LEARNING MADE EASY

2nd Neo4j Special Edition

Graph Data Science (GDS)

for dummies®
A Wiley Brand

Understand graphs and graph data science

Explore your graph opportunities

Use Neo4j as your graph data platform

Brought to you by:

neo4j

Dr. Alicia Frame

Zach Blumenfeld

About Neo4j, Inc.

Neo4j is the leader in graph technology. Neo4j helps brands — including Comcast, eBay, NASA, UBS, and Volvo — reveal and predict how people, processes, and systems are interrelated. With this relationships-first approach, graph technology from Neo4j tackles connected data challenges like fraud detection, recommendation engines, entity resolution, and even supply chain optimization. By using real-time predictions and recommendations, Neo4j powers real-time recommendations and predictions through its graph-native analytics, machine learning models, and data pipeline integrations. Find out more at **neo4j.com.**

About the Authors

Dr. Alicia Frame is a data scientist who runs Product at Neo4j Graph Data Science, focusing on building tools that help users bring connections to their analytics practice. **Zach Blumenfeld** is a data scientist focused on helping other data scientists, engineers, and business leaders implement graph analytics and machine learning to solve challenging problems.

Graph Data Science

2nd Neo4j Special Edition

by Dr. Alicia Frame and Zach Blumenfeld

for dummies®
A Wiley Brand

Graph Data Science For Dummies®, 2nd Neo4j Special Edition

Published by
John Wiley & Sons, Inc.
111 River St.
Hoboken, NJ 07030-5774
www.wiley.com

For general information on our other products and services, or how to create a custom *For Dummies* book for your business or organization, please contact our Business Development Department in the U.S. at 877-409-4177, contact info@dummies.biz, or visit www.wiley.com/go/custompub. For information about licensing the *For Dummies* brand for products or services, contact BrandedRights&Licenses@Wiley.com.

ISBN: 978-1-119-90932-3 (pbk); ISBN: 978-1-119-90933-0 (ebk)

Publisher's Acknowledgments

Some of the people who helped bring this book to market include the following:

Project Manager: Carrie Burchfield-Leighton

Sr. Managing Editor: Rev Mengle

Acquisitions Editor: Ashley Coffey

Business Development Representative: Molly Daugherty

Table of Contents

Introduction

Connectivity is the single most pervasive characteristic of today's networks and systems. From protein interactions to social networks, from communication systems to power grids, and from retail experiences to supply chains, networks with even a modest degree of complexity aren't random, which means connections are neither evenly distributed nor static. Simple statistical analysis alone fails to sufficiently describe, let alone predict, behaviors within connected systems.

As the world becomes increasingly interconnected and systems increasingly complex, using technologies built to leverage relationships and their dynamic characteristics is imperative. Not surprisingly, interest in graph data science and graph analytics has exploded because they were explicitly developed to gain insights from connected data. Graph data science and graph analytics reveal the workings of intricate systems and networks at massive scale.

About This Book

We are passionate about the utility and importance of graph data science and graph analytics, so we wrote this book to help organizations better leverage graphs so they can make new discoveries and develop intelligent solutions faster.

In this book, we focus on the commercial applications of graph analysis and graph-enhanced machine learning (ML), which takes the form of graph data science. We also use Neo4j graph technology to illustrate a graph data science platform. You take a quick look at graph data science and its uses before covering the journey of graph data science adoption. You also review Neo4j technology as a graph data science platform and walk through a fraud detection example.

Icons Used in This Book

The following icons are used in this book:

REMEMBER

Information here can be filed away for later use.

TECHNICAL STUFF

This information may not be critical to most people, but if you like the extra techie tidbits, you'll enjoy the insight here. Otherwise, just skip over it!

TIP

Are you interested in saving time or effort on your projects? Check out these tips to help you do just that.

Beyond the Book

This book is focused on graph data science and relies on graph theory, graph analytics, and graph databases. If you want resources beyond what we can offer you in this short book, we recommend the following:

- `neo4j.com/intro-GDS`: For an introduction to graph data science, this video provides an overview and entry-level context for getting started.
- `neo4j.com/knowledge-graph-report`: In this book, you find out what knowledge graphs are, how to build and use them, and their importance in AI and graph data science.
- `neo4j.com/gds_for_dummies`: If you're new to graph databases, *Graph Databases For Dummies,* Neo4j Special Edition, is a great place to start your journey because it assumes no previous experience and walks you through modeling, querying, and importing graph data, all the way through your first production system.

IN THIS CHAPTER

» Defining a graph

» Understanding graph analytics and graph data science

» Using questions to explore graph data science

Chapter 1 Understanding Graphs and Graph Data Science

Graph approaches to data are exploding in the commercial world to better reveal meaning in data as well as forecast behavior of complex systems. This burst is due to the increasing connectedness of data, breakthroughs in scaling graph technology to enterprise-sized problems, excellent results when integrated with machine learning (ML) and artificial intelligence (AI) solutions, and more accessible tools for general analytics and data science teams.

In this chapter, you discover how we define a graph and the relationship of graphs to analytics and data science. You also get a foundation in how graphs are used to answer tough questions about complex systems.

Explaining What a Graph Is

Networks are a representation, a tool to understand complex systems and the complex connections inherent in today's data. For example, you can represent how a social system works by thinking about interactions between pairs of people. By analyzing the structure of this representation, you can answer questions and make predictions about how the system works or how individuals behave within it. In this sense, network science is a set of technical tools applicable to nearly any domain, and graphs are the mathematical models used to perform analysis. Simply put, graphs are a mathematical representation of complex systems.

Graphs have a history dating back to 1736. The origins of graph theory hail from the city of Königsberg, which included two large islands connected to each other and the two mainland portions of the city by seven bridges. The puzzle was to create a walk through the city, crossing each bridge once and only once. Leonhard Euler solved that puzzle by asking whether it was possible to visit all four areas of a city connected by seven bridges, while only crossing each bridge once. It wasn't.

With the insight that only the connections themselves were relevant to solving this kind of problem, Euler established the groundwork for graph theory and its mathematics. As one of Euler's original sketches, Figure 1-1 depicts Euler's progression:

- **Walking the bridges of Königsberg:** Four main areas of Königsberg with seven bridges. Can you cross each bridge only once and return to your starting point?
- **Euler's insight:** The only relevant data is the main areas and the bridges *connecting* them.
- **Origins of graph theory:** Euler abstracted the problem and created generalized rules based on nodes and relationships that apply to any connected system.

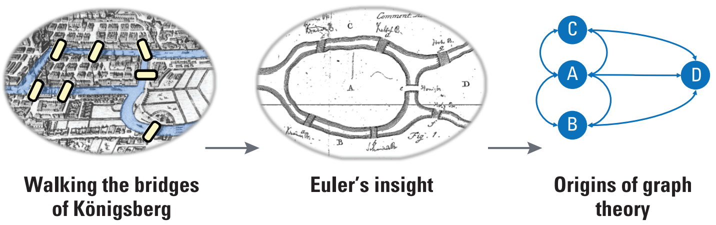


FIGURE 1-1: The origins of graph theory.

REMEMBER

While graphs originated in mathematics, they are also a pragmatic and faithful representation of data for modeling and analysis. A graph is a representation of a network, often illustrated with circles to represent entities, also called *nodes* or *vertices*, and lines between them. Those lines are known as *relationships*, *links*, or *edges*. Think of nodes as the nouns in sentences, and relationships as verbs that give context to the nodes. To avoid any confusion, the graphs we talk about in this book have nothing to do with graphing equations or charts. Take a look at the differences in Figure 1-2.

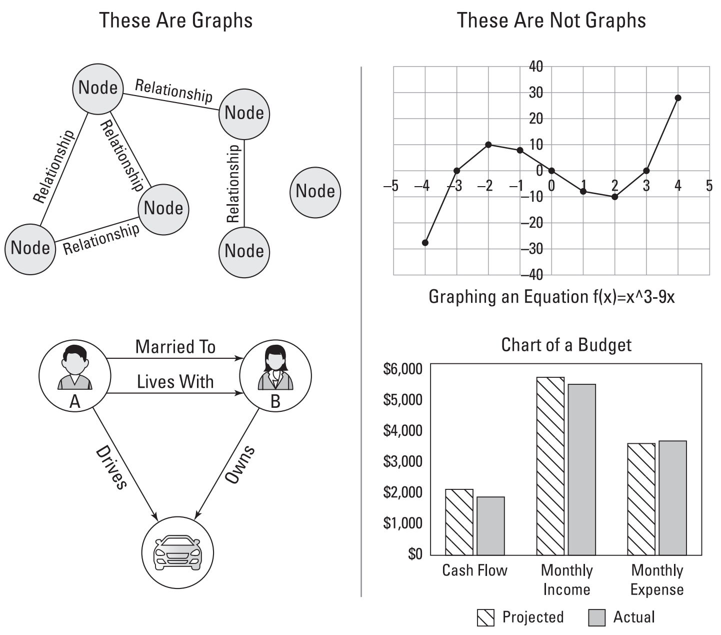


FIGURE 1-2: A graph is a representation of a network.

The bottom graph on the left in Figure 1-2 is a person graph. When looking at that graph, you can construct several sentences to describe it. For example, person A lives with person B who owns a car, and person A drives a car that person B owns. This modeling approach maps easily to the real world and is whiteboard-friendly, which helps align data modeling and analysis.

TECHNICAL STUFF

We often use the phrase "whiteboard-friendly" for anything that's easy to describe with simple drawings that you could illustrate on a whiteboard.

Defining Graph Analytics and Graph Data Science

Modeling graphs is only half of the story. You may also want to analyze them to reveal insights that aren't immediately obvious.

Graph data science is a science-driven approach to gain knowledge from the relationships and structures in data, typically to power predictions. It uses multi-disciplinary workflows that may include queries, statistics, algorithms, and ML. Graph data science can typically be broken down into three areas:

- **Graph statistics** provide basic measures about a graph, such as the number of nodes and distribution of relationships. These insights may influence how you configure and execute more complex analysis as well as interpret results.
- **Graph analytics** build on graph statistics by answering specific questions and gaining insights from connections in existing or historical data. Graph queries and algorithms are typically applied together in "recipes" during graph analytics, and the results are used directly for analysis.
- **Graph-enhanced ML and AI** is the application of graph data and analytics results to train ML models or support probabilistic decisions within an AI system.

Graph statistics and analytics are often used in conjunction to answer certain types of questions about complex systems, and the subsequent insights are then applied to improve ML.

Looking at the Types of Questions for Graph Data Science

REMEMBER

Data scientists try to tackle many types of questions when using graph data science to evaluate interdependencies, infer meaning, and predict behavior. At the most abstract level, these questions fall into a few broad areas: movement, influence, groups and interactions, and patterns, as shown in Figure 1-3.

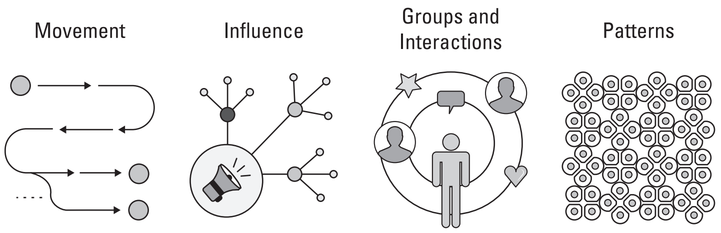


FIGURE 1-3: Graph data science questions fall into four different areas.

The areas in Figure 1-3 answer the following questions:

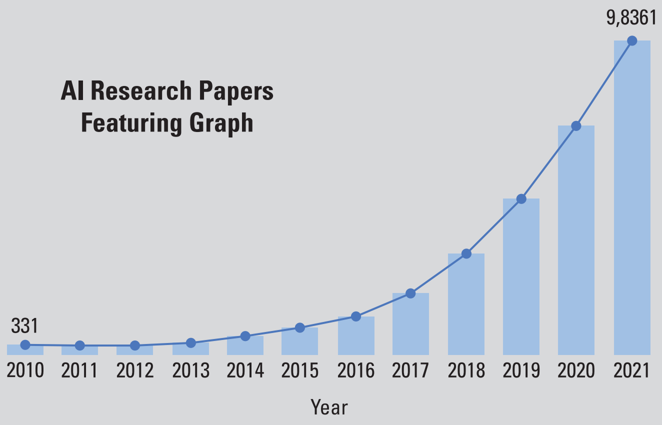
- **How do things travel (move) through a network?** Understanding how things move through a network involves deep path analysis to find propagation pathways, such as the route of diseases or network failures. It can also be used to optimize for the best possible route or for flow constraints.
- **What are the most influential points?** Identifying influencers involves uncovering the structurally well-placed nodes that represent the control points in a network. These influencers can act as fast dissemination points, bridges between less connected groups, or bottlenecks. Influencers can accelerate or slow the flow of items through networks from finances to opinions. The concept of highly connected and influential nodes in a graph is referred to as *centrality.* Centrality algorithms are essential for understanding influence in a network.
- **What are the groups and interactions?** Detecting communities requires grouping and partitioning nodes based on the number and strength of interactions. This method is the primary way to presume group affinity, although neighbor likeness can also be a factor. Link prediction is about inferring future (or unseen) connections based on network structure. Heuristic Link Prediction algorithms are often used to predict behavior. In addition to Community Detection algorithms, Similarity algorithms are also used to understand groupings.
- **What patterns are significant?** Uncovering network patterns reveals similarities and can also be used for general exploration.

  For example, you may look for a known relationship pattern between a few nodes or compare attributes of all your nodes to find similarities. Or perhaps you want to evaluate the entire structure of a network, with its intricate hierarchies, to correlate patterns to certain social behavior to investigate. Aggregating related but ambiguous information in large datasets is a common activity that relies on finding similar and related information. Finding patterns may employ simple queries or various types of algorithms found in Chapter 3.

Multiple types of graphs queries and algorithms are usually applied in a recipe fashion as part of a graph data science workflow. For example, a query to understand the density of relationships in a graph may help determine the appropriate Community Detection algorithm for the most relevant results. Tactically, graph queries and algorithms are the tools for understanding the overall nature of a connected system and for using relationships in various data science pipelines.

THE RISE OF GRAPH DATA SCIENCE

The rise of graph data science is the result of more accessible technologies, increased ability to compute over massive graph datasets, and an awareness of the power of graphs to infer meaning and improve forecasts. Researchers play an essential role in developing awareness and advocating for the best techniques. As data scientists see the potency of structural information, they're increasingly incorporating graphs into their statistics, analytics, and ML practices. In fact, according to the Dimensions Knowledge system for research publications, the use of graph technology in AI research is accelerating. In the last ten years, the number of AI research papers that feature graph technology has increased over 700 percent. For a visual representation, check out this sidebar's figure.



IN THIS CHAPTER

» Seeing how graphs help the health industry

» Using graphs in marketing

» Putting graphs to work to prevent fraud

Chapter 2 Using Graph Data Science in the Real World

Today's most pressing data challenges center around connections, not just tabulating discrete data. The ability for graph data science to uncover and leverage network structure drives a range of use cases from fraud prevention and targeted recommendations to personalized experiences and drug repurposing.

REMEMBER

We can't overstate the impact of improved graph techniques by way of new algorithms or the efforts of applied network science, such as within computational biology. We don't want you to overlook societal projects that use graphs either. However, we believe that the recent explosion of graphs in the business world represents a shift in accessibility and opportunity to drive a democratization of graphs for everyone.

Graph technologies help organizations with many practical use cases across industries and domains. In the past, many businesses began exploring graph technology to create a 360-degree view of their customers or to unify master data, including customer, product, supplier, and logistics information. They may use this kind of tracking to improve customer experience or to meet compliance regulations of recent privacy acts, such as the EU's General Data Protection Regulation (GDPR) and the California Consumer Privacy Act (CCPA). This same kind of complete view and data lineage in graphs is also now used to understand and track data used in machine learning (ML) for more responsible artificial intelligence (AI) applications.

Today, businesses are just as likely to look at using graphs specifically for data science as they recognize the predictive power of relationships, the ability to use network structures to improve their ML, and their own need to innovate. The sections in this chapter highlight a few graph data science use cases in areas of accelerating growth and significant commercial interest.

Looking at Graphs in the Health Industry

It's easy to see how any industry with biological roots would naturally comprehend the importance of interconnected systems. In computational biology, as well as healthcare and life sciences, challenges are part of a larger process.

For serving health and commercial interests, two examples stand out: better patient outcomes and more efficient drug discovery. Check out how Novartis uses graphs to capture the latest biological knowledge for drug discovery at `neo4j.com/casestudy/novartis`.

Discovering more efficient drugs

Safety, speed, and costs are paramount in making new drug solutions accessible. Graphs can help tackle the complexity of intertwined relationships between diseases, genes, drugs, side effects, and demographics — to name just a few considerations.

One impressive knowledge graph in the life sciences industry integrates over 50 years of biomedical data that includes genes, compounds, diseases, and other information such as symptoms and side effects. One of the projects from the graph predicts new uses for drugs by using the graph topology. The graph helps predict new uses for currently approved drugs by evaluating relationships, network structures, and similarities. Drug repurposing significantly reduces costs and time to market compared to developing and testing new drugs — not to mention the benefit of having more real-world information available about side effects and unexpected results when a drug is already in use.

Improving the patient journey

Another area of emerging interest is the use of graphs for mapping, evaluating, and improving patient journeys. When a patient doesn't feel well, many factors are in play that may have evolved over a period of time. Likewise, treatments are rarely a single event, especially for chronic or serious illnesses. The tree of possible symptoms, visits, tests, care givers, treatment plans, outcomes, and then secondary tests and treatments, and so on can branch out into an immense number of possible paths. Imagine the patient treatment options that can be mapped with a graph to better see the sequence alternatives and path splits after each and every test result or visit. In fact, researchers and healthcare providers already use graphs to better understand what influences patient journeys to then improve individual outcomes. They're also using graphs to create and compare optimal paths for patients.

Recommendations and Personalized Marketing

Making relevant product and service recommendations requires correlating product, customer information, historic behavior, inventory, supplier, logistics, and even social sentiment data. Graph-powered recommendations and targeted marketing help companies provide more appropriate services and experiences to a wider range of users. For example, graph Community Detection graph algorithms are used to group customers with interactions or similar behavior for more relevant recommendations. Research shows that graph-enhanced ML can predict customer churn, for example, for uses such as targeted prevention or marketing.

Graph analytics are also used to help target offers to online users that are anonymous in name and demographics but not in site behavior. Insights from analysis performed offline are typically rolled into decision models used in production for real-time recommendations, which can include recommendations for products that ship faster based on shifting stock levels or instantly incorporating data from the customer's current visit. As an example, visit `neo4j.com/usecase/meredith-corp` to see how Meredith Corporation uses graph data science for recommendations.

Fraud Detection

The amount of money lost to fraud each year is growing, despite increased use of AI and ML to detect and prevent it. To uncover more fraud while avoiding costly false positives, organizations look beyond individual data points to the connections and patterns that link them. Organizations use the network structure to augment existing ML pipelines as a practical approach to increase the amount of fraud detected and recovered.

Graph feature engineering allows businesses to extract predictive elements based on graph queries or algorithms and use that information to train ML models. Improving the predictive accuracy in fraud detection by even small percentage points can result in tens of millions of dollars saved in just a few months. Graph data science enables companies to stay ahead of the ever-shifting patterns of fraud as well as recover more losses.

For more information on how graph data science can help you detect fraud, read this white paper: `neo4j.com/gds_fraud_whitepaper`. Also, head to Chapter 5 where we give you a detailed example of detecting fraud with graph data science.

IN THIS CHAPTER

» Bringing together diverse information

» Using graph analytics to understand your network

» Finding, combining, and extracting predictive elements

» Simplifying graphs with embedding

» Taking a new approach with graph networks

Chapter 3 Evolving Your Use of Graph Data Science Technology

Today, graph data science is usually applied in business with one or more major aims in mind: better decisions, increased quality of predictions, and creating new ways to innovate and learn. These goals are increasingly tied to tangible benefits, such as reduced financial loss, faster time to results, increased customer satisfaction, and predictive lift. You may be trying to improve or automate decision-making by people and domain experts that need additional context. Or perhaps your goal is to improve predictive accuracy by using relationships and network structure in analytics and machine learning (ML).

Graphs provide a unique structure for learning that helps evolve ML techniques through better abstraction and interpretability. These business goals strongly map to how organizations integrate graph technology into their data science practices. Figure 3-1 diagrams the major phases of a typical graph data science journey.

Most users start with knowledge graphs and progress as they become more proficient and advanced in their use of graph data science. We cover each phase in this chapter.

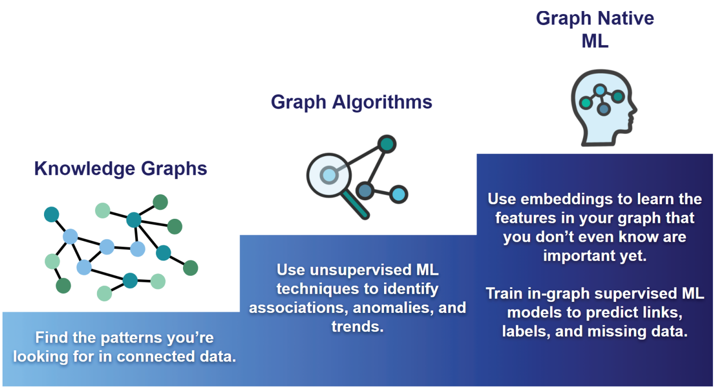

Graph Native ML
Graph Algorithms
Knowledge Graphs
Use embeddings to learn the features in your graph that you don't even know are important yet.
Train in-graph supervised ML models to predict links, labels, and missing data.
Use unsupervised ML techniques to identify associations, anomalies, and trends.
Find the patterns you're looking for in connected data.

FIGURE 3-1: The graph data science journey.

REMEMBER

Your organization can use practical steps to gain immediate value and then layer more sophisticated techniques in a way that continually increases return on your effort.

Knowledge Graphs

Knowledge graphs are the foundation of graph data science and offer a way to streamline workflows, automate responses, and scale intelligent decisions. At a high level, knowledge graphs are interlinked sets of data points and describe real-world entities, facts, or things and their relationship with each other in a human-understandable form. Unlike a simple knowledge base with flat structures and static content, a knowledge graph acquires and integrates adjacent information by using data relationships to derive new knowledge.

As the first phase in graph data science, knowledge graphs are often implemented to bring together diverse information to help domain experts find related content as well as explore the connections in their data. Knowledge graphs can also add context to applications, such as those in artificial intelligence (AI) systems, so they can make better and faster approximating decisions. This approach is used in AI systems, such as chatbots, that use a knowledge graph, for example, to better route a request for a "bat for my husband's birthday." In this case, the graph grasps that the request isn't most likely a flying mammal someone is looking for but instead sporting goods of higher quality for a special occasion. The chatbot can also take into account what's in stock, shipping times, and specialty products combining the context of not only the requestor but also of supply and other logistics.

After you have a knowledge graph, you can start to use queries to find the patterns you know are important. Graph queries are used when you know exactly what you're looking for, such as asking a question like "How many relationships does Mia have?" or "How many fraudsters or flagged accounts are four hops away?" (A *hop* is a connection, or a layer of relationship.) These kinds of queries seem simple because we can imagine standing up and looking at things that are close to us. However, solutions that don't store relationships alongside their data must perform extra processes to look up and join this related information. Graphs store relationships together with data so following the path of relationships is simple and fast. Graph-native databases are particularly good at multiple hop queries. They avoid expensive index lookups and data joins by storing and processing related information adjacently, treating relationships as first-class citizens.

Graph Algorithms

After implementing a knowledge graph (see the preceding section), businesses often start using graph algorithms to understand their networks better and answer specific questions based on relationships and topology. You're often trying to infer meaning based on the network structure: finding clusters, identifying influential nodes, evaluating different pathways. Graph algorithms look at entire graphs for offline analysis of historical data. This process is in contrast to small, real-time transactions and local queries that focus on small areas around a few nodes. Graph algorithms can be used for graph analytics or graph feature engineering, when they inform a predictive model.

Graph algorithms originated from network science to enable reasoning about structure in a more unsupervised fashion. They're used when you know the pattern or indicator that you're looking for but don't know exactly what you'll find. For example, you may be looking for unusually tight communities where nodes have more relationships between each other than you'd expect in a random or normal distribution. To find these communities, you could use the Louvain Modularity algorithm to uncover clusters with higher interaction densities inside, among group members when compared to interactions outside of the group.

The graph algorithms most prevalent in commercial applications fall into roughly five categories:

- **Pathfinding and search:** These algorithms are foundational to graph analytics and explore paths between nodes. They evaluate routes for uses, such as physical logistics and least-cost call or Internet protocol (IP) routing.
- **Centrality (importance):** Centrality algorithms help you uncover the roles of individual nodes and their impact. They identify influential nodes based on their position in the network. These algorithms infer group dynamics, such as credibility, rippling vulnerability, and bridges between groups.
- **Community Detection:** These algorithms find communities where members have more significant interactions. These connections reveal tight clusters, isolated groups, and structures. This information helps predict similar behavior or preferences, estimate resilience, find duplicate entities, or simply prepare data for other analyses.
- **Similarity:** These algorithms employ set comparisons to look at how alike individual nodes are. The properties and attributes of nodes are used to score the likeness between nodes. This approach is used in applications, such as personalized recommendations and developing categorical hierarchies.
- **Topological Link Prediction:** These algorithms consider the proximity of nodes in the network as well as structural elements, such as possible triangles between nodes, to estimate the likelihood of a new relationship forming or that undocumented connections exist. This class of algorithms has many applications from drug repurposing to criminal investigations.

REMEMBER

In graph analytics, you're either asking a targeted question or looking at the graph as a whole to infer meaning or make predictions about future behavior. Graph algorithms are often initially used for graph analytics, which is asking questions about graph topology that can be directly answered with an unsupervised algorithm. For example, they can ask "Which node in my graph is the most important?" or "What's the shortest route between two nodes" or "How is my data clustered?" However, more advanced use cases leverage graph algorithms in predictive models.

Graph feature engineering is the process of finding, combining, and extracting predictive elements from raw graph data to be used in ML tasks. More information generally makes ML models more accurate, but data scientists rarely have as much data as they'd like. Because relationships are extremely predictive of behavior and they inherently exist inside current data, you can employ graph feature engineering to improve predictions and increase ML model accuracy — with the data you already have.

Graph algorithms are one of the simplest ways to translate your connected data into new, more meaningful features. For example, you could calculate the PageRank — a Centrality score — of each node in your graph, or label nodes based on the communities they belong to. These scores and labels can then be extracted to a list or table of numbers and identifiers (also called a *feature vector*) for training ML models. The graph features and resulting ML metrics are often written back to the graph database for persistence and future use.

Figure 3-2 shows how the use of graph features to enhance ML fits into a larger workflow.

With graph-enhanced ML, you'd typically aggregate, explore, and cleanse data and then use graph queries or algorithms for feature engineering. Then you'd prepare the data for ML and split it into training and testing datasets. Although this process isn't completely linear, after you've trained a model and are happy with the results, the model can then be used in production. Although the model may feed a real-time transaction in production, such as approving credit applications online, the graph feature engineering and ML are done offline and periodically updated in a cyclical process. Graph feature engineering offers organizations attainable model improvements without needing to change their ML pipelines.

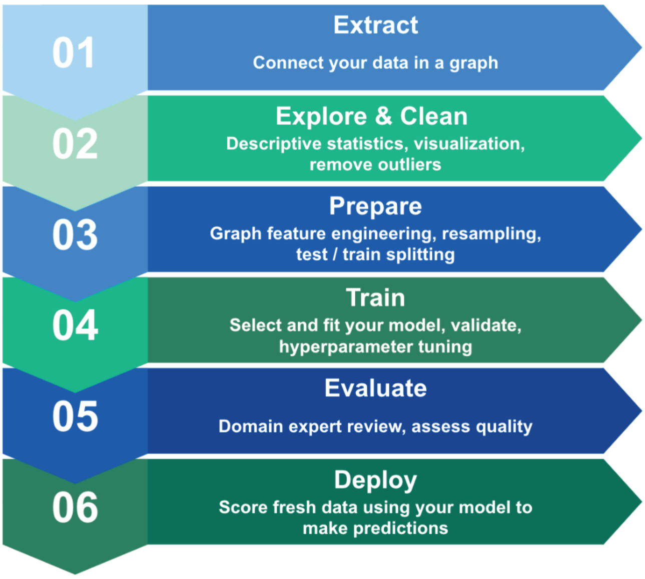

01
Extract
Connect your data in a graph
02
Explore & Clean
Descriptive statistics, visualization, remove outliers
03
Prepare
Graph feature engineering, resampling, test / train splitting
04
Train
Select and fit your model, validate, hyperparameter tuning
05
Evaluate
Domain expert review, assess quality
06
Deploy
Score fresh data using your model to make predictions

FIGURE 3-2: Graph feature engineering is part of a larger ML workflow.

Graph-Native Machine Learning

Graph-native ML is the most advanced type of graph data science. In this case, you're using your graph as the input to your model and making predictions about how your graph will evolve in the future. Unlike graph algorithms, you don't predefine the features of the graph that you think are important (like Centrality scores or community membership). Instead, you use your entire graph as an input and train a predictive model to predict any missing data — or how your graph may change in the future.

Most commonly, for graph-native ML, you need to convert your whole graph into a structure that's compatible with ML techniques. *Graph embeddings* are used to simplify graphs or subsets of graphs into a feature vector, or set of vectors, that are in a lower dimensional form, such as a list of numbers. The goal is to create easily consumable data for tasks like ML that still describe more intricate topology, connectivity, or nodes attributes. For example, you can represent an entire graph or a path as an embedding and then learn based on the graph or paths themselves.

There are three types of graph embeddings:

- **Node embeddings** describe connectivity of each node.
- **Path embeddings** encompass the traversals across a graph.
- **Whole graph embeddings** encode an entire graph into a single vector.

Different types of embeddings are more appropriate for different use cases. Node embeddings are the most widely used and most flexible for different use cases.

REMEMBER

Graph embeddings are often used for more advanced feature engineering that incorporates more complex information, which is why this phase typically comes later in the graph data science journey. Embeddings can also be useful for data exploration, computing similarity between entities, and reducing dimensionality to aid in statistical analysis. Graph embeddings offer the ability to more widely use the rich structures that make up graphs in various data science tasks and learn based on nuanced information.

Graph-native ML is an exciting area of research that represents a new approach to ML that may drastically improve results with less data, make predictions more explainable, and lead to new types of learning itself. Most commonly, models are trained to predict new or missing links — or relationships — that will form in the graph or to predict new or missing labels for nodes. For example, you could predict new relationships between fraudsters and new victims to prevent fraud before it happens, or you could predict missing labels for the customers that are most likely to churn.

Graph-network and *graph-native learning* are terms coined by Peter Battaglia and a group of researchers. They concluded that using graphs for ML was the next major advancement in ML itself because of the graph's ability to abstract topology. Their thinking follows this approach:

1. **Graph-native learning takes a graph as an input, performs learning computations while preserving transient states, and then returns a graph.**
2. **This graph-native learning process allows the domain expert to review and validate the learning path that leads to more explainable predictions.**

3. **With this process comes richer and more accurate predictions that use less data and training cycles.**

REMEMBER

Graph-native learning enables whole-graph learning and multi-task predictions that reduce data requirements and automate the identification of relevant features. Today, the valuable time of data scientists and domain experts is frequently employed to tediously select and test potentially predictive data and collect those features into optimal models. Improving the model accuracy while streamlining the process positively impacts ML processes and results across all applications. We're excited by early progress, and we look forward to seeing ML evolve to be extremely efficient and flexible as well as more accurate and transparent.

IN THIS CHAPTER

» Running algorithms with Neo4j Graph Data Science

» Supporting multiple databases with the Neo4j DBMS

» Looking at Neo4j Desktop and Neo4j Browser

» Finding patterns with Neo4j Bloom

Chapter 4 Using Neo4j as a Graph Data Science Platform

If you're going to use graph data science, you should run it on a workspace that integrates with your existing data science pipeline. This chapter shows you what Neo4j offers to help you be successful in using graphs for data science and analysis. Neo4j is a graph technology company that provides an enterprise-grade graph data science software platform.

REMEMBER

Neo4j supports transactional processing and analytical processing of graph data as well as visualization. It also includes graph storage and compute with data management and analytics tooling. The set of integrated tools includes a common protocol, application programming interface (API), and query language (Cypher) to provide effective access for different uses. It's available on-premises, self-hosted, and fully managed in the cloud.

In this chapter, we cover each of the four areas of the Neo4j platform in a bit more detail to help you see how your graph data science solution fits together.

Neo4j Graph Data Science

Neo4j Graph Data Science offers an enterprise-ready approach to running sophisticated graph algorithms on connected data at scale. Graph analytics and feature engineering add highly predictive relationships to your machine learning (ML) for better results. Algorithms are executed in an analytics workspace that scales computations to handle graphs that contain hundreds of billions of nodes and relationships. For examples, training, and details on how to use Neo4j Graph Data Science, visit `neo4j.com/gds-library-manual`. You can also learn more by going directly to Neo4j Graph Data Science at `neo4j.com/product/gds`.

Neo4j Graph Database Management System

The Neo4j Database Management System (DBMS) supports multiple databases that can run in standalone or clustered installations and supports sharding and federated access to databases. The Neo4j graph database is designed to treat the relationships between data as important as the data itself. It's considered a

CYPHER DECLARATIVE QUERY LANGUAGE

Cypher is the most widely adopted, fully defined, and open query language for property graph databases. It is a declarative, SQL-inspired language for describing visual patterns in graphs by using ASCII-Art syntax. You can state what you want to select, insert, update, or delete from your graph data without describing how to do it. Cypher is intended to be readable. For example, the phrase, "Jennifer likes graph technology," would be written as

```
(p:Person {name: "Jennifer"})-[rel:LIKES]-
   >(g:Technology {type: "Graphs"})
```

Cypher basics and learning resources can be found on the Cypher page for Neo4j developers at `neo4j.com/cypher-documentation`.

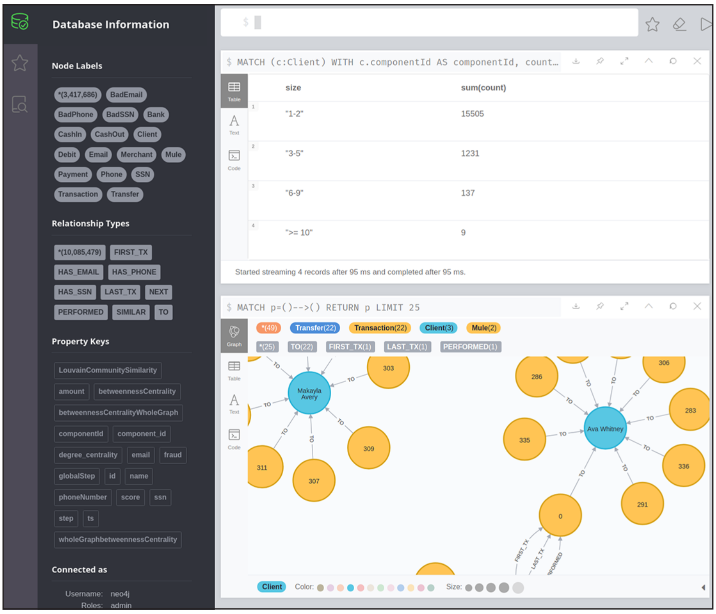
native-graph database because the data is stored together with how each individual entity connects with or is related to others. You can find more information about the Neo4j Graph DBMS at `neo4j.com/gds-graph-database`.

TIP

To discover more about the property graph model that's used by the DBMS and other tools, check out *Graph Databases For Dummies*, Neo4j Special Edition, at `neo4j.com/gds_for_dummies`.

Neo4j Desktop and Browser

Neo4j Desktop is a user interface for operating local databases. Neo4j Browser is a general-purpose user interface for working with the Neo4j graph database and is a core component of Neo4j Desktop. Developers and data scientists can use this tool to query, visualize, administer, and monitor their databases. Figure 4-1 shows the Neo4j Browser being used against a fraud graph.


Database Information
Node Labels
*(3,417,686) BadEmail BadPhone BadSSN Bank CashIn CashOut Client Debit Email Merchant Mule Payment Phone SSN Transaction Transfer
Relationship Types
*(10,085,479) FIRST_TX HAS_EMAIL HAS_PHONE HAS_SSN LAST_TX NEXT PERFORMED SIMILAR TO
Property Keys
LouvainCommunitySimilarity amount betweennessCentrality betweennessCentralityWholeGraph componentId component_id degree_centrality email fraud globalStep id name phoneNumber score ssn step ts wholeGraphbetweennessCentrality
Connected as
Username: neo4j
Roles: admin
*(49) Transfer(22) Transaction(22) Client(3) Mule(2)
*(25) TO(22) FIRST_TX(1) LAST_TX(1) PERFORMED(1)
Makayla Avery
Ava Whitney
Client Color: Size:

```
$ MATCH (c:Client) WITH c.componentId AS componentId, count…
```

	size	sum(count)
1	"1-2"	15505
2	"3-5"	1231
3	"6-9"	137
4	">= 10"	9

```
Started streaming 4 records after 95 ms and completed after 95 ms.
```

```
$ MATCH p=()-->() RETURN p LIMIT 25
```

FIGURE 4-1: The Neo4j Browser interface interacts with Neo4j databases.

Neo4j Bloom

Neo4j Bloom is a graph visualization and exploration tool that allows you to find patterns in a Neo4j graph by using a codeless search paradigm. It uses an interactive point-and-click interface to expand and refine results, find interesting paths, and share insights with others.

Bloom is intended for ad hoc, visual explorations, and fast prototyping with type-ahead search suggestions and direct editing of nodes and relationships. The visual presentation has flexible color, size, and icon schemes to help differentiate influential items with styling that can be based on the results of running algorithms from Neo4j Graph Data Science (see the earlier section in this chapter titled "Neo4j Graph Data Science").

Figure 4-2 shows the Bloom interface for an example of restaurant reviews that can be exported and shared.

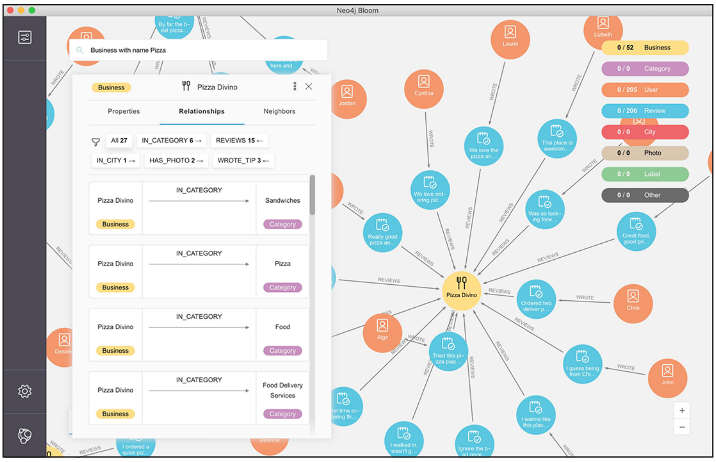


FIGURE 4-2: A visual exploration of Neo4j Bloom graphs with code-free searching.

IN THIS CHAPTER

» Preparing a good dataset

» Exploring suspicious groups

» Predicting fraud by using graph features

Chapter 5 Detecting Fraud with Graph Data Science

In this chapter, we walk you through an example of applying graph data science techniques to investigate and predict financial fraud. After we familiarize you with a sample financial transaction dataset, we then remove the outlier information that may skew your results and identify suspicious clusters of clients. After that, you visually explore one of the clusters for graph-based indicators of fraud and look at how graph-based features help predict fraudulent behavior in the larger dataset.

Finding a Good Fraud Dataset

To simulate a good fraud dataset, you want to create realistic, synthetic data to describe fraudulent transactions. In this section, we give you a model of a finance network, where users make transactions with merchants and each other via mobile devices. This has similar patterns to traditional credit card networks more common in the United States, Canada, and Europe. Figure 5-1 is a graph example that uses a subset of available nodes and relationships from data that we modified with additional identifiers.

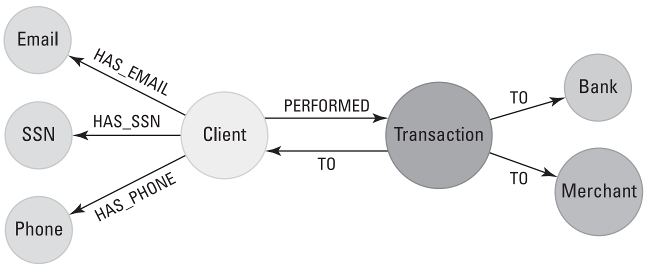


FIGURE 5-1: The fraud dataset.

This example uses the following node labels:

- **Clients:** People who have personally identifiable information (PII) such as Social Security Numbers (SSNs), phone numbers, and email addresses
- **Mules:** Clients known to have fraudulently transferred money
- **Clients' PII:** SSNs; Phone: Phone numbers; Email: Email addresses

These nodes are connected by the following relationship types:

- (Client)-[:HAS_PHONE]->(Phone)
- (Client)-[:HAS_SSN]->(SSN)
- (Client)-[:HAS_EMAIL]->(Email)

REMEMBER

The analysis performed here is focused on the above information, but the dataset also contains additional information, such as transactions performed to banks, merchants, and clients.

Removing Outliers

An important first step when performing fraud analysis is to check the quality of the data. In fraud datasets, you may have outliers that aren't relevant for your analysis. *Outliers* are rare events or items that raise suspicions by being significantly different from most of the data. Outliers in a graph are based on their connectivity and the topology of the graph instead of a property value.

TIP

The Degree Centrality algorithm measures the number of relationships that a node has. Running the Degree Centrality algorithm is therefore a good way of finding potential outliers.

Because you should expect different types of nodes to have different connectivity, you need to project a graph that selects the right nodes and relationships so you can properly identify outliers within the context of your data. For example, each bank should receive many deposits (so it would have a high-degree of centrality), but an individual account holder should receive far fewer. Therefore, you need to compare banks against banks and accounts against accounts.

To find potential outliers, you can run the Degree Centrality algorithm against the fraud dataset with the following queries:

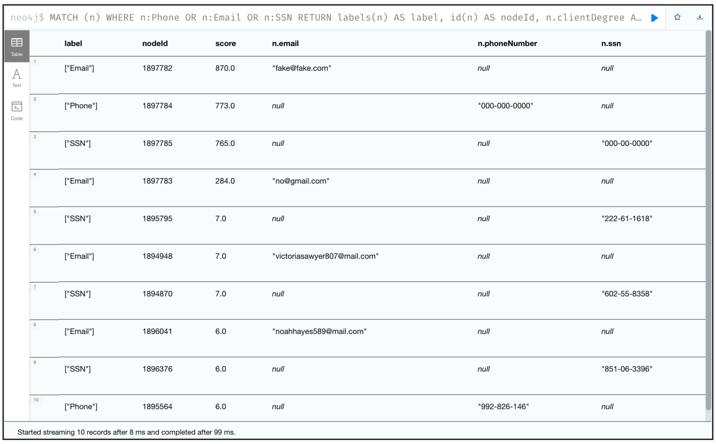
```
CALL gds.graph.project('proj',
  ['Client','Phone','Email','SSN'],
  {
   HAS_PHONE: {orientation: 'REVERSE'},
   HAS_EMAIL: {orientation: 'REVERSE'},
   HAS_SSN: {orientation: 'REVERSE'}
  }
);

CALL gds.degree.mutate('proj', {
  mutateProperty:'clientDegree'
});

CALL gds.graph.writeNodeProperties(
  'proj', ['clientDegree'],
   ['Phone','Email','SSN']
);

MATCH (n) WHERE n:Phone OR n:Email OR n:SSN
RETURN labels(n) AS label, id(n) AS nodeId,
   n.clientDegree AS score,
      n.email, n.phoneNumber, n.ssn
ORDER BY score DESC LIMIT 10;
```

The first query is a graph projection that selects just the clients and identifier nodes and the relationships between them. The relationship orientation is reversed, which allows you to focus on out-degree centrality for the identifier nodes as opposed to the clients. The second query then calculates degree centrality, and the third writes it to each identifier node. When you execute these queries, you find many connections to fake identifiers. The output is shown in Figure 5-2. See the appendix of this book for a full-featured view of this figure.


```
neo4j$ MATCH (n) WHERE n:Phone OR n:Email OR n:SSN RETURN labels(n) AS label, id(n) AS nodeId, n.clientDegree A…
```

	label	nodeId	score	n.email	n.phoneNumber	n.ssn
1	["Email"]	1897782	870.0	"fake@fake.com"	null	null
2	["Phone"]	1897784	773.0	null	"000-000-0000"	null
3	["SSN"]	1897785	765.0	null	null	"000-00-0000"
4	["Email"]	1897783	284.0	"no@gmail.com"	null	null
5	["SSN"]	1895795	7.0	null	null	"222-61-1618"
6	["Email"]	1894948	7.0	"victoriasawyer807@mail.com"	null	null
7	["SSN"]	1894870	7.0	null	null	"602-55-8358"
8	["Email"]	1896041	6.0	"noahhayes589@mail.com"	null	null
9	["SSN"]	1896376	6.0	null	null	"851-06-3396"
10	["Phone"]	1895564	6.0	null	"992-826-146"	null

Started streaming 10 records after 8 ms and completed after 99 ms.

FIGURE 5-2: The results of the Degree Centrality algorithm.

Four big outliers present high scores (column three) for the number of connections. These outliers represent nodes that have fake email accounts, SSNs, and phone numbers.

TIP

Exclude these fake result nodes from your analysis because more than likely these people chose not to fill in the form's information accurately instead of them representing fraudulent activity. If not excluded, you'd find many false positives based on people sharing common, bogus filler information such as an email of "fake@fake.com."

Next, update the labels on these nodes so they'll be easier to exclude from future analysis. The following queries remove the original labels while adding the new "Bad" labels:

```
MATCH (n:Email)
WHERE n.email='fake@fake.com' OR n.email='no@
  gmail.com'
SET n:BadEmail REMOVE n:Email;

MATCH (n:SSN)
```

```
WHERE n.ssn='000-00-0000'
SET n:BadSSN REMOVE n:SSN;

MATCH (n:Phone)
WHERE n.phoneNumber='000-000-0000'
SET n:BadPhone REMOVE n:Phone;
```

Finding Suspicious Clusters

Want to find some actual fraudsters? Now is your time! In first-party fraud, fake accounts are created with no intention of repayment of loans or debt. A common way of finding these fakesters is to look for accounts that share identifiers, like SSNs, phone numbers, and email addresses.

Islands of interacting nodes that have little connection to the larger graph aren't representative of typical financial behavior. You can use this information and the Weakly Connected Components algorithm to find disjointed subgraphs that suspiciously share common identifiers.

TECHNICAL STUFF

The Weakly Connected Components algorithm is a Community Detection algorithm that finds sets of connected nodes in an undirected graph where each node is reachable from any other node in the same set.

The following queries project a graph of client and identifier nodes then run Weakly Connected Components over them.

```
CALL gds.graph.project('proj-clusters',
  ['Client', 'Phone', 'Email', 'SSN'],
  ['HAS_PHONE', 'HAS_EMAIL', 'HAS_SSN']
);
```

```
CALL gds.wcc.write('proj-clusters',{
  writeProperty: 'componentId'
});
```

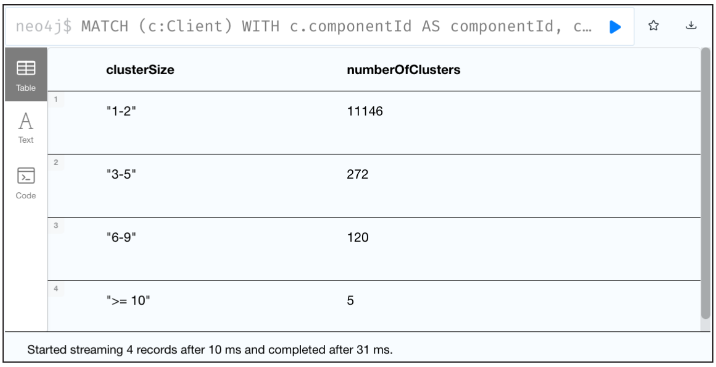
In the previous "Removing Outliers" section, many non-fraudsters use similar bogus form information. We updated the labels of the "Bad" identifier nodes in that section so you can exclude them here by using just the original labels of "Email," "Phone," and "SNN" when building the graph projection. Otherwise, you'd end up with a few exceptionally large clusters due to the commonality of bogus form data.

This Weakly Connected Components algorithm matches clients that share an email, phone number, or SSN, and assigns a label to the componentId property for each node. Nodes that have the same componentId values are considered to be in the same cluster.

After that, you can use the following query to see a distribution of the cluster sizes returned by this algorithm:

```
MATCH (c:Client)
WITH c.componentId AS componentId, count(*) AS
  size
WITH size, count(*) AS countOfComponents
RETURN CASE WHEN 1 <= size <= 2 THEN "1-2"
            WHEN 3 <= size <= 5 THEN "3-5"
            WHEN 6 <= size <= 9 THEN "6-9"
            ELSE ">= 10" END AS clusterSize,
       sum(countOfComponents) AS numberOfClusters
ORDER BY clusterSize;
```

The results of this query are illustrated in Figure 5-3. See the appendix of this book for a full-featured view of this figure.



```
neo4j$ MATCH (c:Client) WITH c.componentId AS componentId, c…
```

	clusterSize	numberOfClusters
1	"1-2"	11146
2	"3-5"	272
3	"6-9"	120
4	">= 10"	5

Started streaming 4 records after 10 ms and completed after 31 ms.

FIGURE 5-3: Most clients are in small clusters.

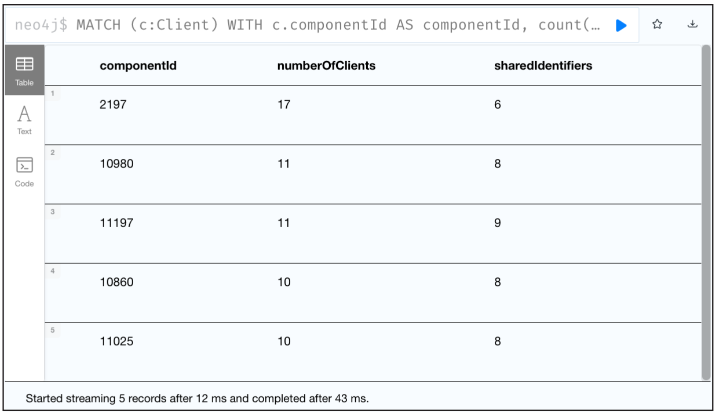
Most clients are in clusters of one or two clients, a few are in clusters of three to five and six to nine, and far fewer are part of clusters of ten clients or larger. You might expect to see this; most people don't share personal identifiers, but there are five clusters with ten or more clients sharing at least one identifier. You can use the two queries below to zoom in on just those clusters with ten or more clients and explore further. The first query adds an additional label to identifiers that are shared by multiple clients which allow the subsequent query (and queries later in this chapter) to filter down to relevant nodes more efficiently. The second query counts clients and shared identifiers in each cluster with more than ten total clients.

```
MATCH (n)
WHERE (n:Email OR n:Phone OR n:SSN) AND
  n.clientDegree > 1
SET n:SharedId;

MATCH (c:Client)
WITH c.componentId AS componentId, count(*) AS
  numberOfClients
WHERE numberOfClients >= 10
WITH collect(componentId) AS componentIds
MATCH
  (n:SharedId)<-[:HAS_PHONE|HAS_EMAIL|HAS_SSN]-(c)
WHERE n.componentId IN componentIds
RETURN n.componentId AS componentId,
  count(DISTINCT c) AS numberOfClients,
  count(DISTINCT n) AS sharedIdentifiers
ORDER BY numberOfClients DESC;
```

These query results are shown in Figure 5-4. See the appendix of this book for a full-featured view of this figure.

Cluster 2197 looks like an interesting one to explore further because it has many clients and six shared identifiers between them. In the next section, you visually investigate this cluster.


```
neo4j$ MATCH (c:Client) WITH c.componentId AS componentId, count(…
```

componentId	numberOfClients	sharedIdentifiers
2197	17	6
10980	11	8
11197	11	9
10860	10	8
11025	10	8

```
Started streaming 5 records after 12 ms and completed after 43 ms.
```

FIGURE 5-4: The results of clusters with ten or more clients.

Visually Exploring a Suspicious Cluster

Exploring cluster 2197 in a tool like Neo4j Bloom, which we cover in Chapter 4, can help you further understand this group. You can visualize the relationships between the clients in that cluster with a Bloom search phrase. A Bloom search phrase is a way that you can define a natural language construct that executes a query against the database for you. The search phrase "explore cluster 2197" finds the relationships in Figure 5-5 between clients in cluster 2197.

The nodes with horseshoe icons represent mules, the ones with people icons are clients, the ones with at-sign icons (@) are email addresses, the ones with card icons are SSNs, and the others are phone numbers.

In this cluster, you have 5 mules, and you can also see three email addresses that are shared by 12 clients. At this point, you probably want to send a list of the people in this cluster to a domain expert to explore further.

From this visualization, you can see that clients in this cluster are sharing just three email accounts. We can imagine a couple of people sharing an email address but having more than that may be something to explore further.

Within the cluster, some of these nodes seem more important, acting as local bridges between clients in different areas of Figure 5-5. You can use the Betweenness Centrality algorithm to confirm your suspicions.

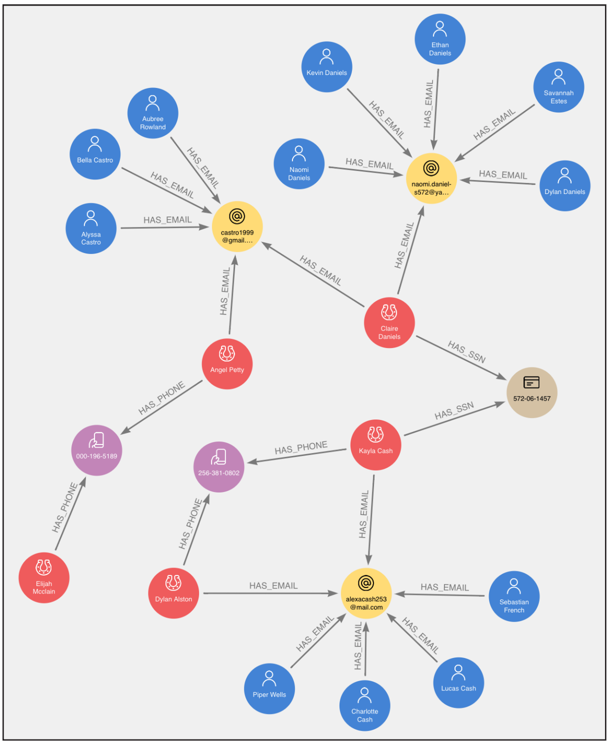


FIGURE 5-5: The resulting graph of the search phrase “explore cluster 2197.”

TECHNICAL STUFF

The Betweenness Centrality algorithm estimates the shortest path between every node pair. Each node receives a score based on the number of shortest paths that run through it. Nodes that most frequently lie on these shortest paths will have higher Betweenness Centrality scores.

Run this algorithm by executing the following queries:

```
CALL gds.graph.project.cypher(
  'proj-cluster-2197',
  'MATCH (c:Client) WHERE c.componentId=2197
  RETURN id(c) AS id, labels(c) AS labels',
  'MATCH
  (c1:Client)-[:HAS_PHONE|HAS_EMAIL|HAS_SSN]-
  >(:SharedId)<-[:HAS_PHONE|HAS_EMAIL|HAS_SSN]-
  (c2:Client)
   WHERE c1.componentId=2197
```

```
    RETURN id(c1) AS source, id(c2) AS target,
   "SHARES_ID" AS type');

CALL gds.betweenness.write('proj-cluster-2197',{
  writeProperty:'betweennessCentrality'
});
```

The first query is a special type of graph projection, a "Cypher projection," that allows you to select just the nodes in the 2197 cluster and aggregate relationships to best facilitate the betweenness computation. The second query calculates the Betweenness Centrality score and stores it in the betweennessCentrality property on each client node for this cluster. After that, you can update the styling rules in Neo4j Bloom to inspect the results in Figure 5-6.

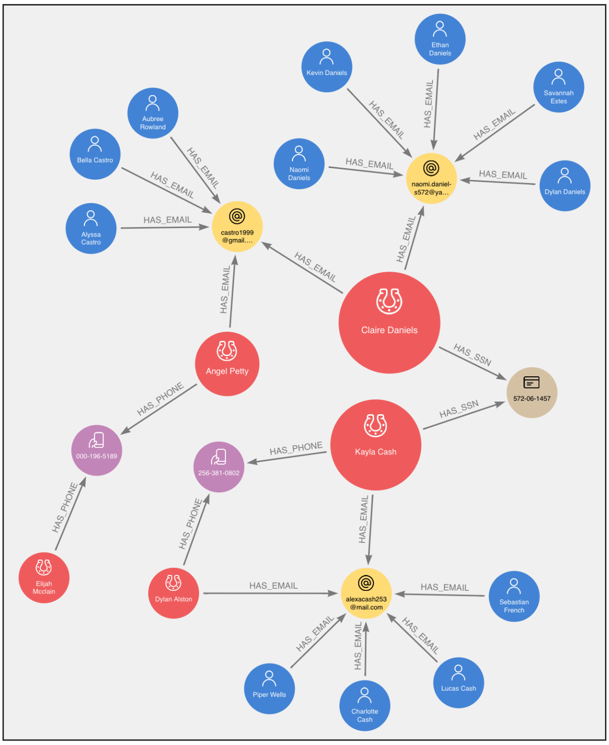


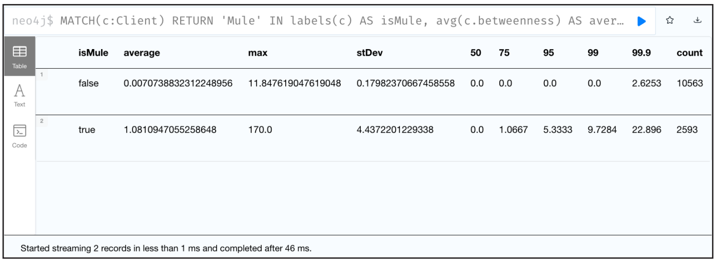
FIGURE 5-6: The result of using the Betweenness Centrality score for node sizing in Neo4j Bloom.

The largest nodes are the most influential nodes in the cluster. These nodes represent mules that are known to commit fraud.

At this point, you've identified suspicious behaviors and clusters. After your fraud analysts confirm this likely nefarious activity, you can use this information to predict mules in the larger dataset.

Using Graph Features to Predict Fraud

In a real dataset, you wouldn't actually know who the mules are, but in the dataset we use, they're identified. This identification allows you to test your prediction that a higher Betweenness Centrality score is predictive of fraud using the whole graph. A quick check of your theory shows that clients with the mule label have on average a 1.0811 Betweenness Centrality score, which is significantly higher than non-mule scores as shown in Figure 5-7. See the appendix of this book for a full-featured view of this figure.



```
neo4j$ MATCH(c:Client) RETURN 'Mule' IN labels(c) AS isMule, avg(c.betweenness) AS aver…
```

	isMule	average	max	stDev	50	75	95	99	99.9	count
1	false	0.0070738832312248956	11.847619047619048	0.17982370667458558	0.0	0.0	0.0	0.0	2.6253	10563
2	true	1.0810947055258648	170.0	4.4372201229338	0.0	1.0667	5.3333	9.7284	22.896	2593

Started streaming 2 records in less than 1 ms and completed after 46 ms.

FIGURE 5-7: Comparing Betweenness Centrality scores for the entire graph.

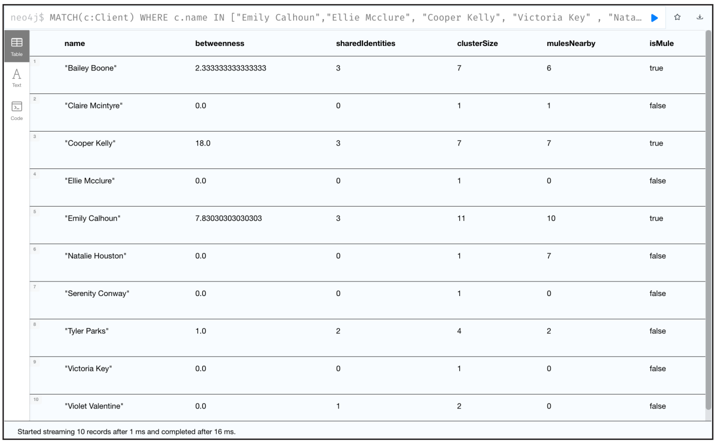
Although this indicator is considerable, the deviation and distribution of scores mean there's overlap that could lead to false positives and negatives. In this situation, you'd want to combine this Betweenness Centrality score with other predictive elements and work with a data scientist to create a machine learning (ML) model.

One ML scenario that you could use is an approach that extracts graph features for use in a binary classifier to predict mules. Examples of graph features include

- The Betweenness Centrality score
- The number of clients sharing identifiers

- The weighting of shared identifiers
- The number of known mules within <n> hops
- The size of clusters

Figure 5-8 shows several graph features we may extract for different people in the graph. See the appendix of this book for a full-featured view of this figure. These features can be extracted to a tabular format for training an ML model.



```
neo4j$ MATCH(c:Client) WHERE c.name IN ["Emily Calhoun","Ellie Mcclure", "Cooper Kelly", "Victoria Key" , "Nata…
```

	name	betweenness	sharedIdentities	clusterSize	mulesNearby	isMule
1	"Bailey Boone"	2.333333333333333	3	7	6	true
2	"Claire Mcintyre"	0.0	0	1	1	false
3	"Cooper Kelly"	18.0	3	7	7	true
4	"Ellie Mcclure"	0.0	0	1	0	false
5	"Emily Calhoun"	7.83030303030303	3	11	10	true
6	"Natalie Houston"	0.0	0	1	7	false
7	"Serenity Conway"	0.0	0	1	0	false
8	"Tyler Parks"	1.0	2	4	2	false
9	"Victoria Key"	0.0	0	1	0	false
10	"Violet Valentine"	0.0	1	2	0	false

Started streaming 10 records after 1 ms and completed after 16 ms.

FIGURE 5-8: A matrix of graph-engineered features and mule classification.

REMEMBER

If you're happy with your fraud detection model, use it in production to identify other mules as your graph evolves. As new information is added to real-world graphs, it's common to iterate on this process and create new graph features and update models.

IN THIS CHAPTER

» **Expanding your knowledge with Neo4j resources**

» **Getting started with a spearhead team**

» **Implementing a strategy to get your project approved**

Chapter 6 Ten Tips with Resources for Successful Graph Data Science

If you're wondering if your project can benefit from a graph or how to get started with graph data science, this chapter can help. We give you some Neo4j resources to guide you to more information, and to help you explore your project's opportunity and successfully move forward from concepts to production, we include these ten tips:

» **Investigate use cases and get comfortable with concepts.** Because graph technology is applied across industries and in various use cases, it can be hard to know where to start. To expand your knowledge and help you get comfortable with graph data science, review these examples:

- **Review use cases.** Get up to speed on the problems graph technology can solve. To read some use cases, visit `neo4j.com/graph-usecases`.
- **Watch talks.** Find out how people use graph data science. Watch presentations from Neo4j Connections for graph data science digital events: `neo4j.com/connections_events`.

    - **Expand your knowledge of key concepts.** Review material that sets graph data science in a larger context. Visit `neo4j.com/graph-resources` for how graphs enhance AI.

- **Identify and engage a spearhead team.** Using graph technology in production can be new to many people, so don't expect teams to understand how to evaluate or compare graph options to other solutions. Assemble a small team that can become your experts in translating business needs into technical requirements and the application of graph data science. Make sure to have representation from key organizations, including business, IT, and data science teams.

  TIP

  Provide your data scientists with more technical information. Your team will likely need time to familiarize itself with the technology so look for resources that allow an easy start. Some examples include

    - **Neo4j Sandbox:** `neo4j.com/gdssandbox`
    - **Neo4j Graph Data Science developer guide:** `neo4j.com/gds-getting-started`
    - **Neo4j Graph Academy:** `neo4j.com/graphacademy-gds`
    - ***Graph Databases For Dummies,* Neo4j Special Edition, e-book:** `neo4j.com/gds_for_dummies`

- **Evaluate your problem.** Graph technology is useful anywhere you have a lot of connected, interdependent information. But at some point, you need to look into what areas of your business to focus on and what kind of project to start with.

  TIP

  Start with an intersection of ideas between users, business, and technology. Consider hosting offsite or virtual innovation sessions with your cross-functional team to define your stakeholders' needs, create connections-related questions, story-board possible solutions, and identify key challenges and opportunities. This collaboration may naturally lead to a prototype that you can share with executives for feedback, but the goal is to uncover promising target use cases.

- **Assess the current state.** After you have a target use case in mind, start with documenting your current state. Consider existing problems as well as how the various parts of your organization will have different experiences and issues. Find out how your business sponsors view this use case and any problems or opportunities. Be as specific as you can. For example, what is the impact per customer of improved online profiles? What's the revenue implication of a half percentage increase in recovered fraud? Also remember to consider external market factors such as customer or transaction growth, competitive factors, emerging opportunities such as new delivery platforms, or productization opportunities.

- **Map the value of the proposed state.** Although your first graph project may spawn many new ideas and future projects, make clear and direct mapping of features of the near-term graph project to business values. Consider the current state and pain points and how your graph target use case can help with business concerns such as cost savings, increased revenue, new market opportunities, time to market, risk mitigation, and the like. For example, uncovering similar customer journeys and using that information in a machine learning (ML) model may increase the accuracy of churn prediction so the business could take early preventative action and reduce revenue loss.
- **Measure ROI.** For each of your value areas, determine how you plan to measure your return on investment (ROI) or success. For example, will you use predictive accuracy or reduced financial loss to estimate the impact of your end state? Compare the soft and hard costs of maintaining existing processes to your graph project. If you're unable to audit your existing state, be more conservative when estimating incremental saving or revenue opportunities. Likewise, it may be difficult to measure the value of net-new capabilities, such as answering previously intractable questions, so you may need to get creative or add qualitative analysis.

REMEMBER

- **Align stakeholders.** Eventually, you need cross-functional agreement on the goals and requirements of your graph project. This process is iterative, not something you tackle at one point in time. Different teams may have alternative views on the project vision, key ROI, and even the role of graph technology. Getting alignment on the goals of the project and how success is measured are essential — and you may want to consider a process for dealing with conflict or dissenting opinions.

- **Get your project approved.** Taking advantage of new technologies like graph data science requires your stakeholders and approvers to be comfortable trying something unfamiliar, so your work to target the right use case, map values, and estimate ROI needs to come together in a concise story that aligns with your company's motivations.

  TIP

  Document stakeholder assumptions about business value. For example, customer churn may be an issue, but is it a priority and why? You may be asked about the competitive landscape as well as alternatives and the costs or lost opportunities if you don't proceed. Clearly document the interdependent system touchpoints that are part of current processes and the impact of your graph solution.

- **Conduct a POC and plan for production.** Larger projects, especially if the technology is new to a team, often require a proof of concept (POC) before approval and deployment. A POC can prepare your team for production and identify any gaps. This process may involve iterating on previous prototypes before you move into data modeling and testing specific workflows.

  REMEMBER

  In graph data science, your data model and algorithm choices are highly dependent on the questions you're trying to answer. Your data scientists and subject matter experts should be involved to ensure the right assumptions are made. Also make sure that your IT teams are involved to raise any red flags and that your end-users are on hand to evaluate any usability concerns.

  TIP

  Vendors that provide POC services can help accelerate your project with their graph experience. Visit `neo4j.com/professional-services` for more info.

- **Get connected and continue your journey.** Applying graph data science is a journey. You may start with one focused project and find yourself answering questions you never knew you had. We highly recommend your team connect and engage with the graph community. Graph communities consist of active groups of users who share new ideas and help with specific, and sometimes unusual, questions. Getting involved in an active community with educational support and certifications helps your team be successful with its first graph project and expands the value of your graphs over time.

  TIP

  Visit the Neo4j community at `neo4j.com/gds-community`, and check out its resources at `graphacademy.neo4j.com`.

Appendix

n this appendix, we formatted some of the figures from Chapter 5 into full-featured tables so you can better see the details in each image.

Figure 5-2

The Results of the Degree Centrality Algorithm

label	nodeId	score	n.email	n.phone Number	n.ssn
["Email"]	1897782	870.0	`"fake@fake.com"`	null	null
["Phone"]	1897784	773.0	null	"000-000-0000"	null
["SSN"]	1897785	765.0	null	null	"000-00-0000"
["Email"]	1897783	284.0	`"no@gmail.com"`	null	null
["SSN"]	1895795	7.0	null	null	"222-61-1618"
["Email"]	1894948	7.0	`"victoriasawyer807@mail.com"`	null	null
["SSN"]	1894870	7.0	null	null	"602-55-8358"
["Email"]	1896041	6.0	`"noahhayes589@mail.com"`	null	null
["SSN"]	1896376	6.0	null	null	"851-06-3396"
["Phone”]	1895564	6.0	null	"992-826-146"	null

Figure 5-3

Most Clients Are in Small Clusters

clusterSize	numberOfClusters
"1-2"	11146
"3-5"	272
"6-9"	120
">= 10"	5

Figure 5-4

Results of Clusters with Ten or More Clients

componentId	numberOfClients	sharedIdentifiers
2197	17	6
10980	11	8
11197	11	9
10860	10	8
11025	10	8

Figure 5-7

Comparing Betweenness Centrality Scores for the Entire Graph

isMule	average	max	stDev	50	75	95	99	99.9	count
False	0.0070738832312248956	11.847619047619048	0.17982370667458558	0.0	0.0	0.0	0.0	2.6253	10563
True	1.0810947055258648	170.0	4.4372201229338	0.0	1.0667	5.3333	9.7284	22.896	2593

Figure 5-8

A Matrix of Graph-Engineered Features for Mule Classification

name	betweenness	sharedI-dentities	clusterSize	mule-sNearby	isMule
"Bailey Boone"	2.3333333333333333	3	7	6	true
"Claire Mcintyre"	0.0	0	1	1	false
"Cooper Kelly"	18.0	3	7	7	true
"Ellie Mcclure"	0.0	0	1	0	false
"Emily Calhoun"	7.83030303030303	3	11	10	true
"Natalie Houston"	0.0	0	1	7	false
"Serenity Conway"	0.0	0	1	0	false
"Tyler Parks"	1.0	2	4	2	false
"Victoria Key"	0.0	0	1	0	false
"Violet Valentine"	0.0	1	2	0	false

neo4j

HARNESS THE PREDICTIVE POWER OF RELATIONSHIPS

Graph data science helps businesses across industries leverage highly predictive relationships and network structures to solve unwieldy data problems.

Discover how Neo4j graph databases with the power of graph data science enhance machine learning and artificial intelligence with connections and context.

Download the free white paper

Graph Embeddings: AI That Learns from Your Data

https://neo4j.com/graph-whitepaper

Answer business-critical questions with graphs

Graph Data Science For Dummies, 2nd Neo4j Special Edition, focuses on the applications of graph analysis and graph-enhanced machine learning, which both take the form of graph data science. You discover the graph data science basics and learn about its adoption. We give you the Neo4j graph database technology to help illustrate our points about the graph data science platform. We also supply you with plenty of resources to guide you outside of what this book provides.

Inside...

- Understanding graph analytics and GDS
- Using questions to explore GDS
- Putting graphs to work in the real world
- Applying graph data science technology
- Tips and resources for successful GDS

neo4j

Image provided by Neo4j, Inc.

ISBN: 978-1-119-90932-3
Not For Resale

9 781119 909323

for dummies®
A Wiley Brand