefas mais comuns estão a classificação, que separa os dados em categorias, e a regressão, que ajusta os dados a uma função contínua.
- b) **Aprendizado não-supervisionado:** Ao contrário do aprendizado supervisionado, o aprendizado não-supervisionado analisa conjuntos de dados não rotulados, sem a necessidade de intervenção humana. Ele é utilizado para extrair características gerativas (criação de dados a partir de modelos treinados), identificar tendências e estruturas significativas, agrupar resultados e realizar análises exploratórias. As tarefas mais comuns incluem agrupamento, redução de dimensionalidade, detecção de anomalias e aprendizado de características.
- c) **Aprendizado semi-supervisionado:** Esta é uma abordagem híbrida que combina características do aprendizado supervisionado e não supervisionado, operando tanto com dados rotulados quanto não rotulados. É útil em cenários em que dados rotulados são escassos, mas há abundância de dados não rotulados. O objetivo é melhorar o desempenho da previsão, superando

os resultados obtidos apenas com os dados rotulados. Áreas como tradução automática, detecção de fraudes e classificação de textos fazem uso dessa técnica.

- d) **Aprendizado por reforço:** No aprendizado por reforço, o agente aprende a tomar decisões em um ambiente com base em recompensas ou punições recebidas pelas suas ações. O objetivo é maximizar a recompensa ao longo do tempo, o que torna esse tipo de aprendizado ideal para otimizar a eficiência de sistemas complexos, como robótica, direção autônoma e logística. No entanto, não é indicado para resolver problemas mais simples ou diretos.

A análise por classificação é um método de aprendizado supervisionado em aprendizado de máquina, sendo um problema de modelagem preditiva, em que um rótulo de classe é previsto para um exemplo fornecido. Matematicamente, ela mapeia uma função ( $f$ ) das variáveis de entrada ( $x$ ) para as variáveis de saída ( $y$ ), que representam os rótulos, alvos ou categorias. A classificação pode ser aplicada a dados estruturados ou não estruturados, bem como pode ser dividida em classificação binária, que envolve tarefas com duas categorias, como “verdadeiro” e “falso” ou “sim” e “não”. Por exemplo, em testes médicos, “câncer não detectado” seria o estado normal e “câncer detectado” o estado anômalo, caracterizando uma classificação binária (SARKER, 2021).

Diversos algoritmos de classificação foram propostos na literatura de aprendizado de máquina e ciência de dados. Dentre os métodos mais comuns e amplamente utilizados em várias áreas de aplicação, temos a árvore de decisão, que é um método de aprendizado supervisionado não paramétrico bem conhecido. Os métodos de aprendizado com árvore de decisão são utilizados tanto para tarefas de classificação quanto de regressão (SARKER, 2021).

O CART (*Classification and Regression Trees*) é um algoritmo popular de aprendizado supervisionado baseado em árvores de decisão que cria uma árvore binária, ou seja, cada nó da árvore tem exatamente dois ramos, e as divisões de dados são feitas de forma a otimizar a pureza dos nós (ou, no caso de regressão, a homogeneidade dos valores) (JAMES *et al.*, 2023; MARTIN KRZYWINKSI, 2017).

O critério de decisão adotado em métodos como o CART, é usado para dividir os dados de forma a otimizar a homogeneidade dos nós. Para classificação, o critério é o índice de Gini, dado por (JAMES *et al.*, 2023; MARTIN KRZYWINKSI, 2017):

$$Gini(D) = 1 - \sum_{i=1}^k p_i^2$$

onde  $p_i$  é a proporção de exemplos de cada classe. O algoritmo busca minimizar esse índice, criando nós mais puros. Para regressão, a divisão é baseada na redução da variância, calculada por (JAMES *et al.*, 2023; MARTIN KRZYWINKSI, 2017):

$$Variância(D) = \frac{1}{|D|} \sum_{i=1}^{|D|} (y_i - \bar{y})^2$$

onde  $y_i$  são os valores das amostras e  $\bar{y}$ , a média. O objetivo é reduzir a variância, formando subgrupos mais homogêneos.

O Random Forest, ou florestas aleatórias (RF) é um algoritmo de aprendizado de máquina baseado em *bagging* (*Bootstrap Aggregating*), que cria um conjunto de árvores de decisão independentes. Cada árvore é treinada com uma amostra aleatória de dados, extraída com reposição e, para dividir cada nó, um subconjunto aleatório de características é considerado. O resultado final do modelo é determinado pela média (no caso de regressão) ou pela votação (no caso de classificação) das previsões das árvores individuais. O modelo é robusto ao *overfitting* e é eficiente para lidar com dados de alta dimensionalidade e dados faltantes (BREIMAN, 2001; SARKER, 2021).

No Random Forest, cada árvore de decisão é construída de forma independente, e o índice de Gini é usado para determinar as divisões em cada árvore. Como o Random Forest é uma coleção de árvores treinadas de maneira independente em amostras diferentes dos dados e com variáveis aleatórias em cada nó, ele combina o poder do índice de Gini em muitas árvores para criar um modelo robusto e mais preciso (BREIMAN, 2001; JAMES *et al.*, 2023; SARKER, 2021).

Por sua vez, o *eXtreme Gradient Boosting* (XGBoost) é um algoritmo de aprendizado baseado em árvores de decisão que utiliza a técnica de *Gradient Boosting* para otimizar a função de perda em problemas de regressão, classificação e *ranking*. Em

vez de simplesmente somar as previsões das árvores, como no *bagging*, o XGBoost ajusta cada árvore de forma sequencial, corrigindo os erros das anteriores ao minimizar o erro residual. A matemática do XGBoost envolve a otimização de uma função objetiva, que mede a discrepância entre as previsões do modelo e os valores reais. Essa função é combinada com um termo de regularização, que controla a complexidade do modelo e previne o *overfitting*. O processo iterativo de adição de novas árvores ajusta os erros residuais do modelo, e a regularização, por meio de parâmetros L1 e L2, ajuda a equilibrar o ajuste aos dados de treinamento e a capacidade de generalização. A otimização é realizada utilizando o gradiente descendente, permitindo que o algoritmo melhore progressivamente as previsões (CHEN; GUESTRIN, 2016).

A análise de agrupamentos (*clustering*), que também pode ser aplicada ao aprendizado de máquina, é uma técnica que visa agrupar pontos de dados semelhantes em conjuntos, chamados *clusters*, sem depender de rótulos ou resultados pré-determinados. O objetivo é organizar os dados de forma que os pontos dentro de um *cluster* sejam mais semelhantes entre si do que com os pontos de outros *clusters* (JORDAN; KLEINBERG; SCHÖLKOPF, 2006; MINGOTI, 2005).

O método mais comum para realizar *clustering* é o K-means, principalmente por ser simples, rápido e eficiente. O algoritmo K-means busca alocar pontos de dados a *clusters* de forma que a distância entre os pontos e os seus centroides (vetor de média amostral) seja minimizada. Ele trabalha identificando um número predefinido de centroides ( $k$ ) e atribuindo cada ponto de dados ao cluster mais próximo, com base na distância euclidiana, por exemplo. A fórmula da distância euclidiana entre dois pontos  $x = (x_1, x_2, \dots, x_d)$  e  $y = (y_1, y_2, \dots, y_d)$  em um espaço  $n$ -dimensional é dada por (JORDAN; KLEINBERG; SCHÖLKOPF, 2006; MINGOTI, 2005):

$$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

O balanceamento de dados destaca-se também como uma etapa crucial em tarefas de classificação com classes desbalanceadas, por exemplo, uma vez que modelos treinados sob distribuições assimétricas tendem a favorecer a classe majoritária, reduzindo a sensibilidade na identificação da classe minoritária. Tradicionalmente,

esse problema é abordado por técnicas de *oversampling*, como o *Random Oversampling* ou o *SMOTE*, que aumentam a representação da classe minoritária por meio da replicação ou geração de exemplos sintéticos. Uma das estratégias mais consagradas para essa finalidade é o Índice de Youden (YAUDEN, 1950), que propõe a escolha do ponto de corte que maximiza a soma entre a sensibilidade e a especificidade do modelo, expressa pela fórmula:

$$J = \text{Sensibilidade} + \text{Especificidade} - 1$$

O limiar ótimo é, portanto, aquele que maximiza  $J$ , sendo amplamente utilizado em contextos diagnósticos e biomédicos por permitir o equilíbrio entre verdadeiros positivos e verdadeiros negativos, mesmo quando os custos dos erros são simétricos. No entanto, uma alternativa recente a esse método foi proposta por Assunção, Izbicki e Prates em 2024, que, partindo de fundamentos teóricos sobre a distribuição de probabilidades condicionais, demonstram que o classificador ótimo sob a perda 0–1 pode ser obtido sem qualquer tipo de *oversampling*, desde que se utilize como limiar a própria prevalência da classe positiva:

$$g^*(x) = \{ 1, \text{se } P(Y = 1 | x) \geq P(Y = 1); 0, \text{caso contrário} \}$$

Essa abordagem elimina a necessidade de gerar amostras sintéticas ou realizar balanceamento explícito do conjunto de dados, sendo especialmente vantajosa do ponto de vista computacional. Enquanto o índice de Youden depende da avaliação empírica da curva ROC em um conjunto de validação e é sensível à variabilidade amostral, o método proposto por Prates e colaboradores fornece uma solução analítica direta baseada na própria distribuição dos dados. Ambos os métodos visam otimizar o desempenho do classificador, sendo suficiente para atingir resultados comparáveis ou superiores àqueles obtidos por técnicas tradicionais de balanceamento (ASSUNÇÃO; IZBICKI; PRATES, 2024; YAUDEN, 1950).

### 3 OBJETIVOS

A seguir, são apresentados o objetivo geral e os objetivos específicos da presente monografia.

#### 3.1 OBJETIVO GERAL

Desenvolver uma aplicação capaz de realizar predições acerca do potencial mutagênico de um dado composto utilizando, para tal finalidade, ferramentas estatísticas clássicas e modelos avançados de aprendizado de máquina.

#### 3.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos estão descritos a seguir:

- a) Criação de um banco de dados de trabalho composto pela variável resposta, representativa da atividade mutagênica, e variáveis preditoras, calculadas por meio de bibliotecas de quimioinformática;
- b) Definição do domínio de aplicabilidade dos modelos gerados;
- c) Avaliação descritiva dos dados;
- d) Treinamento de modelos de aprendizado de máquina utilizando diferentes abordagens de amostragem (espaço químico);
- e) Avaliação das métricas de desempenho dos modelos de predição desenvolvidos;
- f) Aplicação em compostos químicos externos ao banco de dados utilizado para treinamento, testando a generalização dos modelos.

## 4 MATERIAIS E MÉTODOS

Para a composição do banco de dados empregado no presente trabalho, foram utilizados resultados de mutagenicidade obtidos pelo teste de Ames de 6.512 compostos químicos conforme compilado por Hansen et al., (2009). Tal banco de dados fora reunido para o desenvolvimento de métodos de predição *in silico*, sendo que 3.503 compostos são positivos para mutagenicidade e 3.009 negativos para o mesmo desfecho. (HANSEN *et al.*, 2009).

Fazendo-se uso da aplicação de acesso livre RDKit, que consiste em uma coleção de *softwares* para quimioinformática e aprendizado de máquina escrito em C++ e *Python*, foram calculadas as variáveis que, somadas à variável resposta ACTIVITY (mutagenicidade), são representadas no Quadro 1, a seguir:

Quadro 1 – Descrição das variáveis utilizadas no estudo.

| Variável | Natureza            | Tipo                             | Descrição                                                                                                                                                                                                                                                                                                                                 |
|----------|---------------------|----------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ACTIVITY | Resposta            | Qualitativa dicotômica (binária) | Resultados de mutagenicidade (Teste de Ames), sendo "0" negativo e "1" positivo.                                                                                                                                                                                                                                                          |
| SPS      | Score               | Quantitativa numérica contínua   | O <i>Spacial Score</i> (SPS) é um sistema empírico que quantifica a complexidade espacial de um composto com base na fração de carbonos $sp^3$ e estereogênicos, permitindo a comparação entre moléculas. Sua versão normalizada, nSPS, ajusta a pontuação pelo tamanho da molécula, dividindo o SPS pelo número total de átomos pesados. |
| MolWt    | Estrutura molecular | Quantitativa numérica contínua   | Massa molecular.                                                                                                                                                                                                                                                                                                                          |

|                     |                     |                                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|---------------------|---------------------|--------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| NumValenceElectrons | Estrutura molecular | Quantitativa numérica contínua | Número de elétrons de valência na molécula. Ajuda a prever a reatividade e as propriedades de ligação.                                                                                                                                                                                                                                                                                                                                                                                                          |
| FpDensityMorgan1    | <i>Fingerprint</i>  | Quantitativa numérica contínua | Os descritores Morgan <i>fingerprints</i> , calculados em diferentes raios (1, 2, 3), capturam a complexidade estrutural em níveis variados de vizinhança atômica, sendo úteis para similaridade molecular e aprendizado de máquina.                                                                                                                                                                                                                                                                            |
| FpDensityMorgan2    | <i>Fingerprint</i>  | Quantitativa numérica contínua | Morgan <i>fingerprint</i> , radius 2.                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| FpDensityMorgan3    | <i>Fingerprint</i>  | Quantitativa numérica contínua | Morgan <i>fingerprint</i> , radius 3.                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| BCUT2D_MWHI         | Descritor molecular | Quantitativa numérica contínua | Os descritores BCUT2D representam a estrutura bidimensional e as características topológicas de uma molécula, sendo calculados a partir da distribuição de átomos e ligações para gerar um conjunto de 32 valores numéricos.<br>BCUT2D_MWHI e BCUT2D_MWLOW são descritores que refletem a variação do peso molecular em uma molécula, representando, respectivamente, o maior e o menor valor de corte baseado no peso atômico de átomos conectados, sendo úteis para caracterizar sua distribuição estrutural. |

|                |                     |                                |                                                                                                                                                                                                                 |
|----------------|---------------------|--------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| BCUT2D_MWLOW   | Descritor molecular | Quantitativa numérica contínua | <i>Idem</i> BCUT2D_MWHI.                                                                                                                                                                                        |
| BCUT2D_CHGHI   | Descritor molecular | Quantitativa numérica contínua | BCUT2D_CHGHI e BCUT2D_CHGLO representam, respectivamente, os maiores e menores valores de corte baseados na distribuição de cargas atômicas em uma molécula, refletindo a variação das cargas em sua estrutura. |
| BCUT2D_CHGLO   | Descritor molecular | Quantitativa numérica contínua | <i>Idem</i> BCUT2D_CHGHI.                                                                                                                                                                                       |
| BCUT2D_LOGPHI  | Descritor molecular | Quantitativa numérica contínua | BCUT2D_LOGPHI e BCUT2D_LOGPLOW representam, respectivamente, os maiores e menores valores de corte do logP, indicando as regiões mais hidrofóbicas e hidrofílicas da molécula.                                  |
| BCUT2D_LOGPLOW | Descritor molecular | Quantitativa numérica contínua | <i>Idem</i> BCUT2D_LOGPHI.                                                                                                                                                                                      |
| BCUT2D_MRHI    | Descritor molecular | Quantitativa numérica contínua | BCUT2D_MRHI e BCUT2D_MRLOW representam, respectivamente, as regiões da molécula com maior e menor polarizabilidade, baseadas nos valores mais altos e baixos de refração molar.                                 |
| BCUT2D_MRLOW   | Descritor molecular | Quantitativa numérica contínua | <i>Idem</i> BCUT2D_MRHI.                                                                                                                                                                                        |

|               |                     |                                |                                                                                                                                                                                                                                                                                                                              |
|---------------|---------------------|--------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| AvgIpc        | Descritor molecular | Quantitativa numérica contínua | Retorna o conteúdo médio de informação dos coeficientes do polinômio característico da matriz de adjacência de um grafo com hidrogênio suprimido de uma molécula.                                                                                                                                                            |
| BalabanJ      | Descritor molecular | Quantitativa numérica contínua | O índice BalabanJ é um descritor topológico que quantifica a conectividade molecular, útil para discriminar isômeros e correlacionar com propriedades físico-químicas.                                                                                                                                                       |
| BertzCT       | Descritor molecular | Quantitativa numérica contínua | Um índice topológico destinado a quantificar a "complexidade" das moléculas.                                                                                                                                                                                                                                                 |
| Chi0          | Descritor molecular | Quantitativa numérica contínua | Os descritores Chi são índices topológicos propostos por Randić para quantificar a complexidade estrutural das moléculas com base na conectividade e distribuição de átomos. Os descritores Chi0 derivam da matriz de adjacência de um grafo molecular, onde os vértices representam átomos e as arestas, ligações químicas. |
| HallKierAlpha | Descritor molecular | Quantitativa numérica contínua | Este descritor é uma medida topológica. O valor Hall-Kier alpha para uma molécula.                                                                                                                                                                                                                                           |
| Kappa2        | Descritor molecular | Quantitativa numérica contínua | Hall-Kier Kappa2 value: relacionam-se à conectividade e à complexidade da molécula em relação a um grafo ideal (ligação linear). Úteis para comparar a forma de moléculas e a flexibilidade conformacional.                                                                                                                  |

|            |                     |                                |                                                                                                                                                                                                                                                                                                                                              |
|------------|---------------------|--------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Kappa3     | Descritor molecular | Quantitativa numérica contínua | Hall-Kier Kappa3 value: <i>idem</i> Kappa2.                                                                                                                                                                                                                                                                                                  |
| LabuteASA  | Descritor molecular | Quantitativa numérica contínua | Este descritor calcula a área de superfície acessível ao solvente (ASA) usando um algoritmo de Labute. Ele é útil para prever interações molécula-solvente e a solubilidade de moléculas.                                                                                                                                                    |
| PEOE_VSA7  | Descritor molecular | Quantitativa numérica contínua | O descritor MOE Charge VSA combina cargas parciais e contribuições da área de superfície de Van der Waals (VSA) para caracterizar propriedades moleculares. No intervalo $(-0,05 \leq x < 0,00)$ , ele reflete regiões de menor eletronegatividade orbital parcial, sendo útil na modelagem química e na predição de interações moleculares. |
| SMR_VSA10  | Descritor molecular | Quantitativa numérica contínua | Os descritores MOE MR VSA combinam a área de superfície de Van der Waals (VSA) com a refração molar (SMR), refletindo a polarizabilidade molecular. No intervalo $(4,00 \geq x < \infty)$ , eles ajudam a caracterizar regiões da molécula com diferentes contribuições de polarizabilidade e superfície.                                    |
| SlogP_VSA2 | Descritor molecular | Quantitativa numérica contínua | Os descritores MOE logP VSA combinam a área de superfície de Van der Waals (VSA) com o logP, refletindo a lipofilicidade da molécula. No intervalo $(-0,40 \leq x < -0,20)$ , eles caracterizam regiões com menor afinidade por ambientes lipofílicos.                                                                                       |

|                   |                              |                                  |                                                                                                                                                                                                                                                                                                                                      |
|-------------------|------------------------------|----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TPSA              | Propriedades físico-químicas | Quantitativa numérica contínua   | A <i>Topological Polar Surface Area</i> (TPSA) representa a soma das superfícies polares da molécula, sendo útil para prever permeabilidade celular, absorção e biodisponibilidade. Moléculas com TPSA acima de 140 Å podem ter permeação limitada, enquanto valores entre 60 e 140 Å são considerados ideais para absorção celular. |
| Fraction-CSP3     | Estrutura molecular          | Quantitativa discreta (contagem) | Fração de átomos de carbono que estão hibridizados em $sp^3$ .                                                                                                                                                                                                                                                                       |
| NumAliphaticRings | Estrutura molecular          | Quantitativa discreta (contagem) | Número de anéis alifáticos, que são anéis não aromáticos, geralmente compostos por ligações simples e podem ser saturados ou insaturados.                                                                                                                                                                                            |
| NumAromaticRings  | Estrutura molecular          | Quantitativa discreta (contagem) | Número de anéis aromáticos presentes na molécula, onde os elétrons são deslocalizados sobre todos os átomos do anel.                                                                                                                                                                                                                 |
| NumHAcceptors     | Estrutura molecular          | Quantitativa discreta (contagem) | Esse descritor quantifica os aceptores de hidrogênio na molécula, geralmente oxigênio, nitrogênio ou flúor, influenciando a solubilidade em água e interações intermoleculares.                                                                                                                                                      |
| NumHDonors        | Estrutura molecular          | Quantitativa discreta (contagem) | Esse descritor quantifica os doadores de hidrogênio na molécula, geralmente aminas ou hidroxilas, que participam de ligações de hidrogênio ao doar prótons para átomos eletronegativos como oxigênio, nitrogênio ou flúor.                                                                                                           |

|                   |                              |                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-------------------|------------------------------|----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| NumRotatableBonds | Estrutura molecular          | Quantitativa discreta (contagem) | Esse descritor quantifica as ligações simples não cíclicas que podem girar livremente, influenciando a flexibilidade conformacional da molécula.                                                                                                                                                                                                                                                                                                                 |
| NumSaturatedRings | Estrutura molecular          | Quantitativa discreta (contagem) | Refere-se ao número de anéis que contêm apenas ligações simples (não aromáticos). Esses anéis podem influenciar a rigidez da molécula e suas propriedades conformacionais.                                                                                                                                                                                                                                                                                       |
| RingCount         | Estrutura molecular          | Quantitativa discreta (contagem) | Número de anéis presentes na estrutura da molécula, sejam eles aromáticos ou não.                                                                                                                                                                                                                                                                                                                                                                                |
| MolLogP           | Propriedades físico-químicas | Quantitativa numérica contínua   | O <i>Wildman-Crippen LogP value</i> (Wang <i>octanol-water partition coefficient</i> ) mede a partição de uma substância entre duas fases: água e octanol. O MolLogP indica a lipofilicidade de uma molécula, refletindo sua tendência de se acumular em ambientes gordurosos em relação aos aquosos. Valores altos podem influenciar na absorção, distribuição, metabolismo e excreção (ADME) e, em QSAR, predizem permeabilidade celular e biodisponibilidade. |
| MolMR             | Propriedades físico-químicas | Quantitativa numérica contínua   | O AMR (Ghose-Crippen molar refractivity) mede a dispersão da luz por uma molécula, relacionada à polaridade e estrutura eletrônica. Calculado a partir de volumes atômicos e polarizabilidades, ele auxilia na previsão de interações intermoleculares, permeabilidade celular e solubilidade, sendo valioso em QSAR para                                                                                                                                        |

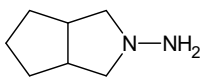
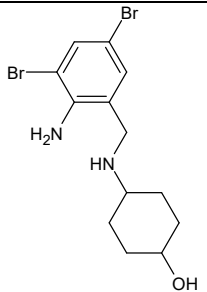
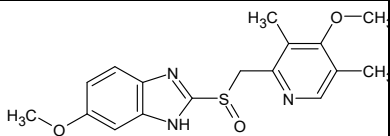
|  |  |  |                                                          |
|--|--|--|----------------------------------------------------------|
|  |  |  | avaliar a relação entre estrutura e atividade molecular. |
|--|--|--|----------------------------------------------------------|

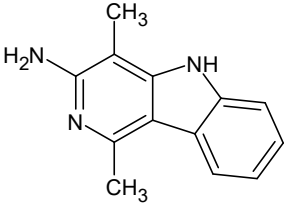
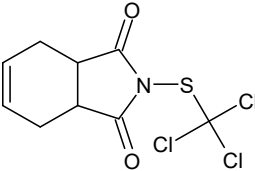
Fonte: RDKit (2025).

A determinação dos resultados para as 6.512 moléculas, considerando as variáveis mencionadas, foi realizada por meio de um programa escrito em *Python* (APÊNDICE A). Esse programa utiliza a biblioteca RDKit para processar as moléculas com base em suas assinaturas químicas, representadas pelo SMILES (*Simplified Molecular-Input Line-Entry System*), um formato que descreve estruturas químicas por meio de caracteres do padrão de codificação *American Standard Code for Information Interchange* (ASCII) (RDKit, 2025).

Através do mesmo programa exibido no APÊNDICE A, os valores correspondentes das variáveis consideradas para este estudo foram obtidos para cinco moléculas com potencial mutagênico reportado na literatura, porém, não presentes no banco de dados utilizado para o treinamento dos modelos de predição (QUADRO 2). Os referidos compostos químicos (moléculas-alvo) foram utilizados para a aplicação dos modelos, enquanto o omeprazol foi empregado no desenvolvimento das propostas.

Quadro 2 – Informações químicas das moléculas-alvo.

| Nome                              | SMILES                                                        | Mutagenicidade* | Estrutura molecular                                                                   |
|-----------------------------------|---------------------------------------------------------------|-----------------|---------------------------------------------------------------------------------------|
| 3-Amino-3-azabicyclo[3.3.0]octane | <chem>C1CC2CN(CC2C1)N</chem>                                  | Positivo        |  |
| Ambroxol                          | <chem>C1CC(CCC1NCC2=C(C=CC(=C2)Br)Br)N)O</chem>               | Negativo        |  |
| Omeprazol                         | <chem>CC1=CN=C(C(=C1OC)C)CS(=O)C2=NC3=C(N2)C=C(C=C3)OC</chem> | Negativo        |  |

|         |                                                 |          |                                                                                     |
|---------|-------------------------------------------------|----------|-------------------------------------------------------------------------------------|
| TRP-P-1 | <chem>CC1=C2C(=C(N=C1N)C)C3=CC=CC=C3N2</chem>   | Positivo |  |
| Captana | <chem>C1C=CCC2C1C(=O)N(C2=O)SC(Cl)(Cl)Cl</chem> | Positivo |  |

Fonte: EPA, 2025; “Mucoangin”, 2010; SNYDER, 2009.

Nota: \*Determinado experimentalmente, portanto, não se trata de uma predição.

Ressalta-se que, a fim de evitar vieses nas análises, influenciados por moléculas atípicas do banco de dados de treinamento, foram desconsideradas moléculas com massa molecular acima de 1.000 Daltons, coeficiente de partição óleo/água (LogP) acima de 10 e com valores faltantes (N.A.), restando 6.467 compostos (99,31% dos dados iniciais). Portanto, através dessas restrições, determinou-se o domínio de aplicação dos modelos gerados, sendo ideal que somente moléculas-alvo dentro dessas margens sejam desafiadas.

Além da observação sobre o domínio de aplicação, as moléculas-alvo devem ser apresentadas em sua forma neutra (base livre), ou seja, sem contraíons ácidos ou básicos característicos de formas salinas. Esta convenção visa seguir a maneira como o banco de dados de treinamento foi criado e está relacionado com os preceitos teóricos da solubilidade química de um composto e mecanismo de ação no órgão-alvo quando biodisponível: mesmo que uma substância esteja em sua forma salina, em solução aquosa ela se dissocia de seu par iônico, ficando disponível na forma neutra para acessar o local de ação tóxica ou terapêutica.

Para a condução das análises estatísticas e manipulação de dados, diversos pacotes do *software* R foram empregados (APÊNDICE I), como se segue (R CORE TEAM, 2025):

- Openxlsx*: Leitura e escrita de arquivos Excel;
- Car*: Diagnóstico de regressão e testes estatísticos;
- ggplot2*: Visualização de dados com gráficos estatísticos personalizáveis;
- corrplot*: Análise gráfica de correlações;

- e) *GGally*: Expansão das funcionalidades do *ggplot2* para visualizações multivariadas;
- f) *readxl*: Importação de planilhas Excel para manipulação de dados;
- g) *dplyr*: Manipulação eficiente de dados, incluindo filtragens, agregações e transformações;
- h) *randomForest*: Implementação de modelos de florestas aleatórias para classificação e regressão;
- i) *xgboost*: Algoritmos baseados em *gradiente boosting* para melhor desempenho preditivo;
- j) *pROC*: Análises de curvas ROC para avaliação de classificadores;
- k) *caret*: Modelagem preditiva e validação cruzada;
- l) *rpart* e *rpart.plot*: Construção e visualização de modelos baseados em árvores de decisão;
- m) *party*: Modelos avançados baseados em árvores de decisão;
- n) *FactoMineR* e *factoextra*: Análise de Componentes Principais (PCA) e análise de agrupamento;
- o) *cluster* e *ggdendro*: Métodos hierárquicos de agrupamento e visualização de dendrogramas;
- p) *prcomp*: Cálculos de decomposição de valores singulares na análise de componentes principais.

Foram utilizados três algoritmos de classificação para a predição de mutagenicidade das moléculas: árvores aleatórias (do inglês, *Random Forest*), árvores de classificação e regressão (do inglês, *Classification and regression trees*, CART) e XGBoost (*eXtreme Gradient Boosting*). Cada modelo foi implementado utilizando a função *train* do pacote *caret*, com o objetivo de comparar o desempenho entre métodos de aprendizado supervisionado baseados em árvores e *boosting*.

A implementação do *Random Forest* foi realizada com o parâmetro *method = "rf"*, utilizando o critério de "ROC" (área sob a curva ROC) para a otimização do desempenho do modelo. O parâmetro *importance = TRUE* foi ativado, permitindo a avaliação da relevância de cada variável para a predição. No treinamento, utilizou-se o

valor padrão de 500 árvores (*ntree* = 500), e o número de variáveis a serem selecionadas em cada divisão foi determinado automaticamente como a raiz quadrada do número total de variáveis preditoras, com  $mtry = \sqrt{p}$ .

O modelo CART foi implementado para a predição de mutagenicidade das moléculas, utilizando a função *rpart* do pacote *caret*. A árvore de decisão foi construída com a métrica ROC, visando otimizar a capacidade discriminativa do modelo entre as classes "Negative" e "Positive". O parâmetro *cp* (*Complexity Parameter*) foi ajustado para controlar a poda da árvore (*pruning*). Valores maiores de *cp* resultam em uma poda mais agressiva, reduzindo a complexidade do modelo e, consequentemente, o risco de *overfitting*, embora um valor muito alto possa levar ao subajuste (*underfitting*).

O XGBoost foi ajustado utilizando uma grade de parâmetros específicos (*tune-Grid*), visando otimizar o desempenho do modelo. Os parâmetros escolhidos foram: *nrounds* = 100 (número de iterações/árvores), *max\_depth* = 3 (profundidade máxima das árvores), *eta* = 0.3 (taxa de aprendizado) e *gamma* = 0 (complexidade mínima para divisão). O critério de otimização também foi a métrica ROC, assegurando a comparação direta com os outros modelos.

Para o treinamento dos modelos, os dados foram divididos em dois conjuntos: um para treinamento e outro para teste. A divisão foi realizada de forma estratificada, garantindo que as proporções de cada classe (positivo e negativo para mutagenicidade) fossem mantidas em ambos os conjuntos. A separação foi feita utilizando a função *createDataPartition* do pacote *caret*, com uma proporção de 80% dos dados alocados para o treinamento e 20% para o teste.

A validação cruzada foi empregada para avaliar a generalização dos modelos de classificação, utilizando a técnica de *k-fold cross-validation* com 10 *folds*, conforme especificado pelo argumento *trainControl(method = "cv", number = 10)* da função *train*. Este processo envolve a divisão do conjunto de dados de treinamento em 10 subconjuntos estratificados, sendo 9 utilizados para treinamento e 1 para validação. Esse ciclo é repetido até que cada subconjunto tenha sido utilizado uma vez como conjunto de validação. Adicionalmente, foi ativada a opção *classProbs = TRUE*, permitindo a obtenção das probabilidades das classes preditas. A função *twoClassSummary* foi utilizada para calcular as métricas de interesse, como sensibilidade, especificidade e AUC, durante o processo de validação.

Utilizando-se das técnicas de classificação mencionadas, foram desenvolvidos modelos de predição (i) com todos os dados do banco ( $n = 6.467$ ); (ii) através da delimitação de um espaço químico (grupo de moléculas com características físico-químicas próximas) definido por técnica de agrupamento; e (iii) utilizando o espaço químico formado por moléculas próximas ( $n = 500$ ) selecionadas através das distâncias euclidianas.

O desempenho dos modelos QSAR propostos foi acessado com o uso das métricas apresentadas no Quadro 3. Foram testados como limiar de corte para a classificação das predições o valor padrão de 0,5, assim como o valor que maximiza a sensibilidade e especificidade, conforme o método de Youden (YODEN, 1950).

Quadro 3 – Métricas para avaliação do desempenho dos modelos de predição.

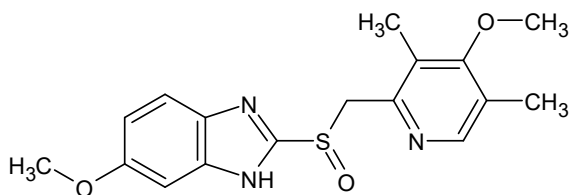
| Métrica                                                   | Cálculo/Descrição                                                                                                                           |
|-----------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| Curva ROC<br>( <i>Receiver Operating Characteristic</i> ) | Fornece uma representação gráfica do intervalo de possíveis pontos de corte com sua sensibilidade associada vs. Especificidade.             |
| Acurácia                                                  | $(VP + VN) \div (VP + VN + FP + FN)$<br>Avalia o desempenho geral da predição demonstrando a fração de compostos corretamente classificada. |
| Sensibilidade                                             | $VP \div (VP + FN)$<br>Mede a habilidade de classificar corretamente os compostos mutagênicos (positivo).                                   |
| Especificidade                                            | $VN \div (FP + VN)$<br>Mede a habilidade de classificar corretamente os compostos não mutagênicos (negativo).                               |
| F1-score                                                  | $2 \times (Sensibilidade \times Precisão) \div (Sensibilidade + Precisão)$<br>Indica a média harmônica da sensibilidade e precisão*.        |

Fonte: (BURGER, 2018; FURUHAMA *et al.*, 2023; HANLEY; MCNEIL, 1982).

Nota: VP – verdadeiro positivo; VN – verdadeiro negativo; FP – falso positivo; FN – falso negativo. \*A precisão mede a proporção de verdadeiros positivos entre todas as previsões positivas feitas por um modelo, calculado pela fórmula  $VP \div (VP + FP)$ .

Por fim, o fármaco omeprazol (FIGURA 1) foi utilizado no desenvolvimento dos modelos propostos como molécula-alvo, ou seja, como referência para a avaliação da capacidade preditiva dos modelos ajustados. E, do mesmo modo que as moléculas presentes no banco de dados de treinamento, os seus descritores moleculares foram calculados com a biblioteca RDKit (APÊNDICE A).

Figura 1 – Estrutura molecular do omeprazol.



Fonte: do autor.

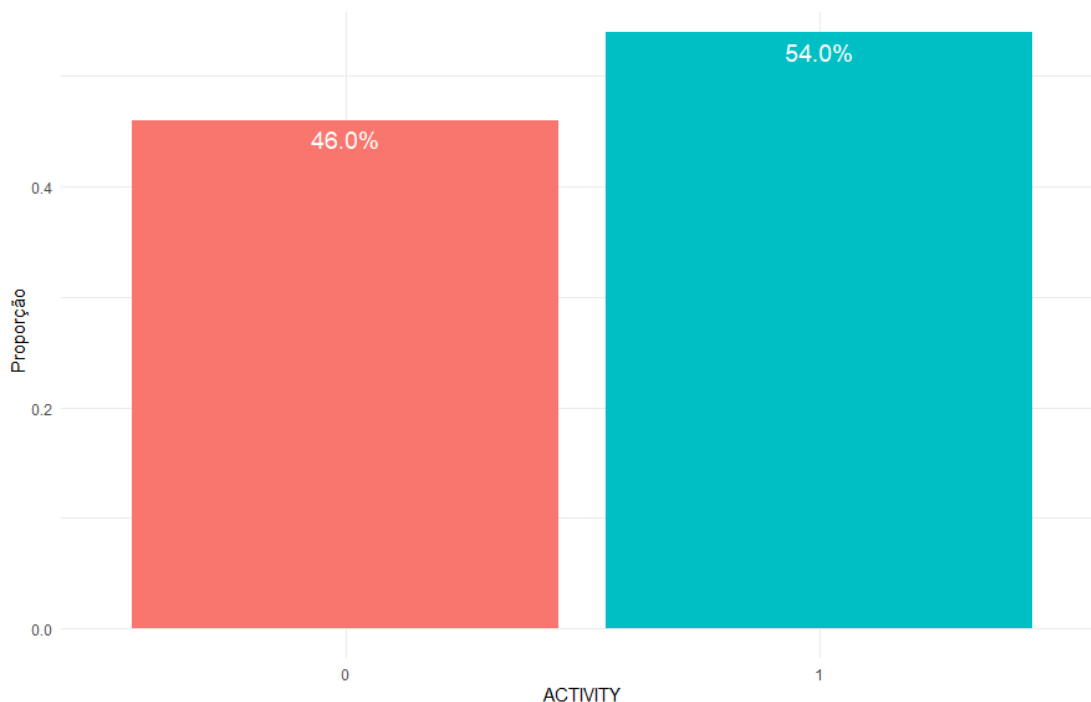
## 5 RESULTADOS E DISCUSSÃO

### 5.1 ANÁLISE DESCRITIVA DOS DADOS

O banco de dados original foi composto por 6.512 moléculas com resultados de mutagenicidade, positivos e negativos. Após a determinação do domínio de aplicação e remoção de moléculas com valores ausentes, restou-se 6.467 moléculas.

A variável *ACTIVITY*, a qual exibe os resultados de mutagenicidade, apresenta 2.976 (46%) moléculas não mutagênicas (0, Negativo) e 3.491 (54%) moléculas mutagênicas (1, Positivo), conforme representação gráfica a seguir (Figura 2):

Figura 2 – Proporção dos resultados de mutagenicidade.



Fonte: Do autor.

Com a adição de variáveis determinadas pela biblioteca RDKit, através da identidade química de cada molécula (SMILES), o banco de dados passou a ter 37 variáveis, sendo uma variável dicotômica dependente (resposta) e 36 variáveis independentes (preditoras) distribuídas entre variáveis quantitativas contínuas e discretas.

Conforme apresentado na Tabela 1, algumas variáveis exibem grande magnitude entre resultados mínimos e máximos, como MolWt, NumValenceElectrons, BertzCT, Kappa3 e MolMR; ou certa discrepância entre os valores de mediana e média, a exemplo das variáveis BCUT2D\_MWHI, BertzCT e SlogP\_VSA2. Esses resultados demonstram a variabilidade na complexidade das moléculas do banco de dados, indicando a necessidade da utilização de técnicas que permitam captar características locais e, conseqüentemente, mais homogêneas, no momento da geração dos modelos de predição.

Tabela 1 – Resumo da análise descritiva dos dados:

| <b>Variáveis</b>    | <b>Mín.</b> | <b>1º Q</b> | <b>Mediana</b> | <b>Média</b> | <b>3º Q</b> | <b>Máx.</b> |
|---------------------|-------------|-------------|----------------|--------------|-------------|-------------|
| SPS                 | 1,000       | 10,105      | 11,375         | 14,857       | 14,917      | 84,429      |
| MolWt               | 26,018      | 164,781     | 229,104        | 243,247      | 293,392     | 985,551     |
| NumValenceElectrons | 10,000      | 60,000      | 82,000         | 88,049       | 106,000     | 384,000     |
| FpDensityMorgan1    | 0,190       | 0,846       | 1,125          | 1,123        | 1,375       | 2,000       |
| FpDensityMorgan2    | 0,286       | 1,417       | 1,778          | 1,723        | 2,000       | 2,667       |
| FpDensityMorgan3    | 0,381       | 1,952       | 2,333          | 2,239        | 2,600       | 3,545       |
| BCUT2D_MWHI         | 12,718      | 16,270      | 16,597         | 23,587       | 32,137      | 126,932     |
| BCUT2D_MWLOW        | 9,328       | 9,984       | 10,130         | 10,165       | 10,268      | 34,453      |
| BCUT2D_CHGHI        | 0,077       | 2,023       | 2,131          | 2,143        | 2,252       | 3,319       |
| BCUT2D_CHGLO        | -2,812      | -2,179      | -2,070         | -2,080       | -1,984      | -0,813      |
| BCUT2D_LOGPHI       | 0,469       | 2,121       | 2,253          | 2,227        | 2,370       | 3,126       |
| BCUT2D_LOGPLOW      | -3,033      | -2,252      | -2,061         | -2,082       | -1,908      | -0,311      |
| BCUT2D_MRHI         | 2,002       | 5,716       | 5,999          | 6,234        | 6,334       | 14,282      |
| BCUT2D_MRLOW        | -1,999      | -0,145      | 0,070          | 0,221        | 0,468       | 4,853       |
| Avglpc              | 0,722       | 2,040       | 2,384          | 2,354        | 2,667       | 3,942       |
| BalabanJ            | 0,852       | 2,111       | 2,457          | 2,558        | 2,939       | 6,948       |
| BertzCT             | 0,000       | 225,545     | 461,891        | 527,338      | 757,144     | 3545,358    |
| Chi0                | 2,000       | 8,272       | 11,502         | 12,184       | 14,795      | 50,567      |
| HallKierAlpha       | -7,900      | -2,360      | -1,640         | -1,615       | -0,840      | 3,480       |
| Kappa2              | 0,206       | 3,064       | 4,141          | 4,886        | 5,781       | 35,282      |
| Kappa3              | -27,040     | 1,425       | 2,102          | 9,026        | 3,547       | 9507,960    |
| LabuteASA           | 12,901      | 67,971      | 95,525         | 100,573      | 123,573     | 391,529     |

|                   |        |        |        |        |        |         |
|-------------------|--------|--------|--------|--------|--------|---------|
| PEOE_VSA7         | 0,000  | 11,984 | 19,914 | 24,399 | 35,392 | 162,832 |
| SMR_VSA10         | 0,000  | 6,076  | 17,062 | 19,837 | 28,383 | 151,068 |
| SlogP_VSA2        | -0,062 | 6,286  | 14,891 | 19,632 | 25,152 | 232,681 |
| TPSA              | 0,000  | 26,020 | 46,530 | 55,620 | 74,600 | 391,200 |
| FractionCSP3      | 0,000  | 0,056  | 0,222  | 0,326  | 0,500  | 1,000   |
| NumAliphaticRings | 0,000  | 0,000  | 0,000  | 0,527  | 1,000  | 9,000   |
| NumAromaticRings  | 0,000  | 0,000  | 1,000  | 1,562  | 2,000  | 11,000  |
| NumHAcceptors     | 0,000  | 2,000  | 3,000  | 3,207  | 4,000  | 24,000  |
| NumHDonors        | 0,000  | 0,000  | 1,000  | 1,050  | 2,000  | 14,000  |
| NumRotatableBonds | 0,000  | 1,000  | 2,000  | 2,737  | 4,000  | 39,000  |
| NumSaturatedRings | 0,000  | 0,000  | 0,000  | 0,263  | 0,000  | 8,000   |
| RingCount         | 0,000  | 1,000  | 2,000  | 2,089  | 3,000  | 11,000  |
| MolLogP           | -8,161 | 1,277  | 2,436  | 2,538  | 3,754  | 9,868   |
| MolMR             | 0,000  | 43,137 | 62,797 | 65,582 | 83,182 | 236,991 |

Fonte: Do autor.

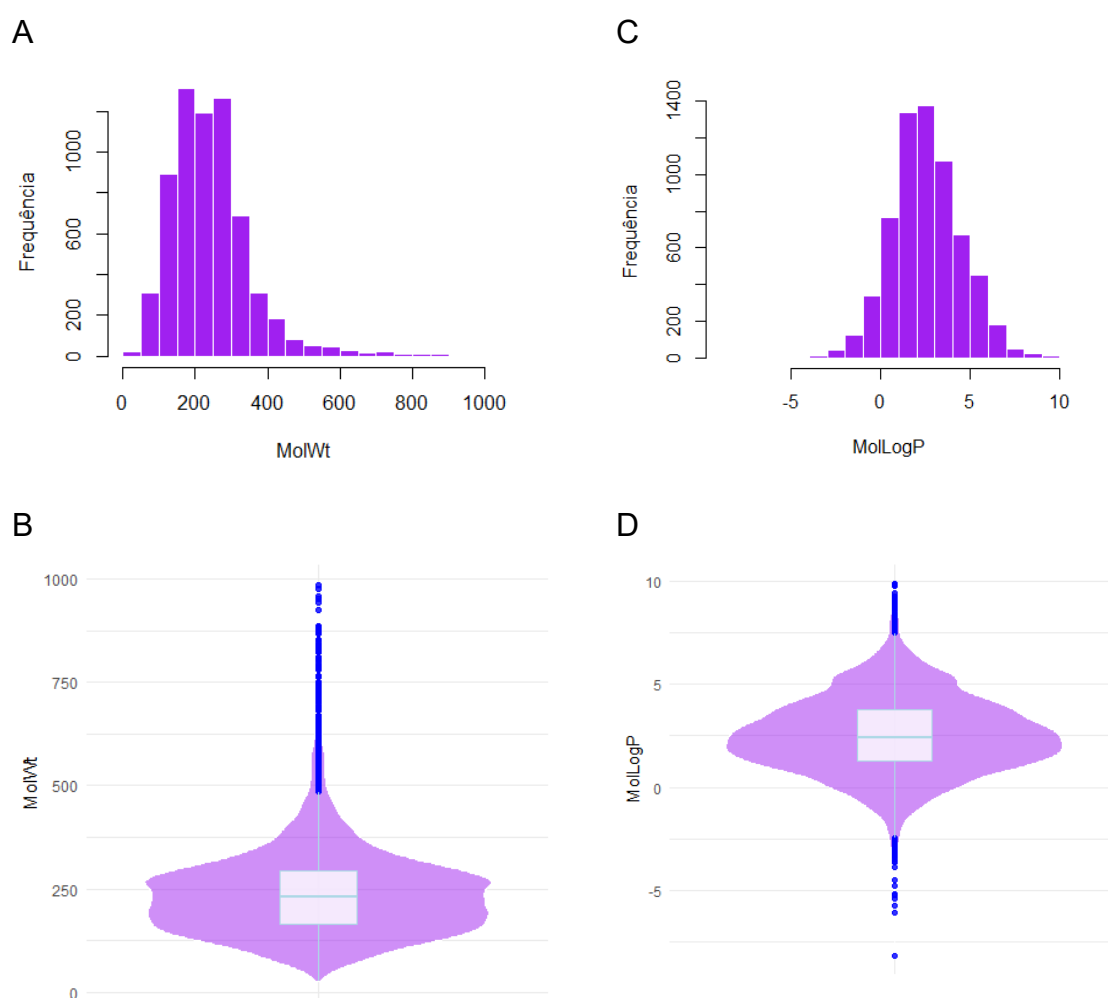
Nota: O *script* em R pode ser consultado no APÊNDICE B.

As variáveis relativas às massas moleculares (MolWt) e coeficiente de partição octanol-água (MolLogP, entre outras) são diretamente relacionadas aos mecanismos de absorção, distribuição, ação, metabolismo e excreção dos compostos. Pela “regra dos 5 de Lipinski” no desenvolvimento de potenciais fármacos, as chances de um composto não ser bem absorvido ou apresentar baixa permeação em tecidos aumentam quando possuem LogP maiores do que 5, massa molecular superiores a 500, mais de 5 doadores de ligações de hidrogênio e mais de 10 receptores de ligações de hidrogênio (LIPINSKI *et al.*, 2001). Desse modo, as variáveis MolWt e MolLogP foram utilizadas como parâmetro para a definição de um domínio de aplicação dos modelos gerados, excluindo substâncias complexas e atípicas que não seriam absorvidas ou distribuídas em organismos regulares e íntegros.

Com uma média de 243,2 Daltons, a variável MolWt foi utilizada para limitar moléculas com mais de 1.000 Daltons. Assim como demonstrado na Tabela 1 e na Figura 3A e 3B, a grande maioria dos dados (3º Quartil) estão abaixo de 300 Daltons, assumindo uma distribuição simétrica em torno da média, se forem desconsiderados

os valores extremos da direita. Por outro lado, a variável MolLogP, importante propriedade físico-química ligada à solubilidade de um composto em solução (por exemplo, fluído corporais), exibiu média de 2,5 e uma distribuição simétrica em torno da média. Através das Figuras 3C e 3D, vemos que foram poucas as moléculas que apresentaram valores discrepantes, estando a grande maioria centradas entre 1,3 e 3,8. Sobre esta variável, vale ressaltar que também fez parte da definição de aplicabilidade, pela remoção de moléculas com LogP superiores a 10.

Figura 3 – Análise gráfica da variável MolWt.

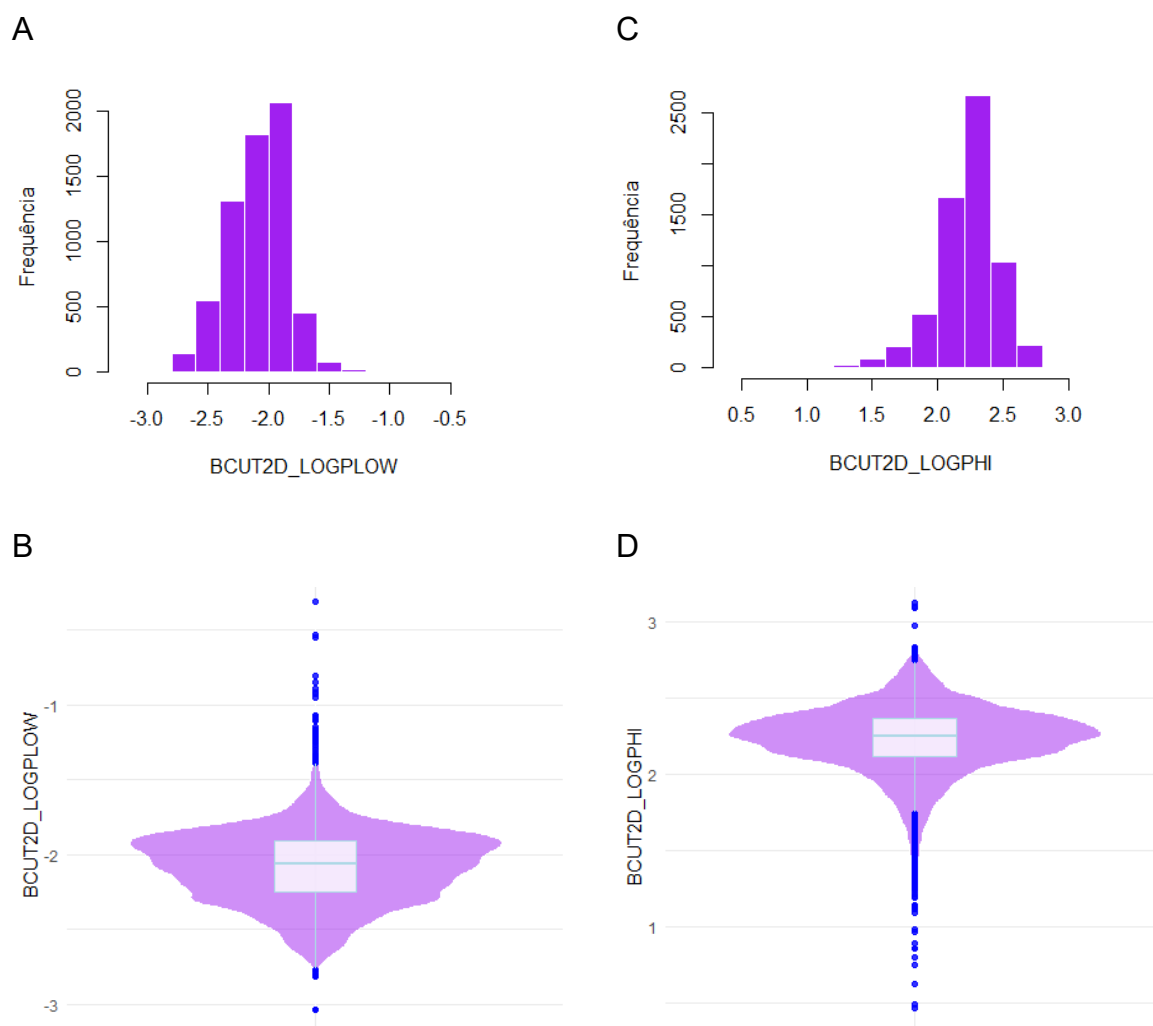


Fonte: Do autor.

Algumas variáveis consideradas na modelagem são determinações locais dentro da própria molécula, explorando máximos e mínimos. Como exemplo, podemos citar as variáveis BCUT2D\_LOGPLOW e BCUT2D\_LOGPHI, que indicam regiões

mais hidrofóbicas e hidrofílicas da molécula, percebidas na Figura 4. Descritores como estes são relevantes dentro do contexto que abrange os mecanismos reacionais entre os grupos químicos das moléculas potencialmente tóxicas e os sítios de ação no organismo, mais especificamente nesse trabalho, nos filamentos de DNA durante a ação mutagênica.

Figura 4 – Análise gráfica das variáveis BCUT2D\_LOGPLOW e BCUT2D\_LOGPHI.

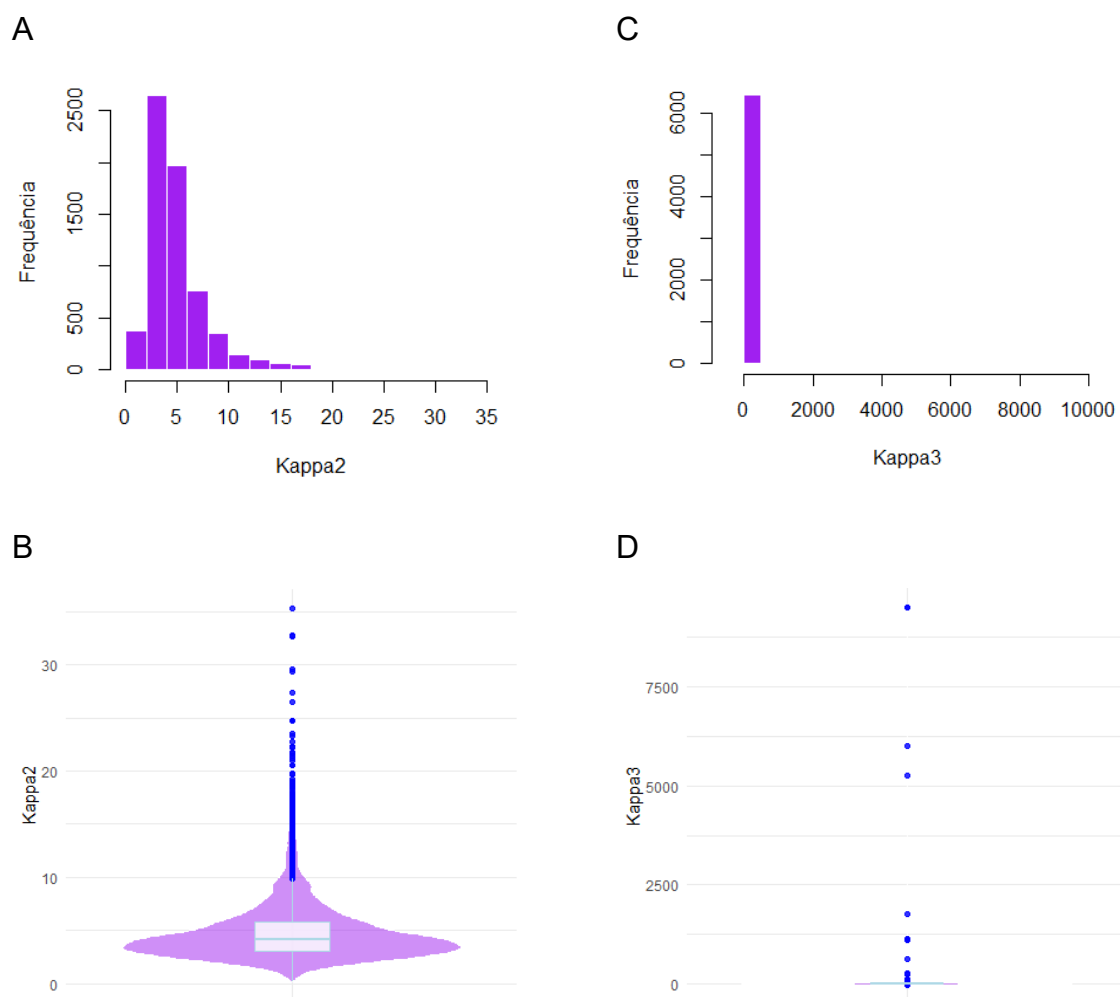


Fonte: Do autor.

Segundo a descrição no Quadro 1, as variáveis Kappa2 e Kappa3 são descritores usados para descrever a complexidade estrutural e a ramificação das moléculas, entretanto, com maneiras diferentes de cálculos (HALL'; KIERT, 1991). Quando olhamos para os valores contidos no primeiro e terceiro quartis, entendemos que Kappa2

e Kappa3 demonstram distribuições parecidas. Entretanto, em decorrência de valores extremamente discrepantes em Kappa3, a análise gráfica não evidencia a real dispersão dos dados (FIGURA 5). Em termos interpretativos, a complexidade da molécula pode ser medida através de parâmetros topológicos (organização dos átomos no espaço). Um farmacóforo (região de interação) requer um arranjo específico de certos átomos para sua atividade, enquanto variações em outras regiões da molécula podem influenciar a atividade biológica. Além disso, a relação estrutural entre átomos não diretamente ligados pode ser crucial para essa variação (HALL'; KIERT, 1991).

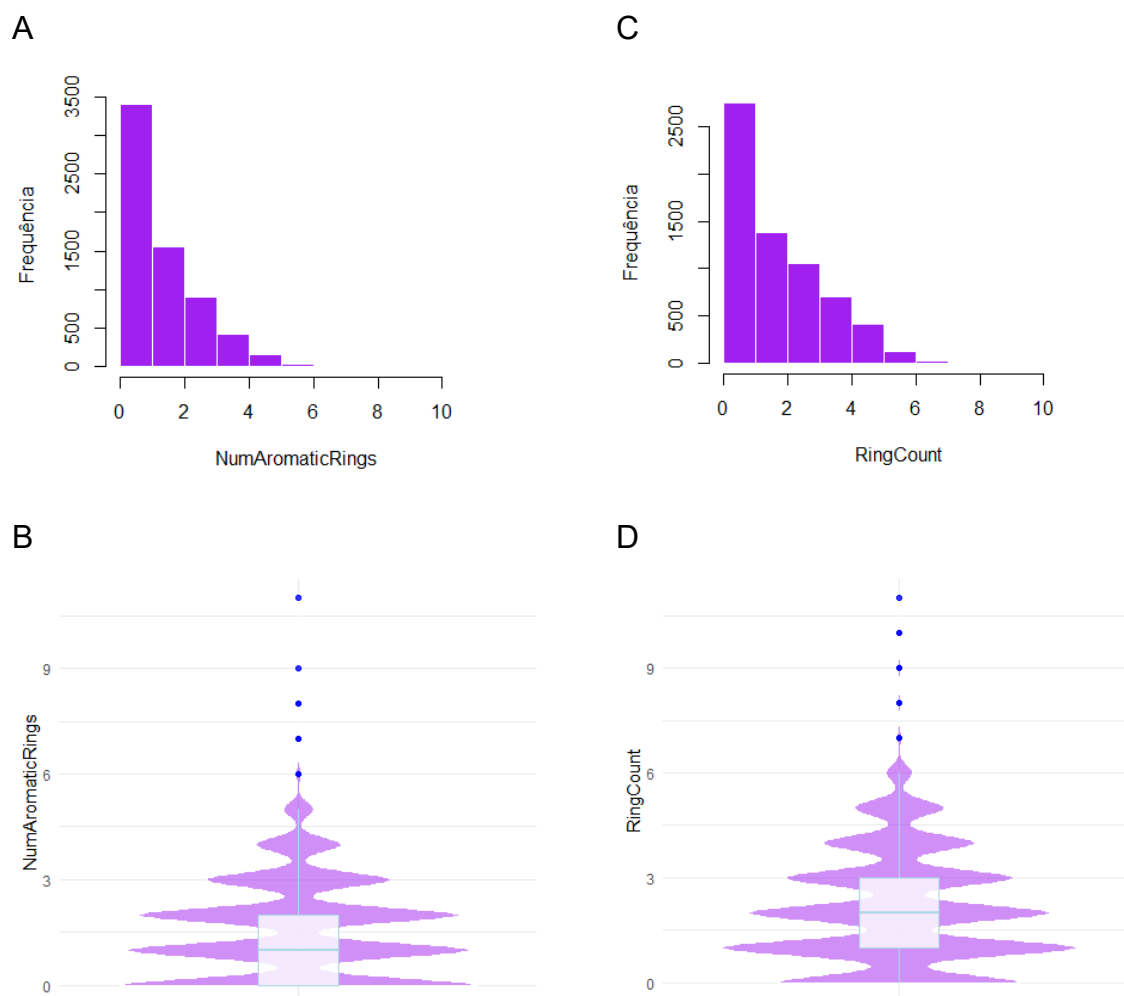
Figura 5 – Análise gráfica das variáveis Kappa 2 e Kappa3.



Fonte: Do autor.

Variáveis como: FractionCSP3, NumAliphaticRings, NumAromaticRings, NumHAcceptors, NumHDonors, NumRotatableBonds, NumSaturatedRings e RingCount, são resultados de contagem de características moleculares estruturais influentes na reatividade intrínseca dos compostos. Na Figura 6, visualmente tem-se a percepção de sobreposição dos dados, suscitando não contribuição individual dos mesmos. Porém, avaliando atentamente e entendendo a origem dados, constata-se que a contagem de anéis, também engloba anéis aromáticos. Como a aromaticidade de anéis confere um maior potencial de interações nos sítios de toxicidade presentes no organismo, esta distinção dos dados pode ser relevante na produção dos modelos de predição.

Figura 6 – Análise gráfica das variáveis NumAromaticRings e RingCount.

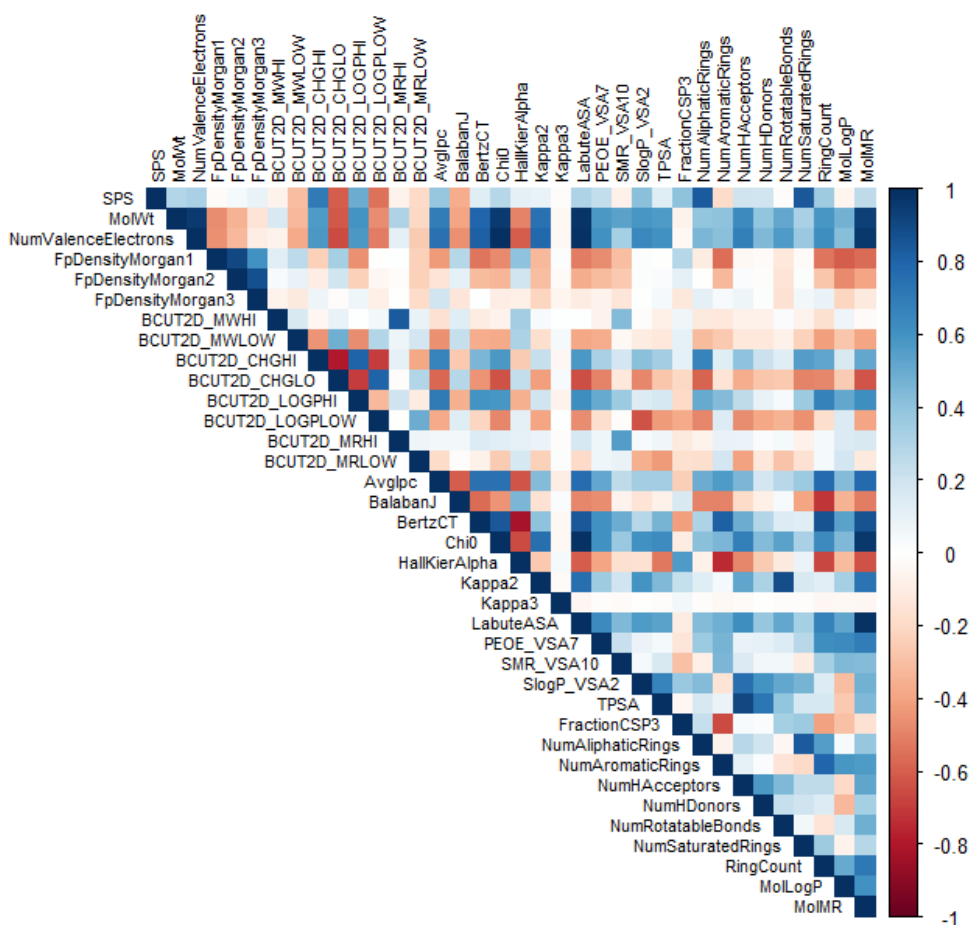


Fonte: Do autor.

Na Figura 7, são representadas as correlações das variáveis preditoras. Dentre as variáveis estudadas, algumas apresentam altas correlações entre elas ( $>0,7$ ), principalmente por serem derivadas de parâmetros comuns, como massa molecular, número de átomos etc., ou por estarem ligadas à complexidade estrutural ou atributos topológicos. No último caso, compostos maiores e de maior massa ou número de átomos, tendem a ser mais complexos e a exibirem aspectos topológicos compatíveis.

Certas variáveis foram altamente correlacionadas positivamente entre si, como Chi0 e NumValenceElectrons, LabuteASA, MolMR, com coeficiente de correlação acima de 0,95. Apesar dessas variáveis apontarem para o mesmo sentido, as relações entre elas não são explicadas diretamente, mas por um somatório de fatores ligados à complexidade. Por outro lado, as variáveis HallKierAlpha e BertzCT demonstraram uma correlação negativa de -0,82, mesmo ambas sendo medidas topológicas. Considerando que HallKierAlpha mede a flexibilidade de uma molécula com base na sua topologia, avaliando a ramificação e a conectividade da molécula, é de se esperar que compostos altamente complexos (com muitos anéis, ramificações e conexões rígidas) tendam a ser menos flexíveis, resultando em valores altos de BertzCT (quanto maior o valor, mais complexa é a estrutura da molécula) e baixos de HallKierAlpha (HALL'; KIERT, 1991).

Figura 7 – Matriz gráfica de correlações entre as variáveis preditoras.



Fonte: Do autor.

## 5.2 MODELOS DE PREDIÇÃO DE MUTAGENICIDADE IN VITRO

As metodologias que envolvem as relações estrutura e atividade das moléculas, tanto nas áreas de desenvolvimento de novos fármacos quanto em avaliações de potenciais efeitos tóxicos de reagentes, aditivos alimentícios, agrotóxicos, produtos de degradação ou metabólicos de xenobióticos, tem sido uma alternativa à estudos antes realizados em animais. Portanto, os chamados modelos QSAR ou *in silico* foram explorados na presente monografia para a classificação de moléculas em mutagênicas ou não mutagênicas, importante desfecho empregado na avaliação da toxicidade e avaliação do risco de compostos de interesse toxicológico. Para tal finalidade, lançou-se mão de técnicas de estatística multivariada e aprendizado de máquina.

### 5.2.1 Primeira abordagem: ajuste sem a definição de espaço químico

O primeiro modelo de predição foi desenvolvido com todos os dados do banco de treinamento, ou seja, sem a definição de um espaço químico criado por afinidade com a molécula-alvo.

Os modelos de regressão utilizados no primeiro ensaio foram: árvores aleatórias, árvores de decisão e XGBoost (APÊNDICE C). Em um primeiro momento, os dados foram divididos em grupo de treinamento (5.174 moléculas; 80%) e grupo teste (1.293 moléculas; 20%) para a validação cruzada externa e avaliação da qualidade de ajuste. Ainda, a validação cruzada por K-Fold foi realizada pela função *trainControl()* da biblioteca *caret*, pelo método de reamostragem *cross-validation* (cv), utilizando k=10 *folds* (subconjuntos) para treino e teste.

Para a otimização do modelo de árvores aleatórias, empregou-se o banco de treinamento, a variável binária ACTIVITY e as 36 variáveis preditoras. O valor escolhido para o parâmetro *mtry* foi 19. No algoritmo Random Forest, esse parâmetro representa o número de variáveis preditoras selecionadas aleatoriamente em cada divisão das árvores do modelo. Portanto, para o presente caso, um valor intermediário maximizou os resultados de área sob a curva ROC, sensibilidade e especificidade (TABELA 2).

Tabela 2 – Resultados do modelo Random Forest.

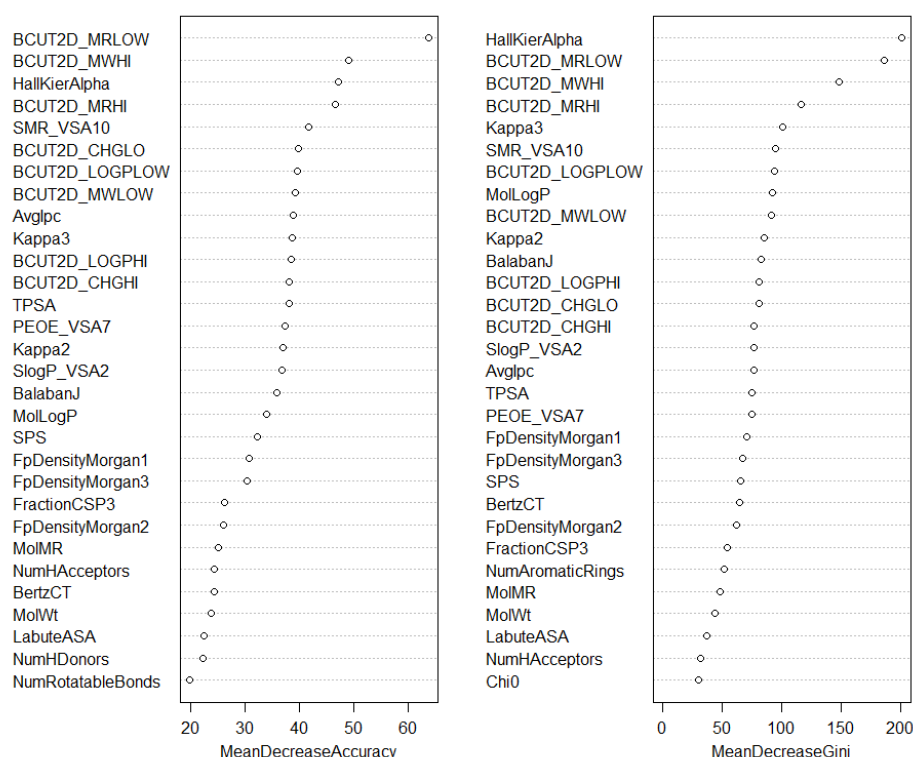
| <i>mtry</i> | AUC ROC | Sensibilidade | Especificidade |
|-------------|---------|---------------|----------------|
| 2           | 0,8577  | 0,7803        | 0,7999         |
| 19          | 0,8661  | 0,7812        | 0,7963         |
| 36          | 0,8621  | 0,7703        | 0,7995         |

Fonte: Do autor.

Na Figura 8, são demonstradas as variáveis que apresentaram maior relevância na predição da mutagenicidade das moléculas. Os critérios foram a *MeanDecreaseAccuracy*, que mede a redução da acurácia do modelo quando a variável é removida e a *MeanDecreaseGini* que, por sua vez, avalia o impacto de cada variável na redução da impureza (o quão misturados os dados estão dentro de um nó), ajudando

a identificar quais variáveis são mais importantes para a predição. O Índice de Gini, especificamente, mede a probabilidade de uma amostra ser classificada incorretamente se fosse escolhida aleatoriamente dentro do nó. Desse modo, as variáveis BCUT2D\_MRLOW, BCUT2D\_MWHI, BCUT2D\_MRHI e HallKierAlpha foram especialmente relevantes, demonstrando que as variáveis topológicas são de alguma maneira importantes na classificação de mutagenicidade.

Figura 8 – Importância das variáveis no Random Forest.



Fonte: Do autor.

O modelo de árvores de decisão CART foi utilizado através do algoritmo *rpart* (*Recursive Partitioning and Regression Trees*), que divide os dados em ramificações baseadas nas variáveis preditoras para prever a variável resposta, ACTIVITY. A área sob a curva ROC foi empregada como métrica para avaliar a capacidade do modelo de distinguir as classes. Assim como no modelo Random Forest, foi usada a validação cruzada por K-Fold.

A avaliação do modelo foi feita testando diferentes valores do parâmetro *cp* (*Complexity Parameter*), que regula o *pruning* (poda) da árvore. Quanto maior o *cp*, mais agressiva é a poda da árvore, reduzindo o tamanho e a complexidade do modelo.

Como demonstrado na Tabela 3, o melhor valor de *cp* foi 0,01596, pois produziu o maior ROC (0,6662), indicando melhor capacidade preditiva.

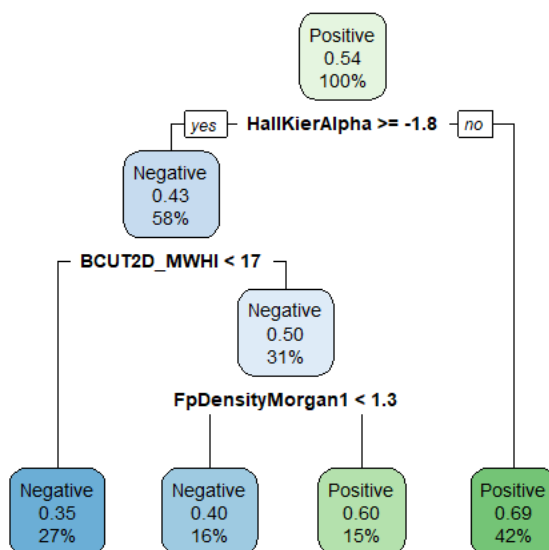
Tabela 3 – Resultados do modelo CART.

| <i>cp</i> | ROC    | Sensibilidade | Especificidade |
|-----------|--------|---------------|----------------|
| 0,01596   | 0,6662 | 0,5801        | 0,7128         |
| 0,03297   | 0,6279 | 0,6636        | 0,5889         |
| 0,18186   | 0,5994 | 0,5354        | 0,6633         |

Fonte: Do autor.

A saída gráfica utilizada para as tomadas de decisão da classificação executada pelo modelo CART é exibida na Figura 9. No esquema podemos verificar a estrutura da árvore de decisão usada para prever a mutagenicidade e as variáveis mais influentes. Nota-se que as variáveis BCUT2D\_MWHI e HallKierAlpha também foram consideradas em grau de importância na regressão por árvores aleatórias, ampliando as evidências sobre o papel dessas variáveis no desfecho estudado.

Figura 9 – Visualização gráfica das árvores de decisão.



Fonte: Do autor.

Com o algoritmo XGBoost (*eXtreme Gradient Boosting*), usando a função *train()* do pacote *caret*, um modelo foi gerado combinando múltiplas árvores de decisão em

um processo de *boosting*, otimizando os erros em cada iteração. Durante o desenvolvimento, notou-se a necessidade de ajustar valores fixos para hiperparâmetros para garantir reprodutibilidade e evitar excessivo trabalho computacional. Para tanto, limitou-se a 100 iterações, com profundidade máxima de árvores de decisão igual a 3, uma taxa de aprendizado (*step wise*) igual a 0,3 e utilização de 100% dos dados para cada árvore. Por fim, assim como apresentada na discussão dos modelos Random Forest e CART, para a validação cruzada também foi usado K-Fold (K-Fold = 10).

Os resultados do modelo XGBoost mostraram um desempenho sólido na predição da mutagenicidade. O valor de ROC (0,846) indica uma boa capacidade do modelo em discriminar entre moléculas mutagênicas e não mutagênicas, sendo um indicativo da qualidade geral da classificação. A sensibilidade (0,7417) reflete a proporção de casos positivos corretamente identificados, sugerindo que o modelo consegue detectar a maioria das moléculas mutagênicas. Já a especificidade (0,7949) mostra a capacidade do modelo em identificar corretamente as moléculas não mutagênicas, reduzindo falsos positivos. Com isso, o XGBoost demonstrou um equilíbrio entre a detecção de mutagenicidade e a minimização de erros.

A comparação dos modelos Random Forest, CART e XGBoost evidenciou diferenças substanciais no desempenho na predição da mutagenicidade (TABELA 4 e FIGURA 10). O Random Forest apresentou o melhor desempenho global, com uma AUC ROC (0,8715) elevada, indicando desejável capacidade discriminativa entre as classes. Além disso, obteve a maior acurácia (0,7981) e F1-score (0,7786), demonstrando um equilíbrio adequado entre precisão e *recall* (sensibilidade). O modelo XGBoost também apresentou um desempenho competitivo, com ROC de 0,8518 e acurácia de 0,7711, o que sugere uma boa capacidade preditiva, embora ligeiramente inferior ao RF. Em contrapartida, o modelo CART obteve os piores resultados, com ROC de 0,6583 e acurácia de 0,6450, refletindo uma menor eficácia na classificação. Sua sensibilidade (0,5714) foi a mais baixa entre os modelos analisados, o que indica dificuldades na identificação correta das moléculas positivas. De maneira geral, os resultados sugerem que Random Forest e XGBoost são modelos adequados para a predição da mutagenicidade, enquanto o CART apresenta limitações significativas para essa aplicação.

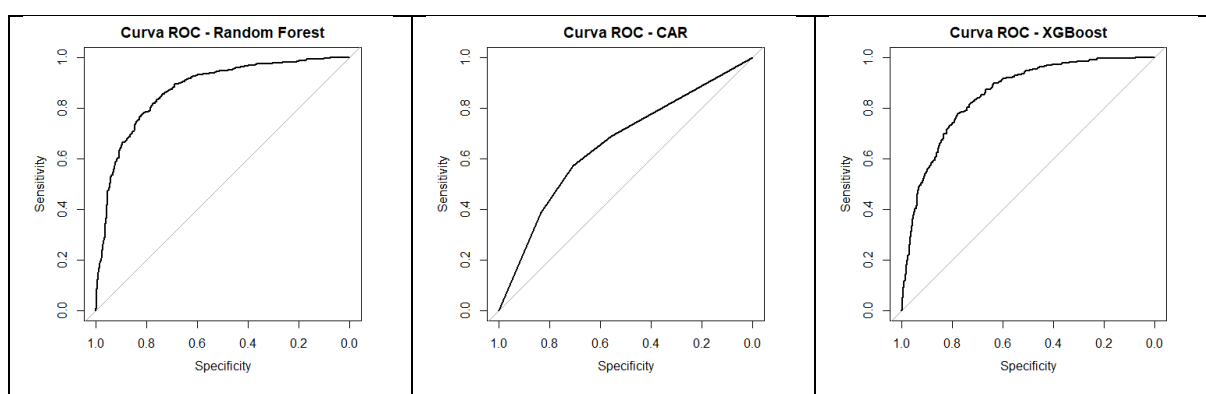
Tabela 4 – Métricas de avaliação dos modelos de regressão (ponto de corte = 0,5).

| Modelo  | ROC    | Acurácia | Sens.  | Esp.   | F1-score | P (+)  | P (-)  | Atividade* |
|---------|--------|----------|--------|--------|----------|--------|--------|------------|
| RF      | 0,8715 | 0,7981   | 0,7714 | 0,8209 | 0,7786   | 0,3960 | 0,8715 | Neg.       |
| CART    | 0,6583 | 0,6450   | 0,5714 | 0,7077 | 0,5970   | 0,6925 | 0,6583 | Pos.       |
| XGBoost | 0,8518 | 0,7711   | 0,7412 | 0,7966 | 0,7487   | 0,4695 | 0,8518 | Neg.       |

Fonte: Do autor.

Nota: Sens.: Sensibilidade; Esp.: Especificidade; P (+): probabilidade associada à predição positiva; P (-): probabilidade associada à predição negativa; Neg.: Negativo; Pos.: Positivo. \*Atividade mutagênica.

Figura 10 – Curva ROC dos modelos analisados.



Fonte: Do autor.

Quando os dados são balanceados, o ponto de corte ótimo geralmente se aproxima do valor padrão de 0,5, pois a distribuição das classes permite que os algoritmos mantenham um equilíbrio entre sensibilidade e especificidade. No entanto, nos casos que os dados são desbalanceados, onde uma classe é significativamente mais representada do que a outra, o uso do *threshold* padrão pode levar a um viés na predição, favorecendo a classe majoritária e resultando em baixa sensibilidade para a classe minoritária. Nesses casos, é essencial ajustar o ponto de corte para maximizar métricas como F1-score ou minimizar custos associados a erros de classificação (YODEN, 1950). Dessa forma, a escolha do ponto de corte ideal deve ser feita com base em métricas que refletem o objetivo do modelo, garantindo uma classificação mais equilibrada e eficaz.

No presente estudo, o ponto de corte ótimo foi determinado automaticamente pelo método de Youden para os modelos Random Forest (RF) e XGBoost, enquanto para o CART, foi utilizada a mediana das probabilidades preditas, garantindo um

ajuste mais adequado às características de cada modelo (YOUDEM, 1950). O *script* em R pode ser acessado no APÊNDICE D.

Portanto, o ponto de corte ótimo para os modelos Random Forest, CART e XGBoost foram, respectivamente, 0,4950, 0,6037 e 0,5379. Como os dados não são desbalanceados, os valores ficaram próximos ao *threshold* padrão (0,5) usado pelos algoritmos.

A aplicação do ponto de corte ótimo, visualizada na Tabela 5, resultou em melhorias sutis nos modelos, especialmente no XGBoost, que apresentou um aumento na sensibilidade (0,7412 para 0,7765) e no F1-score (0,7487 para 0,7624), tornando-se o modelo mais equilibrado entre sensibilidade e especificidade. O Random Forest teve um leve aumento na acurácia (0,7981 para 0,7997) e na especificidade (0,8209 para 0,8281), embora sua sensibilidade tenha caído ligeiramente, mantendo-se um modelo robusto. O CART, por outro lado, não apresentou mudanças significativas, reforçando sua menor capacidade discriminativa (ROC de 0,6583). No geral, os ajustes reforçam a importância da definição do ponto de corte, especialmente em cenários desbalanceados, onde a escolha adequada pode melhorar a detecção de uma classe sem comprometer excessivamente a outra.

Tabela 5 – Métricas de avaliação dos modelos de regressão com ponto de corte ótimo.

| Modelo  | ROC    | Acurácia | Sens.  | Esp.   | F1-score | P (+)  | P (-)  | Atividade* |
|---------|--------|----------|--------|--------|----------|--------|--------|------------|
| RF      | 0,8715 | 0,7997   | 0,7664 | 0,8281 | 0,7788   | 0,3960 | 0,6040 | Neg.       |
| CART    | 0,6583 | 0,6450   | 0,5714 | 0,7077 | 0,5970   | 0,6925 | 0,3075 | Pos.       |
| XGBoost | 0,8518 | 0,7773   | 0,7765 | 0,7779 | 0,7624   | 0,4695 | 0,5305 | Neg.       |

Fonte: Do autor.

Nota: Sens.: Sensibilidade; Esp.: Especificidade; P (+): probabilidade associada à predição positiva; P (-): probabilidade associada à predição negativa; Neg.: Negativo; Pos.: Positivo. \*Atividade mutagênica.

Em suma, ao analisar os resultados de predição de mutagenicidade obtidos pelos três modelos testados nesta primeira abordagem, considerando-os satisfatórios (RF e XGBoost), pode-se concluir que o composto avaliado apresenta uma alta probabilidade de não ser mutagênico, o que está em concordância com os estudos *in vitro* (Teste de Ames) citados na literatura (SNYDER, 2009).

### 5.2.2 Segunda abordagem: espaço químico otimizado por técnicas de agrupamento

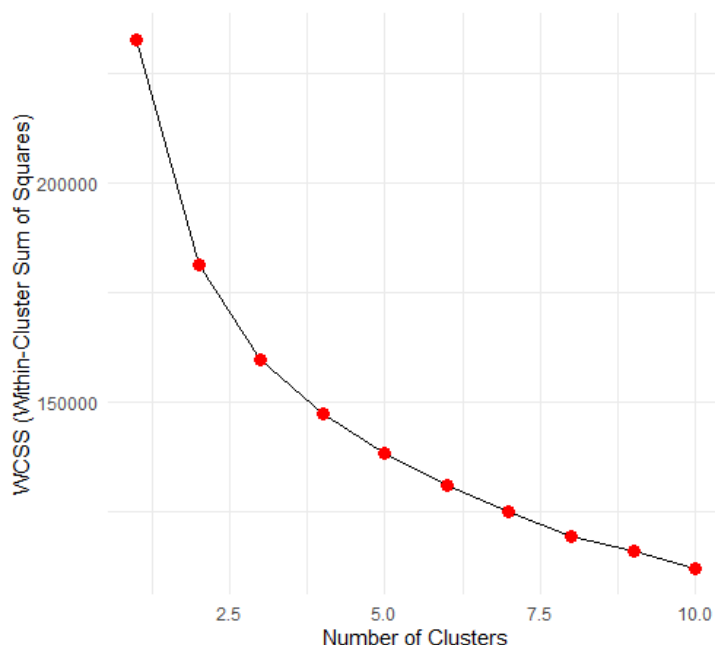
A segunda abordagem desenvolvida foi conduzida com o intuito de aproximar o espaço químico da molécula-alvo e reduzir o número de moléculas usadas para modelagem, visto que o algoritmo Random Forest demanda grande poder computacional e tempo quando aplicado a volumes elevados de dados.

Inicialmente, os dados foram carregados, filtrados (domínio de aplicação: massa molecular < 1.000 Daltons; LogP < 10) e normalizados ( $\mu = 0$ ;  $\sigma = 1$ ). Em seguida, foi aplicada a técnica de agrupamento *K-means*, com o número ideal de *clusters* sendo determinado a partir do método cotovelo (*Elbow Method*) e da análise do coeficiente de silhueta. Para visualizar a distribuição das moléculas no espaço químico, foi utilizada a redução de dimensionalidade por PCA (Análise de Componentes Principais), plotando os componentes PCA1 e PCA2. Além disso, a molécula-alvo foi projetada no espaço PCA para identificar o cluster mais próximo, permitindo a seleção de moléculas similares e reduzindo o número de compostos usados para a modelagem.

Inicialmente, para a definição do número de *clusters* representativos dos espaços químicos possíveis para a análise de mutagenicidade *in silico*, foi utilizado o método *elbow*. O método baseia-se na soma dos erros quadráticos dentro de cada *cluster* (*Within-Cluster Sum of Squares* - WCSS), indicando o quão compactos estão os grupos formados. Inicialmente, a variabilidade *intra-cluster* diminui rapidamente à medida que *k* aumenta, pois os pontos ficam mais próximos dos centróides. No entanto, em determinado ponto, a taxa de redução se torna menos significativa, formando um "cotovelo" na curva. Esse ponto de inflexão representa o melhor compromisso entre coesão e separação dos *clusters*, sendo escolhido como o número ideal de grupos (SCHUBERT, 2022).

Através da aplicação do método *elbow*, como apresentado na Figura 11, não foi possível identificar uma região clara do ponto de inflexão, mesmo após a avaliação de até 10 grupos (*k* = 10).

Figura 11 – Visualização gráfica do número (k) ideal de clusters (Método *Elbow*).

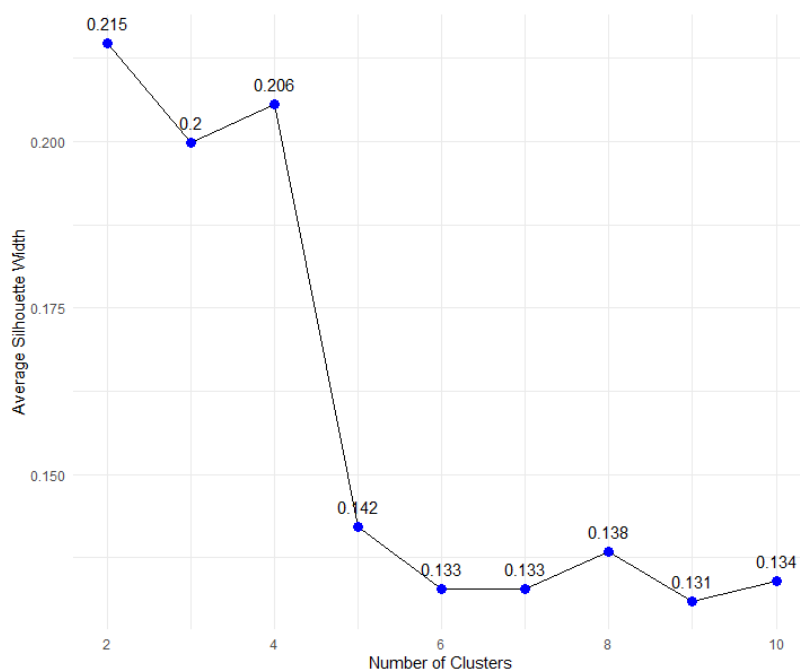


Fonte: Do autor.

Uma vez que o método *elbow* não foi decisivo para a determinação do número ideal de *clusters*, a análise de silhueta foi utilizada (FIGURA 12). Esse método calcula o coeficiente de silhueta para cada amostra, variando de -1 a 1, em que valores mais altos indicam que as moléculas estão bem agrupadas e distantes de *clusters* vizinhos. Especificamente, este coeficiente é calculado comparando duas medidas principais: a coesão, representada pela distância média entre um ponto e os outros pontos do mesmo *cluster*, e a separação, que corresponde à menor distância média entre esse ponto e os elementos do *cluster* mais próximo ao qual ele não pertence. A média desses coeficientes fornece uma medida geral da validade da segmentação adotada, permitindo a identificação de *clusters* mais homogêneos e bem definidos, além de auxiliar na escolha do número ideal de agrupamentos (DE AMORIM; HENNIG, 2015).

Os valores médios do coeficiente de silhueta obtidos para 2, 3 e 4 *clusters* indicam que eles possuem baixa qualidade e grau de separação. Como esse índice varia entre -1 e 1, valores próximos de 0,2 sugerem que muitos pontos estão próximos das fronteiras entre *clusters*, indicando uma sobreposição significativa ou a ausência de uma estrutura bem definida nos dados. Por estes resultados, sugere-se que o banco de dados não possui uma separação natural clara, apesar de fornecer certa segmentação.

Figura 12 – Análise do coeficiente de silhueta.

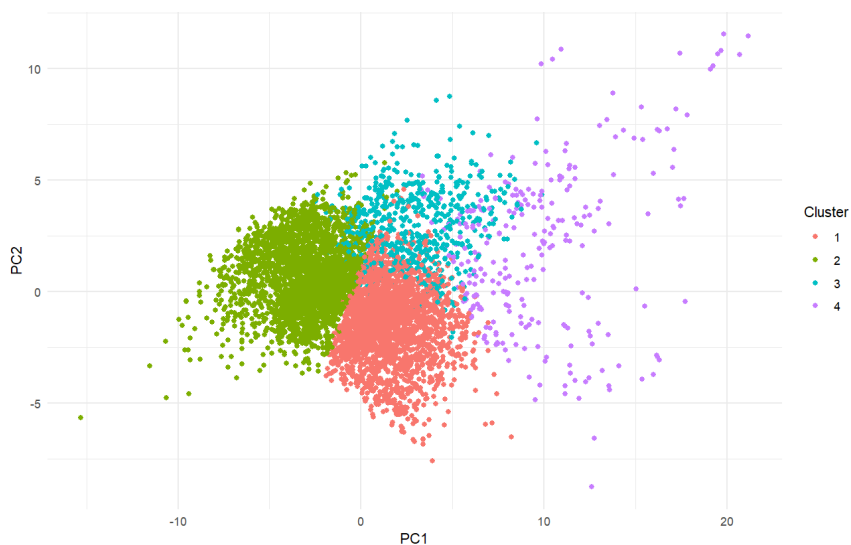


Fonte: Do autor.

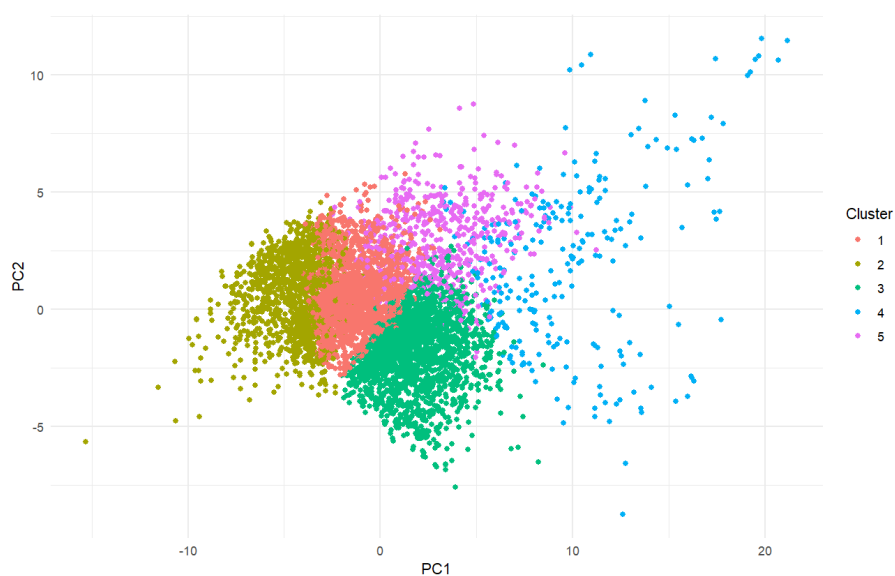
Em vista das conclusões acerca das análises para a identificação do número ideal de agrupamentos, analiticamente poderíamos considerar até 4 clusters (FIGURA 12). Entretanto, ao se observar a separação de grupos na projeção das componentes principais 1 e 2, com 4 clusters (FIGURA 13A) e com 5 clusters (FIGURA 13B), não estaríamos restringindo o espaço químico, ao considerarmos 4 clusters. Desse modo, em atenção ao propósito da redução do banco de dados a clusters com similaridades físico-químicas, seguiu-se  $K = 5$  por exibir grupos com quantidades razoáveis de moléculas.

Figura 13 – Representação gráfica da análise de *clusters*.

A



B

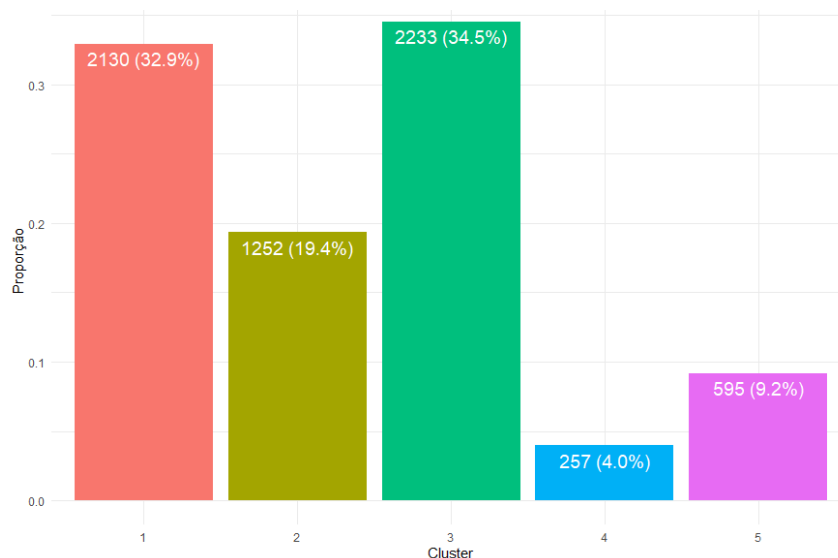


Fonte: Do autor.

Como pode ser observado na Figura 14, o *Cluster 3* apresentou o maior número de amostras (2.233), seguido pelo *Cluster 1* (2.130) e pelo *Cluster 2* (1.252), enquanto os *Clusters 4* e *5* apresentaram quantidades significativamente menores (257 e 595, respectivamente). Essa diferença nos tamanhos dos *clusters* pode afetar o desempenho do modelo, principalmente em algoritmos sensíveis à distribuição dos dados, como Random Forest, que demanda maior poder computacional e tempo de processamento para grandes volumes de dados. *Clusters* com menos amostras, como o

*Cluster* 4, podem ter menor representatividade, dificultando a captura de padrões e reduzindo a confiabilidade das previsões associadas a esse grupo.

Figura 14 – Proporção das moléculas nos clusters avaliados.



Fonte: Do autor.

A análise dos dados revelou discrepâncias significativas na distribuição das classes, como no *Cluster* 3, onde a classe 1 (mutagênica) representou 66,9%, e no *Cluster* 4, onde a classe 0 (não mutagênica) dominou com 62,3% (Figura 15). Para corrigir o desbalanceamento, principalmente nos clusters mais desbalanceados, utilizou-se o ponto de corte ótimo determinado pela estatística de Youden. Como já discutido, essa técnica foi aplicada para identificar o limiar que maximiza simultaneamente a sensibilidade e a especificidade, equilibrando as classificações. A eficácia dessa abordagem foi validada pelas métricas contempladas nesse trabalho, garantindo uma melhor generalização do modelo.

Figura 15 – Avaliação do balanceamento das classes em cada *cluster*.

Fonte: Do autor.

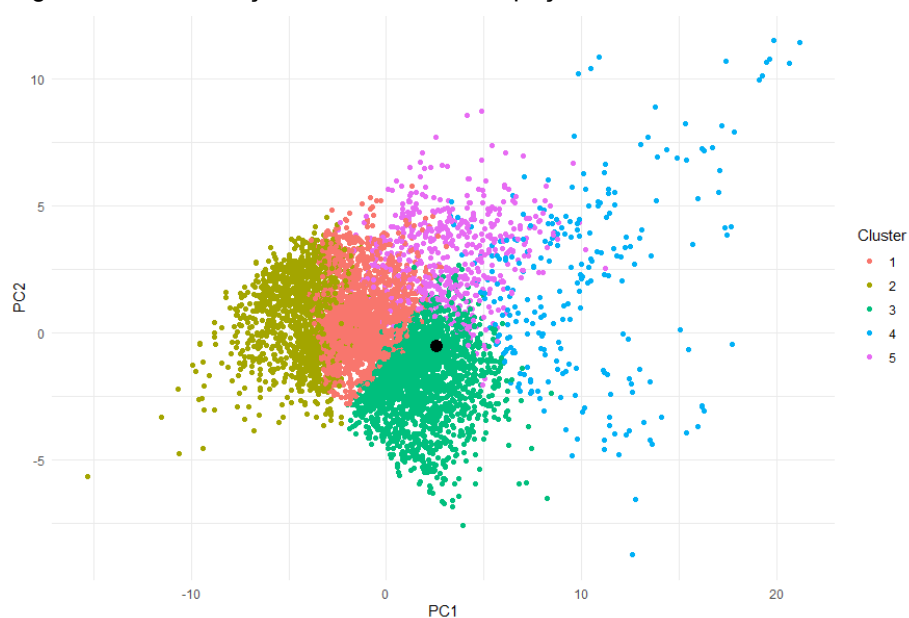
Nota: Atividade (ACTIVITY): 0 – não mutagênico; 1 – mutagênico.

Assume-se que a qualidade das predições poderá variar em função da molécula-alvo utilizada e do respectivo *cluster* mais próximo (espaço químico) devido às diferenças na distribuição e representatividade dos dados, portanto, sendo um fator limitante na aplicação dessa abordagem. O *Cluster 4*, por exemplo, por possuir um número significativamente menor de amostras, apresenta uma maior dispersão, o que pode indicar uma variabilidade estrutural elevada entre as moléculas que o compõem. Essa dispersão pode ser visualizada pela plotagem das componentes principais PCA1 e PCA2 (Figura 16), que revelam a distribuição espacial dos *clusters* no espaço químico.

Enquanto *clusters* maiores podem oferecer um conjunto de dados mais estável para o aprendizado do modelo, *clusters* menores e/ou menos homogêneos, podem sofrer com menor robustez estatística. Isso pode adicionar incertezas nas predições dentro desses grupos, tornando-se essencial a definição de um domínio de aplicação mais restrito (em um contexto prático, moléculas que se associassem a estes *clusters* deveriam ser consideradas “fora de domínio”, demandando avaliação por outras ferramentas).

Para identificar o *cluster* mais próximo da molécula-alvo, utilizou-se a métrica de distância euclidiana entre os centroides dos *clusters* e os valores normalizados da molécula-alvo. Primeiramente, os centroides foram extraídos a partir dos resultados do *K-means*, representando os pontos centrais de cada agrupamento no espaço das características preditoras. Em seguida, para cada centroide, foi calculada a distância euclidiana em relação à molécula-alvo, permitindo a identificação do cluster mais próximo com base na menor distância. Esse método viabilizou a localização da molécula-alvo dentro do espaço químico e, no caso do omeprazol, como ilustrado nas Figuras 16 e 17, evidenciou sua maior proximidade em relação ao Cluster 3.

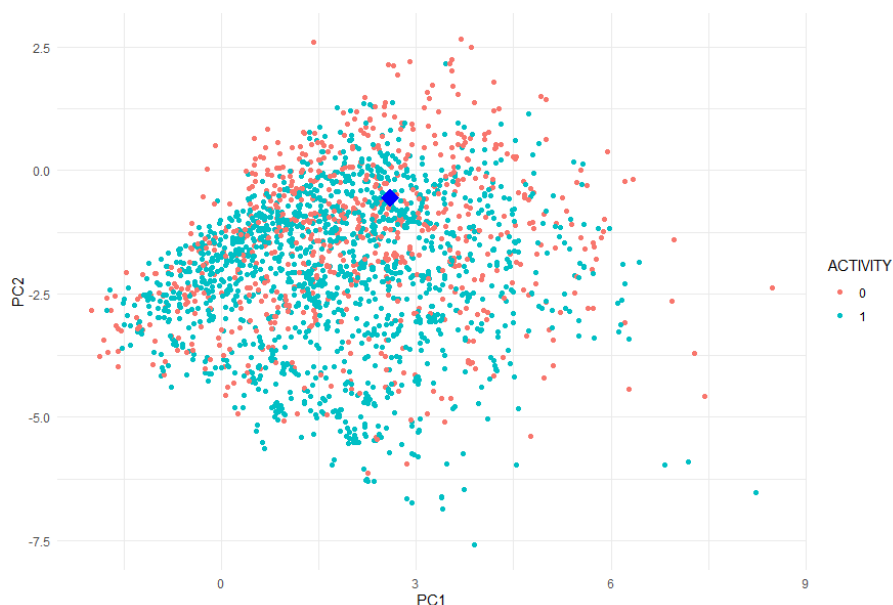
Figura 16 – Distribuição dos clusters no espaço PCA.



Fonte: Do autor.

Nota: O ponto preto (●) representa a localização da molécula-alvo (omeprazol) no espaço químico.

Figura 17 – Localização da molécula-alvo no espaço químico.



Fonte: Do autor.

Nota: Atividade (ACTIVITY): 0 – não mutagênico; 1 – mutagênico. O losango azul (♦) representa a localização da molécula-alvo (omeprazol) no espaço químico (*Cluster 3*).

Por fim, modelos de aprendizado supervisionado, incluindo árvores aleatórias, árvore de decisão (CART) e XGBoost, foram treinados para prever a mutagenicidade das moléculas, considerando apenas aquelas pertencentes ao espaço químico determinado pelo *cluster* mais próximo da molécula-alvo. Foram utilizadas as mesmas configurações de treinamento da primeira abordagem (item 5.2.1), com avaliação de dois pontos de corte: o padrão (0,5) e o ponto de corte ótimo determinado pelo critério de Youden.

Nesta abordagem, o valor escolhido para o parâmetro *mtry* também foi 19, pois apresentou o maior valor para a métrica ROC (0,8700). A análise dos resultados mostra que, à medida que *mtry* aumenta, a sensibilidade do modelo melhora, passando de 63,0% (*mtry* = 2) para 67,3% (*mtry* = 36), o que indica maior capacidade de identificar corretamente as amostras positivas. No entanto, a especificidade sofre uma leve redução, caindo de 90,6% (*mtry* = 2) para 88,9% (*mtry* = 36), sugerindo um pequeno aumento na taxa de falsos positivos. Portanto, o fato de o modelo ótimo ter sido escolhido com *mtry* = 19 indica um equilíbrio entre sensibilidade e especificidade, resultando em um classificador robusto para a previsão da mutagenicidade. Os resultados do modelo podem ser observados na Tabela 6.

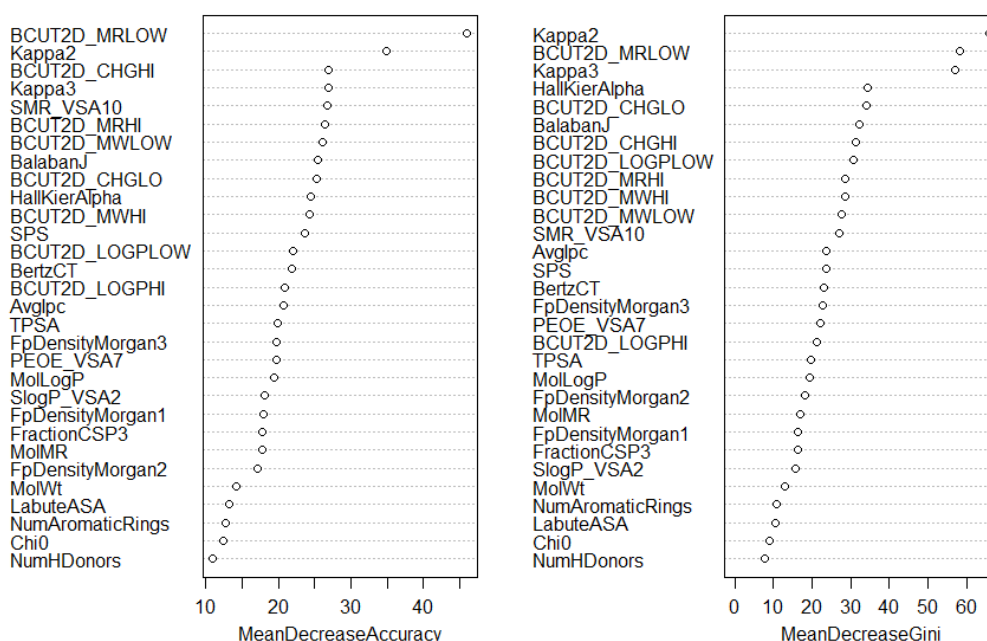
Tabela 6 – Resultados do modelo Random Forest.

| <i>mtry</i> | AUC ROC | Sensibilidade | Especificidade |
|-------------|---------|---------------|----------------|
| 2           | 0,8645  | 0,6303        | 0,9063         |
| 19          | 0,8700  | 0,6658        | 0,8905         |
| 36          | 0,8698  | 0,6726        | 0,8888         |

Fonte: Do autor.

Na Figura 18, são demonstradas as variáveis que apresentaram maior relevância na predição da mutagenicidade das moléculas. Apesar da variável BCUT2D\_MRLOW ter-se mantido como a mais influente, variáveis como Kappa2 e BCUT2D\_CHGHI subiram no ranking. Isso demonstrou que o uso de um ambiente químico mais próximo da molécula-alvo causou alterações na importância das variáveis, sobressaindo aquelas com características ligadas à distribuição de cargas pela molécula e polarização.

Figura 18 - Importância das variáveis no Random Forest.



Fonte: Do autor.

O modelo CART foi avaliado com diferentes valores do hiperparâmetro *cp* (*complexity parameter*). O critério para seleção do modelo ótimo foi o maior valor de ROC, levando à escolha de *cp* = 0,0253, que resultou em ROC = 0,7296 (TABELA 7).



Bem como na primeira abordagem (item 5.2.1), o modelo XGBoost apresentou um bom desempenho, com AUC de 0,853, sensibilidade de 0,650 e especificidade de 0,884, indicando boa capacidade de discriminar moléculas mutagênicas e não mutagênicas. A sensibilidade moderada sugere a presença de falsos negativos, o que pode ser crítico em avaliações toxicológicas, enquanto a alta especificidade reduz falsos positivos. Além disso, para este trabalho os hiperparâmetros do modelo foram mantidos fixos, entretanto, ajustes nos hiperparâmetros e na escolha do ponto de corte (demonstrado a seguir) são opções a serem exploradas para otimizar ainda mais o desempenho.

Os resultados apresentados nas Tabelas 8 e 9 mostram o desempenho dos modelos Random Forest, CART e XGBoost na predição da mutagenicidade utilizando diferentes critérios de ponto de corte. A primeira tabela utiliza o ponto de corte padrão de 0,5, enquanto a segunda aplica o ponto de corte otimizado pelo critério de Youden (0,6170 para o Random Forest, 0,8200 para o CART e 0,7397 para o XGBoost).

Tabela 8 – Métricas de avaliação dos modelos de regressão (ponto de corte = 0,5).

| Modelo  | ROC    | Acurácia | Sens.  | Esp.   | F1-score | P (+)  | P (-)  | Atividade* |
|---------|--------|----------|--------|--------|----------|--------|--------|------------|
| RF      | 0,8598 | 0,8157   | 0,6395 | 0,9027 | 0,6963   | 0,4060 | 0,5940 | Neg.       |
| CART    | 0,6987 | 0,7371   | 0,3741 | 0,9161 | 0,4846   | 0,8200 | 0,1800 | Pos.       |
| XGBoost | 0,8531 | 0,8135   | 0,6054 | 0,9161 | 0,6820   | 0,2932 | 0,7068 | Neg.       |

Fonte: Do autor.

Nota: Sens.: Sensibilidade; Esp.: Especificidade; P (+): probabilidade associada à predição positiva; P (-): probabilidade associada à predição negativa; Neg.: Negativo; Pos.: Positivo. \*Atividade mutagênica.

Tabela 9 – Métricas de avaliação dos modelos de regressão com ponto de corte ótimo.

| Modelo  | ROC    | Acurácia | Sens.  | Esp.   | F1-score | P (+)  | P (-)  | Atividade* |
|---------|--------|----------|--------|--------|----------|--------|--------|------------|
| RF      | 0,8598 | 0,8112   | 0,7211 | 0,8557 | 0,7162   | 0,4060 | 0,5940 | Neg.       |
| CART    | 0,6987 | 0,6742   | 0,6395 | 0,6913 | 0,5646   | 0,8200 | 0,1800 | Pos.       |
| XGBoost | 0,8531 | 0,8067   | 0,7687 | 0,8255 | 0,7244   | 0,2932 | 0,7068 | Neg.       |

Fonte: Do autor.

Nota: Sens.: Sensibilidade; Esp.: Especificidade; P (+): probabilidade associada à predição positiva; P (-): probabilidade associada à predição negativa; Neg.: Negativo; Pos.: Positivo. \*Atividade mutagênica.

Analisando a métrica AUC ROC, que indica a capacidade do modelo de discriminar entre classes, Random Forest e XGBoost apresentaram os melhores desempenhos com AUC de 0,8598 e 0,8531, respectivamente, enquanto o CART demonstrou um desempenho inferior (AUC = 0,6987). Isso sugere que Random Forest e XGBoost possuem uma melhor capacidade preditiva geral em relação ao CART.

Com o ponto de corte padrão, a acurácia foi maior para o Random Forest (0,8157) e o XGBoost (0,8134), enquanto o CART apresentou um desempenho inferior (0,7371). Já ao utilizar o ponto de corte otimizado, a acurácia do Random Forest caiu ligeiramente para 0,8112, enquanto a do XGBoost reduziu para 0,8067, e a do CART teve uma queda mais acentuada para 0,6742. Isso indica que, para o CART, a otimização do ponto de corte comprometeu o equilíbrio entre os acertos e erros, impactando negativamente a performance geral do modelo.

No que se refere à sensibilidade (capacidade de identificar corretamente as moléculas mutagênicas), houve um aumento expressivo para todos os modelos com a otimização do ponto de corte. Para o Random Forest, a sensibilidade subiu de 0,639 para 0,721, para o CART de 0,374 para 0,639, e para o XGBoost de 0,605 para 0,769. Isso indica que, com a otimização, os modelos passaram a identificar mais moléculas mutagênicas corretamente, reduzindo a ocorrência de falsos negativos. Entretanto, essa melhoria veio acompanhada de uma redução na especificidade (capacidade de identificar corretamente as moléculas não mutagênicas), que caiu de 0,902 para 0,855 para o Random Forest, de 0,916 para 0,691 para o CART, e de 0,916 para 0,825 para o XGBoost. Essa queda na especificidade sugere um aumento na taxa de falsos positivos.

No contexto da determinação da segurança toxicológica, é preferível que os modelos apresentem uma menor ocorrência de falsos negativos em relação aos falsos positivos, uma vez que falsos negativos podem resultar na aprovação indevida de compostos potencialmente tóxicos. Em contrapartida, falsos positivos, embora aumentem a necessidade de ensaios *in vitro* para confirmação, representam uma abordagem mais conservadora e segura na triagem de substâncias.

Quanto às probabilidades associadas às predições, percebe-se que, com o ponto de corte padrão, o Random Forest indicou uma probabilidade de 40,6% para a classe positiva (mutagênico) e 59,4% para a negativa (não mutagênico). Com a otimização, a probabilidade positiva subiu para 61,7%, refletindo um ajuste do modelo para

detectar mais mutagênicos. No caso do XGBoost, a probabilidade da classe positiva subiu de 29,3% para 73,9%, reforçando um comportamento mais sensível após a otimização. Já no CART, a probabilidade positiva permaneceu elevada (81,9%) em ambas as abordagens, indicando que o modelo mantém uma tendência mais forte para classificar as moléculas como mutagênicas.

Considerando o cluster-alvo (Cluster 3), esses resultados mostram que a escolha do ponto de corte impacta significativamente a predição. A otimização do ponto de corte demonstrou ser uma estratégia eficiente para aumentar a sensibilidade, tornando os modelos mais capazes de identificar moléculas mutagênicas.

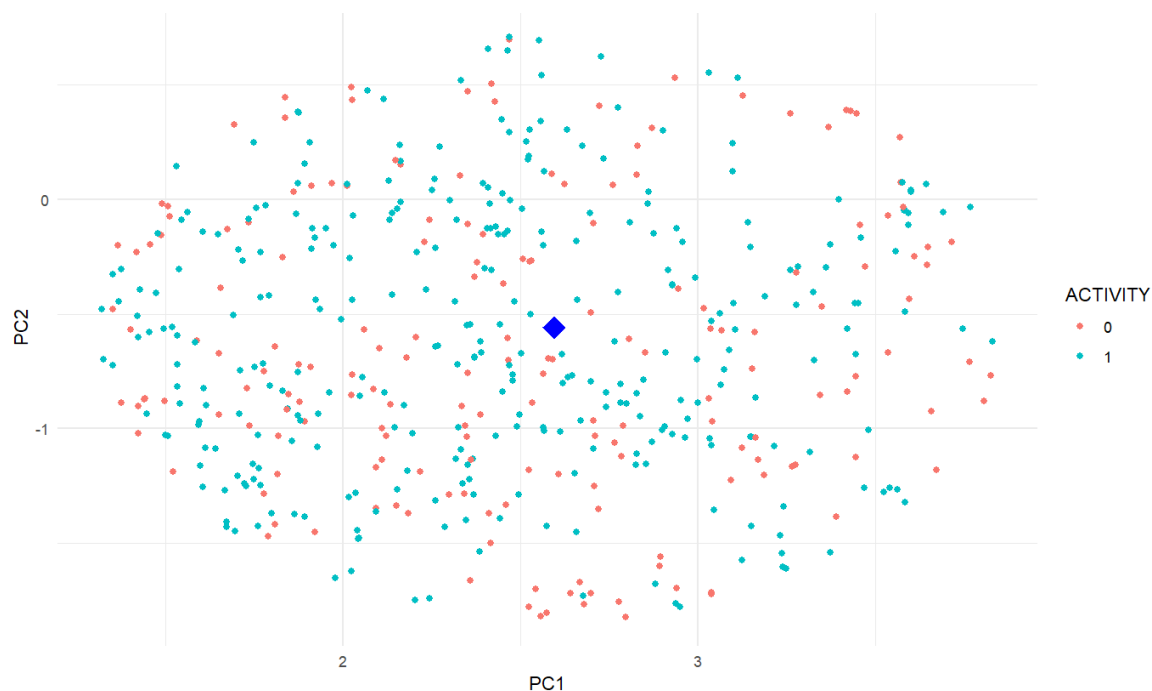
Os *scripts* em R podem ser acessados nos APÊNDICES E e F.

### 5.2.3 Terceira abordagem: espaço químico definido pelas distâncias euclidianas

Uma terceira via baseada na escolha de um espaço químico com número definido de moléculas e distâncias euclidianas centradas na molécula-alvo foi seguida objetivando-se uma melhor captura das propriedades adjacentes.

Inicialmente, os descritores moleculares do conjunto de dados foram normalizados utilizando *escore z*, garantindo que cada variável tenha média zero e desvio padrão igual a um, o que evita vieses devido a diferenças de escala. Em seguida, a matriz de covariância dos descritores foi decomposta para extrair os autovalores e autovetores, resultando em componentes principais ortogonais, ordenados por sua capacidade de explicar a variância total dos dados. Para definir o espaço químico relevante, calculou-se a distância euclidiana entre a molécula-alvo (omeprazol) e as demais moléculas nos dois primeiros componentes principais (PC1 e PC2), que explicam a maior parte da variância do conjunto de dados. As 500 moléculas mais próximas foram selecionadas para formar um subconjunto estatisticamente relevante, garantindo que a análise preditiva fosse baseada em compostos físico-quimicamente similares (FIGURA 20).

Figura 20 – Espaço químico composto pelas moléculas mais próximas do omeprazol.

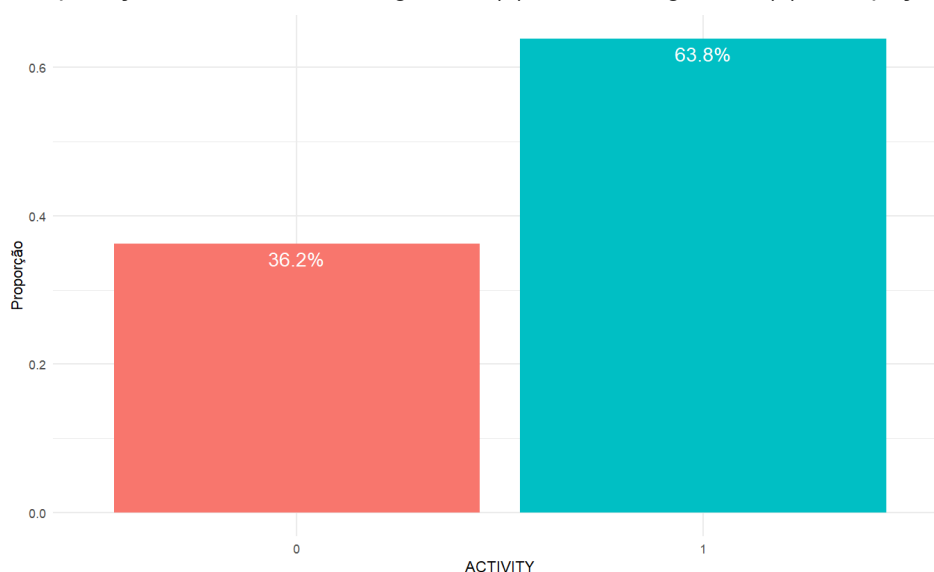


Fonte: Do autor.

Nota: Atividade (ACTIVITY): 0 – não mutagênico; 1 – mutagênico. O losango azul (♦) representa a localização da molécula-alvo (omeprazol) no espaço químico.

Quanto à distribuição de resultados positivos e negativos para o desfecho toxicológico estudado, relativo ao exemplo utilizado (omeprazol), foi constatado o desbalanceamento das classes (FIGURA 21), com 63,8% (319) de resultados positivos e 36,2% (181) negativos. Dessa forma, assim como nas estratégias anteriores, utilizou-se a estatística de Youden para compensar a diferença de proporção entre as classes e otimizar o ponto de corte da predição.

Figura 21 – Proportação de moléculas mutagênicas (1) e não mutagênicas (0) do espaço químico.



Fonte: Do autor.

Para o treinamento, os modelos de regressão foram configurados da mesma forma que os demais discutidos nesse trabalho. O Random Forest, como apresentado na Tabela 10, foi avaliado com diferentes valores de *mtry* e selecionou o valor 19 como o melhor parâmetro, baseado na maior AUC-ROC (0,8436). Embora o modelo tenha apresentado um bom desempenho geral, a sensibilidade ainda foi relativamente baixa (58,5%), o que pode levar à subdetecção de casos positivos. Melhorias futuras podem ser realizadas no ajuste dos hiperparâmetros ou balanceamento das classes. Como demonstrado adiante, trabalhou-se com a compensação do desbalanceamento das classes através do uso de um ponto de corte ótimo.

Tabela 10 – Resultados do modelo Random Forest.

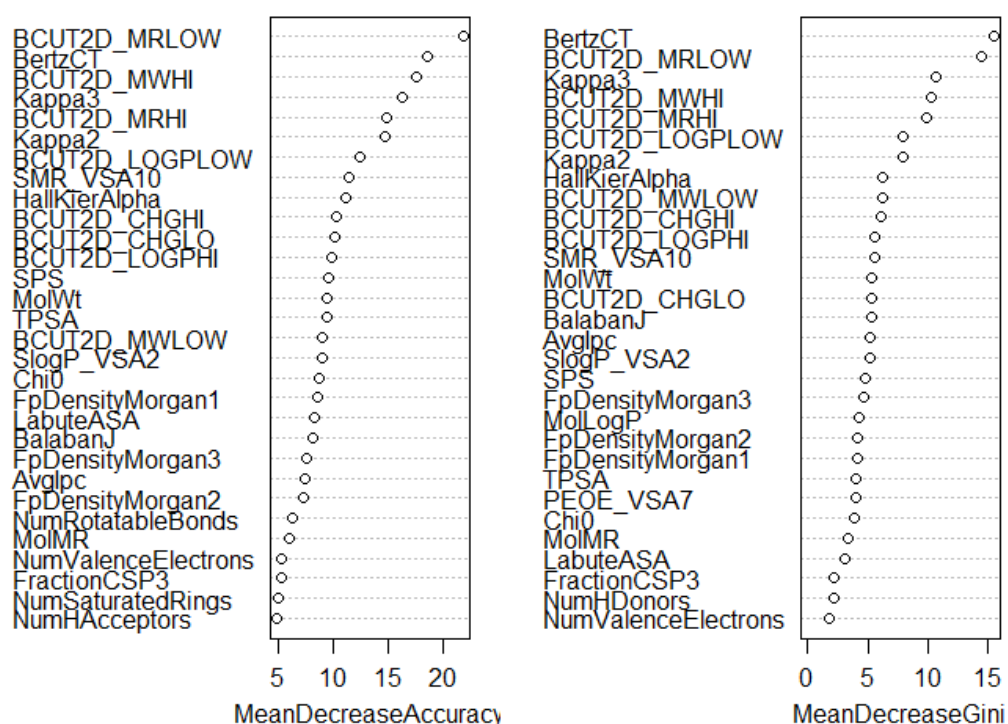
| <i>mtry</i> | AUC ROC | Sensibilidade | Especificidade |
|-------------|---------|---------------|----------------|
| 2           | 0,8367  | 0,5305        | 0,8980         |
| 19          | 0,8436  | 0,5848        | 0,8863         |
| 36          | 0,8422  | 0,5986        | 0,8709         |

Fonte: Do autor.

Na Figura 22, são demonstradas as variáveis que apresentaram maior relevância na predição da mutagenicidade das moléculas. A variável BCUT2D\_MRLOW se manteve como a mais influente, enquanto as variáveis BertzCT, BCUT2D\_MWHI e

Kappa3 também figuraram entre as primeiras posições, ratificando que o uso de um ambiente químico mais próximo da molécula-alvo influencia na dinâmica de seleção das variáveis do modelo. Por conseguinte, as características topológicas e aquelas ligadas à complexidade das moléculas continuaram sendo as mais importantes para a regressão.

Figura 22 - Importância das variáveis no Random Forest.



Fonte: Do autor.

O modelo CART foi treinado com 401 amostras e 36 preditores para classificar entre duas classes: “Negative” e “Positive”. Diferentes valores do parâmetro  $cp$  foram testados, influenciando o desempenho da árvore. O melhor resultado foi obtido com  $cp = 0,0483$ , alcançando um ROC de 0,664, com sensibilidade de 43,0% e especificidade de 85,5% (TABELA 11). Esse modelo equilibrou razoavelmente bem a detecção de positivos e negativos, em comparação com os outros  $cp$  testados, embora a sensibilidade relativamente baixa indique que muitos casos positivos podem estar sendo erroneamente classificados como negativos.

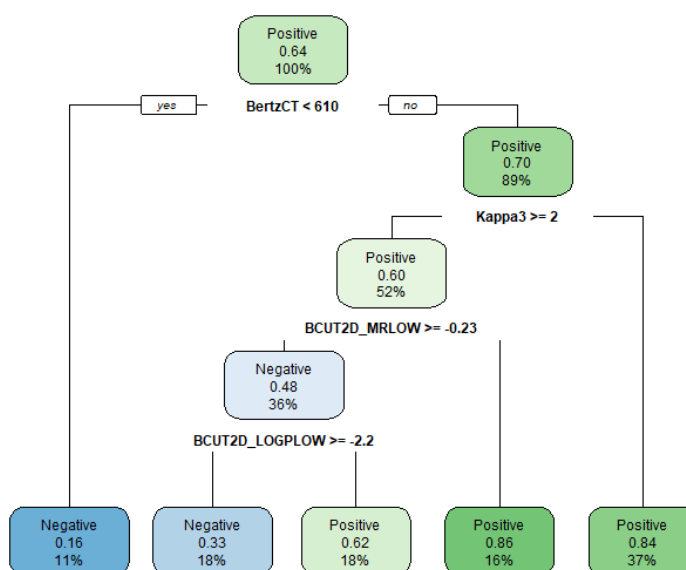
Tabela 11 – Resultados do modelo CART.

| cp     | ROC    | Sensibilidade | Especificidade |
|--------|--------|---------------|----------------|
| 0,0483 | 0,6639 | 0,4295        | 0,8546         |
| 0,0575 | 0,6415 | 0,3867        | 0,8465         |
| 0,2069 | 0,5345 | 0,0886        | 0,9805         |

Fonte: Do autor.

A saída gráfica utilizada para as tomadas de decisão da classificação executada pelo modelo CART pode ser observada na Figura 23. Foi possível identificar que algumas variáveis relevantes coincidiram com as apresentadas na regressão por árvores aleatórias, como BertzCT e Kappa3. Com a redução do espaço químico, também foi verificada uma maior profundidade da árvore criada.

Figura 23 - Visualização gráfica das árvores de decisão.



Fonte: Do autor.

Por fim, avaliando os parâmetros de ajuste do modelo treinado com o algoritmo XGBoost, notou-se uma leve piora do desempenho em relação à abordagem descrita no item 5.2.2., exibindo uma AUC de 0,815, sensibilidade de 0,615 e especificidade de 0,832. Assim como já discutido, a sensibilidade moderada pode ser crítica por aumentar os falsos negativos.

Os dados exibidos nas Tabelas 12 e 13 comparam o desempenho dos modelos Random Forest, CART e XGBoost na predição da mutagenicidade, considerando distintos critérios de ponto de corte. Enquanto a primeira tabela adota o valor padrão de 0,5, a segunda emprega o ponto de corte ajustado conforme o critério de Youden, estabelecendo 0,6550 para o Random Forest, 0,8435 para o CART e 0,6658 para o XGBoost.

Tabela 12 – Métricas de avaliação dos modelos de regressão (ponto de corte = 0,5).

| Modelo  | ROC    | Acurácia | Sens.  | Esp.   | F1-score | P (+)  | P (-)  | Atividade* |
|---------|--------|----------|--------|--------|----------|--------|--------|------------|
| RF      | 0,8649 | 0,8081   | 0,6944 | 0,8730 | 0,7246   | 0,3820 | 0,6180 | Neg.       |
| CART    | 0,6693 | 0,7172   | 0,5556 | 0,8095 | 0,5882   | 0,1591 | 0,8409 | Neg.       |
| XGBoost | 0,8580 | 0,8081   | 0,6944 | 0,8730 | 0,7246   | 0,4330 | 0,5670 | Neg.       |

Fonte: Do autor.

Nota: Sens.: Sensibilidade; Esp.: Especificidade; P (+): probabilidade associada à predição positiva; P (-): probabilidade associada à predição negativa; Neg.: Negativo; Pos.: Positivo. \*Atividade mutagênica.

Tabela 13 – Métricas de avaliação dos modelos de regressão com ponto de corte ótimo.

| Modelo  | ROC    | Acurácia | Sens.  | Esp.   | F1-score | P (+)  | P (-)  | Atividade* |
|---------|--------|----------|--------|--------|----------|--------|--------|------------|
| RF      | 0,8649 | 0,8182   | 0,8889 | 0,7778 | 0,7805   | 0,3820 | 0,6180 | Neg.       |
| CART    | 0,6693 | 0,6465   | 0,6667 | 0,6349 | 0,5783   | 0,1591 | 0,8409 | Neg.       |
| XGBoost | 0,8580 | 0,8182   | 0,7500 | 0,8571 | 0,7500   | 0,4330 | 0,5670 | Neg.       |

Fonte: Do autor.

Nota: Sens.: Sensibilidade; Esp.: Especificidade; P (+): probabilidade associada à predição positiva; P (-): probabilidade associada à predição negativa; Neg.: Negativo; Pos.: Positivo. \*Atividade mutagênica.

Os modelos Random Forest e XGBoost apresentaram alto desempenho, com AUC de 0,865 e 0,858, respectivamente. Entretanto, ao aplicar o ponto de corte ótimo, percebeu-se um leve impacto na performance global. O Random Forest, por exemplo, teve um aumento expressivo na sensibilidade (69,4% para 88,9%), indicando uma melhor detecção de casos positivos. Esse ajuste, entretanto, veio acompanhado de uma leve redução na especificidade (87,3% para 77,8%), refletindo um pequeno aumento nos falsos positivos. Já o XGBoost, ao adotar um ponto de corte otimizado de 0,6658, manteve um equilíbrio entre sensibilidade (75%) e especificidade (85,7%), garantindo uma boa discriminação entre as classes.

O CART, por sua vez, assim como nas abordagens demonstradas nos itens 5.2.1 e 5.2.2, teve um desempenho inferior, sendo menos adequado para cenários onde a precisão da predição é crucial.

Outro efeito importante do ajuste do ponto de corte foi observado no F1-score, que equilibra precisão e sensibilidade. O Random Forest demonstrou um desempenho mais consistente, atingindo um F1-score de 0,780, confirmando sua robustez na predição das classes. O XGBoost, por sua vez, apresentou um F1-score de 0,75, o que, aliado à sua alta especificidade, sugere que esse modelo pode ser mais adequado em cenários onde minimizar falsos positivos é uma prioridade. Já o CART, novamente, teve um desempenho inferior, com F1-score de 0,578, reforçando sua menor capacidade de equilibrar a detecção correta de positivos e negativos.

Os *scripts* em R podem ser acessados nos APÊNDICES G e H.

### 5.3 APLICAÇÃO DOS MODELOS DE PREDIÇÃO *IN SILICO*

As moléculas seleccionadas no Quadro 2 foram utilizadas para a aplicação dos modelos preditivos desenvolvidos, a fim de avaliar a capacidade desses algoritmos em prever a atividade mutagênica de diferentes substâncias. A seguir, as estratégias trabalhadas nos itens 5.2.1, 5.2.2 e 5.2.3, designadas nas tabelas 14, 15, 16 e 17, como grupos 1, 2 e 3, respectivamente, foram avaliadas segundo os resultados de predição e desempenhos dos modelos, considerando as principais métricas exploradas nesta monografia.

Para o ambroxol (TABELA 14), os resultados dos modelos RF e XGBoost foram comparáveis, sem diferenças significativas de desempenho. O modelo Random Forest do grupo 1 obteve os melhores indicadores, com uma área sob a curva ROC de 0,8715, F1-score de 0,7788 e acurácia de 0,7997, seguido de perto pelo XGBoost do mesmo grupo (ROC = 0,8518, acurácia = 0,7773 e F1-score = 0,7624). O pior desempenho foi observado no modelo CART do grupo 2, que teve ROC de 0,4162 e acurácia de 0,5678, indicando baixa capacidade preditiva. Os modelos do grupo 1 classificaram a substância como negativa, em consonância com a literatura, enquanto houve divergência nos grupos 2 e 3. Sobre o RF do grupo 3, pode-se reconhecer que também se

trata de uma alternativa viável em termos de facilidade de execução, já que exibiu métricas balanceadas e acertou a predição.

Tabela 14 – Resultados de predição para o ambroxol (ponto de corte ótimo).

| Grupo - Modelo | ROC    | Acurácia | Sens.  | Esp.   | F1-score | Cluster | Atividade* |
|----------------|--------|----------|--------|--------|----------|---------|------------|
| 1 - RF         | 0,8715 | 0,7997   | 0,7664 | 0,8281 | 0,7788   | --      | Neg.       |
| 1 - CART       | 0,6583 | 0,6450   | 0,5714 | 0,7077 | 0,5970   | --      | Neg.       |
| 1 - XGBoost    | 0,8518 | 0,7773   | 0,7765 | 0,7779 | 0,7624   | --      | Neg.       |
| 2 - RF         | 0,8130 | 0,7203   | 0,6269 | 0,8431 | 0,7179   | 5       | Pos.       |
| 2 - CART       | 0,4162 | 0,5678   | 0,4925 | 0,6667 | 0,5641   | 5       | Pos.       |
| 2 - XGBoost    | 0,7902 | 0,7373   | 0,6716 | 0,8235 | 0,7438   | 5       | Pos.       |
| 3 - RF         | 0,7688 | 0,7400   | 0,7692 | 0,7083 | 0,7547   | --      | Neg.       |
| 3 - CART       | 0,6663 | 0,6200   | 0,5962 | 0,6458 | 0,6200   | --      | Pos.       |
| 3 - XGBoost    | 0,7308 | 0,7200   | 0,7308 | 0,7083 | 0,7308   | --      | Pos.       |

Fonte: Do autor.

Nota: Sens.: Sensibilidade; Esp.: Especificidade; P (+): probabilidade associada à predição positiva; P (-): probabilidade associada à predição negativa; Neg.: Negativo; Pos.: Positivo. \*Atividade mutagênica predita.

No caso da captana (TABELA 15), o RF do grupo 1 também obteve o melhor resultado (ROC = 0,87152, acurácia = 0,79969), seguido pelo XGBoost do grupo 3 (ROC = 0,8430, acurácia = 0,8081). O pior desempenho foi do modelo CART do grupo 3, que apresentou ROC de 0,3661 e acurácia de apenas 0,4444. Os modelos RF e XGBoost classificaram a substância de maneira variada entre os grupos, sendo que o RF dos grupos 1, 2 e 3 acertaram quanto à mutagenicidade do composto.

Tabela 15 – Resultados de predição para o captana (ponto de corte ótimo).

| Modelo      | ROC    | Acurácia | Sens.  | Esp.   | F1-score | Cluster | Atividade* |
|-------------|--------|----------|--------|--------|----------|---------|------------|
| 1 - RF      | 0,8715 | 0,7997   | 0,7664 | 0,8281 | 0,7788   | --      | Pos.       |
| 1 - CART    | 0,6583 | 0,6450   | 0,5714 | 0,7077 | 0,5970   | --      | Neg.       |
| 1 - XGBoost | 0,8518 | 0,7773   | 0,7765 | 0,7779 | 0,7624   | --      | Neg.       |
| 2 - RF      | 0,8130 | 0,7203   | 0,6269 | 0,8431 | 0,7180   | 5       | Pos.       |
| 2 - CART    | 0,4162 | 0,5678   | 0,4925 | 0,6667 | 0,5641   | 5       | Pos.       |
| 2 - XGBoost | 0,7902 | 0,7373   | 0,6716 | 0,8235 | 0,7438   | 5       | Pos.       |
| 3 - RF      | 0,8413 | 0,7677   | 0,6727 | 0,8864 | 0,7629   | --      | Pos.       |
| 3 - CART    | 0,3661 | 0,4444   | 0,0000 | 1,0000 | --       | --      | Pos.       |
| 3 - XGBoost | 0,8430 | 0,8081   | 0,8545 | 0,7500 | 0,8319   | --      | Neg.       |

Fonte: Do autor.

Nota: Sens.: Sensibilidade; Esp.: Especificidade; P (+): probabilidade associada à predição positiva; P (-): probabilidade associada à predição negativa; Neg.: Negativo; Pos.: Positivo. \*Atividade mutagênica predita.

A TRP-P-1 (TABELA 16), reconhecidamente mutagênica, exibiu o pior resultado frente aos modelos. A exemplo das outras moléculas, o RF do grupo 1 acertou a predição, mantendo métricas desejáveis e balanceadas. Entretanto, houve divergência entre os grupos, sendo que o grupo 1 majoritariamente classificou a substância como positiva, enquanto os demais apresentaram variações. Vale destacar o resultado de F1-score apresentado pelo RF do grupo 3 (0,6563), haja vista ter sido o mais baixo entre todas as predições que utilizaram árvores aleatórias. Os motivos podem remontar no espaço químico que a molécula se encontrou, o qual pode não ter captado as propriedades da molécula.

Tabela 16 – Resultados de predição para o TRP-P-1 (ponto de corte ótimo).

| Modelo      | ROC    | Acurácia | Sens.  | Esp.   | F1-score | Cluster | Atividade* |
|-------------|--------|----------|--------|--------|----------|---------|------------|
| 1 - RF      | 0,8715 | 0,7997   | 0,7664 | 0,8281 | 0,7788   | --      | Pos.       |
| 1 - CART    | 0,6583 | 0,6450   | 0,5714 | 0,7077 | 0,5970   | --      | Pos.       |
| 1 - XGBoost | 0,8518 | 0,7773   | 0,7765 | 0,7779 | 0,7624   | --      | Pos.       |
| 2 - RF      | 0,8598 | 0,8112   | 0,7211 | 0,8557 | 0,7162   | 3       | Neg.       |
| 2 - CART    | 0,6987 | 0,6742   | 0,6395 | 0,6913 | 0,5646   | 3       | Pos.       |
| 2 - XGBoost | 0,8531 | 0,8067   | 0,7687 | 0,8255 | 0,7244   | 3       | Neg.       |
| 3 - RF      | 0,8400 | 0,7778   | 0,7500 | 0,7887 | 0,6563   | --      | Neg.       |
| 3 - CART    | 0,4600 | 0,6970   | 0,1786 | 0,9014 | 0,2500   | --      | Neg.       |
| 3 - XGBoost | 0,8431 | 0,8283   | 0,7500 | 0,8592 | 0,7119   | --      | Neg.       |

Fonte: Do autor.

Nota: Sens.: Sensibilidade; Esp.: Especificidade; P (+): probabilidade associada à predição positiva; P (-): probabilidade associada à predição negativa; Neg.: Negativo; Pos.: Positivo. \*Atividade mutagênica predita.

Para a 3-Amino-3-azabicyclo[3.3.0]octano (TABELA 17), os melhores desempenhos foram obtidos pelo RF do grupo 1 (ROC = 0,8715, acurácia = 0,7997) e pelo XGBoost do mesmo grupo (ROC = 0,8518, acurácia = 0,7773). O pior resultado foi observado no CART do grupo 2, com ROC de 0,4162 e acurácia de 0,5678. De forma geral, os modelos RF e XGBoost indicaram classificação negativa para essa substância, com algumas discrepâncias nos modelos do grupo 2. As evidências presentes na literatura sobre o potencial mutagênico desse composto não são conclusivas.

Tabela 17 – Resultados de predição para o 3-Amino-3-azabi-cyclo[3.3.0]octane (ponto de corte ótimo).

| Modelo      | ROC    | Acurácia | Sens.  | Esp.   | F1-score | Cluster | Atividade* |
|-------------|--------|----------|--------|--------|----------|---------|------------|
| 1 - RF      | 0,8715 | 0,7997   | 0,7664 | 0,8281 | 0,7788   | --      | Neg.       |
| 1 - CART    | 0,6583 | 0,6450   | 0,5714 | 0,7077 | 0,5970   | --      | Neg.       |
| 1 - XGBoost | 0,8518 | 0,7773   | 0,7765 | 0,7779 | 0,7624   | --      | Neg.       |
| 2 - RF      | 0,8130 | 0,7203   | 0,6269 | 0,8431 | 0,7179   | 5       | Pos.       |
| 2 - CART    | 0,4162 | 0,5678   | 0,4925 | 0,6667 | 0,5641   | 5       | Pos.       |
| 2 - XGBoost | 0,7902 | 0,7373   | 0,6716 | 0,8235 | 0,7438   | 5       | Neg.       |
| 3 - RF      | 0,7688 | 0,7172   | 0,8095 | 0,6491 | 0,7083   | --      | Neg.       |
| 3 - CART    | 0,3853 | 0,6465   | 0,4762 | 0,7719 | 0,5333   | --      | Neg.       |
| 3 - XGBoost | 0,7966 | 0,7071   | 0,8810 | 0,5789 | 0,7184   | --      | Neg.       |

Fonte: Do autor.

Nota: Sens.: Sensibilidade; Esp.: Especificidade; P (+): probabilidade associada à predição positiva; P (-): probabilidade associada à predição negativa; Neg.: Negativo; Pos.: Positivo. \*Atividade mutagênica predita.

Em termos gerais, a aplicação dos modelos evidenciou que o RF e XGBoost tiveram o melhor desempenho na maioria dos casos, enquanto o CART frequentemente obteve resultados inferiores. Comparando com avaliações de modelos desenvolvidos por empresas e grupos de pesquisa relatados na literatura, as métricas obtidas estão dentro do esperado, considerando o número reduzido de moléculas testadas (FURUHAMA *et al.*, 2023). Salienta-se que devido à complexidade da relação estrutura-atividade para o desfecho estudado (mutagenicidade), uma seleção mais criteriosa das variáveis pode aprimorar a capacidade preditiva dos modelos. Em suma, o modelo RF, utilizando todos os dados, acertou todas as predições, enquanto o XGBoost, que errou apenas uma, surge como uma boa alternativa por ser mais rápido. Assim, os resultados reforçam a robustez dos modelos RF e XGBoost para essa aplicação.

## 6 CONSIDERAÇÕES FINAIS

A escolha do modelo ideal deve levar em conta a necessidade de evitar falsos negativos, sendo a sensibilidade a métrica mais relevante para essa aplicação. Além disso, métricas como ROC, acurácia e F1-score devem ser analisadas para garantir um bom equilíbrio preditivo. Na Abordagem 1 (item 5.2.1), que considera todos os dados do banco de dados, o XGBoost apresentou um desempenho bem equilibrado, com ROC de 0,8518, sensibilidade de 0,7765 e F1-score de 0,7624, sendo uma opção robusta e com menor custo computacional quando comparado ao Random Forest (RF). Já o CART, apesar de ter a menor acurácia (0,6450) e menor ROC (0,6583), foi o único modelo a classificar a mutagenicidade como positiva.

Na Abordagem 2 (item 5.2.2), que utiliza o espaço químico determinado pelo cluster mais próximo, houve um leve aumento na acurácia do RF (0,8112) e do XGBoost (0,8067) em relação à Abordagem 1. No entanto, a sensibilidade do RF caiu para 0,7211, o que pode resultar em relativo aumento no número de falsos negativos. O CART demonstrou uma melhora na sensibilidade (0,6395) quando comparado à primeira abordagem, porém, seu F1-score de 0,5646 indica que ele ainda apresenta um desempenho inferior aos demais modelos.

A Abordagem 3 (item 5.2.3), que define o espaço químico com base nas moléculas mais próximas por distância euclidiana, apresentou os melhores resultados gerais. Esse desempenho superior pode estar relacionado à maior capacidade do modelo de captar as características físico-químicas das moléculas na criação do espaço químico, resultando em uma representação mais fiel da relação entre estrutura molecular e mutagenicidade. O RF obteve a maior sensibilidade (0,8889), ROC (0,8649) e F1-score (0,7805), tornando-se a melhor opção para evitar falsos negativos e garantir uma boa capacidade preditiva. O XGBoost, por sua vez, também apresentou bons resultados, com ROC de 0,8580 e sensibilidade de 0,7500, mantendo um equilíbrio satisfatório entre as métricas. O CART, embora tenha aumentado sua sensibilidade para 0,6667, ainda apresentou os piores valores de ROC (0,6693) e F1-score (0,5783).

Embora os modelos Random Forest e XGBoost tenham demonstrado desempenho satisfatório em todas as abordagens avaliadas, consolidando-se como alternativas viáveis, os modelos selecionados como mais adequados para esta aplicação foram o Random Forest da terceira abordagem (item 5.2.3) e o XGBoost da primeira abordagem (item 5.2.1). Essa escolha deve-se à elevada capacidade de ambos os modelos em identificar verdadeiros positivos, com consequente redução na ocorrência de falsos negativos, e ao baixo tempo de processamento observado em *hardware* convencional. Ademais, o omeprazol, utilizado como molécula modelo, é descrito na literatura como não mutagênico, e todos os modelos, com exceção do CART, foram capazes de prever corretamente essa característica. Esse resultado reforça a robustez e confiabilidade das estratégias avaliadas.

Embora o CART apresente vantagens interpretativas e seja de fácil implementação, suas limitações quanto à estabilidade e à propensão ao sobreajuste tornam-no menos competitivo frente a métodos mais parcimoniosos e estatisticamente robustos, como o Naive Bayes e a Regressão Logística com penalização L1 (Lasso), que podem ser usados no lugar do CART para referenciais iniciais.

O desafio de moléculas-alvo de mutagenicidade conhecida reforçam a superioridade dos modelos RF e XGBoost, independentemente da abordagem, e a necessidade de avaliar atentamente se as moléculas estudadas se enquadram no domínio de aplicação e espaço químico dos modelos treinados.

Como perspectiva futura, sugere-se a implementação de um roteiro híbrido, integrando o XGBoost da primeira abordagem e o Random Forest baseado na terceira abordagem. Essa estratégia poderia equilibrar tempo computacional e desempenho preditivo, otimizando a acurácia e a generalização do modelo. Além disso, a incorporação de um terceiro método complementar, como o CatBoost, poderia fortalecer as bases de evidências, aumentando a segurança e confiabilidade das previsões.

## REFERÊNCIAS

ALVES, V. M. et al. Cheminformatics: An introduction. *Química Nova*, São Paulo: Sociedade Brasileira de Química, v. 41, n. 2, p. 213–222, 2018.

ASSUNÇÃO, G. O.; IZBICKI, R.; PRATES, M. O. Is augmentation effective in improving prediction in imbalanced datasets? *Journal of Data Science*, v. 22, p. 1–16, 2024.

BENIGNI, R.; BOSSA, C. Mechanisms of chemical carcinogenicity and mutagenicity: a review with implications for predictive toxicology. *Chemical Reviews*, v. 111, n. 4, p. 2507–2536, 2011.

BREIMAN, L. Random Forests. *Machine Learning*, v. 45, n. 1, p. 5–32, 2001.

BURGER, S. V. Introduction to machine learning with R: rigorous mathematical analysis. Sebastopol: O'Reilly Media, 2018.

CHEN, T.; GUESTRIN, C. XGBoost: a scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, p. 785–794, 2016.

CRONIN, M. T. D. et al. A scheme to evaluate structural alerts to predict toxicity – assessing confidence by characterising uncertainties. *Regulatory Toxicology and Pharmacology*, v. 135, p. 105259, 2022.

CRONIN, M. T. D. et al. The predictivity of QSARs for toxicity: recommendations for improving model performance. *Computational Toxicology*, v. 33, p. 100583, 2025.

DE AMORIM, R. C.; HENNIG, C. Recovering the number of clusters in data sets with noise features using feature rescaling factors. *Information Sciences*, v. 324, p. 126–145, 2015.

EPA. CompTox Chemicals Dashboard. Washington, D.C.: United States Environmental Protection Agency, 2025.

FURUHAMA, A. et al. Evaluation of QSAR models for predicting mutagenicity: outcome of the Second Ames/QSAR international challenge project. SAR and QSAR in Environmental Research, v. 34, n. 12, p. 983–1001, 2023.

HALL, L. H.; KIERT, L. B. The Molecular Connectivity Chi Indexes and Kappa Shape Indexes in Structure-Property Modeling. Journal of Chemical Information and Computer Sciences, v. 31, n. 1, p. 76–82, 1991.

HANLEY, J. A.; MCNEIL, B. J. The meaning and use of the area under a receiver operating characteristic (ROC) curve. Radiology, v. 143, n. 1, p. 29–36, 1982.

HANSEN, K. et al. Benchmark data set for in silico prediction of Ames mutagenicity. Journal of Chemical Information and Modeling, v. 49, n. 9, p. 2077–2081, 2009.

HEMMERICH, J.; ECKER, G. F. In silico toxicology: from structure–activity relationships towards deep learning and adverse outcome pathways. Wiley Interdisciplinary Reviews: Computational Molecular Science, v. 10, n. 4, e1462, 2020.

ICH. ICH Q9: Quality Risk Management. Geneva: International Council for Harmonisation, 2023a.

ICH. M7(R2): Assessment and control of DNA reactive (mutagenic) impurities. Geneva: International Council for Harmonisation, 2023b.

JAMES, G. et al. An introduction to statistical learning: with applications in R. 2. ed. New York: Springer, 2023.

JORDAN, M. I.; KLEINBERG, J.; SCHÖLKOPF, B. Pattern recognition and machine learning. Cambridge: MIT Press, 2006.

KLEINSTREUER, N. C.; TETKO, I. V.; TONG, W. Introduction to special issue: computational toxicology. *Chemical Research in Toxicology*, v. 34, n. 2, p. 239–240, 2021.

LIPINSKI, C. A. et al. Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Advanced Drug Delivery Reviews*, v. 46, n. 1–3, p. 3–26, 2001.

MACARTHUR CLARK, J. The 3Rs in research: a contemporary approach to replacement, reduction and refinement. *British Journal of Nutrition*, v. 120, n. S1, p. S1–S7, 2018.

MARTIN, K.; KRZYWINSKI, N. A. Classification and regression trees. *Nature Methods*, v. 14, p. 917–918, 2017.

MINGOTI, S. A. Análise de dados através de métodos de estatística multivariada: uma abordagem aplicada. 1. ed. Belo Horizonte: Editora UFMG, 2005.

MUCOANGIN. Bula do medicamento Mucoangin, 2010.

NRC – NATIONAL RESEARCH COUNCIL. Risk assessment in the federal government: managing the process. Washington, D.C.: National Academy Press, 1983.

R CORE TEAM. R: a language and environment for statistical computing. Viena: R Foundation for Statistical Computing, 2025.

RDKIT. RDKit: Open-source cheminformatics. 2025.

RODRÍGUEZ-BELENGUER, P. et al. Usage of model combination in computational toxicology. *Toxicology Letters*, v. 389, p. 34–44, 2023.

SARKER, I. H. Machine learning: algorithms, real-world applications and research directions. SN Computer Science, v. 2, n. 3, 2021.

SCHUBERT, E. Stop using the elbow criterion for k-means and how to choose the number of clusters instead. Data Mining Reports, v. 2022, n. 1, p. 1–5, 2022.

SEWELL, F. et al. New approach methodologies (NAMs): identifying and overcoming hurdles to accelerated adoption. Toxicology Research, v. 13, n. 2, p. 132–145, 2024.

SNYDER, R. D. An update on the genotoxicity and carcinogenicity of marketed pharmaceuticals with reference to in silico predictivity. Environmental and Molecular Mutagenesis, v. 50, n. 6, p. 435–450, 2009.

TODESCHINI, R.; CONSONNI, V. Chemometrics in QSAR. Weinheim: Wiley-VCH, 2009.

WALKER, J. M. Computational toxicology. New York: Springer, 2012.

YOU DEN, W. J. Index for rating diagnostic tests. Cancer, v. 3, n. 1, p. 32–35, 1950.

## APÊNDICE A

```
import pandas as pd
from rdkit import Chem
from rdkit.Chem import Descriptors

def get_all_descriptors(mol):
    """Calculates all available descriptors in RDKit."""
    descriptor_funcs = {desc_name: func for desc_name, func in Descriptors.descList}

    descriptors = {}
    for name, func in descriptor_funcs.items():
        try:
            descriptors[name] = func(mol)
        except Exception as e:
            descriptors[name] = None # Set value to None in case of error

    # Adding specific descriptors manually
    descriptors['Hy'] = Descriptors.NumHDonors(mol) - Descriptors.NumHAcceptors(mol) # Hydro-
    philic factor (simplified)
    descriptors['XLOGP2'] = Descriptors.MolLogP(mol) ** 2 # Wang octanol-water partition coeffi-
    cient squared

    return descriptors

def check_lipinski_rule(mol):
    """Checks Lipinski's rule violations."""
    mw = Descriptors.MolWt(mol)
    logp = Descriptors.MolLogP(mol)
    h_donors = Descriptors.NumHDonors(mol)
    h_acceptors = Descriptors.NumHAcceptors(mol)

    violations = 0
    if mw > 500:
        violations += 1
    if logp > 5:
        violations += 1
    if h_donors > 5:
        violations += 1
    if h_acceptors > 10:
        violations += 1

    return violations

def get_molecule_info(smiles, mol_id):
    mol = Chem.MolFromSmiles(smiles)

    if not mol:
        return None

    # Collect all descriptors
    descriptors = get_all_descriptors(mol)
```

```

# Add Lipinski's rule violations
descriptors['LipinskiViolations'] = check_lipinski_rule(mol)

data = {
    'id': mol_id,
    'SMILES': smiles,
    **descriptors
}

return data

def process_molecules_from_excel(input_filename, output_filename):
    # Read the input spreadsheet
    df_input = pd.read_excel(input_filename)
    smiles_list = df_input['SMILES']

    all_molecules_data = []

    # Process each SMILES
    for idx, smiles in enumerate(smiles_list, 1):
        mol_id = f"mol_{idx:03}"
        molecule_data = get_molecule_info(smiles, mol_id)
        if molecule_data:
            all_molecules_data.append(molecule_data)

    # Convert the data to a DataFrame
    df_output = pd.DataFrame(all_molecules_data)

    # Save the results to an Excel file
    df_output.to_excel(output_filename, sheet_name='Molecule Data', index=False)

# Example usage
if __name__ == "__main__":
    input_filename = 'Mutagenicity_N6512_formatada.xlsx' # Replace with your input file
    output_filename = 'mutagenicity_RDkit.xlsx'
    process_molecules_from_excel(input_filename, output_filename)

```

## APÊNDICE B

```
# Carrega os dados
data_path <- "mutagenicity_m.xlsx"
data <- read_excel(data_path)

# Domínio de aplicação: filtra moléculas com MolWt <= 1000 e MolLogP <= 10
data_filtered <- data %>%
  filter(MolWt <= 1000, MolLogP <= 10)

# Transforma a variável ACTIVITY em fator
data_filtered$ACTIVITY <- as.factor(data_filtered$ACTIVITY)

# Seleciona as variáveis relevantes
selected_vars <- c("ACTIVITY", "SPS", "MolWt", "NumValenceElectrons", "FpDensityMorgan1",
  "FpDensityMorgan2", "FpDensityMorgan3", "BCUT2D_MWHI", "BCUT2D_MWLOW",
  "BCUT2D_CHGHI", "BCUT2D_CHGLO", "BCUT2D_LOGPHI", "BCUT2D_LOGPLOW",
  "BCUT2D_MRHI", "BCUT2D_MRLOW", "AvgIpc", "BalabanJ", "BertzCT", "Chi0",
  "HallKierAlpha", "Kappa2", "Kappa3", "LabuteASA", "PEOE_VSA7",
  "SMR_VSA10", "SlogP_VSA2", "TPSA", "FractionCSP3", "NumAliphaticRings",
  "NumAromaticRings", "NumHAcceptors", "NumHDonors", "NumRotatableBonds",
  "NumSaturatedRings", "RingCount", "MolLogP", "MolMR")

data_filtered <- data_filtered %>%
  dplyr::select(all_of(selected_vars))

# Remove valores ausentes
data_filtered <- na.omit(data_filtered)

# Análise descritiva
str(data_filtered)
ncol(data_filtered)
summary(data_filtered)

# Filtrar apenas colunas numéricas
data_numeric <- data_filtered[sapply(data_filtered, is.numeric)]

# Criar a tabela de resumo com valores separados
summary_df <- do.call(rbind, lapply(data_numeric, function(x) {
  s <- summary(x)
  if(length(s) == 6) s <- c(s, "NA's" = 0)
  as.numeric(s)
}))
colnames(summary_df) <- c("Min", "1st Qu.", "Median", "Mean", "3rd Qu.", "Max", "NA's")
summary_df <- data.frame(Variable = rownames(summary_df), summary_df, row.names = NULL)

# Exportar para Excel com formatação básica
wb <- createWorkbook()
addWorksheet(wb, "Resumo")
writeData(wb, "Resumo", summary_df, headerStyle = createStyle(textDecoration = "bold"))
setColWidths(wb, "Resumo", cols = 1:ncol(summary_df), widths = "auto")
saveWorkbook(wb, "summary_descritivo.xlsx", overwrite = TRUE)

# Criar histograma para cada variável numérica
```

```

numeric_vars <- names(data_filtered)[supply(data_filtered, is.numeric)] # Seleciona apenas variáveis numéricas

for (var in numeric_vars) {
  hist(data_filtered[[var]],
    main = paste("Histograma de", var),
    col = "purple", border = "white",
    xlab = var, ylab = "Frequência")
}

# Criar gráficos de violino sobrepostos ao boxplot
for (variavel in numeric_vars) {
  p <- ggplot(data_filtered, aes(x = factor(1), y = .data[[variavel]])) +
    geom_violin(fill = "purple", color = "white", alpha = 0.5) + # Gráfico de violino semitransparente
    geom_boxplot(width = 0.2, fill = "white", color = "lightblue", outlier.color = "blue", alpha = 0.8) + #
  Boxplot sobreposto
  labs(title = paste("Variável", variavel), x = "", y = variavel) +
  theme_minimal() +
  theme(axis.text.x = element_blank(), axis.ticks.x = element_blank())
  print(p)
}

# Calcular e visualizar a matriz de correlação
cor_matrix <- cor(data_numeric, use = "pairwise.complete.obs")

# Exibir matriz de correlação
print(cor_matrix)

# Visualizar correlação com corrplot
corrplot(cor_matrix, method = "color", type = "upper", tl.col = "black", tl.cex = 0.7)

library(reshape2)
# Criar heatmap com ggplot2
cor_melt <- melt(cor_matrix)

ggplot(cor_melt, aes(Var1, Var2, fill = value)) +
  geom_tile(color = "white") +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white", midpoint = 0, limit = c(-1, 1)) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(title = "Heatmap das Correlações", x = "Variáveis", y = "Variáveis")

# Criar uma tabela de correlações
colnames(cor_melt) <- c("Variável 1", "Variável 2", "Correlação")

# Exibir e visualizar a tabela de correlações
print(cor_melt)
View(cor_melt)

# Criar um contador para cada variável (correlações acima de 0.7 em módulo)
contador_correlacoes <- rowSums(abs(cor_matrix) > 0.7) - 1 # -1 para remover autovalores (diagonal)

# Criar tabela de contagens
tabela_correlacoes <- data.frame(Variável = names(contador_correlacoes), Contador = contador_correlacoes)

# Exibir e salvar a tabela
print(tabela_correlacoes)
View(tabela_correlacoes)

```

## APÊNDICE C

```

#=====
# Descrição:
# Domínio de aplicação: filtra moléculas com MolWt <= 1000 e MolLogP <= 10
# Verifica balanceamento
# Modelos com todos os dados, sem definição de espaço químico
# Predição com ponto de corte padrão (0,5)
#=====

source("Pacotes.R") # Evoca o script de carregamento de pacotes

cat("\014") # Limpa o console
rm(list = ls(all = TRUE)) # Limpa a memória
options(scipen = 999) # Desabilita notação científica

# Carrega os dados
data_path <- "mutagenicity_m.xlsx"
target_path <- "omeprazol_descritores.xlsx"

data <- read_excel(data_path)
target <- read_excel(target_path)

# Domínio de aplicação: filtra moléculas com MolWt <= 1000 e MolLogP <= 10
data_filtered <- data %>%
  filter(MolWt <= 1000, MolLogP <= 10)

# Transforma a variável ACTIVITY em fator
data_filtered$ACTIVITY <- as.factor(data_filtered$ACTIVITY)

# Seleciona as variáveis relevantes
selected_vars <- c("ACTIVITY", "SPS", "MolWt", "NumValenceElectrons", "FpDensityMorgan1",
  "FpDensityMorgan2", "FpDensityMorgan3", "BCUT2D_MWHI", "BCUT2D_MWLOW",
  "BCUT2D_CHGHI", "BCUT2D_CHGLO", "BCUT2D_LOGPHI", "BCUT2D_LOGPLOW",
  "BCUT2D_MRHI", "BCUT2D_MRLOW", "AvgIpc", "BalabanJ", "BertzCT", "Chi0",
  "HallKierAlpha", "Kappa2", "Kappa3", "LabuteASA", "PEOE_VSA7",
  "SMR_VSA10", "SlogP_VSA2", "TPSA", "FractionCSP3", "NumAliphaticRings",
  "NumAromaticRings", "NumHAcceptors", "NumHDonors", "NumRotatableBonds",
  "NumSaturatedRings", "RingCount", "MolLogP", "MolMR")

data_filtered <- data_filtered %>%
  dplyr::select(all_of(selected_vars))

# Remove valores ausentes
data_filtered <- na.omit(data_filtered)

# Seleciona variáveis preditoras
predictors <- setdiff(selected_vars, "ACTIVITY")

# Normaliza descritores moleculares
data_predictors <- scale(data_filtered[, predictors])

#=====
# Verifica o balanceamento dos dados
table(data_filtered$ACTIVITY)
prop.table(table(data_filtered$ACTIVITY)) # Proporção de cada classe

```

```

# Cria a tabela de proporções
prop_data <- as.data.frame(prop.table(table(data_filtered$ACTIVITY)))

# Renomeia as colunas
colnames(prop_data) <- c("ACTIVITY", "Proporção")

# Cria gráfico de barras
ggplot(prop_data, aes(x = ACTIVITY, y = Proporção, fill = ACTIVITY)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = scales::percent(Proporção, accuracy = 0.1)),
            vjust = 1.5, color = "white", size = 5) + # Ajuste do texto dentro das barras
  labs(title = "Proporção dos resultados de mutagenicidade",
        x = "ACTIVITY",
        y = "Proporção") +
  theme_minimal() +
  theme(legend.position = "none") # Remove legenda externa

#=====
#=====
# Modelo de predição de mutagenicidade

# Converte ACTIVITY de 0/1 para "Negative"/"Positive": foi necessário para calc. curva ROC
data_filtered$ACTIVITY <- factor(data_filtered$ACTIVITY,
                                levels = c(0, 1),
                                labels = c("Negative", "Positive"))

# Divide os dados em treino e teste
set.seed(123)
trainIndex <- createDataPartition(data_filtered$ACTIVITY, p = 0.8, list = FALSE)
trainData <- data_filtered[trainIndex, ]
testData <- data_filtered[-trainIndex, ]

# Validação cruzada com Random Forest
train_control <- trainControl(method = "cv", number = 10, classProbs = TRUE, summaryFunction =
twoClassSummary)

# Molécula-alvo
target_predictors <- target[, predictors]

#=====
# Modelo Random Forest
#=====
rf_model <- train(ACTIVITY ~ .,
                  data = trainData[, c("ACTIVITY", predictors)],
                  method = "rf",
                  trControl = train_control,
                  importance = TRUE,
                  metric = "ROC")

# Resumo do modelo
print(rf_model)

# Avaliação do modelo no conjunto de teste
rf_predictions <- predict(rf_model, testData)
rf_probabilities <- predict(rf_model, testData, type = "prob")

# Matriz de confusão

```

```

conf_matrix <- confusionMatrix(rf_predictions, testData$ACTIVITY)
print(conf_matrix)

# Plotando a importância das variáveis
varImpPlot(rf_model$finalModel)

# Predição para a molécula alvo
target_prediction <- predict(rf_model, newdata = target_predictors)
print(paste("Predição para a molécula alvo:", target_prediction))

# Probabilidade da predição
target_prediction_prob <- predict(rf_model, newdata = target_predictors, type = "prob")
print("Probabilidade da predição para a molécula alvo:")
print(target_prediction_prob)

# Curva ROC
roc_curve <- roc(testData$ACTIVITY, rf_probabilities[, 2], levels = rev(levels(testData$ACTIVITY)))
print(auc(roc_curve))
plot(roc_curve, main = "Curva ROC - Random Forest")

# Extraíndo as métricas
precision <- conf_matrix$byClass["Pos Pred Value"] # Precisão
sensitivity <- conf_matrix$byClass["Sensitivity"] # Sensibilidade

# Calculando o F1-Score
f1_score_rf <- 2 * (precision * sensitivity) / (precision + sensitivity)
cat("F1-Score:", round(f1_score_rf, 4), "\n")

#=====
# Modelo CAR
#=====
car_model <- train(ACTIVITY ~ .,
  data = trainData[, c("ACTIVITY", predictors)],
  method = "rpart",
  trControl = train_control,
  metric = "ROC")

# Resumo do modelo
print(car_model)

# Avaliação do modelo no conjunto de teste
car_predictions <- predict(car_model, testData)
car_probabilities <- predict(car_model, testData, type = "prob")

# Matriz de confusão
car_conf_matrix <- confusionMatrix(car_predictions, testData$ACTIVITY)
print(car_conf_matrix)

# Plotando a árvore de decisão
rpart.plot(car_model$finalModel)

# Predição para a molécula alvo
car_target_prediction <- predict(car_model, newdata = target_predictors)
print(paste("Predição para a molécula alvo:", car_target_prediction))

# Probabilidade da predição
car_target_prediction_prob <- predict(car_model, newdata = target_predictors, type = "prob")
print("Probabilidade da predição para a molécula alvo:")
print(car_target_prediction_prob)

```

```

# Curva ROC
car_roc_curve <- roc(testData$ACTIVITY, car_probabilities[, 2], levels = rev(levels(testData$ACTIV-
ITY)))
print(auc(car_roc_curve))
plot(car_roc_curve, main = "Curva ROC - CAR")

# Extrair as métricas
precision <- car_conf_matrix$byClass["Pos Pred Value"] # Precisão
sensitivity <- car_conf_matrix$byClass["Sensitivity"] # Sensibilidade

# Calculando o F1-Score
f1_score_car <- 2 * (precision * sensitivity) / (precision + sensitivity)
cat("F1-Score:", round(f1_score_car, 4), "\n")

#=====
# Modelo XGBoost
#=====
xgb_grid <- expand.grid(nrounds = 100,
                      max_depth = 3,
                      eta = 0.3,
                      gamma = 0,
                      colsample_bytree = 1,
                      min_child_weight = 1,
                      subsample = 1)

xgb_model <- train(ACTIVITY ~ .,
                  data = trainData[, c("ACTIVITY", predictors)],
                  method = "xgbTree",
                  trControl = train_control,
                  tuneGrid = xgb_grid,
                  metric = "ROC",
                  verbose = 0)

# Resumo do modelo
print(xgb_model)

# Avaliação do modelo no conjunto de teste
xgb_predictions <- predict(xgb_model, testData)
xgb_probabilities <- predict(xgb_model, testData, type = "prob")

# Matriz de confusão
xgb_conf_matrix <- confusionMatrix(xgb_predictions, testData$ACTIVITY)
print(xgb_conf_matrix)

# Predição para a molécula alvo
xgb_target_prediction <- predict(xgb_model, newdata = target_predictors)
print(paste("Predição para a molécula alvo:", xgb_target_prediction))

# Probabilidade da predição
xgb_target_prediction_prob <- predict(xgb_model, newdata = target_predictors, type = "prob")
print("Probabilidade da predição para a molécula alvo:")
print(xgb_target_prediction_prob)

# Curva ROC
xgb_roc_curve <- roc(testData$ACTIVITY, xgb_probabilities[, 2], levels = rev(levels(testData$ACTIV-
ITY)))
print(auc(xgb_roc_curve))
plot(xgb_roc_curve, main = "Curva ROC - XGBoost")

```

```

# Extraindo as métricas
precision <- xgb_conf_matrix$byClass["Pos Pred Value"] # Precisão
sensitivity <- xgb_conf_matrix$byClass["Sensitivity"] # Sensibilidade

# Calculando o F1-Score
f1_score_xgb <- 2 * (precision * sensitivity) / (precision + sensitivity)
cat("F1-Score:", round(f1_score_xgb, 4), "\n")

#=====
# Resultados

# Consolidando métricas existentes
results <- data.frame(
  Model = c("Random Forest", "CART", "XGBoost"),
  AUC = c(auc(roc_curve), auc(car_roc_curve), auc(xgb_roc_curve)),
  Accuracy = c(conf_matrix$overall["Accuracy"], car_conf_matrix$overall["Accuracy"], xgb_conf_ma-
    trix$overall["Accuracy"]),
  Kappa = c(conf_matrix$overall["Kappa"], car_conf_matrix$overall["Kappa"], xgb_conf_matrix$over-
    all["Kappa"]),
  Sensitivity = c(conf_matrix$byClass["Sensitivity"], car_conf_matrix$byClass["Sensitivity"],
    xgb_conf_matrix$byClass["Sensitivity"]),
  Specificity = c(conf_matrix$byClass["Specificity"], car_conf_matrix$byClass["Specificity"],
    xgb_conf_matrix$byClass["Specificity"]),
  F1_Score = c(f1_score_rf, f1_score_cart, f1_score_xgb),
  Probability_Positive = c(target_prediction_prob[1, "Positive"], car_target_prediction_prob[1, "Posi-
    tive"], xgb_target_prediction_prob[1, "Positive"]),
  Probability_Negative = c(target_prediction_prob[1, "Negative"], car_target_prediction_prob[1, "Neg-
    ative"], xgb_target_prediction_prob[1, "Negative"]),
  Mutagenicity_Prediction = c(as.character(target_prediction), as.character(car_target_prediction),
    as.character(xgb_target_prediction))
)

# Salvando os resultados em um arquivo Excel
write.xlsx(list(Performance = results), "script_controleomeprazol_results.xlsx", rowNames = FALSE)

```

## APÊNDICE D

```

#=====
# Descrição:
# Domínio de aplicação: filtra moléculas com MolWt <= 1000 e MolLogP <= 10
# Verifica balanceamento
# Modelos com todos os dados, sem definição de espaço químico
# Predição com ponto de corte ótimo
#=====

source("Pacotes.R") # Evoca o script de carregamento de pacotes

cat("\014") # Limpa o console
rm(list = ls(all = TRUE)) # Limpa a memória
options(scipen = 999) # Desabilita notação científica

# Carrega os dados
data_path <- "mutagenicity_m.xlsx"
target_path <- "omeprazol_descritores.xlsx"

data <- read_excel(data_path)
target <- read_excel(target_path)

# Domínio de aplicação: filtra moléculas com MolWt <= 1000 e MolLogP <= 10
data_filtered <- data %>%
  filter(MolWt <= 1000, MolLogP <= 10)

# Transforma a variável ACTIVITY em fator
data_filtered$ACTIVITY <- as.factor(data_filtered$ACTIVITY)

# Seleciona as variáveis relevantes
selected_vars <- c("ACTIVITY", "SPS", "MolWt", "NumValenceElectrons", "FpDensityMorgan1",
  "FpDensityMorgan2", "FpDensityMorgan3", "BCUT2D_MWHI", "BCUT2D_MWLOW",
  "BCUT2D_CHGHI", "BCUT2D_CHGLO", "BCUT2D_LOGPHI", "BCUT2D_LOGPLOW",
  "BCUT2D_MRHI", "BCUT2D_MRLOW", "AvgIpc", "BalabanJ", "BertzCT", "Chi0",
  "HallKierAlpha", "Kappa2", "Kappa3", "LabuteASA", "PEOE_VSA7",
  "SMR_VSA10", "SlogP_VSA2", "TPSA", "FractionCSP3", "NumAliphaticRings",
  "NumAromaticRings", "NumHAcceptors", "NumHDonors", "NumRotatableBonds",
  "NumSaturatedRings", "RingCount", "MolLogP", "MolMR")

data_filtered <- data_filtered %>%
  dplyr::select(all_of(selected_vars))

# Remove valores ausentes
data_filtered <- na.omit(data_filtered)

# Seleciona variáveis preditoras
predictors <- setdiff(selected_vars, "ACTIVITY")

# Normaliza descritores moleculares
data_predictors <- scale(data_filtered[, predictors])

#=====
# Verifica o balanceamento dos dados
table(data_filtered$ACTIVITY)
prop.table(table(data_filtered$ACTIVITY)) # Proporção de cada classe

```

```

# Cria a tabela de proporções
prop_data <- as.data.frame(prop.table(table(data_filtered$ACTIVITY)))

# Renomeia as colunas
colnames(prop_data) <- c("ACTIVITY", "Proporção")

# Cria gráfico de barras
ggplot(prop_data, aes(x = ACTIVITY, y = Proporção, fill = ACTIVITY)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = scales::percent(Proporção, accuracy = 0.1)),
            vjust = 1.5, color = "white", size = 5) + # Ajuste do texto dentro das barras
  labs(title = "Proporção dos resultados de mutagenicidade",
        x = "ACTIVITY",
        y = "Proporção") +
  theme_minimal() +
  theme(legend.position = "none") # Remove legenda externa

=====
# Modelo de predição de mutagenicidade

# Converte ACTIVITY de 0/1 para "Negative"/"Positive": foi necessário para calc. curva ROC
data_filtered$ACTIVITY <- factor(data_filtered$ACTIVITY,
                                levels = c(0, 1),
                                labels = c("Negative", "Positive"))

# Dividi os dados em treino e teste
set.seed(123)
trainIndex <- createDataPartition(data_filtered$ACTIVITY, p = 0.8, list = FALSE)
trainData <- data_filtered[trainIndex, ]
testData <- data_filtered[-trainIndex, ]

# Validação cruzada com Random Forest
train_control <- trainControl(method = "cv", number = 10, classProbs = TRUE, summaryFunction =
twoClassSummary)

# Molécula-alvo
target_predictors <- target[, predictors]

=====
# Modelo Random Forest
=====
rf_model <- train(ACTIVITY ~ .,
                  data = trainData[, c("ACTIVITY", predictors)],
                  method = "rf",
                  trControl = train_control,
                  importance = TRUE,
                  metric = "ROC")

# Resumo do modelo
print(rf_model)

# Plotando a importância das variáveis
varImpPlot(rf_model$finalModel)

# Avaliação do modelo no conjunto de teste
rf_probabilities <- predict(rf_model, testData, type = "prob")

```

```

# Curva ROC
roc_curve <- roc(testData$ACTIVITY, rf_probabilities[, 2], levels = rev(levels(testData$ACTIVITY)))
print(auc(roc_curve))
plot(roc_curve, main = "Curva ROC - Random Forest")

# Determinar ponto de corte ótimo
optimal_cutoff <- as.numeric(coords(roc_curve, "best", ret = "threshold", best.method = "youden"))
#optimal_cutoff <- median(rf_probabilities[, 2]) #manual, usei a mediana
print(paste("Ponto de corte ótimo:", round(optimal_cutoff, 4)))

# Converter probabilidades em previsões com ponto de corte ótimo
rf_predictions_optimal <- ifelse(rf_probabilities[, 2] >= optimal_cutoff, "Positive", "Negative")
rf_predictions_optimal <- factor(rf_predictions_optimal, levels = levels(testData$ACTIVITY))

# Matriz de confusão com ponto de corte ótimo
conf_matrix_optimal <- confusionMatrix(rf_predictions_optimal, testData$ACTIVITY)
print(conf_matrix_optimal)

# Predição para a molécula alvo usando o ponto de corte ótimo
target_prediction_prob <- predict(rf_model, newdata = target_predictors, type = "prob")
target_prediction_optimal <- ifelse(target_prediction_prob[, 2] >= optimal_cutoff, "Positive", "Negative")

# Garantindo que o fator tenha os níveis corretos
target_prediction_optimal <- factor(target_prediction_optimal, levels = c("Negative", "Positive"))

cat("Predição para a molécula alvo:", as.character(target_prediction_optimal), "\n")
cat("Probabilidade da predição para a molécula alvo:\n")
print(target_prediction_prob)

# Extraíndo as métricas
precision <- conf_matrix_optimal$byClass["Pos Pred Value"] # Precisão
sensitivity <- conf_matrix_optimal$byClass["Sensitivity"] # Sensibilidade

# Calculando o F1-Score
f1_score_rf <- 2 * (precision * sensitivity) / (precision + sensitivity)
cat("F1-Score:", round(f1_score_rf, 4), "\n")

#=====
# Modelo CART
#=====
cart_model <- train(ACTIVITY ~ .,
                    data = trainData[, c("ACTIVITY", predictors)],
                    method = "rpart",
                    trControl = train_control,
                    metric = "ROC")

# Resumo do modelo
print(cart_model)

# Avaliação do modelo no conjunto de teste
cart_probabilities <- predict(cart_model, testData, type = "prob")

# Curva ROC
cart_roc_curve <- roc(testData$ACTIVITY, cart_probabilities[, 2], levels = rev(levels(testData$ACTIVITY)))
print(auc(cart_roc_curve))
plot(cart_roc_curve, main = "Curva ROC - CART")

```

```

# Determinar ponto de corte ótimo
#cart_optimal_cutoff <- coords(cart_roc_curve, "best", ret = "threshold", best.method = "youden")[[1]]
#não encontrou ponto de corte válido
cart_optimal_cutoff <- median(cart_probabilities[, 2]) #manual, usei a mediana
print(paste("Ponto de corte ótimo (CART):", round(cart_optimal_cutoff, 4)))

# Converter probabilidades em previsões com ponto de corte ótimo
cart_predictions_optimal <- ifelse(cart_probabilities[, 2] >= cart_optimal_cutoff, "Positive", "Negative")
cart_predictions_optimal <- factor(cart_predictions_optimal, levels = levels(testData$ACTIVITY))

# Matriz de confusão com ponto de corte ótimo
cart_conf_matrix_optimal <- confusionMatrix(cart_predictions_optimal, testData$ACTIVITY)
print(cart_conf_matrix_optimal)

# Plotando a árvore de decisão
rpart.plot(cart_model$finalModel)

# Predição para a molécula alvo usando o ponto de corte ótimo
cart_target_prediction_prob <- predict(cart_model, newdata = target_predictors, type = "prob")
cart_target_prediction_optimal <- ifelse(cart_target_prediction_prob[, 2] >= cart_optimal_cutoff, "Positive", "Negative")

# Garantindo que o fator tenha os níveis corretos
cart_target_prediction_optimal <- factor(cart_target_prediction_optimal, levels = c("Negative", "Positive"))

cat("Predição para a molécula alvo (CART):", as.character(cart_target_prediction_optimal), "\n")
cat("Probabilidade da predição para a molécula alvo (CART):\n")
print(cart_target_prediction_prob)

# Extraindo as métricas
precision <- cart_conf_matrix_optimal$byClass["Pos Pred Value"] # Precisão
sensitivity <- cart_conf_matrix_optimal$byClass["Sensitivity"] # Sensibilidade

# Calculando o F1-Score
f1_score_cart <- 2 * (precision * sensitivity) / (precision + sensitivity)
cat("F1-Score:", round(f1_score_cart, 4), "\n")

#=====
# Modelo XGBoost
#=====

# Definir hiperparâmetros do XGBoost
xgb_grid <- expand.grid(nrounds = 100,
                      max_depth = 3,
                      eta = 0.3,
                      gamma = 0,
                      colsample_bytree = 1,
                      min_child_weight = 1,
                      subsample = 1)

# Treinamento do modelo
xgb_model <- train(ACTIVITY ~ .,
                  data = trainData[, c("ACTIVITY", predictors)],
                  method = "xgbTree",
                  trControl = train_control,
                  tuneGrid = xgb_grid,
                  metric = "ROC",
                  verbose = 0)

```

```

# Resumo do modelo
print(xgb_model)

# Avaliação do modelo no conjunto de teste
xgb_probabilities <- predict(xgb_model, testData, type = "prob")

# Curva ROC
xgb_roc_curve <- roc(testData$ACTIVITY, xgb_probabilities[, 2], levels = rev(levels(testData$ACTIV-
ITY)))
print(auc(xgb_roc_curve))
plot(xgb_roc_curve, main = "Curva ROC - XGBoost")

# Determinar ponto de corte ótimo
xgb_optimal_cutoff <- coords(xgb_roc_curve, "best", ret = "threshold", best.method = "youden")[[1]]
#xgb_optimal_cutoff <- median(xgb_probabilities[, 2]) #manual, usei a mediana
print(paste("Ponto de corte ótimo (XGBoost):", round(xgb_optimal_cutoff, 4)))

# Converter probabilidades em previsões com ponto de corte ótimo
xgb_predictions_optimal <- ifelse(xgb_probabilities[, 2] >= xgb_optimal_cutoff, "Positive", "Negative")
xgb_predictions_optimal <- factor(xgb_predictions_optimal, levels = levels(testData$ACTIVITY))

# Matriz de confusão com ponto de corte ótimo
xgb_conf_matrix_optimal <- confusionMatrix(xgb_predictions_optimal, testData$ACTIVITY)
print(xgb_conf_matrix_optimal)

# Predição para a molécula alvo usando o ponto de corte ótimo
xgb_target_prediction_prob <- predict(xgb_model, newdata = target_predictors, type = "prob")
xgb_target_prediction_optimal <- ifelse(xgb_target_prediction_prob[, 2] >= xgb_optimal_cutoff, "Pos-
itive", "Negative")

# Garantindo que o fator tenha os níveis corretos
xgb_target_prediction_optimal <- factor(xgb_target_prediction_optimal, levels = c("Negative", "Posi-
tive"))

cat("Predição para a molécula alvo (XGBoost):", as.character(xgb_target_prediction_optimal), "\n")
cat("Probabilidade da predição para a molécula alvo (XGBoost):\n")
print(xgb_target_prediction_prob)

# Extraíndo as métricas
precision <- xgb_conf_matrix_optimal$byClass["Pos Pred Value"] # Precisão
sensitivity <- xgb_conf_matrix_optimal$byClass["Sensitivity"] # Sensibilidade

# Calculando o F1-Score
f1_score_xgb <- 2 * (precision * sensitivity) / (precision + sensitivity)
cat("F1-Score:", round(f1_score_xgb, 4), "\n")

#=====
# Resultados

# Consolidando métricas existentes
results <- data.frame(
  Model = c("Random Forest", "CART", "XGBoost"),
  AUC = c(auc(roc_curve), auc(cart_roc_curve), auc(xgb_roc_curve)),
  Accuracy = c(conf_matrix_optimal$overall["Accuracy"], cart_conf_matrix_optimal$overall["Accu-
racy"], xgb_conf_matrix_optimal$overall["Accuracy"]),
  Kappa = c(conf_matrix_optimal$overall["Kappa"], cart_conf_matrix_optimal$overall["Kappa"],
xgb_conf_matrix_optimal$overall["Kappa"]),

```

```

Sensitivity = c(conf_matrix_optimal$byClass["Sensitivity"], cart_conf_matrix_optimal$by-
Class["Sensitivity"], xgb_conf_matrix_optimal$byClass["Sensitivity"]),
Specificity = c(conf_matrix_optimal$byClass["Specificity"], cart_conf_matrix_optimal$by-
Class["Specificity"], xgb_conf_matrix_optimal$byClass["Specificity"]),
F1_Score = c(f1_score_rf, f1_score_cart, f1_score_xgb),
Optimal_Cutoff = c(optimal_cutoff, cart_optimal_cutoff, xgb_optimal_cutoff),
Probability_Positive = c(target_prediction_prob[1, "Positive"], cart_target_prediction_prob[1, "Posi-
tive"], xgb_target_prediction_prob[1, "Positive"]),
Probability_Negative = c(target_prediction_prob[1, "Negative"], cart_target_prediction_prob[1,
"Negative"], xgb_target_prediction_prob[1, "Negative"]),
Mutagenicity_Prediction = c(as.character(target_prediction_optimal), as.character(cart_target_pre-
diction_optimal), as.character(xgb_target_prediction_optimal))
)

# Salvando os resultados em um arquivo Excel
write.xlsx(list(Performance = results), "script_controle_1omeprazol_results.xlsx", rowNames =
FALSE)

```

## APÊNDICE E

```

#=====
# Descrição:
# Domínio de aplicação: filtra moléculas com MolWt <= 1000 e MolLogP <= 10
# Verifica balanceamento
# Modelos com os dados do cluster mais próximo, com definição de espaço químico
# Predição com ponto de corte padrão (0,5)
#=====

source("Pacotes.R") # Evoca o script de carregamento de pacotes

cat("\014") # Limpa o console
rm(list = ls(all = TRUE)) # Limpa a memória
options(scipen = 999) # Desabilita notação científica

# Carrega os dados
data_path <- "mutagenicity_m.xlsx"
target_path <- "omeprazol_descritores.xlsx"

data <- read_excel(data_path)
target <- read_excel(target_path)

# Domínio de aplicação: filtra moléculas com MolWt <= 1000 e MolLogP <= 10
data_filtered <- data %>%
  filter(MolWt <= 1000, MolLogP <= 10)

# Transforma a variável ACTIVITY em fator
data_filtered$ACTIVITY <- as.factor(data_filtered$ACTIVITY)

# Seleciona as variáveis relevantes
selected_vars <- c("ACTIVITY", "SPS", "MolWt", "NumValenceElectrons", "FpDensityMorgan1",
  "FpDensityMorgan2", "FpDensityMorgan3", "BCUT2D_MWHI", "BCUT2D_MWLOW",
  "BCUT2D_CHGHI", "BCUT2D_CHGLO", "BCUT2D_LOGPHI", "BCUT2D_LOGPLOW",
  "BCUT2D_MRHI", "BCUT2D_MRLOW", "AvgIpc", "BalabanJ", "BertzCT", "Chi0",
  "HallKierAlpha", "Kappa2", "Kappa3", "LabuteASA", "PEOE_VSA7",
  "SMR_VSA10", "SlogP_VSA2", "TPSA", "FractionCSP3", "NumAliphaticRings",
  "NumAromaticRings", "NumHAcceptors", "NumHDonors", "NumRotatableBonds",
  "NumSaturatedRings", "RingCount", "MolLogP", "MolMR")

data_filtered <- data_filtered %>%
  dplyr::select(all_of(selected_vars))

# Remove valores ausentes
data_filtered <- na.omit(data_filtered)

# Seleciona variáveis preditoras
predictors <- setdiff(selected_vars, "ACTIVITY")

# Normaliza descritores moleculares
data_predictors <- scale(data_filtered[, predictors])

#=====
# Escolha do número de clusters
#=====

```

```

# Função para calcular o índice de silhueta para diferentes números de clusters
silhouette_analysis <- function(data, max_clusters = 10) {
  avg_sil_width <- numeric(max_clusters)

  for (k in 2:max_clusters) {
    kmeans_res <- kmeans(data, centers = k, nstart = 25, iter.max = 100)
    sil <- silhouette(kmeans_res$cluster, dist(data))
    avg_sil_width[k] <- mean(sil[, 3]) # Média das larguras de silhueta
  }

  # Cria um data frame para plotagem
  sil_data <- data.frame(
    clusters = 2:max_clusters,
    avg_silhouette = avg_sil_width[2:max_clusters]
  )

  # Plota usando ggplot2
  p <- ggplot(sil_data, aes(x = clusters, y = avg_silhouette)) +
    geom_line(color = "black", size = 0.6) +
    geom_point(color = "blue", size = 3) +
    geom_text(aes(label = round(avg_silhouette, 3)), vjust = -1, size = 4) + # Adicionar valores acima
dos pontos
    labs(title = "Silhouette Analysis for K-Means Clustering",
         x = "Number of Clusters",
         y = "Average Silhouette Width") +
    theme_minimal()

  print(p)

  # Retorna o número de clusters com maior índice de silhueta
  optimal_clusters <- which.max(avg_sil_width)
  cat("Número ideal de clusters:", optimal_clusters, "\n")

  return(optimal_clusters)
}

# Executa a análise de silhueta
optimal_k <- silhouette_analysis(data_predictors, max_clusters = 10)

#####
# Elbow method
# Função para calcular a soma dos quadrados intra-cluster para diferentes números de clusters
elbow_method <- function(data, max_clusters = 10) {
  wcss <- numeric(max_clusters)

  for (k in 1:max_clusters) {
    kmeans_res <- kmeans(data, centers = k, nstart = 25, iter.max = 100)
    wcss[k] <- kmeans_res$tot.withinss
  }

  # Cria o gráfico usando ggplot2
  elbow_plot <- ggplot(data.frame(Clusters = 1:max_clusters, WCSS = wcss), aes(x = Clusters, y =
WCSS)) +
    geom_line(color = "black") +
    geom_point(size = 3, color = "red") +
    labs(title = "Elbow Method for Optimal Clusters", x = "Number of Clusters", y = "WCSS (Within-
Cluster Sum of Squares)") +
    theme_minimal()

```

```

    print(elbow_plot)
  }
# Executa a análise do método do cotovelo
elbow_method(data_predictors, max_clusters = 10)

#=====

# Dendrograma
dist_matrix <- dist(data_predictors, method = "euclidean") # Calcula a matriz de distância
hclust_res <- hclust(dist_matrix, method = "ward.D2") # Realiza a aglomeração hierárquica
# Converte o resultado de hclust para um objeto compatível com ggplot
hclust_dendro <- as.dendrogram(hclust_res)
dendro_data <- ggdendro::dendro_data(hclust_dendro)
# Plota o dendrograma usando ggplot2
ggplot() +
  geom_segment(data = dendro_data$segments, aes(x = x, y = y, xend = xend, yend = yend)) +
  labs(title = "Dendrograma - Método Hierárquico", x = "Observações", y = "Distância") +
  geom_hline(yintercept = 110, color = "red", linetype = "dashed") + # Linha de corte
  theme_minimal()

#=====

# Agrupamento utilizando K-means
set.seed(123)
kmeans_res <- kmeans(data_predictors, centers = 5, nstart = 25, iter.max = 100)

table(kmeans_res$cluster)

# Gráficos para verificar distribuição
# Cria a tabela de proporções e contagens dos clusters
cluster_counts <- as.data.frame(table(kmeans_res$cluster))
cluster_counts$Proporção <- prop.table(cluster_counts$Freq)

# Renomeia colunas
colnames(cluster_counts) <- c("Cluster", "Frequência", "Proporção")

# Converte para fator para manter a ordem no gráfico
cluster_counts$Cluster <- as.factor(cluster_counts$Cluster)

# Cria o gráfico de barras com legenda dentro das colunas
ggplot(cluster_counts, aes(x = Cluster, y = Proporção, fill = Cluster)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = paste0(Frequência, " (", scales::percent(Proporção, accuracy = 0.1), ")")),
    vjust = 1.5, color = "white", size = 5) + # Ajuste do texto dentro das barras
  labs(title = "Proporção e Frequência em cada Cluster",
    x = "Cluster",
    y = "Proporção") +
  theme_minimal() +
  theme(legend.position = "none") # Remove legenda externa

# Gráfico para verificar balanceamento dos dados em cada cluster
# Cria a tabela cruzada com as contagens de Activity por cluster
activity_counts <- as.data.frame(table(Cluster = kmeans_res$cluster, Activity = data_filtered$ACTIVITY))

# Calcula proporções dentro de cada cluster
activity_counts <- activity_counts %>%
  group_by(Cluster) %>%
  mutate(Proporção = Freq / sum(Freq)) %>%

```

```

ungroup()

print(activity_counts)

# Converte o Cluster para fator
activity_counts$Cluster <- as.factor(activity_counts$Cluster)

# Cria o gráfico de barras empilhadas
ggplot(activity_counts, aes(x = Cluster, y = Proporção, fill = as.factor(Activity))) +
  geom_bar(stat = "identity", position = "fill") + # "fill" para normalizar em cada cluster
  geom_text(aes(label = scales::percent(Proporção, accuracy = 0.1)),
            position = position_fill(vjust = 0.5), color = "white", size = 5) +
  labs(title = "Distribuição da variável Activity dentro de cada cluster",
        x = "Cluster",
        y = "Proporção",
        fill = "Activity") +
  theme_minimal()

#=====
#Visualização dos dados:
# PCA para redução de dimensionalidade
pca <- prcomp(data_predictors, center = TRUE, scale. = TRUE)
summary(pca)

# Visualiza a variância explicada pela PCA
fviz_eig(pca, addlabels = TRUE, ylim = c(0, 50))

# Transforma os dados para o espaço PCA
data_pca <- data.frame(pca$x)

# Adicionando os clusters aos dados
data_filtered$Cluster <- as.factor(kmeans_res$cluster)
table(data_filtered$Cluster)

# Adiciona as coordenadas PCA
data_filtered$PC1 <- data_pca$PC1
data_filtered$PC2 <- data_pca$PC2

# Plota os clusters
ggplot(data_filtered, aes(x = PC1, y = PC2, color = Cluster)) +
  geom_point() +
  theme_minimal() +
  labs(title = "Clusters de Moléculas no Espaço PCA")

#=====
# Normaliza os descritores da molécula alvo
target_predictors <- target[, predictors]
target_predictors <- scale(target_predictors, center = attr(data_predictors, "scaled:center"), scale =
attr(data_predictors, "scaled:scale"))

#=====
#Espaço químico
Kmeans_centroids <- kmeans_res$centers
distances <- apply(Kmeans_centroids, 1, function(x) sqrt(sum((x - target_predictors)^2))) #distância
euclidiana
cluster_target <- as.numeric(which.min(distances))
closest_molecules <- data_filtered[which(data_filtered$Cluster==cluster_target), c("ACTIVITY", pre-
dictors)]

```

```

#=====
# Visualiza as moléculas do cluster mais próximo
target_pca <- predict(pca, newdata = target_predictors) # Projeção da molécula alvo no espaço PCA

# Plota os clusters com a molécula alvo destacada
ggplot(data_filtered, aes(x = PC1, y = PC2, color = Cluster)) +
  geom_point() +
  geom_point(aes(x = target_pca[1], y = target_pca[2]), color = "black", size = 4, shape = 19) + #
Molécula alvo
  theme_minimal() +
  labs(title = "Clusters de Moléculas no Espaço PCA", x = "PC1", y = "PC2", color = "Cluster")

# Visualiza as moléculas mais próximas
# Seleciona apenas as variáveis preditoras para normalização
closest_molecules_predictors <- closest_molecules[, predictors, drop = FALSE]
# Normaliza com os mesmos parâmetros usados na base original
closest_molecules_predictors <- scale(closest_molecules_predictors,
                                     center = attr(data_predictors, "scaled:center"),
                                     scale = attr(data_predictors, "scaled:scale"))

# Projeção das moléculas mais próximas no espaço PCA
closest_molecules_pca <- predict(pca, newdata = closest_molecules_predictors)
closest_molecules_pca <- as.data.frame(closest_molecules_pca) # Converter para data frame se
necessário

# Adiciona as componentes principais ao conjunto de moléculas mais próximas
closest_molecules$PC1 <- closest_molecules_pca[, "PC1"]
closest_molecules$PC2 <- closest_molecules_pca[, "PC2"]

ggplot(closest_molecules, aes(x = PC1, y = PC2, color = ACTIVITY)) +
  geom_point() +
  geom_point(aes(x = target_pca[1], y = target_pca[2]), color = "blue", size = 6, shape = 18) + #
Molécula alvo
  theme_minimal() +
  labs(title = "Moléculas mais próximas da Molécula Alvo")

#=====
#=====
# Modelo de predição de mutagenicidade

# Converte ACTIVITY de 0/1 para "Negative"/"Positive": foi necessário para calc. curva ROC
closest_molecules$ACTIVITY <- factor(closest_molecules$ACTIVITY,
                                     levels = c(0, 1),
                                     labels = c("Negative", "Positive"))

# Divide os dados em treino e teste
set.seed(123)
trainIndex <- createDataPartition(closest_molecules$ACTIVITY, p = 0.8, list = FALSE)
trainData <- closest_molecules[trainIndex, ]
testData <- closest_molecules[-trainIndex, ]

# Validação cruzada com Random Forest
train_control <- trainControl(method = "cv", number = 10, classProbs = TRUE, summaryFunction =
twoClassSummary)

#=====
# Modelo Random Forest

```

```

#=====
rf_model <- train(ACTIVITY ~ .,
                  data = trainData[, c("ACTIVITY", predictors)],
                  method = "rf",
                  trControl = train_control,
                  importance = TRUE,
                  metric = "ROC")

# Resumo do modelo
print(rf_model)

# Avaliação do modelo no conjunto de teste
rf_predictions <- predict(rf_model, testData)
rf_probabilities <- predict(rf_model, testData, type = "prob")

# Matriz de confusão
conf_matrix <- confusionMatrix(rf_predictions, testData$ACTIVITY)
print(conf_matrix)

# Plotando a importância das variáveis
varImpPlot(rf_model$finalModel)

# Predição para a molécula alvo
target_prediction <- predict(rf_model, newdata = target_predictors)
print(paste("Predição para a molécula alvo:", target_prediction))

# Probabilidade da predição
target_prediction_prob <- predict(rf_model, newdata = target_predictors, type = "prob")
print("Probabilidade da predição para a molécula alvo:")
print(target_prediction_prob)

# Curva ROC
roc_curve <- roc(testData$ACTIVITY, rf_probabilities[, 2], levels = rev(levels(testData$ACTIVITY)))
print(auc(roc_curve))
plot(roc_curve, main = "Curva ROC - Random Forest")

# Extraíndo as métricas
precision <- conf_matrix$byClass["Pos Pred Value"] # Precisão
sensitivity <- conf_matrix$byClass["Sensitivity"] # Sensibilidade

# Calculando o F1-Score
f1_score_rf <- 2 * (precision * sensitivity) / (precision + sensitivity)
cat("F1-Score:", round(f1_score_rf, 4), "\n")

#=====
# Modelo CAR
#=====
car_model <- train(ACTIVITY ~ .,
                  data = trainData[, c("ACTIVITY", predictors)],
                  method = "rpart",
                  trControl = train_control,
                  metric = "ROC")

# Resumo do modelo
print(car_model)

# Avaliação do modelo no conjunto de teste
car_predictions <- predict(car_model, testData)
car_probabilities <- predict(car_model, testData, type = "prob")

```

```

# Matriz de confusão
car_conf_matrix <- confusionMatrix(car_predictions, testData$ACTIVITY)
print(car_conf_matrix)

# Plotando a árvore de decisão
rpart.plot(car_model$finalModel)

# Predição para a molécula alvo
car_target_prediction <- predict(car_model, newdata = target_predictors)
print(paste("Predição para a molécula alvo:", car_target_prediction))

# Probabilidade da predição
car_target_prediction_prob <- predict(car_model, newdata = target_predictors, type = "prob")
print("Probabilidade da predição para a molécula alvo:")
print(car_target_prediction_prob)

# Curva ROC
car_roc_curve <- roc(testData$ACTIVITY, car_probabilities[, 2], levels = rev(levels(testData$ACTIV-
ITY)))
print(auc(car_roc_curve))
plot(car_roc_curve, main = "Curva ROC - CAR")

# Extraíndo as métricas
precision <- car_conf_matrix$byClass["Pos Pred Value"] # Precisão
sensitivity <- car_conf_matrix$byClass["Sensitivity"] # Sensibilidade

# Calculando o F1-Score
f1_score_cart <- 2 * (precision * sensitivity) / (precision + sensitivity)
cat("F1-Score:", round(f1_score_cart, 4), "\n")

#=====
# Modelo XGBoost
#=====
xgb_grid <- expand.grid(nrounds = 100,
                      max_depth = 3,
                      eta = 0.3,
                      gamma = 0,
                      colsample_bytree = 1,
                      min_child_weight = 1,
                      subsample = 1)

xgb_model <- train(ACTIVITY ~ .,
                  data = trainData[, c("ACTIVITY", predictors)],
                  method = "xgbTree",
                  trControl = train_control,
                  tuneGrid = xgb_grid,
                  metric = "ROC",
                  verbose = 0)

# Resumo do modelo
print(xgb_model)

# Avaliação do modelo no conjunto de teste
xgb_predictions <- predict(xgb_model, testData)
xgb_probabilities <- predict(xgb_model, testData, type = "prob")

# Matriz de confusão
xgb_conf_matrix <- confusionMatrix(xgb_predictions, testData$ACTIVITY)

```

```

print(xgb_conf_matrix)

# Predição para a molécula alvo
xgb_target_prediction <- predict(xgb_model, newdata = target_predictors)
print(paste("Predição para a molécula alvo:", xgb_target_prediction))

# Probabilidade da predição
xgb_target_prediction_prob <- predict(xgb_model, newdata = target_predictors, type = "prob")
print("Probabilidade da predição para a molécula alvo:")
print(xgb_target_prediction_prob)

# Curva ROC
xgb_roc_curve <- roc(testData$ACTIVITY, xgb_probabilities[, 2], levels = rev(levels(testData$ACTIV-
ITY)))
print(auc(xgb_roc_curve))
plot(xgb_roc_curve, main = "Curva ROC - XGBoost")

# Extraíndo as métricas
precision <- xgb_conf_matrix$byClass["Pos Pred Value"] # Precisão
sensitivity <- xgb_conf_matrix$byClass["Sensitivity"] # Sensibilidade

# Calculando o F1-Score
f1_score_xgb <- 2 * (precision * sensitivity) / (precision + sensitivity)
cat("F1-Score:", round(f1_score_xgb, 4), "\n")

#=====
# Resultados

# Consolidando métricas existentes
results <- data.frame(
  Model = c("Random Forest", "CART", "XGBoost"),
  AUC = c(auc(roc_curve), auc(car_roc_curve), auc(xgb_roc_curve)),
  Accuracy = c(conf_matrix$overall["Accuracy"], car_conf_matrix$overall["Accuracy"], xgb_conf_ma-
trix$overall["Accuracy"]),
  Kappa = c(conf_matrix$overall["Kappa"], car_conf_matrix$overall["Kappa"], xgb_conf_matrix$over-
all["Kappa"]),
  Sensitivity = c(conf_matrix$byClass["Sensitivity"], car_conf_matrix$byClass["Sensitivity"],
xgb_conf_matrix$byClass["Sensitivity"]),
  Specificity = c(conf_matrix$byClass["Specificity"], car_conf_matrix$byClass["Specificity"],
xgb_conf_matrix$byClass["Specificity"]),
  F1_Score = c(f1_score_rf, f1_score_cart, f1_score_xgb),
  Probability_Positive = c(target_prediction_prob[1, "Positive"], car_target_prediction_prob[1, "Posi-
tive"], xgb_target_prediction_prob[1, "Positive"]),
  Probability_Negative = c(target_prediction_prob[1, "Negative"], car_target_prediction_prob[1, "Neg-
ative"], xgb_target_prediction_prob[1, "Negative"]),
  Cluster_Target = cluster_target,
  Mutagenicity_Prediction = c(as.character(target_prediction), as.character(car_target_prediction),
as.character(xgb_target_prediction))
)

# Salvando os resultados em um arquivo Excel
write.xlsx(list(Performance = results), "script_1omeprazol_results.xlsx", rowNames = FALSE)

```

## APÊNDICE F

```

#=====
# Descrição:
# Domínio de aplicação: filtra moléculas com MolWt <= 1000 e MolLogP <= 10
# Verifica balanceamento
# Modelos com os dados do cluster mais próximo, com definição de espaço químico
# Predição com ponto de corte ótimo
#=====

source("Pacotes.R") # Evoca o script de carregamento de pacotes

cat("\014") # Limpa o console
rm(list = ls(all = TRUE)) # Limpa a memória
options(scipen = 999) # Desabilita notação científica

# Carrega os dados
data_path <- "mutagenicity_m.xlsx"
target_path <- "omeprazol_descritores.xlsx"

data <- read_excel(data_path)
target <- read_excel(target_path)

# Domínio de aplicação: filtra moléculas com MolWt <= 1000 e MolLogP <= 10
data_filtered <- data %>%
  filter(MolWt <= 1000, MolLogP <= 10)

# Transforma a variável ACTIVITY em fator
data_filtered$ACTIVITY <- as.factor(data_filtered$ACTIVITY)

# Seleciona as variáveis relevantes
selected_vars <- c("ACTIVITY", "SPS", "MolWt", "NumValenceElectrons", "FpDensityMorgan1",
  "FpDensityMorgan2", "FpDensityMorgan3", "BCUT2D_MWHI", "BCUT2D_MWLOW",
  "BCUT2D_CHGHI", "BCUT2D_CHGLO", "BCUT2D_LOGPHI", "BCUT2D_LOGPLOW",
  "BCUT2D_MRHI", "BCUT2D_MRLOW", "AvgIpc", "BalabanJ", "BertzCT", "Chi0",
  "HallKierAlpha", "Kappa2", "Kappa3", "LabuteASA", "PEOE_VSA7",
  "SMR_VSA10", "SlogP_VSA2", "TPSA", "FractionCSP3", "NumAliphaticRings",
  "NumAromaticRings", "NumHAcceptors", "NumHDonors", "NumRotatableBonds",
  "NumSaturatedRings", "RingCount", "MolLogP", "MolMR")

data_filtered <- data_filtered %>%
  dplyr::select(all_of(selected_vars))

# Remove valores ausentes
data_filtered <- na.omit(data_filtered)

# Seleciona variáveis preditoras
predictors <- setdiff(selected_vars, "ACTIVITY")

# Normaliza descritores moleculares
data_predictors <- scale(data_filtered[, predictors])

#=====
# Escolha do número de clusters
#=====

```

```

# Função para calcular o índice de silhueta para diferentes números de clusters
silhouette_analysis <- function(data, max_clusters = 10) {
  avg_sil_width <- numeric(max_clusters)

  for (k in 2:max_clusters) {
    kmeans_res <- kmeans(data, centers = k, nstart = 25, iter.max = 100)
    sil <- silhouette(kmeans_res$cluster, dist(data))
    avg_sil_width[k] <- mean(sil[, 3]) # Média das larguras de silhueta
  }

  # Cria um data frame para plotagem
  sil_data <- data.frame(
    clusters = 2:max_clusters,
    avg_silhouette = avg_sil_width[2:max_clusters]
  )

  # Plota usando ggplot2
  p <- ggplot(sil_data, aes(x = clusters, y = avg_silhouette)) +
    geom_line(color = "black", size = 0.6) +
    geom_point(color = "blue", size = 3) +
    geom_text(aes(label = round(avg_silhouette, 3)), vjust = -1, size = 4) + # Adicionar valores acima
dos pontos
    labs(title = "Silhouette Analysis for K-Means Clustering",
         x = "Number of Clusters",
         y = "Average Silhouette Width") +
    theme_minimal()

  print(p)

  # Retorna o número de clusters com maior índice de silhueta
  optimal_clusters <- which.max(avg_sil_width)
  cat("Número ideal de clusters:", optimal_clusters, "\n")

  return(optimal_clusters)
}

# Executa a análise de silhueta
optimal_k <- silhouette_analysis(data_predictors, max_clusters = 10)

#=====
# Elbow method
# Função para calcular a soma dos quadrados intra-cluster para diferentes números de clusters
elbow_method <- function(data, max_clusters = 10) {
  wcss <- numeric(max_clusters)

  for (k in 1:max_clusters) {
    kmeans_res <- kmeans(data, centers = k, nstart = 25, iter.max = 100)
    wcss[k] <- kmeans_res$tot.withinss
  }

  # Cria o gráfico usando ggplot2
  elbow_plot <- ggplot(data.frame(Clusters = 1:max_clusters, WCSS = wcss), aes(x = Clusters, y =
WCSS)) +
    geom_line(color = "black") +
    geom_point(size = 3, color = "red") +
    labs(title = "Elbow Method for Optimal Clusters", x = "Number of Clusters", y = "WCSS (Within-
Cluster Sum of Squares)") +
    theme_minimal()

```

```

print(elbow_plot)
}
# Executa a análise do método do cotovelo
elbow_method(data_predictors, max_clusters = 10)

#=====

# Dendrograma
dist_matrix <- dist(data_predictors, method = "euclidean") # Calcula a matriz de distância
hclust_res <- hclust(dist_matrix, method = "ward.D2") # Realiza a aglomeração hierárquica
# Converte o resultado de hclust para um objeto compatível com ggplot
hclust_dendro <- as.dendrogram(hclust_res)
dendro_data <- ggdendro::dendro_data(hclust_dendro)
# Plota o dendrograma usando ggplot2
ggplot() +
  geom_segment(data = dendro_data$segments, aes(x = x, y = y, xend = xend, yend = yend)) +
  labs(title = "Dendrograma - Método Hierárquico", x = "Observações", y = "Distância") +
  geom_hline(yintercept = 110, color = "red", linetype = "dashed") + # Linha de corte
  theme_minimal()

#=====

# Agrupamento utilizando K-means
set.seed(123)
kmeans_res <- kmeans(data_predictors, centers = 5, nstart = 25, iter.max = 100)

table(kmeans_res$cluster)

# Gráficos para verificar distribuição
# Cria a tabela de proporções e contagens dos clusters
cluster_counts <- as.data.frame(table(kmeans_res$cluster))
cluster_counts$Proporção <- prop.table(cluster_counts$Freq)

# Renomeia colunas
colnames(cluster_counts) <- c("Cluster", "Frequência", "Proporção")

# Converte para fator para manter a ordem no gráfico
cluster_counts$Cluster <- as.factor(cluster_counts$Cluster)

# Cria o gráfico de barras com legenda dentro das colunas
ggplot(cluster_counts, aes(x = Cluster, y = Proporção, fill = Cluster)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = paste0(Frequência, " (", scales::percent(Proporção, accuracy = 0.1), ")")),
    vjust = 1.5, color = "white", size = 5) + # Ajuste do texto dentro das barras
  labs(title = "Proporção e Frequência em cada Cluster",
    x = "Cluster",
    y = "Proporção") +
  theme_minimal() +
  theme(legend.position = "none") # Remove legenda externa

# Gráfico para verificar balanceamento dos dados em cada cluster
# Cria a tabela cruzada com as contagens de Activity por cluster
activity_counts <- as.data.frame(table(Cluster = kmeans_res$cluster, Activity = data_filtered$ACTIVITY))

# Calcula proporções dentro de cada cluster
activity_counts <- activity_counts %>%
  group_by(Cluster) %>%
  mutate(Proporção = Freq / sum(Freq)) %>%
  ungroup()

```

```

print(activity_counts)

# Converte o Cluster para fator
activity_counts$Cluster <- as.factor(activity_counts$Cluster)

# Cria o gráfico de barras empilhadas
ggplot(activity_counts, aes(x = Cluster, y = Proporção, fill = as.factor(Activity))) +
  geom_bar(stat = "identity", position = "fill") + # "fill" para normalizar em cada cluster
  geom_text(aes(label = scales::percent(Proporção, accuracy = 0.1)),
            position = position_fill(vjust = 0.5), color = "white", size = 5) +
  labs(title = "Distribuição da variável Activity dentro de cada cluster",
        x = "Cluster",
        y = "Proporção",
        fill = "Activity") +
  theme_minimal()

#=====
#Visualização dos dados:
# PCA para redução de dimensionalidade
pca <- prcomp(data_predictors, center = TRUE, scale. = TRUE)
summary(pca)

# Visualiza a variância explicada pela PCA
fviz_eig(pca, addlabels = TRUE, ylim = c(0, 50))

# Transforma os dados para o espaço PCA
data_pca <- data.frame(pca$x)

# Adicionando os clusters aos dados
data_filtered$Cluster <- as.factor(kmeans_res$cluster)
table(data_filtered$Cluster)

# Adiciona as coordenadas PCA
data_filtered$PC1 <- data_pca$PC1
data_filtered$PC2 <- data_pca$PC2

# Plota os clusters
ggplot(data_filtered, aes(x = PC1, y = PC2, color = Cluster)) +
  geom_point() +
  theme_minimal() +
  labs(title = "Clusters de Moléculas no Espaço PCA")

#=====
# Normaliza os descritores da molécula alvo
target_predictors <- target[, predictors]
target_predictors <- scale(target_predictors, center = attr(data_predictors, "scaled:center"), scale =
attr(data_predictors, "scaled:scale"))

#=====
#Espaço químico
Kmeans_centroids <- kmeans_res$centers
distances <- apply(Kmeans_centroids, 1, function(x) sqrt(sum((x - target_predictors)^2))) #distância
euclidiana
cluster_target <- as.numeric(which.min(distances))
closest_molecules <- data_filtered[which(data_filtered$Cluster==cluster_target), c("ACTIVITY", pre-
dictors)]

#=====

```

```

# Visualiza as moléculas do cluster mais próximo
target_pca <- predict(pca, newdata = target_predictors) # Projeção da molécula alvo no espaço PCA

# Plota os clusters com a molécula alvo destacada
ggplot(data_filtered, aes(x = PC1, y = PC2, color = Cluster)) +
  geom_point() +
  geom_point(aes(x = target_pca[1], y = target_pca[2]), color = "black", size = 4, shape = 19) + #
Molécula alvo
  theme_minimal() +
  labs(title = "Clusters de Moléculas no Espaço PCA", x = "PC1", y = "PC2", color = "Cluster")

# Visualiza as moléculas mais próximas
# Seleciona apenas as variáveis preditoras para normalização
closest_molecules_predictors <- closest_molecules[, predictors, drop = FALSE]
# Normaliza com os mesmos parâmetros usados na base original
closest_molecules_predictors <- scale(closest_molecules_predictors,
                                     center = attr(data_predictors, "scaled:center"),
                                     scale = attr(data_predictors, "scaled:scale"))

# Projeção das moléculas mais próximas no espaço PCA
closest_molecules_pca <- predict(pca, newdata = closest_molecules_predictors)
closest_molecules_pca <- as.data.frame(closest_molecules_pca) # Converter para data frame se
necessário

# Adiciona as componentes principais ao conjunto de moléculas mais próximas
closest_molecules$PC1 <- closest_molecules_pca[, "PC1"]
closest_molecules$PC2 <- closest_molecules_pca[, "PC2"]

ggplot(closest_molecules, aes(x = PC1, y = PC2, color = ACTIVITY)) +
  geom_point() +
  geom_point(aes(x = target_pca[1], y = target_pca[2]), color = "blue", size = 6, shape = 18) + #
Molécula alvo
  theme_minimal() +
  labs(title = "Moléculas mais próximas da Molécula Alvo")

#=====
#=====
# Modelo de predição de mutagenicidade

# Converte ACTIVITY de 0/1 para "Negative"/"Positive": foi necessário para calc. curva ROC
closest_molecules$ACTIVITY <- factor(closest_molecules$ACTIVITY,
                                     levels = c(0, 1),
                                     labels = c("Negative", "Positive"))

# Divide os dados em treino e teste
set.seed(123)
trainIndex <- createDataPartition(closest_molecules$ACTIVITY, p = 0.8, list = FALSE)
trainData <- closest_molecules[trainIndex, ]
testData <- closest_molecules[-trainIndex, ]

# Validação cruzada com Random Forest
train_control <- trainControl(method = "cv", number = 10, classProbs = TRUE, summaryFunction =
twoClassSummary)

#=====
# Modelo Random Forest
#=====
rf_model <- train(ACTIVITY ~ .,
                  data = trainData[, c("ACTIVITY", predictors)],

```

```

        method = "rf",
        trControl = train_control,
        importance = TRUE,
        metric = "ROC")

# Resumo do modelo
print(rf_model)

# Plotando a importância das variáveis
varImpPlot(rf_model$finalModel)

# Avaliação do modelo no conjunto de teste
rf_probabilities <- predict(rf_model, testData, type = "prob")

# Curva ROC
roc_curve <- roc(testData$ACTIVITY, rf_probabilities[, 2], levels = rev(levels(testData$ACTIVITY)))
print(auc(roc_curve))
plot(roc_curve, main = "Curva ROC - Random Forest")

# Determinar ponto de corte ótimo
optimal_cutoff <- as.numeric(coords(roc_curve, "best", ret = "threshold", best.method = "youden"))
#optimal_cutoff <- median(rf_probabilities[, 2]) #manual, usei a mediana
print(paste("Ponto de corte ótimo:", round(optimal_cutoff, 4)))

# Converter probabilidades em previsões com ponto de corte ótimo
rf_predictions_optimal <- ifelse(rf_probabilities[, 2] >= optimal_cutoff, "Positive", "Negative")
rf_predictions_optimal <- factor(rf_predictions_optimal, levels = levels(testData$ACTIVITY))

# Matriz de confusão com ponto de corte ótimo
conf_matrix_optimal <- confusionMatrix(rf_predictions_optimal, testData$ACTIVITY)
print(conf_matrix_optimal)

# Predição para a molécula alvo usando o ponto de corte ótimo
target_prediction_prob <- predict(rf_model, newdata = target_predictors, type = "prob")
target_prediction_optimal <- ifelse(target_prediction_prob[, 2] >= optimal_cutoff, "Positive", "Negative")

# Garantindo que o fator tenha os níveis corretos
target_prediction_optimal <- factor(target_prediction_optimal, levels = c("Negative", "Positive"))

cat("Predição para a molécula alvo:", as.character(target_prediction_optimal), "\n")
cat("Probabilidade da predição para a molécula alvo:\n")
print(target_prediction_prob)

# Extraíndo as métricas
precision <- conf_matrix_optimal$byClass["Pos Pred Value"] # Precisão
sensitivity <- conf_matrix_optimal$byClass["Sensitivity"] # Sensibilidade

# Calculando o F1-Score
f1_score_rf <- 2 * (precision * sensitivity) / (precision + sensitivity)
cat("F1-Score:", round(f1_score_rf, 4), "\n")

#=====
# Modelo CART
#=====
cart_model <- train(ACTIVITY ~ .,
                    data = trainData[, c("ACTIVITY", predictors)],
                    method = "rpart",
                    trControl = train_control,

```

```

metric = "ROC")

# Resumo do modelo
print(cart_model)

# Avaliação do modelo no conjunto de teste
cart_probabilities <- predict(cart_model, testData, type = "prob")

# Curva ROC
cart_roc_curve <- roc(testData$ACTIVITY, cart_probabilities[, 2], levels = rev(levels(testData$ACTIVITY)))
print(auc(cart_roc_curve))
plot(cart_roc_curve, main = "Curva ROC - CART")

# Determinar ponto de corte ótimo
#cart_optimal_cutoff <- coords(cart_roc_curve, "best", ret = "threshold", best.method = "youden")[[1]]
#não encontrou ponto de corte válido
cart_optimal_cutoff <- median(cart_probabilities[, 2]) #manual, usei a mediana
print(paste("Ponto de corte ótimo (CART):", round(cart_optimal_cutoff, 4)))

# Converter probabilidades em previsões com ponto de corte ótimo
cart_predictions_optimal <- ifelse(cart_probabilities[, 2] >= cart_optimal_cutoff, "Positive", "Negative")
cart_predictions_optimal <- factor(cart_predictions_optimal, levels = levels(testData$ACTIVITY))

# Matriz de confusão com ponto de corte ótimo
cart_conf_matrix_optimal <- confusionMatrix(cart_predictions_optimal, testData$ACTIVITY)
print(cart_conf_matrix_optimal)

# Plotando a árvore de decisão
rpart.plot(cart_model$finalModel)

# Predição para a molécula alvo usando o ponto de corte ótimo
cart_target_prediction_prob <- predict(cart_model, newdata = target_predictors, type = "prob")
cart_target_prediction_optimal <- ifelse(cart_target_prediction_prob[, 2] >= cart_optimal_cutoff, "Positive", "Negative")

# Garantindo que o fator tenha os níveis corretos
cart_target_prediction_optimal <- factor(cart_target_prediction_optimal, levels = c("Negative", "Positive"))

cat("Predição para a molécula alvo (CART):", as.character(cart_target_prediction_optimal), "\n")
cat("Probabilidade da predição para a molécula alvo (CART):\n")
print(cart_target_prediction_prob)

# Extraíndo as métricas
precision <- cart_conf_matrix_optimal$byClass["Pos Pred Value"] # Precisão
sensitivity <- cart_conf_matrix_optimal$byClass["Sensitivity"] # Sensibilidade

# Calculando o F1-Score
f1_score_cart <- 2 * (precision * sensitivity) / (precision + sensitivity)
cat("F1-Score:", round(f1_score_cart, 4), "\n")

#=====
# Modelo XGBoost
#=====

# Definir hiperparâmetros do XGBoost
xgb_grid <- expand.grid(nrounds = 100,
                        max_depth = 3,

```

```

        eta = 0.3,
        gamma = 0,
        colsample_bytree = 1,
        min_child_weight = 1,
        subsample = 1)

# Treinamento do modelo
xgb_model <- train(ACTIVITY ~ .,
                  data = trainData[, c("ACTIVITY", predictors)],
                  method = "xgbTree",
                  trControl = train_control,
                  tuneGrid = xgb_grid,
                  metric = "ROC",
                  verbose = 0)

# Resumo do modelo
print(xgb_model)

# Avaliação do modelo no conjunto de teste
xgb_probabilities <- predict(xgb_model, testData, type = "prob")

# Curva ROC
xgb_roc_curve <- roc(testData$ACTIVITY, xgb_probabilities[, 2], levels = rev(levels(testData$ACTIVITY)))
print(auc(xgb_roc_curve))
plot(xgb_roc_curve, main = "Curva ROC - XGBoost")

# Determinar ponto de corte ótimo
xgb_optimal_cutoff <- coords(xgb_roc_curve, "best", ret = "threshold", best.method = "youden")[[1]]
#xgb_optimal_cutoff <- median(xgb_probabilities[, 2]) #manual, usei a mediana
print(paste("Ponto de corte ótimo (XGBoost):", round(xgb_optimal_cutoff, 4)))

# Converter probabilidades em previsões com ponto de corte ótimo
xgb_predictions_optimal <- ifelse(xgb_probabilities[, 2] >= xgb_optimal_cutoff, "Positive", "Negative")
xgb_predictions_optimal <- factor(xgb_predictions_optimal, levels = levels(testData$ACTIVITY))

# Matriz de confusão com ponto de corte ótimo
xgb_conf_matrix_optimal <- confusionMatrix(xgb_predictions_optimal, testData$ACTIVITY)
print(xgb_conf_matrix_optimal)

# Predição para a molécula alvo usando o ponto de corte ótimo
xgb_target_prediction_prob <- predict(xgb_model, newdata = target_predictors, type = "prob")
xgb_target_prediction_optimal <- ifelse(xgb_target_prediction_prob[, 2] >= xgb_optimal_cutoff, "Positive", "Negative")

# Garantindo que o fator tenha os níveis corretos
xgb_target_prediction_optimal <- factor(xgb_target_prediction_optimal, levels = c("Negative", "Positive"))

cat("Predição para a molécula alvo (XGBoost):", as.character(xgb_target_prediction_optimal), "\n")
cat("Probabilidade da predição para a molécula alvo (XGBoost):\n")
print(xgb_target_prediction_prob)

# Extrair as métricas
precision <- xgb_conf_matrix_optimal$byClass["Pos Pred Value"] # Precisão
sensitivity <- xgb_conf_matrix_optimal$byClass["Sensitivity"] # Sensibilidade

# Calculando o F1-Score
f1_score_xgb <- 2 * (precision * sensitivity) / (precision + sensitivity)

```

```

cat("F1-Score:", round(f1_score_xgb, 4), "\n")

#=====
# Resultados

# Consolidando métricas existentes
results <- data.frame(
  Model = c("Random Forest", "CART", "XGBoost"),
  AUC = c(auc(roc_curve), auc(cart_roc_curve), auc(xgb_roc_curve)),
  Accuracy = c(conf_matrix_optimal$overall["Accuracy"], cart_conf_matrix_optimal$overall["Accuracy"], xgb_conf_matrix_optimal$overall["Accuracy"]),
  Kappa = c(conf_matrix_optimal$overall["Kappa"], cart_conf_matrix_optimal$overall["Kappa"], xgb_conf_matrix_optimal$overall["Kappa"]),
  Sensitivity = c(conf_matrix_optimal$byClass["Sensitivity"], cart_conf_matrix_optimal$byClass["Sensitivity"], xgb_conf_matrix_optimal$byClass["Sensitivity"]),
  Specificity = c(conf_matrix_optimal$byClass["Specificity"], cart_conf_matrix_optimal$byClass["Specificity"], xgb_conf_matrix_optimal$byClass["Specificity"]),
  F1_Score = c(f1_score_rf, f1_score_cart, f1_score_xgb),
  Optimal_Cutoff = c(optimal_cutoff, cart_optimal_cutoff, xgb_optimal_cutoff),
  Probability_Positive = c(target_prediction_prob[1, "Positive"], cart_target_prediction_prob[1, "Positive"], xgb_target_prediction_prob[1, "Positive"]),
  Probability_Negative = c(target_prediction_prob[1, "Negative"], cart_target_prediction_prob[1, "Negative"], xgb_target_prediction_prob[1, "Negative"]),
  Cluster_Target = cluster_target,
  Mutagenicity_Prediction = c(as.character(target_prediction_optimal), as.character(cart_target_prediction_optimal), as.character(xgb_target_prediction_optimal))
)

# Salvando os resultados em um arquivo Excel
write.xlsx(list(Performance = results), "script_1_1omeprazol_results.xlsx", rowNames = FALSE)

```

## APÊNDICE G

```

#=====
# Descrição:
# Domínio de aplicação: filtra moléculas com MolWt <= 1000 e MolLogP <= 10
# Visualização das moléculas com PCA e Clusters;
# Determinação do espaço químico (nearest neighbors) - 500 moléculas;
# Predição usando ponto de corte padrão (0,5).
#=====

source("Pacotes.R") # Evoca o script de carregamento de pacotes

cat("\014") # Limpa o console
rm(list = ls(all = TRUE)) # Limpa a memória
options(scipen = 999) # Desabilita notação científica

# Carrega os dados
data_path <- "mutagenicity_m.xlsx"
target_path <- "omeprazol_descritores.xlsx"

data <- read_excel(data_path)
target <- read_excel(target_path)

# Domínio de aplicação: filtra moléculas com MolWt <= 1000 e MolLogP <= 10
data_filtered <- data %>%
  filter(MolWt <= 1000, MolLogP <= 10)

# Transforma a variável ACTIVITY em fator
data_filtered$ACTIVITY <- as.factor(data_filtered$ACTIVITY)

# Seleciona as variáveis relevantes
selected_vars <- c("ACTIVITY", "SPS", "MolWt", "NumValenceElectrons", "FpDensityMorgan1",
  "FpDensityMorgan2", "FpDensityMorgan3", "BCUT2D_MWHI", "BCUT2D_MWLOW",
  "BCUT2D_CHGHI", "BCUT2D_CHGLO", "BCUT2D_LOGPHI", "BCUT2D_LOGPLOW",
  "BCUT2D_MRHI", "BCUT2D_MRLOW", "AvgIpc", "BalabanJ", "BertzCT", "Chi0",
  "HallKierAlpha", "Kappa2", "Kappa3", "LabuteASA", "PEOE_VSA7",
  "SMR_VSA10", "SlogP_VSA2", "TPSA", "FractionCSP3", "NumAliphaticRings",
  "NumAromaticRings", "NumHAcceptors", "NumHDonors", "NumRotatableBonds",
  "NumSaturatedRings", "RingCount", "MolLogP", "MolMR")

data_filtered <- data_filtered %>%
  dplyr::select(all_of(selected_vars))

# Remove valores ausentes
data_filtered <- na.omit(data_filtered)

# Seleciona variáveis preditoras
predictors <- setdiff(selected_vars, "ACTIVITY")

# Normaliza descritores moleculares
data_predictors <- scale(data_filtered[, predictors])

# PCA para visualização
pca <- prcomp(data_predictors, center = TRUE, scale. = TRUE)

# Visualizando a variância explicada pela PCA

```

```

fviz_eig(pca, addlabels = TRUE, ylim = c(0, 50))

# Transforma os dados para o espaço PCA
data_pca <- data.frame(pca$x)

# Agrupamento K-means
set.seed(123)
kmeans_res <- kmeans(data_pca, centers = 5, nstart = 25, iter.max = 100)

# Adicionando os clusters aos dados
data_filtered$Cluster <- as.factor(kmeans_res$cluster)

# Adicionando as coordenadas PCA
data_filtered$PC1 <- data_pca$PC1
data_filtered$PC2 <- data_pca$PC2

# Plotando os clusters
ggplot(data_filtered, aes(x = PC1, y = PC2, color = Cluster)) +
  geom_point() +
  theme_minimal() +
  labs(title = "Clusters de Moléculas no Espaço PCA")

# Normaliza os descritores da molécula alvo com os mesmos parâmetros da PCA
target_predictors <- target[, predictors]
target_predictors <- scale(target_predictors, center = attr(data_predictors, "scaled:center"), scale =
  attr(data_predictors, "scaled:scale"))

# Projeção da molécula alvo no espaço PCA
target_pca <- predict(pca, newdata = target_predictors)

# Calcula a distância da molécula alvo para cada molécula no dataset
distances <- apply(data_pca[, 1:2], 1, function(x) sqrt(sum((x - target_pca[1:2])^2)))

# Adiciona as distâncias ao dataframe
data_filtered$DistanceToTarget <- distances

# Seleciona as 500 moléculas mais próximas
closest_molecules <- data_filtered %>%
  arrange(DistanceToTarget) %>%
  head(500)

# Visualiza as moléculas mais próximas
ggplot(closest_molecules, aes(x = PC1, y = PC2, color = ACTIVITY)) +
  geom_point() +
  geom_point(aes(x = target_pca[1], y = target_pca[2]), color = "blue", size = 6, shape = 18) + #
  Molécula alvo
  theme_minimal() +
  labs(title = "Moléculas mais próximas da Molécula Alvo")

# Verifica o balanceamento dos dados
table(closest_molecules$ACTIVITY)
prop.table(table(closest_molecules$ACTIVITY)) # Proporção de cada classe

# Cria a tabela de proporções
prop_data <- as.data.frame(prop.table(table(closest_molecules$ACTIVITY)))

# Renomeia as colunas
colnames(prop_data) <- c("ACTIVITY", "Proporção")

```

```

# Cria gráfico de barras para visualizar balanceamento
ggplot(prop_data, aes(x = ACTIVITY, y = Proporção, fill = ACTIVITY)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = scales::percent(Proporção, accuracy = 0.1)),
    vjust = 1.5, color = "white", size = 5) + # Ajuste do texto dentro das barras
labs(title = "Proporção dos resultados de mutagenicidade",
  x = "ACTIVITY",
  y = "Proporção") +
theme_minimal() +
theme(legend.position = "none") # Remove legenda externa

=====
# Modelo de predição de mutagenicidade

# Converte ACTIVITY de 0/1 para "Negative"/"Positive": foi necessário para calc. curva ROC
closest_molecules$ACTIVITY <- factor(closest_molecules$ACTIVITY,
  levels = c(0, 1),
  labels = c("Negative", "Positive"))

# Divide os dados em treino e teste
set.seed(123)
trainIndex <- createDataPartition(closest_molecules$ACTIVITY, p = 0.8, list = FALSE)
trainData <- closest_molecules[trainIndex, ]
testData <- closest_molecules[-trainIndex, ]

# Validação cruzada
train_control <- trainControl(method = "cv", number = 10, classProbs = TRUE, summaryFunction =
twoClassSummary)

=====
# Modelo Random Forest
=====
rf_model <- train(ACTIVITY ~ .,
  data = trainData[, c("ACTIVITY", predictors)],
  method = "rf",
  trControl = train_control,
  importance = TRUE,
  metric = "ROC")

# Resumo do modelo
print(rf_model)

# Avaliação do modelo no conjunto de teste
rf_predictions <- predict(rf_model, testData)
rf_probabilities <- predict(rf_model, testData, type = "prob")

# Matriz de confusão
conf_matrix <- confusionMatrix(rf_predictions, testData$ACTIVITY)
print(conf_matrix)

# Plotando a importância das variáveis
varImpPlot(rf_model$finalModel)

# Predição para a molécula alvo
target_prediction <- predict(rf_model, newdata = target_predictors)
print(paste("Predição para a molécula alvo:", target_prediction))

# Probabilidade da predição

```

```

target_prediction_prob <- predict(rf_model, newdata = target_predictors, type = "prob")
print("Probabilidade da predição para a molécula alvo:")
print(target_prediction_prob)

# Curva ROC
roc_curve <- roc(testData$ACTIVITY, rf_probabilities[, 2], levels = rev(levels(testData$ACTIVITY)))
print(auc(roc_curve))
plot(roc_curve, main = "Curva ROC - Random Forest")

# Extrair as métricas
precision <- conf_matrix$byClass["Pos Pred Value"] # Precisão
sensitivity <- conf_matrix$byClass["Sensitivity"] # Sensibilidade

# Calculando o F1-Score
f1_score_rf <- 2 * (precision * sensitivity) / (precision + sensitivity)
cat("F1-Score:", round(f1_score_rf, 4), "\n")

#=====
# Modelo CAR
#=====
car_model <- train(ACTIVITY ~ .,
  data = trainData[, c("ACTIVITY", predictors)],
  method = "rpart",
  trControl = train_control,
  metric = "ROC")

# Resumo do modelo
print(car_model)

# Avaliação do modelo no conjunto de teste
car_predictions <- predict(car_model, testData)
car_probabilities <- predict(car_model, testData, type = "prob")

# Matriz de confusão
car_conf_matrix <- confusionMatrix(car_predictions, testData$ACTIVITY)
print(car_conf_matrix)

# Plotando a árvore de decisão
rpart.plot(car_model$finalModel)

# Predição para a molécula alvo
car_target_prediction <- predict(car_model, newdata = target_predictors)
print(paste("Predição para a molécula alvo:", car_target_prediction))

# Probabilidade da predição
car_target_prediction_prob <- predict(car_model, newdata = target_predictors, type = "prob")
print("Probabilidade da predição para a molécula alvo:")
print(car_target_prediction_prob)

# Curva ROC
car_roc_curve <- roc(testData$ACTIVITY, car_probabilities[, 2], levels = rev(levels(testData$ACTIV-
ITY)))
print(auc(car_roc_curve))
plot(car_roc_curve, main = "Curva ROC - CAR")

# Extrair as métricas
precision <- car_conf_matrix$byClass["Pos Pred Value"] # Precisão
sensitivity <- car_conf_matrix$byClass["Sensitivity"] # Sensibilidade

```

```

# Calculando o F1-Score
f1_score_cart <- 2 * (precision * sensitivity) / (precision + sensitivity)
cat("F1-Score:", round(f1_score_cart, 4), "\n")

#=====
# Modelo XGBoost
#=====
xgb_grid <- expand.grid(nrounds = 100,
                      max_depth = 3,
                      eta = 0.3,
                      gamma = 0,
                      colsample_bytree = 1,
                      min_child_weight = 1,
                      subsample = 1)

xgb_model <- train(ACTIVITY ~ .,
                  data = trainData[, c("ACTIVITY", predictors)],
                  method = "xgbTree",
                  trControl = train_control,
                  tuneGrid = xgb_grid,
                  metric = "ROC",
                  verbose = 0)

# Resumo do modelo
print(xgb_model)

# Avaliação do modelo no conjunto de teste
xgb_predictions <- predict(xgb_model, testData)
xgb_probabilities <- predict(xgb_model, testData, type = "prob")

# Matriz de confusão
xgb_conf_matrix <- confusionMatrix(xgb_predictions, testData$ACTIVITY)
print(xgb_conf_matrix)

# Predição para a molécula alvo
xgb_target_prediction <- predict(xgb_model, newdata = target_predictors)
print(paste("Predição para a molécula alvo:", xgb_target_prediction))

# Probabilidade da predição
xgb_target_prediction_prob <- predict(xgb_model, newdata = target_predictors, type = "prob")
print("Probabilidade da predição para a molécula alvo:")
print(xgb_target_prediction_prob)

# Curva ROC
xgb_roc_curve <- roc(testData$ACTIVITY, xgb_probabilities[, 2], levels = rev(levels(testData$ACTIVITY)))
print(auc(xgb_roc_curve))
plot(xgb_roc_curve, main = "Curva ROC - XGBoost")

# Extraíndo as métricas
precision <- xgb_conf_matrix$byClass["Pos Pred Value"] # Precisão
sensitivity <- xgb_conf_matrix$byClass["Sensitivity"] # Sensibilidade

# Calculando o F1-Score
f1_score_xgb <- 2 * (precision * sensitivity) / (precision + sensitivity)
cat("F1-Score:", round(f1_score_xgb, 4), "\n")

#=====
# Resultados

```

```

# Consolidando métricas existentes
results <- data.frame(
  Model = c("Random Forest", "CART", "XGBoost"),
  AUC = c(auc(roc_curve), auc(car_roc_curve), auc(xgb_roc_curve)),
  Accuracy = c(conf_matrix$overall["Accuracy"], car_conf_matrix$overall["Accuracy"], xgb_conf_ma-
    trix$overall["Accuracy"]),
  Kappa = c(conf_matrix$overall["Kappa"], car_conf_matrix$overall["Kappa"], xgb_conf_matrix$over-
    all["Kappa"]),
  Sensitivity = c(conf_matrix$byClass["Sensitivity"], car_conf_matrix$byClass["Sensitivity"],
    xgb_conf_matrix$byClass["Sensitivity"]),
  Specificity = c(conf_matrix$byClass["Specificity"], car_conf_matrix$byClass["Specificity"],
    xgb_conf_matrix$byClass["Specificity"]),
  F1_Score = c(f1_score_rf, f1_score_cart, f1_score_xgb),
  Probability_Positive = c(target_prediction_prob[1, "Positive"], car_target_prediction_prob[1, "Posi-
    tive"], xgb_target_prediction_prob[1, "Positive"]),
  Probability_Negative = c(target_prediction_prob[1, "Negative"], car_target_prediction_prob[1, "Neg-
    ative"], xgb_target_prediction_prob[1, "Negative"]),
  Mutagenicity_Prediction = c(as.character(target_prediction), as.character(car_target_prediction),
    as.character(xgb_target_prediction))
)

# Salvando os resultados em um arquivo Excel
write.xlsx(list(Performance = results), "script_2omeprazol_results.xlsx", rowNames = FALSE)

```

## APÊNDICE H

```

#=====
# Descrição:
# Domínio de aplicação: filtra moléculas com MolWt <= 1000 e MolLogP <= 10
# Visualização das moléculas com PCA e Clusters;
# Determinação do espaço químico (nearest neighbors) - 500 moléculas;
# Predição usando ponto de corte ótimo.
#=====

source("Pacotes.R") # Evoca o script de carregamento de pacotes

cat("\014") # Limpa o console
rm(list = ls(all = TRUE)) # Limpa a memória
options(scipen = 999) # Desabilita notação científica

# Carrega os dados
data_path <- "mutagenicity_m.xlsx"
target_path <- "omeprazol_descritores.xlsx"

data <- read_excel(data_path)
target <- read_excel(target_path)

# Domínio de aplicação: filtra moléculas com MolWt <= 1000 e MolLogP <= 10
data_filtered <- data %>%
  filter(MolWt <= 1000, MolLogP <= 10)

# Transforma a variável ACTIVITY em fator
data_filtered$ACTIVITY <- as.factor(data_filtered$ACTIVITY)

# Seleciona as variáveis relevantes
selected_vars <- c("ACTIVITY", "SPS", "MolWt", "NumValenceElectrons", "FpDensityMorgan1",
  "FpDensityMorgan2", "FpDensityMorgan3", "BCUT2D_MWHI", "BCUT2D_MWLOW",
  "BCUT2D_CHGHI", "BCUT2D_CHGLO", "BCUT2D_LOGPHI", "BCUT2D_LOGPLOW",
  "BCUT2D_MRHI", "BCUT2D_MRLOW", "AvgIpc", "BalabanJ", "BertzCT", "Chi0",
  "HallKierAlpha", "Kappa2", "Kappa3", "LabuteASA", "PEOE_VSA7",
  "SMR_VSA10", "SlogP_VSA2", "TPSA", "FractionCSP3", "NumAliphaticRings",
  "NumAromaticRings", "NumHAcceptors", "NumHDonors", "NumRotatableBonds",
  "NumSaturatedRings", "RingCount", "MolLogP", "MolMR")

data_filtered <- data_filtered %>%
  dplyr::select(all_of(selected_vars))

# Remove valores ausentes
data_filtered <- na.omit(data_filtered)

# Seleciona variáveis preditoras
predictors <- setdiff(selected_vars, "ACTIVITY")

# Normaliza descritores moleculares
data_predictors <- scale(data_filtered[, predictors])

# PCA para visualização
pca <- prcomp(data_predictors, center = TRUE, scale. = TRUE)

# Visualizando a variância explicada pela PCA

```

```

fviz_eig(pca, addlabels = TRUE, ylim = c(0, 50))

# Transforma os dados para o espaço PCA
data_pca <- data.frame(pca$x)

# Agrupamento K-means
set.seed(123)
kmeans_res <- kmeans(data_pca, centers = 5, nstart = 25, iter.max = 100)

# Adicionando os clusters aos dados
data_filtered$Cluster <- as.factor(kmeans_res$cluster)

# Adicionando as coordenadas PCA
data_filtered$PC1 <- data_pca$PC1
data_filtered$PC2 <- data_pca$PC2

# Plotando os clusters
ggplot(data_filtered, aes(x = PC1, y = PC2, color = Cluster)) +
  geom_point() +
  theme_minimal() +
  labs(title = "Clusters de Moléculas no Espaço PCA")

# Normaliza os descritores da molécula alvo com os mesmos parâmetros da PCA
target_predictors <- target[, predictors]
target_predictors <- scale(target_predictors, center = attr(data_predictors, "scaled:center"), scale =
  attr(data_predictors, "scaled:scale"))

# Projeção da molécula alvo no espaço PCA
target_pca <- predict(pca, newdata = target_predictors)

# Calcula a distância da molécula alvo para cada molécula no dataset
distances <- apply(data_pca[, 1:2], 1, function(x) sqrt(sum((x - target_pca[1:2])^2)))

# Adiciona distâncias ao dataframe
data_filtered$DistanceToTarget <- distances

# Seleciona as 500 moléculas mais próximas
closest_molecules <- data_filtered %>%
  arrange(DistanceToTarget) %>%
  head(500)

# Visualiza as moléculas mais próximas
ggplot(closest_molecules, aes(x = PC1, y = PC2, color = ACTIVITY)) +
  geom_point() +
  geom_point(aes(x = target_pca[1], y = target_pca[2]), color = "blue", size = 6, shape = 18) + #
  Molécula alvo
  theme_minimal() +
  labs(title = "Moléculas mais próximas da Molécula Alvo")

# Verifica o balanceamento dos dados
table(closest_molecules$ACTIVITY)
prop.table(table(closest_molecules$ACTIVITY)) # Proporção de cada classe

# Cria a tabela de proporções
prop_data <- as.data.frame(prop.table(table(closest_molecules$ACTIVITY)))

# Renomeia as colunas
colnames(prop_data) <- c("ACTIVITY", "Proporção")

```

```

# Cria gráfico de barras para visualizar balanceamento
ggplot(prop_data, aes(x = ACTIVITY, y = Proporção, fill = ACTIVITY)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = scales::percent(Proporção, accuracy = 0.1)),
    vjust = 1.5, color = "white", size = 5) + # Ajuste do texto dentro das barras
labs(title = "Proporção dos resultados de mutagenicidade",
  x = "ACTIVITY",
  y = "Proporção") +
theme_minimal() +
theme(legend.position = "none") # Remove legenda externa

=====
# Modelo de predição de mutagenicidade

# Converte ACTIVITY de 0/1 para "Negative"/"Positive": foi necessário para calc. curva ROC
closest_molecules$ACTIVITY <- factor(closest_molecules$ACTIVITY,
  levels = c(0, 1),
  labels = c("Negative", "Positive"))

# Divide os dados em treino e teste
set.seed(123)
trainIndex <- createDataPartition(closest_molecules$ACTIVITY, p = 0.8, list = FALSE)
trainData <- closest_molecules[trainIndex, ]
testData <- closest_molecules[-trainIndex, ]

# Validação cruzada
train_control <- trainControl(method = "cv", number = 10, classProbs = TRUE, summaryFunction =
twoClassSummary)

=====
# Modelo Random Forest
=====
rf_model <- train(ACTIVITY ~ .,
  data = trainData[, c("ACTIVITY", predictors)],
  method = "rf",
  trControl = train_control,
  importance = TRUE,
  metric = "ROC")

# Resumo do modelo
print(rf_model)

# Plotando a importância das variáveis
varImpPlot(rf_model$finalModel)

# Avaliação do modelo no conjunto de teste
rf_probabilities <- predict(rf_model, testData, type = "prob")

# Curva ROC
roc_curve <- roc(testData$ACTIVITY, rf_probabilities[, 2], levels = rev(levels(testData$ACTIVITY)))
print(auc(roc_curve))
plot(roc_curve, main = "Curva ROC - Random Forest")

# Determina o ponto de corte ótimo
optimal_cutoff <- as.numeric(coords(roc_curve, "best", ret = "threshold", best.method = "youden"))
#optimal_cutoff <- median(rf_probabilities[, 2]) #manual, usei a mediana
print(paste("Ponto de corte ótimo:", round(optimal_cutoff, 4)))

```

```

# Converte probabilidades em previsões com ponto de corte ótimo
rf_predictions_optimal <- ifelse(rf_probabilities[, 2] >= optimal_cutoff, "Positive", "Negative")
rf_predictions_optimal <- factor(rf_predictions_optimal, levels = levels(testData$ACTIVITY))

# Matriz de confusão com ponto de corte ótimo
conf_matrix_optimal <- confusionMatrix(rf_predictions_optimal, testData$ACTIVITY)
print(conf_matrix_optimal)

# Predição para a molécula alvo usando o ponto de corte ótimo
target_prediction_prob <- predict(rf_model, newdata = target_predictors, type = "prob")
target_prediction_optimal <- ifelse(target_prediction_prob[, 2] >= optimal_cutoff, "Positive", "Negative")

# Garantindo que o fator tenha os níveis corretos
target_prediction_optimal <- factor(target_prediction_optimal, levels = c("Negative", "Positive"))

cat("Predição para a molécula alvo:", as.character(target_prediction_optimal), "\n")
cat("Probabilidade da predição para a molécula alvo:\n")
print(target_prediction_prob)

# Extraíndo as métricas
precision <- conf_matrix_optimal$byClass["Pos Pred Value"] # Precisão
sensitivity <- conf_matrix_optimal$byClass["Sensitivity"] # Sensibilidade

# Calculando o F1-Score
f1_score_rf <- 2 * (precision * sensitivity) / (precision + sensitivity)
cat("F1-Score:", round(f1_score_rf, 4), "\n")

#=====
# Modelo CART
#=====
cart_model <- train(ACTIVITY ~ .,
                    data = trainData[, c("ACTIVITY", predictors)],
                    method = "rpart",
                    trControl = train_control,
                    metric = "ROC")

# Resumo do modelo
print(cart_model)

# Avaliação do modelo no conjunto de teste
cart_probabilities <- predict(cart_model, testData, type = "prob")

# Curva ROC
cart_roc_curve <- roc(testData$ACTIVITY, cart_probabilities[, 2], levels = rev(levels(testData$ACTIVITY)))
print(auc(cart_roc_curve))
plot(cart_roc_curve, main = "Curva ROC - CART")

# Determina o ponto de corte ótimo
#cart_optimal_cutoff <- coords(cart_roc_curve, "best", ret = "threshold", best.method = "youden")[[1]]
#não encontrou ponto de corte válido
cart_optimal_cutoff <- median(cart_probabilities[, 2]) #manual, usei a mediana
print(paste("Ponto de corte ótimo (CART):", round(cart_optimal_cutoff, 4)))

# Converte probabilidades em previsões com ponto de corte ótimo
cart_predictions_optimal <- ifelse(cart_probabilities[, 2] >= cart_optimal_cutoff, "Positive", "Negative")
cart_predictions_optimal <- factor(cart_predictions_optimal, levels = levels(testData$ACTIVITY))

```

```

# Matriz de confusão com ponto de corte ótimo
cart_conf_matrix_optimal <- confusionMatrix(cart_predictions_optimal, testData$ACTIVITY)
print(cart_conf_matrix_optimal)

# Plotando a árvore de decisão
rpart.plot(cart_model$finalModel)

# Predição para a molécula alvo usando o ponto de corte ótimo
cart_target_prediction_prob <- predict(cart_model, newdata = target_predictors, type = "prob")
cart_target_prediction_optimal <- ifelse(cart_target_prediction_prob[, 2] >= cart_optimal_cutoff, "Positive", "Negative")

# Garantindo que o fator tenha os níveis corretos
cart_target_prediction_optimal <- factor(cart_target_prediction_optimal, levels = c("Negative", "Positive"))

cat("Predição para a molécula alvo (CART):", as.character(cart_target_prediction_optimal), "\n")
cat("Probabilidade da predição para a molécula alvo (CART):\n")
print(cart_target_prediction_prob)

# Extraíndo as métricas
precision <- cart_conf_matrix_optimal$byClass["Pos Pred Value"] # Precisão
sensitivity <- cart_conf_matrix_optimal$byClass["Sensitivity"] # Sensibilidade

# Calculando o F1-Score
f1_score_cart <- 2 * (precision * sensitivity) / (precision + sensitivity)
cat("F1-Score:", round(f1_score_cart, 4), "\n")

#=====
# Modelo XGBoost
#=====

# Definir hiperparâmetros do XGBoost
xgb_grid <- expand.grid(nrounds = 100,
                      max_depth = 3,
                      eta = 0.3,
                      gamma = 0,
                      colsample_bytree = 1,
                      min_child_weight = 1,
                      subsample = 1)

# Treinamento do modelo
xgb_model <- train(ACTIVITY ~ .,
                  data = trainData[, c("ACTIVITY", predictors)],
                  method = "xgbTree",
                  trControl = train_control,
                  tuneGrid = xgb_grid,
                  metric = "ROC",
                  verbose = 0)

# Resumo do modelo
print(xgb_model)

# Avaliação do modelo no conjunto de teste
xgb_probabilities <- predict(xgb_model, testData, type = "prob")

# Curva ROC
xgb_roc_curve <- roc(testData$ACTIVITY, xgb_probabilities[, 2], levels = rev(levels(testData$ACTIVITY)))

```

```

print(auc(xgb_roc_curve))
plot(xgb_roc_curve, main = "Curva ROC - XGBoost")

# Determina o ponto de corte ótimo
xgb_optimal_cutoff <- coords(xgb_roc_curve, "best", ret = "threshold", best.method = "youden")[[1]]
#xgb_optimal_cutoff <- median(xgb_probabilities[, 2]) #manual, usei a mediana
print(paste("Ponto de corte ótimo (XGBoost):", round(xgb_optimal_cutoff, 4)))

# Converte probabilidades em previsões com ponto de corte ótimo
xgb_predictions_optimal <- ifelse(xgb_probabilities[, 2] >= xgb_optimal_cutoff, "Positive", "Negative")
xgb_predictions_optimal <- factor(xgb_predictions_optimal, levels = levels(testData$ACTIVITY))

# Matriz de confusão com ponto de corte ótimo
xgb_conf_matrix_optimal <- confusionMatrix(xgb_predictions_optimal, testData$ACTIVITY)
print(xgb_conf_matrix_optimal)

# Predição para a molécula alvo usando o ponto de corte ótimo
xgb_target_prediction_prob <- predict(xgb_model, newdata = target_predictors, type = "prob")
xgb_target_prediction_optimal <- ifelse(xgb_target_prediction_prob[, 2] >= xgb_optimal_cutoff, "Positive", "Negative")

# Garantindo que o fator tenha os níveis corretos
xgb_target_prediction_optimal <- factor(xgb_target_prediction_optimal, levels = c("Negative", "Positive"))

cat("Predição para a molécula alvo (XGBoost):", as.character(xgb_target_prediction_optimal), "\n")
cat("Probabilidade da predição para a molécula alvo (XGBoost):\n")
print(xgb_target_prediction_prob)

# Extraíndo as métricas
precision <- xgb_conf_matrix_optimal$byClass["Pos Pred Value"] # Precisão
sensitivity <- xgb_conf_matrix_optimal$byClass["Sensitivity"] # Sensibilidade

# Calculando o F1-Score
f1_score_xgb <- 2 * (precision * sensitivity) / (precision + sensitivity)
cat("F1-Score:", round(f1_score_xgb, 4), "\n")

#=====
# Resultados

# Consolidando métricas existentes
results <- data.frame(
  Model = c("Random Forest", "CART", "XGBoost"),
  AUC = c(auc(roc_curve), auc(cart_roc_curve), auc(xgb_roc_curve)),
  Accuracy = c(conf_matrix_optimal$overall["Accuracy"], cart_conf_matrix_optimal$overall["Accuracy"], xgb_conf_matrix_optimal$overall["Accuracy"]),
  Kappa = c(conf_matrix_optimal$overall["Kappa"], cart_conf_matrix_optimal$overall["Kappa"], xgb_conf_matrix_optimal$overall["Kappa"]),
  Sensitivity = c(conf_matrix_optimal$byClass["Sensitivity"], cart_conf_matrix_optimal$byClass["Sensitivity"], xgb_conf_matrix_optimal$byClass["Sensitivity"]),
  Specificity = c(conf_matrix_optimal$byClass["Specificity"], cart_conf_matrix_optimal$byClass["Specificity"], xgb_conf_matrix_optimal$byClass["Specificity"]),
  F1_Score = c(f1_score_rf, f1_score_cart, f1_score_xgb),
  Optimal_Cutoff = c(optimal_cutoff, cart_optimal_cutoff, xgb_optimal_cutoff),
  Probability_Positive = c(target_prediction_prob[1, "Positive"], cart_target_prediction_prob[1, "Positive"], xgb_target_prediction_prob[1, "Positive"]),
  Probability_Negative = c(target_prediction_prob[1, "Negative"], cart_target_prediction_prob[1, "Negative"], xgb_target_prediction_prob[1, "Negative"]),

```

```
Mutagenicity_Prediction = c(as.character(target_prediction_optimal), as.character(cart_target_prediction_optimal), as.character(xgb_target_prediction_optimal))  
)  
  
# Salvando os resultados em um arquivo Excel  
write.xlsx(list(Performance = results), "script_2_2omeprazol_results.xlsx", rowNames = FALSE)
```

## APÊNDICE I

```
#Pacotes
```

```
if(!require(MASS)){install.packages("MASS"); require(MASS)}
if(!require(openxlsx)){install.packages("openxlsx"); require(openxlsx)}
if(!require(car)){install.packages("car"); require(car)}
if(!require(ggplot2)){install.packages("ggplot2"); require(ggplot2)}
if(!require(readxl)){install.packages("readxl"); require(readxl)}
if(!require(packHV)){install.packages("packHV"); require(packHV)}
if(!require(corrplot)){install.packages("corrplot"); require(corrplot)}
if(!require(jtools)){install.packages("jtools"); require(jtools)}
if(!require(exploreR)){install.packages("exploreR"); require(exploreR)}
if(!require(dplyr)){install.packages("dplyr"); require(dplyr)}
if(!require(randomForest)){install.packages("randomForest"); require(randomForest)}
if(!require(lmtest)){install.packages("lmtest"); require(lmtest)}
if(!require(olsrr)){install.packages("olsrr"); require(olsrr)}
if(!require(EnvStats)){install.packages("EnvStats"); require(EnvStats)}
if(!require(pROC)){install.packages("pROC"); require(pROC)}
if(!require(mlogit)){install.packages("mlogit"); require(mlogit)}
if(!require(caret)){install.packages("caret"); require(caret)}
if(!require(rpart)){install.packages("rpart"); require(rpart)}
if(!require(rpart.plot)){install.packages("rpart.plot"); require(rpart.plot)}
if(!require(party)){install.packages("party"); require(party)}
if (!tinytex::is_tinytex()) tinytex::install_tinytex()
if(!require(GGally)){install.packages("GGally"); require(GGally)}
if(!require(ggspatial)){install.packages("ggspatial"); require(ggspatial)}
if(!require(ggsn)){install.packages("ggsn"); require(ggsn)}
if(!require(raster)){install.packages("raster"); require(raster)}
if(!require(rgdal)){install.packages("rgdal"); require(rgdal)}
if(!require(sf)){install.packages("sf"); require(sf)}
if(!require(sp)){install.packages("sp"); require(sp)}
if(!require(tidyverse)){install.packages("tidyverse"); require(tidyverse)}
if(!require(FactoMineR)){install.packages("FactoMineR"); require(FactoMineR)}
if(!require(factoextra)){install.packages("factoextra"); require(factoextra)}
if(!require(cluster)){install.packages("cluster"); require(cluster)}
if(!require(prcomp)){install.packages("prcomp"); require(prcomp)}
if(!require(ggdendro)){install.packages("ggdendro"); require(ggdendro)}
if(!require(xgboost)){install.packages("xgboost"); require(xgboost)}
if(!require(writexl)){install.packages("writexl"); require(writexl)}
```