OpenCredo
A TRIFORK COMPANY

Connecting the Dots: Harness the Power of Graphs & ML

opencredo.com

CONTENTS

ABOUT THE AUTHOR

Ebru Cucen

Lead Consultant

Ebru is a Lead Consultant at OpenCredo who is passionate about turning data into actionable insights. She believes Graphs hold the key to unearthing novel and interesting answers to some of the most challenging connected data challenges posed by business today.

Building on her BSc in Mathematics, she has over 20 years of experience with a wide perspective in multiple business domains, including Finance, Environmental, Social, and Governance (ESG), Pharma, Retail, and Transportation. Hands-on, she has been involved in the end-to-end design, building and delivering of scalable fault tolerant cloud-native solutions. With a more recent focus on data engineering, graphs and ML/AI, she looks to combine the best of both worlds - gaining new and fresh insights, delivered through stable, reliable solutions.

OpenCredo

A TRIFORK COMPANY

OpenCredo are a hands-on software development and data consultancy, partnering with clients and industry leaders to deliver meaningful insights & solutions at scale.

From MVPs to full scale multi-region data driven systems, our experienced and diverse teams are trusted to review, architect and build adaptable systems which stand the test of time.

We focus on:

- Data, Graph and Platform Engineering
- Cloud Native Architectures
- Application Development

INTRODUCTION

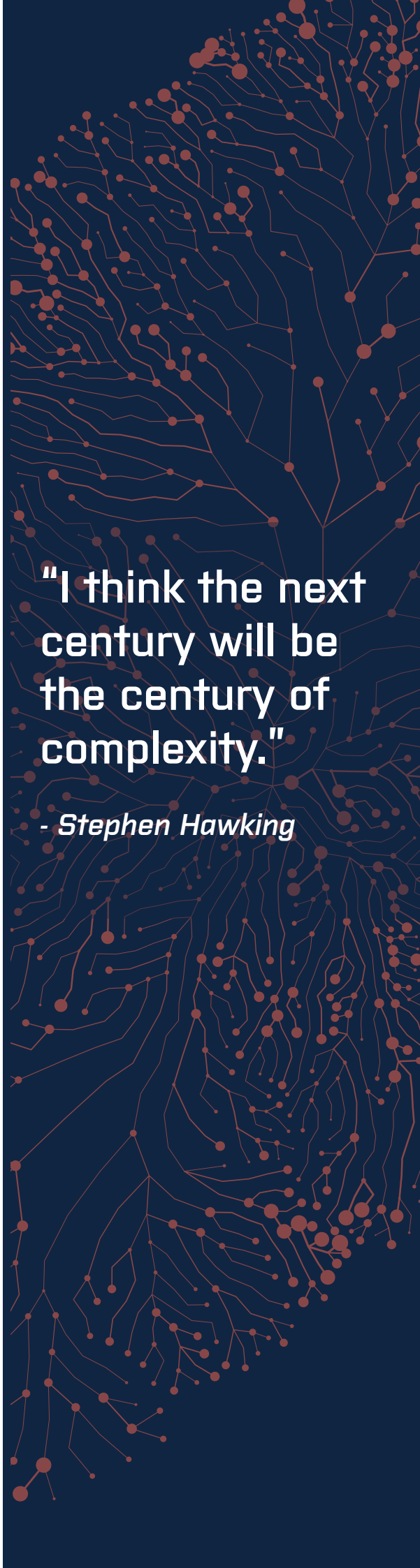
As our world has become more complex, informed decision making has become critical for managing the uncertainty we experience. Machine Learning and Artificial Intelligence methods have very much helped us along this journey. However, even these are continually being disrupted by newer technologies which promise even more accurate predictions for *better decision making*.

In our quest for faster results and greater accuracy, much focus has been concentrated on improving existing, or creating new machine learning algorithms. Yet these often fail through lack of context. Iterating over a model, no matter how good it is, without context fails to connect the unseen dots nor provide any helpful insight. This is where we believe the power of graphs comes in. Graphs (in the mathematical sense) are **extremely good at encapsulating** and being able ***to embody and convey crucial data*** (context) about how and why entities are related to one another. Such information can often be **THE crucial piece of a puzzle** needed to answer complex business questions, or indeed to serve as the secret ingredient to unlock, optimise and improve predictive ML models to help us do this.

Would you value having the power to uncover hidden links and patterns in your data and be able to predict the future? If so, read on!

This book aims to shed light on a real game-changer for those looking to improve upon simplistic answers sometimes arrived at by using traditional ML algorithms and approaches. We show how - in a variety of different ways - you are able to combine the power of both graphs and ML to achieve more accurate answers.

This book is not meant to provide in-depth academic coverage of this subject. Instead we will highlight the intuition behind how these two powerful concepts can be combined, providing some practical insights and examples along the way to help with the explanations.



WHAT'S SO SPECIAL ABOUT GRAPHS?

Connections - the Secret Sauce

You've heard it said many times before - it's more about *who* you know, than what you know. Whilst this may sit uncomfortably with many of us, it is nonetheless a key factor that influences many aspects of life around us.

Consider a company trying to promote their new eco-friendly pet food to clients on a social media platform, but not having much luck. Whilst the company may know for a fact that a certain group of people have indicated an interest in pet food, it's often not enough to clinch the deal. If however they manage to identify a highly connected individual (perhaps a celebrity or eco warrior), and get them to endorse and promote it to their followers, their product may well land up going viral.

We should be vigilant - connections and relationships matter! And if we are able to surface, identify and leverage the information provided through these connections in our businesses (be these human or other) this could prove to be the secret ingredient which helps us to make better, more informed decisions.

More concretely and specifically related to ML (as we see later on), graph representations can enhance existing ML models to explain data, predictions, and algorithms by relying on the power of connections.

Take for example DeepMind's AlphaFold discovery during a recent CASP challenge. They discovered they could accurately predict the structure of a protein in at least 75% of cases by representing the amino acids as a "spatial graph" and solving one of biology's biggest challenges in the process. In addition to this, there are many peer-reviewed academic papers published proving that the graph representation improves the accuracy of the algorithms' predictions. The examples are in multiple fields, such as recommendation systems (GraphSAGE), traffic speed forecasting (STGNNs), and program reasoning (Gated Graph Neural Networks).

Gartner positioned Graph in its Top 10 Data and Analytics Technology Trends for 2020. This report suggested:

50% of AI conversations today involve Graph methods

92% of the organisations it polled are planning to employ a graph technique within 5 years

Graph will form the basis for 80% of the innovation in the data and analytics space by 2025.

View the Gartner Report here

Multi Domain Applicability

Graphs and their algorithms are universal in their application across domains and industries, which makes them very versatile. So whether you are in telecommunications, the social network space, finance or healthcare, whilst the entities represented in your graph may be different (people vs cell phone towers vs diseases), the mathematical calculations and algorithms applied over them remain and work the same irrespective. This characteristic enables discoveries or enhancements in any algorithm made in one domain to be easily repurposed for use in another.

At a high level, Network Science is an area of research built heavily on graph theory, but which originally focused on studying real world phenomena in order to uncover common patterns and interactions in complex networks. Often spanning physical and natural networks (such as biological or social), researchers found that observations and predictive theories uncovered in these settings, often applied and translated equally well to other contexts (e.g. scholarly, transportation or power networks). This resulted in common families of algorithms, arising such as detecting communities or related groupings of entities which apply across many different domains today.

At a lower level, a concrete example would be the Pagerank algorithm, made popular by Google, it is really an extension of the **centrality algorithms** used in social networks. Likewise Amazon used an extension of similarity algorithms to apply recommendation systems on their own domain for many years. It's also worth noting that most new algorithms published academically tend to be evaluated against multiple domains. For example, a recent and popular algorithm, node2vec, tested and verified its approach against a social network (Facebook), a Protein-Protein Interaction network, as well as a Citation network.

Natural & Easy To Reason About

Graphs are also really natural and easy to work with (we provide a quick overview in the next section). Much of the academic and more difficult aspects of working with them has become commoditized, making it easier for lay people to work with. For example: ICIJ (International Consortium of Investigative Journalists) imported and used Neo4j (a graph database) along with Linkurious to model and explore the Panama Papers and FINCen files - this was instrumental in highlighting and finding irregular issues in the financial and political worlds.

Easily Accessible & Usable Software & Platforms

There are also various open source and commercial platforms available that aim to make the visualisation, exploration and ability to gain insight out of connected data more easy. This includes products like Linkurious, Hume, Bloom etc.

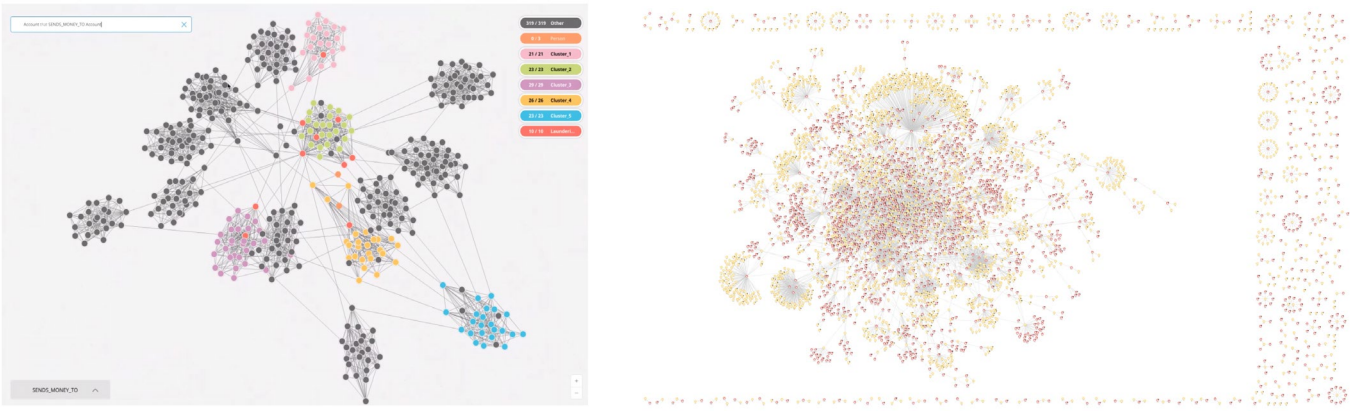


Figure 1.1 Visualisation & Exploration Tools on Amazon Product Reviews (Bloom & Hume)

Many of the major graph platforms and databases have also made their offerings available for consumption in the cloud or as SaaS offerings. This is making it even easier than ever to get up and running. Eg TigerCloud, Neo4j Aura, as well as native cloud offerings like AWS Neptune, Azure CosmosDB etc.

Graph Primer 101

It's worth us briefly describing what a graph is, and establishing some basic language to ensure we are all on the same page.

Graph Theory is a domain of Mathematics that originated in 1735 with Euler's Seven Bridges of Königsberg problem. If you want to explore more, The Fascinating World of Graph Theory book is a great introduction with a comprehensive coverage on the theory and the applications.

A graph representation is a natural data structure - it aligns verbally with how we think and communicate about relationships. Thereby, it aligns better with talking about data and their relationships regarding their modelling of real-world concepts, allowing us to better understand or focus on the derived insight, rather than remapping between relational models back to the real world, which adds more cognitive load and possibly impedes deeper analysis of the data and its relationships.

It is typically constructed using nouns (nodes/vertices) and verb predicates (relationships/edges). In a property based graph (the most common and widely used type) both nodes and relationships can be enhanced by adding properties to them, providing a richer description.

Relationships are further defined as either being directed or undirected, and can also be weighted (i.e. it is able to indicate the strength of one relationship compared to another). This is often achieved by using one of the numeric based properties which can then be used in later algorithms.

By way of an example: imagine we wanted to model a typical e-commerce scenario, where *customers order products*. In such a graph the *customer* and *product entities* become the nodes (vertices) and have an *edge* relationship of ordering going between them. The customer may have properties such as age and location, and the product a price. We could add more depth to the *ordering* relationship by including properties such as how many times the customer viewed the product before buying it, or even other metrics like the amount of time it sat in their basket before purchase.

Below is a visual representation of this example. The nodes (Vertices) are the Customer, Product, Inventory, Region and Country. Then, we have relationships (Edges) representing the directed links between them, namely ORDERED, REVIEWED, STORED_IN, LOCATED_IN and IS_PART_OF.

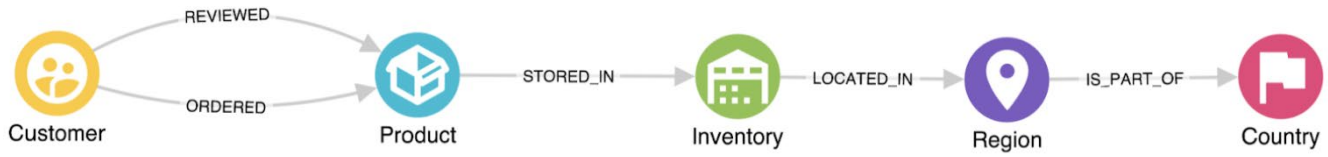


Figure 1.2 A simple supply-chain network

As noted earlier, edges themselves can be extended with properties including weighting values, which can be used for further computations by certain algorithms. In the visualisation below, the ***REVIEWED*** relationship has date, helpfulness, rating, review, and summary properties. The rating value could serve as a weight.

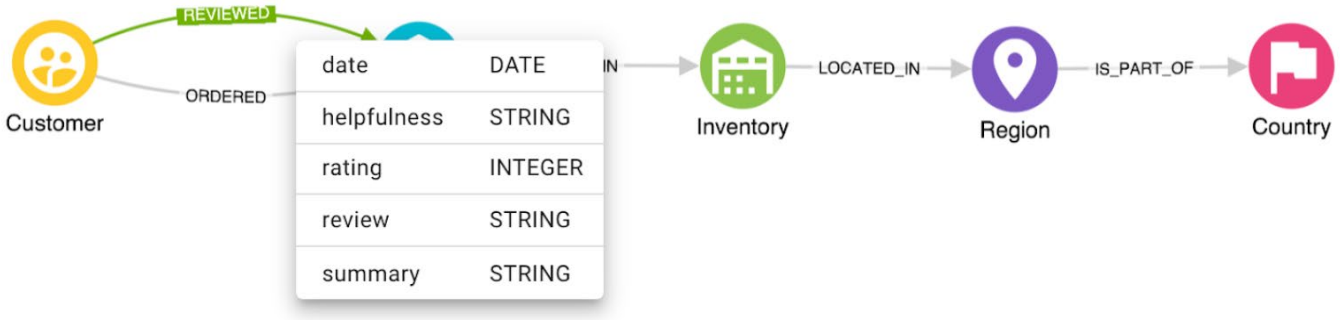


Figure 1.3 An edge is extended with multiple weight properties

This model may look like a relational database model, the difference will become more clear when we want to explore the relations between the customers only to find out how similar or how important they are relying on the relations they have with the other nodes. With 400 million monthly active users pinning 240 billion products, Pinterest needed a recommendation system to suggest products to customers. They solved this problem with graphs, which we will explore further in these next parts of the Ebook.

GNNs improve real-time ETA predictions by 50%

DeepMind, London-based AI lab owned by Google's parent company Alphabet, picks out patterns in the data and uses them to predict future traffic. Google says its new models have improved the accuracy of Google Maps' real-time ETAs by up to 50 % in some cities.

GRAPHS, DATA SCIENCE AND ML - HOW ARE THESE RELATED?

Wikipedia, as well as Cassie Kozyrkov happily defines Data Science as a 'concept to unify statistics, data analysis, machine learning and their related methods' in order to 'understand and analyze actual phenomena' with data. Data Science is an umbrella term which encompasses a few different areas including the use of general statistics and analytics (where graphs and their associated graph algorithms happily contribute already as a subspecialty in and of their own right), as well as AI and Machine Learning.

Ultimately however, regardless of the technique used to get there, the goal of data science is always to be able to understand and extract useful information from data for insights and decision making purposes. As we shall see in the remainder of this Ebook, Graphs and ML can be mutually beneficial to each other. To help us explore the various ways in which graphs can feed into ML models and vice versa, we shall use the diagram below to guide our thinking, referring to it at appropriate points in each section.

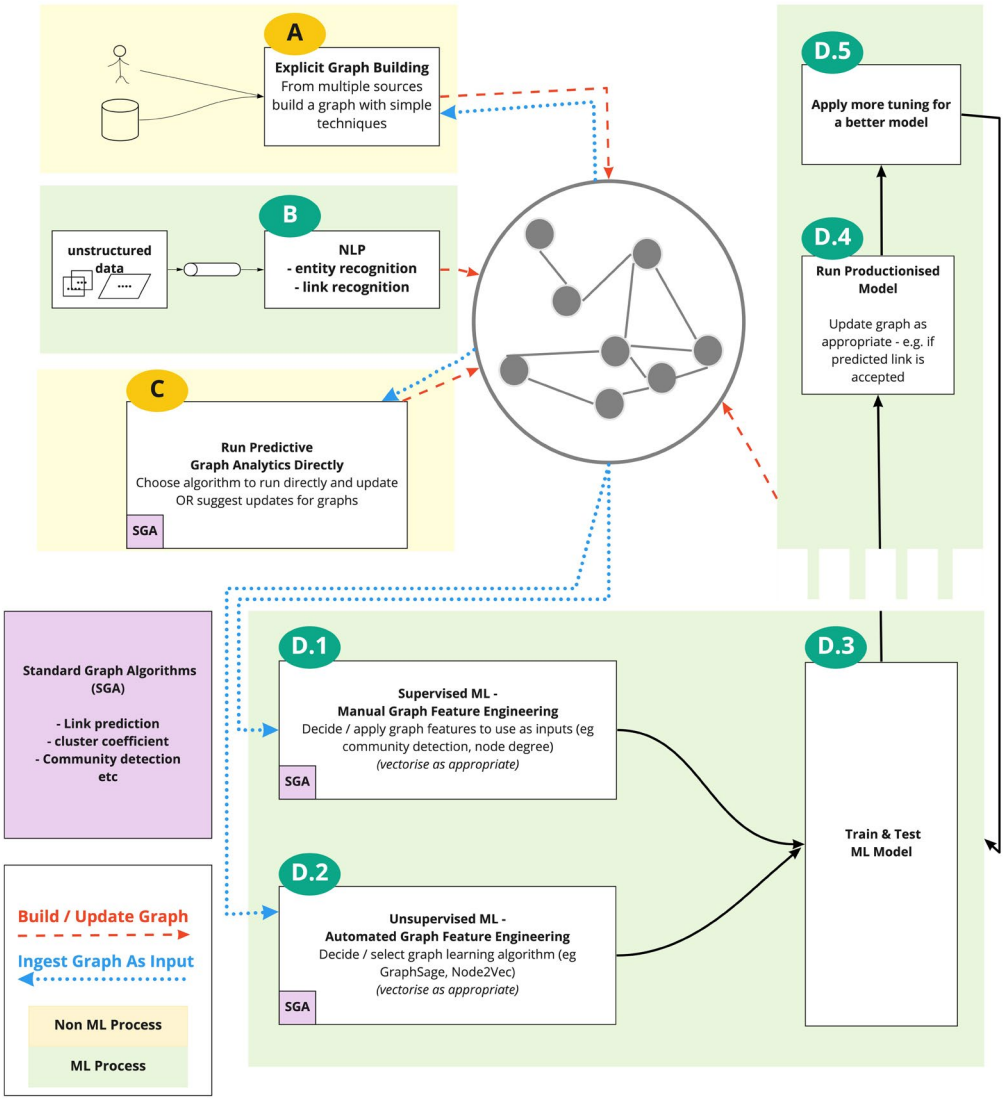


Figure 1.4 Different Interaction Modes For Graphs & ML

The Foundation - Knowledge Graphs

So can you get insights from your data just by modelling and visualising it as a graph? Yes you can! And this is typically the entry point into the world of graphs for most organisations - it is a very natural way to represent and store data which is highly connected and allows for interesting analysis of your network domain - a graph is also often referred to as a network.

The deliberate gathering of data from multiple sources and linking them appropriately, often results in what is referred to as a "knowledge graph". Where data pertaining to your specific domain of interest is intentionally interconnected in a meaningful way to allow for strategic decision making, context and programmatic reasoning and predictions to be made. Organisations sometimes find that they collect more data than they currently need, however they still choose to include this in their graph in the hope that it will prove useful later down the line. This is a little bit like a "data lake" - where having stored much, you know there is lots of good information to be mined, but you are just not sure what it exactly looks like yet. Graphs have a flexible but predictable query structure which make this very easy to do with a better audit trail capability.

Using ML to help build the Graph

Oftentimes the data in the graph is built up as a result of explicit user driven or programmatic rules/user supplied information (ref A in Figure 1.4). For example, one of the common problems organisations face is to assess the skills areas for their employees to increase productivity and performance. To achieve this, data can be collected from users, and linked to other systems to integrate and provide better visibility of the employee profiles.

Natural Language Programming (NLP)

Graphs can just as easily be built up via more automated means, for example by extracting data from more unstructured sources via techniques like NLP (ref B in Figure 1.4). "NLP is a field in machine learning with the ability of a computer to understand, analyze, manipulate, and potentially generate human language" - For example an organisation may scan people's CV's to try and make sense of text, utilising techniques such as entity extraction or named entity recognition to detect "entities" like former places of work, skills etc and building the knowledge into the graph this way. This is one example where AI/ML techniques can be used to construct a new or feed into an existing knowledge graph. This comes with its own set of challenges in terms of being able to actually identify the correct nodes and relationships in the first place, but continues to be a growing area of research in improving the automated creation of knowledge graphs in the face of vast swathes of unstructured data.

CUSTOMER STORY

NationalJournal

OpenCredo has helped many customers to consolidate and model highly connected data projects, such as the National Journal, a Washington D.C. based research and advisory service. Working with the Network and Science Initiative team, the project aimed to consolidate their disparate datasets, forming a single source of truth. This would allow their analysts to access better quality insights with graph visualisations on their clients, projects, and connections using Neo4J, Linkurious running on Google Cloud Platform. You can read more in our case study.

Link Prediction

In some cases you may have a graph, where perhaps some information is missing. You may be missing certain nodes, but more importantly relationships between those nodes. Social networks seldom have all the relationships actually in existence contained within them, and many biological modelled networks are often incomplete. For example consider protein-protein interaction networks, where proteins are tested whether they interact with each other or not. Scientists can typically only predict which protein molecules can interact with others through real world field or laboratory experiments. Today, scientists have identified less than 0.3% of interactions between human proteins. However, modelling this information as a graph, and then being able to infer and suggest interactions (links), some significant cost savings can be gained for those in the bioinformatics space.

A family of algorithms called link prediction can help here. As the name implies, it tries to predict where there may be missing, or indeed possible future links between two currently unconnected nodes in a network. The most common way of doing this is by examining and using current (often local) connections and properties to try and work this out. In layman's terms, one tries to compute the "closeness" between nodes. Specific algorithms include Adamic Adar, Common Neighbors and Preferential Attachment to list a few. Given two nodes, this family of algorithms can produce results which are often used as direct predictors of missing links (ref C in Figure 1.4).

Now whilst such insights are valuable in their own rights, they (along with other properties and information about relationships) can yield much more accurate results when they form part of, or are provided as input into continual learning processing via machine learning (ref D1-D5 in Figure 1.4). This is where we will be concentrating more of our time, but as you can see, we have created a somewhat circular workflow because we have the graphs feeding ML models (via algorithmic results - ref D1 & D2 in Figure 1.4), and the ML models potentially updating the graph in return! (ref D4 in Figure 1.4) We have a feedback loop - which is great, it means our models can continually learn!

ML Mapping of Cancer-beating Molecules

A team at Imperial College London, led by Michael Bronstein, discovered cancer beating molecules "hyperfoods" simply using Graph ML by modelling the "network effects" of their interactions.

Graph Data Science - Extracting Value Via Standard Analytics & Algorithms

There is a whole range of well established and understood graph algorithms which can be used "as is" to extract information, gain insights and make predictions once you have your data represented in a graph - No ML required (ref C on Figure 1.4). We are not going to try and list them all in this Ebook as the focus is more on how to combine them with ML, and many books have been written on these. For completeness however we briefly highlight a few primary categories and algorithms which can be helpful, especially as they relate to usage as input to downstream ML model building.

The book Graph Algorithms by Amy Hoddler and Mark Needham does a great job of taking you through some of these and their various applications demonstrated in both Spark and Neo4j.

Search & Path Finding Algorithms

Foundational algorithms such as Breadth First or Depth First provide us with a few different ways of searching, walking or exploring a graph. In fact many other families of algorithms are built on top of them. Pathfinding more specifically, seeks to answer questions such as what is the shortest path between two nodes, or how many different paths exist between two nodes, and what does that path of connections for them look like? The Dijkstra algorithm typically features heavily here, as does the Random Walk for more exploratory requirements as we discover a bit later.

Use Cases:

Transport and physical logistical networks work well with such algorithms. Working out the fastest route to get a parcel to you is an obvious example. Likewise understanding what downstream houses or businesses are impacted if a particular substation in an electricity grid goes down serves as another good example. Connectivity between nodes/edges does not have to be confined to the physical/geographic layout of related edges. The relationship and distance between the edges can model other metrics, such as time or cost. Supply chain models also benefit from pathfinding algorithms as many possible considerations may need to be taken into account for how to effectively distribute goods. Besides the cost of materials and freight routes, storage can change, and thus finding the shortest path (including using weighted information, i.e. which metric is more important for the use case) is essential for best resource consumption when we have regulations, and unexpected supplier changes occur. For more information on this, you can read our case study on how Sedex has provided a fast Ethical Supply Chain with the help of OpenCredo.

Centrality Algorithms

Centrality Algorithms are used to find the most critical or important nodes (and their associated influence) in a network. They help answer questions and provide insights around group dynamics (i.e. which node, person, org etc is most credible), or identify bridges between different groups. The most straightforward (but also quite limited) algorithm is the Degree Centrality, where we can identify a node as powerful by its total number of degrees (i.e. total number of incoming or outgoing relationships it has). PageRank, popularised by Google to rank web pages, has massively gained in popularity and is being used in many more commercial contexts.

Use Cases:

PageRank is excellent at detecting the central figures in a network, the rising stars, or the malicious threads and enables us to better understand the network we have. It can help us find emerging leaders in fast-growing companies or reveal criminal networks. In one case, it was used to rank spaces in order to predict human movement in an urban environment/city.

Clustering Algorithms
Clustering or Community detection algorithms help us find the groups of nodes which seem to belong, or are related to each other in some way. This is done by looking at which members have more significant interactions with each other and classifying them. It is able to reveal tight clusters, isolated groups, and structures not overtly obvious or labelled upfront. Algorithms in this space include Louvain, Label Propagation and Strongly Connected Components.

Use Cases:
The most famous example is Zachary's karate club. During his studies, anthropological researcher Wayne Zachary observed a conflict between the leaders of the karate club and a member, and he predicted how many subgroups would emerge. Another example from more recent history is the study on Enron's fraud investigations. Email, a standard method for social network extraction, was used to identify the communication patterns before and during Enron's collapse. The studies showed that we could detect fraud systemically as an abnormal behaviour with highly segmented and cohesive groups with little cross-communication. *During the fraud, there were clusters of people conversing that wouldn't normally.*

Link Prediction Algorithms
As noted earlier, this family of algorithms helps predict where there may be missing or possible future links between two currently unconnected nodes in a network. Specific algorithms looking to compute the "closeness" between nodes include Adamic Adar, Common Neighbors and Preferential Attachment.

Use Cases:
To find missing links in networks. Take for example a social network, where the "friendship" suggestions in LinkedIn or Facebook typically reveal a hidden connection - i.e where the user already knows the suggested person but has not connected on the social platform yet. It could likewise identify where there is a high possibility the users will connect in the near future. Recommendation models also make heavy use of link prediction algorithms.

Customer Churn Prediction in Telecommunication

Customer Churn Prediction is a very critical model algorithm for every business. A telecommunications company integrated Social Network Analysis (SNA) in their prediction model. This enhanced the performance of the model from 84 to 93.3% against AUC standard.

Using Graphs to Enhance ML

Whilst we know graphs contain rich predictive information which would be valuable for use in ML models, making it available in a timely manner and format which ML models can consume, is somewhat of a harder problem. Continuing with our diagram from earlier we now explore how graphs can specifically be fed into and used to enhance ML models.

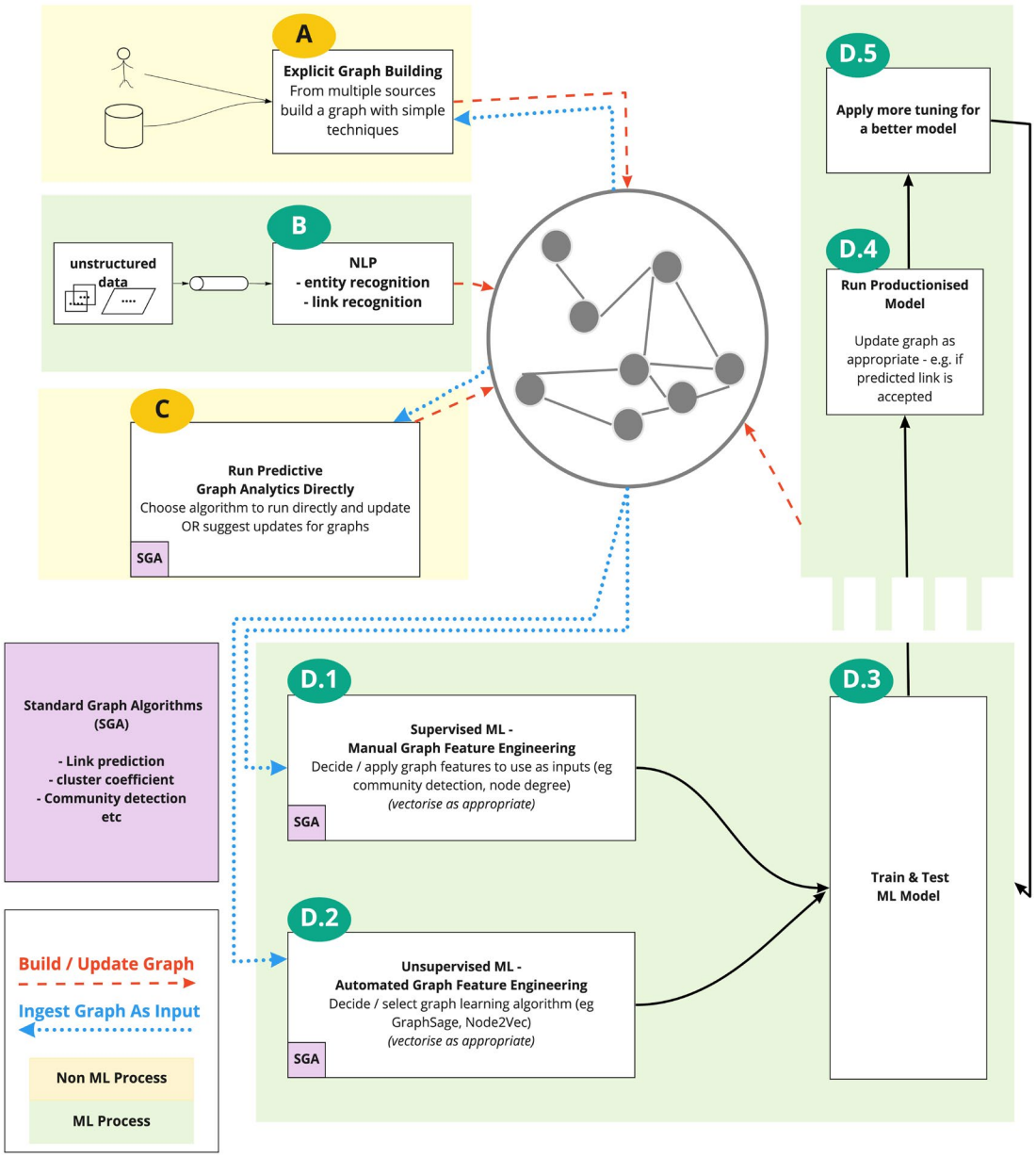


Figure 1.4 (Duplicated) Different Interaction Modes For Graphs & ML

Using Basic Graph Algorithms & Metrics

Let's assume we have identified some standard algorithms which can provide some good predictive elements about our specific domain. Examples may include running community detection in order to label nodes with their respective grouping result (node classification). In another case, using page rank to score how important people are in their network from an influential perspective, and assign a score to them. What do we do with this information?

As a first pass, especially where the results are simple scalar values as in the above examples, they can simply be extracted and added as individual features to an existing ML feature vector (ref D1 in Figure 1.3). Note, to make life easier, these results are often written back into the graph (eg. as a property on a node). This approach is more typical in supervised ML models. See the example feature vector below used for input for a ML model to predict how many people would buy a particular product if the specific person in the network promoted it on their website.

	Traditional property based features (obtained from non graph data)		**Graph Extracted Features**	
Person ID (also Node ID)	**Age**	**Location**	**Community ID**	**Influence (PageRank) Score**
A1UQRSCLF	25	London	1 (Retail Shopper)	0.56
AJ613OLZZ	68	Bristol	2 (Pensioner)	0.12
AZOF9E17R	26	Warwick	1 (Retail Shopper)	0.15

We have essentially engaged in what is known as feature engineering, and have achieved our goal of transforming and including some insightful graph metrics as specific features enabling an existing ML model to use this newly added info.

Graph Embeddings

Relying on supervised, user directed knowledge as described above offers limited improvement for feature engineering however. The features are hand-engineered, and are inflexible as they cannot learn during the learning process. Recent studies however have highlighted new approaches to graph representation learning - namely finding more dynamic exploratory ways of traversing, uncovering and automatically encoding the structure of the network into one or more vector spaces - otherwise known as "graph embedding".

This is typically done by embedding details of specific components themselves (usually the nodes along with info about some of its related edges) in a vector, or sometimes indeed the whole graph itself as a single vector. These more dynamically learned vectors can then be used by ML Models (ref D2 in Figure 1.3).

One such way to do this is via Node2Vec, an algorithmic framework for taking a node and its surrounding neighbourhood and vectorising it. This approach uses multiple **random walks** to move over the graph space, thereby allowing the process to "learn" about its surrounding neighbours. This info is then preserved by capturing it in a vector, where nodes inferred to be similar or part of a community in the original network, are represented as being "close together" in the final vector (embedding) space but in a much condensed format. This transformed/embedded information can then be used by standard ML models as part of decision making.

What's wrong with just using the standard graph analytic algorithms directly?

Nothing necessarily, however executing standard graph analytic algorithms, especially on large graphs, can involve high computational and space requirements where costs can skyrocket quickly if you are not careful. Graph embeddings provides an effective and efficient method to take large complex graph representation data and simplify it (mathematically speaking we convert it into a low dimensional representation of the graph) whilst still preserving its important traits or properties.

Figure 1.5 shows us the results of running and visualising a traditional community detection algorithm on Zachary's Karate Club data (left image), with the resulting visualisation of running Node2Vec on the left. We can see that whilst providing some initial help, the auto discovery of the clusters via Node2Vec is not quite as good as we would like.

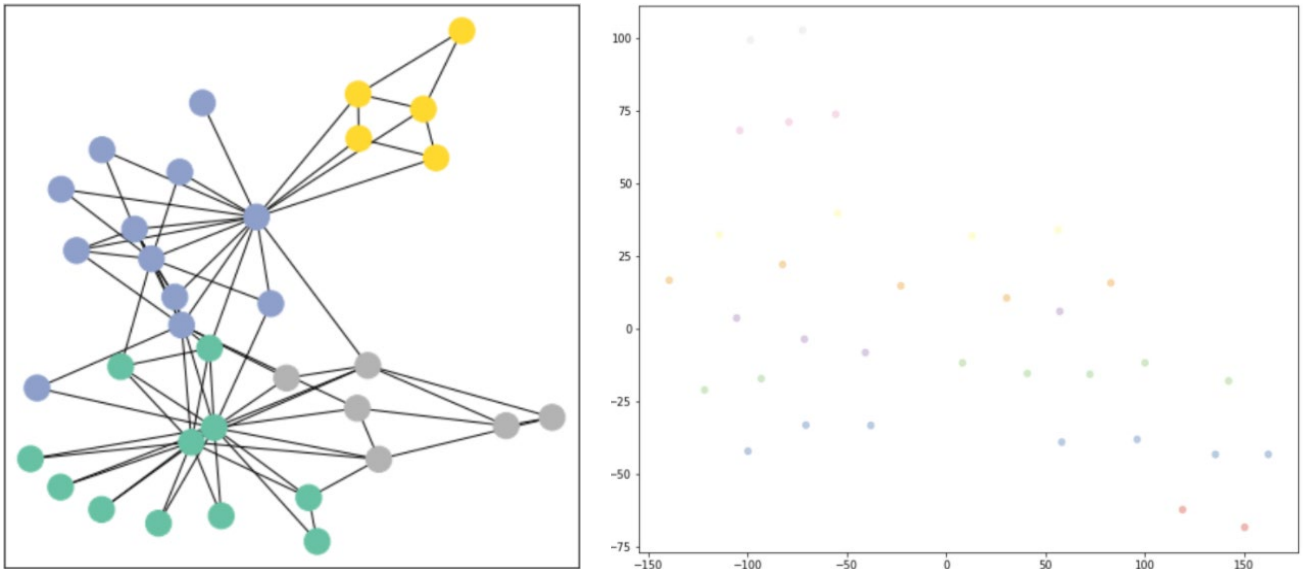


Fig 1.5 Visualisation of Node2Vec results (right image) as run on Zachary's Karate Club data

Node2Vec can be limited, as it only takes a local perspective into account (i.e. walkable local neighbour relationships but no properties). Additionally the assumption that similar nodes are close together as determined by a simple random walk may not necessarily be the most appropriate predictor in some types of graphs.

This is where Graph Neural Networks (GNN) step the learning up another notch. It takes account of both node neighbours, as well as their node properties. Crucially, a GNN's ability to make use of aggregated information from nodes and neighbours, improves precision by virtue of the fact it has more relevant information available for calculation. As per Figure 1.6, we can see the most simplest GNN algorithm GCN can create a better embedding and clustering of the dataset.

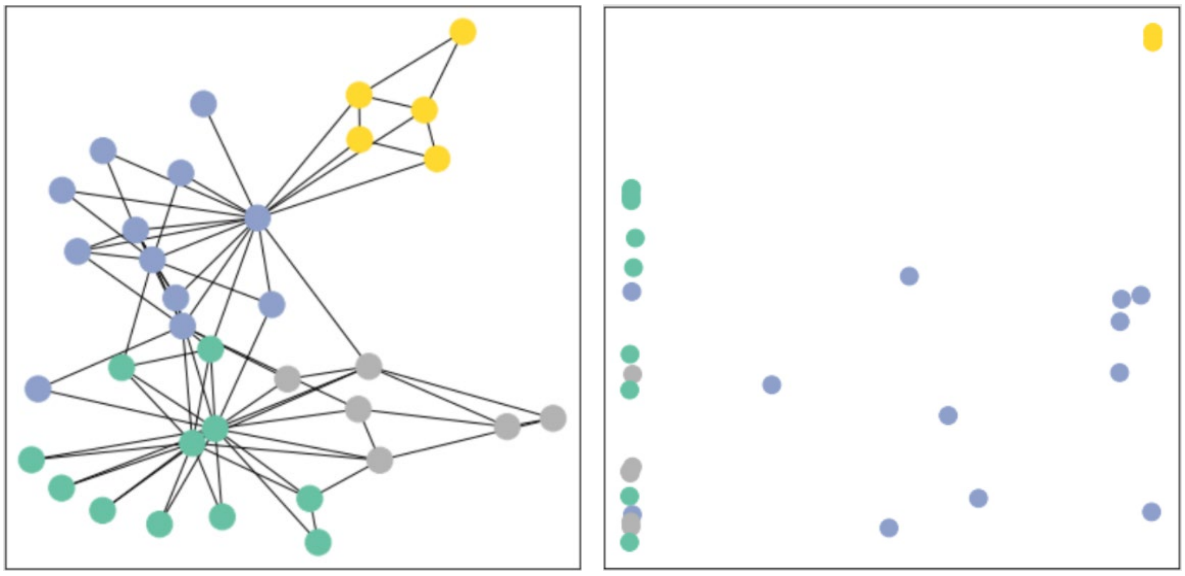


Fig 1.6 Visualisation of GCN based embedding results (right image) as run on Zachary's Karate Club data

GraphSAGE is a recently developed GNN learning algorithm which is *inductive*. This means that it does not rely on a static unchanging graph as input (transductive) each time during training. Instead, it can deal with a changing graph when new nodes are added. The latter is far more common in commercial settings than operating on a static graph which never changes and thus far more preferable. UberEats describe how they saw a 12 percent boost in AUC when using a modified version of GraphSAGE compared to their existing productionised baseline model for making food recommendations to users.

Graph Embeddings can be an advanced topic, if you would like more detail this paper as well as the Stanford University's Machine Learning with Graphs course provides a pretty good starting point.

GRAPH ML CHALLENGES

Similar to traditional data processing, Graph data, from ingestion to examining the resultset, goes through a series of processes and the model's quality is improved with better quality of data. Although the process looks familiar, the problems arise in different forms and shapes, as the representation of a graph requires various perspectives.

#1 Feature Independence

Traditional machine learning models, specifically those based on statistics and requiring supervised human intervention to learn, rely on input features being independent. However, in the graph world, nodes are not independent; they are explicitly linked.

For graphs, we simply need to accept the fact that we need to do things differently because often the best way to predict a type of node, or link between nodes IS to explicitly use information from surrounding neighbours to help us. For example, being able to infer that nodes closer together tend to be of the same type, or that two nodes tend to be linked if many of their fellow nodes are linked. As many of the standard ML independence assumptions break down when working with graphs, we basically convert our process from a "supervised" one to a "semi-supervised" or unsupervised one, allowing us to take advantage of this. For example the use of Node2Vec relies on a conditional independence assumption.

#2

Balancing Issues

With traditional machine learning algorithms, we implement standard checks to ensure the data is of good quality. Specifically, if we are looking at a supervised classification problem, i.e. we want to find out the group of a given record, we will want to determine whether the dataset is ***balanced***. This is important to ensure we don't use overly biased data for training and testing our model leading to skewed results. Taking the more simple case of binary classification (figuring out which of two groups an entity belongs to), we would want to ensure each group has the correct percentage of representation in the dataset, which we will use to train and test on. When we find imbalances, we often apply imputation techniques (the process of replacing missing data with substituted values) such as oversampling and undersampling to correct this.

The quality of the model depends on the quality of the input data. With graphs, classification algorithms suffer from the classic problem of the imbalanced dataset. The input dataset does not have a similar sample ratio when comparing the positive predictions and negative predictions. Standard solutions are either oversampling, where we add more sample data to the minority group or undersample, where we remove some data from the majority group. With Graphs, balancing data with oversampling can be done via GraphSMOTE, a recent algorithm to balance the dataset.

It synthesises new genuine nodes, similar to SMOTE (Synthetic Minority Over-sampling Technique) used in traditional ML algorithms. Also, the GRAPE framework generates the observations and features as two types of nodes in a bipartite graph and the observed feature values as edges. The feature imputation is an edge-level prediction task, and the label prediction is a node-level prediction task.

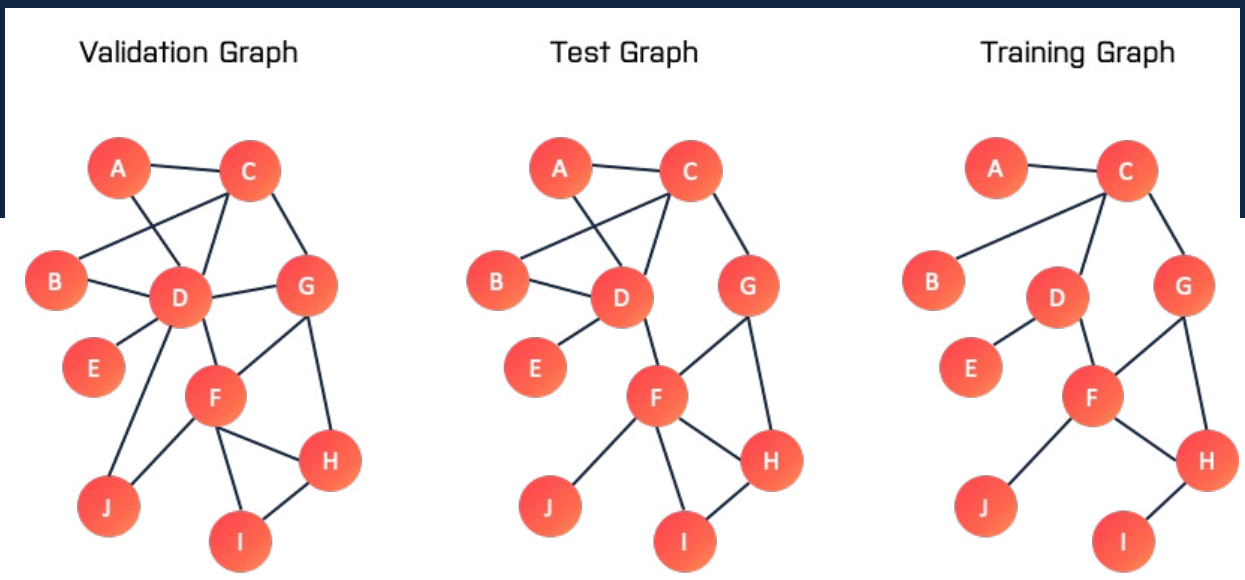
#3

Splitting Datasets

Traditional supervised machine learning algorithms split the dataset into a minimum of 2 sets: the training dataset and the test dataset. The algorithm with initial hyperparameters runs on the training set and fits the model with this data. Then, using the formula it has created, it evaluates the test data to understand how good the method is to predict the ground-truth testing dataset, with the help of the metrics as accuracy, which we will talk about in detail later in the Model Scoring section.

For Graph, splitting the data can be challenging, as it does not have a standard structure. But the good news is that there are some proven ways we can achieve this. The first one is to give new labels to the nodes and assess whether we can predict the nodes correctly. This method does not rely on any information declared via the connections but uses the properties of the node, which is a simple node classification. For an auction dataset, an example would be; we can split it into sellers, buyers, and seller-buyers.

The second method is, if/when we have more insight about the connectivity, we can split it into multiple subgraphs. If we have a timestamp of the links (the graph would then be called a time-variant graph), we can divide the graph with new and old connections. If we don't have time variance, we can split by removing different training and test validation links depending on the domain.



Graph MLOps

To truly be able to harness the power of graphs and ML, you need to have a predictable, reliable (automated) way of updating, looping and feeding back into the graph ML lifecycle. MLOps, the process of automating and productionising your ML model and lifecycle, deserves a book all on its own, let alone when you throw graphs into the mix. Whilst we aren't able to go into detail here, we do hope to cover this topic in more detail at a later date, so stay tuned for more in this space!

CONCLUSION

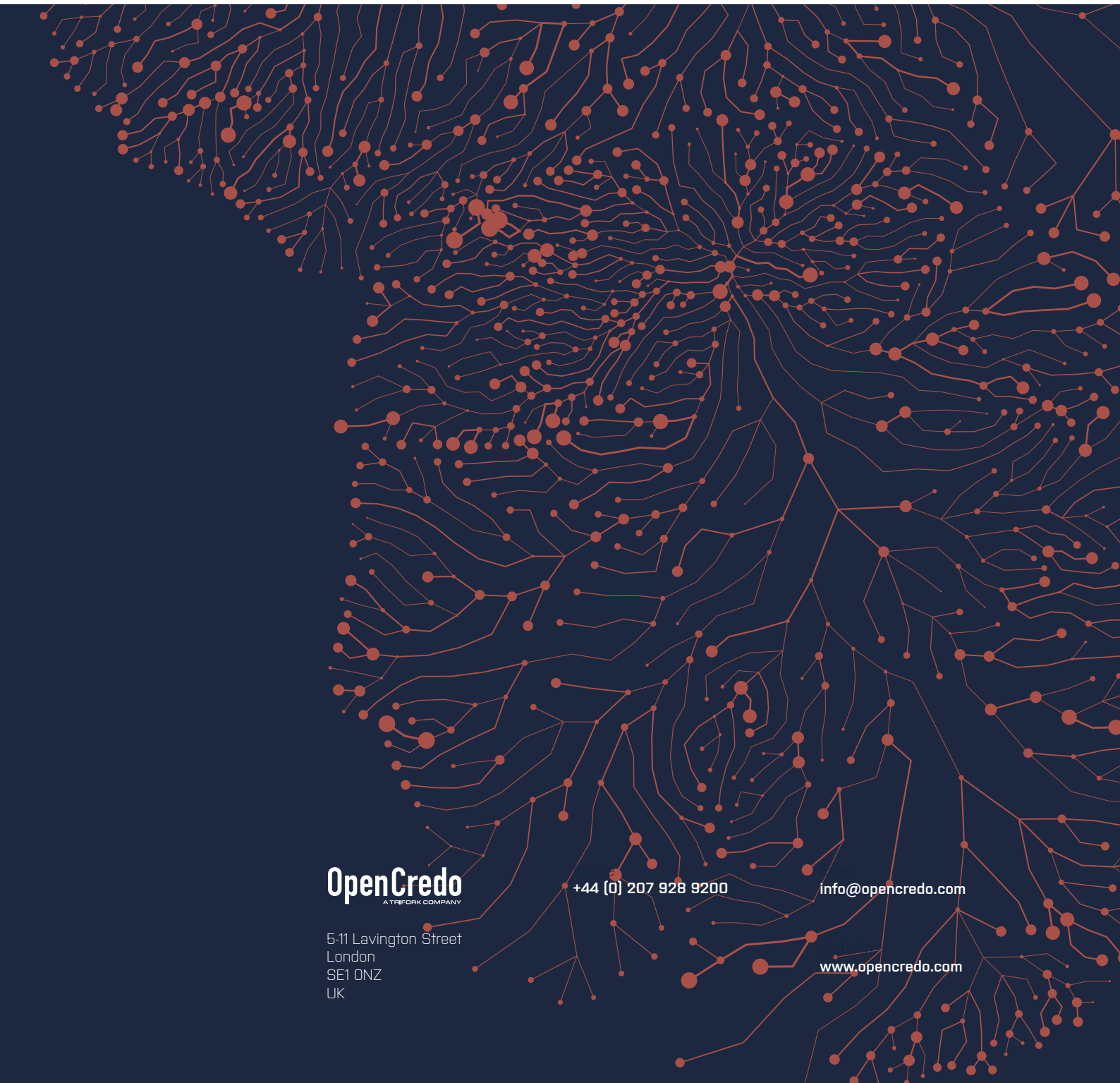
We hope this book has given you a sense for how you are able to improve your machine learning endeavours by incorporating information about the connections inherent in your problem domain in your models. We've seen how even relatively simple approproaches, such as including the results of graph algorithms in existing ML models, can improve both accuracy and predictive capability. But going beyond this, there are opportunities to leverage some of the more state-of-the art approaches and technologies available such as Graph Neural Networks (GNNs) which offer even more improvements. There are of course some unique challenges involved in trying to harness the power of graphs for ML, and one should be aware of these. But overall, even with such challenges, we believe the effort is worth it for the improvements on offer!

We hope to see you soon in the hybrid world of graphs and ML. And if you have a connected data problem and are looking for help, we are more than happy to chat and discuss your options with you - so please do reach out!

Petar Veličković, Senior Researcher at DeepMind:

"2020 has definitively and irreversibly turned graph representation learning into a first-class citizen in ML."

Connecting the Dots:
Harness the Power of Graphs & ML



OpenCredo
A TRIFORK COMPANY

5-11 Lavington Street
London
SE1 0NZ
UK

+44 (0) 207 928 9200

info@opencredo.com

www.opencredo.com