UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Laylla Royer Pereira

# Exploração de sistemas de gerenciamento de banco de dados vetoriais para dados financeiros

**Uberlândia, Brasil**

**2025**

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Laylla Royer Pereira

# Exploração de sistemas de gerenciamento de banco de dados vetoriais para dados financeiros

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Ciência da Computação.

Orientador: Maria Camila Nardini Barioni

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Ciência da Computação

Uberlândia, Brasil

2025

**UNIVERSIDADE FEDERAL DE UBERLÂNDIA**
Faculdade de Computação
Av. João Naves de Ávila, nº 2121, Bloco 1A - Bairro Santa Mônica, Uberlândia-MG, CEP 38400-902
Telefone: (34) 3239-4144 - http://www.portal.facom.ufu.br/ facom@ufu.br

# ATA DE DEFESA - GRADUAÇÃO

| | | | | | |
|---|---|---|---|---|---|
| Curso de Graduação em: | Bacharelado em Ciência da Computação | | | | |
| Defesa de: | Projeto de Graduação 2 - GBC082 | | | | |
| Data: | 09/10/2025 | Hora de início: | 08:00 | Hora de encerramento: | 9:10 |
| Matrícula do Discente: | 12121BCC026 | | | | |
| Nome do Discente: | Laylla Royer Pereira | | | | |
| Título do Trabalho: | Exploração de sistemas de gerenciamento de banco de dados vetoriais para dados financeiros | | | | |
| A carga horária curricular foi cumprida integralmente? | ( x ) Sim ( ) Não | | | | |

Reuniu-se de forma online pela plataforma MS Teams, a Banca Examinadora, designada pelo Colegiado do Curso de Graduação em Bacharelado em Ciência da Computação, assim composta: Professores: Elaine Ribeiro de Faria Paiva FACOM/UFU, Leandro Nogueira Couto FACOM/UFU e Maria Camila Nardini Barioni FACOM/UFU, orientadora da candidata.

Iniciando os trabalhos, a presidente da mesa, Dra. Maria Camila Nardini Barioni, apresentou a Comissão Examinadora e a candidata, agradeceu a presença do público, e concedeu ao discente a palavra, para a exposição do seu trabalho. A duração da apresentação da discente e o tempo de arguição e resposta foram conforme as normas do curso.

A seguir a senhora presidente concedeu a palavra, pela ordem sucessivamente, aos examinadores, que passaram a arguir a candidata. Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando a candidata:

(X) Aprovada Nota [90]

OU

( ) Aprovada sem nota.

Nada mais havendo a tratar foram encerrados os trabalhos. Foi lavrada a presente ata que após lida e achada conforme foi assinada pela Banca Examinadora.

Documento assinado eletronicamente por **Elaine Ribeiro de Faria Paiva**, **Professor(a) do Magistério Superior**, em 09/10/2025, às 09:10, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Leandro Nogueira Couto**, **Professor(a) do Magistério Superior**, em 09/10/2025, às 09:10, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Maria Camila Nardini Barioni**, **Professor(a) do Magistério Superior**, em 09/10/2025, às 09:26, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://www.sei.ufu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **6749776** e o código CRC **E375D1F0**.

# Agradecimentos

Agradeço primeiramente a Deus pela força e sabedoria em toda esta jornada. Aos meus professores, que contribuíram para minha formação, em especial à minha orientadora Prof.ª Dr.ª Maria Camila Nardini Barioni, pela paciência, dedicação e incentivo essenciais para a realização deste trabalho. À minha família, pelo apoio incondicional, amor e compreensão em todos os momentos, fundamentais para que eu chegasse até aqui. E aos meus amigos da faculdade, que tornaram essa caminhada mais leve e especial, em especial Ana Luísa Matias Corsi, pela amizade e companheirismo ao longo do curso.

# Resumo

Este trabalho teve como objetivo principal demonstrar a viabilidade de realizar o processo de clusterização e busca por similaridade de dados financeiros diretamente em um Sistema Gerenciador de Banco de Dados (SGBD) relacional, expandindo o método de trabalho de um estudo anterior. O método de trabalho incluiu a coleta automatizada de dados de ações da NYSE (*New York Stock Exchange*), seguida por um pré-processamento para tratar valores ausentes, remover duplicatas e normalizar os atributos. Para a análise, os dados foram convertidos em vetores e armazenados em um SGBD PostgreSQL, utilizando a extensão `pgvector` para manipulação nativa desses vetores e testes de busca por similaridade. Além disso, o algoritmo K-Means foi implementado como uma *stored procedure* no próprio SGBD. Os resultados da pesquisa demonstram que é possível executar eficientemente a clusterização e as buscas por similaridade diretamente no SGBD, eliminando a necessidade de exportar grandes volumes de dados para ambientes externos e aproveitando o desempenho otimizado da extensão `pgvector`. Conclui-se que a abordagem adotada no trabalho descrito aqui oferece uma solução robusta e escalável para a análise de grandes conjuntos de dados financeiros.

**Palavras-chave**: Clusterização; K-Means; PostgreSQL; pgvector; Análise Financeira.

# Abstract

This work aimed to demonstrate the feasibility of performing clustering and similarity search of financial data directly within a relational Database Management System (DBMS), expanding upon the working method of a previous study. The working method included the automated collection of NYSE (New York Stock Exchange) stock data, followed by preprocessing to handle missing values, remove duplicates, and normalize attributes. For the analysis, the data were converted into high-dimensional vectors and stored in a PostgreSQL DBMS, using the `pgvector` extension for native vector manipulation and similarity search tests. In addition, the K-Means algorithm was implemented as a stored procedure within the DBMS itself. The research results demonstrate that it is possible to efficiently perform clustering and similarity searches directly in the DBMS, eliminating the need to export large volumes of data to external environments and leveraging the optimized performance of the `pgvector` extension. It is concluded that the approach adopted in this thesis provides a robust and scalable solution for the analysis of large financial datasets.

**Keywords**: Clustering; K-Means; PostgreSQL; pgvector; Financial Analysis.

# Lista de ilustrações

# Lista de tabelas

# Lista de abreviaturas e siglas

| | |
|---|---|
| SGBD | Sistema Gerenciador de Banco de Dados |
| SGBDV | Sistema Gerenciador de Banco de Dados Vetorial |
| SGBDR | Sistema Gerenciador de Banco de Dados Relacional |
| NYSE | *New York Stock Exchange* |

# Sumário

# 1 Introdução

A Bolsa de Valores é um mercado organizado onde se negociam ações de sociedades de capital aberto e outros valores mobiliários, proporcionando um ambiente seguro e transparente para essas transações (MANZONI, 2025). O Mercado Financeiro engloba as operações de compra e venda de ativos financeiros, como valores mobiliários, mercadorias e câmbio, funcionando como um ambiente onde ocorrem as operações de investimentos financeiros (ASSAF, 2019). Dentro deste contexto, o Mercado de Ações é uma subdivisão do mercado financeiro, sendo um ambiente público e organizado para negociação de ações, permitindo que investidores adquiram participações em empresas e contribuam para a capitalização das mesmas (ASSAF, 2019). Devido ao grande volume de transações, há uma significativa geração de dados sobre os ativos disponíveis, que podem ser usados em análises do mercado financeiro.

A *New York Stock Exchange* (NYSE) é a principal bolsa de valores dos Estados Unidos e uma das mais importantes do mundo. Fundada em 1792, a partir do chamado *Buttonwood Agreement*, consolidou-se como um centro global para negociação de ações e títulos, desempenhando papel fundamental no desenvolvimento do mercado financeiro mundial (New York Stock Exchange, 2025). Atualmente, a NYSE é referência internacional em liquidez, volume de negociações e capitalização de mercado. No Brasil, a principal bolsa de valores é a B3, onde são negociadas ações das maiores companhias brasileiras, como Ambev, Banco Itaú e Banco Bradesco (MANZONI, 2025). Assim como a NYSE, a B3 exerce um papel estratégico na economia, possibilitando a captação de recursos pelas empresas e o acesso a investimentos diversificados pelos investidores.

É possível obter uma grande quantidade de informações sobre ações e outros ativos financeiros por meio do serviço *Yahoo Finance* (Yahoo Finance, 2025), que disponibiliza cotações, séries históricas de preços, indicadores financeiros e notícias relacionadas ao mercado. Para facilitar o acesso a esses dados, existe a biblioteca `yfinance` em Python (AROUSSI, 2025), que fornece uma interface prática para extração automatizada de informações diretamente da plataforma.

Com a utilização dessa biblioteca, torna-se possível coletar dados detalhados sobre ações, como preços de abertura, fechamento, máximos, mínimos, volume negociado e outros indicadores de mercado. Esses dados podem ser aplicados em análises quantitativas, como a clusterização de ativos, permitindo identificar padrões de comportamento entre diferentes companhias. Além disso, uma abordagem complementar envolve a utilização de um Sistema de Gerenciamento de Banco de Dados (SGBD), como o PostgreSQL com a extensão `pgvector` (PostgreSQL Global Development Group, 2025), onde os dados

podem ser transformados em vetores e posteriormente agrupados por meio de algoritmos de clusterização para avaliar e analisar as ações.

## 1.1 Justificativa

Recentemente, o interesse dos brasileiros pelo mercado financeiro aumentou de forma significativa. Entretanto, muitos investidores iniciantes ainda não possuem o conhecimento e a segurança necessários para desenvolver suas próprias aplicações e estratégias de investimento (CAMPISTA, 2020). Além disso, embora existam diversos dados brutos disponíveis em plataformas como a B3 e órgãos governamentais, poucos já passaram por processos de pré-processamento e encontram-se prontos para uso em aplicações práticas.

No campo acadêmico, diferentes pesquisas têm buscado contribuir para a área. Por exemplo, (SANTOS; COSTA, 2023a) utiliza o modelo Random Forest para identificar os melhores momentos de compra e venda de ativos no mercado brasileiro, enquanto (MOURA, 2021) propõe um sistema de análise em tempo real do desempenho de fundos de investimento, sugerindo opções para investidores com pouco conhecimento financeiro. Há ainda estudos, como o de (FIGUEIREDO, 2024), que demonstram a viabilidade da clusterização de ações como ferramenta de apoio à decisão. Tais iniciativas reforçam a relevância de novas abordagens que integrem técnicas de análise de dados e soluções tecnológicas aplicadas ao mercado de capitais.

## 1.2 Objetivos

Diante desse cenário, da ampla disponibilidade de dados sobre ações e dos desafios de armazenamento e processamento desses dados, este trabalho tem como objetivo principal avaliar o uso do SGBD PostgreSQL com a extensão `pgvector` para análises baseadas em representações vetoriais. Para a coleta das informações, será utilizada a biblioteca `yfinance`, que permite a extração de dados financeiros da NYSE de maneira automatizada. A proposta central consiste em demonstrar a viabilidade de executar operações de clusterização e busca por similaridade diretamente em um SGBD relacional, eliminando a necessidade de ferramentas externas e mantendo todo o fluxo analítico dentro de um único ambiente. Dessa forma, busca-se integrar todas as etapas da análise — desde a coleta e o armazenamento dos dados, passando pelo processamento vetorial, até a aplicação dos algoritmos de aprendizado não supervisionado. Essa integração visa garantir eficiência, consistência, escalabilidade e reprodutibilidade dos resultados, reforçando o papel do Sistema Gerenciador de Banco de Dados (SGBD) como plataforma unificada para o armazenamento e a análise de dados financeiros.

## 1.3 Organização do Trabalho

Este trabalho está estruturado em seis capítulos. O Capítulo 1 apresenta a introdução, contemplando a justificativa, os objetivos e a organização do texto. O Capítulo 2 traz a revisão bibliográfica, abordando os conceitos de bancos de dados vetoriais, as formas de representação dos dados, a biblioteca `yfinance`, a importância da normalização e o funcionamento do algoritmo *K-Means*. No Capítulo 3 são apresentados os trabalhos relacionados, que servem de base teórica para a pesquisa. O Capítulo 4 descreve o método de trabalho, desde a obtenção e o pré-processamento da base de dados até a criação do banco no PostgreSQL, a implementação da busca por similaridade e a aplicação do algoritmo de clusterização. O Capítulo 5 reúne os resultados obtidos, incluindo a análise da base final, as consultas realizadas e a clusterização das ações. Por fim, o Capítulo 6 apresenta as conclusões, destacando a avaliação dos resultados, os pontos positivos, as limitações e as perspectivas para trabalhos futuros.

# 2 Revisão Bibliográfica

A análise de dados financeiros e não financeiros exige uma compreensão abrangente sobre a representação, processamento e interpretação de informações em diferentes formatos. A literatura evidencia que a estruturação adequada dos dados, seja em tabelas ou vetores, é essencial para a aplicação eficiente de técnicas de mineração e aprendizado de máquina, especialmente em cenários que envolvem grandes volumes de informações heterogêneas. Este capítulo revisa conceitos fundamentais relacionados à representação de dados estruturados e não estruturados, métodos de pré-processamento e normalização, estratégias de coleta e armazenamento, além de abordagens de aprendizado de máquina não supervisionado, como a clusterização, que permitem identificar padrões e relações significativas em conjuntos de dados complexos. A revisão enfatiza, ainda, a aplicação de representações vetoriais e SGBD vetoriais, que viabilizam consultas por similaridade e análises mais robustas em contextos de dados financeiros e de alta dimensionalidade.

## 2.1 Representação dos dados

A representação de dados é um conceito fundamental na estatística e na ciência de dados, responsável por organizar e estruturar informações de modo a facilitar a compreensão e a análise. Esse processo pode ocorrer por meio de tabelas, gráficos, vetores e outras formas visuais ou estruturadas, dependendo do contexto e do objetivo da análise. A forma escolhida para representar os dados influencia diretamente na eficiência de técnicas de análise, na interpretação dos resultados e na aplicação de algoritmos de mineração de dados.

### 2.1.1 Dados estruturados vs. dados não estruturados

Os dados estruturados são aqueles organizados em formatos predefinidos, geralmente tabelas com linhas e colunas, em que cada coluna representa um atributo específico e cada linha corresponde a uma instância de dado. Exemplos típicos incluem registros financeiros, transações bancárias e preços históricos de ações, que podem ser consultados e manipulados de forma eficiente por meio de SGBD relacionais e consultas SQL (Amazon Web Services, 2025).

Em contrapartida, os dados não estruturados não seguem um esquema fixo de atributos e podem assumir formas diversas, como textos livres, relatórios financeiros, imagens, áudios e vídeos (Elastic, 2025). Esses dados representam a maior parte da informação gerada atualmente e são mais desafiadores de processar devido à ausência de estrutura

padronizada. O processamento de dados não estruturados exige técnicas avançadas, como mineração de texto, processamento de linguagem natural e aprendizado de máquina, para extrair informações relevantes(IBM, 2025).

Entre os principais desafios do uso de dados não estruturados, destacam-se:

- **Volume e diversidade**: grandes volumes de dados variados dificultam o armazenamento e a análise eficiente (Elastic, 2025).
- **Qualidade e consistência**: dados não estruturados podem conter informações inconsistentes, incompletas ou redundantes, exigindo pré-processamento cuidadoso (IBM, 2025).
- **Complexidade de análise**: técnicas tradicionais de análise de dados estruturados não se aplicam diretamente; é necessário transformar esses dados em representações adequadas, como vetores(PAN; WANG; LI, 2024).
- **Armazenamento e governança**: gerenciar grandes volumes de dados não estruturados demanda soluções escaláveis e políticas de governança eficazes para assegurar segurança e conformidade regulatória (IBM, 2025).

No contexto de dados não estruturados, a recuperação de informações é frequentemente realizada por meio de pesquisa por similaridade, que identifica quais vetores devem ser retornados com base no grau de semelhança entre os vetores armazenados e o vetor de consulta, ao invés de consultas estruturadas tradicionais (PAN; WANG; LI, 2024). Essa abordagem é essencial para a análise de dados financeiros transformados em vetores, permitindo encontrar padrões e relações entre ativos de maneira eficiente.

### 2.1.2 Representação vetorial dos dados

Para lidar com dados não estruturados, uma abordagem eficiente é transformá-los em representações vetoriais. Um vetor de característica, é a representação numérica de um objeto, como um texto, uma imagem ou um vídeo (TAIPALUS, 2024). Essa transformação de dados, conhecida como vetorização, captura as características e os padrões significativos de um objeto em um vetor numérico de $n$-dimensões. Por exemplo, a localização de um ponto em um mapa pode ser representada por um vetor bidimensional. Em casos mais complexos, um vetor pode ter milhares ou milhões de dimensões para representar dados mais ricos, como as características de um texto ou uma imagem, tornando a visualização desses dados inviável para humanos (TAIPALUS, 2024).

Essa capacidade de representar informações complexas de forma vetorial permite diversas aplicações práticas. No contexto da geolocalização, a vetorização representa pontos espaciais, como a localização do usuário e de pontos de interesse, em um SGBD,

utilizando coordenadas bidimensionais (latitude e longitude) ou tridimensionais (com altitude) (TAIPALUS, 2024). A proximidade entre esses pontos pode ser determinada por métricas como distância euclidiana ou similaridade do cosseno, permitindo identificar locais mais próximos de forma eficiente. Por exemplo, ao armazenar em um banco vetorial as posições de um usuário $(6, 7)$, um restaurante $(3, 8)$ e um supermercado $(7, 1)$, é possível calcular a menor distância e fornecer recomendações personalizadas.

De forma análoga, no contexto de dados financeiros, a vetorização permite modelar relações complexas entre entidades, como variações de preços, impactos de eventos no mercado e padrões de comportamento de ativos. Nesse processo, os vetores representam características como o preço de fechamento de uma ação ao longo dos dias e o volume de negociação, facilitando a identificação de tendências e correlações. Ao estruturar esses dados em formato vetorial, torna-se possível aplicar algoritmos avançados de busca e análise para detectar padrões e embasar decisões estratégicas com base em similaridade e comportamento histórico. Essa abordagem diferencia os SGBD vetoriais dos tradicionais, pois possibilita o processamento eficiente e escalável de informações não estruturadas (TAIPALUS, 2024).

Em bancos de dados vetoriais, a similaridade entre vetores é avaliada por meio de uma pontuação de similaridade ou de uma função de distância. Para algumas pontuações, valores maiores indicam maior similaridade, enquanto para outras, valores mais próximos de zero indicam maior similaridade. Essa abordagem permite realizar consultas eficientes e precisas, possibilitando identificar rapidamente vetores que compartilham características próximas no espaço multidimensional (PAN; WANG; LI, 2024). A seguir, serão apresentados os principais funções de similaridade discutidos no trabalho de (PAN; WANG; LI, 2024).

- **Hamming**
  A distância de Hamming mede o número de posições em que dois vetores diferem. Sua definição é dada pela Equação 2.1:

$$d(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^{n} \left(1 - \delta_{a_i b_i}\right) \tag{2.1}$$

  onde $\delta_{a_i b_i}$ é o delta de Kronecker, definido na Equação 2.2:

$$\delta_{ij} = \begin{cases} 1, & \text{se } i = j \\ 0, & \text{se } i \neq j \end{cases} \tag{2.2}$$

  Assim, $d(\mathbf{a}, \mathbf{b})$ retorna quantas dimensões diferem entre os vetores $\mathbf{a}$ e $\mathbf{b}$.

- **Produto Interno**
  O produto interno combina dois vetores para produzir um número escalar, conforme

a Equação 2.3 ou, equivalentemente, a Equação 2.4:

$$f(a,b) = \sum_{i=1}^{n} a_i b_i \tag{2.3}$$

$$f(a,b) = \langle a,b \rangle = a \cdot b \tag{2.4}$$

- **Similaridade Cosseno**
  A similaridade cosseno mede a orientação relativa entre dois vetores, como na Equação 2.5, sendo equivalente à normalização do produto interno na Equação 2.6:

$$f(a,b) = \langle \hat{a}, \hat{b} \rangle \tag{2.5}$$

$$f(a,b) = \frac{\langle a,b \rangle}{\|a\| \, \|b\|} \tag{2.6}$$

- **Distância de Minkowski**
  A distância de Minkowski de ordem $p$ é dada pela Equação 2.7:

$$d(a,b) = \left( \sum_{i=1}^{n} |a_i - b_i|^p \right)^{1/p} \tag{2.7}$$

- **Distância Euclidiana**
  A distância Euclidiana é um caso particular da distância de Minkowski quando $p = 2$. Ela mede a distância em linha reta entre dois vetores em um espaço de $n$ dimensões. Valores mais próximos de zero indicam maior similaridade. A fórmula é dada na Equação 2.8:

$$d(a,b) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2} \tag{2.8}$$

A aplicação dessas métricas depende diretamente da qualidade das representações vetoriais geradas. Entretanto, para que esses vetores reflitam de maneira confiável as características dos dados originais, é necessário que os valores passem por um processo de preparação adequado. Nesse sentido, o pré-processamento e a normalização dos dados desempenham um papel essencial, garantindo consistência, comparabilidade e robustez na análise. Essa etapa será discutida na próxima subseção.

### 2.1.3 Pré-processamento e normalização

Em métodos que dependem de métricas de distância (como o *K-Means*), a normalização dos dados é uma etapa crítica: sem ela, atributos com escalas maiores podem dominar o cálculo de distância e enviesar os agrupamentos. A normalização consiste em transformar os valores dos atributos para uma escala comum (por exemplo, $[0, 1]$), preservando a estrutura relacional entre os dados, mas evitando que variáveis com amplitude maior tenham influência desproporcional (ZHANG; JIN; ZHOU, 2016).

No `scikit-learn`, uma das ferramentas mais utilizadas para normalização é o `MinMaxScaler` (MINMAXSCALER..., ; PREPROCESSING..., ). Esse transformador ajusta cada característica individualmente a um intervalo definido, que por padrão é $[0, 1]$. Internamente, o `MinMaxScaler` calcula primeiro a versão padronizada, conforme a Equação 2.9:

$$X_{\text{std}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \tag{2.9}$$

Em seguida, aplica o redimensionamento para o intervalo desejado, segundo a Equação 2.10:

$$X_{\text{scaled}} = X_{\text{std}} \cdot (\max - \min) + \min \tag{2.10}$$

onde min e max são os limites do intervalo de destino, usualmente 0 e 1. Dessa forma, o valor mínimo original é mapeado para 0 e o máximo para 1, enquanto valores intermediários são interpolados linearmente.

Esse tipo de normalização — linear e mínima — preserva as relações relativas entre os dados, embora não elimine o efeito de outliers, que podem ser "espremidos" no intervalo de escala.

No trabalho (ZHANG; JIN; ZHOU, 2016), os atributos foram normalizados no intervalo $[0, 1]$ para evitar que variáveis em grandes escalas dominassem a métrica de distância. Os autores compararam os índices DBI e DI em dados normalizados e não normalizados e evidenciaram que a normalização melhora tanto a qualidade dos agrupamentos quanto a escolha do valor de $k$. Esses resultados sustentam a decisão adotada neste trabalho de aplicar a normalização via `MinMaxScaler` como etapa obrigatória de pré-processamento, garantindo que todas as variáveis contribuam proporcionalmente para a clusterização e para as consultas por similaridade em banco vetorial.

## 2.2 Coleta de dados

A coleta de dados é a etapa inicial e crucial para a construção de qualquer base de conhecimento, incluindo a preparação de conjuntos de dados financeiros para análise. Diferentes estratégias podem ser empregadas para essa finalidade, dependendo do tipo de informação necessária, da frequência de atualização e da fonte dos dados. A abordagem mais comum é a extração de séries temporais históricas, que registram o comportamento de preços e volumes ao longo do tempo.

Entre as principais fontes para a coleta de dados financeiros, destacam-se:

- **APIs de provedores de dados financeiros**: Muitas plataformas, como a Bloomberg (Bloomberg, ), Refinitiv (Refinitiv, ) e a Alpha Vantage (Alpha Vantage, ), oferecem APIs (Interfaces de Programação de Aplicações) para acesso programático a dados financeiros. Essas APIs são ferramentas robustas e confiáveis, ideais para projetos que requerem grandes volumes de dados de alta qualidade e com baixa latência.
- **Dados públicos de corretoras**: Algumas corretoras e bolsas de valores disponibilizam dados históricos de preços diretamente em seus sites, permitindo a extração manual ou automatizada por meio de *web scraping*. Por exemplo, a B3 oferece uma série histórica de cotações desde 1986 (B3, ).
- **Web scraping**: Consiste na extração de informações de páginas web, uma técnica útil para dados que não estão disponíveis via APIs. No entanto, é uma estratégia mais frágil, já que a estrutura da página pode mudar, exigindo constante manutenção do código.
- **Bibliotecas Python para acesso a dados financeiros**: Ferramentas como a `yfinance` (AROUSSI, ) permitem interagir de forma simples com fontes como o Yahoo Finance. A biblioteca é **open-source** e possibilita a extração de uma ampla gama de dados financeiros, incluindo preços históricos, dividendos, *splits* e dados fundamentais, diretamente para estruturas do `pandas`, como `DataFrames`.

A escolha da estratégia de coleta de dados deve considerar a confiabilidade da fonte, a abrangência das informações, a frequência de atualização e a facilidade de integração com o restante do projeto. Para projetos acadêmicos e de pesquisa, o uso de bibliotecas de acesso a fontes gratuitas e populares, como o Yahoo Finance, é uma abordagem comum e eficiente.

### 2.2.1 Biblioteca `yfinance`

A biblioteca `yfinance` é uma ferramenta **open-source** em Python, criada por Ran Aroussi, que oferece uma interface simples e poderosa para interagir com o Yahoo Finance. Sua principal vantagem é a capacidade de extrair uma ampla gama de dados financeiros diretamente para estruturas de dados do `pandas`, como `DataFrames` e `Series`, simplificando o processo de análise e manipulação. A biblioteca permite o acesso a informações detalhadas, incluindo dados históricos de preços (abertura, fechamento, máximas, mínimas, volume e preço ajustado), bem como dados fundamentais e corporativos, como dividendos e *splits*. Apesar de não ser uma API oficial, sua popularidade se deve à implementação eficiente, que realiza chamadas HTTP a *endpoints* públicos e processa as respostas em formatos como JSON ou HTML para organizar os dados de forma estruturada.

A escolha da `yfinance` para este trabalho foi uma decisão estratégica. Sua adoção se alinha com estudos acadêmicos similares, como o trabalho de (FIGUEIREDO, 2024), que também utilizou a biblioteca para coletar dados financeiros. Essa consistência metodológica facilita a reprodutibilidade do projeto. Além disso, sua simplicidade acelerou o desenvolvimento e a integração com outras ferramentas do ecossistema Python, como o próprio `pandas`.

No entanto, é crucial reconhecer as **limitações** da `yfinance`. Por depender de *endpoints* não oficiais do Yahoo Finance, a biblioteca está sujeita a instabilidades, como alterações na estrutura das páginas ou bloqueios de acesso em caso de uso intensivo. Por isso, a implementação da coleta de dados em projetos mais robustos deve incluir mecanismos de tratamento de erros e estratégias para evitar o excesso de requisições, como a inclusão de pausas entre as chamadas (*sleep*) para garantir a estabilidade do processo.

No trabalho, a `yfinance` foi utilizada para construir um *pipeline* de dados automatizado, permitindo a coleta de séries históricas e métricas essenciais das ações selecionadas. Esses dados foram integrados diretamente ao fluxo de pré-processamento e, posteriormente, armazenados no SGBD, garantindo que toda a análise pudesse ser realizada dentro do mesmo ambiente.

## 2.3 Armazenamento de Dados

O armazenamento de dados não estruturados apresenta desafios específicos, uma vez que esses não seguem um esquema fixo de atributos e, portanto, não podem ser manipulados apenas por consultas tradicionais baseadas em tabelas. Nesses casos, a recuperação ocorre por meio de pesquisas por similaridade, que determinam quais vetores devem ser retornados com base no grau de semelhança entre os vetores armazenados e o vetor de consulta, em contraste com os bancos de dados relacionais, que são otimizados para consultas exatas e estruturadas (PAN; WANG; LI, 2024; CORONEL; MORRIS, 2015).

Os Sistemas Gerenciadores de Banco de Dados Vetoriais (SGBDV) são compostos por módulos como o processador de consultas, operadores lógicos, implementações físicas de consulta e um gerenciador de armazenamento, responsável pela manutenção de índices de busca e pela administração do armazenamento físico dos vetores (PAN; WANG; LI, 2024). Para otimizar o desempenho, os dados passam inicialmente por um processo de **vetorização** e, em seguida, são **indexados**, o que possibilita buscas mais rápidas e eficientes, de forma semelhante a outros modelos de bancos de dados (KRASKA et al., 2018). A escolha do algoritmo de indexação depende da dimensionalidade dos dados e dos requisitos da pesquisa (TAIPALUS, 2024). Uma vez indexados, os vetores podem ser

recuperados por meio de métricas de distância, como a euclidiana definida na Equação 2.8 e a do cosseno definida na Equação 2.5, conforme discutido na Seção 2.1.2.

As operações centrais nos Sistemas de Gerenciadores de Banco de Dados Vetoriais (SGBDV) consistem na busca de vetores com maior grau de similaridade, para as quais se destacam diferentes tipos de consultas, conforme discutido em (PAN; WANG; LI, 2024). As próximas subseções apresentam os principais tipos de consultas por similaridade.

### 2.3.1 Consulta *k-Nearest Neighbors* (k-NN)

O $k$-NN é uma consulta por similaridade que retorna os $k$ vetores mais próximos de um vetor de consulta $q$, utilizando funções de distância para mensurar a proximidade entre eles. Trata-se de uma busca exata, pois o algoritmo identifica precisamente os $k$ vizinhos mais próximos conforme a métrica definida, em contraste com métodos aproximados (*Approximate Nearest Neighbors – ANN*). Formalmente, pode ser definido conforme a Equação 2.11:

$$k\text{-NN}(q) = \underset{S' \subseteq S, |S'|=k}{\operatorname{argmin}} \sum_{x \in S'} d(x, q), \tag{2.11}$$

onde $S$ é o conjunto de vetores e $d(x, q)$ é a métrica de distância.

### 2.3.2 Consulta *Approximate Nearest Neighbors* (ANN)

O ANN é uma busca aproximada que retorna $k$ vetores de um conjunto $S$ dentro de um raio de $c$ vezes a distância do vizinho mais próximo de $q$ como apresentado na Equação 2.12:

$$\exists S' \subseteq S, |S'| = k \quad \text{tal que} \quad d(x', q) \leq c \cdot \min_{x \in S} d(x, q), \quad \forall x' \in S', \tag{2.12}$$

em que $c$ representa o grau de aproximação.

### 2.3.3 Consulta por Raio (*Range Query*)

A consulta por raio retorna todos os vetores dentro de uma distância $r$ do vetor de consulta $q$ como apresentado na Equação 2.13:

$$Range(q, r) = \{x \in S \mid d(x, q) \leq r\}. \tag{2.13}$$

### 2.3.4 Consultas Predicadas (*Hybrid Queries*)

As consultas predicadas combinam a busca vetorial com a filtragem por atributos estruturados. Cada vetor está associado a um conjunto de atributos, e apenas aqueles que satisfazem um predicado booleano são considerados na busca por similaridade.

### 2.3.5 SGBDV nativos vs. extensão

No TCC descrito aqui, adota-se o `PostgreSQL` com a extensão `pgvector`, que adiciona suporte nativo para operações vetoriais em um ambiente relacional (PostgreSQL Global Development Group, 2025). O pgvector permite a criação de índices eficientes por meio de algoritmos como HNSW *(Hierarchical Navigable Small World)*, IVFFlat *(Inverted File Flat)* e DiskANN *(Disk Approximate Nearest Neighbor)* (Microsoft, 2024), projetados para otimizar buscas por vizinhos mais próximos em grandes volumes de dados, equilibrando precisão e desempenho.

Conforme discutido em (TAIPALUS, 2024), as principais diferenças entre SGBDs puramente vetoriais e extensões vetoriais em SGBDs relacionais estão no foco e nos recursos. Enquanto os SGBDV são projetados especificamente para o gerenciamento eficiente de vetores de alta dimensão, oferecendo funcionalidades completas como coleta de metadados, controle de acesso, escalabilidade e otimização dedicada, as extensões vetoriais em sistemas relacionais mantêm o foco principal no modelo relacional. Por esse motivo, extensões como o `pgvector` não substituem um SGBDV, mas oferecem integração conveniente e eficiente em infraestruturas já existentes.

Entre as principais vantagens do uso de uma extensão vetorial em um SGBD relacional, destacam-se:

- **Capacidade de consulta:** suporte a buscas vetoriais utilizando SQL, como consultas $k$-NN ou por raio;
- **Otimização de consultas:** aproveitamento do otimizador já existente do SGBD relacional;
- **Recursos de sistema:** funcionalidades como replicação, controle de concorrência, tolerância a falhas e segurança são herdadas diretamente do SGBD relacional.

Em resumo, essa abordagem híbrida permite unir a robustez e a maturidade dos sistemas relacionais com a flexibilidade das buscas por similaridade em dados vetoriais, tornando-se uma solução eficiente e prática para aplicações que lidam com grandes volumes de dados financeiros.

## 2.4 Análise dos Dados

As técnicas de aprendizado de máquina não supervisionado desempenham um papel fundamental na análise de grandes volumes de dados, permitindo a identificação de padrões e estruturas ocultas sem a necessidade de rótulos previamente definidos. Entre essas técnicas, destacam-se os métodos de **clusterização**, cujo objetivo é agrupar observações em conjuntos (clusters) de modo que itens pertencentes ao mesmo grupo apresentem maior similaridade entre si do que em relação a outros grupos.

### 2.4.1 Técnicas de Clusterização

Diversos algoritmos podem ser aplicados à clusterização de dados financeiros, conforme destacado em (FIGUEIREDO, 2024), cada um apresentando particularidades metodológicas e diferentes contextos de aplicabilidade.

- **K-Means:** algoritmo iterativo amplamente utilizado, que busca particionar os dados em $k$ clusters, minimizando a soma das distâncias quadradas entre os pontos e os centróides de seus respectivos grupos. É eficaz em bases de dados volumosas, especialmente quando os clusters apresentam formato aproximadamente esférico e tamanhos semelhantes, destacando-se pela simplicidade e escalabilidade.
- **Clusterização Hierárquica (*Hierarchical Clustering*):** técnica que organiza os dados em uma estrutura hierárquica representada por meio de um *dendrograma*. Pode ser conduzida de forma *aglomerativa* (bottom-up), na qual cada elemento inicia como um cluster independente e os grupos mais próximos são combinados sucessivamente, ou *divisiva* (top-down), em que todos os elementos começam em um único grupo e são divididos recursivamente. É útil para explorar relações hierárquicas, mas tende a ser computacionalmente mais custosa que o K-Means.
- ***Gaussian Mixture Model* (GMM):** abordagem probabilística que modela os dados como provenientes de uma combinação de distribuições gaussianas. Diferentemente do K-Means, que atribui cada ponto a apenas um cluster, o GMM calcula probabilidades de pertencimento, permitindo que um mesmo item tenha associação parcial com diferentes grupos. Essa flexibilidade torna a técnica adequada para cenários com clusters de formas e densidades variadas.

Na trabalho (FIGUEIREDO, 2024), observa-se que o **K-Means** se destaca pela sua eficiência e qualidade dos resultados em diversos contextos, sendo amplamente empregado em áreas como análise de dados financeiros, segmentação de mercado e biologia computacional. Entretanto, sua aplicação deve considerar limitações inerentes, como a necessidade de definir previamente o número de clusters e a sensibilidade a ruídos e valores

extremos. Já métodos como GMM e clusterização hierárquica oferecem maior flexibilidade e interpretabilidade em cenários específicos, ainda que com maior custo computacional.

### 2.4.2 K-Means

O algoritmo *K-Means* é uma técnica de aprendizado de máquina não supervisionado amplamente utilizada para a clusterização de dados. Seu objetivo é particionar um conjunto de $n$ itens de dados em $k$ clusters distintos, nos quais cada observação pertence ao cluster cujo centroide está mais próximo. Esse método é eficaz para identificar padrões e estruturas subjacentes em dados não rotulados (AHMED; SERAJ; ISLAM, 2020).

O *K-Means* possui diversas aplicações práticas, incluindo segmentação de mercado (identificação de grupos de consumidores com comportamentos semelhantes), análise de imagens (segmentação para reconhecimento de padrões), análise de dados financeiros (agrupamento de ativos com características similares) e biologia computacional (detecção de padrões em dados genômicos).

#### 2.4.2.1 Funcionamento do K-Means

O *K-Means* opera por meio de um processo iterativo que busca minimizar a soma das distâncias quadradas entre os itens de dados e seus respectivos centróides. O procedimento básico é descrito a seguir (XU; WUNSCH, 2005):

1. **Definição do número de clusters ($k$):** o pesquisador define previamente a quantidade de grupos desejada.
2. **Inicialização dos centróides:** selecionam-se aleatoriamente $k$ itens do conjunto de dados como centróides iniciais.
3. **Atribuição dos pontos:** cada item de dados é associado ao cluster cujo centróide esteja mais próximo, geralmente pela distância euclidiana.
4. **Recalcular centróides:** após a atribuição, os centróides são atualizados como a média dos itens do respectivo cluster.
5. **Iteração:** os passos 3 e 4 se repetem até que os centróides se estabilizem ou seja atingido o número máximo de iterações.

Este processo busca minimizar a função objetivo apresentada na Equação 2.14:

$$J = \sum_{i=1}^{k} \sum_{x_j \in S_i} \|x_j - \mu_i\|^2 \tag{2.14}$$

onde $S_i$ representa o conjunto de itens de dados atribuídos ao $i$-ésimo cluster, $\mu_i$ é o centróide do cluster e $\|\cdot\|$ denota a norma Euclidiana (AHMED; SERAJ; ISLAM, 2020).

#### 2.4.2.2 Escolha do Número de Clusters $k$

A definição do número de clusters $k$ é um aspecto central da clusterização. Métodos como o *elbow method*, a análise de silhueta e a validação cruzada são comumente aplicados para auxiliar na escolha de $k$. Ainda assim, a decisão final deve levar em conta o contexto do problema e os objetivos da análise (AHMED; SERAJ; ISLAM, 2020).

#### 2.4.2.3 Vantagens e Desvantagens do K-Means

O algoritmo K-Means é uma das técnicas de clusterização mais utilizadas em análise de dados numéricos. Apesar de sua popularidade, ele apresenta limitações que definem o contexto adequado para sua aplicação. A seguir, são detalhadas as principais vantagens e desvantagens do método, conforme discutido em (JAIN, 2010).

Dentre as vantagens é possível citar:

- **Simplicidade e Popularidade:** O K-Means é um dos algoritmos de agrupamento mais populares e simples, amplamente utilizado por sua facilidade de implementação, simplicidade e eficiência.
- **Eficiência:** O algoritmo é considerado eficiente e seu sucesso empírico o torna uma escolha frequente para tarefas de clusterização. Por ser um algoritmo ganancioso, converge para um mínimo local rapidamente.
- **Adaptabilidade:** Embora o K-Means utilize a métrica euclidiana por padrão, que favorece a detecção de clusters esféricos, ele pode ser adaptado com outras métricas de distância, como a de Mahalanobis, para identificar clusters com formatos mais complexos (hiperelipsoidais).

Dentre as desvantagens estão:

- **Sensibilidade à representação dos dados:** O desempenho do K-Means depende fortemente da forma como os dados são representados. Clusters não compactos ou com formatos complexos, como anéis concêntricos, podem não ser corretamente identificados.
- **Dependência de parâmetros:** O algoritmo requer que o usuário defina o número de clusters $K$, a inicialização dos centróides e a métrica de distância. A escolha do $K$ ideal é crítica e não há um critério matemático perfeito.

- **Variação nos resultados:** Diferentes inicializações podem gerar resultados distintos, já que o algoritmo converge apenas para mínimos locais. Uma prática comum para mitigar isso é executar o K-Means várias vezes e selecionar a execução com menor erro quadrático.

# 3 Trabalhos Relacionados

Nesta seção são apresentados alguns trabalhos desenvolvidos pela comunidade acadêmica que se relacionam com o tema do trabalho descrito aqui ou que utilizam abordagens metodológicas semelhantes. Os estudos selecionados exploram, principalmente, formas de coleta, pré-processamento e armazenamento de dados financeiros, além da aplicação de algoritmos de mineração de dados e o uso de SGBDV.

## 3.1 Bases de Dados Financeiras no Brasil

No trabalho apresentado por (CARDOSO et al., 2021), é disponibilizado um conjunto de dados denominado *BovDB*, construído a partir de informações da Bolsa de Valores Brasileira (B3), abrangendo o período de 1995 a 2020. Nesse estudo, o autor realiza a coleta de dados brutos diretamente do site da B3 e aplica processos de pré-processamento, estruturando-os em um Sistema Gerenciador de Banco de Dados Relacional (SGBDR). O conjunto de dados resultante é disponibilizado publicamente no GitHub, possibilitando sua reutilização em diferentes aplicações de pesquisa e mineração de dados. Este trabalho é relevante para a presente pesquisa, pois oferece uma fonte de dados financeiros estruturados que podem ser utilizados como base inicial.

O estudo de (NASCIMENTO; MIRANDA, 2015) apresenta a ferramenta *Chronos Ações*, desenvolvida para coletar, armazenar e processar dados históricos da então BM&FBOVESPA (atualmente B3). A ferramenta organiza os dados em um banco relacional e oferece funcionalidades que permitem análises descritivas e consultas tradicionais. Embora compartilhe o objetivo de manipular dados financeiros, o presente trabalho diferencia-se ao propor o uso de um sistema de Sistema Gerenciador de Banco de Dados Vetorial (SGBDV), possibilitando consultas por similaridade entre ativos.

## 3.2 Sistemas de Recomendação e Análise de Investimentos

O TCC de (MOURA, 2021) apresenta um sistema eletrônico de recomendação online e agnóstico, com o objetivo de sugerir fundos de investimento com base no perfil do investidor. Para isso, os dados disponibilizados pela CVM (Comissão de Valores Mobiliários) são coletados, pré-processados e armazenados em um SGBDR. A partir dessa estrutura, o sistema gera recomendações personalizadas. Embora o foco seja em fundos de investimento, este trabalho se relaciona com o TCC ao explorar a manipulação de dados financeiros como suporte à tomada de decisão. No entanto, diferencia-se ao utili-

zar técnicas de recomendação, enquanto aqui será explorada a busca por similaridade e clusterização em bancos de dados vetoriais.

No trabalho de (SANTOS; COSTA, 2023b) os autores aplicam o modelo *Random Forest* para classificar os melhores momentos de compra e venda de ativos no mercado brasileiro. A solução utiliza indicadores de mercado derivados de séries históricas de preços e aplica diferenciação fracionária para lidar com dados não estacionários. Este estudo relaciona-se com a presente pesquisa por também utilizar dados históricos como base de análise, mas se diferencia pela aplicação de algoritmos de previsão. Já este trabalho foca em consultas por similaridade e clusterização de ativos para apoiar a análise de estratégias de investimento.

## 3.3 Aplicações de Bancos de Dados Vetoriais

O estudo de (WINNICKI et al., 2024) propõe a construção de um SGBDV para o armazenamento e análise de dados de expressão gênica. A pesquisa destaca o desafio de lidar com grandes volumes de dados não estruturados e explora o uso de VDBMS (*Vector Database Management Systems*) para consultas por similaridade. Embora aplicado ao campo da bioinformática, o trabalho é relevante por demonstrar o potencial dos bancos vetoriais em cenários de dados complexos, servindo como referência conceitual para a aplicação em dados financeiros explorada neste trabalho.

No trabalho (SOUSA, 2025) é demonstrada a aplicação prática de um SGBDV para o tratamento de dados musicais (arquivos de áudio). O autor realiza uma revisão teórica sobre representação e indexação de dados vetoriais, define um modelo de dados para armazenar músicas como vetores de atributos extraídos, e implementa consultas por similaridade dentro de um banco vetorial. Os experimentos incluem avaliação de métricas como precisão e recall para verificar a eficácia das buscas por similaridade no domínio musical. Os resultados apontam que, embora exista espaço para aprimoramentos, o uso de SGBDV permite operações de recuperação de áudio baseada em similaridade com resultados viáveis, o que reforça a relevância dessa tecnologia para domínios com dados complexos como som e música.

O artigo (PAN; WANG; LI, 2024) apresenta uma análise abrangente e atual sobre os Sistemas de Gerenciamento de Bancos de Dados Vetoriais (SGBDVs), evidenciando a evolução de algoritmos isolados para sistemas completos, impulsionada por aplicações intensivas em dados, como os modelos de linguagem de larga escala (LLMs). Os autores identificam cinco desafios centrais no gerenciamento de dados vetoriais: a ambiguidade inerente à similaridade semântica, o elevado tamanho dos vetores, o alto custo computacional das comparações, a ausência de propriedades estruturais adequadas para indexação e a complexidade de consultas híbridas que combinam busca vetorial com atributos tradi-

cionais. O estudo discute como esses obstáculos vêm sendo superados por novas técnicas de processamento, armazenamento, indexação e execução de consultas, classificando os SGBDVs em duas categorias principais: sistemas nativos, projetados especificamente para vetores, e sistemas estendidos, que incorporam capacidades vetoriais a SGBD já existentes. Além disso, são detalhados diferentes tipos de índices (baseados em tabelas, árvores e grafos), operadores para consultas híbridas, bem como a importância da otimização de consultas e do uso de aceleração por hardware. Dessa forma, o trabalho consolida-se como uma referência essencial para pesquisadores e profissionais, oferecendo uma visão crítica das tecnologias atuais e dos principais desafios futuros no campo dos bancos de dados vetoriais.

## 3.4 Clusterização de Dados Financeiros

O trabalho (ZHANG; JIN; ZHOU, 2016) investiga o uso de algoritmos de clusterização na análise de dados financeiros, aplicando diferentes métodos sobre conjuntos que variam de séries temporais a transações. O estudo compara vantagens e limitações de cada técnica, destacando a importância da normalização dos atributos financeiros, já que estes podem estar em escalas distintas (como valores monetários, porcentagens e índices). Os autores mostram que a normalização melhora significativamente a qualidade dos agrupamentos, tornando os atributos comparáveis. Esse aspecto metodológico foi incorporado neste trabalho, que também adota o pré-processamento dos dados antes da aplicação da clusterização.

Um trabalho particularmente relevante é o (FIGUEIREDO, 2024), que combina a Análise de Componentes Principais (PCA) com técnicas de clusterização, em especial o algoritmo *K-Means*, para categorizar empresas listadas na bolsa. A PCA é utilizada para reduzir a dimensionalidade dos indicadores financeiros, como volatilidade, capitalização de mercado e dividendos, preservando os fatores mais relevantes. Em seguida, a clusterização organiza as empresas em grupos que refletem diferentes perfis de risco e retorno: clusters com alta volatilidade e baixo rendimento são associados a investimentos mais arriscados, enquanto grupos com menor oscilação e retornos estáveis alinham-se a estratégias conservadoras. Esse trabalho demonstra como a combinação de PCA e clusterização pode facilitar a avaliação de riscos e oportunidades, tornando a análise mais intuitiva tanto para iniciantes quanto para investidores experientes.

O presente trabalho tem como base a pesquisa (FIGUEIREDO, 2024), mas busca expandir sua aplicação ao integrar técnicas de clusterização a um ambiente de SGBD, utilizando o PostgreSQL com a extensão `pgvector`. Além de aplicar o processo de clusterização, pretende-se implementar consultas por similaridade diretamente no SGBD, otimizando a análise de grandes volumes de dados financeiros. Essa abordagem apresenta

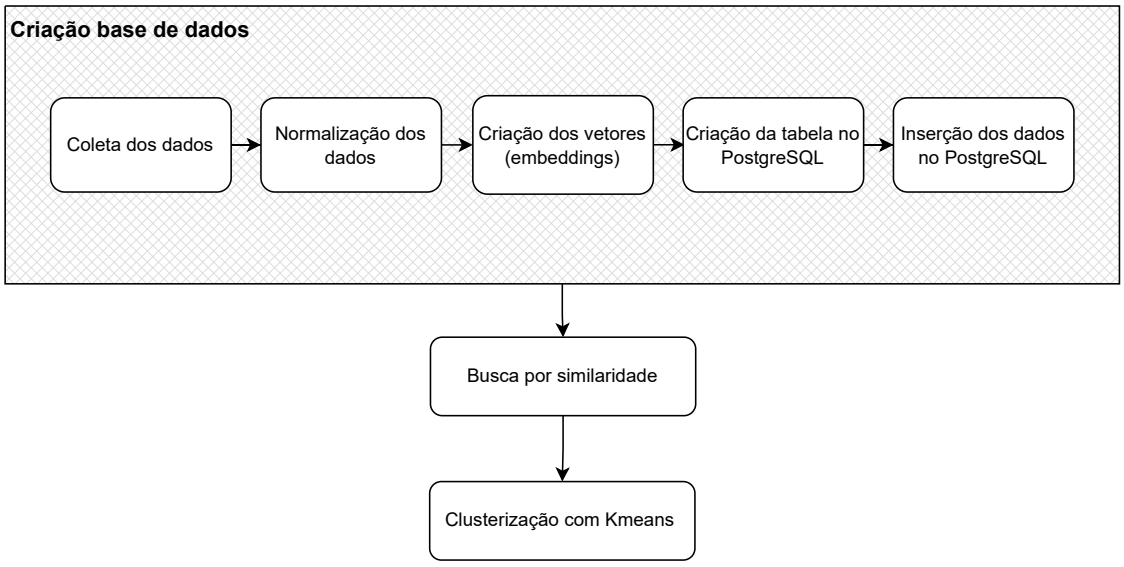
vantagens significativas, uma vez que todo o processo de análise ocorre dentro do próprio SGBD, eliminando a necessidade de exportar os dados para ferramentas externas, reduzindo custos de movimentação e riscos de inconsistência. Além disso, permite explorar recursos nativos do SGBD, como controle de concorrência, replicação, tolerância a falhas e otimização de consultas, resultando em maior desempenho e confiabilidade. Dessa forma, este trabalho não apenas se inspira nos estudos anteriores, mas também avança no sentido de aproximar as técnicas de mineração de dados das ferramentas práticas utilizadas em ambientes reais.

# 4 Método de Trabalho

A metodologia deste trabalho se baseia e expande o trabalho *Analyzing Stock Market Data Using Unsupervised Learning Techniques* (FIGUEIREDO, 2024), com o objetivo de demonstrar a viabilidade de se executar o processo de clusterização e busca por similaridade diretamente em um Sistema Gerenciador de Banco de Dados (SGBD) relacional. Para tal, a abordagem inicia com a coleta de dados de ações da NYSE (*New York Stock Exchange*), realizada com a biblioteca *yfinance*, seguida pelo pré-processamento, que inclui a normalização e a criação dos vetores dos dados. O ponto-chave da pesquisa é o armazenamento desses vetores em um SBGD PostgreSQL com a extensão *pgvector*, que permite o processamento de vetores de alta dimensionalidade de forma nativa. Com os dados estruturados, foi possível implementar o algoritmo *K-Means* e realizar consultas por similaridade dentro do próprio SGBD. Essa metodologia foi aplicada a duas bases de dados: uma base própria, construída com dados da NYSE, e uma base disponibilizada pela autora do trabalho citado em (FIGUEIREDO, 2024), permitindo a comparação entre os resultados.

A Figura 1 apresenta o fluxo completo do método de trabalho adotado nesta pesquisa.

Figura 1 – Fluxo de processamento dos dados.




Fonte: autoria própria.

## 4.1 Obtenção da base de dados

Esta seção detalha o processo de coleta e pré-processamento dos dados, fundamentais para a criação da base de dados que será utilizada para a clusterização. O texto explica as etapas de coleta dos dados da NYSE e o pré-processamento para a normalização e geração dos vetores, preparando-os para a inserção no SGBD.

### 4.1.1 Coleta de Dados e *Tickers*

A análise proposta neste trabalho é baseada em um conjunto de dados financeiros extraídos de companhias listadas na Bolsa de Valores de Nova York (NYSE). O dataset, criado a partir de uma coleta automatizada de *tickers* e dados financeiros, foi projetado para representar o perfil de cada ativo por meio de métricas fundamentais. Atributos como `Volatilidade`, `Beta` e `Índice P/L` foram selecionados por sua relevância na avaliação de risco e retorno, permitindo uma análise comparativa e a identificação de padrões de similaridade, conforme detalhado nas seções a seguir.

Para a etapa de coleta de dados, foi necessário, primeiramente, obter os *tickers* (símbolos de negociação) de todas as companhias listadas na NYSE. A lista completa foi obtida através do site da Wikipédia, conforme a referência (CONTRIBUTORS, 2025), que disponibiliza as ações por ordem alfabética. Para automatizar essa tarefa, foi desenvolvido um *script* em Python que utilizou as bibliotecas *requests* e *pandas* para extrair os *tickers* de cada página da enciclopédia online.

O código, apresentado na Listagem 4.1, percorreu as páginas da Wikipédia, de A a Z e 0-9, buscando e extraindo os dados das tabelas HTML. A saída do *script* foi uma lista com os *tickers* de todas as companhias encontradas, que foram, em seguida, persistidos em um arquivo `.csv`. Este arquivo serviu como a base principal para a coleta dos dados financeiros de cada empresa nas etapas subsequentes do método de trabalho.

```
import requests
import pandas as pd

companies = 'https://en.wikipedia.org/wiki/
    Companies_listed_on_the_New_York_Stock_Exchange_(0%E2%80%939)'

headers = {
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit
        /537.36 "
                  "(KHTML, like Gecko) Chrome/116.0 Safari/537.36"
}

response = requests.get(companies, headers=headers)
response.raise_for_status()  # Garante que n o deu erro HTTP
```

```
data_table = pd.read_html(response.text)
tickers = data_table[0]['Symbol'].tolist()

for letter in 'ABCDEFGHIJKLMNOPQRSTUVWXYZ':
    try:
        companies_url = f'https://en.wikipedia.org/wiki/
            Companies_listed_on_the_New_York_Stock_Exchange_({letter})'
        print(f"Buscando empresas da p gina: {letter}")

        response = requests.get(companies_url, headers=headers)
        response.raise_for_status()

        data_table = pd.read_html(response.text)
        new_tickers = data_table[0]['Symbol'].tolist()
        tickers += new_tickers

        print(f"Adicionados {len(new_tickers)} tickers da p gina {
            letter}")

    except Exception as e:
        print(f"Erro ao processar p gina {letter}: {e}")
        continue

tickers_df = pd.DataFrame({'Ticker': tickers})
tickers_df.to_csv('tickers_nyse.csv', index=False)
```

Listing 4.1 – Coleta dos tickers das companhias listadas na NYSE

Com a lista de *tickers* em mãos, a coleta dos dados financeiros foi realizada utilizando a biblioteca Python *yfinance*, que acessa a base de dados do Yahoo Finance. Durante a coleta, os seguintes atributos, essenciais para a análise do perfil de cada companhia, foram selecionados:

- **Beta**: Um coeficiente que mede a sensibilidade de uma ação em relação às flutuações do mercado. Ele indica a volatilidade de uma ação individual comparada ao risco geral do mercado, sendo que ações com betas maiores que 1 são consideradas mais voláteis.

$$\beta = \frac{\text{Cov}(R_{\text{Ativo}}, R_{\text{Mercado}})}{\text{Var}(R_{\text{Mercado}})} \quad (4.1)$$

  Onde:

  - $\text{Cov}(R_{\text{Ativo}}, R_{\text{Mercado}})$ = Covariância entre o retorno do ativo ($R_{\text{Ativo}}$) e o retorno do mercado ($R_{\text{Mercado}}$). A covariância mede como os retornos do ativo e do mercado se movem em conjunto, com um valor positivo indicando que se movem

na mesma direção e um valor negativo indicando que se movem em direções opostas.

- $\text{Var}(R_{\text{Mercado}})$ = Variância do retorno do mercado.

- **Volatilidade**: Uma medida de risco que indica o quanto os retornos de um ativo ou de um índice de mercado variam ao longo do tempo. É calculada a partir do desvio padrão ou da variância desses retornos, e quanto maior a volatilidade, maior o risco percebido do investimento. Para facilitar a comparação, a volatilidade é frequentemente apresentada de forma anualizada.

$$\text{Volatilidade} = \sigma\sqrt{T} \quad (4.2)$$

Onde:

- $\sigma$ = Desvio padrão dos retornos.
- $T$ = Número de períodos no período de tempo.

- **EPS (Lucro por Ação)**: Uma medida da lucratividade de uma empresa que indica quanto de lucro cada ação ordinária em circulação gerou.

$$\text{EPS} = \frac{\text{NI}}{\text{A}_{\text{Diluídas}}} \quad (4.3)$$

Onde:

- NI = Resultado líquido.
- $\text{A}_{\text{Diluídas}}$ = Número total de ações diluídas em circulação, que inclui não apenas as ações atualmente em circulação, mas também aquelas que poderiam ser emitidas através da conversão de opções, títulos conversíveis e *warrants* (títulos que conferem ao investidor o direito de adquirir ações da empresa a um preço predefinido dentro de um período específico), que são instrumentos financeiros que dão ao titular o direito de comprar ações da empresa a um preço predeterminado dentro de um período específico)

- **Índice P/L (Preço/Lucro)**: Uma métrica usada para avaliar o valor de mercado de uma empresa. Ele é calculado dividindo o preço atual de uma ação pelo lucro que a empresa gerou por cada ação em um determinado período.

- ***Dividend Yield***: Indica o rendimento do dividendo. É um índice para medir a rentabilidade dos dividendos de uma empresa em relação ao preço de sua ação.

$$\text{Dividend Yield} = \frac{\text{D}_{\text{Ação}}}{\text{P}_{\text{Ação}}} \quad (4.5)$$

Onde:

  - $D_{\text{Ação}}$ = Dividendo por ação.
  - $P_{\text{Ação}}$ = Valor de mercado por ação.

- **Capitalização de Mercado**: Uma estimativa do valor de mercado de uma empresa com base nas expectativas sobre futuras condições econômicas e monetárias. Sendo calculada multiplicando o preço atual da ação pelo número total de ações em circulação.

$$\text{Capitalização de Mercado} = P_{\text{Atual}} \times A_{\text{Em Circulação}} \quad (4.6)$$

  Onde:

  - $P_{\text{Atual}}$ = Preço atual da ação.
  - $A_{\text{Em Circulação}}$ = Número total de ações em circulação, que representa o número total de ações atualmente em circulação, excluindo as ações em tesouraria.

- **Crescimento da Receita**: Refere-se ao aumento na receita total de uma empresa ao longo de um período específico.

$$\text{Taxa de Crescimento da Receita} = \frac{R_{\text{Atual}} - R_{\text{Anterior}}}{R_{\text{Anterior}}} \quad (4.7)$$

  Onde:

  - $R_{\text{Atual}}$ = Receita do período atual.
  - $R_{\text{Anterior}}$ = Receita do período anterior.

- **Índice Dívida/Patrimônio Líquido**: Um índice financeiro que indica a proporção relativa de patrimônio líquido e dívida usados para financiar os ativos de uma empresa.

$$\text{Índice Dívida/Patrimônio Líquido} = \frac{P_{\text{Total}}}{PL_{\text{Total}}} \quad (4.8)$$

  Onde:

  - $P_{\text{Total}}$ = Total de passivos, que representam todas as obrigações financeiras de uma empresa.
  - $PL_{\text{Total}}$ = Patrimônio líquido total dos acionistas, que representa a diferença entre o total de ativos de uma empresa e seus passivos totais.

O único atributo que não pôde ser obtido diretamente pela biblioteca *yfinance* foi a **Volatilidade**. Para contornar essa limitação, ela foi calculada a partir do histórico diário de preços de fechamento das ações. A volatilidade anualizada foi estimada utilizando o desvio padrão dos retornos diários, multiplicado pela raiz quadrada de 252, que representa o número médio de dias úteis no mercado financeiro. A equação utilizada para esse cálculo encontra-se na Equação (4.2).

O processo de coleta e pré-processamento dos dados está ilustrado no código Python apresentado na Listagem 4.2. Esse código pode ser dividido em três etapas principais:

1. **Definição da função de coleta**: A função `get_stock_data` recebe como entrada o *ticker* da empresa e acessa os dados disponíveis na API do *Yahoo Finance* por meio da biblioteca *yfinance*. Dentro dela, são extraídos atributos fundamentais como *Beta*, *P/E Ratio*, *Dividend Yield*, *Market Cap*, *Revenue Growth*, *EPS Growth* e *Debt-to-Equity*.

2. **Cálculo da volatilidade**: Caso exista histórico de preços disponível para o ativo no último ano, calcula-se a série de retornos diários (`pct_change()`) a partir dos preços de fechamento. Em seguida, é obtido o desvio padrão desses retornos, que é anualizado multiplicando-se por $\sqrt{252}$.

3. **Criação do DataFrame final**: Após coletar os dados de todos os ativos, os resultados são organizados em um `DataFrame` da biblioteca *pandas*, que é posteriormente tratado para remover possíveis valores nulos antes de sua inserção no banco de dados.

A Listagem 4.2 apresenta a implementação do processo de coleta de dados com a biblioteca *yfinance*.

```
import yfinance as yf
import pandas as pd
import time

def get_stock_data(ticker_symbol):
    try:
        ticker = yf.Ticker(ticker_symbol)
        info = ticker.info
        data = {
            'Ticker': ticker_symbol,
            'Beta': info.get('beta', 'N/A'),
            'P/E Ratio': info.get('trailingPE', 'N/A'),
            'Dividend Yield': info.get('dividendYield', 'N/A'),
            'Market Cap': info.get('marketCap', 'N/A'),
            'Revenue Growth': info.get('revenueGrowth', 'N/A'),
            'EPS Growth': info.get('trailingEps', 'N/A'),
            'Debt-to-Equity': info.get('debtToEquity', 'N/A'),
            'Volatility (1 ano)': 'N/A'
        }

        hist = ticker.history(period="1y")
```

```
        if not hist.empty:
            daily_returns = hist['Close'].pct_change()
            volatility = daily_returns.std() * (252**0.5)
            data['Volatility (1 ano)'] = volatility
        return data

    except Exception as e:
        print(f"Erro ao processar o ticker {ticker_symbol}: {e}")
        return {'Ticker': ticker_symbol, 'Status': 'Erro'}

all_data = []
tickers_df = pd.read_csv('tickers_nyse.csv')
tickers = tickers_df['Ticker'].tolist()

for ticker_symbol in tickers:
    print(f"Coletando dados para {ticker_symbol}...")
    stock_data = get_stock_data(ticker_symbol)
    all_data.append(stock_data)
    time.sleep(1)

df = pd.DataFrame(all_data)
df = df.dropna()
```

Listing 4.2 – Coleta e pré-processamento de dados utilizando a biblioteca yfinance

### 4.1.2 Pré-processamento dos dados

O pré-processamento dos dados foi uma etapa crucial para garantir a qualidade e a consistência do conjunto de dados utilizado na análise. Inicialmente, foi realizado o **tratamento de valores ausentes e de dados duplicados**. Esse passo é fundamental, pois a presença de informações incompletas ou duplicadas pode comprometer a acurácia de métodos de análise estatística.

No código da Listagem 4.3, esse processo ocorreu em três etapas:

1. **Substituição de valores inválidos**: O comando

   ```
   df_clean = df_original.replace([np.inf, -np.inf,'N/A'], pd.NA)
   ```

   substituiu todas as ocorrências de valores considerados inválidos (como `N/A`, `inf` e `-inf`) pelo identificador nativo de valores ausentes do *pandas* (`pd.NA`). Dessa forma, diferentes representações de dados inválidos foram padronizadas para um mesmo formato de ausência.

2. **Remoção de registros incompletos**: Após a padronização, a instrução

   ```
   df_clean = df_clean.dropna()
   ```

eliminou todas as linhas que continham pelo menos um valor ausente. Essa operação garantiu que apenas observações com informações completas fossem preservadas no conjunto de dados final, reduzindo riscos de viés ou falhas em cálculos posteriores.

3. **Remoção de registros duplicados**: Após a remoção de registros incompletos, a instrução

```
df_clean = df_clean.drop_duplicates(subset=['Ticker'], keep='first'
    )
```

eliminou uma das linhas com dados duplicados.

Após essa limpeza inicial, foi aplicada a **normalização dos dados** para adequar todos os atributos à mesma escala. Essa etapa foi necessária porque os atributos financeiros apresentam ordens de magnitude muito diferentes (por exemplo, *Market Cap* pode estar na casa de bilhões, enquanto *Dividend Yield* normalmente varia entre 0 e 10). Para evitar que atributos com valores absolutos maiores dominassem o processo de análise, foi utilizada a função `MinMaxScaler()` da biblioteca *sklearn.preprocessing*, que transforma todos os valores para o intervalo $[0, 1]$.

O processo completo de pré-processamento, incluindo a padronização de valores ausentes, remoção de registros incompletos e normalização dos dados, pode ser visualizado no código Python apresentado na Listagem 4.3.

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

df_original = pd.read_csv('stock_data.csv')

# Tratamento de dados ausentes e infinitos
df_clean = df_original.replace([np.inf, -np.inf,'N/A'], pd.NA)
df_clean = df_clean.dropna()
df_clean = df_clean.drop_duplicates(subset=['Ticker'], keep='first')

numeric_columns = ['Beta', 'Volatility (1 ano)', 'P/E Ratio', 'Dividend
    Yield', 'Market Cap', 'Revenue Growth', 'EPS Growth', 'Debt-to-Equity
    ']

# Normaliza  o dos dados
numeric_columns = ['Beta', 'Volatility (1 ano)', 'P/E Ratio', 'Dividend
    Yield',
                   'Market Cap', 'Revenue Growth', 'EPS Growth', 'Debt-to
                       -Equity']

df_normalized = df_clean.copy()
```

```
scaler = MinMaxScaler()

df_normalized[numeric_columns] = scaler.fit_transform(df_normalized[
    numeric_columns])

df_clean['embedding'] = df_normalized[numeric_columns].apply(
    lambda row: np.array(row.values), axis=1
)

output_file_embeddings = 'stock_data_embeddings.csv'

df_to_save = df_clean.copy()
df_to_save['embedding'] = df_to_save['embedding'].apply(lambda x: ','.
    join(map(str, x)))

df_to_save.to_csv(output_file_embeddings, index=False, encoding='utf-8')
```

Listing 4.3 – Pré-processamento dos dados

## 4.2 Criação do Banco de Dados no PostgreSQL

O banco de dados relacional foi modelado e criado no sistema gerenciador de banco de dados `PostgreSQL`, que oferece suporte nativo à extensão `pgvector`. Esta funcionalidade é crucial para o trabalho, pois permite o armazenamento de vetores de alta dimensionalidade e viabiliza a execução de buscas por similaridade diretamente no SGBD, minimizando a necessidade de movimentação de dados.

A estrutura do banco é composta pela tabela `stock_data`, projetada para armazenar todos os atributos coletados na etapa de pré-processamento. A tabela inclui uma coluna específica do tipo `vector(8)`, dedicada a receber os vetores normalizados de 8 dimensões, conforme ilustrado no código de criação:

```
CREATE TABLE stock_data (
    ticker VARCHAR(10) PRIMARY KEY,
    beta DECIMAL(5, 3),
    volatility DECIMAL(18, 16),
    pe_ratio DECIMAL(10, 4),
    dividend_yield DECIMAL(5, 2),
    market_cap DECIMAL(15, 2),
    revenue_growth DECIMAL(10, 8),
    eps_growth DECIMAL(10, 2),
    debt_to_equity DECIMAL(15, 10),
    embedding vector(8)
);
```

Listing 4.4 – Criação da tabela `stock_data`

### 4.2.1 Inserção de Dados no SGBD PostgreSQL

Com a base de dados estruturada no PostgreSQL e os vetores já normalizados no ambiente Python, a próxima etapa foi a persistência dos dados no SGBD. Para isso, utilizamos a biblioteca `psycopg2`, que facilita a conexão e a execução de comandos SQL a partir de aplicações Python.

O processo de inserção foi realizado em um laço de repetição, onde cada linha do *dataframe* com os dados processados foi lida e enviada para a tabela `stock_data`. O vetor de características normalizadas foi especificamente formatado como uma *string* para atender ao formato da extensão `pgvector` (ex: '[valor1, valor2, ..., valor8]'), garantindo a correta persistência do vetor no banco de dados.

O código a seguir, apresentado na Listagem 4.5, ilustra o processo de conexão, iteração e inserção dos dados.

```
import psycopg2
import pandas as pd

conn = psycopg2.connect(
    dbname="nome_do_banco",
    user="usuario",
    password="senha",
    host="localhost",
    port="5432"
)

cur = conn.cursor()

df = pd.read_csv('stock_data_embeddings.csv')

# Insere os dados
for idx, row in df.iterrows():
    # Prepara os dados de cada linha
    ticker = row["Ticker"]
    beta = row["Beta"]
    volatility = row["Volatility (1 ano)"]
    pe_ratio = row["P/E Ratio"]
    dividend_yield = row["Dividend Yield"]
    market_cap = row["Market Cap"]
    revenue_growth = row["Revenue Growth"]
    eps_growth = row["EPS Growth"]
    debt_to_equity = row["Debt-to-Equity"]
    embedding = row["embedding"]

    # Formata o embedding para o tipo 'vector' do pgvector
    embedding_formatted = f"[{embedding}]"
```

```
    query = """
    INSERT INTO stock_data (
        ticker, beta, volatility, pe_ratio, dividend_yield, market_cap,
        revenue_growth, eps_growth, debt_to_equity, embedding
    )
    VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s::vector)
    """

    # Executa a query com os dados
    cur.execute(query, (
        ticker, beta, volatility, pe_ratio, dividend_yield, market_cap,
        revenue_growth, eps_growth, debt_to_equity, embedding_formatted
    ))

# Confirma as altera  es e fecha a conex o
conn.commit()
cur.close()
conn.close()
```

Listing 4.5 – Inserção dos dados no banco PostgreSQL

## 4.3 Busca por Similaridade

Com os vetores previamente normalizados e armazenados na tabela `stock_data`, foi possível realizar consultas por similaridade utilizando a extensão `pgvector`. Essa extensão permite efetuar comparações vetoriais diretamente no SGBD PostgreSQL, oferecendo suporte a diferentes métricas de distância.

No presente trabalho, foram realizados testes utilizando duas métricas principais: a **distância Euclidiana** (`<->`) e a **distância do Cosseno** (`<=>`). O objetivo foi avaliar se diferentes medidas de similaridade produziam resultados significativamente distintos na ordenação dos ativos mais próximos a um vetor de consulta.

A seguir, são apresentados os principais operadores de distância suportados pelo `pgvector`:

- `<->`: distância Euclidiana
- `<#>`: produto interno negativo
- `<=>`: distância cosseno
- `<+>`: distância de Manhattan

O Exemplo 4.6 apresenta uma consulta SQL utilizando a distância Euclidiana. Nessa consulta, um vetor de referência é definido manualmente, e o SGBD retorna os 10 ativos (`ticker`) mais próximos em relação a esse vetor. Além dos identificadores dos ativos, a consulta também retorna seus atributos financeiros e o valor calculado da distância.

```
SELECT ticker, beta, volatility, pe_ratio, dividend_yield, market_cap,
   revenue_growth, eps_growth, debt_to_equity,
       embedding <-> '[0.6220839813374806,
                   0.24535475817270114,
                   0.0030513422510418825,
                   0.037336154080887654,
                   0.0023415441261014328,
                   0.19826086956521738,
                   0.023230160275128153,
                   0.0025983957805567957]'::vector AS distance
FROM stock_data
ORDER BY distance ASC
LIMIT 10;
```

Listing 4.6 – Consulta de similaridade utilizando a distância Euclidiana

Nesse exemplo, o operador `<->` calcula a distância Euclidiana entre o vetor de consulta e o vetor de cada ativo armazenado na tabela `stock_data`. Em seguida, os resultados são ordenados de forma crescente, de modo que os ativos mais semelhantes (menor distância) apareçam no topo da lista.

De forma análoga, a métrica de **distância do Cosseno** (`<=>`) também foi aplicada, permitindo comparar os resultados obtidos com diferentes medidas de similaridade. Essa análise foi importante para verificar a consistência das recomendações de ações em função da escolha da métrica.

## 4.4 Clusterização com K-Means

Para agrupar os ativos com características financeiras semelhantes, foi implementado o algoritmo *K-Means* diretamente no SGBD PostgreSQL, utilizando a extensão `pgvector` para manipulação de vetores armazenados na tabela `stock_data`.

O processo foi encapsulado em uma *stored procedure* chamada `run_kmeans_stock`, que recebe como parâmetros o número de clusters desejado ($k$) e o número máximo de iterações (`max_iter`). Neste trabalho, foi adotado o valor $k = 3$, fundamentado na análise apresentada em (FIGUEIREDO, 2024), que demonstrou a existência de três grupos principais de comportamento entre ativos financeiros em contextos semelhantes.

As principais etapas da procedure são descritas a seguir:

1. **Inicialização dos centroides**: Os $k$ centroides iniciais são selecionados aleatoriamente a partir dos vetores existentes, e recebem um `cluster_id` sequencial de 1 até $k$. Essa abordagem garante que a identificação dos clusters permaneça consistente ao longo das iterações.

2. **Atribuição dos clusters**: Para cada ativo, é calculada a distância Euclidiana entre o seu vetor de características e cada centroide existente. Cada ativo é então atribuído ao cluster mais próximo. Em seguida, os `cluster_id`s são reindexados utilizando a função `DENSE_RANK()` para garantir uma sequência contínua de 1 a $k$, mesmo que algum cluster fique vazio em determinada iteração.

3. **Verificação de convergência**: Após a atribuição, a procedure verifica se houve alguma mudança na composição dos clusters em relação à iteração anterior. Caso não haja alterações, o algoritmo considera que os clusters convergiram e interrompe as iterações antes de atingir o limite máximo.

4. **Atualização dos centroides**: Para cada cluster, é recalculado o vetor do centroide como a média dos vetores dos ativos pertencentes a ele. Essa operação é realizada em cada iteração, permitindo que os centroides se ajustem progressivamente à distribuição real dos dados.

O código da procedure também utiliza as tabelas auxiliares `centroids` e `clustering` para armazenar, respectivamente, os centroides de cada iteração e a atribuição de cada ativo a um cluster. Esse armazenamento histórico permite acompanhar a evolução da clusterização ao longo das iterações. A definição dessas tabelas é apresentada na Listagem 4.7.

```
CREATE TABLE IF NOT EXISTS centroids (
    iteration INTEGER,
    cluster_id INTEGER,
    centroid vector(8)
);

CREATE TABLE IF NOT EXISTS clustering (
    iteration INTEGER,
    ticker VARCHAR(10),
    cluster_id INTEGER
);

CREATE INDEX IF NOT EXISTS idx_clustering_iter_ticker ON clustering(
   iteration, ticker)
```

Listing 4.7 – Tabelas auxiliares K-means

A implementação da procedure `run_kmeans_stock` em PostgreSQL/PLpgSQL é apresentada na Listagem 4.8.

```
CREATE OR REPLACE PROCEDURE run_kmeans_stock(k INT, max_iter INT)
LANGUAGE plpgsql
AS $$
DECLARE
    iter INT := 1;
    clusters_changed BOOLEAN := TRUE;
BEGIN
    TRUNCATE centroids, clustering RESTART IDENTITY;

    RAISE NOTICE 'Iniciando K-Means com % clusters e at  % itera  es'
        , k, max_iter;

    -- Etapa 0: Inicializa  o dos centroides
    INSERT INTO centroids (iteration, cluster_id, centroid)
    SELECT 1, row_number() OVER (), embedding
    FROM stock_data
    WHERE embedding IS NOT NULL
    ORDER BY random()
    LIMIT k;

    WHILE iter <= max_iter AND clusters_changed LOOP
        RAISE NOTICE 'Itera  o %', iter;

        -- Etapa 1 + 2: Atribui cada ticker ao cluster mais proximo
        DELETE FROM clustering WHERE iteration = iter;

        WITH distances_ranked AS (
            SELECT
                s.ticker,
                c.cluster_id,
                ROW_NUMBER() OVER (
                    PARTITION BY s.ticker
                    ORDER BY s.embedding <-> c.centroid
                ) AS rn
            FROM stock_data s
            JOIN centroids c ON c.iteration = iter
            WHERE s.embedding IS NOT NULL
        ),
        assigned AS (
            SELECT
                iter AS iteration,
                ticker,
                cluster_id
            FROM distances_ranked
```

```
44              WHERE rn = 1
45          ),
46          reindexed AS (
                                                                    --
               (7)
47              SELECT
48                  a.iteration,
49                  a.ticker,
50                  DENSE_RANK() OVER (ORDER BY a.cluster_id) AS cluster_id
51              FROM assigned a
52          )
53          INSERT INTO clustering (iteration, ticker, cluster_id)
54          SELECT iteration, ticker, cluster_id
55          FROM reindexed;
56
57          -- -- Etapa 3: Verifica se houve mudanca em relacao a iteracao
               anterior
58          IF iter > 1 THEN
59              SELECT EXISTS (
60                  SELECT 1
61                  FROM clustering c1
62                  JOIN clustering c2
63                    ON c1.ticker = c2.ticker
64                  WHERE c1.iteration = iter AND c2.iteration = iter - 1
65                    AND c1.cluster_id <> c2.cluster_id
66              ) INTO clusters_changed;
67          END IF;
68
69          -- Etapa 4: Atualiza os centroides (com cluster_id sequencial )
70          INSERT INTO centroids (iteration, cluster_id, centroid)
71          SELECT
72              iter + 1,
73              DENSE_RANK() OVER (ORDER BY c.cluster_id),
74              avg(s.embedding)
75          FROM clustering c
76          JOIN stock_data s ON s.ticker = c.ticker
77          WHERE c.iteration = iter
78          GROUP BY c.cluster_id;
79
80          iter := iter + 1;
81      END LOOP;
82
83      RAISE NOTICE 'K-Means finalizado ap s % itera  es', iter - 1;
84  END;
85  $$;
```

Listing 4.8 – Implementação do algoritmo K-Means em PL/pgSQL

A Listagem 4.8 apresenta a implementação completa do algoritmo *K-Means* em `PL/pgSQL`, explorando os recursos da extensão `pgvector` para manipulação de vetores diretamente no SGBD PostgreSQL. O código é estruturado em etapas que correspondem às fases tradicionais do algoritmo de clusterização.

**Linhas (8-19):** Inicializa o ambiente de execução, removendo dados anteriores das tabelas `centroids` e `clustering`. Em seguida, realiza a inicialização aleatória dos centroides, selecionando $k$ vetores da tabela `stock_data` como pontos de partida para os clusters.

**Linha (20):** Inicia o laço principal do algoritmo, que se repete até atingir o número máximo de iterações ou até que os clusters não sofram mais alterações (convergência).

**Linhas (24-46):** Correspondem às Etapas 1 e 2 do *K-Means*: calcula-se a distância vetorial entre cada vetor de entrada e os centroides utilizando o operador `<->` da extensão `pgvector`, responsável por medir a similaridade entre vetores. Cada ativo é então atribuído ao cluster mais próximo, armazenando o resultado na tabela `clustering`.

**Linha (58):** Implementa a verificação de convergência, comparando as atribuições de cada ativo entre iterações consecutivas. Caso não haja mudanças, o laço é encerrado.

**Linhas (70-78):** Executa a atualização dos centroides, calculando a média dos vetores de cada grupo, o que gera novos centroides para a próxima iteração.

**Linha (80):** Incrementa o contador de iteração, repetindo o processo até que os clusters se estabilizem.

Ao final, o procedimento exibe uma mensagem indicando o número de iterações realizadas. Essa abordagem permite executar o processamento vetorial completo dentro do próprio SGBD, eliminando a necessidade de exportar os dados para linguagens externas e garantindo eficiência, consistência e escalabilidade no processamento de grandes volumes de dados financeiros.

# 5 Resultados

Este capítulo apresenta os resultados obtidos a partir da aplicação do método de trabalho descrito anteriormente, bem como a análise desses resultados. Inicialmente, é detalhada a base de dados final utilizada, incluindo sua dimensão e forma de representação. São apresentados os testes de busca por similaridade com diferentes métricas de distância e, posteriormente, a clusterização das ações com o algoritmo K-Means. Por fim, são discutidas as vantagens da execução diretamente no SGBD PostgreSQL, utilizando a extensão `pgvector` e as limitações identificadas no método proposto.

A implementação do K-Means diretamente no PostgreSQL proporciona diversas vantagens, como a manipulação de grandes volumes de dados sem a necessidade de exportação para linguagens externas e a execução eficiente de operações vetoriais utilizando `pgvector`. Além disso, a escolha de $k = 3$ possibilitou a formação de grupos interpretáveis, alinhados com a literatura de referência.

## 5.1 Base de dados final

Após a aplicação do método de trabalho ao conjunto de dados do estudo correlato, observou-se que apenas nove *tickers*, dentre 698 analisados, foram alocados em clusters distintos, o que evidencia a consistência do procedimento. Neste contexto, o objetivo principal desta etapa foi validar o método de trabalho, considerando os resultados descritos no estudo correlato (FIGUEIREDO, 2024).

Com base nessa validação, procedeu-se à realização de uma nova coleta de dados, visando ampliar a análise e testar a robustez da abordagem em um conjunto de dados mais abrangente. Para isso, foram identificadas as companhias listadas na NYSE, extraídos seus tickers, obtidos os dados financeiros por meio da biblioteca *yfinance* e, em seguida, efetuado o pré-processamento necessário. Essa coleta de dados foi realizada no mês de setembro de 2025, aspecto relevante pois a data pode influenciar o conjunto obtido. O processo resultou em uma base final composta por 819 tickers, abrangendo empresas de diferentes setores, garantindo diversidade e robustez na análise.

A base de dados consolidada contém as seguintes colunas: `Ticker`, `Beta`, `Volatility (1 ano)`, `P/E Ratio`, `Dividend Yield`, `Market Cap`, `Revenue Growth`, `EPS Growth`, `Debt-to-Equity` e `embedding`. As variáveis foram selecionadas por sua relevância financeira: *Beta* indica o risco sistemático do ativo, *P/E Ratio* relaciona preço e lucro, *Dividend Yield* demonstra o retorno ao acionista, enquanto métricas como *Revenue Growth* e *EPS Growth* refletem o desempenho operacional da companhia. O pré-processamento incluiu

a remoção de valores ausentes e a normalização conjunta dos atributos, garantindo que cada variável contribuísse de forma equilibrada para os cálculos de similaridade.

A coluna `embedding` sintetiza todas as demais variáveis em um vetor normalizado de oito dimensões, que serve como base para as análises de similaridade e clusterização. Essa abordagem permite comparar empresas de forma consistente, considerando múltiplos indicadores financeiros simultaneamente.

Para fins de clareza, a base final é apresentada em duas representações: uma com os atributos originais e outra exclusivamente com os vetores, conforme ilustrado na Figura 2. Ressalta-se que, por se tratar de dados públicos obtidos via *yfinance*, a base pode ser replicada e atualizada em pesquisas futuras, permitindo a aplicação do mesmo método em outros contextos ou períodos. A ilustração corresponde à instanciação da tabela cujo `CREATE TABLE` é apresentado na Listagem 4.4, do capítulo anterior.

Figura 2 – Exemplo da base de dados final com atributos originais e vetores.

| Ticker | Beta | Volatility | P/E Ratio | Dividend Yield | Market Cap | Revenue Growth | EPS Growth | Debt-to-Equity | Embedding |
|---|---|---|---|---|---|---|---|---|---|
| MMM | 1.098 | 0.2913 | 21.926 | 1.85 | 84086300672.0 | 0.014 | 7.2 | 316.755 | 0.5748,0.2907,0.0033,0.0369,0.0194,0.2061,0.0467,0.0456 |
| AOS | 1.250 | 0.2531 | 20.214 | 1.87 | 10169234432.0 | -0.013 | 3.59 | 18.035 | 0.6221, 0.2454, 0.0031, 0.0373, 0.0023, 0.1983, 0.0232, 0.0026 |
| ATEN | 1.363 | 0.3415 | 25.290 | 1.38 | 1259062912.0 | 0.155 | 0.69 | 112.670 | 0.6572, 0.3503, 0.0038, 0.0275, 0.0003, 0.2470, 0.0044, 0.0162 |

Fonte: autoria própria.

## 5.2 Consultas por similaridade

Nesta seção, apresentam-se os resultados das consultas por similaridade realizadas sobre a base de dados final, conforme descrito no método de trabalho. Serão comparados os resultados obtidos utilizando a distância Euclidiana com aqueles obtidos por meio da distância do cosseno, permitindo avaliar o impacto da escolha da métrica na identificação de ativos similares.

O dado utilizado nas consultas por similaridade está apresentado na Figura 3. Vale ressaltar que, nesta visualização, apenas os atributos originais dos ativos são exibidos, sem a normalização nem o vetor, que já foram apresentados anteriormente no exemplo da Listagem 4.6.

A primeira consulta realizada utilizou a distância euclidiana para determinar a similaridade entre os ativos, e os resultados estão apresentados na Figura 4. Foram retornados os 10 ativos mais próximos do ticker de consulta, ou seja, aqueles cujos padrões financeiros e vetores apresentam maior semelhança com o ativo base. Observa-se que o

Figura 3 – Dados da ação utilizada nas consultas por similaridade.

| Ticker | Beta | Volatility | P/E Ratio | Dividend Yield | Market Cap | Revenue Growth | EPS Growth | Debt-to-Equity |
|---|---|---|---|---|---|---|---|---|
| AOS | 1.250 | 0.2531 | 20.214 | 1.87 | 10169234432.0 | -0.013 | 3.59 | 18.035 |

Fonte: autoria própria.

primeiro ativo listado é justamente o ticker consultado, validando que os vetores gerados capturam adequadamente as características das ações e permitem realizar buscas coerentes e consistentes. O tempo de execução da consulta foi de `611 ms`, demonstrando a eficiência do procedimento.

Os ativos retornados e suas respectivas métricas financeiras são apresentados na Figura 4, que mostra os resultados da consulta de similaridade.

Figura 4 – Ações resultantes da consulta por similaridade por distância euclidiana

| | ticker [PK] character varying (10) | beta numeric (5,3) | volatility numeric (18,16) | pe_ratio numeric (10,4) | dividend_yield numeric (5,2) | market_cap numeric (15,2) | revenue_growth numeric (10,8) | eps_growth numeric (10,2) | debt_to_equity numeric (15,10) | distance double precision |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | AOS | 1.250 | 0.2531245555822983 | 20.2145 | 1.87 | 10169234432.00 | -0.01300000 | 3.59 | 18.0350000000 | 0 |
| 2 | CSX | 1.246 | 0.2614548489893645 | 19.9202 | 1.60 | 60533174272.00 | -0.03400000 | 1.63 | 159.5060000000 | 0.029638597207077465 |
| 3 | DOV | 1.263 | 0.2690391262142207 | 22.3200 | 1.20 | 23721439232.00 | 0.05200000 | 7.75 | 41.2450000000 | 0.040739349447429365 |
| 4 | UE | 1.288 | 0.2499651121619477 | 24.9059 | 3.59 | 2860194048.00 | 0.07100000 | 0.85 | 119.6790000000 | 0.04976911770587654 |
| 5 | XYL | 1.089 | 0.2573230470220723 | 36.8620 | 1.13 | 34454687744.00 | 0.06100000 | 3.84 | 18.7820000000 | 0.057053728442154054 |
| 6 | FAF | 1.283 | 0.2652168547575730 | 37.0934 | 3.26 | 6872518144.00 | 0.14200000 | 1.82 | 51.5270000000 | 0.057220528157510275 |
| 7 | PPG | 1.136 | 0.2897904325356399 | 20.0580 | 2.57 | 24989503488.00 | -0.00900000 | 5.52 | 106.1730000000 | 0.06068001229798583 |
| 8 | SU | 1.262 | 0.2862440681688629 | 12.7195 | 3.96 | 50638901248.00 | -0.07000000 | 3.28 | 32.0320000000 | 0.060801254314122595 |
| 9 | PNR | 1.085 | 0.2745398291624211 | 30.3142 | 0.90 | 18188144640.00 | 0.02200000 | 3.66 | 43.6550000000 | 0.06151217402941778 |
| 10 | WAB | 1.113 | 0.2767929152515097 | 28.1821 | 0.53 | 32279345152.00 | 0.02300000 | 6.70 | 47.0910000000 | 0.062394865838628875 |

Fonte: autoria própria.

A análise dos resultados da consulta de similaridade revela a eficácia da abordagem por vetores. Os ativos retornados apresentam valores de **Beta**, **Volatilidade** e **P/E Ratio** que são consistentemente próximos aos do ativo consultado. Isso confirma que o método de trabalho é capaz de capturar corretamente as características essenciais de risco e retorno dos ativos.

Em seguida, foi realizada uma nova consulta utilizando a similaridade de cosseno. Os resultados, apresentados na Figura 5, mostram um conjunto de ativos diferente do obtido com a distância euclidiana. Essa diferença ressalta a importância da métrica de similaridade na identificação de perfis de ativos, uma vez que a distância de cosseno foca na orientação do vetor (padrão dos dados), e não na magnitude. O tempo de execução da consulta foi de `212 ms`, evidenciando a eficiência do procedimento.

A distância Euclidiana, ao mensurar a proximidade geométrica entre os vetores, retornou ativos cujos valores financeiros e perfil de risco-retorno estão fortemente alinhados com o ativo de consulta. Como evidenciado pelos dados, os resultados incluem ativos com Beta, Volatilidade e P/E Ratio em faixas semelhantes, indicando que essa abordagem

Figura 5 – Resultados da consulta por similaridade utilizando a distância de cosseno.

| | ticker [PK] character varying (10) | beta numeric (5,3) | volatility numeric (18,16) | pe_ratio numeric (10,4) | dividend_yield numeric (5,2) | market_cap numeric (15,2) | revenue_growth numeric (10,8) | eps_growth numeric (10,2) | debt_to_equity numeric (15,10) | distance double precision |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | AOS | 1.250 | 0.2531245555822983 | 20.2145 | 1.87 | 10169234432.00 | -0.01300000 | 3.59 | 18.0350000000 | 0 |
| 2 | KFY | 1.492 | 0.2852054306134505 | 15.4115 | 2.50 | 3786821376.00 | 0.05000000 | 4.69 | 29.7050000000 | 0.0001550549482736141 |
| 3 | CSX | 1.246 | 0.2614548489893645 | 19.9202 | 1.60 | 60533174272.00 | -0.03400000 | 1.63 | 159.5060000000 | 0.0008979708512090179 |
| 4 | GVA | 1.414 | 0.2670496040481236 | 32.2673 | 0.48 | 4704826880.00 | 0.04000000 | 3.33 | 75.7750000000 | 0.0009233227369617181 |
| 5 | AFL | 0.839 | 0.2219453838793944 | 24.5788 | 2.13 | 58366107648.00 | -0.19000000 | 4.44 | 55.0550000000 | 0.0011644526506016106 |
| 6 | BAM | 1.713 | 0.3299560086097360 | 39.5594 | 3.09 | 91346403328.00 | 0.19000000 | 1.43 | 11.3330000000 | 0.0013400799418965548 |
| 7 | DOV | 1.263 | 0.2690391262142207 | 22.3200 | 1.20 | 23721439232.00 | 0.05200000 | 7.75 | 41.2450000000 | 0.0013781949478411937 |
| 8 | GGG | 1.035 | 0.2198062452927302 | 29.9433 | 1.30 | 13991201792.00 | 0.03400000 | 2.82 | 2.0540000000 | 0.0018198489213131497 |
| 9 | JOE | 1.352 | 0.2747457644022855 | 36.5493 | 1.08 | 3005435648.00 | 0.15700000 | 1.42 | 81.3040000000 | 0.0019040128751445229 |
| 10 | AME | 1.100 | 0.2300538309927190 | 30.5845 | 0.65 | 43865092096.00 | 0.02500000 | 6.21 | 21.1640000000 | 0.0019204281345445162 |

Fonte: autoria própria.

é adequada para identificar ações com características quantitativamente próximas. Dessa forma, valida-se que os vetores capturam de maneira eficaz a essência do perfil financeiro de cada ação.

Por outro lado, a similaridade de Cosseno, que considera apenas a orientação dos vetores e não sua magnitude, revelou um conjunto de ativos mais diversificado. Nesse caso, empresas como KFY e BAM, embora apresentem valores de Market Cap e Revenue Growth distintos do ativo consultado, compartilham padrões de vetores similares. Isso demonstra que a métrica de Cosseno é mais robusta para identificar ativos com comportamentos e tendências financeiras análogas, mesmo quando as escalas absolutas das variáveis diferem significativamente.

Em síntese, a distância Euclidiana é ideal para localizar ativos com perfis financeiros quantitativamente próximos, enquanto a similaridade de Cosseno permite detectar padrões comportamentais e relacionamentos subjacentes nos dados, independentemente da magnitude. A escolha da métrica dependerá, portanto, do objetivo da análise.

## 5.3 Clusterização das ações

Nesta seção são apresentados os resultados obtidos com o processo de clusterização. Para cada cluster, será detalhado o número de ações alocadas, bem como os respectivos tickers. Além disso, serão destacados os 10 tickers mais próximos do centróide de cada grupo, acompanhados de suas métricas financeiras. Também será realizada uma consulta que relaciona os tickers aos clusters e à sua volatilidade. Por fim, serão exibidos dois gráficos, construído a partir desses dados, com o objetivo de ilustrar a distribuição e variação da volatilidade entre os diferentes clusters.

A distribuição de ativos entre os clusters é apresentada na Tabela 1. É possível notar que o Cluster 3 possui um número significativamente menor de ativos, enquanto os Clusters 1 e 2 têm uma quantidade mais equilibrada entre si. A distribuição desigual dos ativos entre os clusters sugere que o algoritmo de agrupamento segmentou a base de dados de forma significativa. O Cluster 3, com um número reduzido de ativos, corresponde a

um nicho ou a um perfil de comportamento menos frequente. Por outro lado, os Clusters 1 e 2 agruparam a maioria das ações, representando os perfis de ativos predominantes na amostra. O processo de clusterização levou `623 ms` para ser concluído, totalizando `30 iterações`.

Tabela 1 – Distribuição de ativos por cluster.

| Cluster | Total de Ativos |
|---|---|
| 1 | 349 |
| 2 | 312 |
| 3 | 158 |

### 5.3.1 Resultados da clusterização

A seguir são apresentados os resultados obtidos a partir do processo de clusterização. Para cada cluster, listam-se os tickers que foram alocados em seus respectivos grupos. Esta organização permite observar como as ações foram distribuídas de acordo com suas características extraídas pelos vetores.

**Cluster 1:**

- AAT, ABM, ABT, ACM, ACP, ADC, ADM, AEE, AEG, AEP, AFG, AFL, AGM, AGO, AHH, AIG, AIZ, AJG, ALE, ALEX, ALLE, ALX, AM, AME, AMH, AMT, AMX, AOD, AON, AOS, APD, APLE, ATO, ATR, AVA, AVB, AVK, AVY, AWF, AWK, AWP, AWR, AXS, BCE, BFS, BG, BIP, BKE, BKH, BR, BRC, BRO, BRX, BTO, BUD, CB, CBT, CCK, CF, CHD, CHT, CIF, CL, CLX, CMI, CMS, CMU, CNA, CNI, CNO, CNP, CP, CPK, CPT, CSV, CSX, CUBE, CVX, CWT, CXE, CXH, D, DCI, DE, DGX, DHR, DKL, DLB, DLR, DMO, DOC, DRI, DTE, DUK, E, EARN, EBF, ECL, ED, EFC, EGP, EHI, EIG, ELS, EMD, EMF, ENB, EOG, EPD, EPR, EQR, ESNT, ESS, ET, ETR, EVTC, EXC, EXR, FAF, FCT, FDP, FDS, FE, FFC, FIS, FLC, FMS, FMX, FMY, FNF, FR, FRT, FSK, FT, G, GAB, GAM, GD, GDV, GGG, GGT, GIB, GPK, GSL, GTY, GUT, HD, HES, HIG, HIW, HIX, HNI, HON, HPF, HPI, HPS, HRL, HTD, IBM, ICE, IDA, IEX, IIM, INFY, INGR, IPG, IQI, ITW, IX, J, JEQ, JHI, JNJ, K, KIM, KMB, KMI, KO, KOF, KR, KRG, KYN, L, LADR, LDOS, LH, LIN, LND, LNN, LNT, LTC, LTM, LXP, MA, MAA, MAIN, MCR, MDT, MDU, MET, MFC, MFM, MHF, MKC, MLM, MMC, MMM, MMS, MMT, MPLX, MSA, MSD, MSI, MSM, MTG, MTN, NCV, NCZ, NEE, NEU, NFG, NGG, NHI, NI, NLY, NMFC, NNI, NNN, NOK, NVGS, NVS, NWE, NWN, O, OGE, OGS, OHI, OIA, OKE, OLP, OMC, ORA, ORI, PAA, PAG, PBA, PCM, PDT, PEG, PFD, PFE, PFG, PFO, PG, PGR, PHI, PKG, PMM, PMT, PNR, PNW, POR, PPG, PPL, PRI, PRU, PSA, PSO,

QGEN, QSR, R, RBA, RCI, RDN, RDY, REG, RIO, RJF, RLI, RMD, ROL, ROP, RPM, RS, RSG, RYN, SAP, SBI, SCHW, SCI, SGU, SHW, SKM, SLF, SNA, SNN, SNY, SO, SON, SPB, STAG, STC, STE, STN, STWD, SWX, SXT, SYK, SYY, T, TAK, TAP, TEI, THG, TJX, TMO, TNC, TR, TRI, TRNO, TRP, TRV, TSN, TT, TTC, TTE, TU, TYG, UAN, UDR, UE, UGI, UHT, UL, UMC, UMH, UNP, UPS, UTL, UVV, V, VALE, VCV, VGM, VICI, VKQ, VLT, VMC, VMO, VPV, VTN, VTR, VVR, VZ, WAB, WCN, WEA, WEC, WELL, WES, WH, WIA, WIT, WIW, WLKP, WM, WMB, WMT, WPC, WSO, WSR, WTS, WU, WY, XOM, XYL, ZBH, ZTS, ZWS

**Cluster 2:**

- A, AA, ABR, ACCO, ACN, AEO, AER, AES, AGCO, AIN, AIT, AKR, AL, ALG, ALSN, ALV, AMG, AMP, APA, APAM, APH, APO, ARES, ARMK, AROC, ASIX, ASX, ATEN, ATKR, AVT, AWI, AXP, AYI, AZZ, BAM, BBY, BC, BCC, BCO, BDC, BEN, BGC, BHE, BKR, BLK, BTE, BWA, BX, BXP, BYD, CAT, CBL, CCJ, CCS, CIM, CLB, CLDT, CMC, CMRE, CNK, CNQ, CNS, CPA, CR, CRH, CRI, CRM, CRS, CSL, CUZ, CVE, CW, CX, DAC, DAL, DAN, DD, DDS, DEI, DHI, DIS, DKS, DLX, DOV, DRH, DVN, EE, EFX, EGY, EME, EMN, EMR, ENS, ERJ, ESE, ESI, ESRT, ETN, EVR, EXP, F, FCX, FIX, FLS, FSS, FTI, FUL, GBX, GE, GEF, GES, GFF, GIL, GLP, GLW, GM, GOLF, GPN, GS, GTN, GVA, H, HAL, HBM, HCI, HEI, HL, HOG, HST, HVT, HXL, HY, IBP, IR, ITT, IVZ, JBL, JCI, JOE, KAI, KBH, KEP, KFY, KKR, KMPR, KMT, KNX, KOP, KRC, KRO, KSS, LAD, LAZ, LEA, LII, LNC, LPG, LPX, LUV, LVS, LXFR, LZB, M, MAS, MATX, MC, MCO, MCS, MCY, MFA, MGA, MITT, MLI, MLR, MOS, MOV, MPC, MS, MT, MTDR, MTH, MTRN, MTX, MUR, MWA, MYE, NGVC, NKE, NOA, NOAH, NOV, NPO, NRG, NSC, NUE, NUS, NVDA, OC, OEC, OPY, ORCL, OSK, OXM, OXY, PFSI, PH, PHM, PIPR, PKX, PLD, PLOW, PSX, PUK, PVH, PWR, RCL, RES, REXR, RHI, RHP, RL, RLJ, RM, ROK, SAH, SB, SCL, SCS, SEE, SEM, SF, SHO, SIG, SITC, SKT, SLB, SM, SNX, SPG, SPLP, SQM, SSD, SSTK, ST, STM, SU, SVM, SWK, SXC, SXI, TECK, TEL, TEX, TFII, TFX, TGLS, TGT, THO, TKR, TOL, TPL, TPR, TPVG, TRGP, TRN, TRTX, TRU, TS, TSM, TX, TXT, UHS, UI, URI, USPH, UTZ, UWMC, VAC, VET, VFC, VHI, VLO, VMI, VNO, VNT, VOYA, VRT, VST, WCC, WD, WHG, WLY, WLYB, WMS, WSM, WST, WT, WWW, XHR, YRD, ZIM

**Cluster 3:**

- ABEV, AEM, AGI, AGRO, AGX, ALL, AQN, ARCO, ASC, ASR, ATHM, AU, AZN, B, BABA, BAH, BBW, BDX, BEKE, BMY, BP, BRFS, BTU, BVN, CAG,

CCU, CHE, CI, CIG, COP, CPAC, CPB, CTRA, CTS, CVS, DEO, DG, DHT, DRD, DSX, EBR, EC, EIX, ELP, ENR, EQT, FINV, FMC, FRO, GFI, GGB, GHC, GIS, GNE, GPC, GPI, GPRK, GSK, HAFN, HCC, HII, HMC, HMN, HMY, HRB, HSY, HUM, IDT, IRS, KBR, KGC, KNOP, KT, LEG, LLY, LMT, LYB, MRK, MUSA, NAT, NC, NEM, NGS, NL, NMR, NOC, NPK, NRP, NSP, NVO, ODC, PAC, PAYC, PBR, PCG, PHG, PKE, PX, RGA, RNR, RRC, SAIC, SBS, SD, SFL, SMP, SPH, SRE, STNG, TFPM, TGNA, TGS, TIMB, TKC, TLK, TM, TME, TNET, TNK, TPB, TUYA, TXO, UGP, UNF, UNH, UNM, UVE, VIPS, VIV, VTS, WDH, WDS, WMK, WPM, WPP, WTM, XYF, YUMC, ZTO

A seguir, são apresentados os **10 ativos mais próximos de cada centróide**, acompanhados dos respectivos valores de seus atributos financeiros. Esses ativos representam os perfis mais característicos de cada grupo, permitindo compreender como as variáveis extraídas pelos vetores financeiros influenciam a formação dos clusters.

Para o **Cluster 1**, o centróide foi representado pelo seguinte vetor:

[0.4939017, 0.2101436, 0.004067465, 0.08843735, 0.00906825, 0.21773182, 0.028049031, 0.017030442]

A Figura 6 apresenta os resultados da consulta, destacando os 10 ativos mais próximos desse centróide.

Figura 6 – Ativos mais próximos do centróide do Cluster 1.

| | ticker character varying (10) | beta numeric (5,3) | volatility numeric (18,16) | pe_ratio numeric (10,4) | dividend_yield numeric (5,2) | market_cap numeric (15,2) | revenue_growth numeric (10,8) | eps_growth numeric (10,2) | debt_to_equity numeric (15,10) |
|---|---|---|---|---|---|---|---|---|---|
| 1 | SLF | 0.822 | 0.2031305869554378 | 14.5528 | 4.30 | 33240762368.00 | 0.05600000 | 4.07 | 52.8270000000 |
| 2 | CPT | 0.816 | 0.2221576740388745 | 76.5804 | 3.84 | 11701472256.00 | 0.02000000 | 1.43 | 82.1540000000 |
| 3 | MAA | 0.779 | 0.2112471559763809 | 29.2593 | 4.26 | 17066985472.00 | 0.00600000 | 4.86 | 83.5910000000 |
| 4 | UDR | 0.881 | 0.2296966588220667 | 98.3846 | 4.48 | 14478803968.00 | 0.02300000 | 0.39 | 139.0580000000 |
| 5 | AVB | 0.880 | 0.2285430651253262 | 23.9742 | 3.59 | 27786043392.00 | 0.04400000 | 8.14 | 72.5750000000 |
| 6 | RCI | 0.893 | 0.2359183309433873 | 17.8060 | 4.05 | 19440697344.00 | 0.02400000 | 2.01 | 254.1280000000 |
| 7 | PSA | 0.849 | 0.2338149248420231 | 31.8431 | 4.11 | 51288420352.00 | 0.01600000 | 9.18 | 110.1720000000 |
| 8 | EQR | 0.940 | 0.2374338539976904 | 24.9774 | 4.17 | 26148259840.00 | 0.04700000 | 2.66 | 75.2590000000 |
| 9 | AMH | 0.761 | 0.2209492768923482 | 30.7477 | 3.52 | 14412655616.00 | 0.08000000 | 1.11 | 65.9570000000 |
| 10 | PBA | 0.883 | 0.2131588546381601 | 18.3411 | 5.27 | 22811197440.00 | -0.03400000 | 2.14 | 78.4960000000 |

Fonte: autoria própria.

Para o **Cluster 2**, o centróide foi representado pelo seguinte vetor:

[0.63988817, 0.42821836, 0.010006647, 0.053782493, 0.009893341, 0.21423356, 0.033266906, 0.020337787]

A Figura 7 apresenta os 10 ativos mais próximos desse centróide, representando aqueles que sintetizam de forma mais fiel o perfil do cluster.

Figura 7 – Ativos mais próximos do centróide do Cluster 2.

| | ticker character varying (10) | beta numeric (5,3) | volatility numeric (18,16) | pe_ratio numeric (10,4) | dividend_yield numeric (5,2) | market_cap numeric (15,2) | revenue_growth numeric (10,8) | eps_growth numeric (10,2) | debt_to_equity numeric (15,10) |
|---|---|---|---|---|---|---|---|---|---|
| 1 | THO | 1.336 | 0.4142423193775131 | 25.1823 | 1.90 | 5551206912.00 | 0.03300000 | 4.17 | 25.1050000000 |
| 2 | FLS | 1.305 | 0.3833881120041298 | 25.4570 | 1.49 | 7357795328.00 | 0.02700000 | 2.21 | 73.1820000000 |
| 3 | NOV | 1.348 | 0.4270693102613654 | 10.6529 | 2.33 | 4786894848.00 | -0.01300000 | 1.21 | 36.1540000000 |
| 4 | MTH | 1.356 | 0.3945673377612252 | 8.9840 | 2.18 | 5606388736.00 | -0.04000000 | 8.77 | 35.7530000000 |
| 5 | BC | 1.351 | 0.4367060720560523 | 74.1818 | 2.63 | 4264794624.00 | 0.00200000 | 0.88 | 126.8400000000 |
| 6 | EE | 1.373 | 0.4061792985478501 | 17.9478 | 1.33 | 2742228992.00 | 0.11600000 | 1.34 | 69.1320000000 |
| 7 | WMS | 1.264 | 0.4204745323948395 | 25.4937 | 0.51 | 11001216000.00 | 0.01800000 | 5.55 | 84.6150000000 |
| 8 | LUV | 1.189 | 0.4113331534942727 | 49.4688 | 2.27 | 16627451904.00 | -0.01500000 | 0.64 | 66.7630000000 |
| 9 | TFII | 1.324 | 0.4295386800198831 | 21.5737 | 1.92 | 7774669312.00 | -0.10000000 | 4.34 | 112.8310000000 |
| 10 | DVN | 1.138 | 0.4111432371068197 | 7.8311 | 2.76 | 22071996416.00 | -0.00700000 | 4.44 | 59.2660000000 |

Fonte: autoria própria.

Por fim, para o **Cluster 3**, o centróide foi representado pelo seguinte vetor:

[0.35581234, 0.3452347, 0.003148306, 0.075106405, 0.009159541, 0.22692718, 0.04370214, 0.022874542]

A Figura 8 ilustra os 10 ativos mais próximos desse centróide, evidenciando os elementos que melhor caracterizam este agrupamento.

Figura 8 – Ativos mais próximos do centróide do Cluster 3.

| | ticker character varying (10) | beta numeric (5,3) | volatility numeric (18,16) | pe_ratio numeric (10,4) | dividend_yield numeric (5,2) | market_cap numeric (15,2) | revenue_growth numeric (10,8) | eps_growth numeric (10,2) | debt_to_equity numeric (15,10) |
|---|---|---|---|---|---|---|---|---|---|
| 1 | HMC | 0.321 | 0.3481407183361230 | 11.4437 | 4.24 | 43867967488.00 | -0.01200000 | 2.93 | 98.4230000000 |
| 2 | DG | 0.292 | 0.3480791544675801 | 19.3333 | 2.26 | 22979065856.00 | 0.05100000 | 5.40 | 213.0260000000 |
| 3 | UNM | 0.385 | 0.3052103639896957 | 9.0216 | 2.45 | 12814349312.00 | 0.04000000 | 8.34 | 35.5790000000 |
| 4 | ELP | 0.368 | 0.3131447092699385 | 15.2787 | 4.54 | 6616696320.00 | 0.13600000 | 0.61 | 79.0870000000 |
| 5 | PKE | 0.460 | 0.3445207414532907 | 55.0882 | 2.67 | 371899136.00 | 0.10200000 | 0.34 | 0.3320000000 |
| 6 | B | 0.428 | 0.3309967533718618 | 18.2579 | 2.07 | 49591078912.00 | 0.16400000 | 1.59 | 14.1490000000 |
| 7 | BDX | 0.297 | 0.2997143297902042 | 34.1572 | 2.23 | 53553389568.00 | 0.10400000 | 5.47 | 75.9300000000 |
| 8 | SAIC | 0.490 | 0.3544775507269203 | 12.9307 | 1.39 | 4897660928.00 | -0.02700000 | 8.23 | 161.1730000000 |
| 9 | VIV | 0.383 | 0.2876744176708978 | 18.8939 | 3.01 | 19988162560.00 | 0.07100000 | 0.66 | 29.8550000000 |
| 10 | BMY | 0.357 | 0.2934294604903122 | 18.5542 | 5.37 | 94037327872.00 | 0.00600000 | 2.49 | 291.5490000000 |

Fonte: autoria própria.

### 5.3.2 Análise da Volatilidade por Cluster

Para analisar a relação entre a volatilidade e os clusters obtidos no processo de *clustering*, foi realizada uma consulta SQL com o objetivo de extrair os *tickers*, suas respectivas volatilidades e o cluster ao qual pertencem. A consulta permitiu organizar os dados de forma que fosse possível comparar a distribuição da volatilidade entre os diferentes clusters identificados pelo algoritmo K-Means. O tempo de execução da consulta foi de `142 ms`, reforçando a eficiência do processo.

O código da consulta SQL é apresentado na Listagem 5.1.

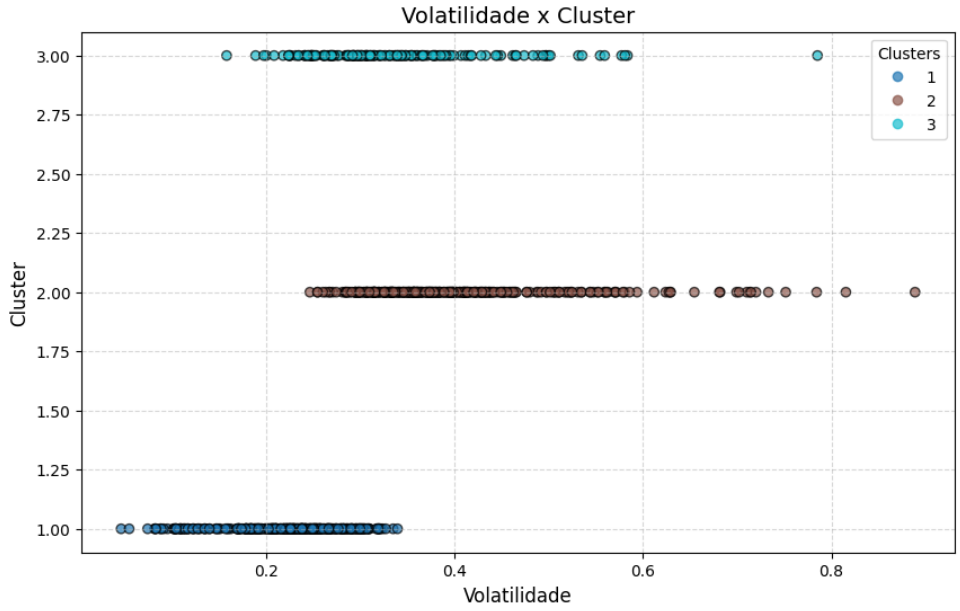
```
SELECT
    s.ticker,
```

```
    s.volatility,
    c.cluster_id
FROM clustering c
JOIN stock_data s ON s.ticker = c.ticker
WHERE c.iteration = (SELECT MAX(iteration) FROM clustering)
ORDER BY c.cluster_id;
```

Listing 5.1 – Consulta SQL para extrair volatilidade e cluster dos tickers

Com os dados resultantes, foram gerados dois gráficos que permitem compreender a distribuição da volatilidade entre os clusters. O primeiro gráfico, apresentado na Figura 9, mostra a relação direta entre a volatilidade dos ativos e os clusters atribuídos.

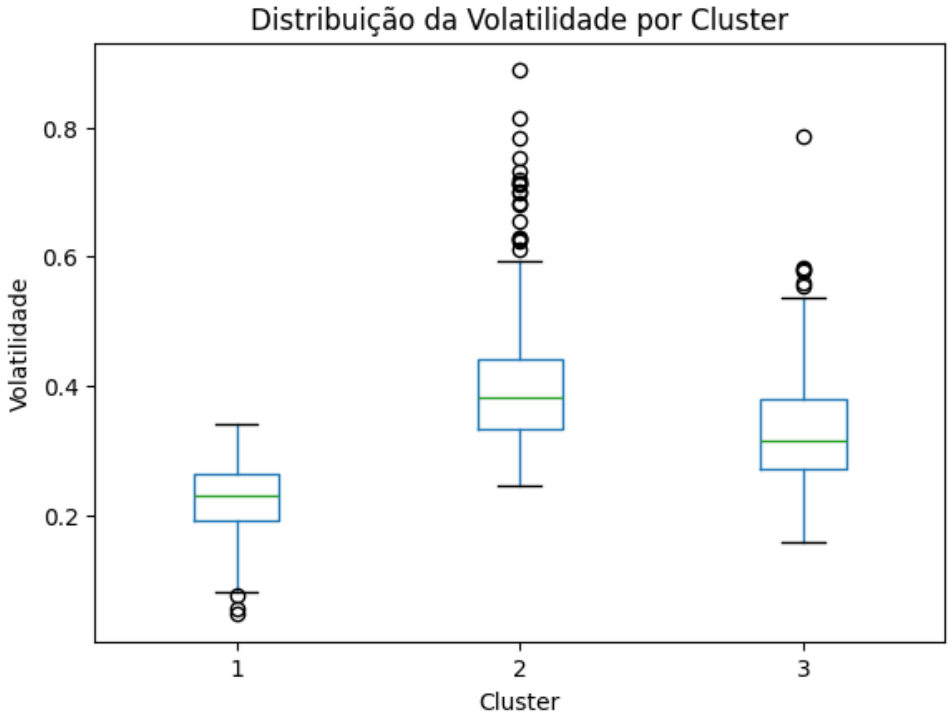
Figura 9 – Distribuição da volatilidade dos ativos em relação aos clusters.




Fonte: autoria própria.

O segundo gráfico, ilustrado na Figura 10, apresenta um *boxplot* que evidencia a dispersão e a mediana da volatilidade em cada cluster, permitindo identificar a presença de assimetrias e *outliers*.

Figura 10 – Distribuição da volatilidade por cluster.




Fonte: autoria própria.

A análise dos gráficos (Figuras 9 e 10) permite destacar os seguintes pontos:

- O **Cluster 1** concentra ativos com baixa volatilidade, distribuídos em um intervalo estreito, o que indica menor risco associado.
- O **Cluster 2** apresenta a maior dispersão, com diversos *outliers*, sugerindo a presença de ativos mais voláteis e heterogêneos.
- O **Cluster 3** possui ativos de volatilidade intermediária, com distribuição mais equilibrada, indicando um perfil de risco moderado.

Esses resultados reforçam que o processo de *clustering* foi capaz de segmentar os ativos em grupos distintos em relação ao nível de risco (volatilidade), o que pode auxiliar na formulação de estratégias de investimento diversificadas.

# 6 Conclusão

O presente trabalho teve como objetivo desenvolver e implementar um método de trabalho de armazenamento e clusterização de dados financeiros utilizando a extensão `pgvector` do PostgreSQL. A partir dos resultados obtidos, foi possível observar que a abordagem proposta permitiu integrar de forma eficiente os dados históricos da NYSE com técnicas de análise de clusters, proporcionando informações relevantes sobre padrões de comportamento das ações.

Para facilitar a replicação e continuidade do estudo, foi criado um repositório público no GitHub contendo todo o código, scripts SQL e exemplos utilizados neste trabalho (ROYER, 2025)[1].

## 6.1 Avaliação dos Resultados

Os experimentos realizados demonstraram que o método implementado é capaz de gerar clusters coerentes com a volatilidade e características dos ativos analisados. A utilização de procedimentos armazenados para executar o algoritmo de K-Means sobre os vetores possibilitou a execução direta no SGBD, reduzindo significativamente a necessidade de manipulação externa e aumentando a eficiência do processo. Além disso, os gráficos e análises de distribuição dos clusters confirmaram a capacidade do modelo em identificar grupos de ativos com comportamentos semelhantes.

## 6.2 Pontos Positivos

Dentre os principais aspectos positivos do método de trabalho, destacam-se:

- **Integração direta dos dados com o modelo de clusterização:** todo o processamento é realizado dentro do SGBD, sem necessidade de exportação ou importação para ferramentas externas.
- **Escalabilidade da solução:** aproveita a eficiência do PostgreSQL e da extensão `pgvector`, permitindo o processamento de grandes volumes de dados.
- **Reprodutibilidade:** a metodologia pode ser replicada de forma simples utilizando apenas scripts SQL e procedimentos armazenados, garantindo consistência nos experimentos.

[1] <https://github.com/layllaroyer/TCC-Final>

## 6.3 Limitações

Apesar dos resultados promissores, algumas limitações foram identificadas:

- **Dependência da qualidade dos vetores:** a eficácia do cluster depende diretamente da representação vetorial dos dados, podendo afetar a separação dos clusters caso os vetores não capturem adequadamente as características dos ativos.
- **Escolha do número de clusters ($k$):** a determinação de $k$ é um desafio clássico em métodos de clusterização e pode impactar a interpretação dos resultados.
- **Possibilidade de clusters vazios ou desbalanceados:** dependendo da distribuição dos dados, alguns clusters podem apresentar poucos elementos ou concentrar grande parte dos ativos, dificultando a análise.

## 6.4 Trabalhos Futuros

Como perspectivas para trabalhos futuros, destacam-se:

- **Otimização e Comparação de Vetores:** Explorar e comparar diferentes modelos de vetores (como modelos pré-treinados em dados financeiros) para verificar como a qualidade do vetor afeta o agrupamento.
- **Análise de Agrupamento Dinâmico:** Implementar um processo automatizado para a determinação do número ideal de clusters ($k$) em diferentes janelas de tempo, permitindo que a solução se adapte a mudanças no mercado.
- **Análise de Outliers:** Aprofundar a análise dos ativos que se encontram em clusters desbalanceados ou que são considerados *outliers*, investigando se esses ativos representam oportunidades de investimento ou comportamentos anômalos.
- **Avaliação de Desempenho do Processamento no SGBD:** Comparar a eficiência de executar o processo de análise inteiramente dentro do SGBD com a abordagem que exporta os dados para ferramentas externas, avaliando tempo de execução, consumo de recursos e escalabilidade.
- **Implementação do Algoritmo DBSCAN:** Desenvolver e avaliar a implementação do algoritmo *DBSCAN* diretamente no PostgreSQL com suporte à extensão `pgvector`, possibilitando a detecção de clusters de densidade variável e a identificação automática de ruídos e outliers sem a necessidade de definir previamente o número de grupos.

Em síntese, o trabalho demonstrou que é viável realizar clusterização diretamente no SGBD utilizando `pgvector`, proporcionando uma solução eficiente, escalável e reprodutível, embora ainda existam desafios a serem explorados e aprimorados em pesquisas futuras.

# Referências

AHMED, M.; SERAJ, R.; ISLAM, S. M. S. The k-means algorithm: A comprehensive survey and performance evaluation. **Electronics**, MDPI, v. 9, n. 8, p. 1295, 2020. Citado 2 vezes nas páginas 24 e 25.

Alpha Vantage. **API Documentation**. <https://www.alphavantage.co/documentation/>. Acesso em: 27 set. 2025. Citado na página 19.

Amazon Web Services. **Structured vs. Unstructured Data: What's the Difference?** 2025. Acesso em: 27 set. 2025. Disponível em: <https://aws.amazon.com/pt/compare/the-difference-between-structured-data-and-unstructured-data/>. Citado na página 14.

AROUSSI, R. **yfinance documentation**. <https://ranaroussi.github.io/yfinance/>. Acesso em: dia mês 2025. Citado na página 19.

_____. **yfinance: Download market data from Yahoo! Finance API**. 2025. PyPI package. Disponível em: <https://pypi.org/project/yfinance/>. Citado na página 11.

ASSAF, A. **Mercado Financeiro**. 13. ed. São Paulo: Atlas, 2019. Citado na página 11.

B3. **Cotações Históricas**. <https://www.b3.com.br/pt_br/market-data-e-indices/servicos-de-dados/market-data/historico/mercado-a-vista/cotacoes-historicas/>. Acesso em: 27 set. 2025. Citado na página 19.

Bloomberg. **Enterprise Data Catalog**. <https://www.bloomberg.com/professional/support/api-library/>. Acesso em: 27 set. 2025. Citado na página 19.

CAMPISTA, M. M. **A DISRUPÇÃO NA INDÚSTRIA BRASILEIRA DE SERVIÇOS FINANCEIROS PELA DEMOCRATIZAÇÃO DO ACESSO A INVESTIMENTOS**. Tese (Doutorado) — Universidade Federal do Rio de Janeiro, 2020. Citado na página 12.

CARDOSO, F.; MALSKA, J.; RAMIRO, P.; LUCCA, G.; BORGES, E.; MATTOS, V.; BERRI, R. Bovdb: A data set of stock quotes for machine learning on all companies from b3 between 1995 and 2020. In: **Anais do III Dataset Showcase Workshop**. Porto Alegre, RS, Brasil: SBC, 2021. p. 21–32. ISSN 0000-0000. Disponível em: <https://sol.sbc.org.br/index.php/dsw/article/view/17411>. Citado na página 27.

CONTRIBUTORS, W. **Lists of companies listed on the New York Stock Exchange**. 2025. <https://en.wikipedia.org/wiki/Lists_of_companies_listed_on_the_New_York_Stock_Exchange>. Último acesso em: 2025-09-07. Citado na página 32.

CORONEL, C.; MORRIS, S. **Database Systems: Design, Implementation, & Management**. 12. ed. Boston: Cengage Learning, 2015. Citado na página 20.

Elastic. **O que são dados não estruturados?** 2025. Acesso em: 27 set. 2025. Disponível em: <https://www.elastic.co/pt/what-is/unstructured-data>. Citado 2 vezes nas páginas 14 e 15.

FIGUEIREDO, C. C. D. **ANALYZING STOCK MARKET DATA USING UNSUPERVISED LEARNING TECHNIQUES**. Dissertação (Mestrado) — Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2024. Trabalho de Conclusão de Curso — Graduação. Disponível em: <https://pantheon.ufrj.br/handle/11422/26176>. Citado 7 vezes nas páginas 12, 20, 23, 29, 31, 42 e 47.

IBM. **Structured vs. Unstructured Data: What's the Difference?** 2025. Acesso em: 27 set. 2025. Disponível em: <https://www.ibm.com/br-pt/think/topics/structured-vs-unstructured-data>. Citado na página 15.

JAIN, A. K. Data clustering: 50 years beyond k-means. **Pattern Recognition Letters**, v. 31, n. 8, p. 651–666, 2010. ISSN 0167-8655. Award winning papers from the 19th International Conference on Pattern Recognition (ICPR). Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167865509002323>. Citado na página 25.

KRASKA, T.; BEUTEL, A.; CHI, E. H.; DEAN, J.; POLYZOTIS, N. The case for learned index structures. In: **Proceedings of the 2018 International Conference on Management of Data**. New York, NY, USA: Association for Computing Machinery, 2018. (SIGMOD '18), p. 489–504. ISBN 9781450347037. Disponível em: <https://doi.org/10.1145/3183713.3196909>. Citado na página 20.

MANZONI, L. **B3: A Bolsa de Valores do Brasil**. 2025. Acesso em: 14 mar. 2025. Disponível em: <https://br.investing.com/academy/trading/b3-a-bolsa-de-valores-do-brasil/>. Citado na página 11.

Microsoft. **Como otimizar o desempenho ao usar pgvector**. 2024. Acessado em: 22 mar. 2025. Disponível em: <https://learn.microsoft.com/pt-br/azure/cosmos-db/postgresql/howto-optimize-performance-pgvector>. Citado na página 22.

MINMAXSCALER — scikit-learn documentation. <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html>. Acesso em: dia mês 2025. Citado na página 18.

MOURA, P. H. B. d. Sistema eletrônico de recomendação agnóstico e online de aplicação em fundos de investimentos. Universidade Presbiteriana Mackenzie, 2021. Citado 2 vezes nas páginas 12 e 27.

NASCIMENTO, T.; MIRANDA, L. Chronos ações: Ferramenta para apoiar a tomada de decisão de investidores da bolsa de valores. In: **Anais do XI Simpósio Brasileiro de Sistemas de Informação**. Porto Alegre, RS, Brasil: SBC, 2015. p. 533–540. ISSN 0000-0000. Disponível em: <https://sol.sbc.org.br/index.php/sbsi/article/view/5858>. Citado na página 27.

New York Stock Exchange. **The History of NYSE**. 2025. <https://www.nyse.com/history-of-nyse>. Acesso em: dia mês 2025. Citado na página 11.

PAN, J. J.; WANG, J.; LI, G. Survey of vector database management systems. **The VLDB Journal**, v. 33, n. 5, p. 1591–1615, 2024. Disponível em: <https://doi.org/10.1007/s00778-024-00864-x>. Citado 5 vezes nas páginas 15, 16, 20, 21 e 28.

PostgreSQL Global Development Group. **pgVector 0.7.0 Released**. 2025. Accessed: 2025-03-13. Disponível em: <https://www.postgresql.org/about/news/pgvector-070-released-2852/>. Citado 2 vezes nas páginas 11 e 22.

PREPROCESSING data — scikit-learn documentation. <https://scikit-learn.org/stable/modules/preprocessing.html>. Acesso em: dia mês 2025. Citado na página 18.

Refinitiv. **Eikon**. <https://www.refinitiv.com/en/products/eikon-trading-software>. Acesso em: 27 set. 2025. Citado na página 19.

ROYER, L. **TCC-Final**. 2025. <https://github.com/layllaroyer/TCC-Final>. Acesso em: 24 set. 2025. Citado na página 57.

SANTOS, T.; COSTA, O. Sistema de tomada de decisão no mercado de ações utilizando aprendizado de máquina. In: **Anais do II Brazilian Workshop on Artificial Intelligence in Finance**. Porto Alegre, RS, Brasil: SBC, 2023. p. 25–36. ISSN 0000-0000. Disponível em: <https://sol.sbc.org.br/index.php/bwaif/article/view/24950>. Citado na página 12.

_____. Sistema de tomada de decisão no mercado de ações utilizando aprendizado de máquina. In: **Anais do II Brazilian Workshop on Artificial Intelligence in Finance**. Porto Alegre, RS, Brasil: SBC, 2023. p. 25–36. ISSN 0000-0000. Disponível em: <https://sol.sbc.org.br/index.php/bwaif/article/view/24950>. Citado na página 28.

SOUSA, V. G. Estudo de caso do uso de um sistema de gerenciamento de bancos de dados vetoriais para a manipulação, análise e recuperação de áudio. Universidade Federal de Uberlândia, 2025. Citado na página 28.

TAIPALUS, T. Vector database management systems: Fundamental concepts, use-cases, and current challenges. **Cognitive Systems Research**, v. 85, p. 101216, 2024. ISSN 1389-0417. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1389041724000093>. Citado 4 vezes nas páginas 15, 16, 20 e 22.

WINNICKI, M. J.; BROWN, C. A.; PORTER, H. L.; GILES, C. B.; WREN, J. D. Biovdb: biological vector database for high-throughput gene expression meta-analysis. **Frontiers in Artificial Intelligence**, v. 7, 2024. ISSN 2624-8212. Disponível em: <https://www.frontiersin.org/journals/artificial-intelligence/articles/10.3389/frai.2024.1366273>. Citado na página 28.

XU, R.; WUNSCH, D. Survey of clustering algorithms. **Neural Networks, IEEE Transactions on**, v. 16, p. 645 – 678, 06 2005. Citado na página 24.

Yahoo Finance. **Yahoo Finance**. 2025. <https://finance.yahoo.com/>. Acesso em: 22 set. 2025. Citado na página 11.

ZHANG, Y.; JIN, R.; ZHOU, Z.-H. Clustering financial data for financial studies. **arXiv preprint arXiv:1609.08520**, 2016. Disponível em: <https://arxiv.org/abs/1609.08520>. Citado 3 vezes nas páginas 17, 18 e 29.