Comprehensive Overview on Graph Database Formulation and Feature Engineering for AI/ML Solutions

March 11, 2025

Contents

1 Introduction

In modern data ecosystems, organizations struggle to derive actionable insights from vast amounts of unstructured and semi-structured data. Traditional relational models lack the flexibility to represent complex relationships. This document explores a graph-based approach using Neo4j to convert unstructured data into actionable knowledge graphs and perform advanced feature engineering for AI/ML pipelines.

2 Business Context and Problem Statement

Healthcare and service industries generate enormous unstructured datasets from:

- Member call logs and transcripts
- Claims and appeals
- Surveys and feedback
- Provider interactions

The challenge lies in integrating this disparate data to predict unfavorable member responses. A graph-based model enables a connected view to understand member behavior and associated risks.

3 Solution Architecture

3.1 Overview

1. **Data Ingestion and ETL**: Extract, clean, and transform data from centralized repositories.
2. **Graph Database Formulation**: Load structured/unstructured data into Neo4j Aura via Apache Spark.
3. **Feature Engineering**: Extract node and graph-level features using Cypher queries.
4. **Model Development Pipeline**: Feed structured datasets into AI/ML models.
5. **Prediction and Insights**: Predict member risk and deliver actionable insights.

4 Graph Database Formulation

4.1 Core Entities (Nodes)

- **Member**: MemberID, Demographics, RiskScore
- **Call**: CallID, Timestamp, SentimentScore, Reason
- **Claim**: ClaimID, Date, Amount, Status

- **Appeal**: AppealID, Date, Reason, Status
- **Survey**: SurveyID, Score, FeedbackType
- **Provider**: ProviderID, Specialty, Location
- **Plan**: PlanID, Type, Premium

4.2 Relationships (Edges)

- `(Member)-[:CALLS]->(Call)`
- `(Member)-[:FILED]->(Claim)`
- `(Member)-[:FILED]->(Appeal)`
- `(Member)-[:RECEIVED]->(Survey)`
- `(Member)-[:ASSOCIATED_WITH]->(Provider)`
- `(Claim)-[:HANDLED_BY]->(Provider)`
- `(Member)-[:ENROLLED_IN]->(Plan)`

4.3 Example Graph Representation

```
(Member)-[:CALLS]->(Call)
(Member)-[:FILED]->(Claim)-[:HANDLED_BY]->(Provider)
(Member)-[:FILED]->(Appeal)
(Member)-[:RECEIVED]->(Survey)
(Member)-[:ENROLLED_IN]->(Plan)
```

5 Feature Engineering using Cypher

5.1 Node-Level Features

- **Total Calls**:

```
MATCH (m:Member)-[:CALLS]->(c:Call) RETURN m.MemberID, COUNT(c) AS TotalCalls
```

- **Average Sentiment Score**:

```
MATCH (m:Member)-[:CALLS]->(c:Call) RETURN m.MemberID, AVG(c.SentimentScore) AS A
```

- **Total Appeals Filed**:

```
MATCH (m:Member)-[:FILED]->(a:Appeal) RETURN m.MemberID, COUNT(a) AS TotalAppeals
```

- **Denied Claims**:

```
MATCH (m:Member)-[:FILED]->(cl:Claim {Status:'Denied'}) RETURN m.MemberID, COUNT(
```

5.2 Graph-Based Features

- **Degree Centrality**: Connectedness of members.
- **PageRank**: Influence of members within the network.
- **Shortest Path to Providers**: Path analysis for quick access.
- **Community Detection**: Identifying clusters of similar members.

5.2.1 Example Cypher Query for Combined Features

```
MATCH (m:Member)
OPTIONAL MATCH (m)-[:CALLS]->(c:Call)
OPTIONAL MATCH (m)-[:FILED]->(a:Appeal)
OPTIONAL MATCH (m)-[:FILED]->(cl:Claim {Status:'Denied'})
RETURN m.MemberID, COUNT(c) AS TotalCalls,
       AVG(c.SentimentScore) AS AvgSentiment,
       COUNT(a) AS TotalAppeals,
       COUNT(cl) AS DeniedClaims
```

6 Feature Table for AI/ML Models

MemberID	TotalCalls	AvgSentiment	Appeals	DeniedClaims	Degree	PageRank	Label
M001	5	-0.85	2	1	0.75	0.12	High
M002	2	0.30	0	0	0.40	0.08	Low

7 ML Pipeline for Classification

7.1 Input Features

- Interaction features: Calls, Sentiment, Appeals, Denials
- Graph-based features: Centrality, PageRank, CommunityID

7.2 Labels

- High, Medium, Low likelihood of unfavorable responses

7.3 Models Used

- Random Forest, Gradient Boosting, XGBoost
- Call Sentiment Analysis (Optional BERT/Transformer models)

8 Business Outcomes

- Early identification of high-risk members for intervention.
- Improved call center performance and reduced repeat calls.
- Targeted member engagement and case management.
- Provider performance monitoring based on claims and appeals.

9 Conclusion

Graph-based modeling provides a powerful way to organize unstructured data into connected insights. By leveraging Neo4j and graph feature engineering, organizations can effectively predict and mitigate member risks, improve service quality, and optimize resource allocation.