

**Can you tell about concrete graph features which can be fed into the ml model**

Here are concrete graph features you can extract from your Neo4j graph to enhance fraud detection ML models, supported by industry implementations and research:

**1. Centrality Metrics**

**Betweenness Centrality**:

* Identifies accounts/nodes acting as bridges in money laundering schemes[[1]](#fn1)
* High scores indicate entities facilitating suspicious transaction flows

**PageRank**:

* Flags accounts with unusual transaction influence (e.g., synthetic identities funneling funds)[[1]](#fn1)[[2]](#fn2)
* Example: Accounts with >95th percentile PageRank scores in your network

**Degree Centrality**:

* Measures direct connections (e.g., accounts linked to >10 devices/emails)[[3]](#fn3)

**2. Community Features**

**Louvain Community ID**:

* Assigns cluster IDs to detect fraud rings (e.g., 15 accounts sharing devices in Cluster #X)[[1]](#fn1)[[3]](#fn3)

**WCC (Weakly Connected Component) Size**:

* Large WCCs (>50 nodes) may indicate coordinated attacks[[1]](#fn1)
* Used in XGBoost models to improve accuracy by 14%[[1]](#fn1)

**Cluster Density**:

* Fraudulent communities often show higher-than-average connection density[[3]](#fn3)

**3. Shared Attribute Patterns**

**Device/SSN/Email Reuse**:

* Count of accounts per device: MATCH (d:Device)<-[:USES]-(a) RETURN d.id, count(a)[[3]](#fn3)
* Flag devices linked to >3 accounts in 7 days

**Geographic Anomalies**:

* Transactions from IP/Location inconsistent with account history[[4]](#fn4)

**4. Temporal Graph Features**

**Relationship Velocity**:

* New connections formed/hour (e.g., account adding 5 devices in 1 hour)[[2]](#fn2)
* Transaction burst detection: CALL temporal.patterns.suddenIncrease()

**Historical Link Analysis**:

* Paths reactivating dormant accounts (e.g., fraudster reusing old SSN-email pairs)

**5. Path-Based Signals**

**Shortest Path to Known Fraud**:

* MATCH path=shortestPath((a)-[\*..3]-(fraud))[[1]](#fn1)
* Accounts within 2 hops of flagged entities get +0.3 risk score

**Transaction Chain Depth**:

* Multi-hop money trails (e.g., A→B→C→D within 10 minutes)[[4]](#fn4)

**6. Graph Embeddings**

**FastRP Embeddings**:

* 128-dimension vector representations of account/device nodes[[2]](#fn2)
* Captures latent relationship patterns for ML input

**Node2Vec Walks**:

* Embeddings preserving structural equivalence (e.g., similar "mule account" roles)

**Implementation Example (Neo4j Cypher/GDS):**

// Calculate PageRank for accounts  
CALL gds.pageRank.stream('account-network', {maxIterations: 20})   
YIELD nodeId, score  
SET gds.util.asNode(nodeId).pagerank = score  
  
// Extract device-sharing features  
MATCH (a:Account)-[:USES]->(d:Device)  
WITH d.id AS device, count(a) AS users  
WHERE users > 3  
RETURN device, users AS risk\_feature

**Impact Validation (From Sources):**

|  |  |  |
| --- | --- | --- |
| Feature Type | Model Improvement | Source |
| PageRank + WCC | +14% accuracy in XGBoost | [[1]](#fn1) |
| Community Detection | 87.5% faster fraud ring identification | [[3]](#fn3) |
| Graph Embeddings | 50% reduction in false positives | [[2]](#fn2) |

These features overcome tabular limitations by modeling multi-hop relationships and dynamic networks - critical for detecting modern fraud patterns like synthetic identity fraud[[4]](#fn4) and bust-out schemes[[5]](#fn5).

⁂

1. <https://www.tigergraph.com/blog/using-graph-machine-learning-to-improve-fraud-detection-rates/>

1. <https://linkurious.com/blog/fraud-detection-machine-learning-graph/>

1. <https://engineering.grab.com/graph-for-fraud-detection>

1. <https://aws.amazon.com/neptune/fraud-graphs-on-aws/>

1. <https://github.com/safe-graph/graph-fraud-detection-papers>