WEBINAR

Boost ML-Powered Recommendation Engine & Fraud Detection Workloads

Kevin Phillips (he/him)

Amazon Neptune Specialist Solutions Architect

What’s On Today?

- Introducing Graph Databases
- Amazon Neptune
- Machine Learning on Graph
- Neptune ML
- Using Neptune for Fraud Detection
  - Demo: Performing real-time inference
- Summary
- Resources

Introducing Graph Databases

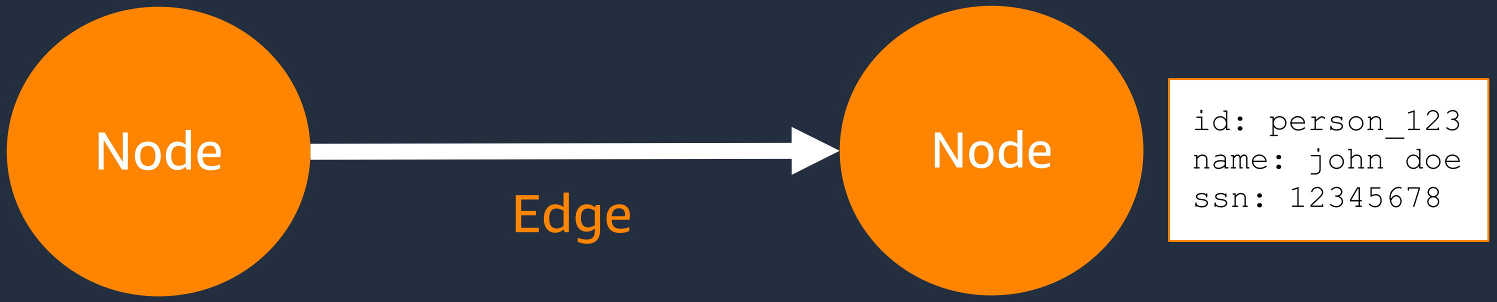
What is a graph?

Graphs are purpose-built to store and navigate relationships

Nodes represent real-world objects

Edges store relationships between objects

Properties and labels can be added to both nodes and edges



Technical challenges that graphs help solve

Combining data across data silos

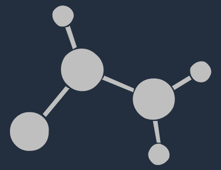
Finding common connections or paths

Technical challenges that graphs help solve

Combining data across data silos

Finding common connections or paths

Working with heterogeneous data with complex relationships



Data full of many-to-many relationships

Amazon Neptune

Amazon Neptune

Open

Supports Apache TinkerPop & W3C RDF graph models

Fast

Query billions of relationships with millisecond latency

Reliable

6 replicas of your data across 3 AZs with full backup and restore

Flexible

Build powerful queries easily with Gremlin, openCypher, and SPARQL

(Some) Service integrations

AWS OpenSearch Service

AWS Athena

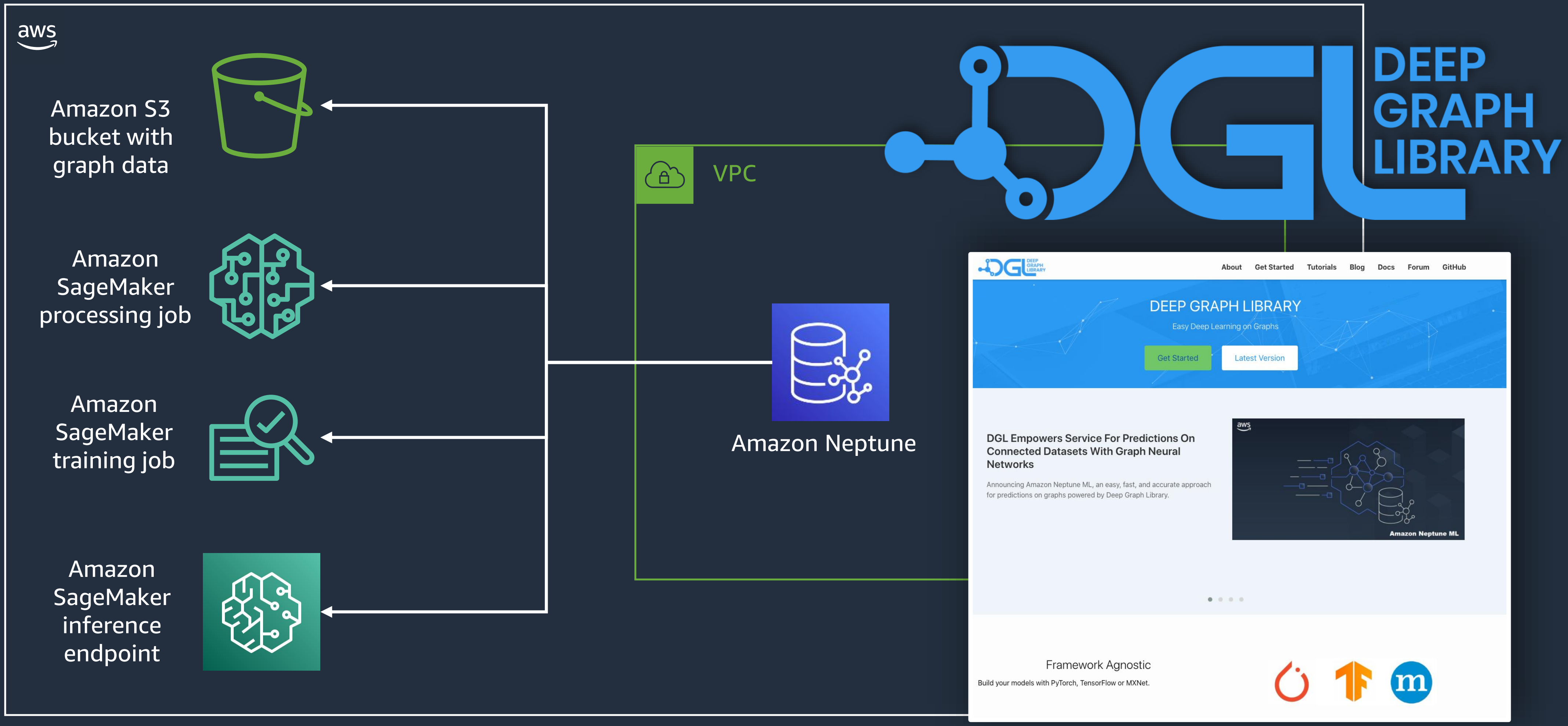
AWS Glue

Amazon SageMaker

AWS Database Migration Service

Amazon CloudWatch

Neptune and SageMaker: Neptune ML



Machine Learning on Graph

Why Machine Learning on graph data?

Graphs provide machine learning with novel features based on connections within the data

Leverage connectedness

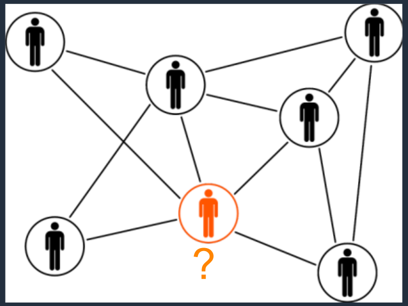
Unique insights and increased prediction accuracy leveraged by ML to detect patterns deep into data connections

Adapt over time

Common ML use-cases such as fraud, require the flexibility provided by graphs to model data and patterns as they evolve

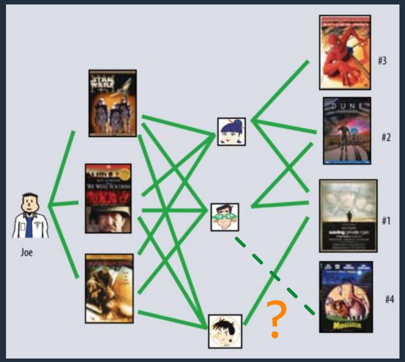
See data at scale

Graphs and graph embeddings provide a unique ability to efficiently represent data with billions of relationships



Node Classification & Regression

Customer targeting, malicious transactions, etc.



Link Prediction

Recommendations, hidden relationships, etc.

Why Machine Learning on graph data?

Graphs provide machine learning with novel features based on connections within the data

Leverage connectedness

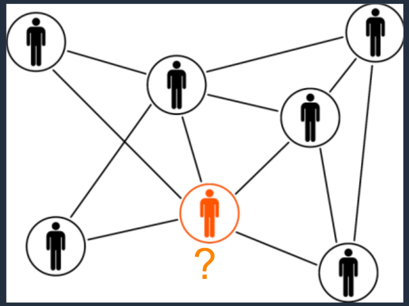
Unique insights and increased prediction accuracy leveraged by ML to detect patterns deep into data connections

Adapt over time

Common ML use-cases such as fraud, require the flexibility provided by graphs to model data and patterns as they evolve

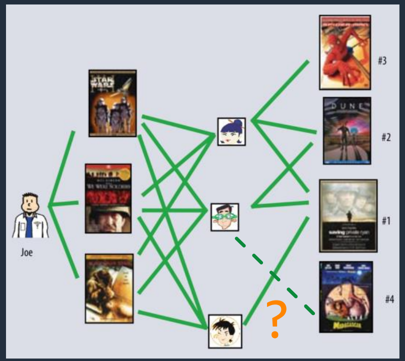
See data at scale

Graphs and graph embeddings provide a unique ability to efficiently represent data with billions of relationships



Node Classification & Regression

Customer targeting, malicious transactions, etc.



Link Prediction

Recommendations, hidden relationships, etc.

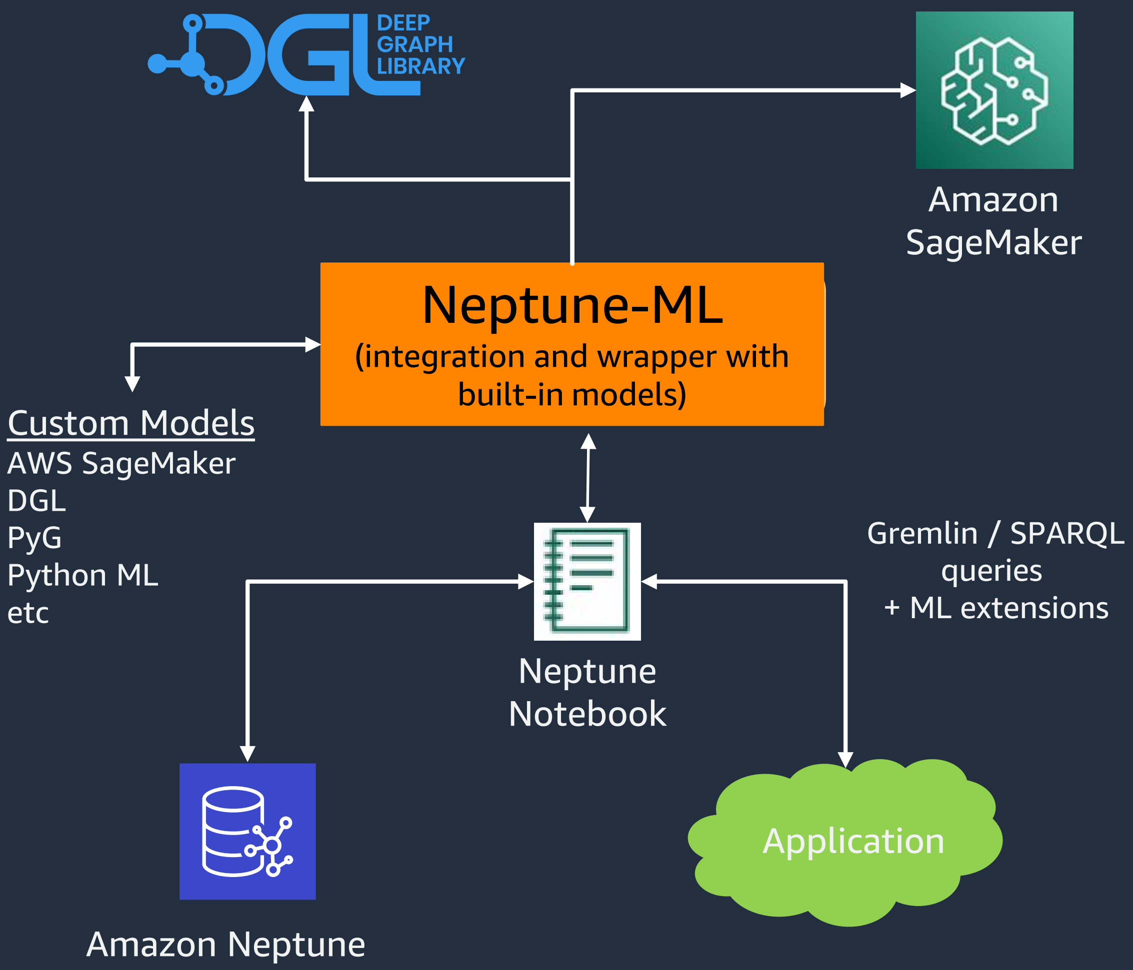
Traditional Path for Graph ML Solution

Building a ML solution for graph data requires:

- Experts in graph DBs, deep learning frameworks, productionizing models, etc.
- Export data from graph database
- Perform feature engineering
- Develop and iterate/innovate models
- Build infrastructure
- Select and deploy the best model
- Automate the process to adjust to dynamic data

Repetitive workflow requires sequence of manual steps and is work intensive

Graph Machine Learning simplified by Neptune ML



Graph predictions using Neptune ML

Automatically choose and train the best ML model
Seamless integration with AWS SageMaker and DGL

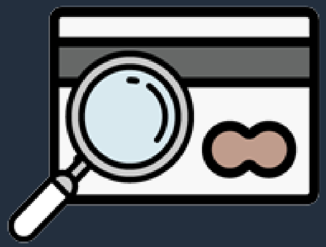
Use state-of-the-art graph ML

Higher accuracy by using GNNs embeddings and inference, leveraging the relationships and features in graphs

Scale runtime inferences to large graphs

Infer data within the query mid-traversal across the billions of nodes/relationships

Use-cases for Neptune Machine Learning



Fraud Detection

Product Recommendations

Identity Resolution

Knowledge Graph

Neptune ML

Amazon Neptune ML/GNN Workflow

Adds machine-learning based inference in 4 simple steps

Amazon Neptune

Amazon S3

Data Export &
Feature Engineering

Neptune-ML with

DEEP GRAPH LIBRARY

Build & Train
GNN ML
Model

Amazon Neptune ML/GNN Workflow

Adds machine-learning based inference in 4 simple steps

Amazon Neptune   Amazon S3

Data Export & Feature Engineering

Neptune-ML with DEEP GRAPH LIBRARY

Build & Train GNN ML Model

Neptune-ML with Amazon SageMaker APIs

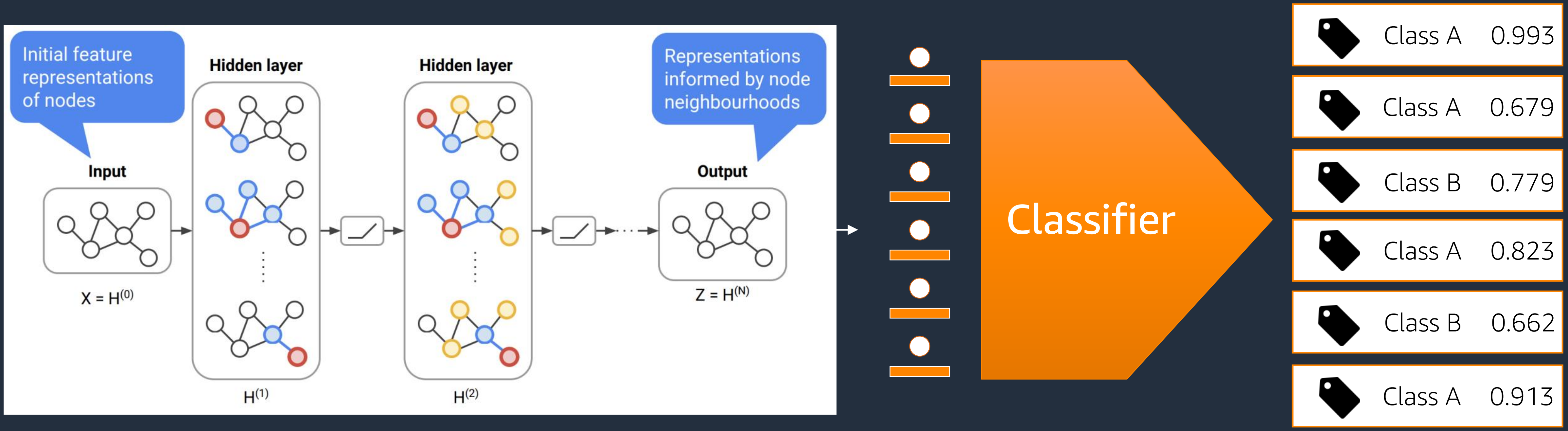
Create Inference Endpoint

SPARQL

Infer during Traversal

What are Graph Neural Networks (GNNs)?

A family of deep neural networks that learn inductive node embeddings



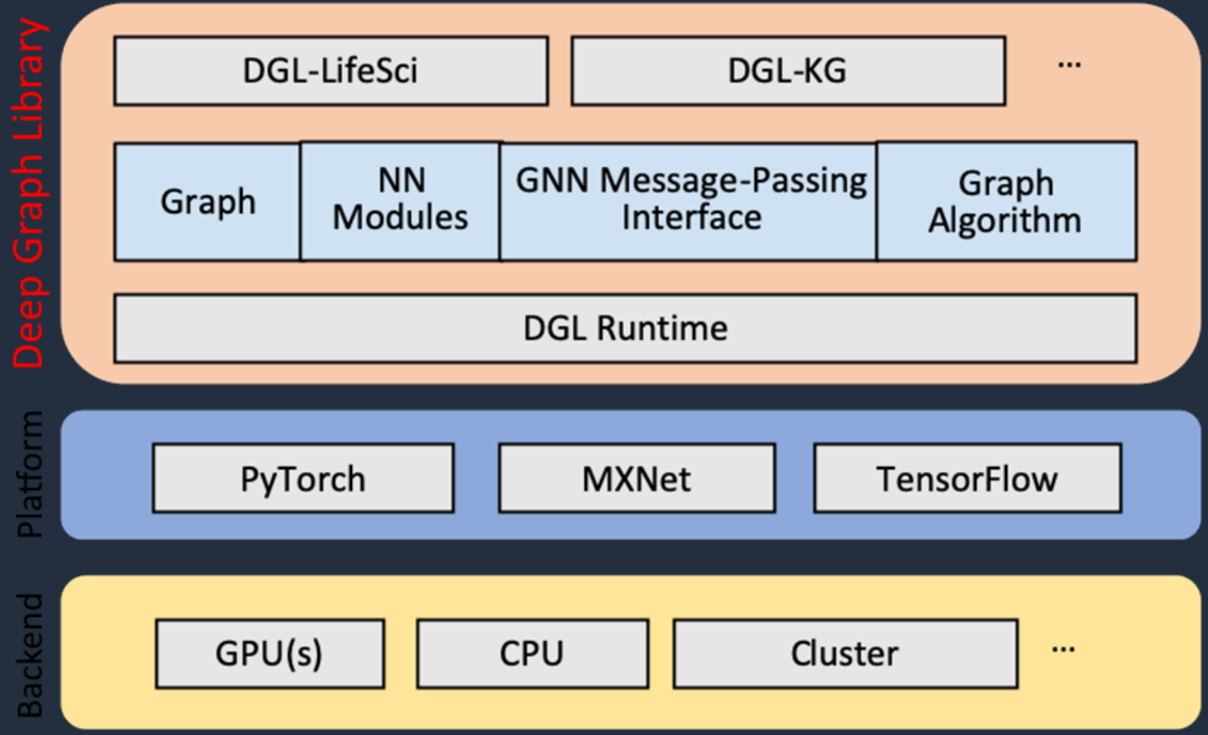
Compute embeddings using graph structure and node/edge features

Nonlinearly integrates topologically distant information

GNNs can be trained end-to-end

Deep Graph Library (DGL) powers Neptune ML

Easy deep learning on graphs

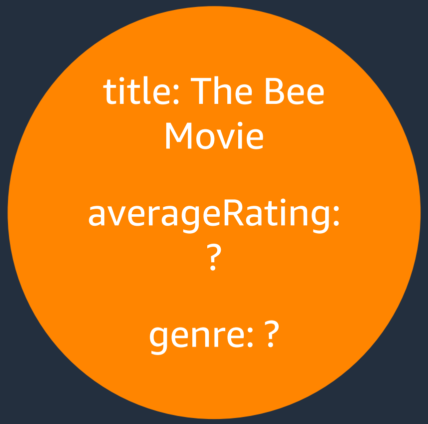


- Easy to develop new models
- Seamless integration with existing frameworks
- Fast and scalable
- Use built-in NN modules for popular GNN models e.g. GraphSage, GCN, RGCN, etc.

What can you do with Neptune ML?

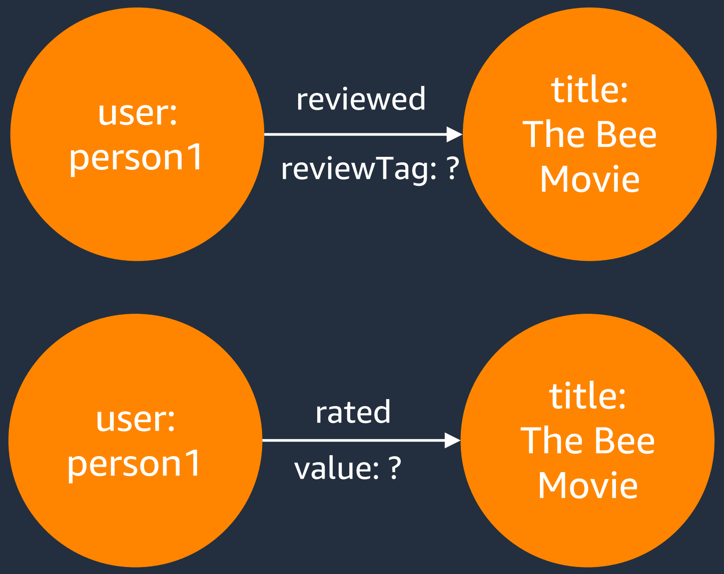
Node Prediction

Predict the values of categorical or numerical properties for nodes



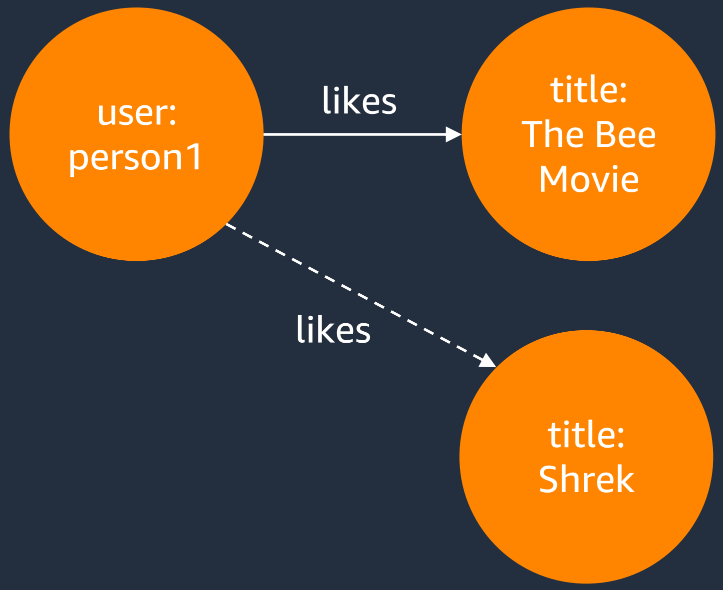
Edge Prediction

Predict the values of categorical or numerical properties for edges



Link Prediction

Predict the connections between nodes



What inferencing options do I have?

Transductive inference

Nodes/edges being queried for prediction were in the graph at the time of model training

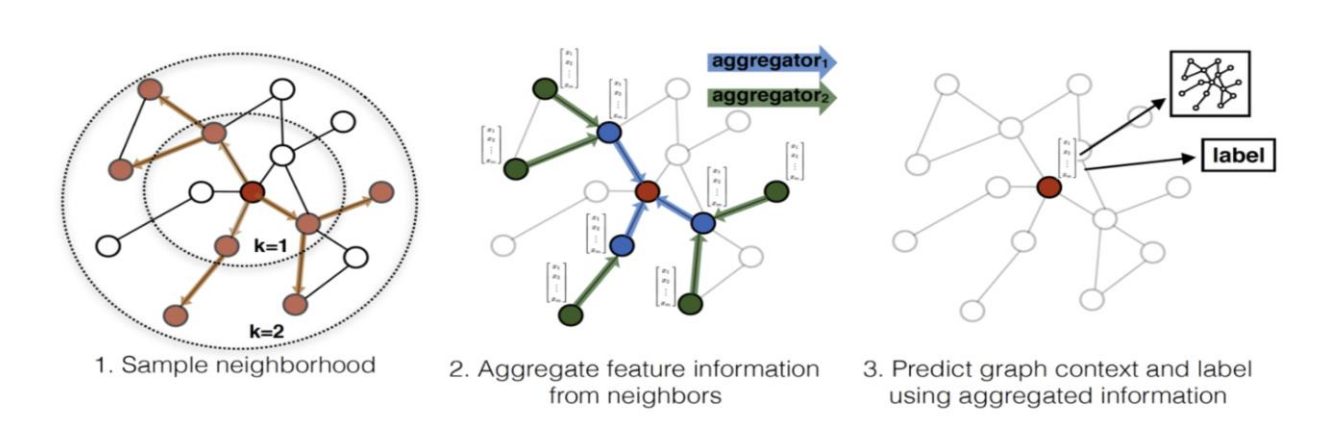
What inferencing options do I have?

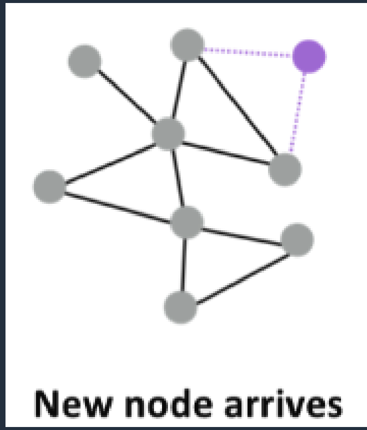
Transductive inference

Nodes/edges being queried for prediction were in the graph at the time of model training

Inductive inference

Enable predictions on nodes, edges and properties that were added to the graph after the ML model training process.





Using Neptune ML for Fraud Detection

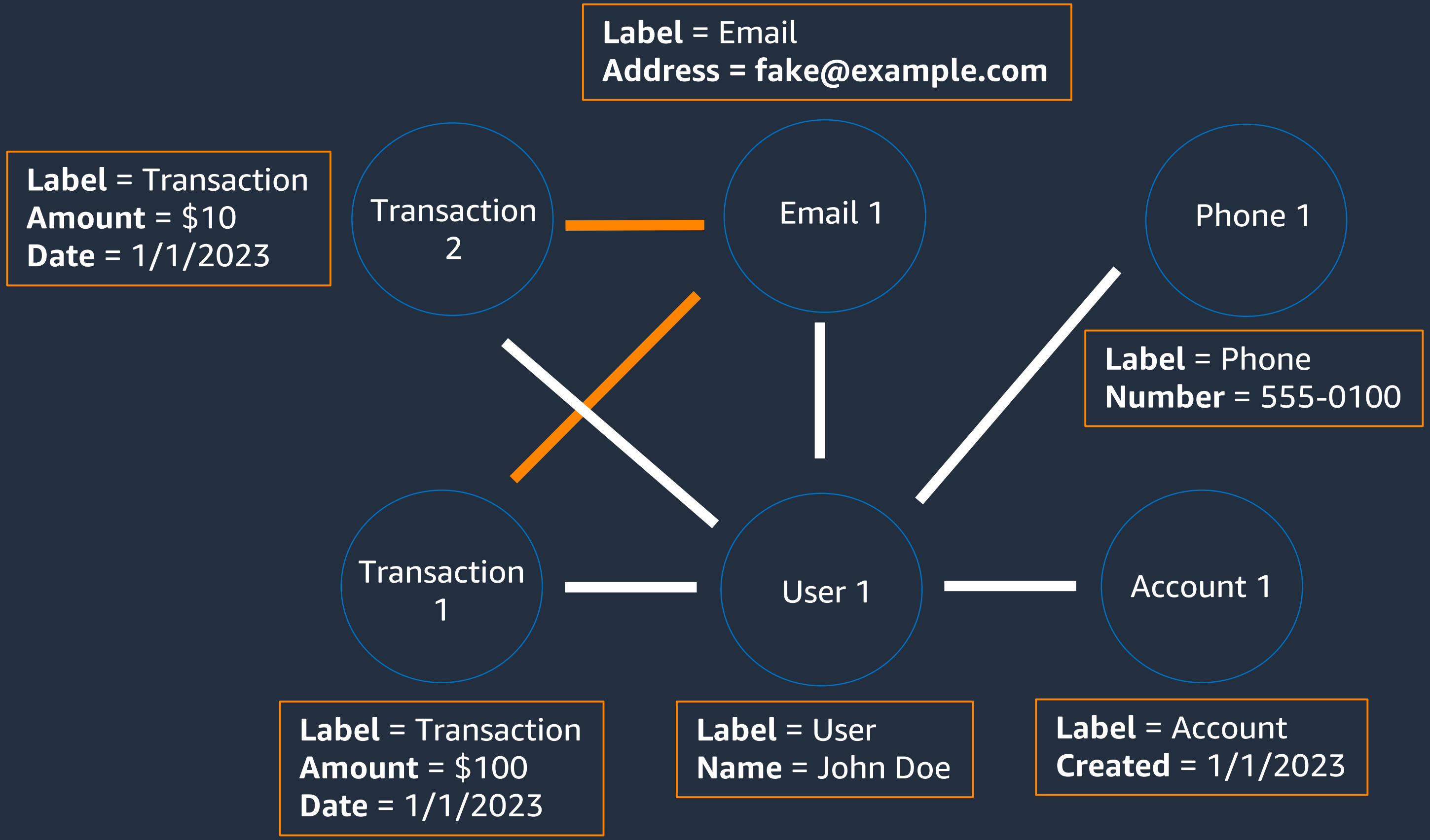
A fraud graph model

Conceptual fraud model

A fraud graph model

Conceptual fraud model

Fraud graph model



Fraud Detection with Neptune ML & GNN

Identify source datasets

```
%%gremlin
g.V().groupCount().by(label).unfold().order().by(keys)
```

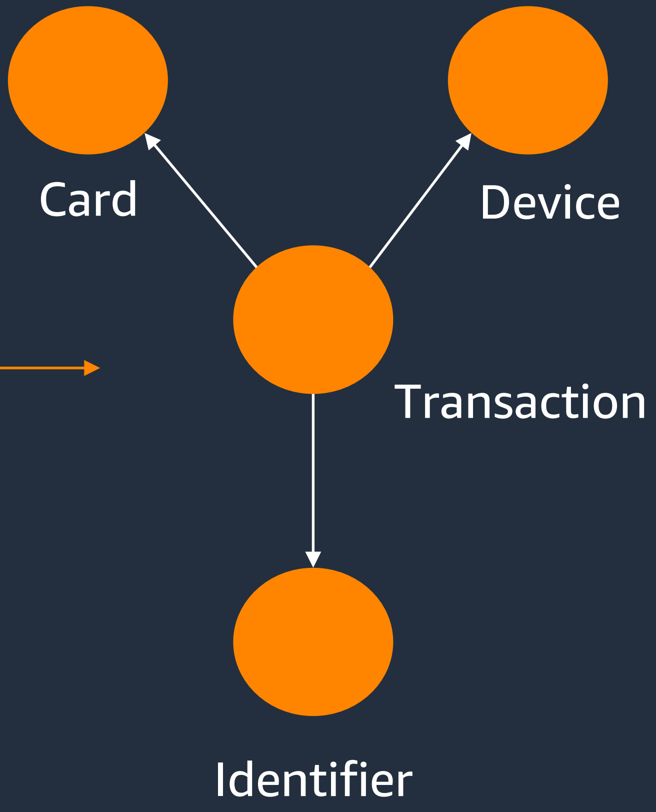
The output should contain:

- 406 Addresses
- 14286 Cards
- 1788 Devices
- 60 Emails
- 119193 Identifiers
- 5 Products
- 590540 Transactions

Using IEEE CIS Dataset (Anonymised Fraud Dataset)

Data Source:
https://www.kaggle.com/c/ieee-fraud-detection/data

Use only the data needed to answer the business questions

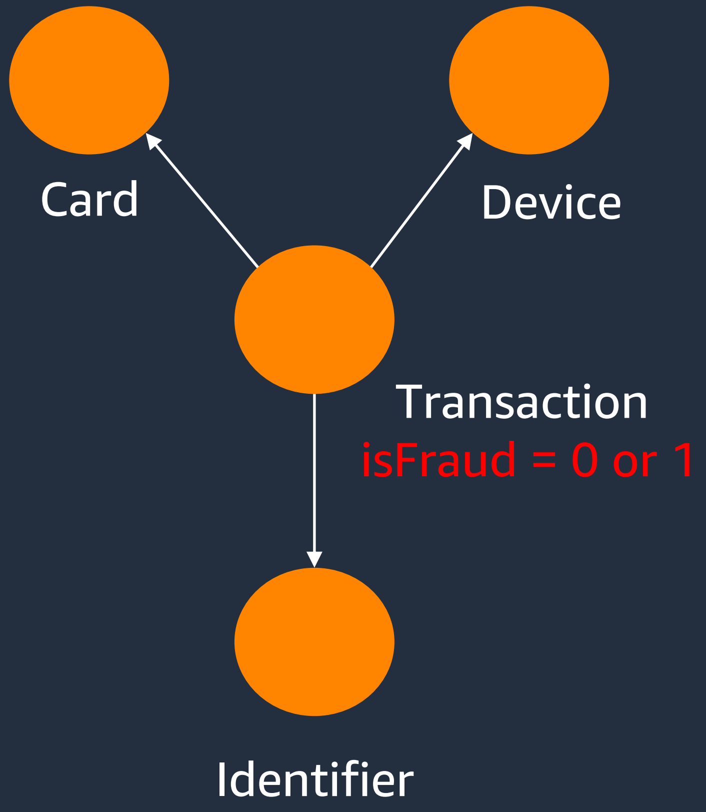


Fraud Detection with Neptune ML & GNN

Define Machine Learning Prediction Target

Identifying the prediction target

The data already contains 1.5% of fraudulent labelled transactions – defined by an “isFraud” property of 0 or 1.



Fraud Detection with Neptune ML & GNN

Define Machine Learning Prediction Target

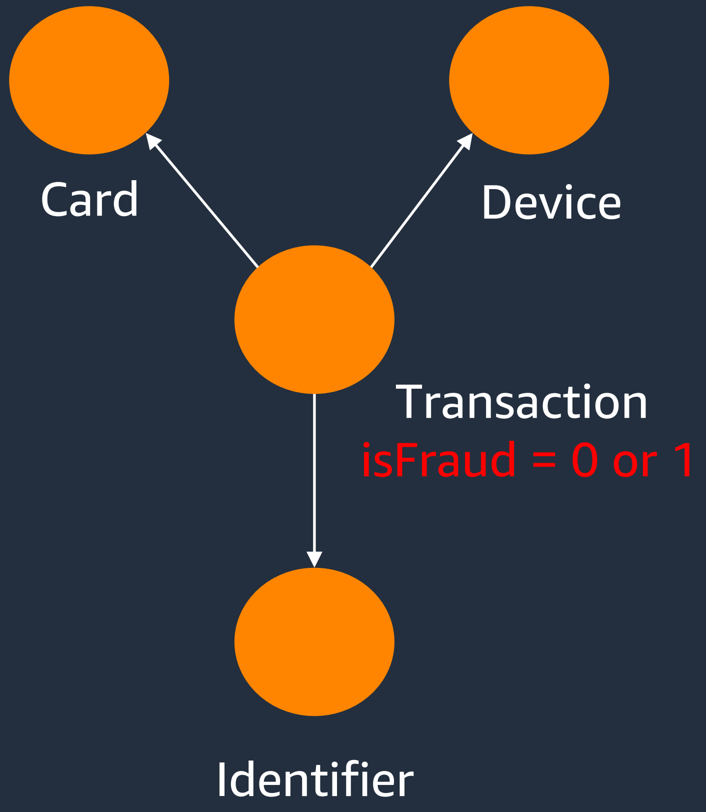
Identifying the prediction target

The data already contains 1.5% of fraudulent labelled transactions – defined by an “isFraud” property of 0 or 1.

Prediction target used for training

Classify the nodes labelled as “Transaction” with the property “isFraud”

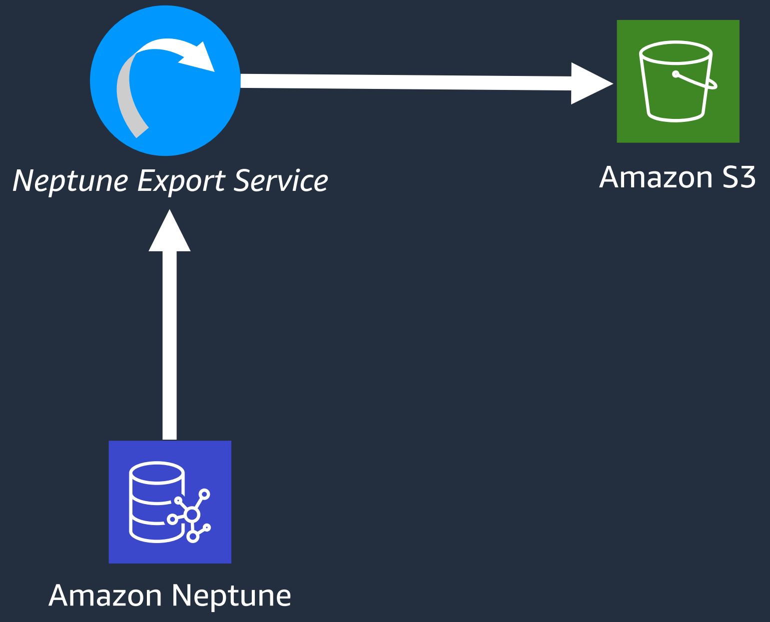
```
"neptune_ml": {
   "version": "v2.0",
   "targets": [{
       "node": "Transaction",
       "property": "isFraud",
       "type": "classification"
     }]
}
```



Export Data and Configuration

Neptune ML automates the data export and configuration generation process

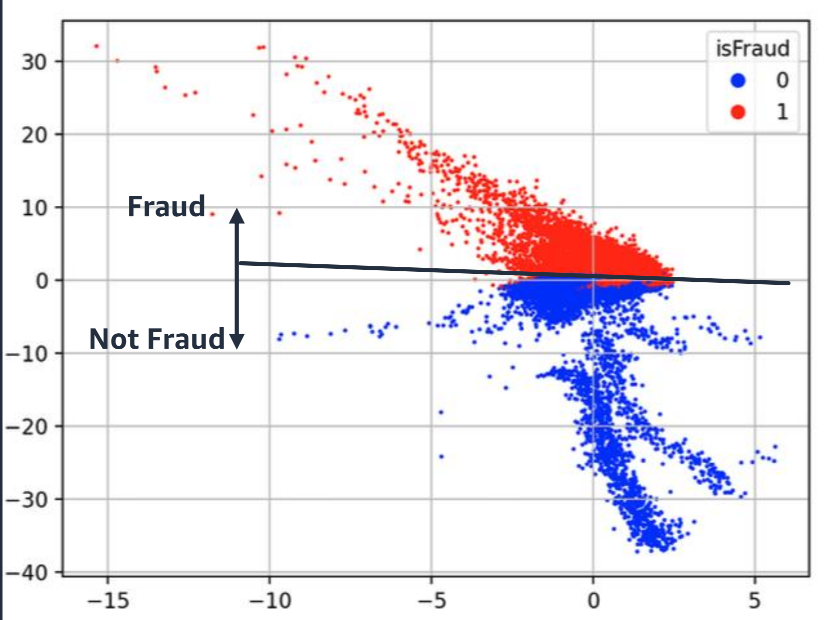
```
export_params={
    "command": "export-pg",
    "params": {
        "endpoint": neptune_ml.get_host(),
        "profile": "neptune_ml",
        "useIamAuth": neptune_ml.get_iam(),
        "cloneCluster": True,
        "cloneClusterInstanceType": "r5.12xlarge",
        "nodeLabels": ["Transaction", "Identifier", "Device", "Card"],
        "edgeLabels": ["identified_by", "purchased_by", "associated_with"]
    },
    "outputS3Path": f'{s3_bucket_uri}/neptune-export',
    "additionalParams": {
        "neptune_ml": {
            "version": "v2.0",
            "targets": [
              {
                "node": "Transaction",
                "property": "isFraud",
                "type": "classification"
              }
            ]
        }
    },
    "jobSize": "medium"
}
```



Fraud Detection with Neptune ML & GNN

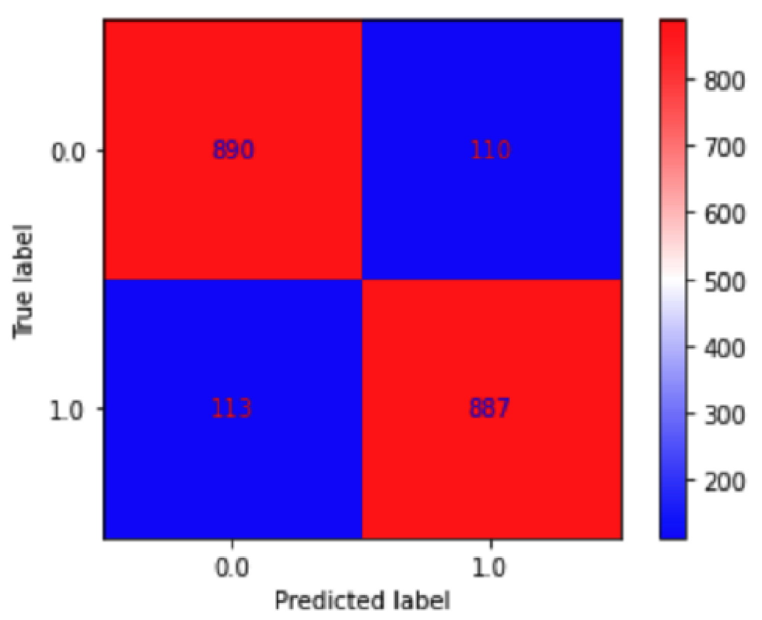
Resulting ML model and evaluating performance

Resulting Embeddings



570K non-fraud transactions
20K fraud transactions

Confusion Matrix




```
overall accuracy = 88.8%
correctly predicted legit = 89.0%
correctly predicted fraud = 88.7%
```

Fraud Detection with Neptune ML & GNN Example

Transductive prediction queries via Neptune Gremlin query language

Predict isFraud for "transaction" node "3001822"

```
g.with('Neptune#ml.endpoint', '${endpoint}')
    .V('3001822').properties('isFraud', 'Neptune#ml.score')
    .with('Neptune#ml.classification').value()
```

→ Result: 0, 0.644

Predict isFraud for all "transaction" nodes

```
g.with('Neptune#ml.endpoint', '${endpoint}')
.V().hasLabel('Transaction').hasNot('isFraud')
.properties('isFraud').with('Neptune#ml.classification').value()
```

Now, all the "Transaction" nodes have the label isFraud with value 0 or 1.

Fraud Detection with Neptune ML & GNN Example

Inductive prediction queries via Neptune Gremlin query language

Add a new transaction node

```
g.addV('Transaction').property(T.id, '9999999’)
            .property('id_01', -5.0)
            .property('id_02', 222578.0)
            ...
            .property('id_12', 'NotFound’)
            .property('id_15', 'New’)
            .property('id_16', 'NotFound’).as('newT’)
.addE('associated_with’)
            .from('newT’)
            .to(V('DeviceType:mobile’))
            .property(T.id, 'transaction_9999999-associated-with-mobile’)
.addE('purchased_by’)
            .from('newT’)
            .to(V('card3:150.0’))
            .property(T.id, 'transaction_9999999-purchased-by-card’)
.addE('identified_by’)
            .from('newT’)
            .to(V('id_05:0.0’))
            .property(T.id, 'transaction_9999999-identified-by-id_05’)
.addE('identified_by’)
            .from('newT’)
            .to(V('id_38:F’))
            .property(T.id, 'transaction_9999999-identified-by-id_38')
```

Fraud Detection with Neptune ML & GNN Example

Inductive prediction queries via Neptune Gremlin query language

Add a new transaction node

```
g.addV('Transaction').property(T.id, '9999999’)
            .property('id_01', -5.0)
            .property('id_02', 222578.0)
            ...
            .property('id_12', 'NotFound’)
            .property('id_15', 'New’)
            .property('id_16', 'NotFound’).as('newT’)
.addE('associated_with’)
            .from('newT’)
            .to(V('DeviceType:mobile’))
            .property(T.id, 'transaction_9999999-associated-with-mobile’)
.addE('purchased_by’)
            .from('newT’)
            .to(V('card3:150.0’))
            .property(T.id, 'transaction_9999999-purchased-by-card’)
.addE('identified_by’)
            .from('newT’)
            .to(V('id_05:0.0’))
            .property(T.id, 'transaction_9999999-identified-by-id_05’)
.addE('identified_by’)
            .from('newT’)
            .to(V('id_38:F’))
            .property(T.id, 'transaction_9999999-identified-by-id_38')
```

Inductive prediction on new node

```
g.with('Neptune#ml.endpoint','${endpoint}')
.V('9999999').properties('isFraud')
.with('Neptune#ml.classification')
.with('Neptune#ml.inductiveInference')
.with('Neptune#ml.deterministic').value()
```

Result:

1, 0.5687

Careem

What started as a vehicle-for-hire app, Careem now connects consumers to a variety of services for transportation, delivery, and payments in over 100 cities across 14 countries.

Challenge:

Detect and stop losses from first-party and third-party fraud. Prior methods of fraud detection involved rules and ML models, but was reactive instead of proactive.

Solution:

- Identity graph in Neptune containing information about users and shared features
- Node classification using Neptune ML to infer fraud

Learn more: https://go.aws/3IoHGL5

Careem

What started as a vehicle-for-hire app, Careem now connects consumers to a variety of services for transportation, delivery, and payments in over 100 cities across 14 countries.

Challenge:

Detect and stop losses from first-party and third-party fraud. Prior methods of fraud detection involved rules and ML models, but was reactive instead of proactive.

Solution:

- Identity graph in Neptune containing information about users and shared features
- Node classification using Neptune ML to infer fraud

Learn more: https://go.aws/3IoHGL5

Demo: Performing real-time inference using Neptune ML

Neptune

- Databases
- Snapshots
- Notebooks
- Subnet groups
- Parameter groups
- Events
- Event subscriptions

Database Services

Amazon Neptune
Fast and reliable graph database

Amazon Neptune is a fast, reliable graph database service that makes it easy to build and run applications that work with highly connected datasets.

Graph database

Create an Amazon Neptune database cluster.

Launch Amazon Neptune

Getting started

Using the AWS console, you can quickly provision Amazon Neptune graph database instances and connect them to your application using an Apache TinkerPop™ Gremlin Websockets Server or an RDF/SPARQL endpoint. You can also use the console to setup and configure high availability, read-replicas, and encryption at rest for your instance.

How it works

Amazon Neptune is a fast, scalable graph database service. Neptune efficiently stores and navigates highly connected data. Its query processing engine is optimized for leading graph query languages, Apache TinkerPop™ Gremlin and the W3C's RDF SPARQL. Neptune provides high performance through the open and standard APIs of these graph frameworks. And, Neptune is fully managed, so you no longer need to worry about database management tasks such as hardware provisioning, software patching, setup, configuration, or backups.

Cost

Amazon Neptune is a fast, reliable graph database service that makes it easy to build and run applications that work with highly connected datasets. Neptune makes it cost-effective to operate a graph database with enterprise class features regardless of instance size. You can create an instance with up to 64 TB of auto-scaling storage, 6-way

Benefits and features

High performance and scalability

Amazon Neptune is optimized for low-latency and high-throughput applications. Neptune is capable of processing graph queries over billions of relationships in milliseconds. It supports up to 15 read-

High availability, durability, and security

Amazon Neptune offers greater than 99.99% availability. It replicates six copies of your data across three Availability Zones and instance failover typically takes less than 30

Summary

Neptune ML enables Graph ML and GNN Pipeline

Seamless integration and abstract interfacing with AWS SageMaker and Deep Graph Library	No need for custom integrations, moving data around, learn separate tools, or even have deep ML experience	Make real-time Inductive Inference for applications in seconds

- End-to-end data ingestion pipeline for streaming events and bulk data
- Generate continuous value (confidence) score rather than a hard decision
- GNN - better generalisation for new (unseen) patterns
- Efficient grouping of data for higher accuracy classification or regressions
- Leverage data relationships as well as data features
- Use real-time graph data input without the need to retrain
- Less labour intensive than rule based or analytic based solutions

Amazon Neptune

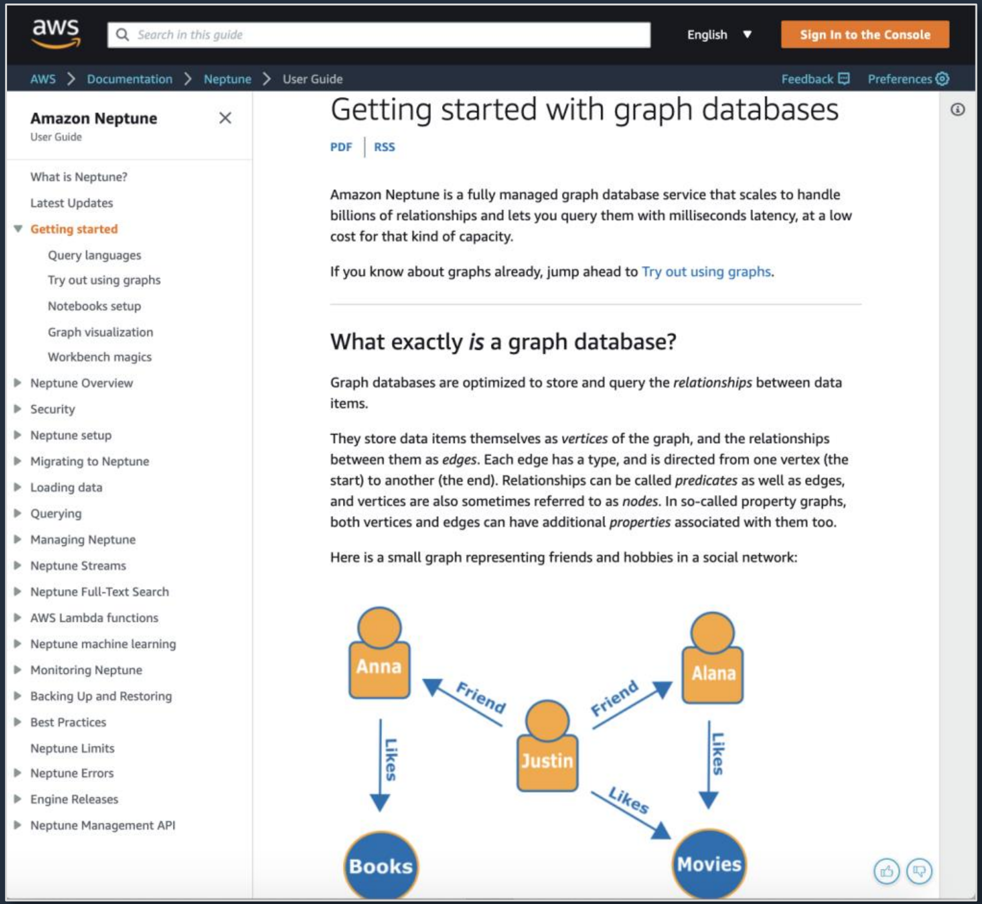
Amazon SageMaker

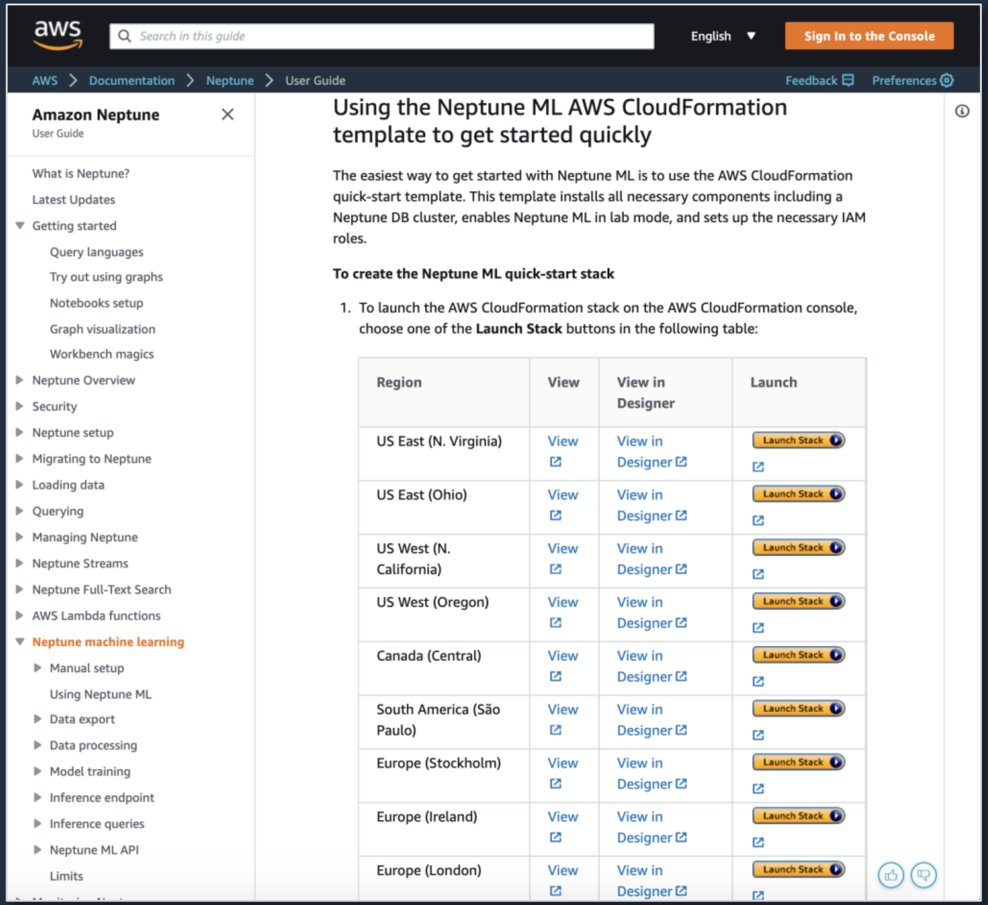
DEEP GRAPH LIBRARY

Resources

Neptune ML Documentation

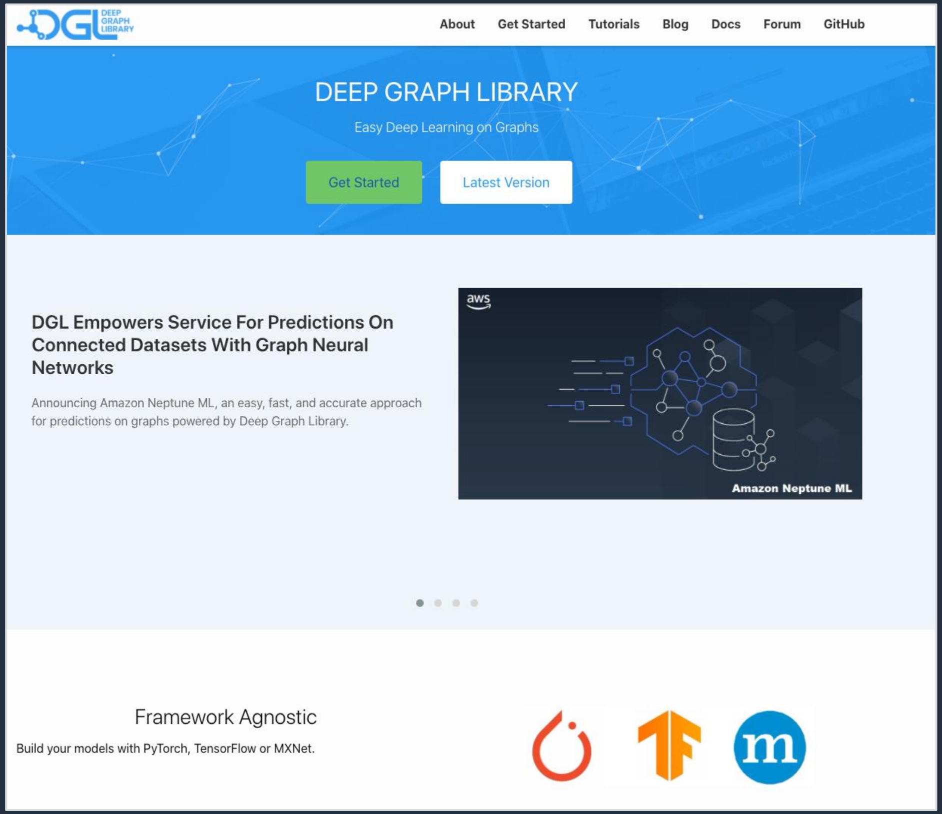
Start with the 'Getting Started' and 'Neptune Machine Learning' sections

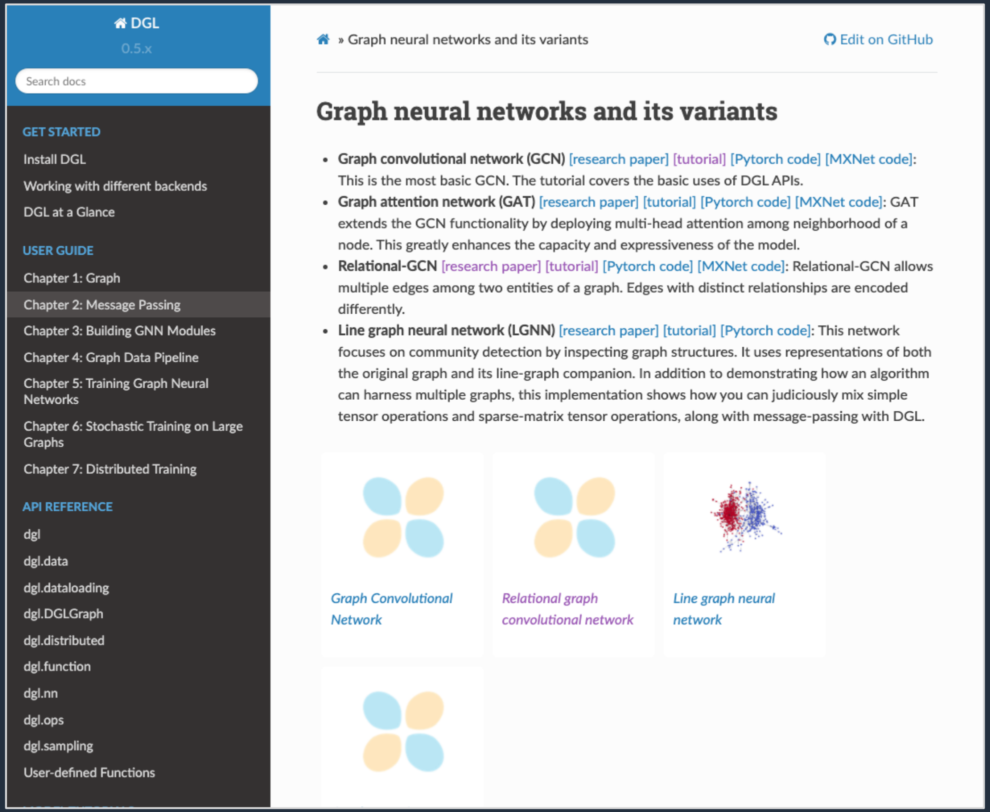




DGL Documentation

Start with the 'Getting Started' and 'Tutorials' sections





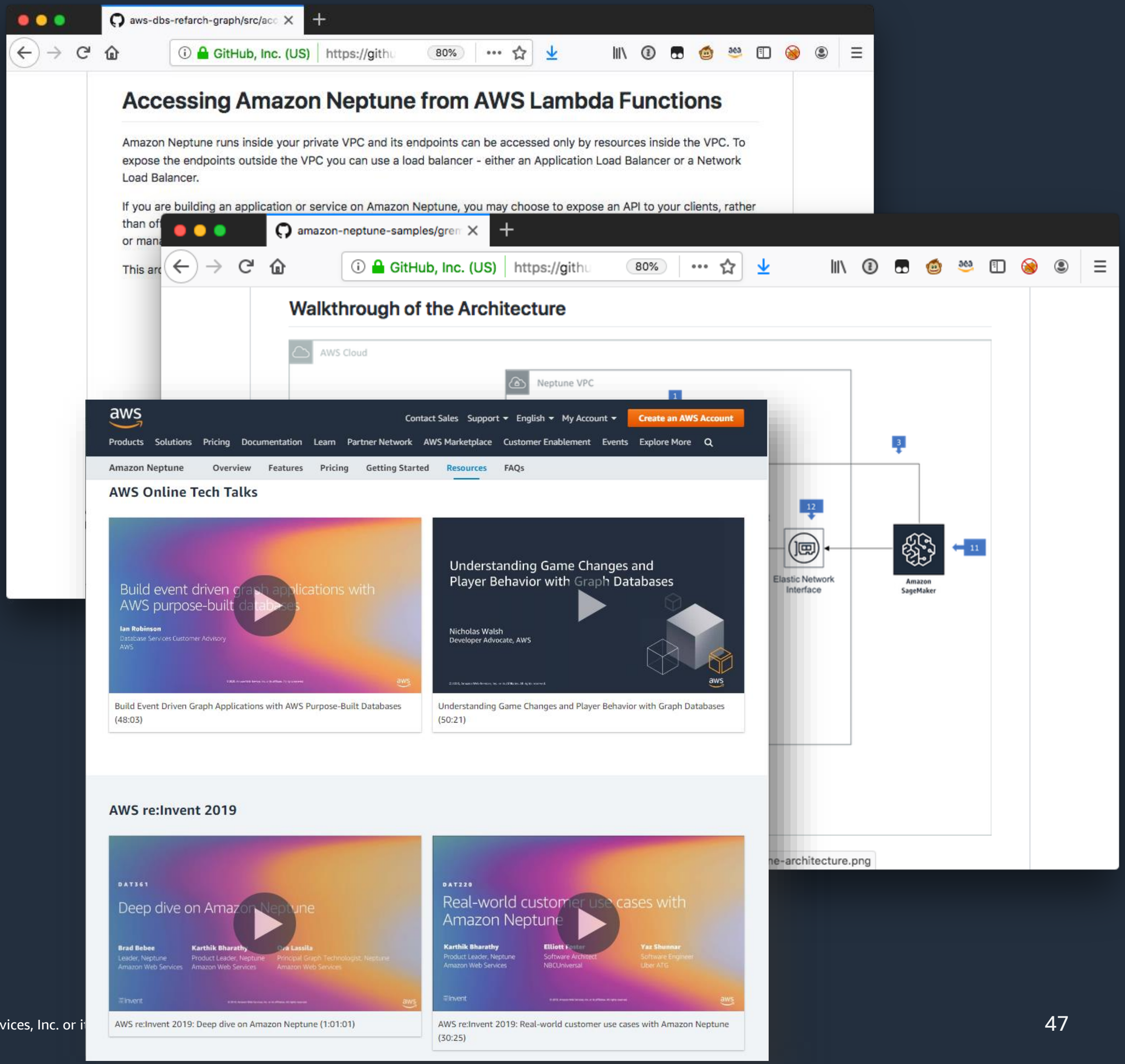
Additional Resources

[Neptune Notebooks/Graph Notebook]

[Neptune Reference Architectures]

[Neptune Sample Applications]

[Use Cases, Videos, Blogs, Code ….]



Thank you!

Kevin Phillips

LinkedIn: kevinphillips81