ERCEMAPI

Escola Regional de Computação do Ceará, Maranhão e Piauí

LIVRO DE MINICURSOS
XI EDIÇÃO/2023

ORGANIZADORES

DIEGO ROCHA LIMA

GUILHERME ALVARO R.M. ESMERALDO

Organização:

UFMA Universidade Federal do Maranhão

UFPI 1971

UFCA Universidade Federal do Cariri

Universidade Federal do Ceará

Instituto Federal Maranhão

Instituto Federal Piauí

Instituto Federal Ceará

Realização:

SBC Sociedade Brasileira de Computação

Instituto Federal Ceará Campus Aracati

ERCEMAPI
Escola Regional de Computação do Ceará, Maranhão e Piauí

XI Escola Regional de Computação do Ceará, Maranhão e Piauí

23 a 25 de novembro de 2023

LIVRO DE MINICURSOS

ERCEMAPI 2023

Organização do Livro

DIEGO ROCHA LIMA (IFCE)

GUILHERME ALVARO R. M. ESMERALDO (IFCE)

Sociedade Brasileira de Computação

Porto Alegre

2023

Sociedade Brasileira de Computação – SBC
Av. Bento Gonçalves, 9500. Setor 4 - Prédio 43.412 - Sala 219.
Bairro Agronomia. Porto Alegre - RS.
CEP: 91.509-900.

CNPJ: 29.532.264/0001-78
http://www.sbc.org.br

Dados Internacionais de Catalogação na Publicação (CIP)

E65 Escola Regional de Computação do Ceará, Maranhão e Piauí (11. : 23 – 25 novembro 2023 : Aracati)
Minicursos da ERCEMAPI 2023 [recurso eletrônico] / organização: Diego Rocha Lima ; Guilherme Alvaro R. M. Esmeraldo. Dados eletrônicos. – Porto Alegre: Sociedade Brasileira de Computação, 2023.
79 p. : il. : PDF ; 12.8 MB

Modo de acesso: World Wide Web. Inclui bibliografia
ISBN 978-85-7669-595-0 (e-book)

1. Computação – Brasil – Evento. 2. Segurança da informação. 3. Análise de dados. I. Lima, Diego Rocha. II. Esmeraldo, Guilherme Alvaro R. M.. III. Sociedade Brasileira de Computação. VI. Título.

CDU 004(063)

Ficha catalográfica elaborada por Annie Casali – CRB-10/2339
Biblioteca Digital da SBC – SBC OpenLib

Índices para catálogo sistemático:

1. Ciência e tecnologia dos computadores : Informática – Publicação de conferências, congressos e simpósios etc. ... 004(063)

Secretarias Regionais

Danielo Gonçalves Gomes (UFC – SR Ceará)
Davi Viana (UFMA – SR Maranhão)
Eduilson Lívio Neves da Costa Carneiro (IFPI – SR Piauí)

Coordenação Geral do Evento

Diego Rocha Lima (IFCE)
Guilherme Alvaro R. M. Esmeraldo (IFCE)

Coordenação do Comitê do Programa de Minicursos

Otávio Alcântara de Lima Júnior (IFCE)
Evandro José dos Santos Diniz (IFMA)
Romuere Veloso e Silva (UFPI)

Comitê do Programa de Minicursos

Atslands Rocha (UFC)
Camila H. S. Oliveira (UFCA)
Francisco Jose Silva (UFMA)
Omar Andres Carmona Cortes (IFMA)
Saulo Oliveira (IFCE)
Thiago Alves Rocha (IFCE)

Mensagem da Coordenação Geral da ERCEMAPI 2023

As Escolas Regionais de Computação da SBC desempenham um papel fundamental na disseminação e atualização do conhecimento em computação, especialmente adaptando-se às demandas e características locais. A ERCEMAPI, que abrange Ceará, Maranhão e Piauí, destaca-se como um evento de extrema relevância para a promoção da computação nesses estados. Ao reunir estudantes, profissionais e pesquisadores, a ERCEMAPI facilita o intercâmbio de ideias e inovações, além de proporcionar acesso a conteúdos atuais e incentivar a colaboração regional. A divulgação da computação através da ERCEMAPI auxilia na capacitação da comunidade local, fomenta o desenvolvimento tecnológico e integra esses estados às tendências nacionais e globais da área.

Ao realizar uma escola regional de computação um pouco mais distante dos grandes centros, temos a oportunidade de democratizar o acesso a pesquisas de ponta, trazendo assuntos que estão no estado da arte da computação às regiões mais afastadas. Isso enriquece o ecossistema de tecnologia, fortalece redes de contato e parcerias, além de aproximar acadêmicos e profissionais, com foco em pesquisa e inovação em computação.

Quando tivemos a oportunidade de realizar a ERCEMAPI no IFCE Campus Aracati, em formato híbrido (presencial e remoto), não pensamos duas vezes em aceitar. Pois, após o período de pandemia, muitos alunos do curso de Ciência da Computação do nosso campus iriam concluir a graduação sem a oportunidade de participar ou organizar um evento dessa grandeza de forma presencial. A partir de então, formamos uma comissão de alunos que trabalhou na organização local do evento, sendo responsáveis por toda a parte logística, apresentações, palestras, minicursos, divulgação, cobertura do evento, entre outros. Além do mais, muitos alunos tiveram a oportunidade de submeter seus trabalhos e colher excelentes feedbacks dos revisores.

A organização da parte científica do evento foi um outro desafio. Mas com a colaboração de coordenadores e comitês científicos de programa de artigos e minicursos, conseguimos entregar para o evento trabalhos de altíssima qualidade. As palestras foram excepcionais. E ao analisar tudo que foi apresentado no evento, pudemos verificar claramente que as áreas de Análise de Dados e outras que contam com apoio de Inteligência Artificial, estão ganhando espaço.

Por fim, foi um prazer ajudar na organização da ERCEMAPI 2023 e realizá-la no IFCE Campus Aracati. Foi extremamente gratificante ver a participação dos alunos e a

qualidade técnica do evento. Esperamos que o evento cresça a cada ano, que a quantidade de submissões continue aumentando e que a computação possa ser cada vez mais difundida em nossos estados através da ERCEMAPI. Desejamos a todos uma ótima leitura dos artigos publicados nos anais do evento e dos trabalhos publicados neste livro de minicursos.

Coordenação Geral da XI ERCEMAPI

Prof. Dr. Diego Rocha Lima

Prof. Dr. Guilherme Alvaro Rodrigues Maia Esmeraldo

SINOPSE EM PORTUGUÊS

Este livro oferece uma perspectiva abrangente sobre o sensoriamento remoto, a segurança da informação e a análise de dados de mobilidade urbana, explorando aplicações práticas e conceitos fundamentais em cada área. No primeiro capítulo, “Google Earth Engine no Ensino de Introdução ao Sensoriamento Remoto e Geoprocessamento”, o leitor é conduzido por uma jornada pelo mundo do geoprocessamento e do sensoriamento remoto, apresentando a plataforma Google Earth Engine e suas aplicações educacionais. Conceitos introdutórios são abordados em conjunto com práticas, utilizando a plataforma em nuvem, proporcionando uma visão abrangente e acessível sobre o tema. No segundo capítulo, intitulado “Uma Visão sobre Sistemas de Detecção de Intrusão Baseados em Anomalias”, os leitores adentram o universo da segurança da informação, explorando sistemas de detecção de intrusão e seu aprimoramento com técnicas de Aprendizado de Máquina. Os desafios e estratégias na detecção de ameaças cibernéticas são discutidos em detalhes, oferecendo uma compreensão ampla e aprofundada sobre o assunto. Por fim, o terceiro capítulo, “Tratamento e Análise de Dados de Mobilidade Urbana: Uma Metodologia Teórica e Prática”, aborda a análise de dados de mobilidade urbana, destacando os desafios e oportunidades encontrados nesse campo. Uma metodologia teórica e prática é apresentada, fornecendo insights valiosos sobre o tratamento e análise de dados de mobilidade, juntamente com exemplos práticos e comparações entre bibliotecas Python disponíveis. Estes capítulos tratam de temas relevantes para área da computação e contribuirão para melhor compreensão dos detalhes acerca do sensoriamento remoto, a segurança da informação e a análise de dados de mobilidade urbana.

SINOPSE EM INGLÊS

This book offers a comprehensive perspective on remote sensing, information security, and urban mobility data analysis, exploring practical applications and fundamental concepts in each area. In the first chapter, "Google Earth Engine in Teaching Introduction to Remote Sensing and Geoprocessing", the reader is taken on a journey through the world of geoprocessing and remote sensing, presenting the Google Earth Engine platform and its educational applications. Introductory concepts are covered together with practices, using the cloud platform, providing a comprehensive and accessible view of the topic. In the second chapter,

entitled "An Insight into Anomaly-Based Intrusion Detection Systems", readers enter the world of information security, exploring intrusion detection systems and their improvement with Machine Learning techniques. The challenges and strategies in detecting cyber threats are discussed in detail, offering a broad and in-depth understanding of the subject. Finally, the third chapter, "Treatment and Analysis of Urban Mobility Data: A Theoretical and Practical Methodology", addresses the analysis of urban mobility data, highlighting the challenges and opportunities found in this field. A theoretical and practical methodology is presented, providing valuable insights into the processing and analysis of mobility data, along with practical examples and comparisons between available Python libraries. These chapters deal with topics relevant to the area of computing and will contribute to a better understanding of the details about remote sensing, information security and the analysis of urban mobility data.

SUMÁRIO

Capítulo 1

Google Earth Engine no Ensino de Introdução ao Sensoriamento Remoto e Geoprocessamento

Sérgio Souza Costa, Eduilson Lívio Neves da Costa Carneiro, Denilson da Silva Bezerra, Thomas Victor de Sousa Malheiros Rocha, Luis Fernando Cirqueira da Silva Correia

Abstract

This text discusses basic ideas about geoprocessing and remote sensing combined with practices using a cloud-based platform that offers data storage and computational processing. This approach helps avoid the need for non-specialized computer labs. To do this, we first introduce the Google Earth Engine platform and its main parts. Then, the various kinds of geographic data are presented within the platform's framework, allowing for the discussion of data selection and providing a few examples of processing afterward.

Resumo

Este texto aborda conceitos introdutórios sobre geoprocessamento e sensoriamento remoto integrado com práticas usando a plataforma em nuvem que disponibiliza armazenamento de dados e processamento computacional. Evitando assim a necessidade de laboratórios de informática não especializados. Para isso, inicialmente é apresenta a plataforma Google Earth Engine e seus principais componentes. Depois, os tipos de dados geográficos são apresentadas dentro do contexto da plataforma para que em seguida possa se apresentada a seleção e alguns exemplos de processamento.

1.1. Introdução

O tempo e o espaço são relevantes para a compreensão de diversos fenômenos geográficos naturais (ex: rios, formações rochosas, áreas de floresta) ou sociais como segregação, violência e epidemias. De acordo com [Longley 2005], os fenômenos geográficos são aqueles que possuem tempo e espaço bem definidos. O autor exemplifica essa ligação entre atributos, espaço e tempo com uma simples afirmação:

"A temperatura ao meio-dia local na altitude 34 *graus e* 35 *minutos norte, longitude,* 120 *graus* 0 *minutos oeste, era* 19 *graus Celsius".*

O estudo de fenômenos geográficos fazem parte da ementa de disciplinas de diversos cursos, principalmente na área de Geociências. Nos primeiros semestres é comum a oferta de uma ou duas disciplinas de introdução ao geoprocessamento e/ou sensoriamento remoto. Estas disciplinas requerem laboratórios de informática equipados com soluções de software proprietário como ArcGIS[1] ou abertos como QGIS[2] e Spring[3].

Os custos na montagem destes laboratórios podem ser reduzidos com soluções abertas e gratuitas. Contudo, ainda seria necessário a aquisição de computadores com alto poder computacional, espaço de armazenamento e uma boa conectividade para a realização de diversas práticas, principalmente para aquelas que envolvam processamento de imagens de satélite. Que além de poder computacional requerem espaço em disco e tempo para poder baixar diversas imagens. Acredita-se que muitos dos laboratórios de informática nas universidades brasileiras não possuem estas características. Então, este trabalho propõe a utilização de uma solução que possa aproveitar melhor os laboratórios existentes equipados com computadores de baixo poder computacional e sem a necessidade de instalação de software especializado.

A proposta se fundamenta na utilização de uma plataforma computacional baseada na nuvem, mais especificamente, o Google Earth Engine (GEE) [Gorelick et al. 2017]. Esta plataforma concentra um grande volume de algoritmos e de dados provenientes de diferentes fontes como modelo digital de elevação, séries históricas de imagens de diversos programas espaciais. Essa abordagem evita a necessidade de baixar um grande volume de dados, além de migrar todo o processamento para uma arquitetura computacional na nuvem. Deste modo, este minicurso pretende demonstrar como essa plataforma pode ser usada por docentes de diversos cursos e formação em suas disciplinas introdutórias de geoprocessamento e sensoriamento remoto.

1.2. Conhecendo o Google Earth Engine (GEE)

A plataforma Google Earth Engine (GEE) representa uma infraestrutura de processamento e análise de imagens geoespaciais, concebida e mantida pela corporação Google, voltada para a realização de investigações científicas e análises geoespaciais avançadas. Ela combina uma vasta coleção de imagens de satélite e dados geoespaciais com recursos de processamento em nuvem e ferramentas de análise para permitir que os usuários estudem e visualizem informações em escala global [Gorelick et al. 2017]. Dentre as principais características destacam-se: extenso catálogo de imagens de satélite, algoritmos de processamento, processamento em nuvem e visualização interativa.

O GEE disponibiliza um extenso catálogo de imagens de satélite provenientes de diversas fontes, incluindo as missões Landsat, Sentinel e MODIS. Tal abrangência de fontes de dados permite aos pesquisadores e usuários acessarem informações tanto atuais quanto históricas, permitindo, assim, o monitoramento de alterações na superfície terres-

[1] https://www.arcgis.com/
[2] https://qgis.org/
[3] https://www.dpi.inpe.br/spring/

tre ao longo do tempo. O GEE disponibiliza uma variada gama de algoritmos prontos e acessíveis para o processamento de dados matriciais e vetoriais, incluindo ferramentas de classificação de imagens. Tais algoritmos são executados na infraestrutura de servidores da Google, aliviando os usuários das preocupações inerentes à aquisição e à gestão de vastos conjuntos de dados. Além da exigência de capacidade computacional necessária para o processamento dessas informações. A plataforma oferece também recursos de visualização interativa que facultam aos utilizadores a criação de mapas interativos, bem como a personalização de apresentações e painéis com vistas à exploração e ao compartilhamento dos resultados obtidos.

Além destes recursos, a plataforma apresenta uma diversidade de aplicações, abrangendo áreas como o monitoramento ambiental, a gestão de recursos naturais, a agricultura de precisão e a detecção de desastres naturais, dentre outras. Pesquisadores, cientistas, desenvolvedores e entidades diversas têm aproveitado a plataforma para a realização de análises geoespaciais avançadas. A título de exemplo, no contexto brasileiro, destaca-se o projeto MapBiomas [MapBiomas 2021], que ilustra a utilização bem-sucedida do GEE. Estas aplicações e algoritmos estão integradas mediante bibliotecas disponíveis para as linguagens JavaScript e Python. Nesse texto, iremos usar as bibliotecas em JavaScript por estarem disponíveis para a utilização através do portal `https://code.earthengine.google.com/`, como mostra a Figura 1.1.

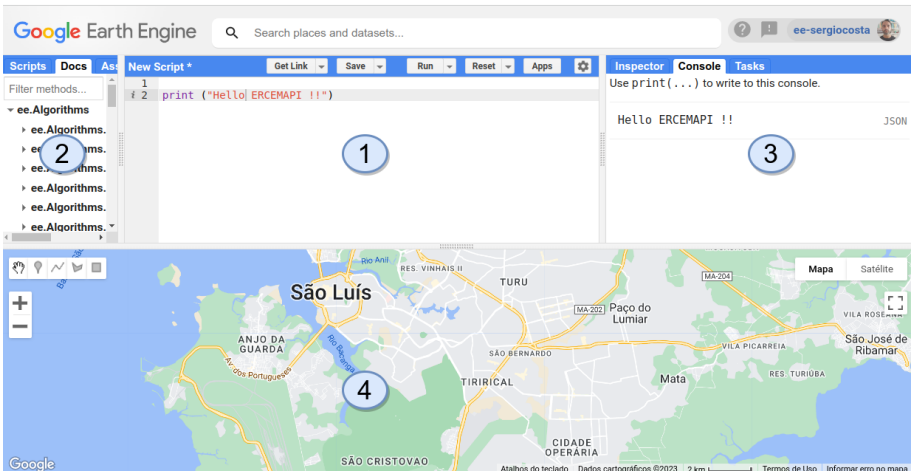


Figura 1.1. Editor de Código do GEE

A Figura 1.1 apresenta o editor destacando 4 principais divisões:

1. **Área de Código**: nesse espaço, os *scripts* são desenvolvidos para carregar, processar e visualizar imagens e dados geoespaciais. O editor comumente oferece realce de sintaxe para facilitar a escrita e a correção de erros no código.

2. **Abas à Esquerda**: nesse espaço estão disponíveis três abas que exibem os Scripts organizados em arquivos e pastas, Docs com a documentação dos principais comandos e funções e os Assets com acesso aos dados vetoriais e matriciais do usuário.

3. **Abas à Direita**: nesse espaço são encontradas abas ou guias que oferecem acesso a funcionalidades adicionais. Incluindo guias para os resultados da execução de um

script, gráficos gerados, registros de saída e progresso de tarefas de importação e exportação.

4. **Área do Mapa**: nesse espaço é exibido um mapa interativos, onde é possível definir parâmetros de visualização, seleção de camada e desenho de dados vetoriais.

O Google Earth Engine foi projetado para poder ser usado por profissionais de diversas áreas sem experiência com programação. Contudo, a melhor utilização desta plataforma pode ser alcançado com a compreensão de alguns conceitos que serão descritos na Seção 1.2.1.

1.2.1. JavaScript no Google Earth Engine

A linguagem JavaScript foi criada por Brendan Eich em 1995 na Netscape com objetivo inicial de ser executada nos navegadores web tornando as páginas mais interativas [Wirfs-Brock and Eich 2020]. Desde então, ela evoluiu e atualmente não está limitada apenas aos navegadores. Antes de ser escolhida para acessar as funcionalidades da plataforma do Google Earth Engine, ela já tinha sido escolhida como a linguagem de script para integrar as aplicações de escritório como Google Docs e Google Spreadsheet [Ferreira 2014]. É importante ressaltar que um *script* consiste em uma sequência de comandos que um computador segue para realizar determinado processo, análogo a uma receita culinária. Portanto, é imprescindível ter conhecimento acerca das instruções ou comandos individuais e compreender a maneira adequada de combiná-los.

Existem algumas instruções que apenas causam um efeito, como a impressão de dados no console (aba à direita da Figura 1.1) ou executa uma operação de zoom na área do mapa, como no código a seguir:

```
1 print ("Hello ERCEMAPI !!")
2 Map.setCenter(-44.2162345711303,-2.548309158409205,12)
```

Enquanto outras instruções geram novos valores a partir de uma busca em uma base de dados ou a partir do processamento de valores criados previamente. Por exemplo, a instrução `ee.Image` busca um objeto do tipo imagem no catálogo a partir de um identificador. Nestes casos, é comum associar esses objetos (ou valores) retornados a um nome, ou seja, armazena-se o resultado das instruções em variáveis. Por exemplo, pode-se associar uma dada imagem ao nome "saoluis" como no exemplo a seguir.

```
1 var saoluis = ee.Image ("LANDSAT/LC08/C01/T1_TOA/LC08_220062_20190504")
```

O uso de variáveis é necessário para poder referenciar esses nomes durante cada um dos três principais passos exemplificados na Figura 1.2. Na **Seleção/Entrada** é criado um objeto do tipo imagem a partir de um identificador e guardado na variável `javascript`. Esse objeto passa por um **Processamento** gerando um novo objeto que será então guardado na variável `javascript`. Essa variável é então adicionado ao mapa para poder ser enviada para a **Apresentação/Saída**.

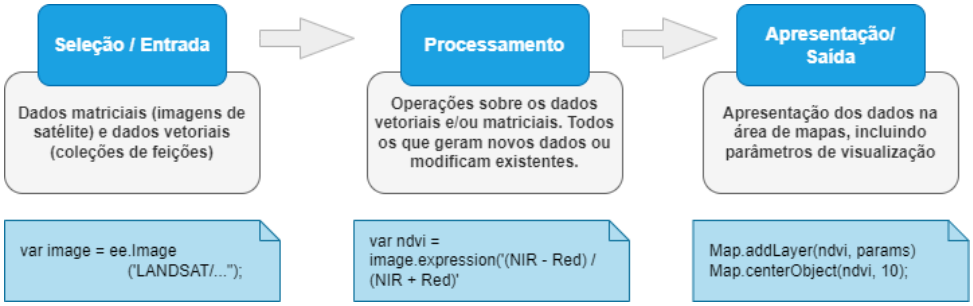


Figura 1.2. Os três principais passos em um *script*

Em algumas situações, pode ser interessante agrupar mais de um valor a mesma variável (ou nome). Como, por exemplo, as informações (metadados) de uma imagem de satélite:

```
var metadados = {
  dataAquisicao: '2019-09-15',  resolucaoEspacial: 30,
  sistemaCoordenadas: 'EPSG:4326'
};
```

Com estas informações agrupadas em um mesmo nome, é possível acessá-las como a seguir:

```
print('Data de Aquisição:', metadados.dataAquisicao);
print('Resolução Espacial:', metadados.resolucaoEspacial);
```

Neste exemplo, a variável `metadados` é um dicionário que armazena informações sobre a imagem, facilitando acessar essas informações quando necessárias. Dicionários são fundamentais e serão usados na definição dos atributos das feições geográficas e nos parâmetros de algumas instruções, como na visualização.

O processamento ilustrado na Figura 1.2 é executado mediante uma única instrução. Contudo, em numerosas situações, o processamento requererá várias instruções, as quais podem ser organizadas e encapsuladas por meio de funções que dado uma entrada, ela será então processada e depois retornada, como ilustrado na Figura 1.3.

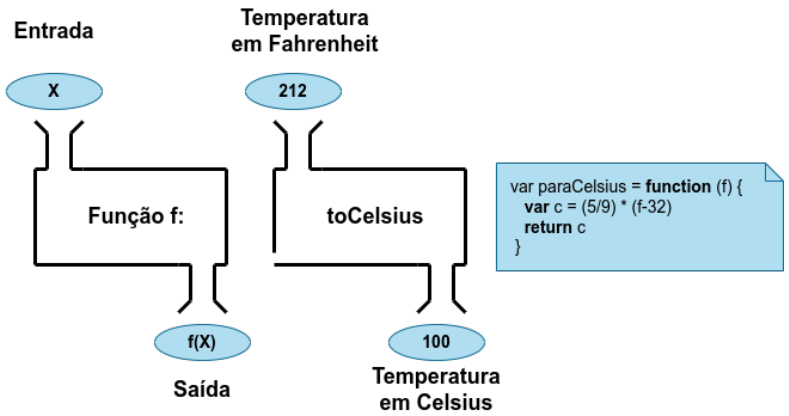


Figura 1.3. Conceito e exemplo de funções

Na Figura 1.3, o lado esquerdo representa o conceito fundamental de uma função, que envolve três elementos essenciais: entrada, processamento e saída. O lado direito da Figura 1.3 ilustra um exemplo concreto de uma função projetada para converter uma temperatura dada em graus Fahrenheit em sua equivalente em graus Celsius, realizando o cálculo e fornecendo o resultado como saída.

1.3. Tipos de dados geográficos

Os tipos de dados geográficos desempenham um papel fundamental na análise espacial e na tomada de decisões em uma variedade de campos, desde a geografia até a ciência ambiental. Como mencionado em [Smith et al. 2007], esses dados podem ser categorizados em dois tipos principais: dados vetoriais e dados matriciais (referido em algumas referências como *raster*). Os dados vetoriais representam objetos geográficos como pontos, linhas e polígonos, enquanto os dados matriciais dividem o espaço em células regulares com valores associados.

1.3.1. Dados vetoriais

Nos dados vetoriais, a localização e a forma de cada objeto são representados por um ou mais pares de coordenadas espaciais (geometrias), podendo incluir atributos não-espaciais que os descrevem e identificam univocamente [DAVIS and Fonseca 2001]. Esses pares de coordenadas são usadas para representar geometrias simples como ponto, linha e polígono (área), Figura 1.4.

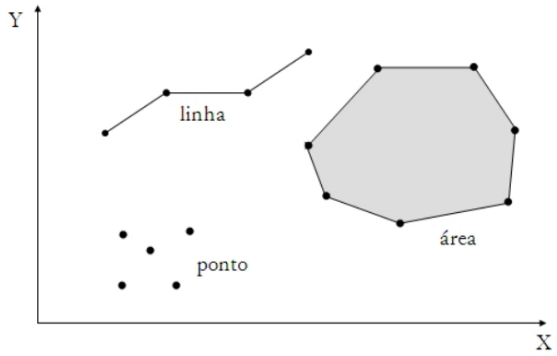


Figura 1.4. Tipos geométricos primitivos. Fonte: [Câmara 2005]

Estas três primitivas podem ser representadas usando diferentes formatos, mas nesse texto será usado formato WKT (*Well-Known Text*) por ser um formato mais legível ajudando a melhor compreender sua estrutura interna. Esse formato inclue informações sobre o tipo do objeto e as coordenadas (longitude e latitude) que o formam, ou seja, é o nome da representação seguido de pares de coordenada. Por exemplo, as três primitivas seriam representadas como:

- POINT(-44.29777178028425 -2.5313470294034044)
- LINESTRING(-44.300075363904284 -2.543551794770009, ... ,-44.338527512341784 -2.564387934650711)

- POLYGON ((-44.29622682789167 -2.5235440348039972, ... ,-44.29622682789167 -2.5235440348039972))

O GEE suporta essas primitivas de modo similar, através dos tipos de dados `ee.Geometry.Point`, `ee.Geometry.LineString` e `ee.Geometry.Polygon` com os pares de coordenadas separadas por vírgulas e entre colchetes. Essas geometrias podem ser criadas especificando os pares de coordenadas, através do editor de geometrias na área de mapa ou importadas a partir de fontes externas. O código a seguir mostra como um ponto poderia ser criado especificando seus pares de coordenadas:

```
1 // WKT - POINT(-44.29777178028425 -2.5313470294034044)
2 var geo = ee.Geometry.Point([-44.29777178028425, -2.5313470294034044]);
```

O código anterior já poderia ser executado na plataforma, contudo para visualizar os dados é necessário adicioná-lo ao mapa e aplicar uma ampliação até ele[1]:

```
1 // passo 1: criação do objeto ponto
2 var geo = ee.Geometry.Point([-44.29777178028425, -2.5313470294034044]);
3 // passo 2: este exemplo não teve nenhum processamento
4 // passo 3: apresentação
5 Map.addLayer(geo , {},'Em algum lugar em São Luís');
6 Map.centerObject(geo, 15);
```

As entidades geográficas codificadas usando o modelo de dados vetoriais são denominadas de feições geográficas (*features*). As feições são objetos vetoriais dos tipos ponto, linha ou polígono [Longley 2005]. Além da forma geométrica, elas são associadas a outros atributos. Por exemplo, uma feição poligonal que representa um município pode ter dados de atributos que incluem o nome do município, população e índice de desenvolvimento humano.

No Google Earth Engine, uma feição é um objeto que consiste em uma geometria e um conjunto de atributos associados. Estes atributos são geralmente representadas como um dicionário[2] de chave-valor, onde cada chave é um nome que descreve aquele atributo e o valor pode ser de qualquer tipo de dado. Por exemplo, o Código 1.1 cria uma geometria com um polígono representando os limites da Universidade Federal do Maranhão e dois atributos. Observe que na definição dos atributos as chaves foram **nome** e **sigla** tendo respectivamente os valores "Universidade Federal do Maranhão" e "UFMA".

```
1 var geo = ee.Geometry.Polygon(
2         [[[-44.305962613096874, -2.5505973845937544],
3           [-44.31527524279658, -2.5613584194753107],
4           [-44.308923771849315, -2.564230871681558],
5           [-44.30261521624629, -2.5549275329820973]]]);
6 var attrs = {nome: "Universidade Federal do Maranhão", sigla:"UFMA"}
7 var feature = ee.Feature(geo, attrs )
8 Map.addLayer(feature)
9 Map.centerObject(feature, 16)
```

Código 1.1. Exemplo de uma simples feição geográfica (*feature*)

[1]Observe que este seria o passo 3 (apresentação/saída) como descrito na Seção 1.2.1

[2]Observe aqui uma aplicação do dicionário citado na Seção 1.2.1

Usuários de sistemas de informação geográfica estão habituados a trabalhar com coleções de feições. Por exemplo, o mapa do estado do Maranhão é uma coleção de feições onde cada município é uma feição. No GEE, esse conceito é reprsentado pelo tipo de dados `ee.FeatureCollection` que é basicamente um conjunto de `ee.Feature`. Essa coleção ser criadas a partir de uma lista de feições (Código 1.2), podem ser carregados diretamente do catálogo plataforma ou importados de outras fontes.

```
1  var s1 = ee.Geometry.Point([-44.28622329382671,-2.5107472461135463]);
2  var s2 = ee.Geometry.Point([-44.25510924089217,-2.527503498708386]);
3  var s3 = ee.Geometry.Point([-44.22563878039214,-2.5338115592006343]);
4
5  var shoppingCollection = ee.FeatureCollection([
6    ee.Feature(s1, { nome: 'Shopping São Luís' }),
7    ee.Feature(s2, { nome: 'Shopping da Ilha' }),
8    ee.Feature(s3, { nome: 'Shopping Rio Anil' })
9  ]);
10
11 Map.addLayer(shoppingCollection , {},'Maiores shoppings');
12 Map.centerObject(shoppingCollection, 10);
```

Código 1.2. Exemplo simples de uma coleção de feições geográficas (*feature*)

O Código 1.2 apresentou um exemplo simples onde foram definidos três feições representando os principais shoppings do município de São Luis do Maranhão. Neste caso, a coleção foi criada explicitamente por uma lista com as 3 feições. Contudo, o mais comum é que estas coleções venham de outras fontes de dados como será visto na Seção 1.4.

1.3.2. Dados matriciais

Em uma representação matricial o espaço é divido em uma malha retangular de células (geralmente quadradas). Nessa representação, o espaço é concebido como uma matriz $P(m,n)$ com m colunas e n linhas, em que cada célula possui um número de linha, um número de coluna e um valor correspondente ao atributo em análise [Câmara 2005]. Um exemplo comum de dados matriciais são as imagens geradas a partir dos sensores a bordo de satélites ou aerotransportado. A resolução espacial destes dados depende da relação entre o tamanho da célula e a área que ela representa no terreno, Figura 1.5.

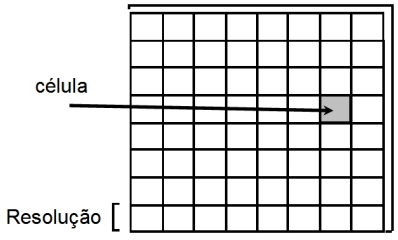


Figura 1.5. Dados matriciais

Geralmente, a resolução espacial é expressa em unidades de comprimento, como metros ou centímetros. Quanto menor for esse valor, maior será a resolução espacial, o que significa que a imagem será capaz de capturar detalhes menores. Por exemplo, na Figura 1.5 apresenta uma dada cena em diferentes resoluções espaciais. Além da resolução

espacial, é importante conhecer outros tipos de resolução, como a temporal, espectral e radiométrica. A resolução temporal se refere ao intervalo que uma dada imagem e capturada, por exemplo, tem satélites que passam regularmente sobre um dado ponto a cada 15 dias[1]. Esse intervalo define a resolução temporal. A resolução espectral é a capacidade de um sensor remoto de dividir o espectro eletromagnético em várias bandas espectrais ou canais de detecção. Cada banda representa uma faixa específica de comprimentos de onda da luz visível, infravermelha, entre outras. Essa divisão do espectro permite que o sensor detecte e registre informações em diferentes partes do espectro eletromagnético.

Uma imagem de sensoriamento remoto pode ser composta por várias bandas espectrais, que são sensíveis as diferentes faixas do espectro eletromagnéticos como vermelho, verde e infravermelho próximo. Cada banda captura informações em uma faixa específica do espectro. A Figura 1.6 representa a faixa do espectro eletromagnético e as bandas do Landsat 9 em cada faixa.

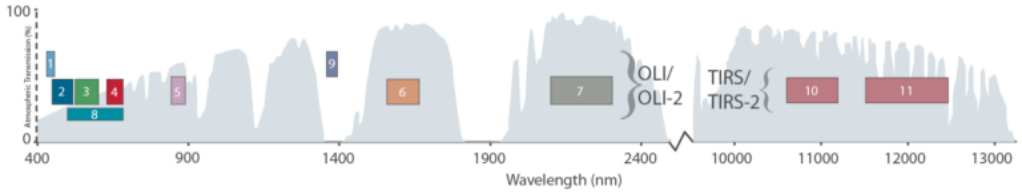


Figura 1.6. O espectro eletromagnético.

A resolução radiométrica refere-se à capacidade de um sensor em registrar variações na intensidade da radiação eletromagnética (geralmente na forma de energia refletida ou emitida) em suas várias bandas espectrais. Em termos mais simples, é a capacidade do sensor de distinguir entre tons de cinza em uma imagem. Por exemplo, em uma imagem com resolução radiométrica de 8 bits, existem 256 tons de cinza diferentes que podem ser representados, variando de preto (valor mínimo) ao branco (valor máximo).

O Google Earth Engine oferece uma variedade de recursos para trabalhar com dados de sensoriamento remoto através de dois tipos de dados: `ee.Image` e `ImageCollection`. O tipo de dados `ee.Image` representa uma única imagem de satélite que pode ser composta por várias bandas espectrais, cada uma representando informações em uma faixa específica do espectro eletromagnético (por exemplo, vermelho, verde, infravermelho próximo, etc.), como ilustrado na Figura 1.6. Cada *pixel* em uma imagem possui valores de intensidade para cada banda espectral, e os dados geoespaciais são associados a coordenadas de localização e sistema de projeção. Similar a `ee.FeatureCollection`, pode-se carregar uma imagem a partir do seu identificador como no Código 1.3.

```
1 var saoluis = ee.Image (
2     "LANDSAT/LC08/C01/T1_TOA/LC08_220062_20190504" )
3 // os parametros de visualização serão explicados posteriormente
4 var vizparams = {bands: ["B4", "B3", "B2"], min: 0, max: 0.5}
5 Map.addLayer(saoluis, vizparams)
6 Map.centerObject(saoluis, 8)
```

Código 1.3. Carregando uma imagem a partir do seu identificador

[1]Existem também os satélites geoestacionários que estão sempre sobre um ponto fixo

No Código 1.3 a variável `saoluis` representa uma única imagem Landsat 8 capturada em 04 de maio de 2019, na região com valor 200 para o *path* e 062 para *row*[1]. Essa imagem contém várias bandas espectrais, como vermelho, verde, infravermelho próximo, entre outras, que podem ser usadas para análises espectrais. Com base nas diferentes respostas espectrais, é possível discernir os diferentes alvos sobre a superfície terrestre, estimar temperaturas entre outras aplicações.

Uma `ee.ImageCollection` é uma estrutura de dados que contém várias imagens organizadas por data de aquisição, localização ou outros critérios. Ela permite representar conjuntos de dados geoespaciais em diferentes momentos, sendo crucial para análises temporais no Google Earth Engine.

```
1 var saoLuis = ee.Geometry.Point(-122.081, 37.419)
2
3 var colecao = ee.ImageCollection('LANDSAT/LC09/C02/T1')
4 .filterDate('2022-01-01','2022-10-30')
5 .filterBounds(saoLuis)
6
7 print("Quantidade de Imagens: ", colecao.size().getInfo())
```

Código 1.4. Exemplo simples de uma feição geográfica (*feature*)r

Com `ee.ImageCollection` é possível fazer análises temporais, como detecção de mudanças e identificação de tendências sazonais. Reduções de ruídos, alcançado através da obtenção de uma média de múltiplas imagens, o que resulta em resultados mais suaves e confiáveis, minimizando assim os efeitos indesejados de variações aleatórias nos dados. Além de operações em lote, dado que a plataforma permite a aplicação de operações em toda a coleção de imagens de maneira conjunta. Isso simplifica consideravelmente a condução das análises, resultando em maior eficiência e produtividade.

1.4. Seleção e visualização de dados geográficos

Nas seções anteriores foram apresentados alguns conceitos básicos da plataforma e sobres os tipos de dados geográficos. Nesta seção será visto melhor como selecionar os dados de interesse.

1.4.1. Seleção e visualização de dados vetoriais

Na Seção 1.3.1 foram apresentados os tipos de dados vetoriais usados na representação computacional dos objetos discretos dentro dos sistemas de informação e banco de dados geográficos. Nessa Seção será apresentado como carregar estes dados diretamente da plataforma do Google Earth Engine ou de outras fontes.

1.4.1.1. Acessando os dados vetoriais

O catálogo do Google Earth Engine é predominantemente composto por dados matriciais. No entanto, também inclui algumas coleções de dados vetoriais, como informações sobre limites de países e outros conjuntos de dados fornecidos principalmente por agências dos

[1] As coordenadas *path* e *row* permitem especificar uma imagem Landsat única em um determinado local na Terra.

Estados Unidos. Para encontrar estes dados, pode-se buscar o seu identificador (ID) no catálogo como na Figura 1.7.

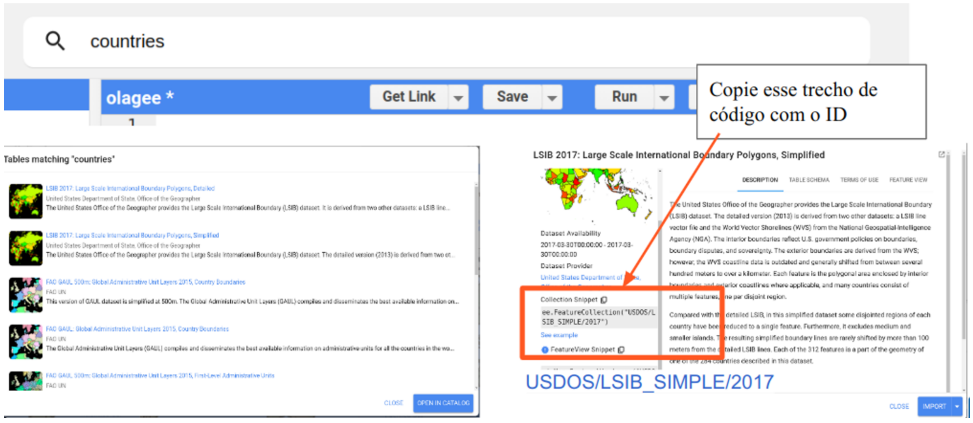


Figura 1.7. Buscando dados vetoriais no catalogo

Com esse identificador, é possível carregar essa coleção de feições como no código a seguir:

```
1 var paises = ee.FeatureCollection("USDOS/LSIB_SIMPLE/2017")
```

Conforme discutido anteriormente, o catálogo atualmente apresenta um número limitado de coleções de dados vetoriais. No entanto, é possível importar informações vetoriais de outras fontes, desde que estejam no formato Shapefile. No portal do Instituto Brasileiro de Geografia e Estatística (IBGE) é possível baixar malhar territoriais de todo país em formato Shapefile que poderão ser transferidas para a plataforma do Google Earth Engine, seguindo os procedimentos detalhados na Figura 1.8.

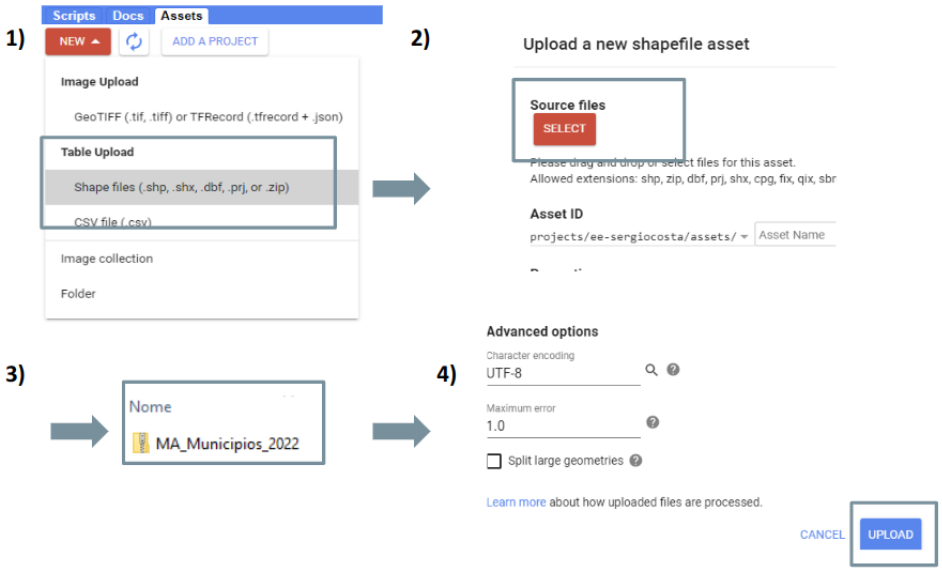


Figura 1.8. Enviando dados vetoriais

Na aba *Assets* inicia-se o processo clicando em *New* e, em seguida, escolher *Shapefiles*. Ao realizar essas etapas, uma janela será exibida para o carregamento dos arquivos. Nessa janela, clique em *Select* para escolher o arquivo que está armazenado em uma pasta local de um computador e em seguida clica-se em *Upload*. Após o arquivo ser carregado, será possível acessá-lo na aba *Assets* e copiar o seu identificador, como mostra a Figura 1.9.

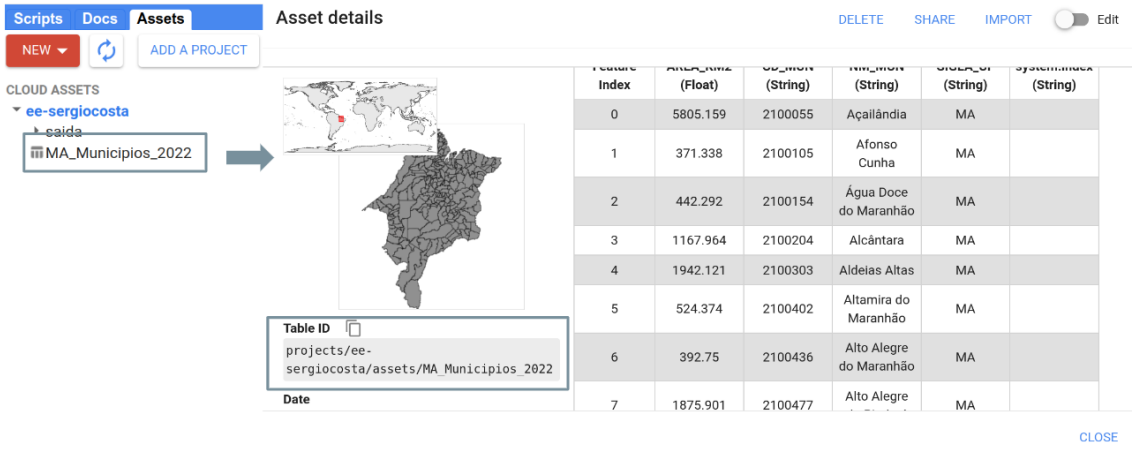


Figura 1.9. Coleções de feições enviadas para o GEEE

Com o uso desse ID, os dados podem ser carregados em um *script* no Google Earth Engine da mesma forma que foi feito com a coleção de países.

```
var municipios = ee.FeatureCollection(
    "projects/ee-sergiocosta/assets/MA_Municipios_2022"
)
```

> O formato Shapefile (também conhecido como ESRI Shapefile) é um padrão amplamente utilizado para armazenar dados geoespaciais vetoriais e tornou-se um formato comum para compartilhar, distribuir e utilizar dados geoespaciais em muitas aplicações.

1.4.1.2. Visualizando os dados vetoriais

A Seção 1.4.1.1 apresentou como acessar os dados vetoriais e associá-los a uma variável. No passo de visualização é comum adicionar os dados ao mapa com a seguinte operação `Map.addLayer(municipios)`. Porém, é possível passar alguns parâmetros adicionais, por exemplo, no código a seguir as geometrias serão apresentadas na cor vermelha e o nome **Municipios** no gerenciador:

```
Map.addLayer(municipios,{color: "FF0000"}, "Municipios")
```

Neste exemplo, a borda e o preenchimento estarão na mesma cor. Para uma definição mais detalhada, usa-se o metodo `.style` onde é possível definir diversos parâmetros,

como a cor, a largura e o estilo da linha, cor do preenchimento entre outros[1]. Por exemplo, para definir a cor da borda em preto e preenchimento em vermelho:

```
Map.addLayer(municipios.style({
  fillColor: 'FF0000', // Cor de preenchimento
  color: '000000',     // Cor da borda
  width: 2            // Largura da borda
}),{}, "Municipios")
```

No exemplo anterior todas as feições são apresentadas na mesma cor, ou seja, usou-se uma simbologia simples. Contudo, é comum a necessidade da elaboração de mapas categóricos, onde cada categoria é atribuída a uma cor, símbolo ou rótulo específico. Essas categorias podem incluir, por exemplo, tipos de uso da terra, zonas de vegetação, classes de solo, divisões administrativas, entre outros. A título de ilustração, considere um conjunto de dados contendo informações sobre as unidades federativas, onde cada unidade está associada a um atributo numérico que representa uma das cinco regiões geográficas do país. Esse código pode ser associado a diferentes cores, e para isso cria-se inicialmente uma imagem vazia:

```
var empty = ee.Image().byte();
```

A seguir, pode ser usar a função `.paint()`[2] que tem como entrada uma coleção de feições (`ufs`) e qual atributo irá ser usado definir o valor dos pixels. Neste caso, valores de 1 a 5.

```
var fills = empty.paint(ufs, 'CD_REGIAO');
```

Após a geração da imagem, é estabelecida uma paleta de cores na qual cada código está vinculado a uma cor específica. Nessa paleta, a cor "408F70" é atribuída ao valor de pixel 1, a cor "FF0000" ao valor de pixel 2 e assim por diante, seguindo a correspondência adequada..

```
var palette = ['408F70', 'FF0000', 'E7EB1B', 'FF8C4D', '9091C8'];
Map.addLayer(fills, {palette: palette, min:1, max:5}, 'colored fills');
```

O efeito pode ser observado na Figura 1.10.

1.4.1.3. Selecionando os dados vetoriais

As operações de seleção de dados vetoriais em SIGs (Sistemas de Informação Geográfica) são cruciais para analisar e extrair informações relevantes de conjuntos de dados geográficos. Essas operações possibilitam aos usuários a capacidade de filtrar, consultar e extrair elementos de dados vetoriais com base em critérios que incluem localização geográfica, atributos não espaciais, relações espaciais ou interações diretas, como, por exemplo, a seleção de objetos por meio de cliques. Da mesma forma, o Google Earth Engine (GEE) oferece a funcionalidade de realizar grande parte dessas seleções.

[1] Veja mais em https://developers.google.com/earth-engine/apidocs/ee-featurecollection-style

[2] A função `.paint` realiza uma operação conhecida como rasterização, que refere o processo de converter dados vetoriais em pixels ou pontos individuais em uma matriz (ou "raster").

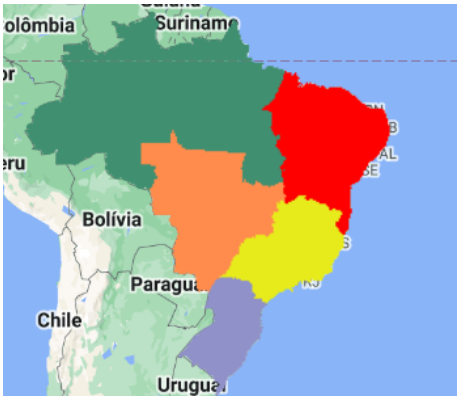


Figura 1.10. Exemplo de impressao de um mapa com cores diferentes para cada região

A **seleção por atributos** no Google Earth Engine (GEE) para dados vetoriais envolve a criação de uma expressão que filtra os elementos de uma coleção com base em atributos específicos. O GEE usa a linguagem de expressão do Earth Engine (EE) para realizar essa seleção. Considere que existe uma coleção de feições armazenadas na variável `municipios` e que agora é necessário selecionar apenas os municípios com mais de 8000 km^2. Assumindo que a área de todas as feições estão armazenadas no atributo `AREA_KM2` é possível fazer a seguinte seleção com base em atributos:

```
1 var filtroArea = ee.Filter.gt ('AREA_KM2', 8000)
2 var filtrados = municipios.filter(filtroArea)
```

A **seleção espacial** envolve o processo de filtragem de elementos com base em sua localização geográfica, como a escolha de elementos contidos em uma área de interesse delimitada por um polígono, aqueles próximos a um ponto de referência específico ou ao longo de uma linha designada. A título de exemplo, considere uma coleção contendo informações sobre todos os municípios do estado do Maranhão e outra coleção representando a delimitação da Amazônia Legal. Com esse contexto, é possível realizar a seleção apenas dos municípios que possuam pelo menos alguma porção de sua extensão geográfica inserida na área demarcada como parte da Amazônia Legal, mediante a aplicação do seguinte filtro.

```
1 var dentroAmazonia = ee.Filter.bounds(amazonialegal)
2 var selecao = municipios.filter(dentroAmazonia)
```

É possível combinar filtros, o que significa que, caso seja necessário, é possível aplicar a negação de um filtro, obtendo assim o resultado inverso.

```
1 var foraDaAmazonia = ee.Filter.bounds(amazonialegal).not()
2 var selecao = municipios.filter(foraDaAmazonia)
```

É possível combinar um filtro espacial e um filtro de atributos, como, por exemplo, para selecionar todos os municípios que não pertencem à área da Amazônia Legal e que apresentam uma área superior a 3.000 km^2:

```
1 var foraDaAmazonia = ee.Filter.bounds(amazonialegal).not()
2 var filtroArea = ee.Filter.gt ('AREA_KM2', 3000)
3 var selecao = municipios.filter(foraDaAmazonia).filter(filtroArea)
```

1.4.2. Seleção e visualização de dados matriciais

A principal fonte de dados matriciais é o catálogo do Google Earth Engine, o qual abriga coleções de imagens provenientes dos principais programas internacionais de sensoriamento remoto. A coleção Landsat se destaca por ser reconhecida como uma das fontes mais antigas e confiáveis de imagens de satélite da Terra, tornando-a uma escolha ideal para a condução de monitoramento de mudanças no uso da terra, identificação de áreas de desmatamento, análise agrícola e diversas outras aplicações. Os satélites Sentinel, que integram o programa Copernicus da União Europeia, disponibilizam uma ampla gama de dados, incluindo imagens ópticas e de radar, sendo empregados com êxito em atividades de monitoramento ambiental, detecção de incêndios florestais e acompanhamento de eventos climáticos extremos.

Adicionalmente, merecem destaque os dados provenientes do sensor MODIS (Moderate Resolution Imaging Spectroradiometer) e da missão Shuttle Radar Topography Mission (SRTM), entre outros. Além disso, é importante ressaltar a capacidade do envio de imagens provenientes de outros programas que, atualmente, não fazem parte da coleção, como é o caso do CBERS (China-Brazil Earth Resources Satellite).

1.4.2.1. Acessando os dados matriciais

No caso de imagens de satélite, é mais comum trabalhar sobre coleções de imagens ao invés de imagens individuais. Porém, similar aos dados vetoriais, é possível acessar uma dada imagem com base em seu identificador, como no código a seguir:

```
1 var image = ee.Image('LANDSAT/LC08/C01/T1_TOA/LC08_220062_20190504');
```

Sabendo como acessar uma dada imagem, a próxima seção apresenta alguns conceitos sobre visualização de dados matriciais no GEE.

1.4.2.2. Visualização de dados matriciais

A aplicação precisa dos parâmetros de visualização desempenha um papel crucial na interpretação de imagens de satélite. A título de exemplo, uma imagem de Índice de Vegetação por Diferença Normalizada gerada a partir do código a seguir possuirá valores variando de -1 a 1.

```
1 var image = ee.Image('LANDSAT/LC08/C01/T1_TOA/LC08_220062_20190504');
2 var ndvi = image.normalizedDifference(['B5', 'B4']);
```

De forma padrão, a representação de uma imagem ocorrerá em escala de cinza; contudo, é possível alterar essa configuração ao estabelecer uma paleta de cores, conforme ilustrado a seguir:

```
1 var npal = ['red', 'yellow', 'green'];
```

Utilizando essa paleta, a correspondência seria estabelecida da seguinte maneira: o valor -1 seria mapeado para o vermelho, enquanto o valor 1 seria relacionado à cor

verde. Os valores intermediários seriam alocados de forma linear nessa escala, transicionando gradualmente do vermelho para o verde e, consequentemente, pelo amarelo. Além da paleta de cores, os parâmetros para os valores mínimo e máximo têm um impacto significativo na visualização conforme demonstrado na Figura 1.11. Na imagem (a), a visualização é realizada sem a definição de uma paleta. Na imagem (b), é apresentada a visualização com a paleta definida anteriormente, porém com valores mínimos e máximos estabelecidos entre -1 e 1. Já na imagem (c), a visualização apresenta os valores mínimo e máximo ajustados para -0.5 e 0.5.

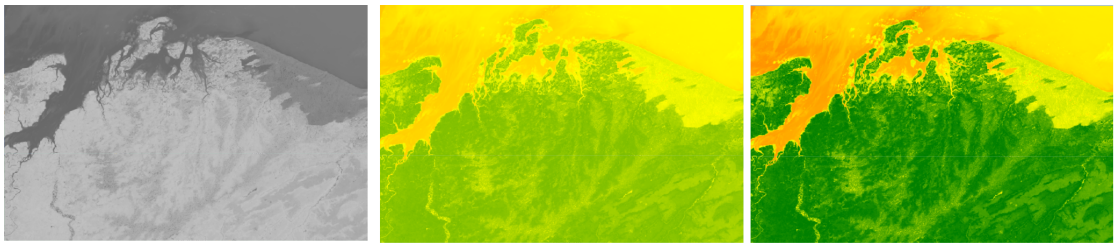


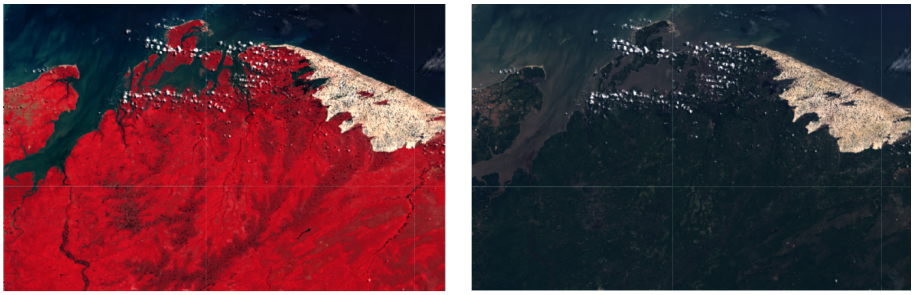
Figura 1.11. Mesma cena com diferentes parâmetros de visualização

Na interpretação de imagens, é uma prática comum a criação de composições coloridas com o propósito de destacar informações específicas. Essa técnica envolve a fusão de diferentes bandas espectrais nos canais de cores vermelho, verde e azul para gerar uma representação colorida da imagem. Nas composições em cores verdadeiras, as bandas são atribuídas aos canais correspondentes, ou seja, a banda do vermelho é vinculada ao canal vermelho, a banda do verde ao canal verde e a banda do azul ao canal azul. Nas composições em cores falsas, outras combinações de bandas são utilizadas, por exemplo, a composição B5 (infravermelho próximo), B4 (vermelho) e B3 (verde) pode ser usada para realçar a vegetação.

```
1 var cor_verdadeira = ['B4', 'B3', 'B2']
2 var falsa_cor = ['B5', 'B4', 'B3']
3 Map.addLayer(image,{bands: falsa_cor, min: 0.03, max: 0.4}, "falsa cor"
    )
4 Map.addLayer(image,{bands: cor_verdadeira, min: 0.03, max: 0.4}, "cor
    verdadeira")
```

Antes de realizar as composições coloridas é importante saber qual faixa de espectro está cada banda da imagem e qual a assinatura espectral do alvo a ser realçado. Cada alvo possui uma assinatura espectral única devido às suas propriedades físicas, químicas e estruturais, o que permite aos cientistas e analistas de sensoriamento remoto distinguir e mapear esses alvos com base nas características espectrais de suas respostas à radiação eletromagnética.

O resultado destas diferentes composições pode ser observado na Figura 1.12 que apresenta uma cena destacando os lençóis maranhenses.



a) Falsa cor b) Cor verdadeira

Figura 1.12. Exemplo de diferentes composições para uma mesma cena

1.4.2.3. Seleção de dados matriciais

Começando com um exemplo de **seleção espacial** onde deseja apenas as imagens de satélite que cobrem um dado município. Considerando que os limites do município de São Luis estão associados a variável `saoluis`, pode-se fazer a seleção como a seguir:

```
1 var imagens = ee.ImageCollection('LANDSAT/LC09/C02/T1')
2 .filterBounds(saoluis)
```

Em aplicações de sensoriamento remoto é comum a necessidade de uma **seleção temporal**, por exemplo, o código a seguir seleciona apenas as imagens de 1º de janeiro de 2022 a 30 de outubro de 2022:

```
1 var imagens = ee.ImageCollection('LANDSAT/LC09/C02/T1')
2 .filterBounds(saoluis)
3 .filterDate('2022-01-01','2022-10-30')
```

Em vez de se basear na seleção por atributos, no contexto das imagens de satélite, é uma prática comum realizar a **seleção por metadados** [1]. Esses metadados contêm informações cruciais para cada imagem, como o percentual de cobertura de nuvem. Tomando o exemplo anterior, é possível aprimorar a consulta selecionando imagens com base na cobertura de nuvens. O código a seguir demonstra como selecionar apenas as imagens com menos de 20% de cobertura de nuvens. Além disso, o resultado dessa seleção será apresentado em ordem ascendente de acordo com o nível de cobertura de nuvens.

```
1 var imagens = ee.ImageCollection('LANDSAT/LC09/C02/T1')
2 .filterBounds(saoluis)
3 .filterDate('2022-01-01','2022-10-30')
4 .filterMetadata('CLOUD_COVER', 'less_than', 20)
5 .sort('CLOUD_COVER');
```

Outra informação relevante é o par de valores “path” e “row” que é usado para indexar e localizar imagens específicas capturadas pelo sistema Landsat. O primeiro descreve a trajetória longitudinal e a segundo a latitudinal. Nesse caso, não será necessário usar o `.filterBounds` dado que esses pares ja definem a localização, como no código a seguir:

[1] Metadados são informações que descrevem outros dados, fornecendo contexto, detalhes sobre a origem, estrutura, significado e outras características dos dados principais

```
var imagens = ee.ImageCollection('LANDSAT/LC09/C02/T1')
.filterDate('2022-01-01','2022-10-30')
.filterMetadata('CLOUD_COVER', 'less_than', 20)
.filterMetadata('WRS_PATH', 'equals', 220)
.filterMetadata('WRS_ROW', 'equals', 62)
.sort('CLOUD_COVER');
```

Estes critérios são muito comuns, inclusive encontradas nos formulários de busca em catálogos de imagens de satélite. A utilização de *scripts* representa uma automação do processo que, de outra forma, exigiria que um usuário inserisse manualmente os valores em cada campo de entrada e realizasse a filtragem por meio de cliques. Além da facilidade de replicação, essa abordagem permite uma ampla variedade de combinações, que possibilitam a criação de filtros mais complexos, ficando a cargo do leitor explorar e experimentar essas possibilidades.

1.5. Processamento

Antes de avançar, é essencial adquirir uma compreensão do padrão conhecido como *filter-map-reduce*, amplamente empregado em diversas aplicações, especialmente nas que lidam com vastos conjuntos de dados. No âmbito das aplicações do Google Earth Engine, essa abordagem é uma prática comum para o processamento eficiente e escalável de dados geoespaciais, conforme exemplificado na Figura 1.13.

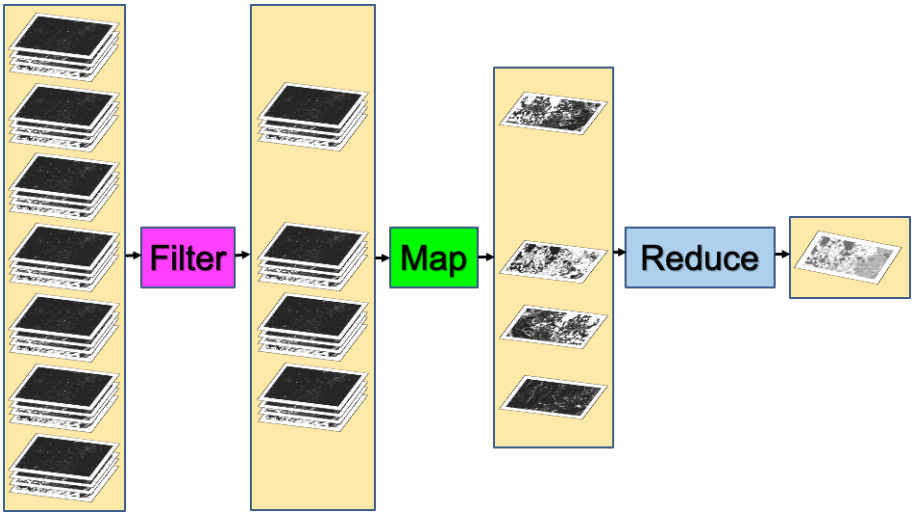


Figura 1.13. O padrão filter-map-reduce no GEE. Fonte [Cardille 2022]

A primeira etapa do processo consiste na seleção ou filtragem dos dados, como explicado na Seção 1.4. Isso é feito por meio da função `.filter()`, que permite escolher um conjunto de imagens ou feições com base em critérios espaciais, temporais, de atributos e/ou metadados. Após a conclusão da etapa de filtragem, a transição para a fase de mapeamento (*map*) ocorre por meio da utilização da função `.map()`. Nessa etapa, uma função específica é aplicada a cada elemento do conjunto de dados filtrados, independentemente de se tratar de imagens ou feições. Essa abordagem permite a realização de cálculos ou transformações nos dados. O resultado do processo de mapeamento é, por sua vez, uma coleção que pode ser combinada utilizando a função `.reduce()`.

No contexto do Google Earth Engine, essa função é frequentemente empregada para calcular estatísticas resumidas, tais como médias, somas, desvios padrão, e assim por diante, com base nos valores mapeados em cada imagem da coleção. Isso se revela valioso para a geração de mapas de resumo ou para a obtenção de estatísticas a partir de uma série temporal de imagens, como será discutido na próxima seção.

Em resumo, o padrão filter-map-reduce no contexto do Google Earth Engine permite que você selecione dados relevantes, aplique operações específicas a esses dados e, em seguida, reduza os resultados para obter informações agregadas.

1.5.1. Processando dados vetoriais

As operações sobre dados vetoriais são comumente conhecidas como operações de geoprocessamento, aqui serão apresentados alguns exemplos mais comuns.

1.5.1.1. Geração de faixas de distância

Essa operação permite identificar influência ou zonas de proximidade ao redor de entidades geográficas específicas, como pontos, linhas ou polígonos [Longley 2005]. Operações de buffer podem ser usadas em análise de proximidade, por exemplo, as autoridades municipais podem usar buffers para definir zonas de restrição ou regulamentação em torno de recursos sensíveis, como rios, reservatórios ou áreas de preservação ambiental. Para o planejamento de transporte, essa operação permite identificar áreas que estão em uma determinada distância de estações de metrô, paradas de ônibus ou estações de trem.

Para realizar essa operação, é necessário selecionar a entidade geográfica de interesse, que pode ser um ponto, uma linha ou um polígono, e definir a distância desejada em unidades geográficas, geralmente em metros. Por exemplo, imagine que desejamos investigar a possível relação entre a atividade de desmatamento e a proximidade a polos madeireiros. Suponhamos que tenhamos uma coleção de dados representando a localização dos polos madeireiros, denominada como variável `polosMadeireiros`. Podemos aplicar uma operação de criação de buffer com uma distância de 50 quilômetros utilizando o seguinte código:

```
1 var fbuffer = function (feat) { return feat.buffer(50000) }
2 var bufferPolos = polosMadeireiros.map(fbuffer)
```

1.5.1.2. Sobreposição de polígonos

A sobreposição de polígonos é uma operação fundamental em Sistemas de Informação Geográfica (SIGs) pois permitem analisar e manipular dados geoespaciais de diferentes maneiras. Essa operação é essencial para várias aplicações, como planejamento urbano, gerenciamento de recursos naturais, análise de riscos e muitos outros campos relacionados à geoinformática. As três operações principais de sobreposição de polígonos em SIGs incluem: a) intersecção, b) união e c) diferença.

A operação de intersecção é um procedimento que consiste na identificação das regiões de sobreposição entre duas ou mais geometrias. Essa operação desempenha um papel fundamental na análise de dados geoespaciais, permitindo a determinação das áreas em que diferentes conjuntos de dados compartilham elementos em comum. Por exemplo, no contexto da pesquisa sobre conflitos territoriais, a operação de intersecção é valiosa para identificar áreas onde as jurisdições de várias entidades se sobrepõem, possibilitando uma análise mais precisa e detalhada. Continuando o exemplo anterior, é possível desenvolver um *script* que identifica as áreas de terras indígenas que se encontram dentro das regiões de buffer, o que pode ser essencial para avaliar o impacto de potenciais atividades ou intervenções dentro dessas áreas.

```
var intersecao = bufferPolos.map(function(feat1) {
      return tisPA.map(function(feat2) {
            var geo = feat1.intersection(feat2);
            return ee.Feature(geo);
      });
}).flatten();
```

A operação de união consiste na combinação de polígonos, gerando um novo polígono que abrange a totalidade da área englobada pelos polígonos originais. Essa técnica é aplicada quando é necessário consolidar informações provenientes de diversas fontes ou criar um único polígono contínuo a partir de várias unidades geográficas menores. Em contextos como a gestão de recursos naturais, a união de múltiplas áreas de vegetação pode resultar em um único polígono representando uma extensa área de habitat. Como exemplo prático, considera-se a fusão de vários municípios para delinear uma determinada região de estudo. O Código 1.5 ilustra a criação de uma feição que representa a área de estudo com quatro cidades localizadas no estado do Maranhão.

```
var filtro =  ee.Filter.or(
                ee.Filter.eq('NM_MUN', 'Barra do Corda'),
                ee.Filter.eq('NM_MUN', 'Grajaú'),
                ee.Filter.eq('NM_MUN', 'Jenipapo dos Vieiras'),
                ee.Filter.eq('NM_MUN', 'Fernando Falcão')
)
var areaEstudo = ee.FeatureCollection(
    "projects/ee-sergiocosta/assets/MA_Municipios_2022")
  .filter( filtro )
  .union()

Map.addLayer(areaEstudo)
```

Código 1.5. Criando uma feição a partir da união de uma coleção de feições

A operação de diferença envolve a subtração de um polígono a partir de outro, sendo aplicada para eliminar uma área específica de um polígono maior. Por exemplo, essa técnica pode ser utilizada para remover uma área construída de um polígono que originalmente representa uma zona de conservação ambiental. Em um cenário semelhante ao apresentado no Código 1.5, suponhamos que haja uma feição representando uma área de estudo (`areaEstudo`). No entanto, durante a análise, o usuário pode desejar excluir um município específico (`grajau`). Nesse caso, é possível realizar a remoção desse município com o seguinte código:

```
var calcularDiferenca = function(feature) {
  var diferenca = grajau.map(function(feat2) {
    return feature.difference(feat2);  });
  return diferenca;
};
var resultadoDiferenca = areaEstudo.map(calcularDiferenca).flatten();
```

1.5.1.3. Cálculo de Centroides

Uma operação bastante comum envolve o cálculo do ponto central, também conhecido como centro de massa, de uma geometria, o que é denominado de centroide. Essa operação é frequentemente empregada para posicionar rótulos em mapas temáticos, realizar análises de redes, avaliar densidade e em diversas outras aplicações. Ela é aplicada à geometria de uma feição, portanto, para calcular os centroides de uma coleção, é necessário utilizar a função `.map()` para aplicá-la a toda a coleção. Por exemplo, para criar uma coleção contendo os centroides de uma coleção de municípios.

```
var calcularCentroide = function(feature) {
  var centroid = feature.geometry().centroid();
  return ee.Feature(centroid)
};
var centroides = municipios.map(calcularCentroide);
```

1.5.1.4. Gerando Dados Amostrais

A geração de amostras aleatórias é uma etapa fundamental em diversas aplicações de sensoriamento remoto e análise geoespacial. Essas amostras podem ser utilizadas para avaliar a qualidade de classificações de imagem, treinar algoritmos de aprendizado de máquina ou conduzir análises estatísticas. Utilizando o método `.randomPoints`, especificamos a região de interesse e o número de pontos aleatórios desejados. Os pontos aleatórios gerados são armazenados em uma coleção de entidades (`FeatureCollection`). No exemplo a seguir é criado uma amostra de 100 pontos em uma dada área de estudo:

```
var randomPoints = ee.FeatureCollection.randomPoints(areaEstudo, 100);
```

1.5.2. Operações sobre dados matriciais

Muitas das operações sobre dados matriciais podem ser melhor entendidas dentro do conceito de algebra de mapas [Tomlin et al. 1990]. Tomlin define as seguintes três operações:

- Operações locais: O valor de uma localização no mapa de saída é calculado a partir dos valores da mesma localização em um ou mais mapas de entrada. Elas incluem expressões lógicas, como “classificar como alto risco todas as áreas sem vegetação com inclinação superior a 15%” (Figura 1.14.a).
- Operações focais: O valor de uma localização no mapa de saída é calculado a partir dos valores da vizinhança da mesma localização no mapa de entrada. Elas incluem expressões como “calcular a média local dos valores do mapa” (Figura 1.14.b).

- Operações zonais: O valor de uma localização no mapa de saída é calculado a partir dos valores de uma vizinhança espacial da mesma localização em um mapa de entrada. Essa vizinhança é uma restrição em relação a um segundo mapa de entrada. Elas incluem expressões como "dado um mapa de cidades e um modelo digital de terreno, calcular a altitude média para cada cidade" (Figura 1.14.c).

Estas operações são ilustradas na Figura 1.14.

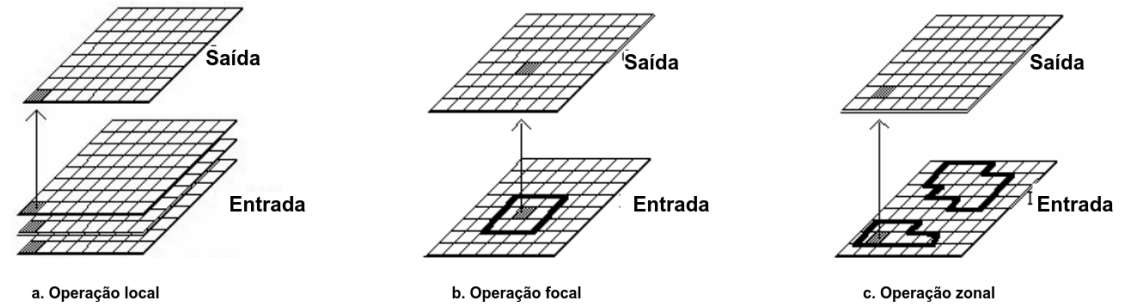


Figura 1.14. Operação da algebra de mapas de Tomlin. Adaptado de [Tomlin et al. 1990]

1.5.2.1. Operadores Locais

Operadores locais realizam operações em células ou pixels individuais em um conjunto de dados matriciais sem considerar os valores nas células vizinhas. Exemplos de operadores locais incluem operações aritméticas básicas (adição, subtração, multiplicação, divisão), bem como funções mais complexas, como logaritmos, exponenciação e operações trigonométricas. No Google Earth Engine, é possível realizar operações aritméticas básicas nas bandas ou imagens como a seguir:

```
1 var resultado = imagem1.add(imagem2);
2 var imagemRaizQuadrada = imagem.sqrt();
```

Um exemplo mais interessante é o cálculo de NDVI (*Normalized Difference Vegetation Index*). Este índice é amplamente utilizado em sensoriamento remoto para avaliar a saúde e a densidade da vegetação em uma determinada área. É um exemplo clássico de uma operação local, onde o valor do índice é calculado pixel a pixel, considerando os valores individuais de cada pixel nas bandas de um conjunto de dados de imagem. A fórmula básica para calcular o NDVI é a seguinte:

$$NDVI = \frac{(NIR - Red)}{(NIR + Red)} \tag{1}$$

Onde NIR (*Near-Infrared*) representa o valor do pixel na banda de infravermelho próximo enquanto Red representa o valor do pixel na banda vermelha. O resultado do cálculo do NDVI é um valor que varia de -1 a 1, onde valores próximos a 1 indicam

vegetação densa e saudável, valores próximos a 0 sugerem áreas com pouca ou nenhuma vegetação e valores próximos a -1 estão associados a corpos d'água ou superfícies não vegetadas. Supondo que a variável `imagem` é uma imagem landsat e as bandas 5 e 4 equivalem ao Red e NIR respectivamente, o seguinte código mostra explicitamente a aplicação da aritmética de bandas para o cálculo do NDVI.

```
var ndvi = image.expression('(NIR - Red) / (NIR + Red)', {
  'NIR': image.select('B5'), // Banda NIR
  'Red': image.select('B4')  // Banda Red
});
```

O mesmo resultado pode ser alcançado utilizando a seguinte função específica, que efetua a diferença normalizada:

```
var ndvi = image.normalizedDifference(['B5', 'B4']);
```

1.5.2.2. Operadores focais

Operadores focais, também conhecidos como operadores de vizinhança, consideram uma vizinhança específica de células ao redor de cada célula de destino ao realizar operações. Esses operadores são usados para tarefas como filtragem, suavização ou realce de características específicas nos dados. Um exemplo comum é o filtro de média, que substitui o valor de cada célula pela média dos valores de suas células vizinhas em uma vizinhança definida. Com base na imagem de NDVI criada anteriormente, é possível a geração de uma nova imagem ao aplicar a média em uma vizinhança de 3 por 3:

```
var ndvisuavizado = ndvi.reduceNeighborhood({
  reducer: ee.Reducer.mean(),
  kernel: ee.Kernel.square(3),
});
```

A Figura 1.15 destacar o efeito da operação focal em parte de uma dada cena.

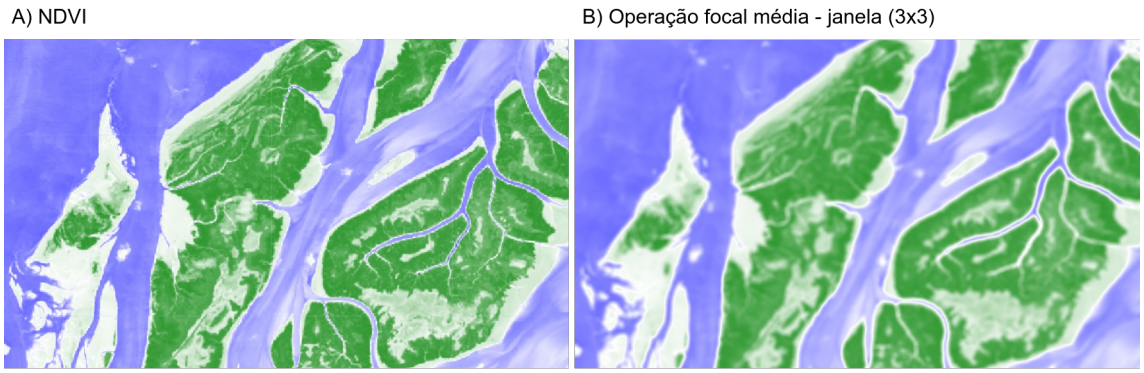


Figura 1.15. Comparando o NDVI antes e depois da aplicação da operação focal

1.5.2.3. Operadores Zonais

Operadores zonais operam em grupos de células dentro de zonas ou regiões definidas. Eles são frequentemente usados para analisar e resumir dados dentro de áreas geográficas ou limites administrativos específicos, como municípios, estados ou áreas de estudo. Operadores zonais podem calcular estatísticas como soma, média, valor máximo ou mínimo das células dentro de cada zona. Com um mapa de NDVI, previamente gerado, é possível calcular a média do NDVI para o município de São Luís:

```
var ndviSaoLuis = ndvi.reduceRegion({
  reducer: ee.Reducer.mean(),
  geometry: saoluis.geometry(),
});
```

Além destes 3 operadores, pode-se considerar ainda um quarto que são os operadores globais. Estes operadores consideram o conjunto de dados matriciais na totalidade ao realizar operações. Eles são usados para tarefas que requerem informações de todo o conjunto de dados, como equalização de histograma, estiramento de contraste ou redimensionamento de todo o conjunto de dados. No Google Earth Engine pode-se usar a função `.reduceRegion` mas sem definir uma região específica.

```
var ndvi_medio = ndvi.reduceRegion({
  reducer: ee.Reducer.mean(),
  bestEffort: true,
});
```

Observe que aqui foi usado um atributo adicional para a operação. Isso ocorreu por que neste caso o tamanho da imagem possui mais pixels do que o valor máximo esperado. Neste caso, pode-se aumentar esse valor máximo ou deixar para o Google Earth Engine decidir qual a melhor estratégia para executar essa operação. Para isso o atributo `bestEffort` foi definido para verdadeiro. Em aplicações mais complexas e com custo operacional alto, o usuário precisará analisar cada caso.

1.6. Conclusões

A abordagem de utilização da plataforma Google Earth Engine (GEE) demonstra ser uma alternativa viável e eficaz para aprimorar a acessibilidade e a eficiência no ensino de geoprocessamento e sensoriamento remoto, particularmente nas disciplinas introdutórias. O GEE proporciona acesso a uma vasta gama de algoritmos e conjuntos de dados geoespaciais, eliminando a necessidade de instalação de softwares especializados e minimizando os requisitos de recursos computacionais nos laboratórios de informática das universidades.

Essa plataforma baseada na nuvem oferece uma solução prática para a análise de fenômenos geográficos com base temporal e espacial bem definidos, abrindo oportunidades para o estudo de diversos aspectos naturais e sociais. Além disso, o GEE integra dados de diferentes fontes, como imagens Landsat, MODIS, Sentinel e dados SRTM, o que enriquece o ambiente de aprendizado e pesquisa.

Ao adotar essa abordagem, os docentes podem focar no ensino de conceitos e métodos de geoprocessamento, em vez de gastar tempo e recursos configurando laboratórios

computacionais complexos. Portanto, a utilização do Google Earth Engine como uma ferramenta educacional nas disciplinas introdutórias de geoprocessamento e sensoriamento remoto se mostra promissora e benéfica para a comunidade acadêmica.

Este texto é complementado por uma página (disponível em: https://lambdageo.github.io/minicurso-gee/) na qual é possível encontrar exemplos de códigos completos, bem como os resultados esperados em cada execução. Além disso, na página estão disponíveis links para os dados extras utilizados em alguns dos exemplos.

Referências

[Câmara 2005] Câmara, G. (2005). Representação computacional de dados geográficos. *CASANOVA, MA et al. Banco de dados geográficos. Curitiba: Mundogeo*, pages 11–52.

[Cardille 2022] Cardille, J. A. (2022). Interpreting image series. https://google-earth-engine.com/Interpreting-Image-Series/Filter-Map-Reduce/.

[DAVIS and Fonseca 2001] DAVIS, C. and Fonseca, F. (2001). Introdução aos sistemas de informação geográficos. *Belo Horizonte: Departamento de Cartografia/UFMG*.

[Ferreira 2014] Ferreira, J. (2014). *Google Apps Script: Web Application Development Essentials*. "O'Reilly Media, Inc.".

[Gorelick et al. 2017] Gorelick, N., Hancher, M., Dixon, M., Ilyushchenko, S., Thau, D., and Moore, R. (2017). Google earth engine: Planetary-scale geospatial analysis for everyone. *Remote sensing of Environment*, 202:18–27.

[Longley 2005] Longley, P. (2005). *Geographic information systems and science*. John Wiley & Sons.

[MapBiomas 2021] MapBiomas (2021). Mapbiomas project-collection 6.0 of the annual series of land use and land cover maps of brazil.

[Smith et al. 2007] Smith, M. J., Goodchild, M. F., and Longley, P. A. (2007). *Geospatial Analysis: A Comprehensive Guide to Principles, Techniques and Software Tools*. Troubador Publishing, 2nd edition.

[Tomlin et al. 1990] Tomlin, C. D. et al. (1990). *Geographic information systems and cartographic modeling*, volume 249. Prentice Hall Englewood Cliffs, NJ.

[Wirfs-Brock and Eich 2020] Wirfs-Brock, A. and Eich, B. (2020). Javascript: the first 20 years. *Proceedings of the ACM on Programming Languages*, 4(HOPL):1–189.

Capítulo 2

Uma Visão sobre Sistemas de Detecção de Intrusão Baseados em Anomalias

Kelson Carvalho Santos, Rodrigo Sanches Miani, Flávio de Oliveira Silva

Abstract

This short course covers Intrusion Detection Systems (IDS) in information security, particularly Anomaly-based IDS, and their enhancement through Machine Learning. In addition, it highlights the types of systems and the utilization of Machine Learning in creating more advanced IDS. In the same way, it discusses the relevance of datasets for improving the adaptability of IDS models to detect unknown attacks. Thus, by elucidating anomaly-based intrusion detection strategies, the short course assists in comprehending these concepts while unveiling the challenges faced in the practical implementation of IDS in an environment of constantly evolving cyber threats.

Resumo

Este minicurso aborda os Sistemas de Detecção de Intrusão (Intrusion Detection Systems - IDS) na segurança da informação, sobretudo os IDS baseados em Anomalias e o seu impulsionamento com o Aprendizado de Máquina. Além disso, são destacados os tipos de sistemas e a aplicação do Aprendizado de Máquina na criação de IDS mais avançados. Da mesma forma, é discutido a relevância dos conjuntos de dados para aprimorar a capacidade de adaptação dos modelos de IDS para detecção de ataques desconhecidos. Assim, ao elucidar estratégias de detecção de intrusão baseados em anomalias, o minicurso auxilia na compreensão desses conceitos, enquanto revela os desafios enfrentados na aplicação prática dos IDS em um ambiente de constante evolução de ameaças cibernéticas.

2.1. Introdução

A crescente dependência da tecnologia digital tem ampliado as fronteiras cibernéticas, tornando as redes e os sistemas potenciais alvos para ataques maliciosos [9]. Assim, a proliferação de ameaças cibernéticas exige abordagens avançadas de proteção. Nesse

cenário, os Sistemas de Detecção de Intrusão *(Intrusion Detection Systems - IDS)* desempenham um papel importante na identificação e prevenção de atividades maliciosas.

Os IDS são mecanismos de segurança cibernética projetados para monitorar, analisar e alertar sobre atividades suspeitas ou maliciosas em redes de computadores ou sistemas de informação. Essas atividades podem incluir tentativas de acessar dados não autorizados, explorações de vulnerabilidades, ataques de *malwares* e outras ameaças à segurança.

No geral, existem dois tipos principais de Sistemas de Detecção de Intrusão [23]: IDS baseados em Rede *(Network-based IDS - NIDS)* e IDS baseados em Host *(Host-based IDS - HIDS)*, conforme ilustrado na Figura 2.1.

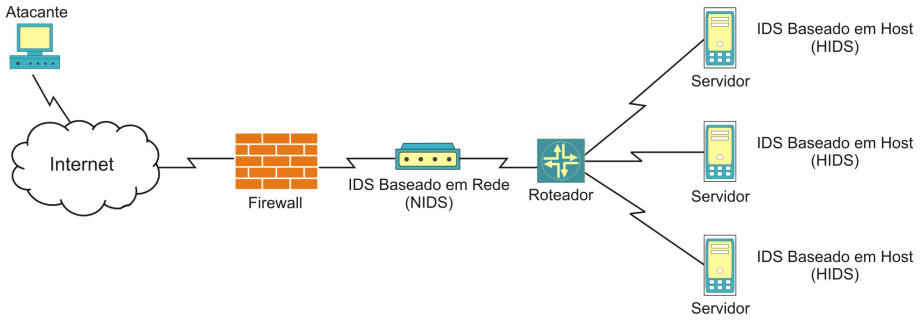


Figura 2.1. Posicionamento dos NIDS e HIDS na estrutura da rede.

Os NIDS são sistemas que monitoram o tráfego de rede em busca de padrões de atividades que possam indicar uma intrusão. Esses sistemas são implantados em pontos estratégicos da rede e examinam os pacotes de dados à medida que trafegam pela rede. Eles não exigem a instalação de *softwares* em *hosts* individuais, tornando-os mais escaláveis.

Por sua vez, os HIDS são sistemas que operam em nível de *host*, monitorando as atividades que ocorrem em um sistema específico, como um servidor ou uma estação de trabalho. Esses sistemas analisam registros de eventos, *logs* do sistema operacional e outras fontes de dados para identificar atividades potencialmente maliciosas. Os HIDS podem detectar ataques direcionados a um único *host*, como tentativas de escalonamento de privilégios, modificações indevidas de arquivos e atividades suspeitas de processos.

Além dessas categorias básicas, os IDS também são classificados com base em suas abordagens de detecção. As duas principais são: Detecção baseada em Assinaturas *(Signature-based Detection)* [11] e Detecção baseada em Anomalias *(Anomaly-based Detection)* [6] .

A Detecção baseada em Assinaturas é uma abordagem que envolve comparar as atividades em tempo real com um banco de dados de padrões conhecidos de ataques e comportamentos maliciosos. Quando uma correspondência *(match)* é encontrada, o IDS emite um alerta. No entanto, essa abordagem é limitada a detectar ameaças previamente identificadas e não é eficaz contra ataques desconhecidos [20].

Por outro lado, a Detecção baseada em Anomalias é uma abordagem que envolve a criação de um perfil de comportamento normal do sistema ou da rede. Assim, o IDS

monitora as atividades em busca de desvios significativos desse comportamento padrão. Isso possibilita a detecção de ataques não identificados anteriormente, mas pode gerar um número maior de falsos positivos, já que em alguns casos, as variações legítimas também podem ser consideradas anômalas.

A Detecção baseada em Anomalias pode ser realizada usando métodos estatísticos, algoritmos de aprendizado de máquina ou uma combinação de ambos [11]. Essa abordagem de detecção é mais difundida com o aprendizado de máquina, que usa algoritmos para identificar padrões complexos que podem indicar atividades maliciosas.

A utilização do aprendizado de máquina para detecção de intrusão pode aumentar a capacidade de identificar ataques anteriormente desconhecidos ou variações de padrões de ataque conhecidos [13].

No aprendizado de máquina existem vários algoritmos de classificação que podem ser aplicados no desenvolvimento de modelos para detecção de intrusão, como *Decision Tree, Random Forest, SVM (Support Vector Machine), kNN (k-Nearest Neighbor), ANN (Artificial Neural Network), XGBoost (eXtreme Gradient-Boosting), LightGBM (Light Gradient-Boosting Machine),* entre outros.

Um modelo de aprendizado de máquina para detecção de intrusão deve ser treinado, validado e testado antes da implementação. Portanto, é nesse contexto que o aprendizado de máquina está sendo bastante utilizado na detecção de intrusão, que baseado em dados, aprende a diferenciar o tráfego de rede entre normal e malicioso [16].

Para o bom desempenho no treinamento dos modelos de aprendizado de máquina para detecção de intrusão, são necessários conjuntos de dados que contenham informações relevantes. Assim é possível estudar os padrões de ataques e as atividades anormais de um sistema de rede.

Diante do exposto, este minicurso discute os Sistemas de Detecção de Intrusão baseados em Anomalias com uso de Aprendizado de Máquina, ou simplesmente, IDS baseados em Aprendizado de Máquina *(ML-based IDS)*. Esses sistemas analisam os dados do tráfego de rede para diferenciar o que é considerado normal e malicioso, de acordo com um modelo de classificação que é treinado, validado e testado.

Considerando esses pontos, o minicurso tem o objetivo de apresentar uma introdução aos IDS baseados em Aprendizado de Máquina, por meio de abordagens teóricas e enriquecidas com demonstrações práticas. Assim, o minicurso auxilia a compreender, projetar e aplicar Sistemas de Detecção de Intrusão baseados em Anomalias.

Além deste material que faz parte do minicurso, também serão disponibilizados os conjuntos de dados e códigos que serão trabalhados nas demonstrações práticas.

A continuidade deste material está organizado da seguinte forma: A Seção 2 apresenta um resumo sobre os métodos do aprendizado de máquina na criação de modelos de classificação para detecção de intrusão; A Seção 3 discute a relevância dos conjuntos de dados na criação de modelos para detecção de intrusão; A Seção 4 descreve uma visão geral sobre as etapas básicas do desenvolvimento de um IDS baseado em Aprendizado de Máquina; A Seção 5 destaca os problemas e desafios encontrados nos IDS baseados em Anomalias; A Seção 6 descreve a metodologia utilizada na aplicação do minicurso,

detalhando as etapas: teórica e prática; A Seção 7 apresenta uma breve discussão sobre a relevância do minicurso na divulgação da pesquisa.

2.2. Aprendizado de Máquina aplicado em IDS

O aprendizado de máquina é uma abordagem essencial para aprimorar os Sistemas de Detecção de Intrusão *(Intrusion Detection Systems - IDS)*, afim de fortalecer a capacidade de identificar ameaças cibernéticas desconhecidas [8].

Ao analisar padrões de comportamento e anomalias em dados de rede, os algoritmos de aprendizado de máquina podem detectar atividades maliciosas que escapam das regras tradicionais de detecção. Isso resulta em uma detecção mais precisa de ataques desconhecidos e uma redução significativa de falsos positivos.

Vários algoritmos de aprendizado de máquina, como *Decision Tree, Random Forest, SVM (Support Vector Machine), kNN (k-Nearest Neighbor), Logical Regression, XGBoost (eXtreme Gradient-Boosting), ANN (Artificial Neural Network)*, entre outros, podem ser empregados na construção de modelos de classificação para detecção de intrusão (consulte a Tabela 2.1).

Os modelos construídos permitem que os IDS se adaptem dinamicamente a novos vetores de ataque, sendo evoluídos para acompanhar as táticas em constante mudança dos invasores [33]. Esses modelos ao serem treinados com conjuntos de dados representativos e diversificados, permitem que os IDS aprendam a reconhecer padrões e comportamentos anômalos, aprimorando a sua eficácia na proteção de redes e sistemas [25].

Os modelos de aprendizado de máquina são identificados de acordo com os métodos de aprendizado, a seguir: Aprendizado Supervisionado, Aprendizado Não Supervisionado e Aprendizado Semi-supervisionado (ver Figura 2.2).

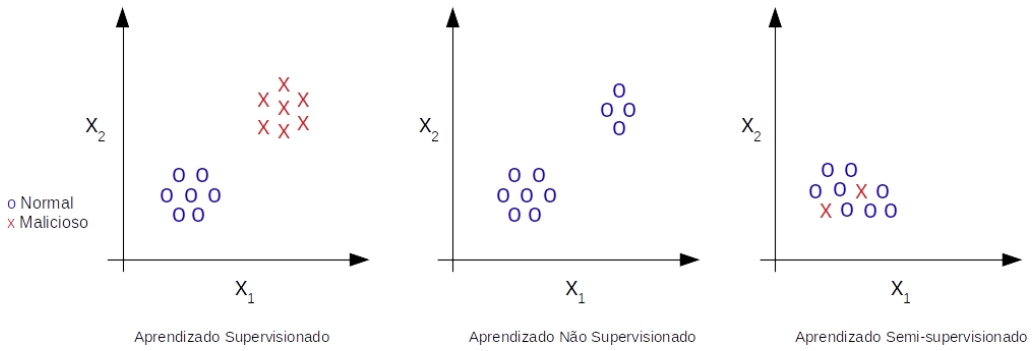


Figura 2.2. Tipos de aprendizado de máquina.

O Aprendizado Supervisionado é utilizado com mais frequência nos modelos de IDS, onde requer um conjunto de dados totalmente rotulado para encontrar a relação entre os dados e suas classes, durante as etapas de treinamento, validação e teste [27]. A rotulagem manual dos dados é cara e demorada, consequentemente, a falta de dados rotulados constitui um problema nesse tipo de aprendizado [14].

Por outro lado, o Aprendizado Não Supervisionado pode ser usado para identificar anomalias ou valores discrepantes no conjunto de dados. Esse tipo de aprendizado é

Tabela 2.1. Algoritmos de aprendizado de máquina.

Algoritmos	Descrições
Decision Tree	Este algoritmo cria uma árvore de decisão que pode ser interpretável. Nela é possível excluir automaticamente recursos irrelevantes e redundantes. O processo de aprendizado inclui a seleção de atributos, a geração de árvores e a poda de árvores. O algoritmo seleciona os atributos mais adequados e gera nós filhos a partir do nó raiz [14].
Random Forest	Este algoritmo consiste em múltiplas árvores de decisão, que funciona criando *n* árvores diferentes com a seleção aleatória de várias amostras do conjunto de dados. Em seguida é realizado a classificação (predição) para cada árvore de decisão, havendo uma votação e selecionado o resultado que obtiver a maioria dos votos [24].
SVM (Support Vector Machine)	A estratégia deste algoritmo é encontrar um hiperplano de separação de margem máxima no espaço de atributos de *n* dimensões. O algoritmo pode alcançar resultados gratificantes mesmo com conjuntos de treinamento em pequena escala, porque o hiperplano de separação é determinado apenas por um pequeno número de vetores de suporte. No entanto, é sensível ao ruído próximo ao hiperplano. As funções de *kernel*, geralmente, são utilizadas para dados não lineares [14].
kNN (k-Nearest Neighbors)	Este algorimto é baseado na hipótese múltipla, onde a maioria dos vizinhos de uma amostra pertencer a mesma classe, a amostra tem uma alta probabilidade de pertencer aquela classe. Assim, o resultado da classificação está relacionado apenas aos *k*-vizinhos mais próximos. O parâmetro *k* influência muito no desempenho do modelo. Quanto menor for *k*, mais complexo é o modelo e maior o risco de sobreajuste do modelo. Por outro lado, quanto maior for *k*, mais simples é o modelo e mais fraca é a capacidade de ajuste [14].
Logical Regression	Este é um tipo de algoritmo linear logarítmico, que calcula as probabilidades de diferentes classes por meio de distribuição logística paramétrica. O algoritmo não consegue lidar bem com dados não lineares, o que limita a sua aplicação [14].
XGBoost (eXtreme Gradient-Boosting)	Este algoritmo foi projetado, principalmente, para ganho de velocidade e desempenho usando árvores de decisão. Tem a vantagem do processamento paralelo, que utiliza todos os núcleos da máquina. É altamente escalável e eficaz para lidar com questões de classificação e pré-processamento de dados em alto nível [4].
ANN (Artificial Neural Network)	São algoritmos bastante utilizados para tarefas de classificação em múltiplos domínios [3]. Em relação a detecção de intrusão, este algoritmos são capazes de capturar relações altamente complexas e não lineares entre variáveis dependentes e independentes, sem o conhecimento prévio de ambas [31].

útil em cenários onde a maioria dos dados pertence a uma classe e a principal tarefa é detectar instâncias que se desviam do padrão normal. [29]. Para isso, é comum a aplicação de técnicas de agrupamentos, onde os algoritmos só reconhecem os dados considerados normais e todo o resto é classificado como malicioso [11].

Já o Aprendizado Semi-supervisionado pode ser empregado quando os rótulos estão disponíveis somente nas instâncias de dados normais [7]. Sendo as anomalias detectadas pelas instâncias de dados dos atributos que se desviam, significativamente, do modelo construído. Esse tipo de aprendizado também pode utilizar uma pequena quantidade de dados rotulados, juntamente, com um conjunto maior de dados não rotulados. Um exemplo, pode ser o algoritmo *One-class Support Vector Machine (OSVM)*, que pode ser aplicado em cenários de detecção de intrusão onde os dados de anomalias rotulados são escassos [2].

Embora os métodos de aprendizado de máquina existam há bastante tempo, ainda não foi estabelecido quais métodos são mais eficientes para a detecção de intrusão [35]. Portanto, é necessária a realização de uma avaliação de desempenho dos dados de referência para comparar os diferentes métodos.

Com isso, a Tabela Verdade, também conhecida como Matriz de Confusão e ilustrada na Figura 2.3, serve de base para avaliação de desempenho dos modelos de aprendizado de máquina. A partir desta tabela, um conjunto de métricas (consulte a Tabela 2.2) podem ser empregadas para avaliar de forma abrangente o desempenho dos modelos.

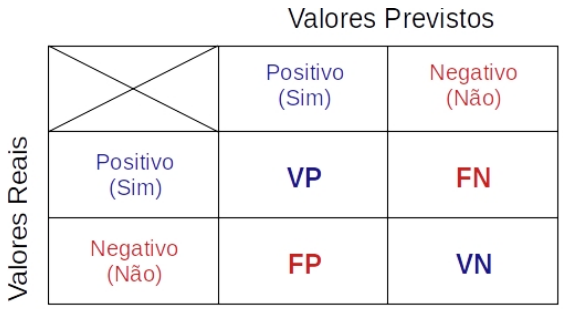


Figura 2.3. Tabela verdade.

A Tabela Verdade (Matriz de Confusão) é composto por:

- VP (Verdadeiro Positivo): significa a proporção de eventos maliciosos classificados corretamente;
- VN (Verdadeiro Negativo): significa a proporção de eventos normais classificados corretamente;
- FP (Falso Positivo): significa a proporção de eventos normais classificados erroneamente como maliciosos;
- FN (Falso Negativo): significa a proporção de eventos maliciosos classificados erroneamente como normais.

Tabela 2.2. Métricas de avaliação de desempenho dos modelos.

Métricas	Descrições	Fórmulas
Precision	É a porcentagem de eventos verdadeiros positivos sobre o número total de positivos identificados.	$\frac{TP}{TP+FP}$
Recall	É a porcentagem de eventos verdadeiros positivos identificados como maliciosos sobre o total de eventos.	$\frac{TP}{TP+FN}$
Accuracy	É a porcentagem de previsões corretas (verdadeiras e falsas).	$\frac{TP}{TP+FN}$
False Alarm Rate	É a porcentagem de eventos maliciosos classificadas incorretamente sobre o número total de eventos normais.	$\frac{FP}{TN+FP}$
Miss Rate	É a porcentagem de eventos maliciosos classificados incorretamente sobre o número total de eventos maliciosos.	$\frac{FN}{FN+TP}$
Error Rate	É a porcentagem de eventos classificadas incorretamente sobre o total de eventos.	$\frac{FP+FN}{TP+TN+FP+FN}$
False Positive Ratio	É a porcentagem de falsos positivos sobre o número total de positivos identificados.	$\frac{FP}{TP+FP}$
False Negative Ratio	É a porcentagem de falsos negativos sobre o número total de negativos identificados.	$\frac{FN}{TN+FN}$
F-Measure	Utiliza a porcentagem do *Precision* e do *Recall* para medir a exatidão do método.	$2 * \frac{Precision*Recall}{Precision+Recall}$
Total Cost Ratio (TCR)	É a porcentagem do custo de eventos classificados incorretamente, onde λ é o custo relativo de ambos os erros.	$\frac{FN+TP}{\lambda(FP+FN)}$
Weighted Error (W Err)	É a porcentagem do erro ponderado que é calculado usando um peso específico λ.	$\frac{\lambda TN+TP}{\lambda FP}$
ROC Curve	É a porcentagem da taxa de verdadeiros positivos, plotada em relação à taxa de falsos positivos.	não existe

Fonte: Adaptado de [17].

Em resumo, o aprendizado de máquina fomenta os Sistemas de Detecção de Intrusões (IDS) a um mecanismo de defesa proativo e inteligente, aumentando a sua capacidade de oferecer uma defesa mais robusta contra ameaças cibernéticas emergentes. Com o aprendizado de máquina os IDS proporcionam uma abordagem flexível e adaptável para detecção de intrusão. Dessa forma, os IDS podem discernir padrões e anomalias no tráfego de rede com maior precisão, minimizando falsos positivos e melhorando a acurácia da detecção de ameaças.

2.3. Conjuntos de Dados para Detecção de Intrusão

Os conjuntos de dados para detecção de intrusão contêm informações que são necessárias para estudar os padrões de ataques e as atividades anormais de um sistema de rede [8]. Essas informações são os dados de coleta de entrada e saıda do tráfego de rede.

Ao fornecer os dados do tráfego de rede, os conjuntos de dados possibilitam o treinamento, validação e teste dos modelos de aprendizado de máquina para detecção de intrusão, com o intuito de identificar atividades suspeitas ou maliciosas.

A variedade de cenários e ataques existentes nos conjuntos de dados proporcionam a criação de IDS robustos, capazes de lidar com ameaças desconhecidas [13]. No entanto, a escolha do conjunto de dados adequado para o treinamento, validação e teste do modelo é considerado crucial, pois deve refletir com precisão as condições mais próximas do mundo real. Assim é possível assegurar resultados confiáveis na detecção de intrusão.

De acordo com [25], os conjuntos de dados podem ser criados com base em eventos reais, emulados ou sintéticos. Os criados com base em eventos reais são registrados sobre uma topologia com configuração de rede completa, contendo por exemplo, modens, *firewalls*, *switches*, roteadores, servidores e *hosts* com diferentes sistemas operacionais [8]. Já os conjuntos de dados de eventos emulados são registrados em um ambiente de teste ou de emulação de rede [25].

Por sua vez, os baseados em eventos sintéticos são registrados sobre um tráfego de rede injetado por um gerador de tráfego. A injeção de ataque sintético, também, pode ser usada para introduzir ataques a um conjunto de dados existente, por exemplo, para equilibrar as classes de ataques [8].

Na criação de um conjunto de dados para detecção de intrusão, normalmente, o tráfego de rede é capturado no formato de pacotes ou no formato de fluxos [25].

Os dados no formato de pacotes abrangem informações completas sobre a carga útil do tráfego de rede e são disponibilizados em arquivos PCAP *(Packet CAPture)*. A captura, geralmente, é realizada com o espelhamento das portas nos dispositivos de rede.

Os dados no formato de fluxos possuem informações mais compactas, contendo apenas os metadados das conexões, como endereço IP de origem e destino, porta de origem e destino, protocolos de transporte e aplicação, tempo de duração do fluxo, etc. Dessa forma, os fluxos podem aparecer nas formas unidirecionais ou bidirecionais, contendo os pacotes que compartilham os metadados dentro da janela de tempo do fluxo de rede (ver a Figura 2.4).

Na forma unidirecional são reunidos os pacotes do *host* de origem ao *host* de destino, que compartilham os metadados do fluxo. Na direção contrária, ou seja, do *host* destino ao *host* de origem, os pacotes são reunidos em outro fluxo unidirecional. Enquanto isso, na forma bidirecional são contidos, nos mesmos fluxos, todos os pacotes entre os *hosts* de origem e destino e entre os *hosts* de destino e origem, independente da direção.

Além disso, no tráfego de rede os pacotes ou fluxos de dados são capturados e

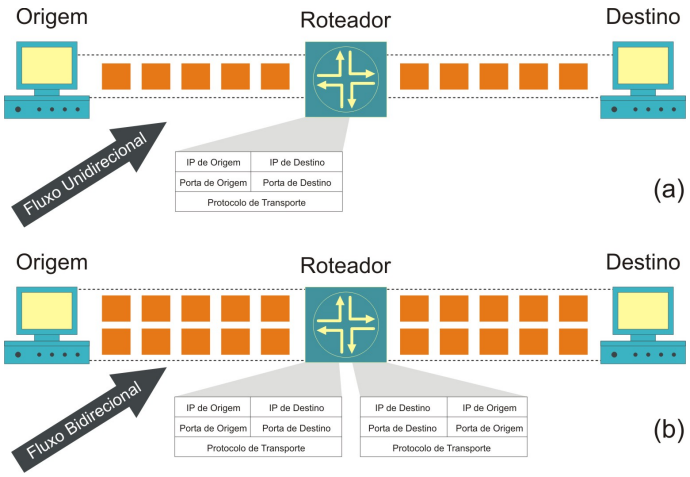


Figura 2.4. Fluxos de rede unidirecional (a) e bidirecional (b)

registrados de forma desordenada e precisam ser pré-processados para a seleção[1] e extração[2] de dados relevantes que possam caracterizar os eventos da rede [18]. Isso permite que os modelos de aprendizado de máquina possam classificar e detectar as anomalias.

Em geral, os conjuntos de dados para detecção de intrusão devem conter os seguintes critérios [34]: (i) tráfego de rede que permita a classificação dos eventos em normais e maliciosos; (ii) tráfego de rede que representa características de cenários reais de redes; (iii) rótulos para classificação dos eventos como normal ou malicioso; (iv) tráfego de rede de cenários distintos; (v) estrutura flexível para fácil atualização ou reprodução, afim de comparação com outros conjuntos de dados; (vi) ausência de dados confidenciais que impossibilite o compartilhamento; (vii) documentação disponível, informando as limitações e os métodos utilizados durante a sua construção.

Os critérios descritos acima são importantes para analisar a qualidade dos conjuntos de dados. Ainda assim, é possível encontrar irregularidades que podem influenciar na qualidade e na confiabilidade dos modelos de classificação para detecção de intrusão, como valores ausentes, ruídos, inconsistências e redundâncias de dados.

As correções para tais irregularidades podem ser realizadas com métodos de pré-processamento de dados [1], que ajudam a mitigar a influência desses problemas nos conjuntos de dados, durante as etapas treinamento, validação e teste.

O pré-processamento de dados é importante na transformação dos dados, tornando os dados mais fáceis de manipular e reduzindo o tempo de execução dos modelos. São empregadas entre as técnicas mais comuns de transformação dos dados: a normalização e a codificação [21].

[1] A seleção de dados é responsável por criar um subconjunto relevante de atributos contidos no conjunto de dados, para uma tarefa específica, com base em determinados critérios ou condições. Isso pode ser feito para melhorar a robustez do modelo, reduzir os custos computacionais ou focar em aspectos específicos dos dados [26].

[2] A extração de dados envolve o processo de recuperação ou extração de informações ou atributos específicos do conjunto de dados, com o objetivo de identificar e isolar atributos ou características relevantes que são cruciais para a tarefa do modelo de classificação [26].

A normalização de dados permite que atributos que contenham instâncias de valores muito baixos, não sejam superados por aqueles que possuem instâncias de valores muito altos. Assim, é realizado o dimensionamento de ambos para um intervalo especificado, como 0.0 e 1.0. A principal vantagem da normalização é a redução de tempo no processamento do modelo, já que as instâncias ficam numa mesma escala numérica reduzida [1]. Algumas técnicas de normalização de dados incluem o *decimal scale*, o *z-score* e o *min-max scale* [22].

Por a sua vez, a codificação é o processo de conversão de dados categóricos em um formato que os modelos podem usar para melhorar a classificação [16]. Dessa forma, se o conjunto de dados possuir dados categóricos, os mesmos são codificados para valores numéricos antes de serem utilizados pelos modelos de classificação.

Para fins de pesquisas existem vários conjuntos de dados para detecção de intrusão disponíveis publicamente para o desenvolvimento de trabalhos que possam contribuir com a evolução dos IDS, como NSL-KDD [32], UNSW-NB15 [19], UGR'16 [15], CIC-IDS2017 [30], CSE-CIC-IDS2018 [30], entre outros.

O NSL-KDD *(National Science Laboratory - Knowledge Discovery in Databases)* foi desenvolvido em 2009, na Universidade de *New Brunswick*, Canadá, como um aprimoramento do conjunto de dados original KDD Cup'99. O NSL-KDD foi criado para abordar os problemas e limitações encontrados KDD Cup'99, visando uma representação mais realista de cenários de intrusão de rede. Entre as novidades foi eliminado a redundância, introduzido novas instâncias de ataques e garantido a distribuição equilibrada dos tipos de ataques. O NSL-KDD compreende quatro tipos principais de ataques: *Denial of Service (DoS), Probe, User to Root (U2R) e Remote to Local (R2L)*. Esses ataques simulam diversas ameaças à segurança, fornecendo um conjunto abrangente e diversificado de cenários para avaliar IDS. O NSL-KDD tornou-se uma referência amplamente utilizada para avaliar a robustez dos modelos de detecção de intrusão.

O UNSW-NB15 foi desenvolvido em 2015, na Universidade de *New South Wales (UNSW)*, Austrália, como um conjunto abrangente de dados de tráfego de rede para pesquisa de detecção de intrusão. O conjunto de dados foi criado capturando o tráfego do mundo real em um ambiente controlado, incluindo atividades normais e maliciosas. O UNSW-NB15 consiste em nove categorias principais de ataque, abrangendo vários tipos de intrusão, como *Fuzzers, Analysis, Backdoors, Denial of Service (DoS, Exploits, Generic, Reconnaissance, Shellcode e Worms*. O seu desenvolvimento teve como consequência fornecer uma representação mais realista dos desafios modernos de segurança de redes, tornando-o um recurso valioso para avaliar e melhorar os IDS.

O UGR'16 foi desenvolvido em 2016, na Universidade de Granada, Espanha, como referência para avaliação de IDS. Foi gerado capturando o tráfego de rede em um ambiente controlado, combinando atividades normais e maliciosas para simular ameaças cibernéticas do mundo real. O UGR'16 inclui uma variedade de ataques, categorizados em diferentes classes, como *Denial of Service (DoS), Distributed Denial of Service (DDoS), Reconnaissance, User to Root (U2R), Remote to Local (R2L)* e vazamentos de dados. O seu desenvolvimento teve como objetivo responder à necessidade de dados diversos e relevantes para avaliar a robustez dos modelos de detecção de intrusão em uma vasta gama de ameaças cibernéticas.

O CIC-IDS2017 foi desenvolvido em 2017, no Instituto Canadense de Segurança Cibernética *(Canadian Institute for Cybersecurity - CIC)*, da Universidade de *New Brunswick*, Canadá. O conjunto de dados foi criado capturando o tráfego de rede em um ambiente controlado para servir como referência para avaliação de IDS. O CIC-IDS2017 foi gerado usando uma variedade de cenários realistas para emular o comportamento normal da rede e diversas ameaças cibernéticas. O conjunto de dados inclui uma grande diversidade de ataques, abrangendo categorias como *Denial of Service (DoS), Distributed Denial of Service (DDoS), Brute Force Attacks, Network Scans* e infecções por *Malwares*. O desenvolvimento do CIC-IDS2017 concentrou-se em fornecer uma amostra abrangente e representativa das ameaças cibernéticas contemporâneas, tornando-o um recurso valioso para pesquisas na área da segurança cibernética.

O CSE-CIC-IDS2018 foi desenvolvido em 2018, no Instituto Canadense de Segurança Cibernética *(Canadian Institute for Cybersecurity - CIC)*, da Universidade de *New Brunswick*, Canadá. Foi criado como parte do projeto Avaliação de Segurança Cibernética *(Cybersecurity Evaluation - CSE)*, com o objetivo de fornecer um conjunto de dados realista e diversificado para avaliar IDS, podendo ser considerado uma atualização do CIC-IDS2017. O CSE-CIC-IDS2018 foi gerado capturando o tráfego de rede em um ambiente controlado, combinando atividades normais e maliciosas. O CSE-CIC-IDS2018 abrange uma ampla gama de ataques, incluindo *Denial of Service (DoS), Distributed Denial of Service (DDoS), Brute Force Attacks*, tentativas de infiltração e várias infecções por *Malwares*. O desenvolvimento do conjunto de dados objetivou refletir as ameaças contemporâneas à cibersegurança, garantindo a sua relevância como referência para avaliar a robustez dos métodos de detecção de intrusão.

2.4. Processo de Desenvolvimento de um IDS baseado em Aprendizado de Máquina

O desenvolvimento de um Sistema de Detecção de Intrusão *(Intrusion Detection System - IDS)* baseado em Aprendizado de Máquina é um processo dinâmico que exige uma compreensão abrangente dos dados, seleção criteriosa de algoritmos e avaliação rigorosa do modelo. A visão geral das etapas básicas é ilustrada na Figura 2.5.

2.4.1. Etapa 1: Seleção do Conjunto de Dados

A seleção do conjunto de dados adequado é fundamental para construção do modelo de detecção de intrusão. Os dados contidos no conjunto irão servir de base para a criação dos padrões desejados para a classificação das instâncias.

O conjunto de dados deve abranger uma gama diversificada de atividades normais e maliciosas para garantir a robustez do modelo na identificação dos padrões. Uma compreensão abrangente dos dados contidos no conjunto, incluindo a documentação com informações sobre o formato (binário ou multiclasse) e as características dos dados, são fundamentais para adaptar os estágios seguintes de desenvolvimento do modelo de detecção de intrusão.

A Seção 2.3 aborda com mais detalhes os conjuntos de dados, inclusive descrevendo alguns exemplos de conjunto de dados públicos encontrados na literatura.

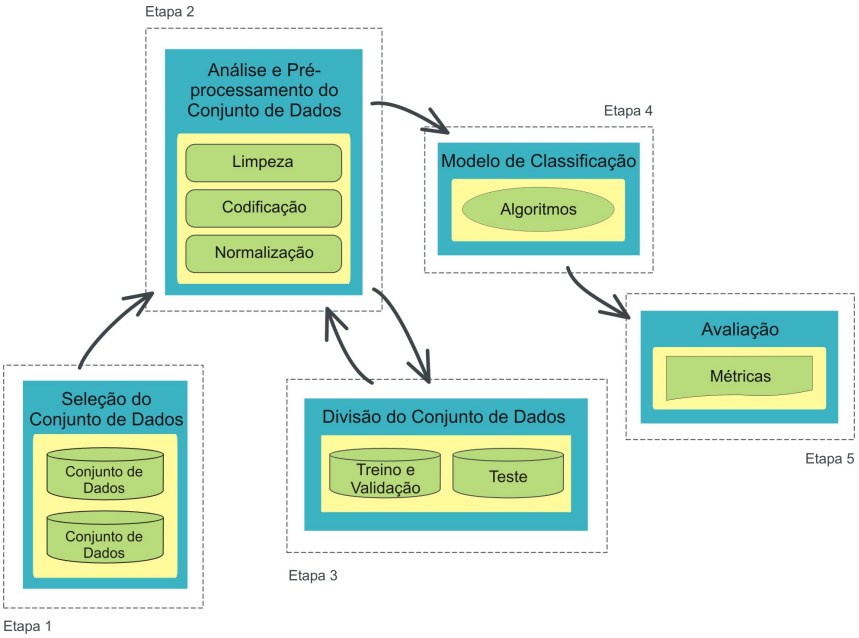


Figura 2.5. Etapas básicas para o desenvolvimento de um IDS baseado em Aprendizado de Máquina.

2.4.2. Etapa 2: Análise e Pré-processamento do Conjunto de Dados

Geralmente, os conjuntos de dados possuem irregularidades que devem ser tratadas. Em sua maioria é possível encontrar valores ausentes, ruídos, inconsistências e redundâncias de dados. Esses problemas podem influenciar na qualidade e na confiabilidade dos modelos de classificação para detecção de intrusão.

Assim a etapa de análise e pré-processamento dos dados é fundamental para a correção desses problemas. Isso envolve a aplicação de métodos consideradas comuns: (i) a limpeza de dados para corrigir problemas como valores ausentes, valores discrepantes e inconsistências; (ii) a codificação que serve para transformar os valores categóricos dos atributos em valores numéricos, visto que dados categóricos não atendem as exigências dos modelos; e (iii) a normalização que é empregada para trazer uniformidade aos valores existentes no conjunto de dados.

Uma boa realização da análise e do pré-processamento de dados pode estabelecer as bases para um modelo robusto, capaz de discernir nuances sutis nos dados, de forma a melhorar a precisão na detecção de intrusão. Alguns métodos de pré-processamento de dados podem ser empregados antes ou após a divisão do conjunto de dados.

A Seção 2.3 aborda, resumidamente, os métodos de pré-processamento de dados empregados no tratamento dos conjuntos de dados, inclusive detalhando os métodos mais comuns.

2.4.3. Etapa 3: Divisão do Conjunto de Dados

Para treinar um modelo de aprendizado de máquina de maneira eficaz, é necessário particionar o conjunto de dados em subconjuntos distintos para treinamento, validação e teste.

Dessa forma, o conjunto de treinamento serve para o modelo aprender os padrões existentes dentro dos dados. Já o conjunto de validação auxilia no ajuste de hiperparâmetros para um bom treinamento do modelo. Por sua vez, o conjunto de teste avalia a generalização e o desempenho do modelo para dados ainda não vistos.

É importante encontrar o equilíbrio certo na divisão dos conjuntos de dados. Assim é possível evitar *overfitting*[3] ou *underfitting*[4], garantindo a robustez do modelo em diferentes cenários.

2.4.4. Etapa 4: Modelos de Classificação

A etapa de escolha do algoritmo de classificação é uma decisão crítica que molda as capacidades de aprendizado do IDS. Existem diferentes algoritmos, como os detalhados na Tabela 2.1, que oferecem pontos fortes e vantagens únicas.

A seleção do algoritmo adequado para o treinamento do modelo de classificação, depende da natureza dos dados e dos requisitos específicos da tarefa de detecção de intrusão. Essa etapa exige um equilíbrio criterioso entre a complexidade do algoritmo, a interpretabilidade e o custo computacional.

A Seção 2.2 aborda com mais detalhes os algoritmos de aprendizado de máquina para a construção de modelos de detecção de intrusão, inclusive descrevendo alguns exemplos.

2.4.5. Etapa 5: Avaliação de desempenho dos modelos

A etapa final é o teste do modelo construído para a detecção de intrusão, que reside na avaliação de desempenho. As métricas mais comuns na avaliação de um IDS são: *Accuracy, Precision, Recall e F1-score*. Essas métricas oferecem *insights* sobre a robustez do modelo na tarefa de classificação das instâncias entre normais e maliciosas.

Além disso, a Matriz de Confusão e a Curva ROC *(Receiver Operating Characteristic)* mostram ainda mais a capacidade do modelo na tarefa de classificação. Essa etapa não apenas valida a confiabilidade do IDS, mas também informa possíveis ajustes e refinamentos para melhorar o seu desempenho futuro.

A Seção 2.2 aborda com mais detalhes as métricas de avaliação, inclusive descrevendo alguns exemplos.

2.5. Problemas e Desafios dos IDS baseados em Anomalias

A literatura mostra que embora existam muitas pesquisas voltadas para a melhoria dos IDS, várias questões ainda precisam ser resolvidas [11, 33]. Os IDS devem ser mais precisos, com a capacidade de detectar uma variedade distinta de intrusões, gerar e atualizar

[3] O *overfitting* é um problema comum no aprendizado de máquina, onde o modelo aprende bem com os dados de treinamento, mas captura os ruídos presentes nos dados. Como resultado, o modelo sobreajustado tende a ter um desempenho ruim em novos dados. Isso ocorre porque o modelo memorizou o conjunto de treinamento, em vez de aprender os padrões gerais que podem ser aplicados as novas instâncias [12].

[4] O *underfitting* ocorre quando o modelo não é complexo o suficiente para aprender as nuances e complexidades do conjunto de dados. Com isso, o resultado é uma falha no ajuste adequado do modelo em relação aos dados de treinamento. Esse problema pode levar o modelo a altas taxas de falsos positivos e falsos negativos, comprometendo a capacidade de detectar e classificar com precisão intrusões na rede [12].

informações sobre novos ataques e emitir menos alarmes falsos.

Um valor de falso positivo (alarme falso) representa o estado em que um evento não é intrusão, mas o modelo classifica erroneamente como intrusão [7]. Dessa forma, é esperado que um IDS robusto tenha a taxa de detecção de intrusão muito alta e a taxa de falsos positivos muito baixa.

Em relação aos conjuntos de dados, alguns estudos apontam que a maioria dos conjuntos de dados públicos não possuem dados suficientes para permitir uma cobertura elevada dos modelos de aprendizado máquina na detecção de diferentes tipos de ataques [25, 18, 10].

Em razão disso, a criação de novos conjuntos de dados para detecção de intrusão que caracterizam ambientes diversificados no tráfego de rede, com relação ao reconhecimento de padrões que permitam ampliar a detecção de ataques e identificar ataques desconhecidos, podem refletir na melhoria dos modelos de aprendizado de máquina. Assim, pode haver um impacto na melhoria dos IDS em relação a detecção de ataques diversificados e a redução de alarmes falsos.

Além disso, é possível encontrar estudos destacando o pré-processamento de dados como uma etapa essencial para melhorar a qualidade dos conjuntos de dados [22, 1, 21]. Isso mostra que é possível explorar os métodos de pré-processamento de dados, visando a criação de novas técnicas que possam diversificar os atributos do tráfego de rede nos conjuntos de dados e, consequentemente, impactar no desempenho dos modelos de aprendizado de máquina na detecção de intrusão.

A seguir, são apresentados algumas limitações e desafios encontrados nos conjuntos de dados para detecção de intrusão, que podem ser explorados em novas pesquisas, para ajudar a melhorar e difundir os IDS baseados em Anomalias:

- A cobertura de ataques é considerado um dos maiores desafios, que impedem os IDS de serem implementados em ambientes de redes reais. Apenas 33%, aproximadamente, dos ataques conhecidos são cobertos pelos conjuntos de dados [8];
- Em relação a simulações que caracterizam estruturas de redes reais, apenas 11% dos IDS usam conjuntos de dados gerados em ambientes reais ou simulados, ou seja, gerados em ambientes que caracterizam tráfegos de rede próximos dos reais [8];
- Os ataques estão evoluindo em um ritmo com o qual os conjuntos de dados não estão conseguindo acompanhar. Assim, novas técnicas de geração de conjuntos de dados são necessárias para mitigar os ataques desconhecidos e ataque de dia zero[5] [29]; e
- A ausência de documentação e informações detalhadas sobre a geração dos conjuntos de dados, contendo as formas de coleta do tráfego de rede, a extração de dados,

[5]Um ataque de dia zero é difícil de prevenir, refere-se à uma vulnerabilidade ainda não explorada. O ataque aproveita a falha de segurança do sistema para o qual nenhuma solução ou defesa foi desenvolvida. Os invasores exploram essas vulnerabilidades antes que os desenvolvedores possam lançar *patches* ou atualizações. Assim é o termo *dia zero*, que indica zero dias de proteção [28].

o pré-processamento de dados e a carga bruta *(Packet Capture - PCAP)* do tráfego, dificultam a comparação de diferentes métodos de detecção de intrusão [5].

Além das limitações e desafios apresentados acima, a Figura 2.6 mostra outras lacunas existentes nos conjuntos de dados para detecção de intrusão, que impactam diretamente nos IDS baseados em Anomalias. Dessa forma, a realização de novas pesquisas para as lacunas nos IDS são cruciais para melhorar as defesas da cibersegurança.

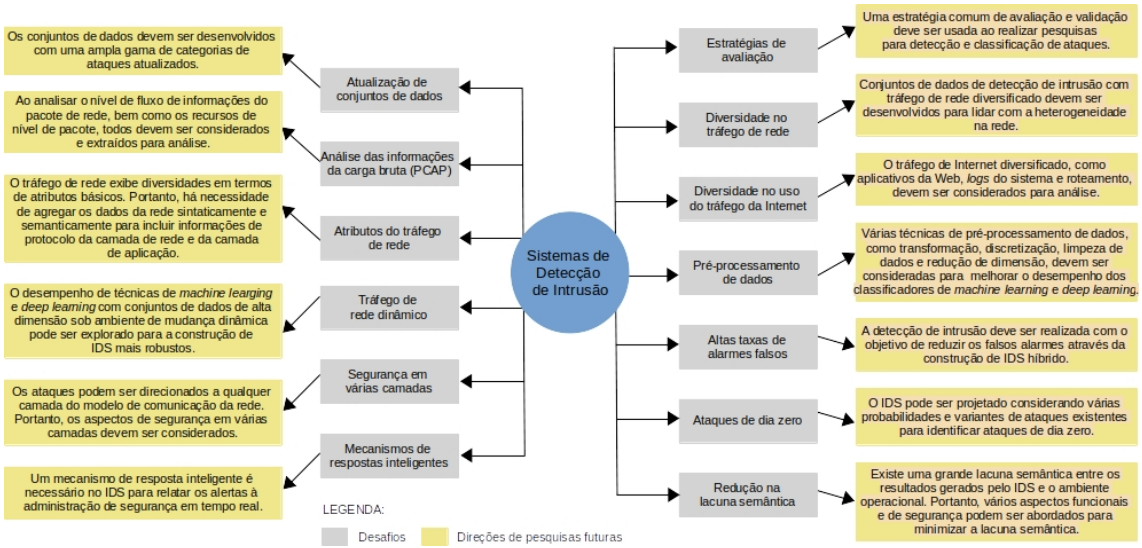


Figura 2.6. Esquema de desafios e direções de pesquisas futuras.

Fonte: Adaptado de [33].

2.6. Materiais e Métodos

Este minicurso visa despertar o interesse dos participantes nos Sistema de Detecção de Intrusão *(Intrusion Detection Systems - IDS)* baseados em anomalias com uso de aprendizado de máquina, ou simplesmente, IDS baseados em Aprendizado de Máquina *(ML-based IDS)*. Para isso, o minicurso foi elaborado para oferecer uma abordagem com *insights* teóricos e aplicações práticas.

A abordagem teórica será interativa, por meio de uma palestra que apresentará material derivado de uma revisão da literatura. Com isso, o principal objetivo é transmitir os fundamentos básicos sobre os seguintes temas apresentados neste material (Figura 2.7):

- Sistemas de Detecção de Intrusão (IDS): descrito na Seção 2.1;
- Aprendizado de Máquina aplicado em IDS: descrito na Seção 2.2;
- Conjuntos de Dados para Detecção de Intrusão: descrito na Seção 2.3;
- Processo de Desenvolvimento de um IDS baseado em Aprendizado de Máquina: descrito na Seção 2.4 ; e
- Problemas e Desafios dos IDS baseados em Anomalias: descrito na Seção 2.5.

Figura 2.7. Temas abordados na apresentação teórica do minicurso.

A abordagem prática também será interativa, por meio de implementação prática com a linguagem de programação Python e bibliotecas de aprendizado de máquina. Com isso, os participantes terão a oportunidade de visualizar a aplicação de algoritmos de aprendizado de máquina em conjuntos de dados para detecção de intrusão, adquirindo experiência prática na construção de IDS baseado em anomalias.

Por se tratar de fundamentos básicos, durante a exploração dos conteúdos práticos, será empregado somente o método de aprendizado supervisionado, que é utilizado com mais frequência nos modelos de IDS para detecção de intrusão. Nesse tipo de método, os conjuntos de dados são rotulados para encontrar a relação entre os dados e suas classes, durante as etapas de treinamento, validação e teste.

Assim, o conjunto de dados CSE-CIC-IDS2018 [30] será empregado na abordagem prática para a construção do modelo de aprendizado de máquina para detecção de intrusão. Trata-se de um conjunto de dados composto pelas seguintes categorias de ataques: *DoS, DDoS, Brute Force, BotNet, Infiltration e Webattack*. Entre essas categorias existem diversas variações.

Além disso, serão aplicadas as técnicas de pré-processamentos mais comuns nos conjuntos de dados para detecção de intrusão, como limpeza de dados, codificação e normalização.

A limpeza de dados é aplicada para lidar com valores ausentes, inconsistentes e redundantes que são comuns em conjuntos de dados. Se esses dados não forem tratados, podem causar ruídos e gerar *outliers* que atrapalham o treinamento e a tomada de decisão do modelo.

Já a codificação é aplicada em atributos com valores categóricos que alguns modelos não podem processar. Com isso, os dados são transformados em valores numéricos para atender aos requisitos do modelo.

Por sua vez, a normalização é aplicada em atributos para reduzir e padronizar es-

calas de dados. Essa técnica elimina ruídos que podem causar *outliers* e ajusta os valores para facilitar a interpretação do modelo.

Em relação ao algoritmos de classificação, serão construídos modelos com o *Random Forest*, que apresenta bons resultados na avaliação de desempenho de conjuntos de dados com método supervisionado. Portanto, para fins de comparação também serão aplicados os algoritmos *kNN (k-Nearest Neighbors)* e *SVM (Support Vector Machine)*.

Por fim, os modelos construídos serão avaliados com as métricas Matriz de Confusão, *Accuracy, Precision, Recall e F1-Score*. Essas métricas são bastante utilizadas na literatura para avaliar modelos de detecção de ataques. A Figura 2.8 ilustra uma visão geral dos métodos utilizados na abordagem prática, já descritos nessa seção.

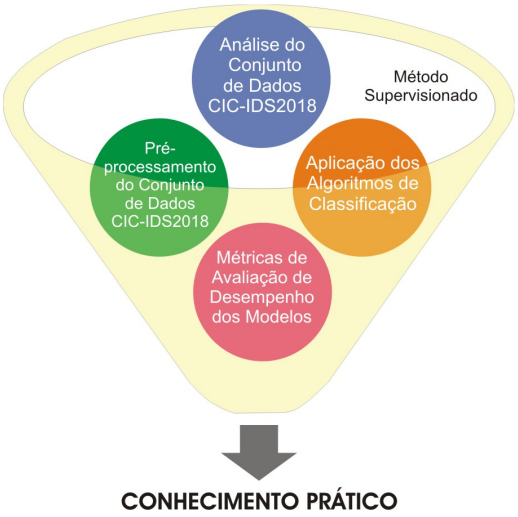


Figura 2.8. Conteúdos aplicados na prática do minicurso.

Em resumo, a abordagens teórica permitirá que os participantes mergulhem nos fundamentos básicos teóricos e compreendam as dificuldades e desafios que esses sistemas enfrentam para serem aplicados no mundo real. Enquanto a abordagem prática promoverá o enriquecimento e o melhor entendimento dos tópicos teóricos abordados.

2.7. Considerações Finais

Ao final do minicurso, os participantes terão uma sólida compreensão das estratégias de detecção de intrusão baseadas em anomalias e sua aplicabilidade na segurança cibernética. Sobretudo, será reforçado, entre os participantes, a importância da atualização contínua das habilidades para enfrentar ameaças em constante evolução e incentivado a exploração mais aprofundada dos tópicos abordados.

Assim, este minicurso ajuda a divulgar conteúdos sobre pesquisas focadas em Sistemas de Detecção de Intrusão *(Intrusion Detection Systems – IDS)* baseados em Anomalias. Dessa forma, melhora a compreensão da evolução das ameaças cibernéticas e da necessidade de mecanismos avançados para detecção de intrusão.

A disseminação da pesquisa promove a colaboração entre profissionais de segurança cibernética, fomentando o desenvolvimento de técnicas de detecção de intrusão

mais robustas e adaptativas. Além disso, a divulgação das pesquisas garantem que as organizações se mantenham informadas sobre as inovações mais recentes.

Portanto, ao partilhar resultados de investigação sobre IDS baseados em Anomalias, a comunidade de segurança cibernética pode contribuir colectivamente para a melhoria das capacidades de detecção de intrusão.

Referências

[1] Ahmad, T. and Aziz, M. N. (2019). Data preprocessing and feature selection for machine learning intrusion detection systems. ICIC Express Lett, 13(2):93–101.

[2] Bezerra, V. H., da Costa, V. G. T., Junior, S. B., Miani, R. S., and Zarpelao, B. B. (2018). One-class classification to detect botnets in iot devices. Anais do XVIII Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais (SBSeg), Natal, RN, 22-25 Outubro 2018.

[3] Choraś, M. and Pawlicki, M. (2021). Intrusion detection approach based on optimised artificial neural network. Neurocomputing, 452:705–715.

[4] Dhaliwal, S. S., Nahid, A., and Abbas, R. (2018). Effective intrusion detection system using xgboost. Information, 9(7):1–24.

[5] Engelen, G., Rimmer, V., and Joosen, W. (2021). Troubleshooting an intrusion detection dataset: the cicids2017 case study. 2021 IEEE Security and Privacy Workshops (SPW), San Francisco, CA, USA, 27 May 2011.

[6] García-Teodoro, P., Díaz-Verdejo, J., Maciá-Fernández, G., and Vázquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. Computers Security, 28(1):18–28.

[7] Hajj, S., El Sibai, R., Bou Abdo, J., Demerjian, J., Makhoul, A., and Guyeux, C. (2021). Anomaly-based intrusion detection systems: The requirements, methods, measurements, and datasets. Transactions on Emerging Telecommunications Technologies, 32(4):1–36.

[8] Hindy, H., Brosset, D., Bayne, E., Seeam, A. K., Tachtatzis, C., Atkinson, R., and Bellekens, X. (2020). A taxonomy of network threats and the effect of current datasets on intrusion detection systems. IEEE Access, 8:104650–104675.

[9] International Telecommunication Union (2021). Global Cybersecurity Index 2020: Measuring commitment to cybersecurity. ITUPublications, Geneva, Switerland, 1 edition.

[10] Kenyon, A., Deka, L., and D., E. (2020). Are public intrusion datasets fit for purpose characterising the state of the art in intrusion event datasets. Computers Security, 99:1–26.

[11] Khraisat, A., Gondal, I., Vamplew, P., and Kamruzzaman, J. (2019). Survey of intrusion detection systems: techniques, datasets and challenges. Cybersecurity, 2(20):1–22.

[12] Koehrsen, W. (2018). Overfitting vs. underfitting: A complete example. Towards Data Science, 405.

[13] Layeghy, S. and Portmann, M. (2022). On generalisability of machine learning-based network intrusion detection systems. arXiv preprint arXiv:2205.04112, [s.n.]:1–12.

[14] Liu, H. and Lang, B. (2019). Machine learning and deep learning methods for intrusion detection systems: A survey. Applied Sciences, 9(20):1–28.

[15] Maciá-Fernández, G., Camacho, J., Magán-Carrión, R., García-Teodoro, P., and Theró, R. (2018). Ugr'16: A new dataset for the evaluation of cyclostationarity-based network idss. Computers Security, 73:411–424.

[16] Mahfouz, A., Abuhussein, A., Venugopal, D., and Shiva, S. (2020). Ensemble classifiers for network intrusion detection using a novel network attack dataset. Future Internet, 12(11):1–19.

[17] Martínez Torres, J., Iglesias Comesaña, C., and García-Nieto, P. J. (2019). Machine learning techniques applied to cybersecurity. International Journal of Machine Learning and Cybernetics, 10:2823–2836.

[18] Molina-Coronado, B., Mori, U., Mendiburu, A., and Miguel-Alonso, J. (2020). Survey of network intrusion detection methods from the perspective of the knowledge discovery in databases process. IEEE Transactions on Network and Service Management, 17(4):2451–2479.

[19] Moustafa, N. and Slay, J. (2015). Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). Paper presented at the 2015 Military Communications and Information Systems Conference (MilCIS), Canberra, ACT, Australia, 10-12 November 2015.

[20] Naseer, M., Rusdi, J. F., Shanono, N. M., Salam, S., Muslim, Z. B., Abu, N. A., and Abadi, I. (2021). Malware detection: issues and challenges. Cybersecurity, 1807(1):1–6.

[21] Obaid, H. S., Dheyab, S. A., and Sabry, S. S. (2019). The impact of data pre-processing techniques and dimensionality reduction on the accuracy of machine learning. Paper presented at the 9th Annual Information Technology, Electromechanical Engineering and Microelectronics Conference (IEMECON), Jaipur, India, 13–15 March 2019.

[22] Paulauskas, N. and Auskalnis, J. (2017). Analysis of data pre-processing influence on intrusion detection using nsl-kdd dataset.

[23] Putra, W. and Huang, J. J. (2019). A survey of intrusion detection system. International Journal of Informatics and Computation, 1(1):1–19.

[24] Resende, P. A. A. and Drummond, A. C. (2018). A survey of random forest based methods for intrusion detection systems. ACM Computing Surveys (CSUR), 51(3):1–36.

[25] Ring, M., Wunderlich, S., Scheuring, D., Landes, D., and Hotho, A. (2019). A survey of network-based intrusion detection data sets. Computers Security, 86:147–167.

[26] Salih, A. A. and Abdulrazaq, M. B. (2019). Combining best features selection using three classifiers in intrusion detection system. 2019 International Conference on Advanced Science and Engineering (ICOASE), Zakho - Duhok, Iraq, 02-04 April 2019.

[27] Saranya, T., Sridevi, S., Deisy, C., Chung, T. D., and Khan, M. A. (2020). Performance analysis of machine learning algorithms in intrusion detection system: A review. Procedia Computer Science, 171:1251–1260.

[28] Sarhan, M., Layeghy, S., Gallagher, M., and Portmann, M. (2023). From zero-shot machine learning to zero-day attack detection. International Journal of Information Security, 22:947–959.

[29] Sarker, I. H., Kayes, A., Badsha, S., Alqahtani, H., Watters, P., and Ng, A. (2020). Cybersecurity data science: an overview from machine learning perspective. Journal of Big data, 7(1):1–29.

[30] Sharafaldin, I., Lashkari, A. H., and Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. Paper presented at the 4th International Conference on Information Systems Security and Privacy (ICISSp), Funchal, Madeira, Portugal, 22–24 January 2018.

[31] Shenfield, A., Day, D., and Ayesh, A. (2018). Intelligent intrusion detection systems using artificial neural networks. Ict Express, 4(2):95–99.

[32] Tavallaee, M., Bagheri, E., Lu, W., and Ghorbani, A. A. (2009). A detailed analysis of the kdd cup 99 data set. Paper presented at the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, ON, Canada, 08-10 July 2009.

[33] Thakkar, A. and Lohiya, R. (2022). A survey on intrusion detection system: feature selection, model, performance measures, application perspective, challenges, and future research directions. Artificial Intelligence Review, 55:453–563.

[34] Viegas, E. K., Santin, A. O., and Oliveira, L. S. (2017). Toward a reliable anomaly-based intrusion detection in real-world environments. Computer Networks, 127:200–216.

[35] Zaman, M. and Lung, C. (2018). Evaluation of machine learning techniques for network intrusion detection. NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium, Taipei, Taiwan, 23-27 April 2018.

Capítulo 3

Tratamento e Análise de Dados de Mobilidade Urbana: Uma metodologia teórica e prática

Edgar Oliveira, Clayson Celes, Carina Oliveira, Reinaldo Braga

Abstract

The growing availability of mobility data in the urban domain brings unprecedented opportunities for knowledge extraction and decision-making in several areas. However, obtaining valuable insights from a large volume of raw data is not a trivial task and requires a detailed process involving techniques, tools, and insights. The aim of this chapter is to present the foundations for the treatment and analysis of mobility data through a proposed methodology that encompasses such a process. Also, it is performed a comparison among Python libraries available in the literature that provide resources for mobility analysis. The libraries are categorized and associated with specific phases of the methodology. The limitations of these libraries are identified, including the lack of support for basic data processing tasks as well as the absence of solutions for trajectory classification, anomaly event detection, and anomalous trajectory detection. These limitations are opportunities for future research and the development of more comprehensive and enhanced methods for mobility analysis. Lastly, we present a real-world use case to exemplify the practical approach, providing results and corresponding source code.

Resumo

A crescente disponibilidade de dados de mobilidade no domínio urbano traz oportunidades sem precedentes para a extração de conhecimento e a tomada de decisão em diversas áreas. No entanto, obter informações valiosas a partir de um grande volume de dados brutos não é uma tarefa simples e exige um processo detalhado com técnicas, ferramentas e insights. O objetivo deste capítulo é apresentar os fundamentos para o tratamento e a análise de dados de mobilidade por meio de uma metodologia proposta que engloba tal processo. Também é realizada uma comparação entre bibliotecas Python disponíveis na literatura que oferecem recursos para a análise de mobilidade. As bibliotecas são classificadas e associadas a fases específicas da metodologia. As limitações das

bibliotecas são identificadas, incluindo a falta de suporte para tarefas básicas de tratamento de dados, bem como ausências de soluções para a classificação de trajetórias, detecção de eventos atípicos e detecção de trajetórias anômalas. Essas limitações são oportunidades para pesquisas futuras e o desenvolvimento de métodos mais abrangentes e aprimorados para a análise de mobilidade. Por fim, um caso de uso com dados reais é empregado para exemplificar a abordagem prática, fornecendo resultados e o código-fonte correspondente.

3.1. Introdução

A disciplina de Ciência de Dados tem ganhado significativa importância em diversas áreas. No contexto de mobilidade urbana, diversas iniciativas têm evidenciado como o conhecimento derivado da análise de dados pode ser altamente benéfico [7, 41, 21]. No entanto, a extração desse conhecimento nem sempre se revela uma tarefa simples e, frequentemente, demanda um conhecimento multidisciplinar. Essa complexidade é ampliada quando se trata de dados de mobilidade [44], que, na maioria das vezes, são dados não estruturados, contendo ruídos, além de serem suscetíveis a imprecisões e lacunas. Consequentemente, a comunidade científica tem dedicado consideráveis esforços no desenvolvimento de ferramentas e bibliotecas destinadas a superar os desafios inerentes ao processo de mineração de dados nesse contexto.

A linguagem de programação Python tem sido adotada no processo de aquisição de conhecimento a partir de dados, especialmente quando se trata de bibliotecas de código aberto. Essa adoção generalizada também se reflete no campo da mobilidade urbana, no qual bibliotecas voltadas para a mineração de dados de mobilidade são amplamente utilizadas [31]. Essa popularidade pode ser atribuída à extensa comunidade que cerca a linguagem, à sua facilidade de uso e ao vasto ecossistema de bibliotecas disponíveis para uma ampla gama de tarefas na ciência de dados, incluindo Pandas [26] e Numpy [13].

Na área de mineração de dados geográficos, que requer a execução de tarefas específicas, encontra-se a biblioteca GeoPandas [16], desenvolvida com base na biblioteca Pandas e desempenhando uma função semelhante para dados geoespaciais. Especificamente no campo da mobilidade, o ecossistema dessas bibliotecas oferece uma variedade de ferramentas valiosas para análise e exploração de dados de mobilidade urbana.

Table 3.1. Bibliotecas para tratamento e análise de dados de mobilidade.

Trackintel [20]	Yupi [27]
Mobilipy[25]	Mobvis [33]
PTRAIL [12]	Mobilkit [39]
Traja [32]	MovingPandas [10]
Mobipy [19]	PyMove [3]
Scikit-Mobility [24]	Tracktable [28]
SMAFramework [30]	mocha [34]
Teetool [8]	

O ecossistema Python oferece diversas bibliotecas para tratamento e a análise de dados de mobilidade urbana, conforme detalhado na Tabela 3.1. Apesar de alguns estudos

realizarem comparações entre algumas dessas bibliotecas disponíveis, a literatura carece de uma análise quantitativa abrangente do estado da arte das bibliotecas Python de código aberto para análise de mobilidade urbana. Uma outra deficiência observada na literatura é a falta de um estudo que apresente uma metodologia e a aplique em um cenário de uso prático, fazendo uso das ferramentas disponíveis no ecossistema de bibliotecas Python. Nesse sentido, o presente estudo abrange desde a teoria até a prática.

Diante desse cenário, o objetivo deste capítulo é apresentar as particularidades e técnicas envolvidas no tratamento e análise de dados de mobilidade urbana. Isso inclui abordar o estado atual da arte e realizar uma comparação quantitativa das bibliotecas disponíveis, além de oferecer uma visão geral de estudos específicos que destacam as tendências de pesquisa e os principais desafios nessa área. Adicionalmente, apresenta-se uma metodologia para a aplicação da análise de dados em projetos relacionados ao contexto urbano, permitindo que os interessados se familiarizem com a área através de um guia prático de trabalho, como ilustrado na Figura 3.1.

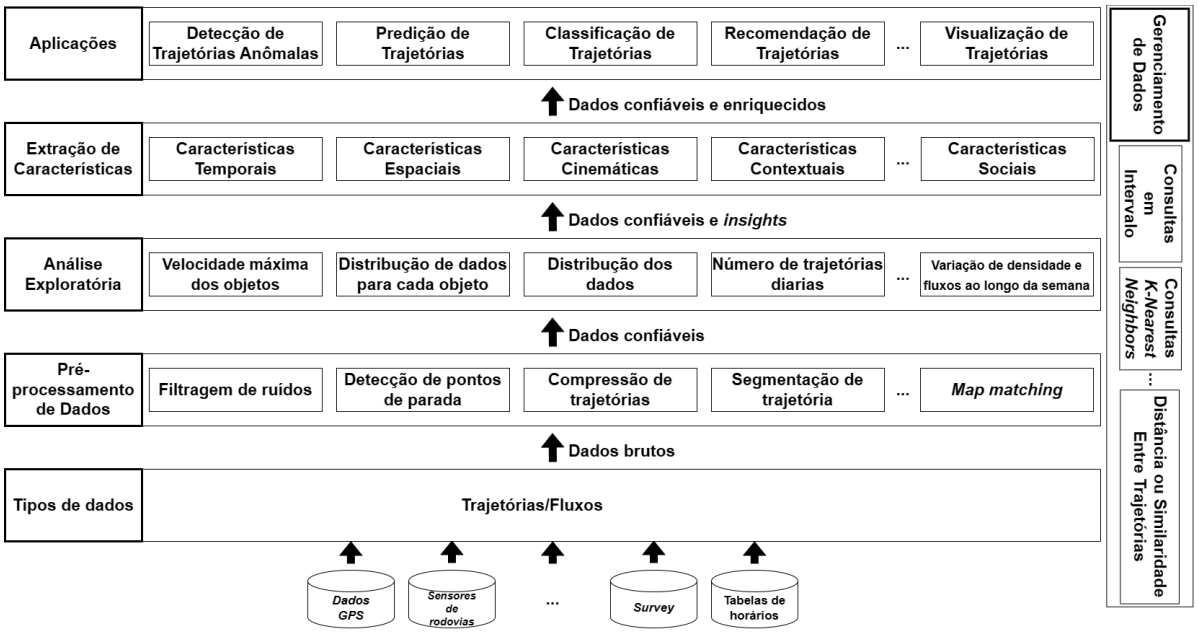


Figure 3.1. Um arcabouço para tratamento e análise de dados de mobilidade.

O capítulo está estruturado da seguinte forma: Na Seção 2 é introduzida a metodologia proposta. A Seção 3 aborda o processo de comparação quantitativa. Na Seção 4 são mostrados os resultados. Por fim, na Seção 5 é realizada uma discussão sobre o conteúdo deste capítulo.

3.2. Metodologia para Tratamento e Análise de Mobilidade

Esta seção aborda a metodologia usada, descrevendo os passos essenciais para adquirir conhecimento, derivados de um paradigma sólido [44], a serem ajustados para incluir outras entidades. A estrutura de tarefas subdivididas permanece, facilitando a busca por recursos nas bibliotecas. Além disso, seu propósito é apresentar informações significativas que vão além do código, fornecendo uma visão abrangente dos contribuintes na

manutenção das bibliotecas. Alguns tópicos relevantes para esta metodologia são: (i) Tipos de fontes geradoras de dados de mobilidade; (ii) Pré-processamento de dados; (iii) Gerenciamento de dados; (iv) Tratamento de incerteza; (v) Mineração de padrões de trajetórias; (vi) Classificação de trajetórias; (vii) Detecção de anomalias em uma trajetória; (viii) Suporte à visualização de mobilidade.

Apesar de o paradigma proposto por Yu Zheng apresentar uma delimitação consistente das categorias de tarefas relacionadas à análise de trajetórias, é importante destacar que a mobilidade pode ser representada por meio de outras entidades. Portanto, com o objetivo de abranger essas diversas entidades, como fluxos, é necessário definir as características das entidades e como as bibliotecas interagem com elas em um momento posterior.

Adicionalmente, reestruturamos o paradigma proposto, mantendo suas subdivisões, de modo a representá-lo por meio de perguntas. Essa abordagem facilita a busca por recursos específicos nas bibliotecas mencionadas, tornando o processo mais eficiente e direcionado.

3.2.1. Tipos de fontes geradoras de dados de mobilidade

De acordo com o trabalho de referência [4], os principais tipos de fontes de dados de mobilidade de veículos podem ser classificados em seis categorias principais, conforme listado a seguir:

- ***Survey***: Estes dados são coletados por meio de questionários respondidos por indivíduos específicos, abrangendo informações como origem e destino da viagem, tempo de viagem, frequência e rotas. Geralmente, são realizados em intervalos de tempo espaçados.

- ***Schedule***: Estes dados são obtidos a partir de agendas pré-determinadas de veículos, geralmente abrangendo um único tipo de transporte, como ônibus.

- ***Road Sensors***: Estes dados são adquiridos por sensores instalados nas margens ou ao longo das vias. Podem fornecer informações como contagem de veículos, fluxo de tráfego e velocidade.

- ***Closed-Circuit Television***: Estes dados consistem em imagens capturadas por câmeras em veículos ou nas vias, que podem ser usadas para extrair informações por meio de técnicas de visão computacional. Assim como os *Road Sensors*, eles têm a desvantagem de exigir um alto custo de infraestrutura para instalação e o monitoramento é restrito a estradas e áreas específicas.

- ***Global Navigation Satellite System (GNSS)***: Dispositivos equipados com GNSS, como Sistema de Posicionamento Global (*Global Positioning System* - GPS), podem registrar a posição geográfica durante o movimento. Portanto, essa fonte de dados fornece informações de posicionamento refinadas e movimento quase em tempo real de um determinado veículo. Geralmente, dados de veículos particulares não são públicos, sendo mais comuns em táxis e ônibus.

- ***Sensing-as-a-Service (SaaS)***: Estas são plataformas que fornecem informações gerais sobre dados de tráfego, geralmente usando técnicas de fusão de dados para classificar a intensidade e fornecer informações de tráfego em ruas específicas, como apresentado no Google Maps ou Waze.

3.2.2. Módulos

Com base na análise nas subdivisões de processos para analisar trajetórias, retiraremos perguntas que mantenham as mesmas tarefas e que facilitem a busca de recursos por parte do leitor. Tais tarefas, descritas em [44] são descritas a seguir e, posteriormente, apresentadas às perguntas.

Trajetórias podem ser representadas de várias maneiras, incluindo pontos geoespaciais, fluxos, imagens, vídeos e texto. Este capítulo lida exclusivamente com trajetórias de GPS e fluxos, conforme definido a seguir.

Neste capítulo, a representação de trajetória utilizada é a de uma sequência de pontos, onde cada elemento P_i pertencente a P consiste em uma tupla com latitude, longitude e tempo. Desta forma, a trajetória é representada por uma sequência de pontos geoespaciais ordenados pelo tempo em que foram coletados, como ilustrado na Figura 3.2.

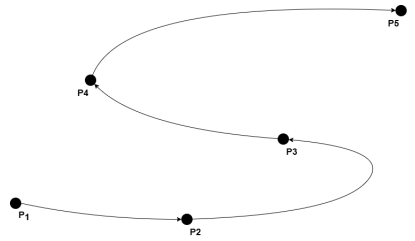


Figure 3.2. Ilustração de uma trajetória.

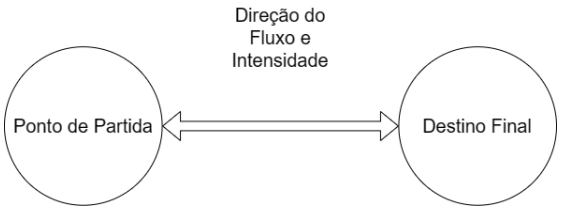


Figure 3.3. Ilustração de fluxo.

Conforme ilustrado na Figura 3.3, neste capítulo, um fluxo refere-se a pontos geoespaciais que podem ou não ter uma direção associada, acompanhados da intensidade do fluxo que ocorre entre esses dois pontos.

3.2.2.1. Classificar as fontes geradoras de trajetória

Durante esta fase da análise de trajetória, é necessária a classificação das fontes geradoras de dados em quatro grandes grupos: mobilidade de pessoas, mobilidade de veículos, mobilidade de animais e mobilidade de eventos naturais. Entretanto, atualmente, não existe nenhuma biblioteca disponível que ofereça funcionalidades para essa tarefa, o que torna impossível a extração de perguntas relacionadas. É importante ressaltar que a classificação das fontes geradoras de dados geralmente é realizada pela entidade responsável pela coleta dos dados e é disponibilizada como metadados.

Adicionalmente, é fundamental levar em consideração que a coleta de dados é uma tarefa complexa e a presença de incertezas é inevitável. Portanto, muitas bibliotecas

apresentam soluções para mitigar esse problema. É importante, assim, considerar a capacidade da biblioteca de representar diferentes tipos de mobilidade e a maneira como os dados são gerenciados.

Diante desta etapa podemos extrair as seguintes Questões de Pesquisa (QP):

QP1) Quais tipo(s) de dados de mobilidade (por exemplo, ponto(s) GPS, segmento(s)) estão(ão) disponível(is) como representação(ões) para análise nas bibliotecas?

QP2) Explora qual(is) tipo(s) de representação de mobilidade?

3.2.2.2. Pré-processamento de Dados

Este estágio da análise de trajetória envolve o tratamento dos dados coletados na etapa anterior. As tarefas aqui se dividem em cinco categorias, descritas a seguir:

FILTRAGEM DE RUÍDOS. Os dados de trajetória frequentemente apresentam erros devido a diversos fatores, tais como a qualidade dos sensores ou falhas de comunicação. Nestes casos, é necessário remover as falhas da trajetória usando algoritmos que se dividem em três grandes subcategorias: *Mean Filter*, Kalman e *Heuristics-Based Outlier Detection*.

DETECÇÃO DE PONTOS DE PARADA. Em uma trajetória, alguns pontos podem revelar informações importantes, como uma parada para abastecer o carro, chamados de pontos de parada. Em alguns casos, é necessário extrair esses pontos para melhorar a acurácia, enquanto em outros, como para estimar o tempo de viagem, é preciso removê-los e computar apenas os deslocamentos.

COMPRESSÃO DE TRAJETÓRIAS. Para tornar o envio, armazenamento e computação de trajetórias mais eficientes, é necessário reduzir o número de pontos sem perder muito de sua precisão. A literatura apresenta diversos algoritmos para realizar esta tarefa, incluindo *Distance Metric*, *Offline Compression*, *Online Data Reduction* e *Compression with Semantic Meaning*.

SEGMENTAÇÃO DE TRAJETÓRIA. Além de reduzir o poder computacional necessário para a análise, a divisão de trajetórias em partes menores permite extrair padrões de subtrajetórias e enriquecer o conhecimento. A segmentação ou clusterização de trajetória pode ser feita com base em espaços temporais, mudanças de ângulo e usando algoritmos de simplificação de linha.

***Map matching*.** O processo de *map-matching* consiste em reposicionar pontos de uma trajetória em uma estrada ou rodovia, corrigindo erros causados por imprecisão na coleta dos dados. As informações adicionais utilizadas ou o intervalo de pontos de amostragem podem ser levados em conta durante o processo.

Diante desta etapa podemos extrair a seguinte Questão de Pesquisa (QP):

QP3) Qual(is) técnica(s) de pré-processamento esta(ão) implementada(s)?

3.2.2.3. Gerenciamento de Dados

Esta etapa é especialmente importante à medida que a quantidade de dados analisados aumenta, já que o acesso a várias partes de trajetórias em um grande volume de dados pode ser oneroso. Assim, é desejável que as ferramentas utilizadas para análise possuam um gerenciamento eficiente de dados e ofereçam técnicas de busca eficazes, tais como as listadas a seguir.

CONSULTAS EM INTERVALO. Existem diversas abordagens para o gerenciamento eficiente de trajetórias. Uma delas é organizar as trajetórias em uma estrutura de árvore como *3D-Rtree*, o que permite realizar buscas espaço-temporais representadas com uma caixa 3D, onde o resultado são os nós da árvore que estiverem dentro da caixa. Outra abordagem possível é dividir um período em vários intervalos de tempo, construindo um índice espacial individual como uma *R-tree* para as trajetórias geradas em cada intervalo. Por fim, é possível particionar um espaço geográfico em grades e construir um índice para as trajetórias que caem em cada grade.

CONSULTAS *K-Nearest Neighbors* (KNN). A abordagem de encontrar k trajetórias com a menor distância agregada em relação a alguns pontos podem ser útil em diversas situações, como na identificação de todos os carros que passaram por determinada estrada ou rodovia. Essa técnica é especialmente importante em análises de dados de mobilidade urbana, permitindo uma compreensão mais precisa do tráfego em uma determinada região e fornecendo informações valiosas para o planejamento urbano.

DISTÂNCIA OU SIMILARIDADES ENTRE TRAJETÓRIAS. A similaridade entre duas trajetórias é frequentemente calculada por meio da distância agregada entre os pontos, utilizando algoritmos como *Closest-Pair Distance*, *Sum-of-Pairs Distance*, *Dynamic Time Wrapping*, *Longest Common Sub-Sequence* (LCSS), *LCSS-based Distance*, entre outros.

Diante desta etapa podemos extrair a seguinte Questão de Pesquisa (QP):

QP4) Qual(is) recurso(s) para gerenciamento de dados esta(ão) disponível(is)?

3.2.2.4. Incerteza

Como o movimento de um objeto no espaço é contínuo, enquanto seu registro é discreto, as informações sobre uma trajetória sempre terão lacunas e, portanto, incertezas. Para reduzir essa incerteza, existem diversas técnicas, bem como métodos para aumentar a incerteza e proteger a privacidade do usuário.

REDUZINDO A INCERTEZA DE DADOS DE TRAJETÓRIA. Existem duas categorias principais de métodos para gerenciar a incerteza em trajetórias. A primeira envolve modelos que recuperam trajetórias existentes usando diferentes tipos de busca na base de dados, como por meio de objetos geométricos ou funções de densidade e probabilidade independentes em cada ponto do tempo. Já a segunda categoria envolve a inferência de caminhos a partir das lacunas existentes, interpolando os pontos faltantes para construir a trajetória [5]. Em algumas situações, é possível melhorar as inferências utilizando algoritmos adicionais, como o *map-matching*.

PRIVACIDADE DE DADOS DE TRAJETÓRIA. Serviços baseados em localização em tempo real podem revelar informações sensíveis sobre a localização de um usuário. Para evitar isso, existem diversas abordagens, como o uso de técnicas de ocultação de localização, como *spatial cloaking* [6] e *mix-zones* [2]. No caso de trajetórias históricas, publicar múltiplas trajetórias de um mesmo indivíduo pode permitir a inferência de informações sensíveis, como sua residência ou local de trabalho. Algoritmos como *clustering-based* [1], *generalization-based* [23], *suppression-based* [38], *grid-based* [9] e outros podem ser aplicados para proteger a privacidade em tais cenários.

A partir desta etapa, podemos extrair as seguintes perguntas:

QP5) Qual(is) método(s) para lidar com a incerteza esta(ão) implementado(s)?

QP6) É possível gerar dados de mobilidade?

3.2.2.5. Mineração de Padrões de Trajetória

A extração de padrões, seja de uma única trajetória ou de um conjunto delas, se divide em quatro grandes categorias, que descrevemos a seguir.

***Moving Together*.** Mostrar que um grupo de objetos se movem juntos durante um intervalo de tempo pode ser feito usando algoritmos como *flock* [11], *convoy* [14, 15], *swarm* [18], *travelling companion* [37, 36] e *gathering* [42, 43].

***Trajectory Clustering*.** Ao buscar por um caminho representativo ou uma tendência compartilhada por um conjunto de trajetórias, é comum agrupá-las em *clusters*. No cenário de redes rodoviárias, existem vários estudos sobre o agrupamento de trajetórias, como o trabalho de [17]. Para solucionar esse problema, pode-se combinar *map-matching* com algoritmos de agrupamento de grafos.

MINERAÇÃO DE PADRÕES SEQUENCIAIS DE TRAJETÓRIA. Na análise de trajetórias, um padrão sequencial é identificado quando um determinado número de objetos em movimento percorre uma sequência comum de locais em um intervalo de tempo semelhante. Para que uma sequência seja considerada um padrão, geralmente é necessário que ela seja encontrada mais vezes do que um valor de limiar (*threshold*) previamente estabelecido.

MINERAÇÃO DE PADRÕES PERIÓDICOS DE TRAJETÓRIAS. É comum encontrar padrões periódicos em dados de trajetória, pois os locais que uma pessoa visita diariamente tendem a se repetir. Encontrar esses padrões podem ser útil para prever movimentos futuros de objetos, além de ajudar na compressão de suas trajetórias, reduzindo o espaço necessário para armazená-las. Diversos algoritmos têm sido desenvolvidos para a detecção de padrões periódicos, como *Fourier transform* [2] e algoritmos baseados em *clustering* [22].

Diante desta etapa podemos extrair a seguinte Questão de Pesquisa (QP):

QP7) Qual(is) técnica(s) de mineração de padrões de trajetórias esta(ão) implementada(s)?

3.2.2.6. Classificação de Trajetória

A classificação de trajetórias tem como objetivo encontrar diferenças entre trajetórias ou segmentos de trajetórias em diferentes aspectos, como movimentos, modos de transporte e atividades humanas. Tipicamente, esse processo é dividido em três etapas: primeiro, as trajetórias são divididas em segmentos usando métodos de segmentação; em seguida, características são extraídas desses segmentos; e, finalmente, um modelo é criado para classificar cada segmento. A segmentação pode ser baseada em vários critérios, como a velocidade ou a direção da trajetória. As características podem ser extraídas de várias maneiras, como a distância percorrida, a aceleração e a direção. Várias técnicas de aprendizado de máquina são usadas para criar modelos de classificação, incluindo KNN, *Support Vector Machine* (SVM) e redes neurais.

Diante desta etapa podemos extrair a seguinte Questão de Pesquisa (QP):

QP8) Qual(is) técnica(s) para predição ou classificação de trajetórias esta(ão) implementada(s)?

3.2.2.7. Detecção de Anomalias em uma Trajetória

A detecção de anomalias em trajetórias é uma tarefa importante em diversas aplicações, como monitoramento de veículos e detecção de comportamentos suspeitos. As anomalias podem ser identificadas como pontos ou segmentos que se destacam dos demais em relação a alguma métrica, como a distância espacial percorrida ou a quebra de um padrão encontrado nas demais trajetórias. Existem diversas técnicas para a detecção de anomalias, que variam desde a análise visual até o uso de modelos estatísticos e de aprendizado de máquina. A detecção de anomalias pode ser realizada tanto em tempo real quanto em trajetórias históricas, permitindo a identificação de eventos incomuns e a tomada de decisões mais eficientes.

DETECTANDO TRAJETÓRIAS ATÍPICAS. Trajetórias atípicas são aquelas que se diferenciam das demais em termos de forma ou tempo de viagem. Essas trajetórias podem ser detectadas por meio de algoritmos de *clustering* baseados em densidade ou por meio de mineração de padrões de frequência. Se uma trajetória ou segmento não se encaixa em nenhum *cluster* ou não é frequente o suficiente, é considerada uma anomalia.

DETECÇÃO DE EVENTOS ATÍPICOS EM TRAJETÓRIAS. Além da detecção de anomalias em trajetórias individuais, é possível identificar eventos anormais por meio do uso de um grande número de trajetórias. Uma abordagem para detectar anomalias no trânsito é representar o evento como um subgrafo das redes de rodovias da cidade, onde os motoristas exibem comportamentos diferentes de seus padrões históricos. É possível, então, descrever o evento por meio da mineração de termos significativos postados em redes sociais no local da ocorrência.

Diante desta etapa podemos extrair as seguintes Questões de Pesquisa (QP):

QP9) Quais técnicas para detecção de anomalias estão implementadas?

3.2.2.8. Suporte à Visualização de Dados de Trajetória

A visualização de trajetórias não se limita apenas à exibição de dados brutos, como em uma tabela na biblioteca [26]. É importante destacar que a exibição simples de dados pode não ser esclarecedora para os usuários, pois não explora as informações geoespaciais. Portanto, é fundamental que haja métodos que enriqueçam as informações geográficas, por meio de imagens, gráficos e mapas interativos.

Diante desta etapa podemos extrair a seguinte Questão de Pesquisa (QP):

QP10) Existe suporte para visualização?

3.3. Bibliotecas e uma Comparação Qualitativa

Nesta seção é realizada uma descrição e uma comparação de bibliotecas para tratamento e análise de mobilidade. Essa comparação se torna essencial para destacar os métodos presentes no estado da arte das bibliotecas do ecossistema Python, bem como para identificar as fortalezas e fraquezas de cada ferramenta ao longo das diferentes etapas da descoberta de informações. Dessa forma, diversas oportunidades de melhoria e trabalhos futuros podem ser reveladas.

3.3.1. Descrição das Bibliotecas

Nesta seção, são apresentadas as bibliotecas listadas na Tabela 3.1. Além disso, métricas comparativas com base nas informações disponíveis no GitHub podem ser encontradas juntamente com o código-fonte do capítulo[1]. Dessa forma, mais informações estão disponíveis externamente para manter o capítulo conciso.

TRACKINTEL [20]: O *Trackintel* tem como objetivo preencher a lacuna de ferramentas voltadas para a etapa de pré-processamento, fornecendo recursos para padronizar essas tarefas e aumentar a reprodutibilidade e comparabilidade de estudos científicos. Embora a mineração de trajetórias não seja o seu foco principal, a biblioteca apresenta algumas limitações em relação a outras bibliotecas em termos de métodos disponíveis para a etapa de mineração.

YUPI [27]: O *Yet Underused Path Instruments* (*Yupi*) é uma biblioteca Python para mineração de trajetórias que busca adicionar novos recursos enquanto aproveita recursos já existentes. Para isso, *Yupi* oferece APIs e métodos que permitem o uso de outras bibliotecas para análise de trajetórias, além de disponibilizar recursos próprios, como a extração de trajetórias de vídeos. Além disso, a biblioteca é altamente configurável e extensível, permitindo que os usuários personalizem o processo de análise de acordo com suas necessidades.

MOBILIPY [25]: Esta biblioteca é uma ferramenta completa para análise de mobilidade, fornecendo recursos para todos os estágios do processo de mineração. Além de permitir a segmentação de trajetórias em viagens e atividades, com detecção de locais de "casa" e "trabalho", o pacote também disponibiliza ferramentas para proteger a privacidade dos usuários, tornando a desanonimização dos dados de mobilidade mais difícil.

[1] https://github.com/edgarsoliveira1/MINICURSO-ERCEMAPI2023

MOBVIS [33]: O *MobVis* é uma biblioteca para visualização de métricas de mobilidade, com suporte para pré-processamento e extração de métricas. Embora seja uma biblioteca nova e não tenha documentação, é possível usar *docstrings* para entender melhor a ferramenta.

PTRAIL [12]: Desenvolvida para o pré-processamento de dados de trajetória, o *PTRAIL* oferece funcionalidades como filtragem de ruídos, remoção de *outliers*, interpolação e extração de *features*. O pacote também se destaca pela execução paralela e vetorizada dessas tarefas, aumentando a eficiência do processamento. Apesar de não oferecer ferramentas para análise de dados, o *PTRAIL* apresenta uma integração bem definida com outras bibliotecas, como GeoPandas[16].

MOBILKIT [39]: Diferenciando-se das outras bibliotecas livres listadas, o *Mobilkit* possui um enfoque especial na análise de mobilidade pós-desastres naturais, como terremotos e enchentes. Construído sobre o framework *Dask* [29], que tem como objetivo facilitar o processamento de grandes volumes de dados, o *Mobilkit* oferece recursos adicionais para análise de dados geo-temporais. No entanto, uma limitação desta biblioteca é a falta de recursos para proteger a privacidade dos usuários, o que é uma preocupação crescente no campo da mineração de dados.

TRAJA [32]: O *Traja* é uma biblioteca Python com foco na análise de trajetórias de animais. Além das ferramentas convencionais de mineração de trajetórias, como aquelas baseadas em redes neurais, a biblioteca oferece recursos específicos para este contexto, como a extração de trajetórias de vídeos que atendam a determinadas restrições.

MOVINGPANDAS [10]: É uma biblioteca de código aberto que oferece recursos para explorar e analisar dados de mobilidade. A biblioteca é uma extensão do Pandas, Geopandas [16] e HoloViz [35], e fornece estruturas de dados e funções para cálculo de velocidade e direção, extração de pontos de parada, divisão de trajetórias em sub-viagens, agregação, generalização e visualização estática e dinâmica dos dados de trajetória.

MOBIPY [19]: Construída em cima da biblioteca Pandas, *Mobipy* busca simplificar o cálculo de métricas e padrões de mobilidade de dados geo-temporais. Com uma sólida base de manipulação de dados na Mineração de Dados, possui recursos para competir com as outras bibliotecas aqui listadas em termos de desempenho. No entanto, é notável a limitação de métricas disponíveis, mesmo quando contamos com seus módulos de funções utilitárias internas.

PYMOVE [3]: O *PyMove* é uma biblioteca em Python para o processamento e visualização de trajetórias. Uma de suas principais características é a ênfase na flexibilidade e extensibilidade do código, permitindo que o usuário adicione funcionalidades conforme necessário. Para isso, o código segue padrões como as regras *PEP8* [40] e possibilita a alteração da estrutura de baixo nível, como a utilização de dados que não cabem em memória, utilizando a biblioteca *Dask* [29]. Essas características tornam o *PyMove* uma ferramenta interessante para a comunidade de análise de mobilidade que busca soluções personalizáveis e escaláveis para o processamento de grandes volumes de dados de trajetórias.

SCIKIT-MOBILITY [24]: É um software estatístico que fornece suporte para cientistas e profissionais em diversos aspectos da análise de mobilidade. Essa biblioteca é bas-

tante popular em comparação com outras na área de mineração de mobilidade. Além das funcionalidades básicas, o *Scikit-Mobility* oferece ferramentas para análise de riscos à privacidade dos usuários e a representação de fluxos, algo que não é encontrado em outros pacotes. Isso permite que os usuários possam lidar com questões críticas de segurança em suas análises de mobilidade. Outra característica notável do *Scikit-Mobility* é a sua capacidade de trabalhar com dados de mobilidade em grande escala, permitindo a análise de conjuntos de dados de milhões de pontos em tempo hábil.

TRACKTABLE [28]: Uma biblioteca que se destaca pela sua eficiência no processamento de grandes volumes de dados de trajetória, graças à implementação de suas principais estruturas de dados e algoritmos em C++. Além de recursos comuns em outras bibliotecas, como a visualização e análise de trajetórias, ela oferece recursos únicos, como a geração de mapas de calor e a detecção de quadrantes em trajetórias. Esses recursos podem ser úteis em diversas aplicações, desde o monitoramento de veículos em tempo real até a análise de movimentos de animais.

SMAFRAMEWORK [30]: O *SMAFramework* é uma biblioteca de análise de mobilidade urbana que foca na integração de múltiplos *datasets* heterogêneos. Possui ferramentas para gerenciamento e análise de dados, mas não possui documentação, o que torna seu uso mais desafiador em comparação com as demais bibliotecas.

MOCHA [34]: O *MOCHA* é uma ferramenta de pré-processamento de trajetórias que extrai métricas temporais, espaciais e sociais, além de classificar as trajetórias com base em suas distribuições de probabilidade. Também produz guias visuais para comparação de traços e modelos de mobilidade, útil para avaliar a qualidade de trajetórias reais ou sintéticas.

TEETOOL [8]: Com o objetivo de fornecer recursos para a análise de trajetórias em duas ou três dimensões, o *Teetool* oferece ferramentas para minerar dados e gerar predições. Além de métricas comuns, como distância e velocidade, também apresenta métricas específicas, como a região de confiança, úteis para a análise de trajetórias de objetos aéreos. Embora disponha de algumas ferramentas para outras etapas da mineração, como visualização, é necessário recorrer a outras bibliotecas para realizar todas as tarefas envolvidas na mineração de trajetórias.

3.3.2. Uma Comparação Qualitativa

Baseados nos módulos previamente apresentados e nas questões de pesquisas (QP) levantadas na seção 3.2.2, elaborou-se uma comparação qualitativa para verificar a abragência e a cobertura das bibliotecas nas tarefas de tratamento e análise de dados de mobilidade. Por meio dessa comparação, torna-se possível apresentar uma classificação para cada biblioteca, definindo qual desempenha melhor por cenário.

3.3.2.1. Quais bibliotecas escolher com base nos dados?

Todos os tipos de representação de mobilidade abordados neste estudo, GNSS, são suportados por todas as ferramentas apresentadas. No entanto, ao considerar o tipo de entidade conforme a Tabela 3.3, é perceptível que, ao lidar com fluxos, a escolha se restringe

a cinco bibliotecas. Nesse aspecto, é importante ressaltar que *MovingPandas*, *Scikit-Mobility* e *PyMove* oferecem um suporte mais abrangente para outras tarefas. Nesse sentido, com base na Tabela 3.3, as seguintes questões de pesquisas são respondidas: **QP1) Quais tipo(s) de dados de mobilidade (por exemplo, ponto(s) GPS, segmento(s)) estão(ão) disponível(is) como representação(ões) para análise nas bibliotecas?** e **QP2) Explora qual(is) tipo(s) de representação de mobilidade?**.

Table 3.3. Tipo de entidades móveis.

Bibliotecas	Trackintel	Yupi	Mobilipy	MobVis	PTRAIL	Mobilkit	Traja	MovingPandas	Mobipy	PyMove	Scikit-Mobility	Tracktable	SMAFramework	MOCHA	Teetool
Trajetória	+	+	+	+	+	+	+	+	+	+	+	+	-	+	+
Fluxo	-	-	+	-	-	-	-	+	-	+	+	-	+	-	-

3.3.2.2. Bibliotecas para tarefas de pré-processamento

Ao analisar os dados apresentados na Tabela 3.5, torna-se evidente a única biblioteca que abrange as outras cinco tarefas mencionadas é a *PyMove*. No entanto, considerando a possibilidade da utilização de várias bibliotecas, como *MovingPandas* e *Scikit-Mobility*, nas quais os dados são armazenados de maneira compatível, é factível que um desempenho semelhante seja alcançado. Portanto, é viável que a combinação dessas bibliotecas seja explorada para a obtenção de suporte quase completo para as atividades de pré-processamento. Nesse sentido, com base na Tabela 3.5, a seguinte questão de pesquisa é respondida: **QP3) Qual(is) técnica(s) de pré-processamento esta(ão) implementada(s)?**.

Table 3.5. Técnicas de pré-processamento.

Bibliotecas	Trackintel	Yupi	Mobilipy	MobVis	PTRAIL	Mobilkit	Traja	MovingPandas	Mobipy	PyMove	Scikit-Mobility	Tracktable	SMAFramework	MOCHA	Teetool
Filtragem de ruídos	-	+	-	-	+	-	-	+	-	+	+	+	-	-	-
Detecção de pontos de parada	+	-	+	+	-	+	-	+	-	+	+	-	-	-	-
Compressão de trajetórias	-	-	+	-	-	+	+	-	-	+	+	-	-	-	-
Segmentação de trajetória	-	-	+	-	+	-	+	+	-	+	-	-	-	-	-
Map matching	-	-	-	-	-	-	-	-	-	+	-	-	-	-	-

3.3.2.3. Bibliotecas para gerenciar mobilidade

Uma conclusão que podemos derivar da Tabela 3.6 é que, ao lidar com um grande volume de dados, é viável empregar a ferramenta *PyMove* para a filtragem das tabelas específicas

desejadas. Posteriormente, caso seja necessário, pode-se recorrer a outras bibliotecas para realizar tarefas adicionais. Essa abordagem permite uma manipulação de dados de mobilidade eficaz e modular. Nesse sentido, com base na Tabela 3.6, a seguinte questão de pesquisa é respondida: **QP4) Qual(is) recurso(s) para gerenciamento de dados esta(ão) disponível(is)?**.

Table 3.6. Recursos para gerenciamento de dados.

Bibliotecas	Trackintel	Yupi	Mobilipy	MobVis	PTRAIL	Mobilkit	Traja	MovingPandas	Mobipy	PyMove	Scikit-Mobility	Tracktable	SMAFramework	MOCHA	Teetool
Consultas em Intervalo	-	-	-	-	+	+	+	+	-	+	-	+	-	-	-
Consultas KNN	-	-	+	-	-	-	-	-	-	+	-	-	-	-	-
Similaridade de Trajetórias	-	-	-	-	-	-	-	-	+	+	-	-	-	-	-

3.3.2.4. Bibliotecas para lidar com incerteza e análise de privacidade

O exame da Tabela 3.7 revela que a escassez de métodos para lidar com incerteza é perceptível, uma vez que apenas 3 métodos oferecem ferramentas para a suavização da trajetória, a saber, *MovingPandas*, *Traja* e *PTRAIL*. Além disso, a análise de riscos para a privacidade é apoiada apenas pela biblioteca *Scikit-Mobility*, o que evidencia uma oportunidade para a implementação de novas abordagens. Nesse sentido, com base na Tabela 3.7, a seguinte questão de pesquisa é respondida: **QP5) Qual(is) método(s) para lidar com a incerteza esta(ão) implementado(s)?**.

Table 3.7. Métodos para lidar com incerteza.

Bibliotecas	Trackintel	Yupi	Mobilipy	MobVis	PTRAIL	Mobilkit	Traja	MovingPandas	Mobipy	PyMove	Scikit-Mobility	Tracktable	SMAFramework	MOCHA	Teetool
Reduzindo a Incerteza	-	-	-	-	+	-	+	+	-	-	-	-	-	-	-
Análise de Privacidade	-	-	-	-	-	-	-	-	-	-	+	-	-	-	-

3.3.2.5. Bibliotecas para gerar mobilidade

Na tarefa de geração de mobilidade sintética, conforme demonstrado na Tabela 3.8, observa-se que nenhum dos recursos disponíveis nas bibliotecas se equipara em abrangência aos oferecidos pela *Scikit-Mobility*. No entanto, em outras bibliotecas, estão disponíveis recursos de geração de trajetórias por meio de métodos mais simples, como o *Random Walk*. A ausência de métodos mais avançados revela uma carência de abordagens robustas para

a geração de mobilidade sintética. Este é um ponto que merece atenção e pode ser explorado como uma oportunidade para o desenvolvimento de novas técnicas e métodos nessa área. Nesse sentido, com base na Tabela 3.8, a seguinte questão de pesquisa é respondida: **QP6) É possível gerar dados de mobilidade?**.

Table 3.8. Métodos para gerar mobilidade.

Bibliotecas	Trackintel	Yupi	Mobilipy	MobVis	PTRAIL	Mobilkit	Traja	MovingPandas	Mobipy	PyMove	Scikit-Mobility	Tracktable	SMAFramework	MOCHA	Teetool
Gerar Trajetória	-	+	-	-	-	-	+	-	-	-	+	+	-	-	+
Gerar Fluxo	-	-	-	-	-	-	-	-	-	-	+	-	-	-	-
Mobility Diary	-	-	-	-	-	-	-	-	-	-	+	-	-	-	-

3.3.2.6. Bibliotecas para mineração de padrões de trajetórias

É possível que várias ferramentas sejam utilizadas em conjunto para obter uma gama mais completa de opções, como ilustrado na Tabela 3.9, combinando o uso do *Scikit-Mobility* com o *MovingPandas* ou *Traja*, que são compatíveis entre si. No entanto, não foi encontrada uma biblioteca, dentre as listadas, que disponha de recursos específicos para a mineração de padrões de movimentação conjunta (*Moving Together*), além da escassez de suporte para a mineração de padrões sequenciais e periódicos. Esses são aspectos que indicam possíveis áreas de desenvolvimento e aprimoramento nas bibliotecas existentes. Nesse sentido, com base na Tabela 3.9, a seguinte questão de pesquisa é respondida: **QP7) Qual(is) técnica(s) de mineração de padrões de trajetórias esta(ão) implementada(s)?**.

Table 3.9. Técnicas de minerações de padrões.

Bibliotecas	Trackintel	Yupi	Mobilipy	MobVis	PTRAIL	Mobilkit	Traja	MovingPandas	Mobipy	PyMove	Scikit-Mobility	Tracktable	SMAFramework	MOCHA	Teetool
Moving Together	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Agrupamento de Trajetórias	-	-	+	-	-	+	+	+	+	+	+	+	+	-	-
Padrões Sequenciais	-	-	-	-	-	-	-	-	-	-	+	-	-	-	+
Padrões Periódicos	-	-	-	-	-	-	+	+	-	-	-	-	-	-	-

3.3.2.7. Bibliotecas para tarefa de classificação ou predição de trajetória

Conforme observado na Tabela 3.10, nenhuma das ferramentas listadas apresenta uma solução específica para a classificação. Isso indica que há uma lacuna nas bibliotecas

existentes em relação a esse aspecto particular da mineração de trajetórias. Seria interessante explorar o desenvolvimento de métodos ou a integração de outras bibliotecas para abordar essa necessidade.

Não obstante, apenas a biblioteca *Traja* oferece uma solução para o problema de predição de trajetória. Isso destaca a importância de explorar e implementar abordagens específicas para essa tarefa, considerando sua relevância na análise de dados de mobilidade. Nesse sentido, com base na Tabela 3.10, a seguinte questão de pesquisa é respondida: **QP8) Qual(is) técnica(s) para predição ou classificação de trajetórias esta(ão) implementada(s)?**.

Table 3.10. Técnicas para predição ou classificação de trajetórias

Bibliotecas	Trackintel	Yupi	Mobilipy	MobVis	PTRAIL	Mobilkit	Traja	MovingPandas	Mobipy	PyMove	Scikit-Mobility	Tracktable	SMAFramework	MOCHA	Teetool
Predição	-	-	-	-	-	-	+	-	-	-	-	-	-	-	-
Classificação	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

3.3.2.8. Bibliotecas para detecção de anomalias

Conforme pode ser observado na Tabela 3.11, outra lacuna é evidenciada no estado da arte das bibliotecas Python para mineração de dados de mobilidade. Nenhuma solução é apresentada para a detecção de eventos atípicos. Apenas a biblioteca *Tracktable* oferece uma solução para a detecção de trajetórias anômalas. Isso realça a necessidade de ferramentas adicionais e o desenvolvimento de métodos específicos para a detecção de anomalias em dados de mobilidade, com o propósito de explorar todo o potencial dessas informações e identificar padrões incomuns ou comportamentos fora do esperado. Nesse sentido, com base na Tabela 3.11, a seguinte questão de pesquisa é respondida: **QP9) Quais técnicas para detecção de anomalias estão implementadas?**.

Table 3.11. Técnicas para detecção de anomalias.

Bibliotecas	Trackintel	Yupi	Mobilipy	MobVis	PTRAIL	Mobilkit	Traja	MovingPandas	Mobipy	PyMove	Scikit-Mobility	Tracktable	SMAFramework	MOCHA	Teetool
Detectando Trajetórias Atípicas	-	-	-	-	-	-	-	-	-	-	-	+	-	-	-
Detecção de Eventos Atípicos	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

3.3.2.9. Bibliotecas para visualização de mobilidade

Ao analisar os dados apresentados na Tabela 3.12, a ampla variedade de métodos disponíveis para a visualização de dados de mobilidade é evidente. É importante ressaltar que a biblioteca *MobVis* tem como foco exclusivo essa tarefa específica da mineração de dados de mobilidade. Portanto, a ausência de recursos para outras tarefas não diminui a efetividade de seu uso em conjunto com outras bibliotecas que abordam diferentes aspectos da análise de dados de mobilidade. A combinação dessas ferramentas pode oferecer uma abordagem completa e abrangente na visualização e exploração dos dados, proporcionando *insights* valiosos para a compreensão da mobilidade e a tomada de decisões informadas. Nesse sentido, com base na Tabela 3.12, a seguinte questão de pesquisa é respondida: **QP10) Existe suporte para visualização?**.

Table 3.12. Tipos de visualização disponível.

Bibliotecas	Trackintel	Yupi	Mobilipy	MobVis	PTRAIL	Mobilkit	Traja	MovingPandas	Mobipy	PyMove	Scikit-Mobility	Tracktable	SMAFramework	MOCHA	Teetool
Imagens	+	+	-	+	+	+	+	+	-	+	+	+	-	+	+
Gráficos	+	+	-	+	+	+	+	+	-	+	+	+	-	+	+
Mapas Interativos	-	-	+	+	+	-	-	+	-	+	+	-	+	-	-

É importante ressaltar que algumas bibliotecas delegam as funções de visualização para outras bibliotecas. Portanto, possibilitam o uso de todos os recursos disponíveis nestas bibliotecas de visualização, sendo necessário entendê-las para tirar total proveito de seus recursos. Isso adiciona uma camada de complexidade para o uso destes recursos.

3.4. Abordagem Prática para Tratamento e Análise de Mobilidade

Na abordagem prática, é apresentado um conjunto de dados reais, demonstrando a aplicação dos recursos já presentes nas bibliotecas comparadas. Essa aplicação deve abranger a execução de todas as tarefas relacionadas à necessidade de uso de cada biblioteca.

A metodologia proposta será aplicada da seguinte forma:

1. **Conjunto de dados**: A apresentação do conjunto de dados que será analisado revela que o *dataset* utilizado foi coletado em uma cidade urbana e contém informações de trajetórias registradas por dispositivos GPS em veículos de transporte público, táxis.

2. **Pré-processamento dos dados**: A realização do pré-processamento dos dados de trajetória será conduzida utilizando as bibliotecas *SciKit-Mobility* e *MovingPandas*. Isso incluirá a limpeza dos dados, a filtragem de ruídos, a detecção de pontos de parada, a segmentação de trajetória e a compressão.

3. **Análise exploratória**: A biblioteca *MovingPandas* será empregada para a realização de uma análise exploratória dos dados. Isso envolverá a visualização das

trajetórias, a identificação de padrões e a obtenção de *insights* iniciais sobre a mobilidade.

4. **Extração de características**: A biblioteca *PTRAIL* será utilizada para a extração de características relevantes dos dados de trajetória. Isso pode incluir informações como velocidade, distância percorrida, duração da viagem, entre outras.

5. **Predição de trajetória**: A modelagem dos dados de mobilidade será realizada por meio da biblioteca *Traja*. Isso pode envolver a aplicação de algoritmos de aprendizado de máquina ou técnicas de mineração de dados para a identificação de padrões e a predição dos dados de mobilidade.

Ao seguir essa metodologia, uma análise completa dos dados de mobilidade pode ser realizada, abrangendo desde o pré-processamento até a obtenção de conhecimentos e *insights* relevantes. Essa instância da metodologia é aplicada a um conjunto de dados real, possibilitando uma demonstração prática e aplicada dos conceitos discutidos nas seções anteriores.

3.4.1. Conjunto de dados

O uso do conjunto de dados *Taxi Service Trajectory*[2] é realizado para conduzir a análise. Este conjunto de dados descreve um ano completo das trajetórias de todos os 442 táxis que operam na cidade do Porto, em Portugal. Esses táxis operam por meio de uma central de despacho de táxis, utilizando terminais de dados móveis instalados nos veículos. As viagens são categorizadas em três categorias: (A) baseada na central de táxis, (B) baseada em ponto de táxi ou (C) não baseada na central de táxis. Para a primeira categoria, foi fornecido um ID anonimizado quando essa informação estava disponível por meio da ligação telefônica. As duas últimas categorias referem-se a serviços que foram solicitados diretamente aos motoristas de táxi em um (B) ponto de táxi ou em uma (C) rua aleatória. Este conjunto de dados oferece uma oportunidade valiosa para aplicar a metodologia discutida e obter *insights* significativos sobre a mobilidade na cidade do Porto.

3.4.2. Pré-processamento dos dados

LIMPEZA DOS DADOS

Antes de prosseguir para as etapas subsequentes, é fundamental realizar a limpeza e formatação dos dados. Para assegurar a integridade dos dados, foram excluídas todas as trajetórias em que a coluna 'MISSING_DATA' continha um valor diferente de '*False*'. Além disso, as colunas que continham valores ausentes ou apresentavam apenas um valor em todo o conjunto de dados também foram removidas. Após essa filtragem, uma amostra aleatória das trajetórias remanescentes foi selecionada, correspondendo a aproximadamente 0,05% do total. Dessa amostra, apenas as trajetórias com mais de 5 pontos foram mantidas. Por fim, houve uma modificação na representação das trajetórias: anteriormente, cada linha do conjunto de dados representava uma trajetória completa, mas agora cada linha corresponde a um ponto específico, compartilhando os mesmos valores para o restante da trajetória.

[2]https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i

FILTRAGEM DE RUÍDOS

A seguir, é apresentado um exemplo específico de uma trajetória que ilustra claramente a presença de ruídos ou valores discrepantes, nos quais dois pontos são repentinamente distanciados consideravelmente de seus vizinhos. Para lidar com esses casos, é necessário que a filtragem dos dados seja realizada, e esse problema é abordado por várias bibliotecas disponíveis. Nesta instância, a solução fornecida pela biblioteca *Scikit-Mobility* foi utilizada. Por meio dessa biblioteca, todos os pontos que excedem a velocidade máxima de 300 km/h foram removidos. O resultado, conforme ilustrado na Figura 3.4, é apresentado na comparação entre a trajetória original, representada pela linha tracejada vermelha, e a nova trajetória representada pela linha azul, na qual os pontos removidos são destacados em verde.

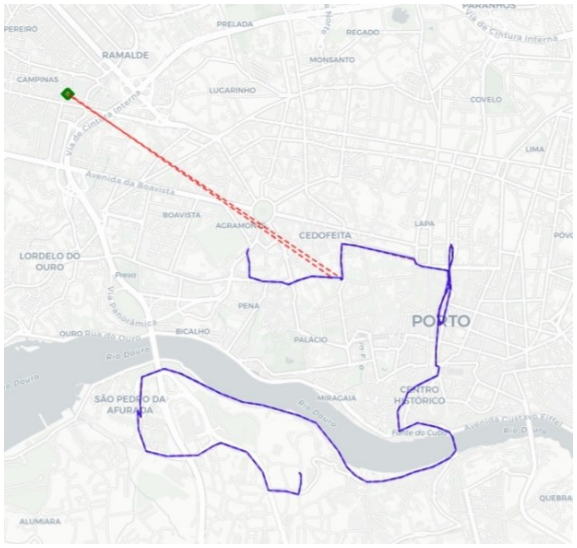


Figure 3.4. Exemplo de filtragem de ruídos

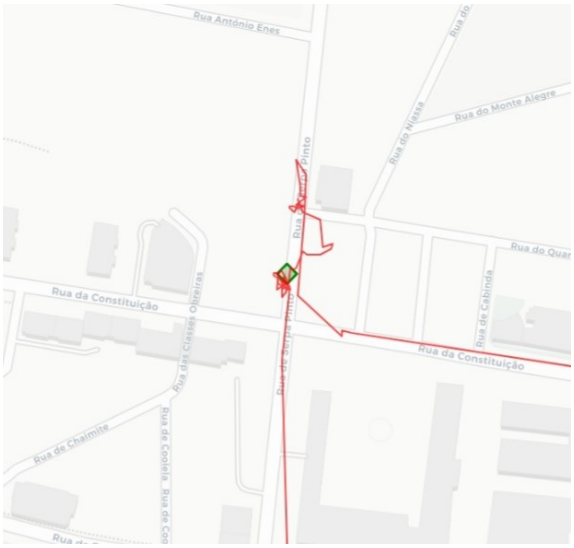


Figure 3.5. Exemplo de detecção de pontos de parada

DETECÇÃO DE PONTOS DE PARADA

Para ilustrar mais um exemplo, outra trajetória foi selecionada na qual um ponto de parada foi detectado, como pode ser visto na Figura 3.5. A detecção desse ponto de parada foi realizada por meio da funcionalidade fornecida pelo *Scikit-Mobility*. Nesse caso, os critérios utilizados para a classificação de um ponto como parada foram aqueles em que o objeto permaneceu por mais de 5 minutos a uma distância de 0.2 quilômetros. O ponto de parada detectado está destacado em verde na Figura 3.5.

COMPRESSÃO DE TRAJETÓRIA

Para exemplificar a compressão de uma trajetória, uma instância do *dataset* foi selecionada e o método de compressão fornecido pelo *Scikit-Mobility* foi aplicado. Esse método substitui conjuntos de pontos que estejam a uma distância de 0.2 quilômetros por um único ponto médio. A Figura 3.6 apresenta a trajetória original em azul, a nova trajetória comprimida em vermelho tracejado e destaca em verde os pontos que foram removidos durante o processo de compressão.

SEGMENTAÇÃO DE TRAJETÓRIA

Para realizar essa tarefa, os recursos da biblioteca *MovingPandas* são utilizados para seg-

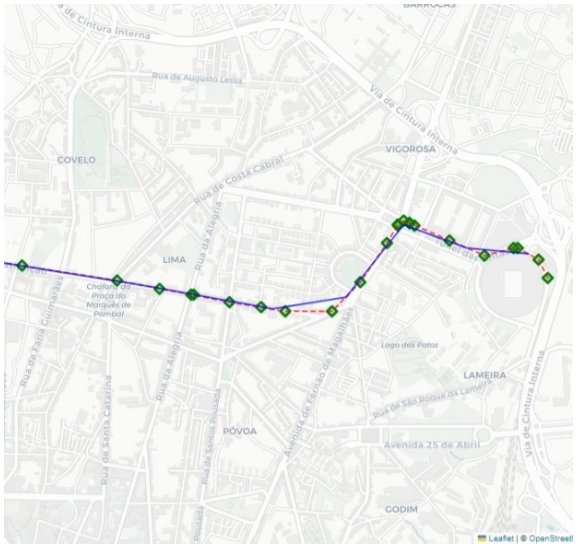


Figure 3.6. Exemplo de compressão de trajetória

Figure 3.7. Exemplo de segmentação de trajetória

mentar uma trajetória, exemplificando critérios que podem ser empregados para essa divisão. Na Figura 3.7, a trajetória original é dividida em três novos grupos de subtrajetórias. O primeiro grupo é criado com base em lacunas temporais, o segundo é formado pela detecção de pontos de parada, e o terceiro é resultado da análise das diferenças de velocidade.

3.4.3. Análise exploratória

Esta etapa busca aumentar o conhecimento sobre os dados disponíveis. Para exemplificá-la, serão respondidas algumas perguntas, listadas a seguir.

1. ***Qual foi a maior velocidade registrada?*** A maior velocidade registrada foi de 292.46 km/h, possivelmente originada de um *outlier* não filtrado.

2. ***Onde estão localizados os pontos mais significativos, os agrupamentos com maior densidade e qual é o fluxo entre eles?*** A Figura 3.11 torna visível que o centro da cidade tende a conter uma maior densidade de veículos.

3. ***Existe uma lacuna temporal nos dados de mobilidade?*** Os dados apresentam uma forma semelhante a uma distribuição sinusoidal ao visualizar a quantidade de pontos para cada dia, começando em '2013-07-01 05:55:41' e terminando em '2014-06-30 09:22:04'.

4. ***Qual a distribuição de viajem em cada mês e dia da semana?***

   Nas Figuras 3.8 e 3.9, é perceptível que há uma maior quantidade de viagens nas segundas e sextas-feiras. De forma semelhando, é notável o aumento de viagens nos meses próximos a junho e outubro.

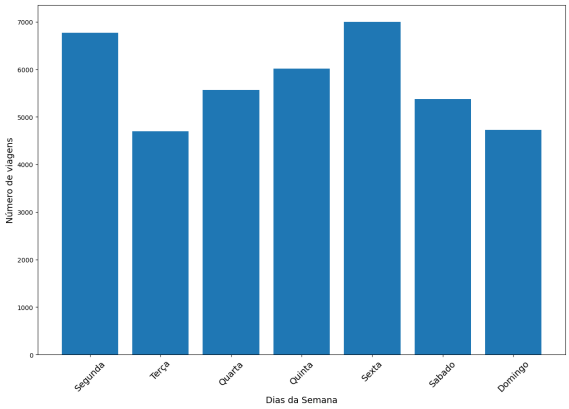


Figure 3.8. Número de viagens em cada dia da semana.

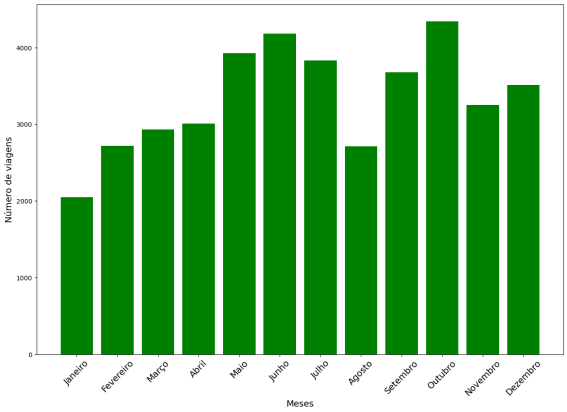


Figure 3.9. Número de viagens em cada mês.

3.4.4. Extração de características

Nesta fase, métodos de extração de características presentes na biblioteca *PTRAIL* foram empregados para enriquecer o conjunto de dados, adicionando informações temporais, como dia da semana e hora do dia, além de informações derivadas dos movimentos, como distância, velocidade e aceleração.

Por fim, modelos disponíveis na biblioteca *scikit-learn*[3] foram utilizados para exemplificar o impacto que a extração de características pode ter em tarefas relacionadas à análise de mobilidade, como a melhoria na acurácia da classificação dos tipos de chamadas (call_type). Essa abordagem demonstra como a utilização adequada das características extraídas pode contribuir significativamente para o aprimoramento dos resultados em aplicações de análise de mobilidade.

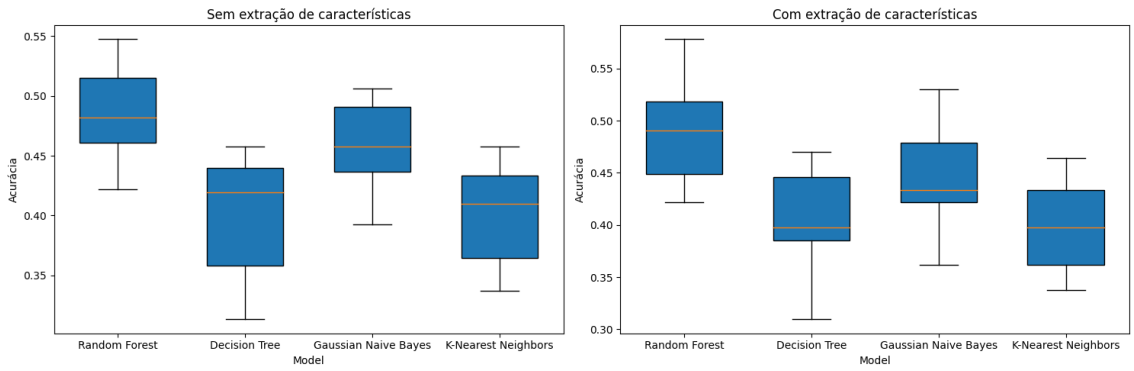


Figure 3.10. Impacto da extração de caraterísticas, na classificação de trajetórias.

3.4.5. Predição de trajetória

Com base na biblioteca Python *Traja*, realizou-se um experimento para prever os três próximos pontos de uma trajetória. Para isso, utilizamos o modelo *Long Short-Term Memory* (LSTM) disponibilizado pela biblioteca, o qual foi treinado e validado em sub-

[3]https://scikit-learn.org/stable/index.html

conjuntos distintos da amostra de dados. Após o processo de treinamento e validação, a predição com o menor Erro Quadrático Médio foi selecionada. Os resultados obtidos foram apresentados na Figura 3.12, demonstrando a acurácia da predição.

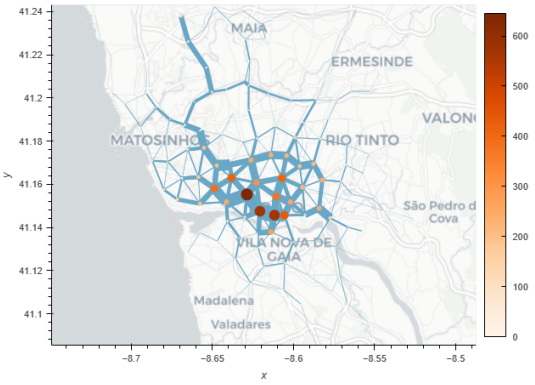


Figure 3.11. Número de pontos de densidade e fluxos.

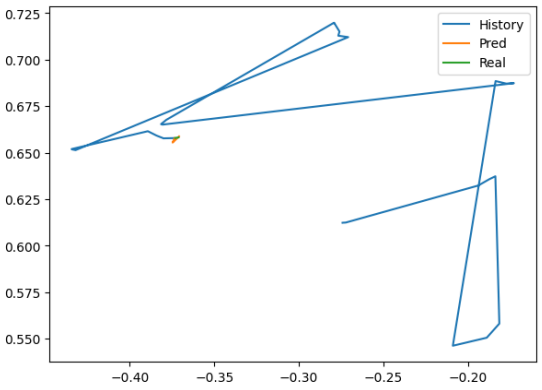


Figure 3.12. Predição de trajetória.

3.5. Conclusão

O objetivo principal deste estudo foi realizar uma comparação quantitativa entre bibliotecas Python que disponibilizam recursos para a análise de mobilidade. Posteriormente, as bibliotecas foram classificadas e associadas a fases específicas da aquisição de conhecimento de dados de trajetórias, seguindo um paradigma similar ao apresentado por [44].

A combinação dessas fases em uma metodologia ou *pipeline*, juntamente com os resultados obtidos na comparação, foi exemplificada por meio de um caso de uso. Esse caso de uso ilustrou resultados extraídos de uma base de dados real e disponibilizou o código-fonte[4] utilizado, demonstrando a relevância desses métodos na aquisição de conhecimento em dados de mobilidade.

Os *insights* gerados pela comparação das bibliotecas revelaram várias informações importantes. Foi identificada a quase ausência de suporte para tarefas de *map matching*, poucos métodos de geração de trajetória baseados em modelagem de conjuntos de dados e a inexistência de bibliotecas que permitam a mineração de movimentação conjunta (Moving Together). Além disso, observou-se a falta de suporte para a classificação de trajetórias e detecção de eventos atípicos dentro das trajetórias. Também foi notada uma escassez de soluções para análise de privacidade, predição e detecção de trajetórias atípicas.

Em resumo, a comparação, a metodologia e o caso de uso apresentados neste artigo proporcionam auxílio na tomada de decisões relacionadas ao uso dos recursos disponíveis nas bibliotecas Python para análise de mobilidade. Além disso, as lacunas identificadas revelam oportunidades para o avanço da pesquisa nessa área, incentivando o desenvolvimento de métodos mais abrangentes e aprimorados.

[4]https://github.com/edgarsoliveira1/MINICURSO-ERCEMAPI2023

References

[1] Osman Abul, Francesco Bonchi, and Mirco Nanni. “Never walk alone: Uncertainty for anonymity in moving objects databases”. In: *2008 IEEE 24th international conference on data engineering*. Ieee. 2008, pp. 376–385.

[2] Alastair R Beresford and Frank Stajano. “Location privacy in pervasive computing”. In: *IEEE Pervasive computing* 2.1 (2003), pp. 46–55.

[3] Maurício Cavalcante Bráz. “Implementação de algoritmos para análise de similaridade de trajetória na biblioteca PyMove”. In: *Monografia. Universidade Federal do Ceará* (2020).

[4] Clayson Celes, Azzedine Boukerche, and Antonio AF Loureiro. “From mobility traces to knowledge: Design guidance for intelligent vehicular networks”. In: *IEEE Network* 34.4 (2020), pp. 227–233.

[5] Clayson Celes et al. “Improving vanet simulation with calibrated vehicular mobility traces”. In: *IEEE Transactions on Mobile Computing* 16.12 (2017), pp. 3376–3389.

[6] Chi-Yin Chow, Mohamed F Mokbel, and Walid G Aref. “Casper* Query processing for location services without compromising privacy”. In: *ACM Transactions on Database Systems (TODS)* 34.4 (2009), pp. 1–48.

[7] Deyu Deng et al. “Spatial-temporal data science of COVID-19 data”. In: *2021 IEEE 15th International Conference on Big Data Science and Engineering (BigDataSE)*. IEEE. 2021, pp. 7–14.

[8] Willem Eerland et al. “Teetool–a probabilistic trajectory analysis tool”. In: *Journal of Open Research Software* 5.1 (2017).

[9] Gyozo Gidofalvi, Xuegang Huang, and Torben Bach Pedersen. “Privacy-preserving data mining on moving object trajectories”. In: *2007 International Conference on Mobile Data Management*. IEEE. 2007, pp. 60–68.

[10] Anita Graser and Melitta Dragaschnig. “Exploring movement data in notebook environments”. In: *IEEE VIS 2020 - MoVis*. 2020.

[11] Joachim Gudmundsson and Marc Van Kreveld. “Computing longest duration flocks in trajectory data”. In: *Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems*. 2006, pp. 35–42.

[12] Salman Haidri et al. “PTRAIL—A python package for parallel trajectory data preprocessing”. In: *SoftwareX* 19 (2022), p. 101176.

[13] Charles R. Harris et al. “Array programming with NumPy”. In: *Nature* 585 (2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2.

[14] Hoyoung Jeung, Heng Tao Shen, and Xiaofang Zhou. “Convoy queries in spatio-temporal databases”. In: *2008 IEEE 24th International Conference on Data Engineering*. IEEE. 2008, pp. 1457–1459.

[15] Hoyoung Jeung et al. “Discovery of convoys in trajectory databases”. In: *arXiv preprint arXiv:1002.0963* (2010).

[16] Kelsey Jordahl et al. “geopandas/geopandas: v0. 6.0”. In: *Zenodo* (2019).

[17] Ahmed Kharrat et al. “Clustering algorithm for network constraint trajectories”. In: *Headway in Spatial Data Handling*. Springer, 2008, pp. 631–647.

[18] Zhenhui Li et al. *Swarm: Mining relaxed temporal moving object clusters*. Tech. rep. ILLINOIS UNIV AT URBANA-CHAMPAIGN DEPT OF COMPUTER SCIENCE, 2010.

[19] Pedro HC Maia, Claudio EC Campelo, and Campina Grande–PB–Brazil. “Mobipy-A Python Library for Analyzing Mobility Patterns.” In: *GeoInfo*. 2020, pp. 141–152.

[20] Henry Martin et al. “Trackintel: An open-source Python library for human mobility analysis”. In: *Computers, Environment and Urban Systems* 101 (2023), p. 101938. DOI: 10.1016/j.compenvurbsys.2023.101938.

[21] Rui Moreno and Carina Oliveira. “Avaliação de Desempenho de Redes em Malha Sem Fio em Cenários de Cidades Inteligentes”. In: *Anais da X Escola Regional de Computação do Ceará, Maranhão e Piauí*. São Luís: SBC, 2022, pp. 109–118. DOI: 10.5753/ercemapi.2022.226045. URL: https://sol.sbc.org.br/index.php/ercemapi/article/view/21965.

[22] Brendan Morris and Mohan Trivedi. “Learning trajectory patterns by clustering: Experimental studies and comparative evaluation”. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2009, pp. 312–319.

[23] Mehmet Ercan Nergiz, Maurizio Atzori, and Yucel Saygin. “Towards trajectory anonymization: a generalization-based approach”. In: *Proceedings of the SIGSPATIAL ACM GIS 2008 International Workshop on Security and Privacy in GIS and LBS*. 2008, pp. 52–61.

[24] Luca Pappalardo et al. “scikit-mobility: A Python Library for the Analysis, Generation, and Risk Assessment of Mobility Data”. In: *Journal of Statistical Software* 103.4 (2022), pp. 1–38. DOI: 10.18637/jss.v103.i04. URL: https://www.jstatsoft.org/index.php/jss/article/view/v103i04.

[25] Michal Pleskowicz. *Mobilipy-from raw GPS data to mobility data*. Tech. rep. 2022.

[26] Jeff Reback et al. “pandas-dev/pandas: Pandas 1.0. 5”. In: *Zenodo* (2020).

[27] A Reyes et al. “yupi: Generation, Tracking and Analysis of Trajectory data in Python”. In: *Environmental Modelling & Software* 163 (2023), p. 105679.

[28] Mark Daniel Rintoul. *ML for Trajectories: Bridging the Gap from Computer to Analyst with Tracktable.* Tech. rep. Sandia National Lab.(SNL-NM), Albuquerque, NM (USA), 2020.

[29] Matthew Rocklin et al. “Dask: Parallel computation with blocked algorithms and task scheduling”. In: *Proceedings of the 14th python in science conference*. Vol. 130. SciPy Austin, TX. 2015, p. 136.

[30] Diego O Rodrigues and Leandro Villas. “SMAFramework: Arcabouço para Integração de Dados Urbanos Cientes da Correlação Espaço-Temporal”. In: *Anais Estendidos do XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. SBC. 2019, pp. 113–120.

[31] AL Sayeth Saabith, MMM Fareez, and T Vinothraj. “Python current trend applications-an overview”. In: *International Journal of Advance Engineering and Research Development* 6.10 (2019).

[32] Justin Shenk et al. “Traja: A Python toolbox for animal trajectory analysis”. In: *Journal of Open Source Software* 6.63 (2021), p. 3202.

[33] Lucas N Silva et al. “MobVis: A Framework for Analysis and Visualization of Mobility Traces”. In: *2022 IEEE Symposium on Computers and Communications (ISCC)*. IEEE. 2022, pp. 1–6.

[34] Fabrício R de Souza et al. “Mocha: A tool for mobility characterization”. In: *Proceedings of the 21st ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. 2018, pp. 281–288.

[35] Jean-Luc R Stevens, Philipp Rudiger, and James A Bednar. “HoloViews: Building complex visualizations easily for reproducible science”. In: *Proceedings of the 14th Python in Science Conference*. SciPy. 2015, pp. 61–69.

[36] Lu-An Tang et al. “A framework of traveling companion discovery on trajectory data streams”. In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 5.1 (2014), pp. 1–34.

[37] ZhengYu TangLu’an et al. “Discover of Traveling COmpanions from Streaming Trajectoris”. In: *ICDE, WashingtonDC, USA* (2012).

[38] Manolis Terrovitis and Nikos Mamoulis. “Privacy preservation in the publication of trajectories”. In: *The Ninth international conference on mobile data management (mdm 2008)*. IEEE. 2008, pp. 65–72.

[39] Enrico Ubaldi et al. “Mobilkit: A Python Toolkit for Urban Resilience and Disaster Risk Management Analytics using High Frequency Human Mobility Data”. In: *arXiv preprint arXiv:2107.14297* (2021).

[40] Guido Van Rossum, Barry Warsaw, and Nick Coghlan. “PEP 8–style guide for python code”. In: *Python. org* 1565 (2001), p. 28.

[41] Haitao Yuan and Guoliang Li. “A survey of traffic prediction: from spatio-temporal data to intelligent transportation”. In: *Data Science and Engineering* 6 (2021), pp. 63–85.

[42] Kai Zheng et al. “On discovery of gathering patterns from trajectories”. In: *2013 IEEE 29th international conference on data engineering (ICDE)*. IEEE. 2013, pp. 242–253.

[43] Kai Zheng et al. “Online discovery of gathering patterns over trajectories”. In: *IEEE Transactions on Knowledge and Data Engineering* 26.8 (2013), pp. 1974–1988.

[44] Yu Zheng. “Trajectory data mining: an overview”. In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 6.3 (2015), pp. 1–41.