Universidade Federal de Santa Catarina
Campus Reitor João David Ferreira Lima
Departamento de Informática e Estatística

Luis Henrique Goulart Stemmer

DESENVOLVIMENTO DE FERRAMENTA DE ANÁLISE E MINERAÇÃO DE DADOS DE SUICÍDIO DO DATASUS

Florianópolis
2023/1

Luis Henrique Goulart Stemmer

DESENVOLVIMENTO DE FERRAMENTA DE ANÁLISE E MINERAÇÃO DE DADOS DE SUICÍDIO DO DATASUS

Trabalho de Conclusão de Curso apresentado à Universidade Federal de Santa Catarina como parte dos requisitos necessários para a obtenção do título de Bacharel em Ciência da Computação.
Orientador: Prof. Dr. Mateus Grellert

Universidade Federal de Santa Catarina
Campus Reitor João David Ferreira Lima
Departamento de Informática e Estatística

Florianópolis
2023/1

Luis Henrique Goulart Stemmer

DESENVOLVIMENTO DE FERRAMENTA DE ANÁLISE E MINERAÇÃO DE DADOS DE SUICÍDIO DO DATASUS

Trabalho de Conclusão de Curso apresentado à Universidade Federal de Santa Catarina como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Comissão Examinadora

Prof. Dr. Mateus Grellert
Universidade Federal de Santa Catarina
Orientador

Prof. Dr. Jônata Tyska Carvalho
Universidade Federal de Santa Catarina

Profa. Dra. Carina Friedrich Dorneles
Universidade Federal de Santa Catarina

Florianópolis, 11 de julho de 2023

Resumo

O suicídio é a causa de mais de 700 mil mortes por ano ao redor do mundo. A prevenção do suicídio é um tema bastante complexo em razão da grande quantidade de fatores difíceis de definir e mensurar, com graus de correlação incertos e inconsistentes, desde comportamento e sociabilidade de um indivíduo, relações interpessoais em ambiente familiar, escolar e de trabalho, à infraestrutura hospitalar e indicadores demográficos e socioeconômicos do local onde vive. Técnicas de Inteligência Artificial, acompanhadas de bancos de dados bem estruturados e preparados, têm se mostrado úteis em múltiplas áreas de conhecimento – especialmente na medicina. A grande capacidade de processamento que fornecem, além de viabilizar novas análises, pode auxiliar profissionais da saúde a obter diagnósticos mais eficazes e precisos, possibilitando que tratamentos corretos sejam iniciados em tempo hábil. Este projeto tem como objetivo principal o desenvolvimento de uma ferramenta que, através de uma interface gráfica intuitiva, facilite o processo de análise e mineração de dados de suicídio no Brasil, disponibilizados pelo SUS. A ferramenta realiza a coleta e o pré-processamento dos dados, oferece uma variedade de funcionalidades para cálculo de estatísticas descritivas e visualização de gráficos, além de possibilitar a aplicação de algoritmos de *clustering* hierárquico e a avaliação da qualidade dos *clusters* através da métrica *silhouette score*. Resultados experimentais apontam que, utilizando o método *complete linkage* para definição de *clusters*, o melhor agrupamento acontece com uma distância máxima de 60 unidades entre elementos de cada *cluster*, que obteve um *silhouette score* de 0.52. Este trabalho busca contribuir com a comunidade de especialistas em saúde mental e gestores responsáveis pela criação de políticas públicas, proporcionando uma ferramenta útil para entender melhor um fenômeno tão complexo como o suicídio.

Palavras-Chave: Suicídio; Análise de Dados; Mineração de Dados; Aprendizado de Máquina; *Clustering*

Abstract

Suicide is the cause of over 700 thousand deaths every year around the world. Suicide prevention is a very complex subject due to the large quantity of possible factors that are hard to define and measure, with uncertain and inconsistent correlation levels, that go from an individual's behaviour and sociability, interpersonal relationships in family, school and work environments, to hospital infrastructure and demographic indicators in their place of residence. Artificial Intelligence techniques, allied to well structured and prepared data, have been helpful in multiple areas of knowledge – especially in medicine. The great processing capacity they provide not only enables new analysis but also assist health professionals in achieving effective and accurate diagnoses, allowing correct treatment to be initiated in time. The main objective of this project is the development of a tool that, through an intuitive graphic interface, facilitates the process of analysis and mining of suicide data in Brazil, provided by SUS. The tool performs data collection and preprocessing and offers a variety of functionalities to calculate descriptive statistics and visualize graphs, in addition to enabling the application of hierarchical clustering algorithms and the evaluation of clusters.

Keywords: Suicide; Data Analysis; Data Mining; Machine Learning; Clustering

Lista de figuras

Lista de tabelas

Lista de Siglas e Abreviaturas

API	*Application Programming Interface*
CNES	*Cadastro Nacional de Estabelecimentos de Saúde*
CRISP-DM	*Cross Industry Standard Process for Data Mining*
DATASUS	*Departamento de Informática do Sistema Único de Saúde*
IA	*Inteligência Artificial*
ML	*Machine Learning*
OMS	*Organização Mundial da Saúde*
SINAN	*Sistema de Informação de Agravos de Notificação*
SIM	*Sistema de Informação sobre Mortalidade*
SSQ	*Sum of Squared Distances*
SUS	*Sistema Único de Saúde*
TCC	*Trabalho de Conclusão de Curso*

Sumário

1 Introdução

O suicídio está entre as pricipais causas de mortes evitáveis no Brasil e no mundo há anos (World Health Organization, 2021). Segundo a Organização Mundial da Saúde (OMS), é a causa de mais de 700.000 mortes por ano - uma a cada 40 segundos. Em 2019, o suicídio representou 1,3% do total de óbitos no mundo e 1,7% no continente americano, sendo que 77% das mortes ocorreram em países de média ou baixa renda. No Brasil, o suicídio foi a terceira principal causa de morte de jovens brasileiros entre 15 e 29 anos e o país figura consistentemente entre as 10 nações com maior número absoluto de casos (World Health Organization, 2019; World Health Organization, 2021).

O cenário torna-se ainda mais preocupante após a eclosão da pandemia de COVID-19, período em que observou-se uma intensa deterioração dos quadros de saúde mental. Ainda de acordo com a OMS(World Health Organization, 2022), em apenas um ano desde o início da pandemia houve um aumento de 26% e 28% em casos de ansiedade e depressão, respectivamente; condições bastante associadas à ideação suicida.

A complexidade das interações entre fatores agravantes do fenômeno do suicídio (HEERINGEN; MANN, 2014), somada ao estigma relacionado não só a esse fenômeno, mas também a quaisquer transtornos mentais e à psiquiatria como um todo, dificulta a identificação e o tratamento preventivo de indivíduos com tendências suicidas. Nesse contexto, técnicas de aprendizado de máquina são uma ferramenta poderosa, já que possuem ótima capacidade de reconhecimento de padrões e viabilizam análises eficientes de grandes quantidades de dados.

A fim de melhorar a disponibilidade, qualidade e variedade de dados sobre saúde pública no Brasil, o Departamento de Informática do Sistema Único de Saúde (DATASUS), órgão associado ao Ministério da Saúde, tem centralizado e mantido diversos bancos de dados desde sua criação, em 1991. O repositório de dados é alimentado por uma rede de sistemas de informações em saúde, responsáveis pela coleta de dados de diferentes áreas de abrangência do SUS, como o Sistema de Informações sobre Mortalidade (SIM) e o Cadastro Nacional de Estabelecimentos de Saúde (CNES) (SILVA; AUTRAN, 2019). Os dados são disponibilizados por meio de planilhas eletrônicas padronizadas no *website* do DATASUS e, mais recentemente, tem sido desenvolvidas também interfaces de programação para simplificar a coleta e o tratamento desses dados. Através do Sistema IBGE de Recuperação Automática (SIDRA), pode-se ainda obter dados agregados e resultados de pesquisas fornecidos pelo Instituto Brasileiro de Geografia e Estatística (IBGE).

Este trabalho visa analisar, transformar e integrar dados coletados por meio desses sistemas, utilizar algoritmos de *clustering* para identificar fatores e grupos de risco no que diz respeito ao suicídio e produzir uma ferramenta de alto nível de abstração com uma

interface gráfica intuitiva para coletar, selecionar, visualizar e aplicar outros algoritmos sobre os dados pré-processados.

1.1 Justificativa

O fenômeno do suicídio é um problema de saúde pública global e representa mais de 1% de todas as mortes anuais. Em 2019, no Brasil, houve mais de 14 mil ocorrências desse fenômeno.

A epidemiologia do suicídio indica que a falta de qualidade de vida e o sentimento de inadequação às normas sociais, que pode ser causado por uma variedade de condições e transtornos, podem induzir a violência autoinfligida (SILVA; MARCOLAN, 2021). Além disso, a psiquiatria e a neuropsicologia são áreas de conhecimento complexas, cujos principais objetos de estudo – o cérebro e o comportamento humano – ainda são pouco conhecidos e cuidados com a saúde mental são permeados por estigma, o que desestimula a busca por ajuda profissional.

Tendo em vista a dificuldade de identificação de fatores agravantes do suicídio e a complexidade das interações entre eles, análises por meio de técnicas de aprendizado de máquina apresentam novas possibilidades. Sendo assim, ferramentas que simplificam a coleta e tratamento de dados e a aplicação dessas técnicas podem ter contribuições bastante significativas para a construção do conhecimento científico.

1.2 Objetivos

1.2.1 Objetivo Geral

O objetivo geral deste projeto é o desenvolvimento de uma ferramenta de coleta e análise de dados de suicídios no Brasil, no formato de uma aplicação web, para auxiliar na realização de pesquisas sobre o tema e facilitar a aplicação de técnicas de *clustering* para identificação de grupos e fatores de risco.

1.2.2 Objetivos Específicos

A ferramenta deve:

- Coletar, tratar e enriquecer dados de mortalidade do SIM, integrando atributos de outros bancos de dados do DATASUS como o CNES;
- Disponibilizar funcionalidades para realizar uma análise descritiva dos dados de suicídio;

- Possibilitar a aplicação de técnicas de *clustering* para identificação de grupos e fatores de risco entre vítimas de suicídio;
- Oferecer diversas opções de visualização dos dados.

1.3 Organização do Trabalho

Este documento está organizado da seguinte forma:

- Capítulo 2: disserta sobre a fundamentação teórica relacionada à mineração de dados, incluindo metodologia e técnicas utilizadas;
- Capítulo 3: discorre sobre trabalhos relacionados;
- Capítulo 4: trata sobre a proposta deste TCC, detalhando o processo de desenvolvimento da ferramenta e as funcionalidades implementadas;
- Capítulo 5: demonstra o funcionamento da ferramenta e análises que podem ser realizadas.
- Capítulo 6: apresenta uma breve conclusão e maneiras de dar continuidade a este trabalho.

2 Fundamentação Teórica

Este capítulo apresenta conceitos básicos e algoritmos relacionados à mineração de dados importantes para melhor compreender a solução proposta por este trabalho. Na última seção, apresenta o Departamento de Informática do SUS, órgão responsável pela disponibilização de dados de saúde pública no Brasil.

2.1 Mineração de Dados

Mineração de dados (em inglês, *Data Mining*) é o estudo de técnicas de coleta, limpeza, processamento e análise de dados muito volumosos (AGGARWAL, 2015).

Em linhas gerais, o processo de mineração de dados contém três etapas principais:

- Coleta de dados: O andamento desta etapa é bastante dependente do projeto em questão e da disponibilidade dos dados. A coleta frequentemente requer o uso de ferramentas específicas, como sensores ou programas de raspagem de dados (*web scraping*). Em muitos casos, é necessário também trabalho manual, como a criação e aplicação de formulários específicos.

- Pré-processamento de dados: Nesta segunda etapa, os dados são preparados para a aplicação de algoritmos de mineração. É bastante comum que conjuntos de dados possuam uma série de problemas de formatação e representação, como valores incorretos ou inválidos, diferentes tipos de dados misturados arbitrariamente em um único atributo e ainda colunas com muitos dados faltantes. Durante o pré-processamento, são realizadas a limpeza, a extração e transformação de atributos e, quando necessária, a integração de dados de fontes diferentes.

- Processamento analítico: A parte final do processo de mineração de dados é o desenvolvimento de um modelo de análise automatizada, capaz de extrair eficientemente conhecimento efetivo sobre determinado conjunto de dados. Nesta etapa, é imprescindível o uso de algoritmos de aprendizado de máquina.

Existem diferentes abordagens para a caracterização do processo de mineração de dados. Na subseção a seguir, será apresentada uma metodologia genérica, muito conhecida e utilizada como base de projetos desse gênero.

2.1.1 CRISP-DM

O fluxo de desenvolvimento deste projeto é inspirado pela metodologia CRISP-DM (*Cross Industry Standard Process - Data Mining*), que é um modelo de processo padrão

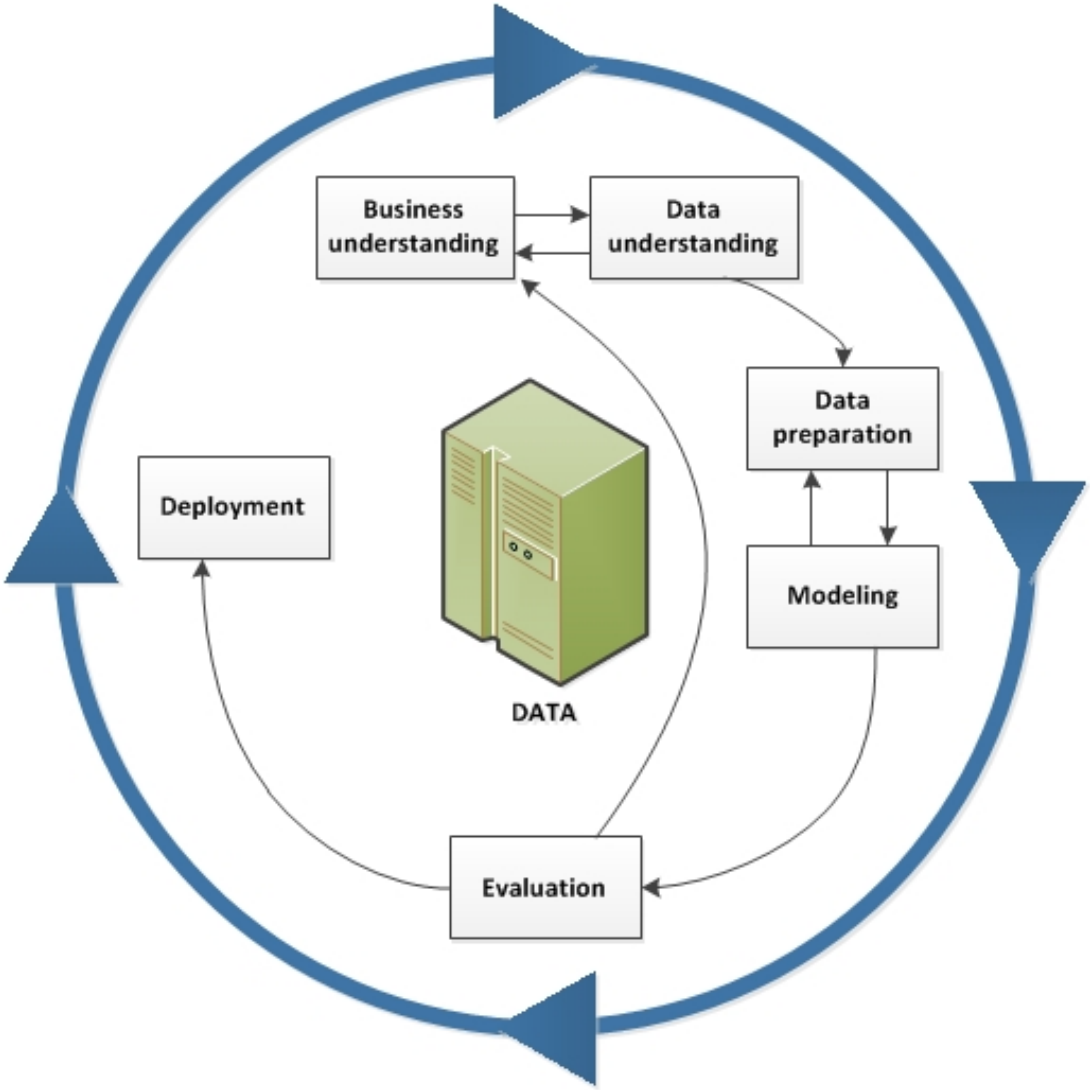


Figura 1 – Metodologia CRISP-DM

para mineração de dados, independente de ferramentas e pode ser aplicado em qualquer tipo de negócio. O CRISP-DM foi desenvolvido em 1996 como forma de apoio ao processo de descoberta de conhecimento em bancos de dados. É dividido em seis partes, como mostra a Figura 1.

Na etapa de **entendimento do negócio**, é feita a análise do problema e das necessidades e expectativas das partes envolvidas para definição dos objetivos do projeto. Em seguida, na fase de **entendimento dos dados**, são estudados os dados disponíveis para a execução do projeto, o tipo de informação que podem prover e métodos que possam ser utilizados.

A **preparação dos dados** costuma ser a etapa mais demorada, em que os atributos relevantes para o contexto serão selecionados e será feita a limpeza e formatação dos dados (livrando-os de inconsistências e valores incorretos ou inválidos). Nesta fase também ocorre a engenharia de atributos, em que novos atributos serão derivados daqueles que foram previamente selecionados.

Na fase de **modelagem** são aplicadas as técnicas de mineração de dados propriamente ditas, como algoritmos de aprendizado de máquina, para desenvolvimento de um modelo. Em seguida, durante a **avaliação**, a qualidade dos resultados obtidos pela modelagem será analisada e, eventualmente, adaptações serão feitas ao modelo.

Por fim, a etapa de **implantação** consiste na elaboração de uma maneira de aplicar ou divulgar os resultados obtidos ao longo do projeto.

2.2 Aprendizado de Máquina

Aprendizado de máquina ou, em inglês, *machine learning* (ML), é um subconjunto de técnicas de Inteligência Artificial que se baseiam no processo de aprendizado humano. Nesse contexto, fez-se necessária uma definição do conceito de aprendizado que possa ser estendida para a computação. Uma possível definição genérica é: considera-se que um programa de computador aprende a executar uma classe de tarefas se seu desempenho na execução dessas tarefas é melhorado por meio da experiência (MITCHELL, 1997).

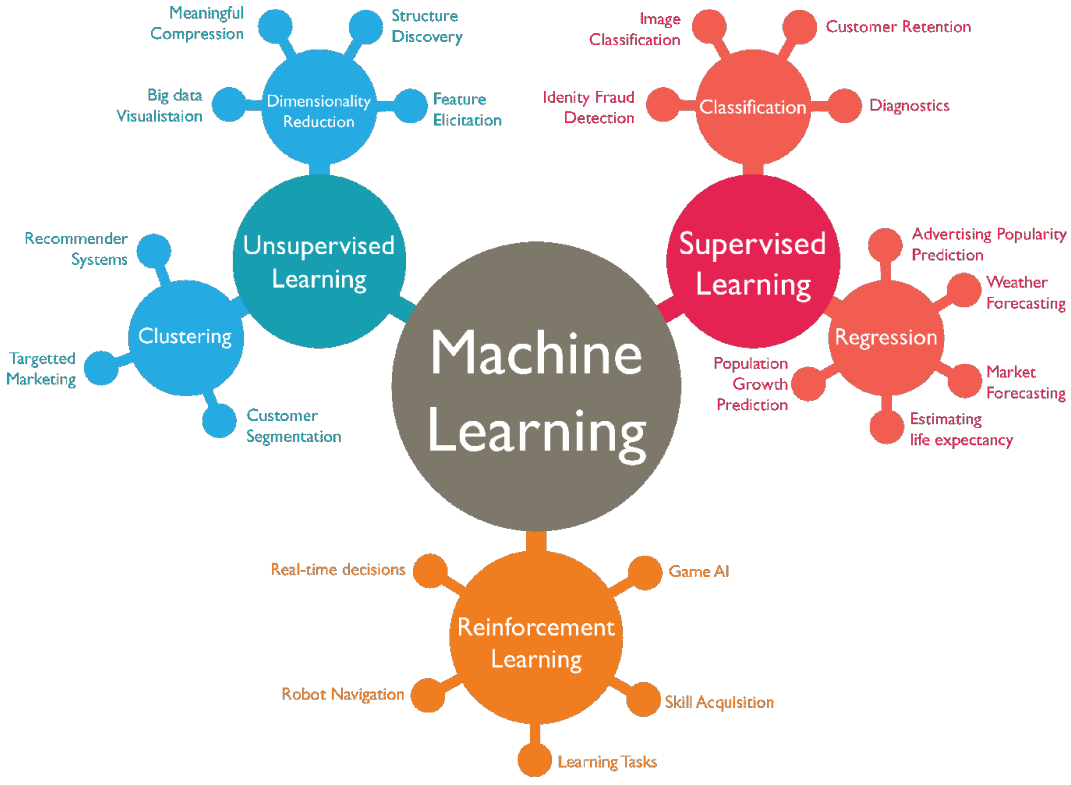


Figura 2 – Técnicas e aplicações de ML

Em linhas gerais, algoritmos de ML podem ser separados em três principais tipos de aprendizado (RUSSELL; NORVIG, 1995):

Em **aprendizado supervisionado**, o conhecimento é inferido a partir de exemplos de entradas e saídas. Acontece quando os dados do conjunto analisado já estão classificados,

e o objetivo é desenvolver um modelo capaz de caracterizar as classes e rotular novos dados.

Em **aprendizado por reforço**, o modelo aprende ao ser recompensado ou punido por seu desempenho. Esses retornos são definidos com uma espécie de pontuação que indica ao modelo se fez algo certo ou errado.

Em **aprendizado não-supervisionado**, a obtenção de conhecimento se dá por meio da detecção de padrões nos dados de entrada; não há uma saída predefinida. A tarefa de aprendizado não-supervisionado mais comum é a análise de *clusters*.

A Figura 2 mostra um diagrama sobre os tipos de aprendizado aqui descritos e algumas de suas principais aplicações.

2.3 Análise de *Clusters*

A análise de *clusters* (ou simplesmente *clustering*) é o nome dado à tarefa de aprendizado não-supervisionado que visa agrupar um conjunto de objetos em função de alguma métrica de distância ou de similaridade, de modo que elementos associados a um determinado grupo são mais semelhantes entre si e diferentes de elementos de outros grupos. Existem muitos algoritmos de *clustering* que definem grupos com base em diferentes critérios.

Neste trabalho, será utilizada a técnica de *clustering* hierárquico. Existem dois tipos de algoritmos hierárquicos, caracterizados pelo modo com que a árvore hierárquica (ver Figura 3) é construída (AGGARWAL, 2015):

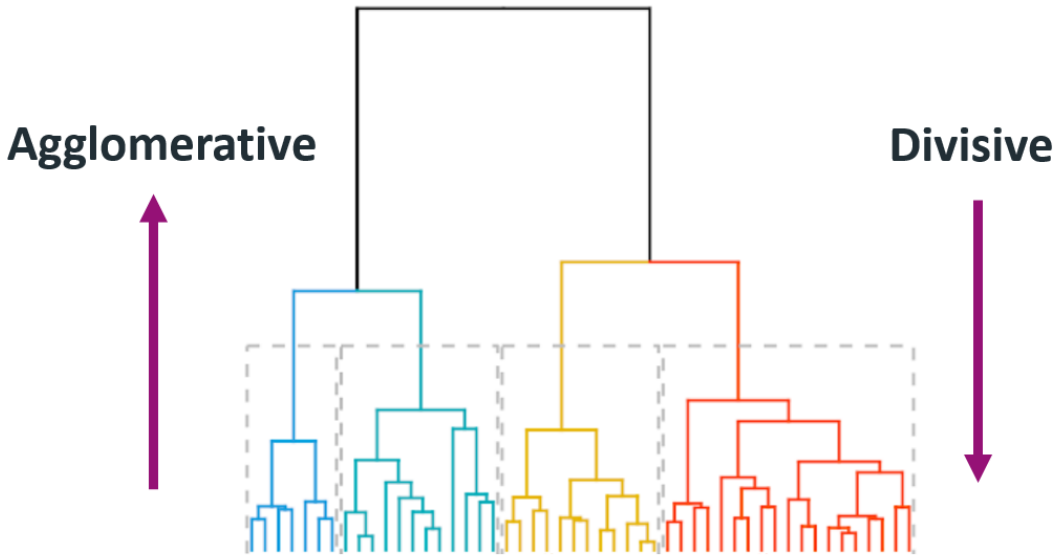


Figura 3 – Tipos de *clustering* hierárquico

- Métodos aglomerativos (*bottom-up*): Os dados são agrupados sucessivamente em grupos de maior nível conforme a função de junção utilizada.
- Métodos divisivos (*top-down*): Nesse caso, um conjunto de dados é particionado em uma estrutura de árvore.

Enquanto métodos divisivos permitem maior controle sobre a quantidade de dados por nodo, métodos aglomerativos formam, naturalmente, árvores binárias (ilustradas por dendrogramas); ou seja, a estrutura da árvore hierárquica é muito mais flexível em métodos divisivos, tornando-os desejáveis para aplicações taxonômicas.

No caso deste trabalho, serão utilizados essencialmente métodos aglomerativos.

2.3.1 Critérios de Avaliação

Após a aplicação de técnicas de *clustering* em um conjunto de dados, é necessário avaliar a qualidade dos agrupamentos formados. Critérios de avaliação de *clusters* são separados em duas categorias: avaliação interna e avaliação externa (AGGARWAL, 2015).

Critérios de avaliação externa são utilizados quando se tem conhecimento concreto dos *clusters* nos dados subjacentes. Não é uma situação comum na maioria dos conjuntos de dados.

Critérios de avaliação interna são normalmente derivados da própria função objetiva de um algoritmo e, portanto, podem produzir resultados enviesados quando utilizados para avaliar algoritmos com características diferentes. São úteis para otimização de parâmetros, mas seus valores devem ser analisados com cuidado.

Uma das métricas mais utilizadas é o índice de silhueta. É definido dentro do intervalo (-1, 1) e valores próximos de 1 indicam *clusters* bem definidos. Outros critérios de avaliação interna incluem:

- Soma de distâncias quadráticas até o centroide (SSQ): Valores menores apontam maior qualidade. Não otimizado para algoritmos que não sejam baseados em distância.
- Taxa de distância média *intracluster* sobre *intercluster*: Uma versão mais detalhada da medida SSQ, calcula a média das distâncias entre pontos de um mesmo *cluster* e divide pela média das distâncias entre diferentes *clusters*.

2.4 DATASUS

O Departamento de Informática do Sistema Único de Saúde (DATASUS) é um órgão da Secretaria de Gestão Estratégica e Participativa do Ministério da Saúde e é responsável

por coletar, processar e disponibilizar informações sobre saúde (DATASUS, Online). Foi criado em 1991 e teve as seguintes competências definidas:

1. Fomentar, regulamentar e avaliar as ações de informatização do SUS, direcionadas para a manutenção e desenvolvimento do sistema de informações em saúde e dos sistemas internos de gestão do Ministério;

2. Desenvolver, pesquisar e incorporar tecnologias de informática que possibilitem a implementação de sistemas e a disseminação de informações necessárias às ações de saúde;

3. Definir padrões, diretrizes, normas e procedimentos para transferência de informações e contratação de bens e serviços de informática no âmbito dos órgãos e entidades do Ministério;

4. Definir padrões para a captação e transferência de informações em saúde, visando à integração operacional das bases de dados e dos sistemas desenvolvidos e implantados no âmbito do SUS;

5. Manter o acervo das bases de dados necessárias ao sistema de informações em saúde e aos sistemas internos de gestão institucional;

6. Assegurar aos gestores do SUS e órgãos congêneres o acesso aos serviços de informática e bases de dados, mantidos pelo Ministério;

7. Definir programas de cooperação técnica com entidades de pesquisa e ensino para prospecção e transferência de tecnologia e metodologias de informação e informática em saúde;

8. Apoiar Estados, Municípios e o Distrito Federal, na informatização das atividades do SUS;

9. Coordenar a implementação do sistema nacional de informação em saúde, nos termos da legislação vigente.

Desde a sua criação, o DATASUS já desenvolveu uma variedade de sistemas de informação que auxiliam diretamente o Ministério da Saúde no processo de construção e fortalecimento do SUS. Entre eles, está o principal banco de dados utilizado neste projeto: o Sistema de Informação sobre Mortalidade (SIM). Foi criado pelo Ministério da Saúde em 1975 e informatizado em 1979, e armazena dados coletados por atestados ou declarações de óbito, que são formulários preenchidos por médicos e testemunhas com informações sobre o falecido e a *causa mortis*.

O DATASUS também gerencia outros conjuntos de dados interessantes para o desenvolvimento deste trabalho, como o Cadastro Nacional de Estabelecimentos de Saúde

(CNES), que contém todas as informações equipamentos, leitos, profissionais e estabelecimentos instalados para atendimento à população brasileira. O CNES pode ser utilizado para enriquecer os dados de suicídios obtidos a partir do SIM com informações sobre a disponibilidade e qualidade de atendimento médico em cada município. Os sistemas de saúde administrados pelo DATASUS não são integrados entre si, mas a junção pode ser realizada através dos códigos de municípios, definidos conforme padrão estabelecido pelo Instituto Brasileiro de Geografia e Estatística (IBGE).

3 Trabalhos Relacionados

Este capítulo discorre sobre o tema, a metodologia e os resultados de trabalhos relacionados à análise de dados de suicídio por meio de técnicas de aprendizado de máquina. Ao final, apresenta um resumo comparativo entre os trabalhos mencionados e este TCC.

3.1 Prediction by data mining, of suicide attempts in Korean adolescents: a national study

Neste artigo (BAE; LEE; LEE, 2015), os autores visam desenvolver um modelo preditivo para tentativas de suicídio entre adolescentes coreanos. Os dados utilizados foram obtidos por meio de uma pesquisa sobre saúde mental feita pelo Instituto Nacional de Políticas da Juventude da Coreia do Sul (INPJ-CS) em 2011. Numa tarefa de classificação, os autores utilizaram o método de árvores de decisão e fixaram "tentativa de suicídio"como variável alvo. Essa variável foi obtida através da seguinte pergunta feita aos estudantes: "você tentou cometer suicídio nos últimos 12 meses?", cujas respostas poderiam ser apenas "sim"ou "não".

Para as demais colunas, foram selecionados atributos sociodemográficos (descritos na Tabela 1) e também definidas variáveis intra e extrapessoais a partir de diversos formulários e métodos, apresentadas na Tabela 2.

Tabela 1 – Atributos sociodemográficos de estudantes coreanos

Atributo	**Descrição**
Sexo	Masculino ou feminino
Idade	Idade em anos
Escolaridade	Ensino fundamental ou médio
Histórico escolar	Excelente, bom, mediano ou ruim
Formação acadêmica dos pais	-
Situação empregatícia dos pais	-
Localização da escola	Metropolitana, micropolitana ou área rural
Classe socioeconômica	Alta, média ou baixa
Estrutura familiar	Pai e mãe, pai ou mãe solteira, pais divorciados, etc

Foram analisados dados de 2754 estudantes de ensino fundamental e médio, dos quais 9.5% haviam tentado cometer suicídio nos últimos 12 meses. Os autores observaram que a variável com maior influência sobre tentativas de suicídio foi a depressão, mensurada

Tabela 2 – Atributos intra e extrapessoais de estudantes coreanos

Atributo	**Descrição**
Depressão	Nível de depressão conforme BDI
Estresse	Nível de estresse diário conforme questionário criado pelo INPJ-CS
Delinquência	Medida conforme questionário desenvolvido pelo INPJ-CS, que inclui itens como uso de drogas, violência, roubo, vandalismo, etc
Auto-estima	Medida conforme questionário criado pelo INPJ-CS
Otimismo	Medida conforme questionário criado pelo INPJ-CS
Intimidade com a família	Medida conforme questionário criado pelo INPJ-CS
Suporte comunitário	Medida conforme questionário criado pelo INPJ-CS
Adaptação à escola	Medida conforme questionário criado pelo INPJ-CS

por meio de um questionário conhecido como *Beck Depression Inventory* (BDI). Assim, o modelo dividiu os casos em três grupos: com depressão, com possível depressão e sem depressão (ver Figura 4).

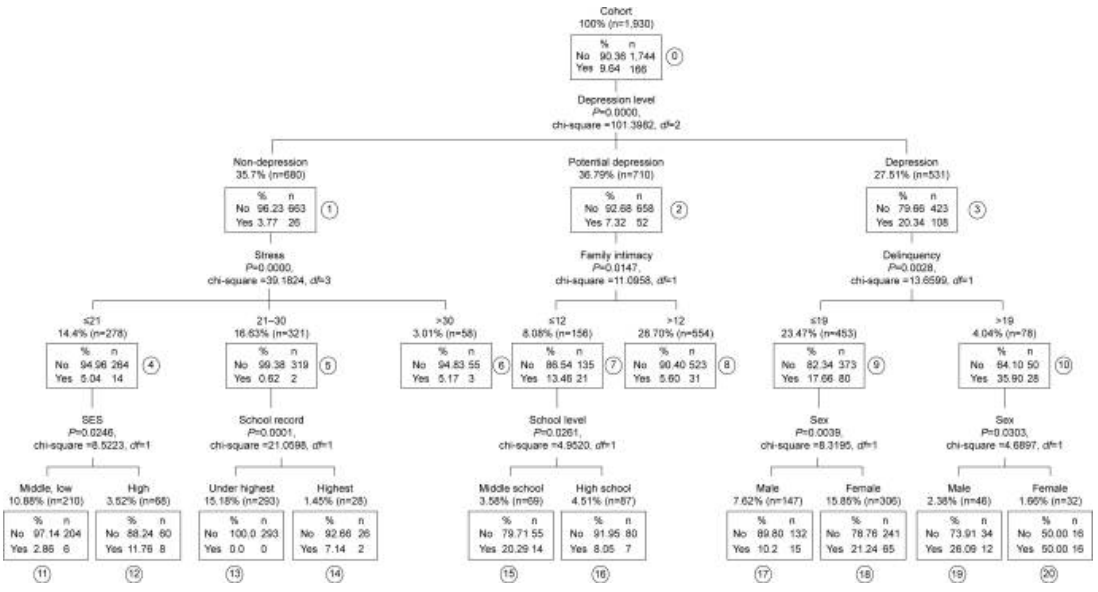


Figura 4 – Árvore de decisão para predição de tentativas de suicídio

A taxa de tentativas de suicídio foi de 20.34% no grupo com depressão, 7.32% no grupo com possível depressão e 3.77% no grupo sem depressão. Os próximos atributos com maiores ganhos de informação também foram diferentes para cada grupo. Entre estudantes depressivos, aqueles com altas pontuações no conceito "delinquência"representam aproximadamente 15% do grupo, no entanto mostraram uma taxa de tentativas de suicídio de 35.90%, enquanto os demais apresentaram uma taxa de 17.66%.

No grupo com possível depressão, a principal influência sobre a taxa de tentativas de

suicídio foi a intimidade com a família. Estudantes com alta intimidade apresentaram uma taxa de 5.60%, contra 13.46% entre aqueles com menor intimidade.

Por fim, no grupo sem depressão, o atributo que mais afetou a taxa foi o nível de estresse. Estudantes com alto nível de estresse mostraram uma taxa de 5.17%, enquanto aqueles com baixo nível de estresse exibiram apenas 0.62%.

3.2 Clustering suicides: A data-driven, exploratory machine learning approach

Este artigo (LUDWIG et al., 2019) propõe a utilização de *clustering* hierárquico para, a princípio, analisar a validade da classificação de métodos de suicídio em "violentos"e "não-violentos"e buscar outras possibilidades de agrupá-los. Os autores afirmam que essa dicotomia é ambígua, sendo que não há consenso sobre onde cada método se encaixaria. Ademais, a análise realizada neste artigo contribui também para a discussão sobre identificação de padrões em vítimas de suicídio.

Os dados foram obtidos por meio da *Statistik Austria*, a agência estatística federal da Áustria. Foram utilizados dados de todos os suicídios confirmados oficialmente no país, entre 1970 e 2016, totalizando 77894 casos.

Os atributos selecionados para esta análise estão apresentados na Tabela 3. Utilizando códigos da CID-10, os autores agruparam os métodos de suicídio em cinco categorias: envenenamento (X60 a X69), enforcamento (X70), afogamento (X71), arma de fogo (X72 a X74) e salto (X80 e X81). Os demais métodos (X75 a X79, X82 a X84) foram classificados como "outros". A princípio, a categoria de suicídios violentos seria composta por mortes por enforcamento, afogamento, arma de fogo ou salto de grande altura; suicídios não-violentos seriam aqueles cometidos por envenenamento.

Tabela 3 – Atributos selecionados para *clustering* hierárquico

Atributo	**Descrição**
Idade	Vinte grupos definidos em intervalos de cinco anos
Sexo	Masculino ou feminino
Método	Método de suicídio: envenenamento, enforcamento, afogamento, arma de fogo ou salto
Mês	Mês do óbito

Os autores observaram que 72% das vítimas de suicídio analisadas são do sexo masculino. Quanto à idade, a mediana entre homens foi dada pelo grupo de 50-54 anos e, entre as mortes do sexo feminino, foi o grupo de 55-59 anos. O método de suicídio mais comum foi enforcamento (46.3%).

Quanto aos resultados do *clustering* hierárquico, a amostra total apresentou maior similaridade entre os métodos de enforcamento e arma de fogo, que em seguida foram agrupados com métodos de afogamento e saltos de grandes alturas. Para visualização dos dados, uma superfície 3D foi formada para cada método de suicídio, com eixos representados pela idade, mês do óbito e número de mortes. Essas superfícies e o dendrograma mostrando a distância entre *clusters* podem ser vistos na Figura 5.

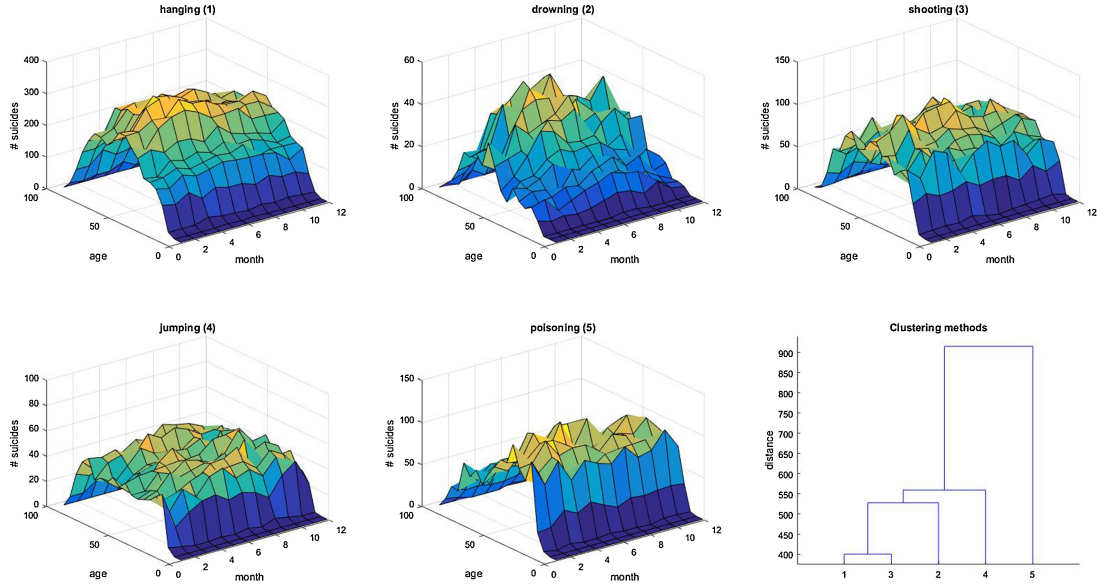


Figura 5 – Visualização da amostra total de suicídios na Áustria, separados por método, e do dendrograma mostrando os agrupamentos resultantes

Na amostra total, percebe-se que há de fato uma separação entre mortes por envenenamento e os demais métodos. No entanto, algumas fontes sugerem uma aproximação entre suicídios por envenenamento e afogamento, classificados como não-violentos, e isso não é sustentado pelos resultados desse trabalho.

Além disso, uma segunda análise conduzida, estratificada por sexo, indica que essa classificação não se aplica às mulheres. Quanto às vítimas do sexo masculino, a estrutura do dendrograma é bastante semelhante ao da amostra total. Contudo, entre as vítimas do sexo feminino, os métodos de arma de fogo e saltos se assemelham mais a envenenamento, que seria considerado não-violento (ver Figura 6).

3.3 Cluster analysis reveals risk factors for repeated suicide attempts in a multi-ethnic Asian population

Este estudo (CHOO et al., 2014) investiga os fatores de risco de repetidas tentativas de suicídio em uma população asiática multiétnica (n = 418). Os pesquisadores coletaram

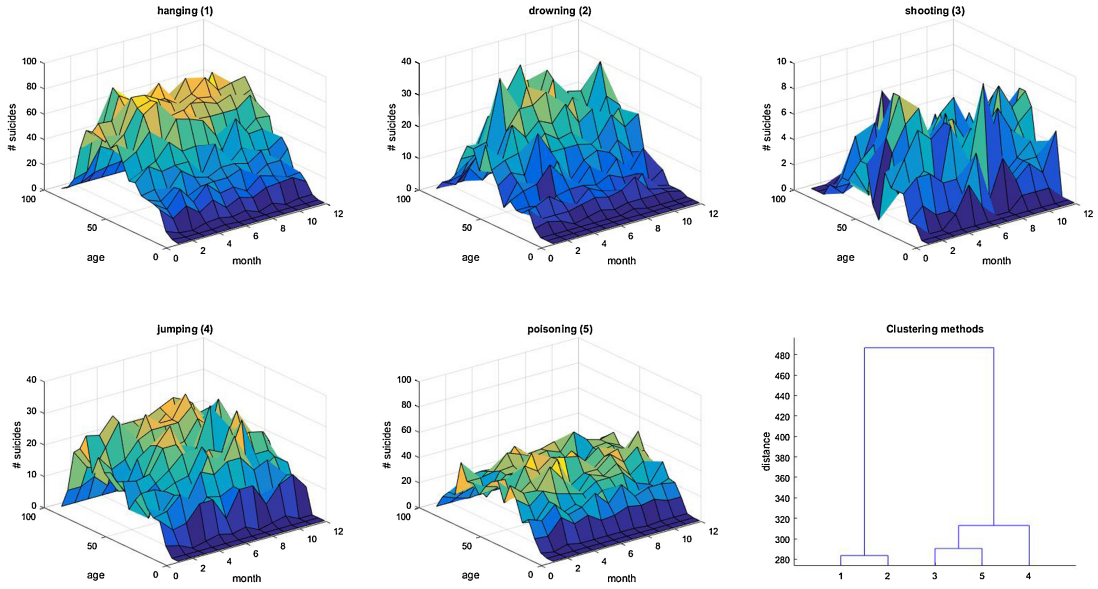


Figura 6 – Visualização da amostra de suicídios do sexo feminino, separados por método, e do dendrograma mostrando os agrupamentos resultantes

dados de avaliações psiquiátricas de pacientes que já haviam tentado cometer suicídio alguma vez e realizaram uma análise de *clusters* utilizando o algoritmo de duas etapas.

O artigo demonstrou que fatores demográficos como gênero, idade e etnia contribuíram significativamente para diferenças estatísticas entre pacientes que haviam tentado suicídio diversas vezes e pacientes que tentaram uma única vez. Mais especificamente, a pesquisa identificou que as tentativas repetidas de suicídio eram mais comuns em pessoas do sexo feminino, mais jovens e de etnia malaia. Além disso, observou-se uma relação direta entre repetidas tentativas de suicídio e a existência de certos diagnósticos psiquiátricos como depressão, abuso de substâncias e transtorno de personalidade borderline (TPD).

Além dos fatores demográficos e psiquiátricos, o estudo analisou também a ocorrência de eventos estressantes, como múltiplas internações em hospitais psiquiátricos, desemprego, divórcio e brigas com familiares. A aplicação do algoritmo de duas etapas formou dois *clusters* de tamanhos significativamente diferentes, com 353 pacientes (84.4%) no primeiro *cluster* e 65 (15.6%) no segundo. O primeiro *cluster* foi caracterizado por menor ocorrência dos fatores de risco analisados e foi interpretado como uma representação de um grupo de pacientes com bom prognóstico. O segundo *cluster* apresentou características contrárias, com mais casos de TPD e depressão e relatos de dores de cabeça, alucinações e insônia, sendo assim interpretado como uma representação de pacientes com prognóstico ruim.

3.4 Resumo Comparativo

Acima, foram discutidos artigos que demonstram a validade da aplicação de técnicas de aprendizado de máquina, em especial *clustering*, para a análise de dados de suicídio,

justificando a utilidade de uma ferramenta para auxiliar nessa abordagem. Os estudos utilizaram dados com quantidades e dimensionalidades um pouco diferentes, mas de natureza semelhante. Além disso, as pesquisas foram realizadas sobre indivíduos de diferentes origens, em continentes distintos.

A Tabela 4 apresenta um resumo comparativo.

Tabela 4 – Resumo comparativo entre trabalhos relacionados

Trabalho	**Objetivo**	**Dados**	**País**	**Algoritmo**	**Resultados**
(BAE; LEE; LEE, 2015)	Predição de risco de tentativas de suicídio entre adolescentes	Formulários preenchidos por 2574 estudantes	Coreia do Sul	Árvores de Decisão	Depressão é o principal fator de risco
(LUDWIG et al., 2019)	Identificação e validação de categorias de métodos de suicídio	77894 suicídios registrados oficialmente	Áustria	*Clustering* Hierárquico	A classificação de métodos como violentos ou não-violentos é inconsistente
(CHOO et al., 2014)	Identificação de fatores de risco de suicídio	Relatórios médicos de 418 tentativas de suicídio	Singapura	*Two-Step Clustering*	Transtornos psiquiátricos e situações de estresse são fatores de risco significativos
Este TCC	Desenvolvimento de ferramenta para análise de dados de suicídio	Dados públicos de mortalidade do DATASUS	Brasil	*Clustering* Hierárquico	Ferramenta de análise, descrita no Capítulo 4

4 Desenvolvimento da Ferramenta de Análise

Neste capítulo, será apresentado o processo de desenvolvimento da ferramenta[1] de análise de dados projetada para facilitar a exploração e visualização de dados de mortalidade por suicídio no sul do Brasil. O objetivo desta ferramenta é fornecer aos usuários uma interface intuitiva para baixar dados pré-processados, analisar diversos atributos com gráficos customizáveis e ainda aplicar técnicas de *clustering*. Com essas funcionalidades, visa especialmente auxiliar médicos, pesquisadores, e também o público geral a compreender tendências e padrões em casos de suicídio. A ferramenta deve ser expansível e adaptável para incrementar a análise com dados de outras regiões e possibilitar a utilização de suas funcionalidades em outros contextos.

A Figura 7 apresenta o fluxo de coleta, pré-processamento e *clustering* hierárquico executado pela ferramenta no formato de uma aplicação *web*.

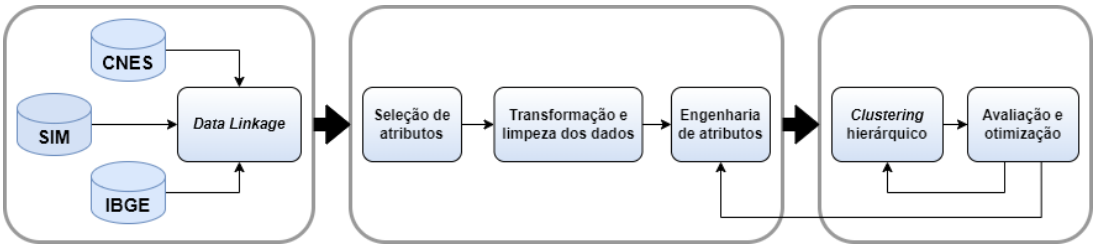


Figura 7 – Fluxo de execução da ferramenta desenvolvida

4.1 Design

A ferramenta foi projetada com a intenção de possibilitar que usuários realizem análises em tempo real sem a necessidade de instalar programas complexos. Segue o modelo de arquitetura cliente-servidor, utilizando Python para processamento e visualização dos dados no *back-end* e o *framework* de desenvolvimento web Streamlit para implementar uma interface web.

A página principal permite ao usuário, de maneira simples e intuitiva, selecionar atributos de interesse, especificar intervalos de tempo e escolher entre múltiplas possibilidades de visualização. Em uma segunda página, dedicada à aplicação de *clustering* hierárquico sobre os dados pré-processados, o usuário pode escolher o método a ser utilizado, visualizar o dendrograma resultante e obter métricas de avaliação dos *clusters*, como o coeficiente

[1] A ferramenta está disponível em <https://simexplorer.streamlit.app> e o código-fonte está disponível no repositório <https://github.com/lust2k/SIM_Explorer.git>

de silhueta. O processo de desenvolvimento foi bastante simplificado pelo Streamlit, que disponibiliza *widgets* prontos para criar elementos interativos.

No *back-end*, a biblioteca PySUS é importada para fazer a coleta de dados de diferentes bancos do DATASUS e do IBGE e armazená-los na memória. Outras bibliotecas como Pandas e NumPy são essenciais para a integração, transformação e enriquecimento dos dados originais e, para gerar as diversas visualizações, foram utilizadas ainda bibliotecas como Matplotlib e Seaborn.

4.2 Requisitos funcionais

1. Interface de usuário:
    - Implementar uma interface clara e intuitiva, fácil de utilizar;
    - Possibilitar a coleta, o processamento e a visualização de dados por meio de *widgets* como caixas de seleção e botões.
2. Coleta e pré-processamento de dados:
    - Permitir a coleta dos dados não tratados de bancos do DATASUS;
    - Oferecer funções para pré-processamento dos dados;
    - Armazenar os dados coletados localmente para melhorar a performance.
3. Análise de dados
    - Prover filtros para personalização da análise;
    - Calcular estatísticas descritivas de atributos selecionados;
    - Implementar análise de *clusters* customizável, com funções de avaliação para auxiliar na otimização de hiperparâmetros.
4. Opções de visualização
    - Disponibilizar uma variedade de opções de visualização dos dados, incluindo gráficos de barra, gráficos de dispersão e mapas geoespaciais.
    - Permitir a customização dos gráficos.

4.3 Coleta e Pré-processamento dos Dados

A disponibilidade de dados confiáveis, atualizados e em grande quantidade é essencial para qualquer análise. Para o desenvolvimento deste trabalho, foram utilizados dados do Departamento de Informática do Sistema Único de Saúde (DATASUS) e do Sistema IBGE de Recuperação Automática (SIDRA). O DATASUS gerencia uma variedade de

bancos de dados alimentados pelos Sistemas de Informação em Saúde (SIS), que são centrais para este trabalho. Entre eles, estão o Sistema de Informação sobre Mortalidade (SIM), de onde são obtidos os dados sobre vítimas de suicídio no Brasil e o Cadastro Nacional de Estabelecimentos de Saúde (CNES), que possui uma gama de informações sobre estabelecimentos de saúde como hospitais e clínicas, desde os serviços prestados e a existência de vínculo com o SUS até questões sobre imposto de renda e natureza do estabelecimento.

4.3.1 Coleta e integração

O processo de coleta dos dados foi bastante facilitado pela biblioteca de código aberto PySUS, que vem sendo desenvolvida desde 2016 com o intuito específico de coletar e tratar dados do DATASUS. A biblioteca possui um diretório dedicado à obtenção dos dados via *File Transfer Protocol* (FTP) chamado *pysus.online_data*. Cada módulo desse diretório faz a coleta de um conjunto de dados diferente e, entre eles, existe ainda um *wrapper* para a API do SIDRA; isto é, as tabelas de dados do IBGE também podem ser coletadas através de PySUS. Os dados do SIM e do CNES são coletados, respectivamente, pelas funções *SIM.download* e *CNES.download*. Os estados e anos a serem coletados devem ser informados e, tendo em vista que dados do SIM são coletados em diferentes formatos de acordo com a região e a época, foram selecionados apenas dados da região Sul do país (Paraná, Santa Catarina e Rio Grande do Sul) entre os anos de 2011 e 2019. A partir de 2011, a consistência dos dados é bastante melhor, com menos lacunas e valores inválidos. Futuramente, dados de 2020 e anos seguintes podem ser coletados para análises comparativas entre períodos pré e pós-pandemia de COVID-19.

Desde a atualização 0.9.1 da biblioteca PySUS, os dados são coletados em formato de *parquets* e armazenados em diretórios separados por estado e ano. Por praticidade, a ferramenta desenvolvida neste trabalho faz a leitura de todos os arquivos *.parquet* e os concatena em um único arquivo para ser processado. O SIM fornece, à parte, uma tabela de códigos da décima revisão da Classificação Internacional de Doenças (CID-10) e suas descrições, que também é coletada através do módulo *pysus.online_data.SIM* para enriquecimento semântico do conjunto de dados sobre mortalidade. Para a integração de dados dos municípios de residência das vítimas, que possibilita o cálculo de taxas de mortalidade e de disponibilidade de assistência médica, foi coletada a tabela de número 1505 do SIDRA. Informando o nível territorial (neste caso, municipal) é possível obter o código, nome e população de cada município. Além do enriquecimento semântico proporcionado pela inclusão dos nomes dos municípios, com a população do município foi possível obter a taxa de mortalidade por suicídio (por 100.000 habitantes) e a mesma taxa foi calculada para os estabelecimentos de saúde. Toda a integração teve como chave o código dos municípios, conforme definidos pelo IBGE.

4.3.2 Seleção e transformação

A coleta de dados de mortalidade no Brasil é feita há muito tempo, e os formulários utilizados variam conforme a Unidade Federativa e o ano em que foram aplicados. Portanto, possuem uma grande quantidade de atributos, dos quais muitos são pouco relevantes para o contexto deste trabalho ou não são consistentes nas múltiplas tabelas.

Foram selecionados os atributos necessários para identificar perfis sociodemográficos (sexo, idade, escolaridade etc.) e aqueles que contextualizam o suicídio, como método, local, data e hora de ocorrência. Todos os atributos e suas descrições estão apresentados na Tabela 5.

Dados faltantes, frequentemente representados de maneira inconsistente no conjunto de dados original, foram padronizados. Utilizando funções da biblioteca PySUS, foram decodificados os seguintes atributos: idades, cujo valor originalmente continha um dígito representando a unidade (idade em dias, meses ou anos) e estavam em formato de *strings*; e datas, que foram convertidas de *strings* para o tipo *datetime*.

Tabela 5 – Atributos selecionados

Atributo	**Descrição**
"SEXO"	Masculino ou feminino.
"IDADE"	Idade em anos
"RACACOR"	Raça/cor: branca, preta, amarela, parda ou indígena
"ESC"	Nível de escolaridade: sem escolaridade, ensino fundamental I, fundamental II, médio ou superior.
"ESTCIV"	Estado civil: solteiro, casado, viúvo, divorciado ou em união estável
"CAUSABAS"	Código de causa da morte conforme a Classificação Internacional de Doenças
"LOCOCOR"	Local de ocorrência: estabelecimento de saúde, domicílio, via pública ou outro.
"DTOBITO"	Data do óbito
"HORAOBITO"	Hora do óbito
"CODMUNRES"	Código de município de residência conforme IBGE

Os atributos desenvolvidos a partir da integração de dados proporcionaram um enriquecimento semântico do conjunto, com a adição da descrição dos métodos de suicídio e os nomes dos municípios de residência dos falecidos. Os atributos originais do SIM também foram traduzidos de seus valores numéricos para linguagem natural. Outros atributos, como as taxas de mortalidade e de disponibilidade de assistência médica, calculadas anualmente, possibilitam a análise de certas tendências.

A Tabela 6 apresenta todos os atributos adicionados ao conjunto de dados de mortalidade original.

Tabela 6 – Atributos extraídos

Origem	**Novo atributo**	**Descrição**
"DTOBITO"	"year"	Ano
	"month"	Mês
	"day"	Dia do mês
	"weekday"	Dia da semana
	"holiday"	Ocorrência em feriados ou dias próximos?
	"season"	Estação do ano
"HORAOBITO"	"day_period"	Período do dia
"CAUSABAS"	"method"	Método do suicídio
"IDADE"	"age_group"	Faixa etária (intervalos de 10 anos)
"CODMUN"	"name_muni"	Nome do município de residência
	"pop_muni"	População do município de residência
	"state"	Sigla do estado (UF) do município de residência
	"facility_rate"	Estabelecimentos de saúde mental por 1000 habitantes no município de residência
	"average_suicide_rate"	Média de suicídios por 100 mil habitantes no município de residência

O conjunto de dados resultante desta etapa é alimentado à aplicação web e armazenado localmente em formato CSV.

4.4 *Clustering* e Avaliação

Para a aplicação de algoritmos de *clustering*, os dados são submetidos à outra etapa de pré-processamento. Os valores nulos são imputados com mediana para atributos numéricos e moda para categóricos. Utilizando a biblioteca Pandas, atributos categóricos não-ordinais são codificados em formato one-hot (*dummy variables*).

Funções para aplicação e visualização de algoritmos de *clustering* hierárquico são importadas da biblioteca SciPy. A função "*linkage*"recebe um parâmetro que especifica o método de definição dos *clusters*, como "*single*", "*average*"ou "*complete*"*linkage* e retorna a matriz de distâncias, que pode ser visualizada através de um dendrograma.

Para associar os dados a um *cluster*, é utilizada a função "*fcluster*"e é necessário definir o parâmetro de distância máxima entre pontos. Possíveis valores para esse parâmetro são

indicados pelo eixo y do dendrograma e a otimização dessa escolha pode ser feita por meio de funções de avaliação de *clusters*.

Métricas de avaliação foram importadas da biblioteca scikit-learn, como o coeficiente de silhueta e o índice de Calinski-Harabasz, e foram desenvolvidas funções para calcular seus valores para uma lista de parâmetros arbitrários. Além dos valores numéricos dessas métricas, diagramas de silhueta podem ser gerados e visualizados.

5 Resultados

Este capítulo está dividido em duas seções: a primeira apresenta a interface e o modo de utilização da ferramenta desenvolvida neste projeto; a segunda discorre sobre análises que podem ser realizadas por meio dela.

5.1 Interface e Utilização da Ferramenta

A ferramenta coleta, transforma e armazena os dados pré-processados assim que é executada. Suas funcionalidades estão separadas em duas páginas principais, sendo uma para realizar análises descritivas e auxiliar no entendimento dos dados não-rotulados e outra para realizar análise de *clusters*.

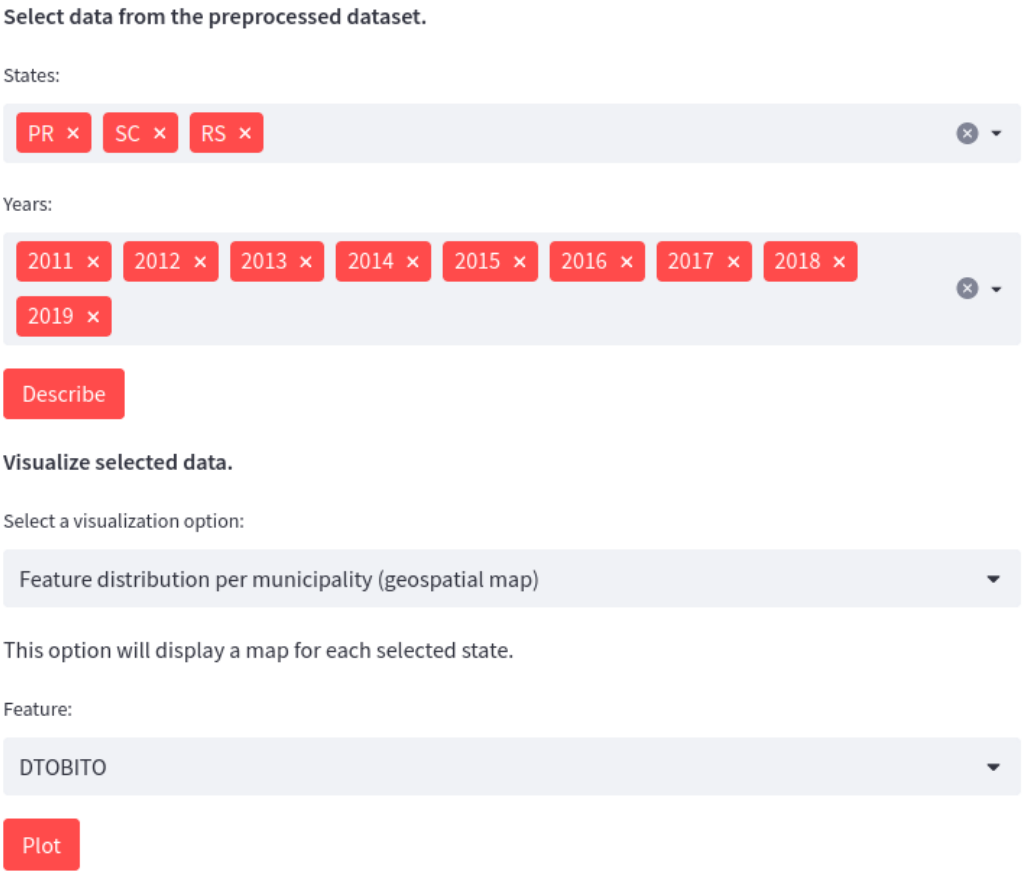


Figura 8 – Página para análise descritiva e entendimento dos dados pré-processados

A Figura 8 mostra a interface da primeira página. Primeiramente, o usuário filtra os dados por estados e anos e, ao clicar no botão "Describe", vê duas tabelas com estatísticas descritivas para atributos numéricos e categóricos, respectivamente. Em seguida, gráficos personalizados podem ser visualizados. Existem duas opções de visualização disponíveis:

um mapa dos estados selecionados e seus municípios, apresentando a distribuição geográfica de um atributo; e um gráfico de barras com a distribuição de um atributo em relação a outro.

A segunda página da ferramenta é dedicada para a aplicação de técnicas de *clustering* hierárquico (ver Figura 9). Nela, além de ser possível filtrar os dados por estados e anos, o usuário também pode selecionar atributos do conjunto de dados pré-processados e o método de definição de *clusters*. Ao clicar no botão "Plot Dendrogram", o algoritmo será aplicado e o dendrograma resultante será mostrado.

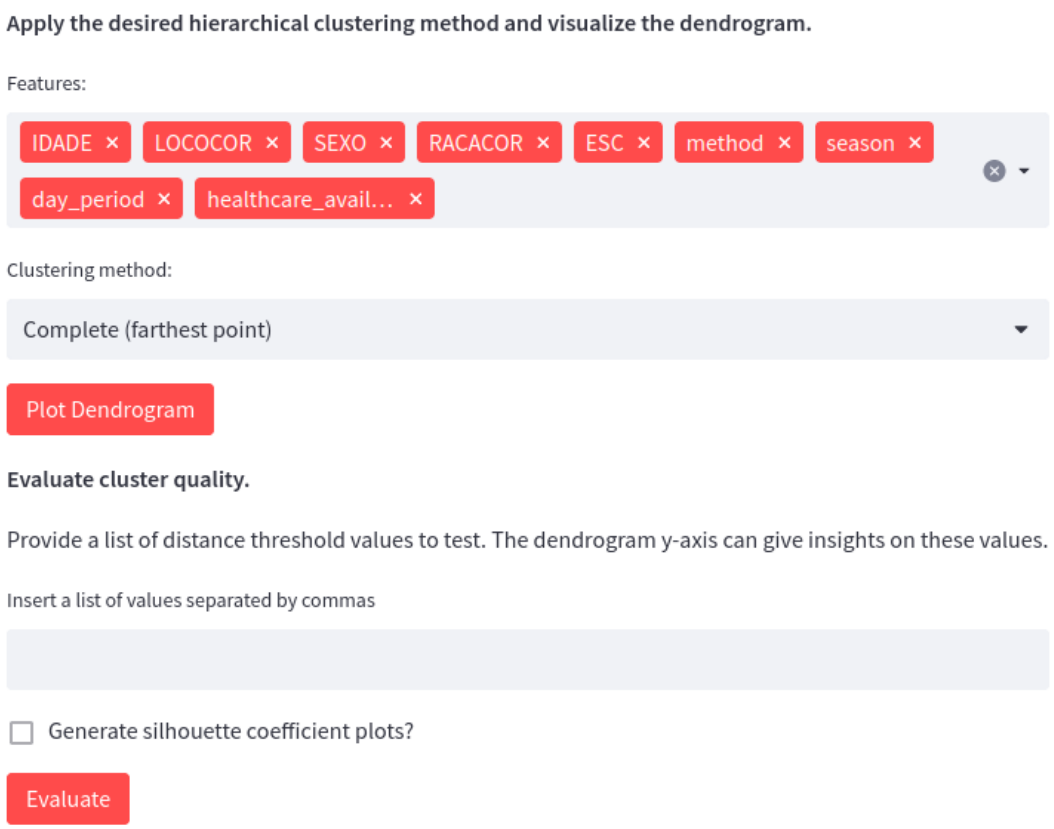


Figura 9 – Página para análise de *clusters*

Para rotular os dados, é necessário ainda definir um limite de distância para formação dos *clusters*. A ferramenta auxilia a otimizar esse parâmetro, possibilitando a avaliação dos *clusters* formados por uma lista de valores diferentes. Basta inserir os valores desejados no campo de texto, separados por vírgulas, e clicar no botão "Evaluate". Métricas de avaliação como os índices de silhueta e de Calinski-Harabasz (CH) serão calculadas e impressas na tela. Caso a caixa de seleção acima do botão esteja marcada, serão gerados gráficos de coeficientes de silhueta.

Uma vez escolhido o limite de distância intra-*cluster*, o usuário pode informá-lo e a ferramenta irá efetivamente rotular os dados, adicionando um atributo que identifica a que *cluster* cada instância pertence. Por fim, opções de visualização dos dados rotulados são disponibilizadas. É possível observar a distribuição geográfica dos *clusters* em mapas dos

estados e seus municípios, e também a distribuição de cada atributo por *cluster* através de mapas de calor.

5.2 Análise de *Clusters* em Dados de Suicídio no Sul do Brasil

Para esta análise, foram selecionados todos os dados tratados neste trabalho. Isto é, os três estados da região sul do Brasil - Paraná, Santa Catarina e Rio Grande do Sul - e os anos entre 2011 e 2020.

Primeiramente, a ferramenta desenvolvida foi utilizada para o entendimento dos dados. As estatísticas descritivas para variáveis numéricas e categóricas foram calculadas por meio da biblioteca Pandas. As tabelas resultantes (ver Figuras 10 e 11) contém também a quantidade de dados não-nulos para cada coluna do conjunto de dados.

	CODMUN	IDADE	year	month	day	pop_muni	facility_rate	average_suicide_rate
count	23,213	23,193	23,213	23,213	23,213	23,179	23,179	23,179
mean	422,970.1978	45.4403	2,015.2982	6.526	15.5423	213,657.1923	2.5962	11.6525
std	8,490.8158	17.7296	2.5938	3.5385	8.7157	408,727.0548	1.5312	5.9447
min	410,000	7	2,011	1	1	1,216	0	1.0805
25%	412,545	31	2,013	3	8	15,373	1.6315	7.3592
50%	421,650	45	2,015	7	15	54,643	2.4048	10.4325
75%	431,330	58	2,018	10	23	197,228	3.1394	14.2602
max	432,380	103	2,019	12	31	1,751,907	14.1041	62.7493

Figura 10 – Estatísticas descritivas para variáveis numéricas

	DTOBITO	HORAOBITO	CAUSABAS	LOCOCOR	SEXO	RACACOR	ESC	ESTCIV	season	weekday	state	name_muni	age_group	method	day_period
count	23213	19677	23213	23170	23211	22983	17908	21510	23213	23213	23213	23179	23178	23213	19675
unique	3284	1040	25	4	2	5	5	5	4	7	3	1146	9	9	4
top	2019-06-03	0800	X70	Domicilio	Masculino	Branca	Fundamental II	Solteiro	Summer	Monday	RS	Porto Alegre	(40, 50]	Estrangulamento	Morning
freq	22	583	16569	15296	18337	20232	6139	9847	6132	3707	10801	906	4441	16569	6394

Figura 11 – Estatísticas descritivas para variáveis categóricas

Observando a moda das variáveis categóricas, obtém-se uma noção inicial das características dos mais de 23 mil suicídios registrados. A imensa maioria ocorreu entre pessoas do sexo masculino, em domicílio, por estrangulamento ou enforcamento. Percebe-se também uma prevalência de casos entre indivíduos solteiros e de nível de escolaridade mais baixo, sem ensino médio ou superior. Para atributos em que a frequência da moda (indicado por "*freq*") é muito próxima da quantidade total de dados dividida pelo número de categorias

possíveis (indicado por "*unique*"), conclui-se que possuem uma distribuição relativamente uniforme e menor influência como um fator de risco. Por exemplo, a estação do ano com mais suicídios contabilizados foi o verão, com 6132 ocorrências. Contudo, sabendo que o total de casos foi de 23213 e existem 4 estações, o valor semântico desse atributo é baixo. De maneira análoga, o mesmo pode ser dito sobre os atributos "*weekday*"(dia da semana) e "*day_period*". Quanto à raça ou cor de pele das vítimas, há uma preponderância bastante acentuada de brancos; porém, essa informação deve ser considerada com cuidado, já que a população da região analisada é composta por, majoritariamente, brancos.

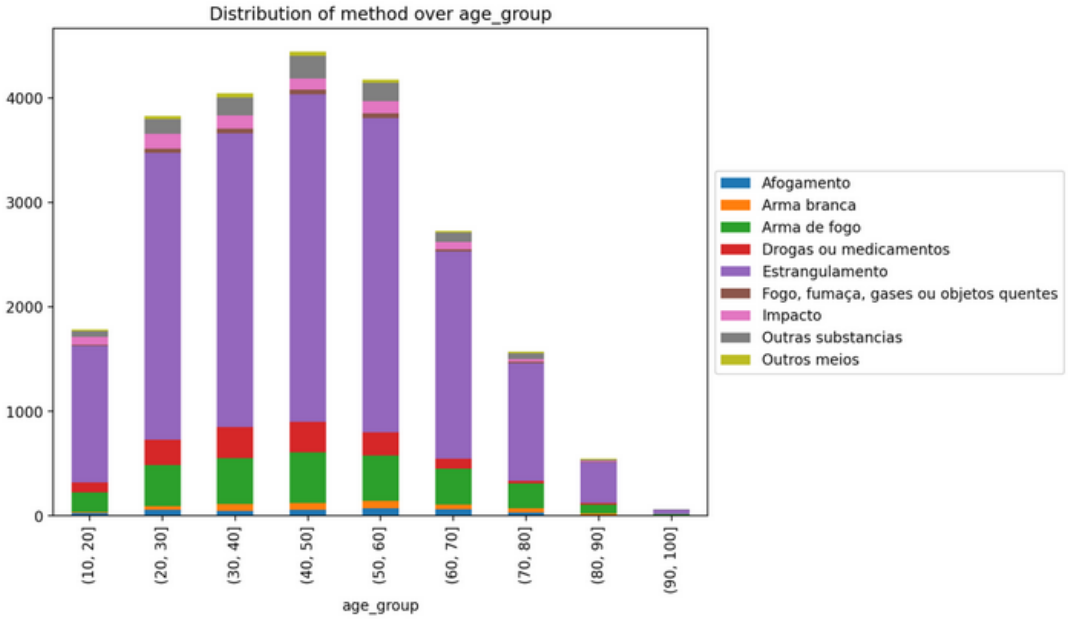


Figura 12 – Distribuição de métodos de suicídio por faixa etária

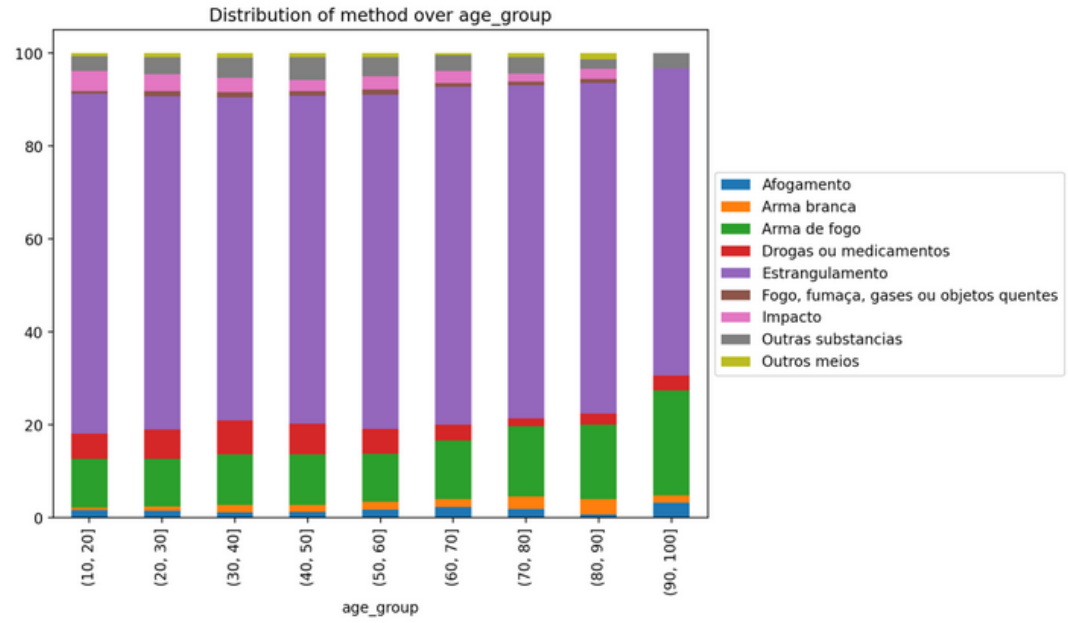


Figura 13 – Distribuição de métodos de suicídio por faixa etária (em %)

Em seguida, foram observadas as distribuições de atributos em relação a outros utilizando uma das opções de visualização disponíveis - gráficos de barra customizáveis. Por exemplo, a distribuição de métodos de suicídio por faixa etária pode ser vista na Figura 12. Como indicado, o estrangulamento prevalece como método mais comum em valores absolutos para todas as idades. Desenhando o gráfico com as mesmas variáveis, mas com o eixo y indicando porcentagens, algumas tendências podem ser vistas com maior clareza (ver Figura 13). Mortes por auto-intoxicação intencional através de drogas, medicamentos ou outras substâncias foram mais comuns entre jovens, em particular aqueles com idades entre 20 e 29 anos. Já na população mais velha, nota-se um aumento percentual no uso de armas de fogo.

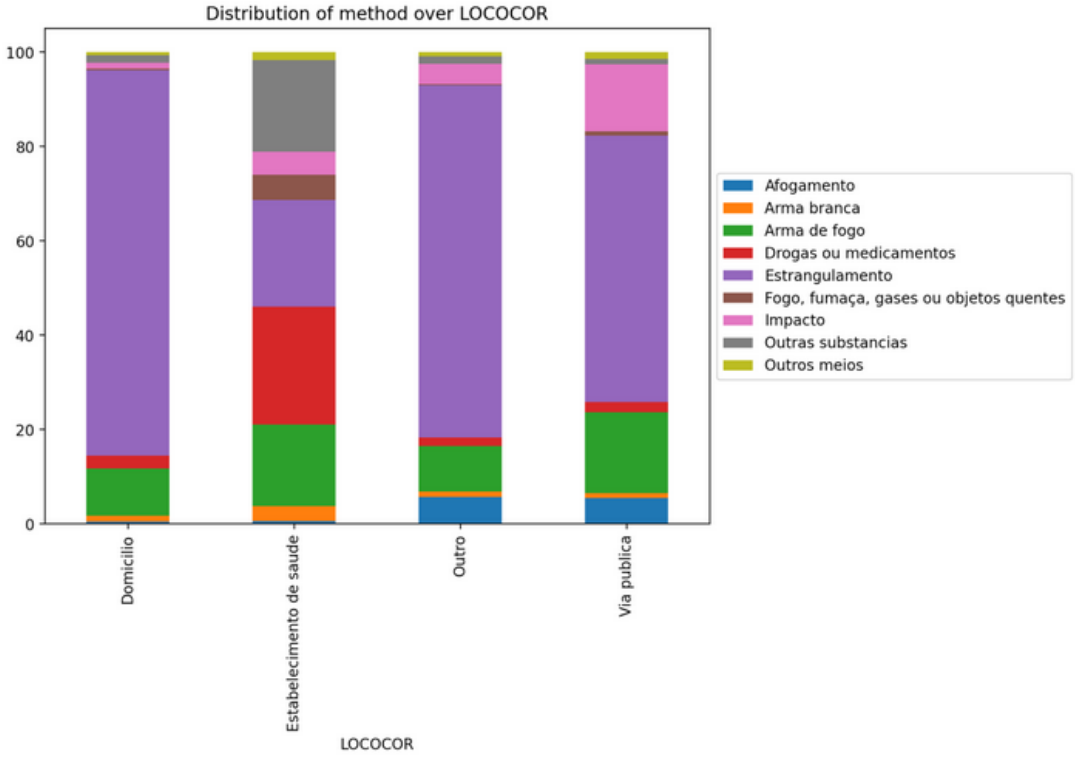


Figura 14 – Distribuição de métodos de suicídio por local de ocorrência (em %)

Outra relação interessante foi observada entre métodos de suicídio por local de ocorrência. A distribuição (em porcentagem) pode ser vista na Figura 14. Ocorrências em domicílio, em via pública ou em outros locais (exceto estabelecimentos de saúde) mostraram a mesma tendência identificada anteriormente, sendo estrangulamento a causa de mais de dois terços dos óbitos. No entanto, a situação foi bastante diferente em estabelecimentos de saúde: nesses locais, os casos de auto-intoxicação intencional por drogas, medicamentos ou outras substâncias representaram mais da metade dos suicídios. Essa informação sugere uma dificuldade ou até mesmo negligência quanto ao controle de acesso de medicamentos em hospitais e outros estabelecimentos do tipo.

A outra opção de visualização fornecida pela ferramenta possibilita a análise da dis-

tribuição geográfica dos atributos. Nas Figuras 15, 16 e 17 pode-se observar a taxa média de suicídios por 100 mil habitantes em cada município dos estados selecionados para esta análise.

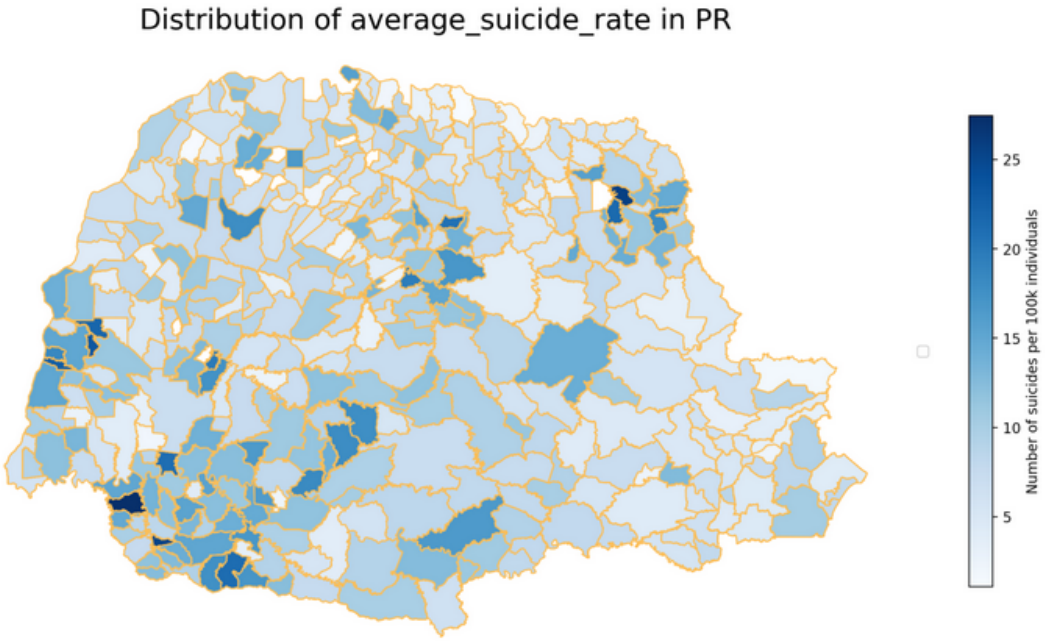


Figura 15 – Taxa média de suicídios por 100 mil habitantes (PR)

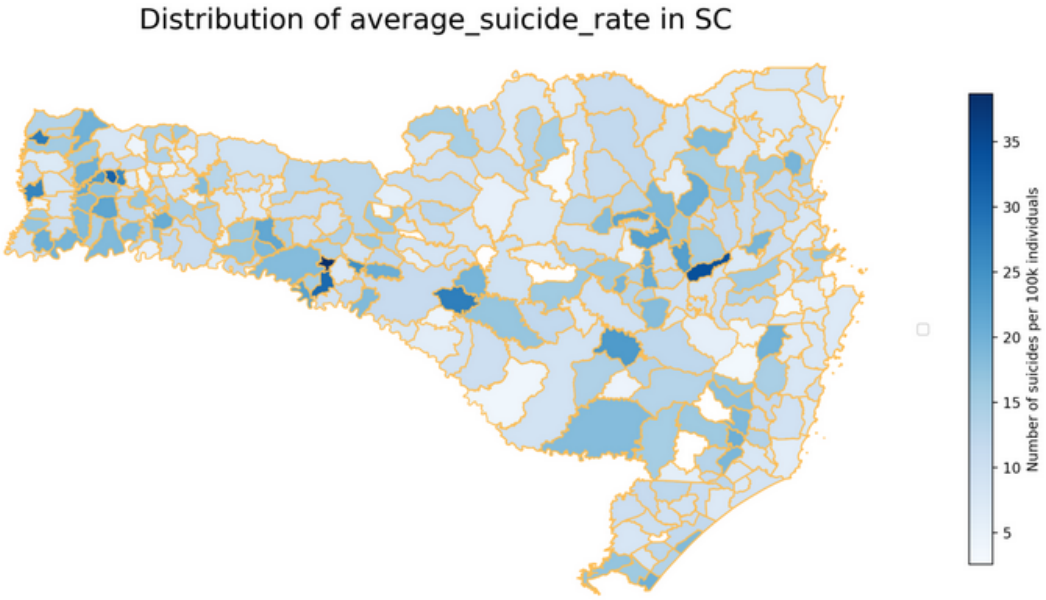


Figura 16 – Taxa média de suicídios por 100 mil habitantes (SC)

No Paraná, percebe-se uma concentração dos municípios com taxas mais altas no extremo oeste, na região de Foz do Iguaçu. Essa tendência se repete em Santa Catarina,

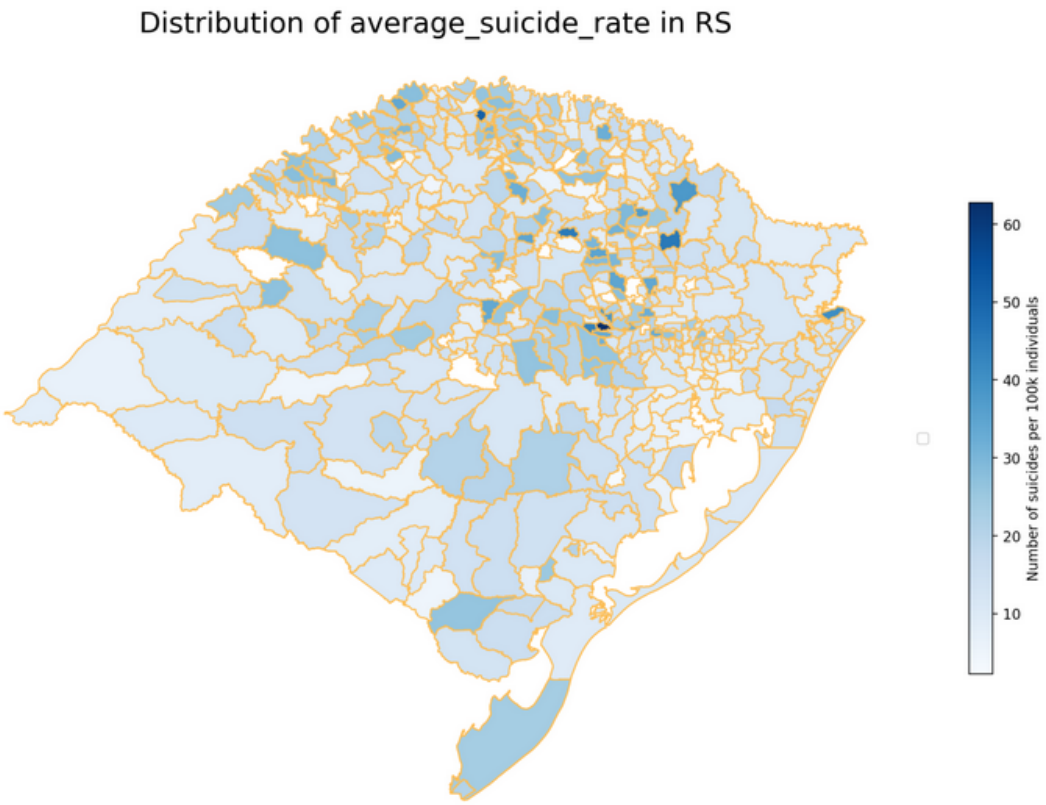


Figura 17 – Taxa média de suicídios por 100 mil habitantes (RS)

porém mais dispersa. Apesar da concentração de municípios com taxas de suicídio elevadas na região oeste e meio-oeste do estado, outros polos aparecem no Vale do Itajaí, próximo a Blumenau, e na Serra Catarinense. Já no Rio Grande do Sul, a dispersão geográfica das taxas de suicídio é bem mais acentuada. Nota-se, em geral, valores maiores no interior do estado, em especial na região centro-leste.

Uma vez familiarizado com os dados, o usuário da ferramenta pode prosseguir para a aplicação de *clustering* hierárquico. A Tabela 7 contém a lista de variáveis do conjunto de dados pré-processados selecionadas para esta análise específica. O método escolhido para a definição dos *clusters* foi *complete linkage*. O dendrograma resultante pode ser visto na Figura 18.

Observando o dendrograma, percebe-se que definir a distância limite de 40 unidades formaria quatro *clusters*, 50 unidades formaria três e, por sua vez, 60 unidades formaria apenas dois *clusters*. Utilizando a ferramenta para avaliar os *clusters* formados por esses três valores, vê-se na Figura 19 que os índices de silhueta e de Calinski-Harabasz (CH) indicam que a distância de 60 unidades forma *clusters* mais bem definidos.

Os gráficos dos coeficientes de silhueta (ver Figura 20) permitem a visualização da qualidade da definição dos *clusters*. Os pontos de dados com coeficientes mais baixos são aqueles que não estão bem caracterizados pelos *clusters* a que foram atribuídos e sugerem a existência de interseções entre os *clusters*.

Tabela 7 – Atributos selecionados

Atributo	**Descrição**
"SEXO"	Masculino ou feminino.
"IDADE"	Idade em anos
"RACACOR"	Raça/cor: branca, preta, amarela, parda ou indígena
"ESC"	Nível de escolaridade: sem escolaridade, ensino fundamental I, fundamental II, médio ou superior
"LOCOCOR"	Local de ocorrência: estabelecimento de saúde, domicílio, via pública ou outro
"ESTCIV"	Estado civil: solteiro, casado, viúvo, divorciado ou em união estável
"method"	Descrição da *causa mortis*, isto é, o método do suicídio
"age_group"	Faixa etária (em intervalos de 10 anos)
"facility_rate"	Estabelecimentos de saúde mental por 1000 habitantes no município
"season"	Estação do ano em que ocorreu o óbito
"weekday"	Dia da semana em que ocorreu o óbito
"day_period"	Período do dia em que ocorreu o óbito

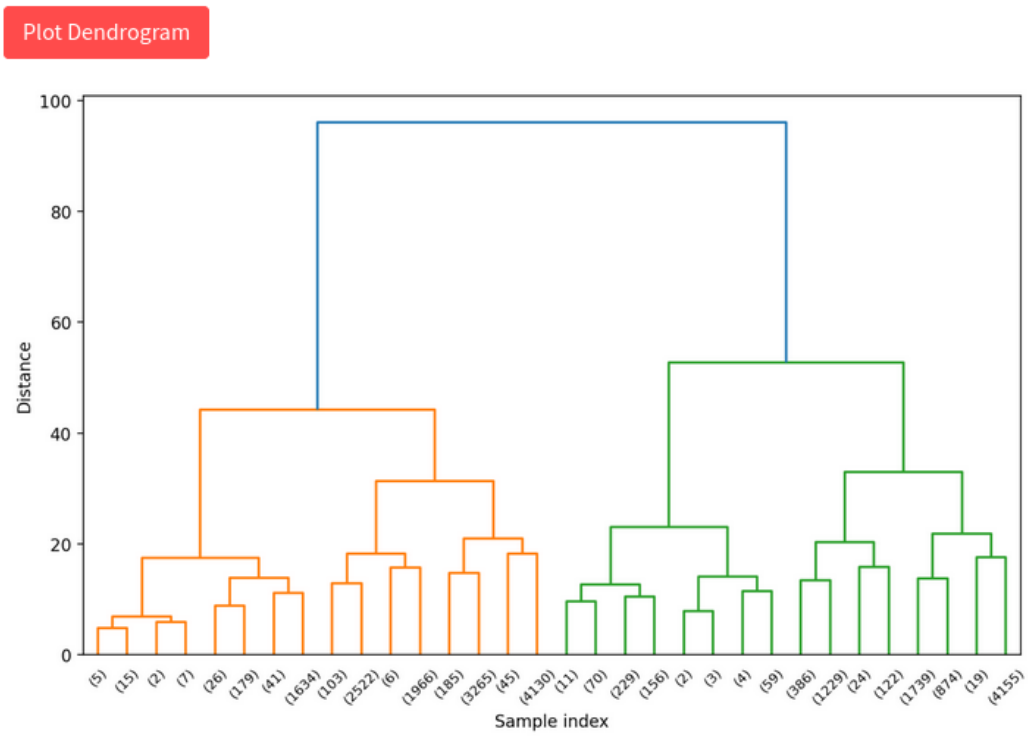
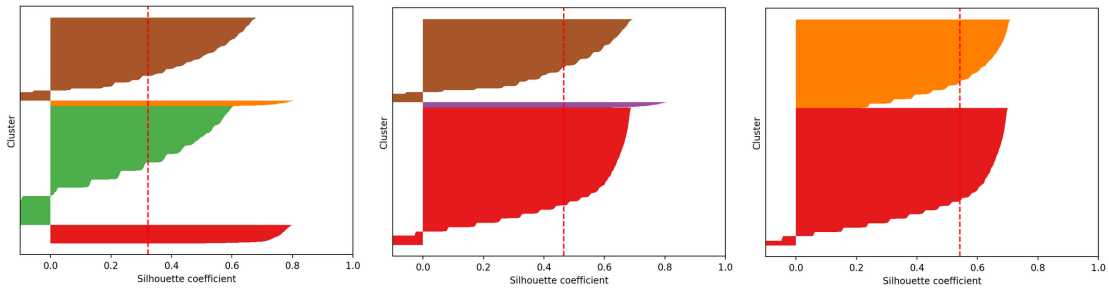


Figura 18 – Dendrograma para *complete linkage*

	Parameter	Clusters (k)	Silhouette score	CH score
0	40	4	0.3226	26,761.849
1	50	3	0.465	27,052.4734
2	60	2	0.5417	45,485.8071

Figura 19 – Avaliação dos *clusters* para diferentes parâmetros



(a) dist = 40 (b) dist = 50 (c) dist = 60

Figura 20 – Gráficos de coeficientes de silhueta para diferentes parâmetros

A escolha do limite de distância como 40 unidades formou quatro *clusters*, sendo um deles (representado pela cor laranja) bastante menor que os outros; além disso, os *clusters* representados pelas cores verde e marrom não foram bem definidos e o valor médio do índice de silhueta foi baixo (0.32). A avaliação da segunda alternativa, com um limite de distância de 50 unidades, formou três *clusters* e apresentou menor sobreposição entre eles. Assim como na primeira avaliação, observou-se a existência de um *cluster* muito pequeno e bem definido. O valor médio do índice de silhueta também foi significativamente maior (0.46). Por fim, o limite de distância de 60 unidades formou dois *clusters* de tamanho semelhante, com pouca sobreposição e o maior valor médio do índice de silhueta (0.54).

A seguir, os *clusters* formados nas diferentes situações são analisados e descritos por meio das opções de visualização disponíveis na ferramenta.

Com a distância de 40 unidades, os quatro *clusters* formados apresentam diferenças mais significativas entre os seguintes atributos: faixa etária ("age_group"), nível de escolaridade ("ESC") e estado civil ("ESTCIV").

A faixa etária foi separada entre os *clusters* com quase nenhuma intersecção (ver Figura 21). O primeiro *cluster* (n = 1909) contém indivíduos mais jovens, sendo 93% deles entre 10 e 19 anos. O segundo e maior *cluster* (n = 12222) agrupou vítimas de três faixas etárias consecutivas, com idades entre 20 e 49 anos. O terceiro e menor *cluster* (n = 534) possui os indivíduos acima de 80 anos, com 87% deles entre 80 e 89 anos. Por fim, o quarto *cluster* (n = 8548) juntou as três demais faixas etárias, com pessoas entre 50 e

79 anos de idade - contudo, apresentou uma concentração maior de vítimas entre 50 e 59 anos.

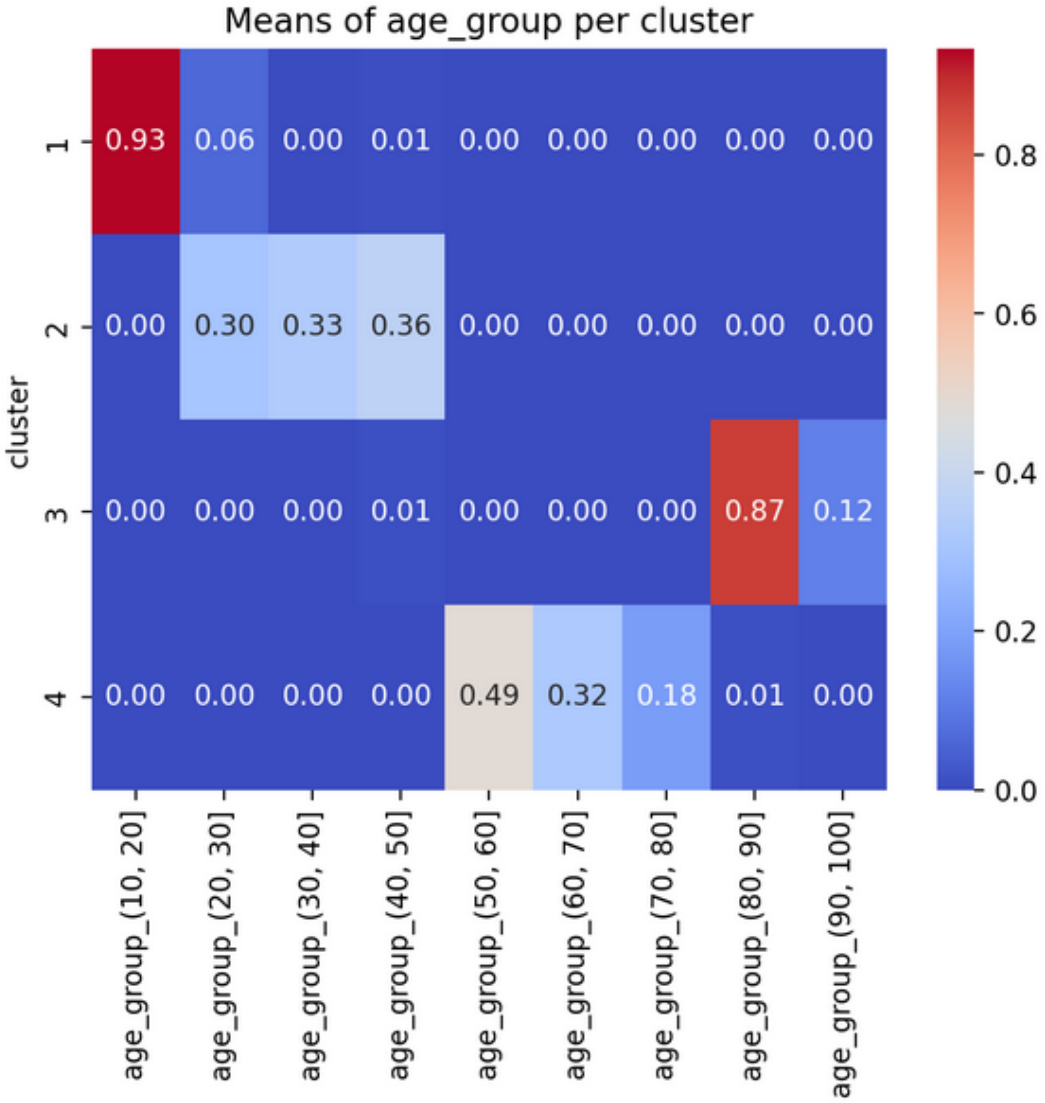


Figura 21 – Faixa etária por *cluster* (k = 4)

Quanto à escolaridade (ver Figura 22), observou-se um nível significativamente melhor entre indivíduos do *cluster* 1 - quase 50% deles possuíam ensino médio ou superior. Em contraste, o *cluster* 3 concentrou indivíduos com níveis mais baixos de escolaridade, dos quais 83% possuíam apenas ensino fundamental e 8% não possuíam escolaridade alguma. Os *clusters* 2 e 4 apresentaram semelhanças com os *clusters* 1 e 3, respectivamente. Contudo, notou-se que o *cluster* 2 agrupou a maior parte dos indivíduos com ensino superior.

Em relação ao estado civil (ver Figura 23), observou-se uma concentração de solteiros (97%) no primeiro *cluster*. O segundo *cluster* também apresentou maior concentração de solteiros, mas não tão sobressalente (63%). Houve uma aparente diluição das vítimas entre casados, divorciados e em união estável. O terceiro *cluster* agrupou quase todos os viúvos do conjunto de dados, que representam 46% deste agrupamento. Outros 36% eram

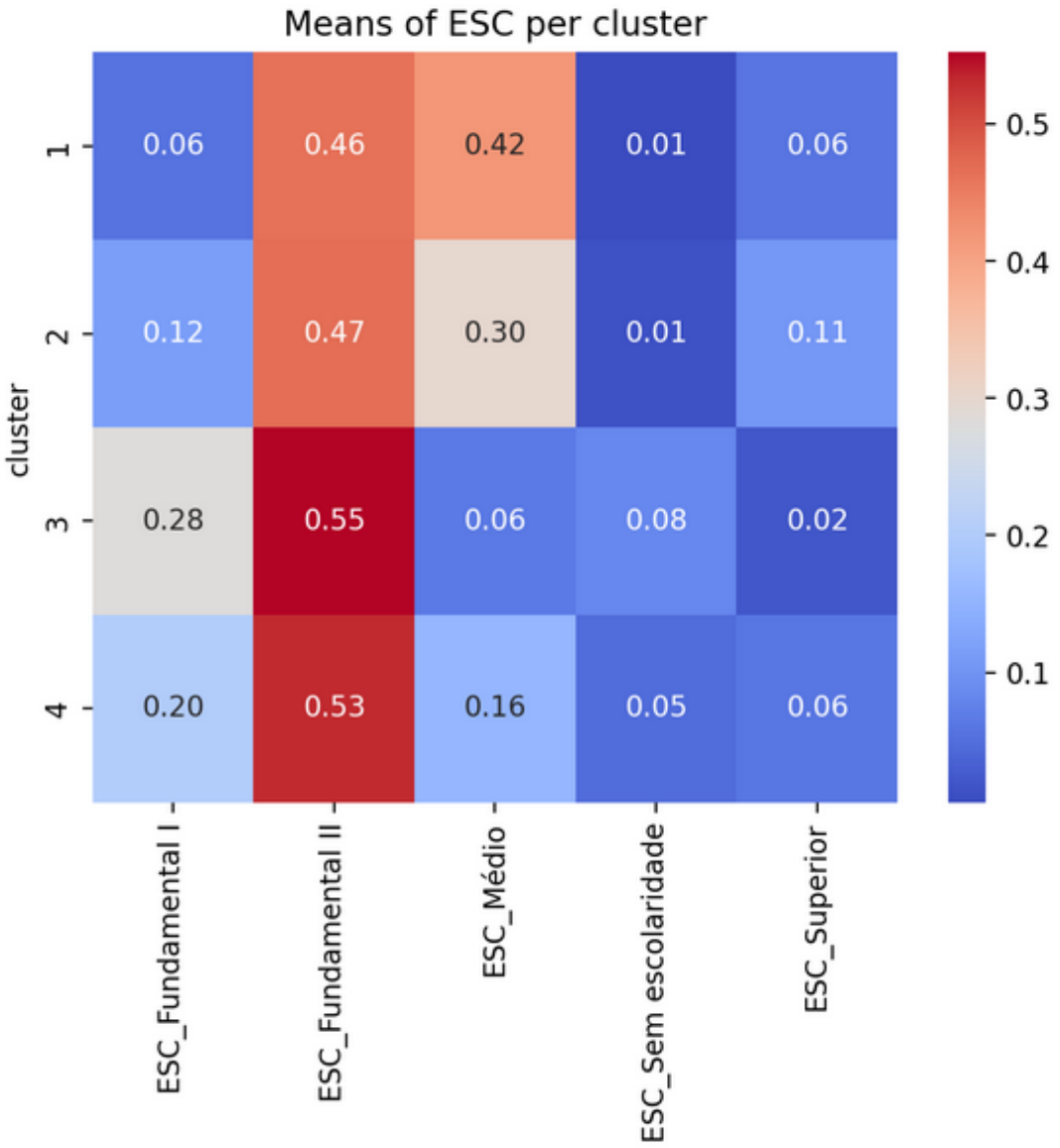


Figura 22 – Nível de escolaridade por *cluster* (k = 4)

casados. Por sua vez, mais da metade do quarto *cluster* foi composto por casados e ainda apresentou o maior percentual de divorciados (12%)

Os demais atributos não apresentaram diferenças muito significativas entre os *clusters* e, em geral, seguiram as distribuições observadas nos dados não-rotulados. A lista abaixo resume as características principais de cada *cluster*, para uma distância máxima de 40 unidades e k = 4.

- *Cluster* 1 (n = 1909): solteiros, entre 10 e 19 anos de idade, com alto nível de escolaridade.
- *Cluster* 2 (n = 12222): solteiros e casados, entre 20 e 49 anos de idade, com alto nível de escolaridade.

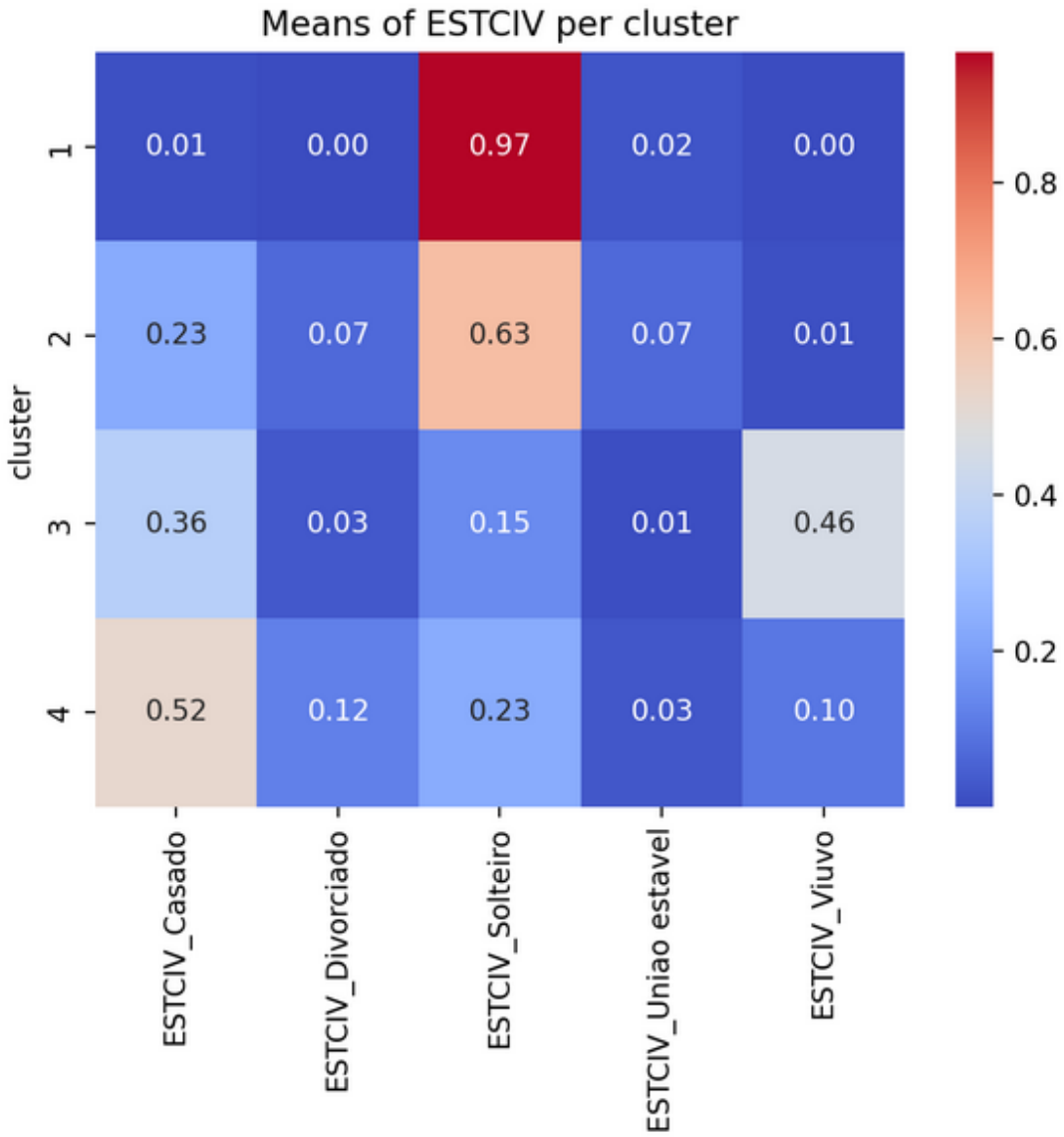


Figura 23 – Estado civil por *cluster* (k = 4)

- *Cluster* 3 (n = 534): casados e viúvos, acima de 80 anos de idade, com baixo nível de escolaridade.
- *Cluster* 4 (n = 8548): casados e divorciados, entre 50 e 79 anos de idade, com baixo nível de escolaridade.

Definindo a distância máxima como 50 unidades, foram formados três *clusters*. Como já indicado pela etapa de avaliação do *clustering* aplicado, houve pouca diferença entre os resultados obtidos com quatro *clusters* e, agora, com três. Através dos gráficos de coeficientes de silhueta na Figura 20, pode-se perceber que o maior *cluster* (representado em verde) possuía uma sobreposição significativa com outro *cluster*. Ao aumentar a distância limite, essa sobreposição deixou de existir, uma vez que os *clusters* 1 e 2 se juntaram. No entanto, os demais *clusters* se mantiveram intactos e, portanto, suas características

definitivas também não sofreram alterações. A lista seguinte descreve os três *clusters* restantes:

- *Cluster* 1 (n = 14131): solteiros e casados, entre 10 e 49 anos de idade, com alto nível de escolaridade.
- *Cluster* 2 (n = 534): casados e viúvos, acima de 80 anos de idade, com baixo nível de escolaridade.
- *Cluster* 3 (n = 8548): casados e divorciados, entre 50 e 79 anos de idade, com baixo nível de escolaridade.

De maneira análoga ao que ocorreu após o aumento da distância máxima intra-*cluster*, ao elevar esse valor para 60 unidades o pequeno *cluster* de indivíduos mais velhos foi assimilado pelo que anteriormente era o *cluster* 3. Nesse caso, os dois *clusters* formados podem ser descritos como:

- *Cluster* 1 (n = 14131): solteiros e casados, entre 10 e 49 anos de idade, com alto nível de escolaridade.
- *Cluster* 2 (n = 9082): casados, divorciados e viúvos, acima de 50 anos de idade, com baixo nível de escolaridade.

Apesar das métricas de avaliação indicarem melhor qualidade do *clustering*, para determinados contextos (como a identificação de grupos de risco) a assimilação desses pequenos *clusters* pode configurar perda de informação. Para continuar investigando padrões nos dados de suicídios, outras análises podem fazer uso de outros métodos de *clustering* hierárquico, como *average* ou *ward linkage*, além de selecionar outras variáveis, anos ou estados.

6 Conclusão

Ao longo deste trabalho, foi desenvolvida uma ferramenta web para análise e mineração de dados públicos de mortalidade por suicídio, coletados e disponibilizados pelo SUS através de seu departamento de informática. O objetivo principal que orientou este projeto foi a criação de uma plataforma interativa e intuitiva para facilitar a coleta, o enriquecimento, tratamento e a visualização dos dados, bem como a aplicação de algoritmos de *clustering*, capacitando usuários a explorar padrões, tendências e correlações no conjunto de dados.

O *back-end* da ferramenta foi feito em Python e utilizou diversas bibliotecas como Pandas, Matplotlib e SciPy para manipular e visualizar os dados. Para o desenvolvimento da interface, foi escolhido o *framework* Streamlit, cuja simplicidade sintática e variedade de funcionalidades para criação de *widgets* é ideal. Além da interface gráfica, os demais requisitos funcionais idealizados e descritos no Capítulo 4 foram implementados.

Para a coleta dos dados de mortalidade (SIM), de estabelecimentos de saúde (CNES) e de municípios brasileiros (IBGE), a ferramenta utiliza a biblioteca PySUS. O pré-processamento inclui a seleção de atributos dos conjuntos de dados originais, o enriquecimento semântico para melhorar a compreensão do usuário, a criação de novos atributos, o tratamento de dados faltantes ou incorretos e a codificação one-hot de atributos categóricos não-ordinais. Os dados pré-processados são armazenados localmente em formato CSV.

Quanto a análise dos dados, a ferramenta permite o cálculo de estatísticas descritivas, a visualização de gráficos customizáveis e a aplicação de algoritmos de *clustering* hierárquico. As opções de visualização são parte central deste projeto e incluem gráficos de barra, mapas de calor e mapas geoespaciais de estados brasileiros e seus municípios.

6.1 Considerações Finais

A implementação dos requisitos funcionais previstos cumpre o objetivo deste projeto, conferindo aos usuários uma maneira intuitiva, prática e eficiente de analisar e obter conhecimento sobre tendências, fatores e grupos de risco relacionados ao fenômeno do suicídio. Apesar disso, a ferramenta ainda possui diversas limitações e, tendo isso em mente, foi projetada para ser facilmente expansível.

Trabalhos futuros poderão aprofundar as capacidades de análise da ferramenta por meio da ampliação do processo de coleta, integração e enriquecimento dos dados e da disposição de outras opções de visualização e outros algoritmos de mineração de dados. Dados sobre tentativas de suicídio podem ser coletados a partir do Sistema de Informação

de Agravos de Notificação (SINAN), também disponibilizado pelo DATASUS, e outros atributos do CNES podem ser selecionados, tratados e utilizados para obter informações sobre a natureza (pública ou privada) e a qualidade da assistência médica em determinada região.

Referências Bibliográficas

AGGARWAL, C. C. *Data Mining: The Textbook*. [S.l.]: Springer, 2015. 13, 16, 17

BAE, S. M.; LEE, S. A.; LEE, S.-H. Prediction by data mining, of suicide attempts in korean adolescents: a national study. *Neuropsychiatric disease and treatment*, v. 11, 2015. 20, 25

CHOO, C. et al. Cluster analysis reveals risk factors for repeated suicide attempts in a multi-ethnic asian population. *Asian Journal of Psychiatry*, v. 8, p. 38–42, 2014. ISSN 18762018. 23, 25

DATASUS. *DATASUS: Departamento de Informática do SUS*. Online. <https://datasus.saude.gov.br/sobre-o-datasus/>. Acesso em: 12 jun. 2023. 18

HEERINGEN, K. V.; MANN, J. J. The neurobiology of suicide. *The Lancet Psychiatry*, Elsevier Ltd, v. 1, p. 63–72, 6 2014. ISSN 22150374. 10

LUDWIG, B. et al. Clustering suicides: A data-driven, exploratory machine learning approach. *European Psychiatry*, Cambridge University Press, v. 62, p. 15–19, 2019. 22, 25

MITCHELL, T. M. *Machine Learning*. [S.l.]: McGraw-Hill, 1997. 15

RUSSELL, S. J.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. [S.l.]: Pearson Education, 1995. 15

SILVA, D. A. da; MARCOLAN, J. F. Tentativa de suicídio e suicídio no brasil: análise epidemiológica. *Medicina (Ribeirão Preto)*, v. 54, n. 4, 2021. 11

SILVA, P. M. de Souza e; AUTRAN, M. M. M. de. Repositório datasus: Organização e relevância dos dados abertos em saúde para a vigilância epidemiológica. *P2P E INOVAÇÃO*, v. 6, n. 1, p. 50–59, 2019. 10

World Health Organization. Suicide in the world: global health estimates. *World Health Organization: Geneva*, 2019. 10

World Health Organization. Suicide worldwide in 2019: global health estimates. *World Health Organization: Geneva*, 2021. 10

World Health Organization. *Mental Disorders*. 2022. <https://www.who.int/news-room/fact-sheets/detail/mental-disorders>. Acesso em: 31 jul. 2022. 10

Apêndices

VSCODE_PRINT_SCRIPT_TAGS

Selected files

8 printable files

download.py
clustering.py
figures.py
util.py
Home.py
pages/1_Data_Description.py
pages/2_Cluster_Analysis.py
pages/3_Data_Dictionary.py

download.py

```python
"""
Functions to collect data.
"""

import pandas as pd
import numpy as np
import os

from pysus.online_data import SIM, CNES, IBGE, parquets_to_dataframe

import util

download_states = util.available_states
download_years = util.available_years

def get_SIM(states: list = download_states, years: list = download_years) -> pd.DataFrame:
    """
    Get preprocessed SIM data.
    """
    df_SIM: pd.DataFrame()
    try:
        df_SIM = pd.read_csv('./data/preprocessed_SIM.csv')
        print("get_SIM: preprocessed data found in cache.")
    except FileNotFoundError:
        print("get_SIM: preprocessed data not found in cache, working on it...")
        df_SIM_raw = get_db_raw('SIM', states=states, years=years)
        SIM_selection = ['DTOBITO', 'HORAOBITO', 'CAUSABAS', 'LOCOCOR', 'CODMUNRES', 'IDADE', 'SEXO', 'RACACOR', 'ESC', 'ESTCIV']
        df_SIM = df_SIM_raw[SIM_selection]
        df_SIM = df_SIM.rename(columns={'CODMUNRES': 'CODMUN'})
        # Fix: remove white spaces from data
        for col in df_SIM:
            df_SIM[col] = df_SIM[col].astype(str).apply(str.strip)
        # Select only suicide deaths
        df_SIM['CAUSABAS'] = df_SIM['CAUSABAS'].str[:3]
        df_SIM = df_SIM.loc[df_SIM['CAUSABAS'].isin(util.dict_methods.keys())]
        # Decode features
        df_SIM['IDADE'] = df_SIM['IDADE'].apply(util.decode_age)
```

```
        df_SIM['DTOBITO'] = df_SIM['DTOBITO'].apply(util.decode_date)
        translate_SIM(df_SIM)
        df_SIM['CODMUN'] = df_SIM['CODMUN'].astype(int)

        ### Feature Extraction ###
        # 'DTOBITO' -> 'ano_obito', 'dia_obito', 'mes_obito', 'fim_semana', 'feriado', 'estacao_ano'
        df_SIM['year'] = df_SIM['DTOBITO'].apply(util.get_year)
        df_SIM['month'] = df_SIM['DTOBITO'].apply(util.get_month)
        df_SIM['day'] = df_SIM['DTOBITO'].apply(util.get_day)
        df_SIM['season'] = df_SIM['DTOBITO'].apply(util.get_season)
        df_SIM['weekday'] = df_SIM['DTOBITO'].apply(util.get_weekday)
        df_SIM['holiday'] = df_SIM['DTOBITO'].apply(util.is_holiday, args=[1])
        # Get municipality data and calculate suicide rates
        df_muni = get_municipality()
        suicide_rates = []
        drop_list = []
        for year in years:
            # Get number of deaths per municipality and year
            suicides_year = df_SIM.loc[df_SIM['year'] == year]
            suicides_year = suicides_year['CODMUN'].value_counts().reset_index()
            suicides_year.columns = ['CODMUN', f'suicides_{year}']
            df_muni = df_muni.merge(suicides_year, how='left', on='CODMUN')
            df_muni[f'suicides_{year}'] = df_muni[f'suicides_{year}'].fillna(0)
            df_muni[f'suicide_rate_{year}'] = df_muni[f'suicides_{year}'] / df_muni['pop_muni'].astype(int) * 100000
            df_muni = df_muni.drop(columns=f'suicides_{year}')
            drop_list.append(f'suicide_rate_{year}')
            suicide_rates.append(f'suicide_rate_{year}')
        df_muni['average_suicide_rate'] = df_muni[suicide_rates].mean(axis=1)
        df_muni = df_muni.drop(columns=drop_list)
        # Merge with municipality dataframe on municipality code
        # 'CODMUN' -> 'state', 'name_muni', 'pop_muni', 'average_suicide_rate', 'facility_rate'
        df_SIM['state'] = df_SIM['CODMUN'].apply(util.get_state)
        df_SIM = df_SIM.merge(df_muni, how='left', on='CODMUN')
        df_SIM = df_SIM.drop(columns='num_facilities')
        # 'IDADE' -> 'age_group'
        grupos = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
        df_SIM['age_group'] = pd.cut(x=df_SIM['IDADE'], bins=grupos)
        # 'CAUSABAS' -> 'method'
        df_SIM['method'] = df_SIM['CAUSABAS'].apply(util.get_suicide_method)
        # 'HORAOBITO' -> 'periodo_dia'
        df_SIM['day_period'] = df_SIM['HORAOBITO'].apply(util.get_period)
        # Fix dtypes and save as .csv
        df_SIM = util.fix_numerical_dtypes(df_SIM)
        df_SIM.to_csv('./data/preprocessed_SIM.csv', index=False)
    return df_SIM

def get_CNES(states: list = download_states, years: list = download_years) -> pd.DataFrame:
    """
    Transforming CNES
    """
    df_CNES = pd.DataFrame()
    try:
        df_CNES = pd.read_csv('./data/preprocessed_CNES.csv')
        print("get_CNES: preprocessed data found in cache.")
    except:
```

```
        print("get_CNES: preprocessed data not found in cache, working on it...")
        df_CNES_raw = get_db_raw('CNES', states=states, years=years)
        CNES_selection = ['CNES', 'COMPETEN', 'CODUFMUN', 'COD_CEP', 'NATUREZA', 'VINC_SUS', 'TP_UNID', 'SERAP02P', 'SERAP02T']
        df_CNES = df_CNES_raw[CNES_selection]
        # Select only facilities that provide psychotherapy support (SADT) or Social Service
        df_CNES = df_CNES.loc[(df_CNES['TP_UNID'] == '39') | (df_CNES['SERAP02P'] == '1') | (df_CNES['SERAP02T'] == '1')]
        # Extracting feature 'year' from 'COMPETEN'
        df_CNES['year'] = df_CNES['COMPETEN'].str[:4]
        # Make it readable
        df_CNES['NATUREZA'] = df_CNES['NATUREZA'].astype('object').map({
                                '01': 'Publica', # MS
                                '02': 'Outra', # Outros órgãos
                                '03': 'Publica', # Autarquia
                                '04': 'Publica', # Fundação pública
                                '05': 'Publica', # Empresa pública
                                '06': 'Publica', # Organização pública
                                '07': 'Privada', # Empresa privada
                                '08': 'Privada', # Fundação privada
                                '09': 'Outra', # Cooperativa
                                '10': 'Privada', # Serviço Social autônomo
                                '11': 'Publica', # Entidade beneficente
                                '12': 'Outra', # Economia mista
                                '13': 'Outra' # Sindicato
                            })
        # Fix dtypes and save as .csv
        df_CNES = util.fix_numerical_dtypes(df_CNES)
        df_CNES.to_csv('./data/preprocessed_CNES.csv', index=False)
    return df_CNES

def get_municipality() -> pd.DataFrame:
    """
    Get preprocessed municipality data.
    """
    df_muni: pd.DataFrame()
    try:
        df_muni = pd.read_csv('./data/municipality_data.csv')
        print("get_municipality: preprocessed data found in cache.")
    except FileNotFoundError:
        print("get_municipality: preprocessed data not found in cache, working on it...")
        df_muni_raw = get_municipality_raw()
        # Select municipality code, name and population
        df_muni = df_muni_raw[['D1C', 'D1N', 'V']].rename(columns={'D1C': 'CODMUN', 'D1N': 'name_muni', 'V': 'pop_muni'})
        df_muni['CODMUN'] = df_muni['CODMUN'].astype(str).str[:-1].astype(int) # Remove verification digit
        df_muni['name_muni'] = df_muni['name_muni'].str.rsplit(' ', 2).str[0] # Remove state from municipality name
        # Get number of healthcare facilities from CNES
        df_CNES = get_CNES()
        facilities_muni = df_CNES['CODUFMUN'].value_counts().reset_index().astype(int)
        facilities_muni.columns = ['CODMUN', 'num_facilities']
        df_muni = df_muni.merge(facilities_muni, how='left', on='CODMUN')
        df_muni['num_facilities'] = df_muni['num_facilities'].fillna(0)
        df_muni['facility_rate'] = df_muni['num_facilities'] / df_muni['pop_muni'] * 1000
```

```
144         #df_muni['mental_healthcare'] =
    df_muni['facility_rate'].apply(util.healthcare)
145         # Fix dtypes and save as .csv
146         df_muni = util.fix_numerical_dtypes(df_muni)
147         df_muni.to_csv('./data/municipality_data.csv', index=False)
148     return df_muni
149
150 def get_db_raw(database: str, states: list = download_states, years: list =
    download_years) -> pd.DataFrame:
151     """
152     Read cached file containing the database. If no file is found, download it.
153     """
154     raw_df = pd.DataFrame()
155     try:
156         raw_df = pd.read_parquet(f'./data/rawdata/{database}.parquet')
157         print(f"get_db_raw: raw {database} data found in cache.")
158     except FileNotFoundError:
159         print(f"get_db_raw: raw {database} data not found in cache, downloading
    from DATASUS...")
160         raw_df = download_db(database, states, years)
161     return raw_df
162
163 def get_municipality_raw() -> pd.DataFrame:
164     """
165     Download municipality data from IBGE (code, name, population).
166     """
167     df_muni = pd.DataFrame()
168     try:
169         df_muni = pd.read_csv('./data/rawdata/municipality_raw.csv')
170         print("get_municipality_raw: raw municipality data found in cache.")
171     except FileNotFoundError:
172         print("get_municipality_raw: raw municipality data not found in cache,
    downloading from IBGE...")
173         df_muni = IBGE.get_sidra_table(table_id=1505, territorial_level=6,
    variables=93,
174                                        classification=12017, categories=0,
    headers='n')
175         # Save as .csv
176         df_muni.to_csv('./data/rawdata/municipality_raw.csv', index=False)
177     return df_muni
178
179 def download_db(database: str, states: list = download_states, years: list =
    download_years) -> pd.DataFrame:
180     """
181     Download parquets with PySUS and concatenate them all into a single
    dataframe.
182     """
183     parquets = []
184     if database == "SIM":
185         parquets = SIM.download(states=states, years=years)
186     elif database == "CNES":
187         parquets = CNES.download(group='ST', states=states, years=years,
    months=1)
188     else:
189         raise ValueError("download.get_database: available databases are SIM
    and CNES\n")
190     print(f"download_db: {database} parquet files downloaded.")
191     df_list = []
192     for path in parquets:
193         df_list.append(parquets_to_dataframe(path))
194     raw_df = pd.concat(df_list, ignore_index=True)
```

```
    raw_df.to_parquet(f'./data/rawdata/{database}.parquet')
    print(f"download_db: {database} data concatenated and stored.")
    return raw_df

def parquets_to_df(data_dir: str) -> pd.DataFrame:
    """
    Read all parquet files inside a directory into one pandas dataframe.
    """
    df_list = []
    for item in os.listdir(data_dir):
        item_path = os.path.join(data_dir, item)
        # If item is a parquet file, read and append to list
        if os.path.isfile(item_path) and item.endswith(".parquet"):
            df = pd.read_parquet(item_path)
            df_list.append(df)
        # If item is a directory, look for parquet files inside it
        elif os.path.isdir(item_path):
            df = parquets_to_df(item_path)
            df_list.append(df)
    concatenated_df = pd.concat(df_list, ignore_index=True)
    return concatenated_df


def get_ICD() -> pd.DataFrame:
    """
    Get ICD10 table, filter for suicide codes and rename key ('CID10') to merge with SIM.
    This was meant to get code descriptions but they're written in a very weird format, with all words abbreviated.
    """
    df_CID = SIM.get_CID10_table()
    df_CID = df_CID.loc[df_CID['CID10'].isin(util.icd_suicide_codes)]
    df_CID = df_CID[['CID10', 'DESCR']].rename(columns={'CID10': 'CAUSABAS', 'DESCR': 'method'})
    return df_CID

def get_CBO() -> pd.DataFrame:
    df_CBO = SIM.get_ocupations()
    df_CBO = df_CBO.rename(columns={'CODIGO': 'OCUP', 'DESCRICAO': 'occupation'})
    return df_CBO

def translate_SIM(df_SIM: pd.DataFrame) -> None:
    """
    Translate SIM attribute values to natural language.
    """
    df_SIM['LOCOCOR'] = df_SIM['LOCOCOR'].map({
                            '1': "Estabelecimento de saude",
                            '2': "Estabelecimento de saude",
                            '3': "Domicilio",
                            '4': "Via publica",
                            '5': "Outro"
                        })
    df_SIM['SEXO'] = df_SIM['SEXO'].map({
                            '1': "Masculino",
                            '2': "Feminino"
                        })
    df_SIM['RACACOR'] = df_SIM['RACACOR'].map({
                            '1': "Branca",
                            '2': "Preta",
```

```
                            '3': "Amarela",
                            '4': "Parda",
                            '5': "Indigena"
                        })
    # Note: 'ESC2010' is only available after 2011, hence rendered useless
    df_SIM['ESC'] = df_SIM['ESC'].map({
                            '1': "Sem escolaridade",
                            '2': "Fundamental I",
                            '3': "Fundamental II",
                            '4': "Médio",
                            '5': "Superior"
                        })
    df_SIM['ESTCIV'] = df_SIM['ESTCIV'].map({
                            '1': "Solteiro",
                            '2': "Casado",
                            '3': "Viuvo",
                            '4': "Divorciado",
                            '5': "Uniao estavel"
                        })


```

clustering.py

```
'''
Functions for cluster analysis.
'''

import pandas as pd
import numpy as np
from scipy.cluster.hierarchy import linkage, fcluster
from scipy.spatial.distance import pdist
from sklearn.metrics import silhouette_score, calinski_harabasz_score

from util import impute_df
from figures import plot_silhouette

def evaluate_clustering(df, linkage_matrix, dist_values, gen_plots=False) -> tuple:
    """
    Evaluate clustering results for a list of distance threshold values.
    """
    results = []
    plots = []
    for dist in dist_values:
        labels = fcluster(linkage_matrix, t=dist, criterion='distance')
        n_clusters = len(np.unique(labels))
        sil = silhouette_score(df.values, labels)
        ch = calinski_harabasz_score(df.values, labels)
        results.append((dist, n_clusters, sil, ch))
        #results.append(f"Parameter = {dist}: Silhouette score = {sil}; CH score = {ch}; Number of clusters = {n_clusters}")
        if gen_plots:
            plots.append(plot_silhouette(df.values, labels))
    results = pd.DataFrame(results, columns=['Parameter', 'Clusters (k)', 'Silhouette score', 'CH score'])
    return (results, plots)
```

```
31
32 def apply_linkage(df, selected_method='complete') -> tuple:
33     """
34     Apply clustering algorithm. Return one-hot encoded data and the linkage
   matrix.
35     """
36     # Impute missing data
37     df = impute_df(df)
38     # Apply one-hot encoding to categorical features
39     categorical_features = df.select_dtypes(exclude=
   [np.number]).columns.tolist()
40     df = pd.get_dummies(df, columns=categorical_features, dtype=float)
41     # Get the linkage matrix
42     dist_matrix = pdist(df)
43     linkage_matrix = linkage(dist_matrix, method=selected_method)
44     return (df, linkage_matrix)
45
46 def apply_labels(df, linkage_matrix, dist: int) -> pd.DataFrame:
47     labels = fcluster(linkage_matrix, t=dist, criterion='distance')
48     df['cluster'] = labels
49     df.to_csv('./data/labeled_data.csv', index=False)
50     return df
```

figures.py

```
1  """
2  Functions for data visualization.
3  """
4
5  import pandas as pd
6  import matplotlib.pyplot as plt
7  import seaborn as sns
8  import geobr
9  from scipy.cluster.hierarchy import dendrogram
10 from sklearn.metrics import silhouette_score, silhouette_samples
11
12 from unidecode import unidecode
13
14 import util
15
16 available_states = util.available_states
17 available_years = util.available_years
18
19 def plot_dendrogram(linkage_matrix, levels: int) -> plt.figure:
20     """
21     Generate dendrogram.
22     """
23     fig, ax = plt.subplots(figsize=(10, 6))
24     dendrogram(linkage_matrix, truncate_mode='level', p=levels)
25     ax.set_xlabel('Sample index')
26     ax.set_ylabel('Distance')
27     return fig
28
29 def plot_silhouette(df, labels) -> plt.figure:
30     """
31     Generate silhouette plot.
32     """
```

```
    silhouette_avg = silhouette_score(df, labels)
    silhouette_vals = silhouette_samples(df, labels)
    y_lower, y_upper = 0, 0
    fig, ax = plt.subplots()
    for i, cluster in enumerate(set(labels)):
        cluster_silhouette_vals = silhouette_vals[labels == cluster]
        cluster_silhouette_vals.sort()
        y_upper += len(cluster_silhouette_vals)
        color = plt.cm.Set1(i / len(set(labels)))
        ax.barh(range(y_lower, y_upper), cluster_silhouette_vals, height=1.0,
                edgecolor='none', color=color)
        y_lower += len(cluster_silhouette_vals)

    ax.axvline(silhouette_avg, color="red", linestyle="--")
    ax.set_yticks([])
    ax.set_xlim([-0.1, 1])
    ax.set_xlabel("Silhouette coefficient")
    ax.set_ylabel("Cluster")
    return fig

def two_feature_barplot(df: pd.DataFrame, plot_feature: str, axis_feature: str,
percent_y: bool = False):
    """
    Generate a barplot of the distribution of a feature over another feature.
    """
    # Group the data by year and method
    df = df.groupby([axis_feature, plot_feature]).size().unstack(fill_value=0)
    if percent_y:
        df = df.apply(lambda x: x / x.sum() * 100, axis=1)
    fig, ax = plt.subplots(figsize=(8, 6))
    df.plot(kind='bar', stacked=True, legend=True, ax=ax)
    ax.set_xlabel(f"{axis_feature}")
    ax.set_title(f"Distribution of {plot_feature} over {axis_feature}")
    ax.legend(loc='center left', bbox_to_anchor=(1, 0.5))
    return fig

def feature_cluster_heatmap(df, feature: str) -> plt.figure:
    """
    Generate heatmap for feature distribution per cluster.
    """
    features = ["cluster"]
    # Select all columns related to the feature (handle one-hot encoded
features)
    for column in df.columns:
        if feature in column:
            features.append(column)

    # Create a new dataframe with the cluster labels and the selected feature
    df = df.loc[:, features]
    means = df.groupby('cluster').mean()
    fig, ax = plt.subplots()
    ax = sns.heatmap(means, cmap='coolwarm', annot=True, fmt='.2f')
    ax.set_title("Means of %s per cluster" % feature)
    return fig

def state_geomap(df: pd.DataFrame, state, feature: str) -> plt.figure:
    """
    Generate a map of the state with distribution of a feature per municipality
    """
```

```
    state_map = geobr.read_municipality(code_muni=state)
    #state_map['code_muni'] = (state_map['code_muni'] / 10).astype(int)
    #df['CODMUN'] = df['CODMUN'].astype(int)
    #state_map = state_map.merge(df, how='left', left_on='code_muni',
right_on='CODMUN')
    state_map['name_muni'] = state_map['name_muni'].str.lower()
    df['name_muni'] = df['name_muni'].str.lower()
    state_map = state_map.merge(df, how='left', on='name_muni')
    fig, ax = plt.subplots(figsize=(15, 15), dpi=300)
    state_map.plot(
        column=feature,
        cmap="Blues",
        edgecolor="#FEBF57",
        legend=True,
        legend_kwds={
            "label": "Number of suicides per 100k inhabitants",
            "orientation": "vertical",
            "shrink": 0.4,
        },
        ax=ax,
    )
    ax.set_title(f"Distribution of {feature} in {state}", fontsize=20)
    ax.legend(loc='center left', bbox_to_anchor=(1, 0.5))
    ax.axis("off")
    return fig

```

util.py

```
"""
Utility functions.
"""

import pandas as pd
import numpy as np

import holidays
import calendar
from datetime import date, timedelta
#from pysus.online_data import SIM
from pysus.preprocessing import decoders

available_states = ['PR', 'SC', 'RS']
available_years = [2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019]

dict_states = {
    '11': 'RO', '12': 'AC', '13': 'AM', '14': 'RR', '15': 'PA', '16': 'AP',
'17': 'TO',
    '21': 'MA', '22': 'PI', '23': 'CE', '24': 'RN', '25': 'PB', '26': 'PE',
'27': 'AL', '28': 'SE', '29': 'BA',
    '31': 'MG', '32': 'ES', '33': 'RJ', '34': 'SP',
    '41': 'PR', '42': 'SC', '43': 'RS',
    '50': 'MS', '51': 'MT', '52': 'GO', '53': 'DF'
}

# Suicide codes (ICD10)
#    X60 to X69 -> self-poisoning
```

```
#    X70 to X84 -> self-inflicted injury
# Grouping causes and rewriting descriptions...
dict_methods = {
    'X60' : "Drogas ou medicamentos", # Analgesicos nao-opiaceos
    'X61' : "Drogas ou medicamentos", # Anticonvulsionantes e psicotropicos
    'X62' : "Drogas ou medicamentos", # Narcoticos e psicodislepticos
    'X63' : "Drogas ou medicamentos", # Outras substancias farmacologicas de acao sobre o sist. nervoso
    'X64' : "Drogas ou medicamentos", # Outras drogas e substancias nao especificadas
    'X65' : "Drogas ou medicamentos", # Alcool
    'X66' : "Outras substancias", # Solventes organicos e seus vapores
    'X67' : "Outras substancias", # Outros gases e vapores
    'X68' : "Outras substancias", # Pesticidas
    'X69' : "Outras substancias", # Substancias nao identificadas
    'X70' : "Estrangulamento",
    'X71' : "Afogamento",
    'X72' : "Arma de fogo", # Arma de fogo de mão (pistolas e revólveres)
    'X73' : "Arma de fogo", # Arma de fogo de calibre maior (espingardas, carabinas)
    'X74' : "Arma de fogo", # Arma de fogo nao especificada
    'X75' : "Outros meios", # Dispositivos explosivos
    'X76' : "Fogo, fumaça, gases ou objetos quentes", # Fumaca e fogo
    'X77' : "Fogo, fumaça, gases ou objetos quentes", # Vapor de agua, gases ou objetos quentes
    'X78' : "Arma branca", # Objeto cortante ou penetrante
    'X79' : "Arma branca", # Objeto contundente
    'X80' : "Impacto", # Precipitacao de lugar elevado
    'X81' : "Impacto", # Permanencia diante de objeto em movimento
    'X82' : "Impacto", # Impacto de um veículo motorizado
    'X83' : "Outros meios",
    'X84' : "Outros meios" # Nao especificado
}

# Seasons
Y = 2000 # Any leap year
seasons = [("Summer", (date(Y,  1,  1),  date(Y,  3, 20))),
           ("Autumn", (date(Y,  3, 21),  date(Y,  6, 20))),
           ("Winter", (date(Y,  6, 21),  date(Y,  9, 22))),
           ("Spring", (date(Y,  9, 23),  date(Y, 12, 20))),
           ("Summer", (date(Y, 12, 21),  date(Y, 12, 31)))]

br_holidays = holidays.country_holidays('BR', years=available_years)

def decode_age(age: str) -> int:
    return decoders.decodifica_idade_SIM(age, 'Y')

def decode_date(date_str: str) -> date:
    return decoders.decodifica_data_SIM(date_str)

def get_state(codmun) -> str:
    return dict_states.get(str(codmun)[:2])

def get_suicide_method(icd: str) -> str:
    return dict_methods.get(icd)

def get_season(data: date) -> str:
    data = data.replace(year=Y)
    for season, (start, end) in seasons:
        if start <= data <= end:
```

```
            return season

def get_period(hora: int) -> str:
    # Valores alfanuméricos ou nulos
    if hora.isnumeric() == False:
        return np.nan
    # Madrugada (00:00 - 05:59)
    elif 0 <= int(hora) < 600:
        return "Night"
    # Manhã (06:00 - 11:59)
    elif 600 <= int(hora) < 1200:
        return "Morning"
    # Tarde (12:00 - 17:59)
    elif 1200 <= int(hora) < 1800:
        return "Afternoon"
    # Noite (18:00 - 23:59)
    elif 1800 <= int(hora) < 2400:
        return "Evening"
    # Valores numéricos inválidos (ex. 9999)
    return np.nan

def get_weekday(data: date) -> str:
    weekday = calendar.weekday(data.year, data.month, data.day) # Segunda = 0,
...
    if weekday == 0:
        return "Monday"
    elif weekday == 1:
        return "Tuesday"
    elif weekday == 2:
        return "Wednesday"
    elif weekday == 3:
        return "Thursday"
    elif weekday == 4:
        return "Friday"
    elif weekday == 5:
        return "Saturday"
    elif weekday == 6:
        return "Sunday"
    return np.nan

def get_year(data: date) -> int:
    return data.year

def get_month(data: date) -> int:
    return data.month

def get_day(data: date) -> int:
    return data.day

def is_holiday(data: date, interval: int) -> bool:
    """
    Verify if date is a holiday or close to one.
    """
    delta = timedelta(days=interval)
    for holiday in br_holidays:
        if holiday-delta <= data <= holiday+delta:
            return True
    return False

```

```python
def healthcare(facility_rate: float) -> str:
    if facility_rate == 0: return "None"
    elif facility_rate >= 5: return "High"
    elif facility_rate >= 2: return "Moderate"
    else: return "Low"

def impute_column(column: pd.Series) -> pd.Series:
    """
    Impute missing data in a column. Median for numerical features, mode for categorical.
    """
    missing_data = column.isna().sum()

    if missing_data / len(column) < 0.3: # 0.03
        # Impute numerical column with median
        if np.issubdtype(column.dtype, np.number):
            column = column.fillna(column.median())
        # Impute categorical column with mode
        else: # column.dtype == 'object'
            column = column.fillna(column.mode()[0])

    # TO DO: Impute columns with more than X% missing data using KNN?
    else:
        if np.issubdtype(column.dtype, np.number):
            pass
        else: # column.dtype == 'object'
            pass
    return column

def impute_df(df: pd.DataFrame) -> pd.DataFrame:
    """
    Impute missing data in all columns of a dataframe.
    """
    for column_name in df.columns:
        df[column_name] = impute_column(df[column_name])
    return df

def column_nulls(df: pd.DataFrame) -> list:
    """
    Get a list of the columns that have missing data in a dataframe.
    """
    columns = []
    for col in df.columns:
        missing_data = df[col].isna().sum()
        if missing_data > 0:
            columns.append(col)
            print(f"{col}: {missing_data}")
    return columns

def fix_numerical_dtypes(df: pd.DataFrame) -> pd.DataFrame:
    """
    Fix numerical column dtypes.
    """
    numerical_features = df.select_dtypes(include=[np.number]).columns
    for feature in numerical_features:
        df[feature] = pd.to_numeric(df[feature], errors='coerce')
    return df

```

Home.py

```
"""
Streamlit web application.
"""

import streamlit as st
import pandas as pd

st.set_page_config(page_title="SIM analytics")

st.markdown(
    """
    ### About
    This app is a tool to analyze suicide data in Brazil.
    It provides a preprocessed mortality database and functions to visualize the data, as well as a simple framework to apply clustering algorithms.
    """
)

```

pages/1_Data_Description.py

```
"""
Web app: data description functions.
"""

import streamlit as st
import numpy as np
import pandas as pd

import download as dl
import util
from figures import two_feature_barplot, state_geomap

st.write(
    """
    ### Data Description
    **Select data from the preprocessed dataset.**
    """
)

plot_opt = ["Feature distribution over another feature (stacked barplot)",
            "Feature distribution per municipality (geospatial map)"]

preprocessed_data = dl.get_SIM()
selected_states = st.multiselect("States: ", options=util.available_states, default=util.available_states)
selected_years = st.multiselect("Years: ", options=util.available_years, default=util.available_years)
selected_data = preprocessed_data.loc[(preprocessed_data['state'].isin(selected_states)) &

(preprocessed_data['year'].isin(selected_years))]
categorical_features = selected_data.select_dtypes(exclude=[np.number]).columns.tolist()
```

```
29 categorical_features.append("year")
30
31 if st.button(label="Describe", type="primary"):
32     st.write("Descriptive statistics for numerical features: ",
   selected_data.describe(include=np.number),
33              "Descriptive statistics for categorical features: ",
   selected_data.describe(include=['O']))
34
35 st.write("**Visualize selected data.**")
36
37 plot = st.selectbox("Select a visualization option: ", options=plot_opt)
38
39 if plot == plot_opt[0]:
40     # "Feature distribution per year (barplot)"
41     plot_feature = st.selectbox("Plot feature: ", options=categorical_features)
42     axis_feature = st.selectbox("Axis feature: ", options=categorical_features)
43     percent_y = st.checkbox("Percentage y-axis?")
44     #age_group = st.selectbox("Age group: ",
   options=preprocessed_data['age_group'].unique())
45     if st.button(label="Plot", type="primary"):
46         st.pyplot(two_feature_barplot(selected_data, plot_feature, axis_feature,
   percent_y))
47
48 elif plot == plot_opt[1]:
49     # "Feature distribution per municipality"
50     st.write("This option will display a map for each selected state.")
51     feature = st.selectbox("Feature: ", options=preprocessed_data.columns)
52     if st.button(label="Plot", type="primary"):
53         for state in selected_states:
54             st.pyplot(state_geomap(preprocessed_data, state, feature))
55
```

pages/2_Cluster_Analysis.py

```
1  """
2  Web app: cluster analysis functions.
3  """
4
5  import streamlit as st
6  import pandas as pd
7  import numpy as np
8  import re
9
10 import download as dl
11 import clustering as cl
12 import util
13 from figures import plot_dendrogram, feature_cluster_heatmap, state_geomap
14
15 st.write(
16     """
17     ### Cluster Analysis
18     **Select data, apply the desired hierarchical clustering method and visualize
   dendrogram.**
19     """
20 )
21
22 preprocessed_data = dl.get_SIM()
```

```
selected_states = st.multiselect("States: ", options=util.available_states, default=util.available_states)
selected_years = st.multiselect("Years: ", options=util.available_years, default=util.available_years)
default_columns = ['IDADE', 'LOCOCOR', 'SEXO', 'RACACOR', 'ESC', 'ESTCIV', 'age_gr
'method', 'season', 'day_period', 'weekday', 'facility_rate']
selected_feats = st.multiselect("Features: ", options=preprocessed_data.columns.to_list(), default=default_columns)
selected_method = st.selectbox("Clustering method: ", options=['Single (nearest po
                                                               'Complete (farthest point)',
                                                               'Average (UPGMA)',
                                                               'Weighted (WPGMA)',
                                                               'Centroid (UPGMC)',
                                                               'Median (WPGMC)',
                                                               'Ward']).split()[0].lower()
selected_data = preprocessed_data[selected_feats].loc[(preprocessed_data['state'].isin(selected_st
&

(preprocessed_data['year'].isin(selected_years))]

if st.button(label="Plot Dendrogram", type='primary'):
    (onehot_data, linkage_matrix) = cl.apply_linkage(selected_data, selected_metho
    dendro = plot_dendrogram(linkage_matrix, levels=4)
    st.pyplot(dendro)

# --------

st.write(
    """
    **Evaluate cluster quality.**\n
    Provide a list of distance threshold values to test. The dendrogram y-axis can
insights on these values.
    """
)

list_input = st.text_input("Insert a list of values separated by commas:")
gen_plots = st.checkbox("Generate silhouette coefficient plots?")

collect_numbers = lambda x : [int(i) for i in re.split("[^0-9]", x) if i != ""]
dist_values = collect_numbers(list_input)
if st.button(label="Evaluate", type='primary'):
    (onehot_data, linkage_matrix) = cl.apply_linkage(selected_data, selected_metho
    (results, plots) = cl.evaluate_clustering(onehot_data, linkage_matrix, dist_va
gen_plots)
    st.write(results)
    if gen_plots:
        for plot in plots:
            st.pyplot(plot)

st.write("""
    **Set the distance threshold and label the data.**\n
    A column named "cluster" will be added to the dataset.
    """
)

labeled_data = pd.read_csv('./data/labeled_data.csv')
dist_threshold = st.number_input(label="Distance threshold:", min_value=1, step=1)
if st.button(label="Apply", type='primary'):
    (onehot_data, linkage_matrix) = cl.apply_linkage(selected_data, selected_metho
    labeled_data = cl.apply_labels(onehot_data, linkage_matrix, dist_threshold)
```

```
75
76  st.write(
77      """
78      Current clusters:
79      """,
80      labeled_data['cluster'].value_counts()
81  )
82
83  categorical_features = labeled_data.select_dtypes(exclude=[np.number]).columns.tol
84
85  plot_opt = ["Feature distribution per cluster (heatmap)",
86              "Geographic distribution of clusters (geospatial map)"]
87
88  plot = st.selectbox("Select a plot to visualize: ", options=plot_opt)
89  if plot == plot_opt[0]:
90      feature = st.selectbox("Feature:", options=selected_feats)
91      if st.button(key="heatmap", label="Plot", type="primary"):
92          st.pyplot(feature_cluster_heatmap(labeled_data, feature))
93  elif plot == plot_opt[1]:
94      state = st.selectbox("Feature:", options=categorical_features)
95      if st.button(key="geomap", label="Plot", type="primary"):
96          st.pyplot(state_geomap(labeled_data, state))
97
```

pages/3_Data_Dictionary.py

```
1   """
2   Web app: data dictionary.
3   """
4
5   import streamlit as st
6   import pandas as pd
7
8   def data_dict():
9       columns = ['Variable name', 'Type', 'Description']
10      data_dict = [
11          ('DTOBITO', 'datetime', "Date of death"),
12          ('HORAOBITO', 'string', "Time of death"),
13          ('CAUSABAS', 'string', "Cause of death codes as defined by ICD"),
14          ('LOCOCOR', 'string', "Place of death"),
15          ('CODMUN', 'int', "Municipality of residence codes as defined by IBGE"),
16          ('IDADE', 'int', "Age"),
17          ('SEXO', 'string', "Sex"),
18          ('RACACOR', 'string', "Race/color as classified by IBGE"),
19          ('ESC', 'string', "Highest level of education"),
20          ('ESTCIV', 'string', "Civil status"),
21          ('age_group', 'category', "Age group"),
22          ('method', 'string', "Suicide method"),
23          ('name_muni', 'string', "Municipality of residence's name"),
24          ('pop_muni', 'int', "Municipality of residence's population"),
25          ('facility_rate', 'float', "Number of healthcare facilities with mental
    health support per 1000 inhabitants in the municipality"),
26          ('average_suicide_rate', 'float', "Number of suicides per 100.000
    inhabitants in the municipality"),
27          ('year', 'int', "Year of death"),
28          ('month', 'int', "Month of death"),
29          ('day', 'int', "Day of death"),
```

```
        ('season', 'string', "Season of the year of death"),
        ('weekday', 'string', "Weekday of death"),
        ('holiday', 'bool', "Death happened on a holiday?"),
        ('period', 'string', "Day period of death")
    ]
    df_dict = pd.DataFrame(data_dict, columns=columns)
    df_dict.to_csv('./data/data_dict.csv', index=False)
    return df_dict

st.write(
    """
    ### Data Dictionary
    <div style='text-align:justify;'>
    The original databases used in this project are provided by DATASUS (Unified Health System's IT Department).
    SIM (Mortality Information System) data includes cause of death as classified by ICD and the victims' demographics;
    CNES (National Registry of Healthcare Facilities) contains a wide variety of data on all healthcare facilities in the country, regardless of their legal nature or whether they integrate SUS.
    More information on these and other databases is available <a href="https://datasus.saude.gov.br/">here</a>.
    <br><br>
    After selecting and extracting features, cleaning, transforming and linking the data, the dataset contains the following attributes:
    </div>
    """,
    unsafe_allow_html=True
)

data_dict = data_dict()

st.table(data_dict)

```

Desenvolvimento de Ferramenta de Análise e Mineração de Dados de Suicídio do DATASUS

Luis H. G. Stemmer[1], **Mateus Grellert**[2]

[1] Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)
CEP 88040-900 – Trindade – Florianópolis – SC

[2]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
CEP 90040-060 – Farroupilha – Porto Alegre – RS

luis.stemmer@grad.ufsc.br, mateus.grellert@ufsc.br

Abstract. *Suicide is the cause of over 700 thousand deaths every year around the world. Suicide prevention is a very complex subject due to the large quantity of possible factors that are hard to define and measure, with uncertain and inconsistent correlation levels. The main objective of this project is the development of a tool that, through an intuitive graphic interface, facilitates the process of analysis and mining of suicide data in Brazil, provided by SUS. The tool performs data collection and preprocessing and offers a variety of functionalities to calculate descriptive statistics and visualize graphs, in addition to enabling the application of hierarchical clustering algorithms and the evaluation of clusters.*

Resumo. *O suicídio é a causa de mais de 700 mil mortes por ano ao redor do mundo. A prevenção do suicídio é um tema bastante complexo em razão da grande quantidade de fatores difíceis de definir e mensurar, com graus de correlação incertos e inconsistentes. Este projeto tem como objetivo principal o desenvolvimento de uma ferramenta que, através de uma interface gráfica intuitiva, facilite o processo de análise e mineração de dados de suicídio no Brasil, disponibilizados pelo SUS. A ferramenta realiza a coleta e o pré-processamento dos dados, oferece uma variedade de funcionalidades para cálculo de estatísticas descritivas e visualização de gráficos, além de possibilitar a aplicação de algoritmos de clustering hierárquico e a avaliação da qualidade dos clusters.*

1. Introdução

O suicídio está entre as pricipais causas de mortes evitáveis no Brasil e no mundo há anos [World Health Organization 2021]. Segundo a Organização Mundial da Saúde (OMS), é a causa de mais de 700.000 mortes por ano - uma a cada 40 segundos. Em 2019, o suicídio representou 1,3% do total de óbitos no mundo e 1,7% no continente americano, sendo que 77% das mortes ocorreram em países de média ou baixa renda. No Brasil, o suicídio foi a terceira principal causa de morte de jovens brasileiros entre 15 e 29 anos e o país figura consistentemente entre as 10 nações com maior número absoluto de casos [World Health Organization 2019, World Health Organization 2021].

O cenário torna-se ainda mais preocupante após a eclosão da pandemia de COVID-19, período em que observou-se uma intensa deterioração dos quadros de saúde mental. Ainda de acordo com a OMS[World Health Organization 2022], em apenas um

ano desde o início da pandemia houve um aumento de 26% e 28% em casos de ansiedade e depressão, respectivamente; condições bastante associadas à ideação suicida.

A complexidade das interações entre fatores agravantes do fenômeno do suicídio [Heeringen and Mann 2014], somada ao estigma relacionado não só a esse fenômeno, mas também a quaisquer transtornos mentais e à psiquiatria como um todo, dificulta a identificação e o tratamento preventivo de indivíduos com tendências suicidas. Nesse contexto, técnicas de aprendizado de máquina são uma ferramenta poderosa, já que possuem ótima capacidade de reconhecimento de padrões e viabilizam análises eficientes de grandes quantidades de dados.

A fim de melhorar a disponibilidade, qualidade e variedade de dados sobre saúde pública no Brasil, o Departamento de Informática do Sistema Único de Saúde (DATASUS), órgão associado ao Ministério da Saúde, tem centralizado e mantido diversos bancos de dados desde sua criação, em 1991. O repositório de dados é alimentado por uma rede de sistemas de informações em saúde, responsáveis pela coleta de dados de diferentes áreas de abrangência do SUS, como o Sistema de Informações sobre Mortalidade (SIM) e o Cadastro Nacional de Estabelecimentos de Saúde (CNES) [Souza e Silva and Autran 2019]. Os dados são disponibilizados por meio de planilhas eletrônicas padronizadas no *website* do DATASUS e, mais recentemente, tem sido desenvolvidas também interfaces de programação para simplificar a coleta e o tratamento desses dados. Através do Sistema IBGE de Recuperação Automática (SIDRA), pode-se ainda obter dados agregados e resultados de pesquisas fornecidos pelo Instituto Brasileiro de Geografia e Estatística (IBGE).

Este trabalho visa analisar, transformar e integrar dados coletados por meio desses sistemas, utilizar algoritmos de *clustering* para identificar fatores e grupos de risco no que diz respeito ao suicídio e produzir uma ferramenta de alto nível de abstração com uma interface gráfica intuitiva para coletar, selecionar, visualizar e aplicar outros algoritmos sobre os dados pré-processados.

2. Desenvolvimento da Ferramenta

A Figura 1 apresenta o fluxo de coleta, pré-processamento e *clustering* hierárquico executado pela ferramenta no formato de uma aplicação *web*.

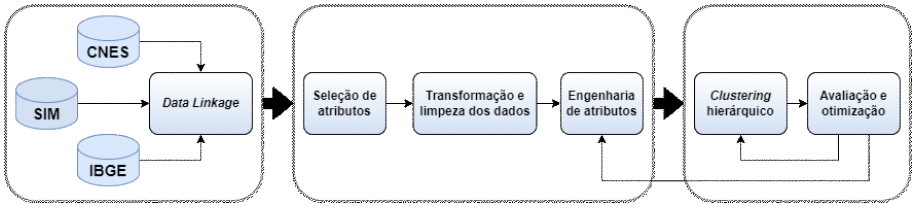


Figure 1. Fluxo de execução da ferramenta desenvolvida

A ferramenta foi projetada com a intenção de possibilitar que usuários realizem análises em tempo real sem a necessidade de instalar programas complexos. Segue o modelo de arquitetura cliente-servidor, utilizando Python para processamento e visualização dos dados no *back-end* e o *framework* de desenvolvimento web Streamlit para implementar uma interface web.

A página principal permite ao usuário, de maneira simples e intuitiva, selecionar atributos de interesse, especificar intervalos de tempo e escolher entre múltiplas possibilidades de visualização. Em uma segunda página, dedicada à aplicação de *clustering* hierárquico sobre os dados pré-processados, o usuário pode escolher o método a ser utilizado, visualizar o dendrograma resultante e obter métricas de avaliação dos *clusters*, como o coeficiente de silhueta. O processo de desenvolvimento foi bastante simplificado pelo Streamlit, que disponibiliza *widgets* prontos para criar elementos interativos.

No *back-end*, a biblioteca PySUS é importada para fazer a coleta de dados de diferentes bancos do DATASUS e do IBGE e armazená-los na memória. Outras bibliotecas como Pandas e NumPy são essenciais para a integração, transformação e enriquecimento dos dados originais e, para gerar as diversas visualizações, foram utilizadas ainda bibliotecas como Matplotlib e Seaborn.

2.1. Requisitos Funcionais

1. Interface de usuário:
    - Implementar uma interface clara e intuitiva, fácil de utilizar;
    - Possibilitar a coleta, o processamento e a visualização de dados por meio de *widgets* como caixas de seleção e botões.
2. Coleta e pré-processamento de dados:
    - Permitir a coleta dos dados não tratados de bancos do DATASUS;
    - Oferecer funções para pré-processamento dos dados;
    - Armazenar os dados coletados localmente para melhorar a performance.
3. Análise de dados
    - Prover filtros para personalização da análise;
    - Calcular estatísticas descritivas de atributos selecionados;
    - Implementar análise de *clusters* customizável, com funções de avaliação para auxiliar na otimização de hiperparâmetros.
4. Opções de visualização
    - Disponibilizar uma variedade de opções de visualização dos dados, incluindo gráficos de barra, gráficos de dispersão e mapas geoespaciais.
    - Permitir a customização dos gráficos.

3. Resultados

A ferramenta[1] coleta, transforma e armazena os dados pré-processados assim que é executada. Suas funcionalidades estão separadas em duas páginas principais, sendo uma para realizar análises descritivas e auxiliar no entendimento dos dados não-rotulados e outra para realizar análise de *clusters*.

A Figura 2 mostra a interface da primeira página. Primeiramente, o usuário filtra os dados por estados e anos e, ao clicar no botão ”Describe”, vê duas tabelas com estatísticas descritivas para atributos numéricos e categóricos, respectivamente. Em seguida, gráficos personalizados podem ser visualizados. Existem duas opções de visualização disponíveis: um mapa dos estados selecionados e seus municípios, apresentando a

[1]A ferramenta está disponível em `https://simexplorer.streamlit.app` e o código-fonte está disponível no repositório `https://github.com/lust2k/SIM_Explorer.git`

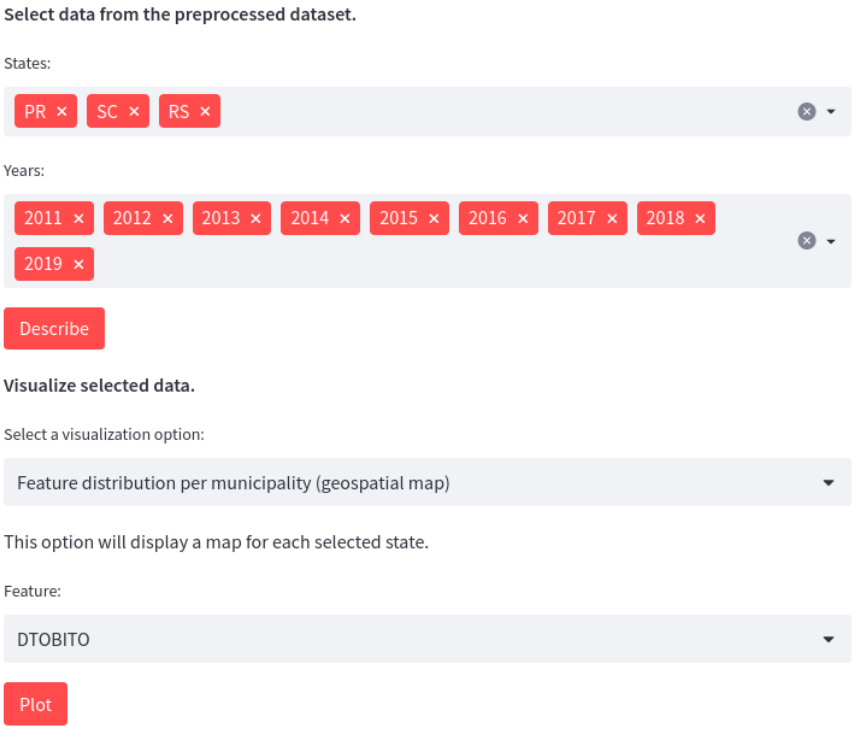


Figure 2. Página para análise descritiva e entendimento dos dados pré-processados

distribuição geográfica de um atributo; e um gráfico de barras com a distribuição de um atributo em relação a outro.

A segunda página da ferramenta é dedicada para a aplicação de técnicas de *clustering* hierárquico (ver Figura 3). Nela, além de ser possível filtrar os dados por estados e anos, o usuário também pode selecionar atributos do conjunto de dados pré-processados e o método de definição de *clusters*. Ao clicar no botão ”Plot Dendrogram”, o algoritmo será aplicado e o dendrograma resultante será mostrado.

Para rotular os dados, é necessário ainda definir um limite de distância para formação dos *clusters*. A ferramenta auxilia a otimizar esse parâmetro, possibilitando a avaliação dos *clusters* formados por uma lista de valores diferentes. Basta inserir os valores desejados no campo de texto, separados por vírgulas, e clicar no botão ”Evaluate”. Métricas de avaliação como os índices de silhueta e de Calinski-Harabasz (CH) serão calculadas e impressas na tela. Caso a caixa de seleção acima do botão esteja marcada, serão gerados gráficos de coeficientes de silhueta.

Uma vez escolhido o limite de distância intra-*cluster*, o usuário pode informá-lo e a ferramenta irá efetivamente rotular os dados, adicionando um atributo que identifica a que *cluster* cada instância pertence. Por fim, opções de visualização dos dados rotulados são disponibilizadas. É possível observar a distribuição geográfica dos *clusters* em mapas dos estados e seus municípios, e também a distribuição de cada atributo por *cluster* através de mapas de calor.

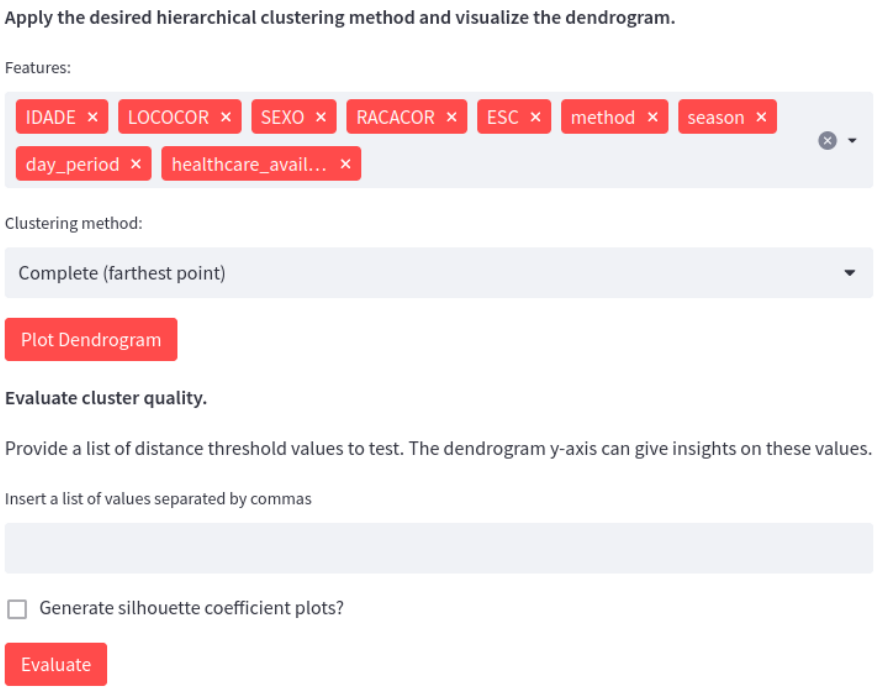


Figure 3. Página para análise de *clusters*

3.1. Análise de Suicídios no Sul do Brasil

Esta subseção ilustra as funcionalidades da ferramenta e as informações que fornece. Para esta análise, foram selecionados todos os dados tratados neste trabalho. Isto é, os três estados da região sul do Brasil - Paraná, Santa Catarina e Rio Grande do Sul - e os anos entre 2011 e 2020.

Primeiramente, a ferramenta desenvolvida foi utilizada para o entendimento dos dados. As estatísticas descritivas para variáveis numéricas e categóricas foram calculadas por meio da biblioteca Pandas. Em seguida, foram observadas as distribuições de atributos em relação a outros utilizando uma das opções de visualização disponíveis - gráficos de barra customizáveis. Por exemplo, a distribuição de métodos de suicídio por faixa etária pode ser vista na Figura 4.

Como indicado, o estrangulamento prevalece como método mais comum para todas as idades. Desenhando o gráfico com as mesmas variáveis, mas com o eixo y indicando porcentagens, algumas tendências podem ser vistas com maior clareza. Mortes por auto-intoxicação intencional através de drogas, medicamentos ou outras substâncias foram mais comuns entre jovens, em particular aqueles com idades entre 20 e 29 anos. Já na população mais velha, nota-se um aumento percentual no uso de armas de fogo.

Uma vez familiarizado com os dados, o usuário da ferramenta pode prosseguir para a aplicação de *clustering* hierárquico. Neste exemplo, o método escolhido para a definição dos *clusters* foi *complete linkage* e o dendrograma resultante pode ser visto na Figura 5.

Observando o dendrograma e utilizando a ferramenta para avaliar os *clusters* formados por diferentes parâmetros, obtém-se os gráficos de coeficientes de silhueta mostra-

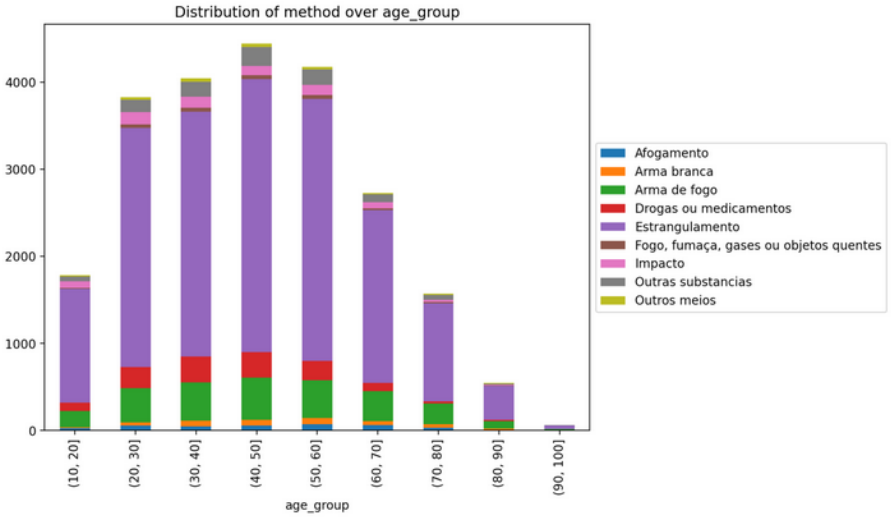


Figure 4. Distribuição de métodos de suicídio por faixa etária

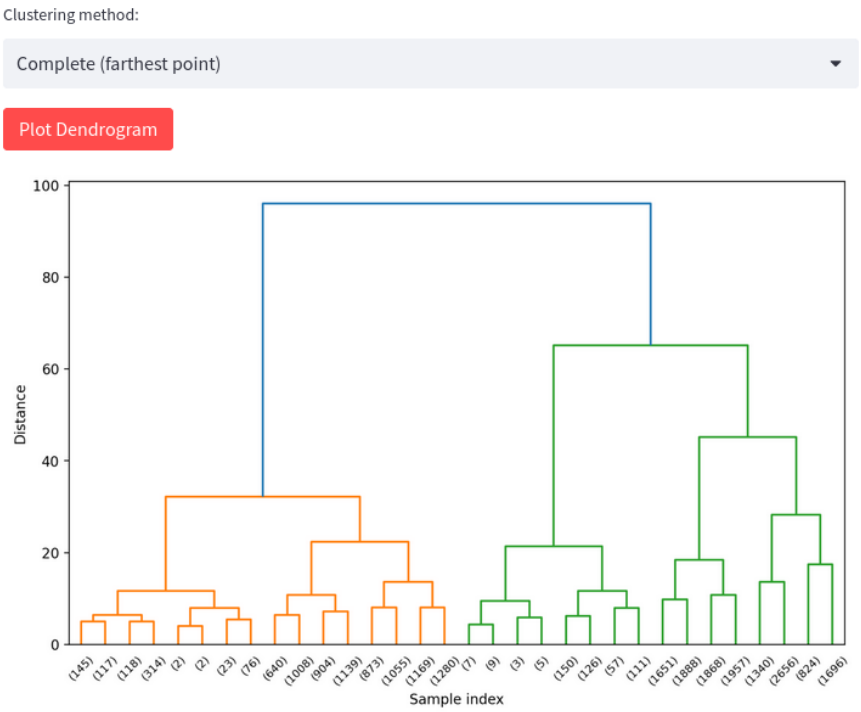
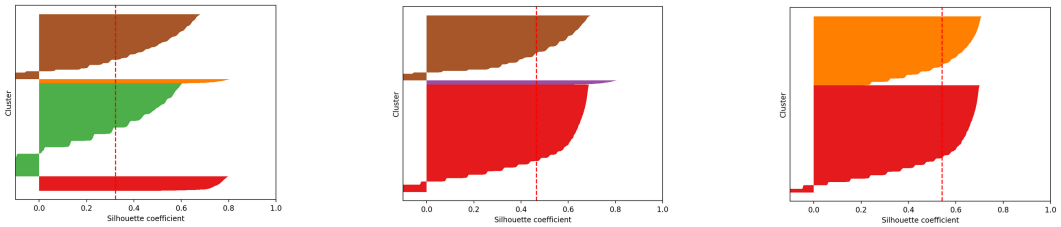


Figure 5. Dendrograma para *complete linkage*

dos na Figura 6. A escolha do limite de distância como 40 unidades formou quatro *clusters*, sendo um deles (representado pela cor laranja) bastante menor que os outros; além disso, os *clusters* representados pelas cores verde e marrom não foram bem definidos e o valor médio do índice de silhueta foi baixo (0.32). A avaliação da segunda alternativa, com um limite de distância de 50 unidades, formou três *clusters* e apresentou menor sobreposição entre eles. Assim como na primeira avaliação, observou-se a existência de um *cluster* muito pequeno e bem definido. O valor médio do índice de silhueta também foi significativamente maior (0.46). Por fim, o limite de distância de 60 unidades formou



(a) dist = 40 (b) dist = 50 (b) dist = 60

Figure 6. Gráficos de coeficientes de silhueta para diferentes parâmetros

dois *clusters* de tamanho semelhante, com pouca sobreposição e o maior valor médio do índice de silhueta (0.54).

4. Conclusão

Ao longo deste trabalho, foi desenvolvida uma ferramenta web para análise e mineração de dados públicos de mortalidade por suicídio, coletados e disponibilizados pelo SUS através de seu departamento de informática. O objetivo principal que orientou este projeto foi a criação de uma plataforma interativa e intuitiva para facilitar a coleta, o enriquecimento, tratamento e a visualização dos dados, bem como a aplicação de algoritmos de *clustering*, capacitando usuários a explorar padrões, tendências e correlações no conjunto de dados.

O *back-end* da ferramenta foi feito em Python e utilizou diversas bibliotecas como Pandas, Matplotlib e SciPy para manipular e visualizar os dados. Para o desenvolvimento da interface, foi escolhido o *framework* Streamlit, cuja simplicidade sintática e variedade de funcionalidades para criação de *widgets* é ideal. Além da interface gráfica, os demais requisitos funcionais idealizados foram implementados.

Para a coleta dos dados de mortalidade (SIM), de estabelecimentos de saúde (CNES) e de municípios brasileiros (IBGE), a ferramenta utiliza a biblioteca PySUS. O pré-processamento inclui a seleção de atributos dos conjuntos de dados originais, o enriquecimento semântico para melhorar a compreensão do usuário, a criação de novos atributos, o tratamento de dados faltantes ou incorretos e a codificação one-hot de atributos categóricos não-ordinais. Os dados pré-processados são armazenados localmente em formato CSV.

Quanto a análise dos dados, a ferramenta permite o cálculo de estatísticas descritivas, a visualização de gráficos customizáveis e a aplicação de algoritmos de *clustering* hierárquico. As opções de visualização são parte central deste projeto e incluem gráficos de barra, mapas de calor e mapas geoespaciais de estados brasileiros e seus municípios.

Trabalhos futuros poderão aprofundar as capacidades de análise da ferramenta por meio da ampliação do processo de coleta, integração e enriquecimento dos dados e da disposição de outras opções de visualização e outros algoritmos de mineração de dados. Dados sobre tentativas de suicídio podem ser coletados a partir do Sistema de Informação de Agravos de Notificação (SINAN), também disponibilizado pelo DATASUS, e outros atributos do CNES podem ser selecionados, tratados e utilizados para obter informações sobre a natureza (pública ou privada) e a qualidade da assistência médica em determinada região.

5. References

References

Heeringen, K. V. and Mann, J. J. (2014). The neurobiology of suicide. *The Lancet Psychiatry*, 1:63–72.

Souza e Silva, P. M. d. and Autran, M. M. M. d. (2019). Repositório datasus: Organização e relevância dos dados abertos em saúde para a vigilância epidemiológica. *P2P E INOVAÇÃO*, 6(1):50–59.

World Health Organization (2019). Suicide in the world: global health estimates. *World Health Organization: Geneva*.

World Health Organization (2021). Suicide worldwide in 2019: global health estimates. *World Health Organization: Geneva*.

World Health Organization (2022). Mental disorders. `https://www.who.int/news-room/fact-sheets/detail/mental-disorders`. Acesso em: 31 jul. 2022.