Graph Databases versus SQL for Data Science:

Identifying ‘Graph-y’ Problems in Your Data

Clair J. Sullivan, PhD
Data Science Advocate
Twitter: @CJLovesData1
Medium: https://medium.com/@cj2001

The most important questions!

- This workshop IS being recorded
- You will have access to this recording afterwards
- There is a repository where you can get all of the code
- All of the slides are available in the repository

What are we going to do today?

- “My data is all in SQL. Why should I bother with a graph database?”
  - How to identify a graph-y problem
- Why graphs can be better than tables for graph-y problems
- SQL demo
- Neo4j demo
- Final thoughts

https://dev.neo4j.com/graphy_problems

Two Key Concepts

1. It isn’t always obvious that you have a graph-y problem, but the importance of relationships between the data or a lot of SQL JOINs are hints

2. It is easier and faster to solve graph-y problems in a graph database rather than an RDBMS

“My data is all in BigQuery and scattered among several tables. I am trying to answer a question about our users, but the query takes forever.”

“How many queries do you have?”

“Well, there are a lot.”

Big O of common SQL queries

```
SELECT
    COUNT(*)
FROM
    Table
```

O(1)

O(n) complexity without a primary key

https://blog.devgenius.io/estimate-time-complexity-of-java-and-sql-query-afa13a88a981

Big O of common SQL queries

```
SELECT
    u.Name
FROM
    User u
```

O(n)

Big O of common SQL queries

```
SELECT
    u.Name,
    p.Comment
FROM
    User u
JOIN
    Post p
ON
    u.Id = p.UserId
WHERE
    u.Status=True
AND
    u.Name LIKE ‘Emil Eifrem%’
```

(A hash JOIN.)

O(N + M)

where:

N = hashed table

M = lookup table

Big O of common SQL queries

```
SELECT
    u.Name,
    p.Comment
FROM
    User u,
    Post p
WHERE
    u.Id = p.UserId
AND
    u.Status=True
AND
    p.NumAnswers >0
```

O(M + N)

(depending on index usage)

Big O of common SQL queries

```
SELECT
    u.Id,
    u.Name,
    p.Id,
    p.Comment
FROM
    User u
LEFT OUTER JOIN
    Post p
ON
    u.Id = p.UserId
```

O(M * N)

And let’s not forget how long and complicated SQL queries can get, particularly those with multiple JOINs!

Graph databases allow for…

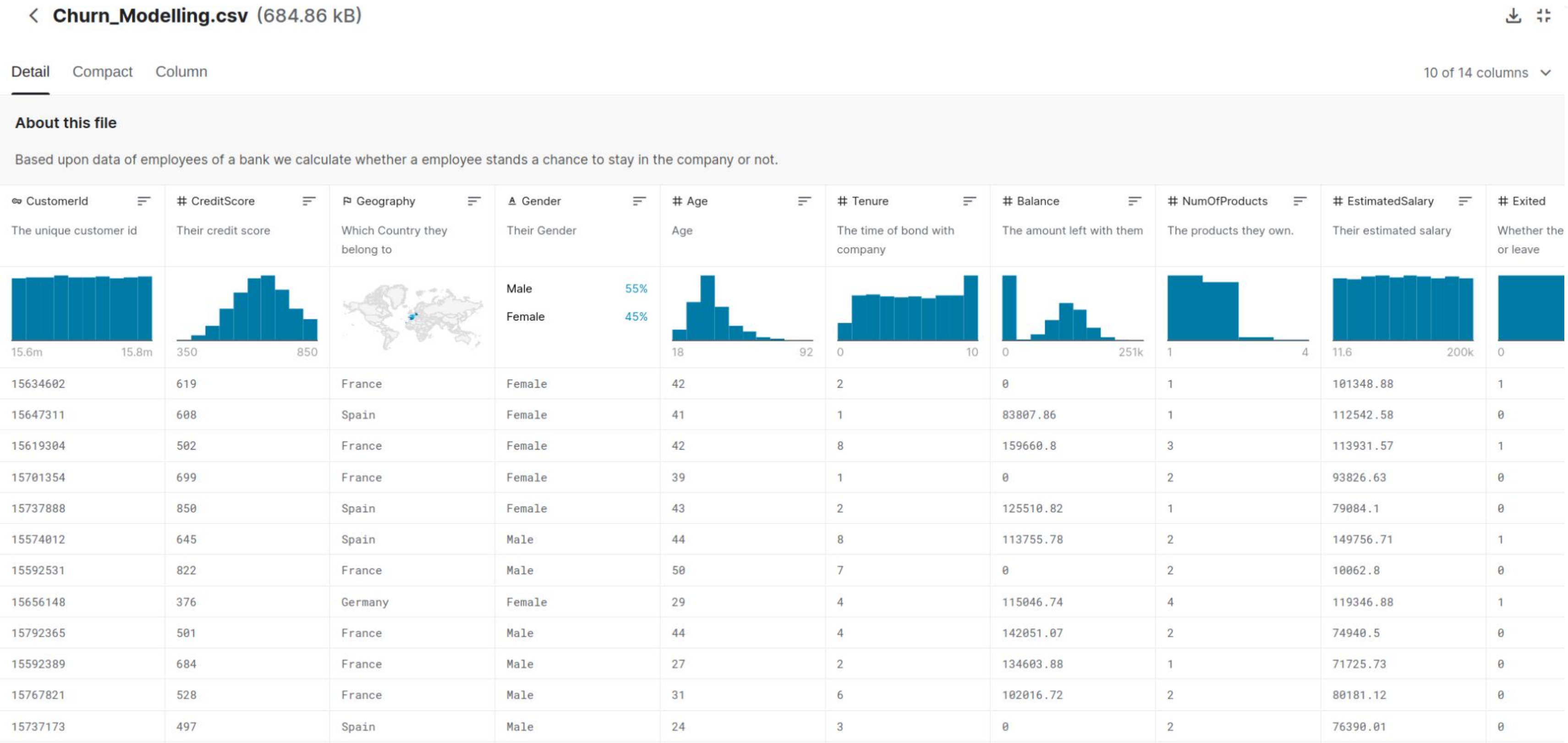
- Faster queries
- More intuitive queries
- Queries designed to take advantage of the relationships within the data

How to identify a graph-y problem

How do you know if you have a graph-y problem?

When relationships between data points matter (but they might be subtle!)

A churn example



< **Churn_Modelling.csv** (684.86 kB)

Detail Compact Column

10 of 14 columns

About this file

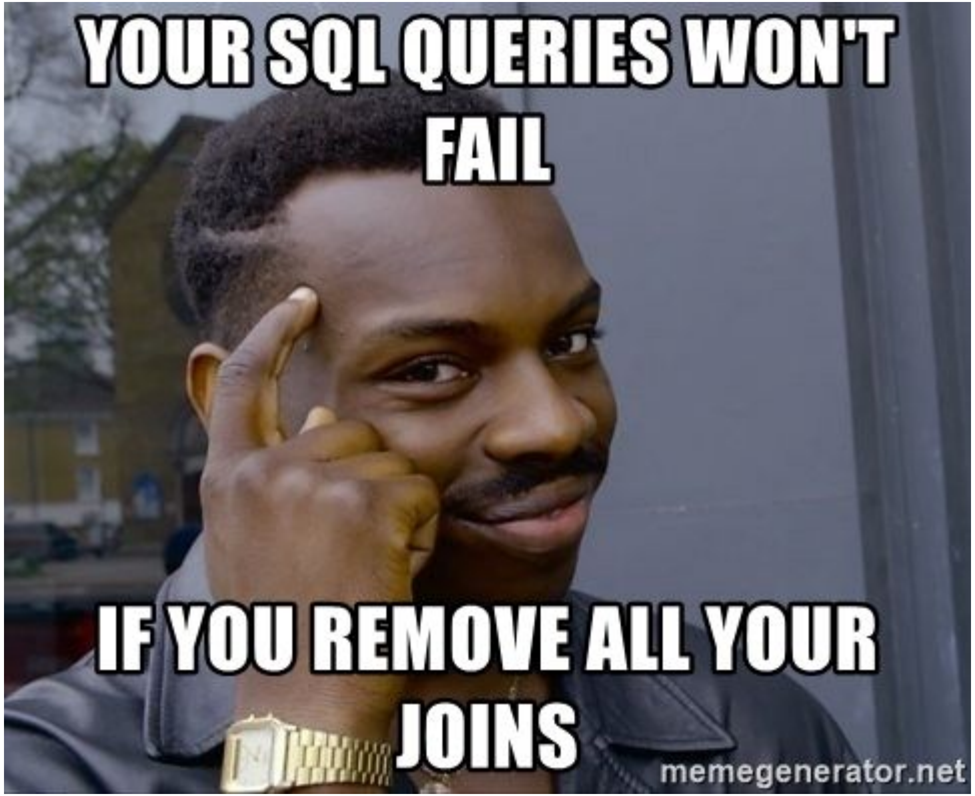
Based upon data of employees of a bank we calculate whether a employee stands a chance to stay in the company or not.

CustomerId	# CreditScore	Geography	Gender	# Age	# Tenure	# Balance	# NumOfProducts	# EstimatedSalary	# Exited
The unique customer id	Their credit score	Which Country they belong to	Their Gender	Age	The time of bond with company	The amount left with them	The products they own.	Their estimated salary	Whether the or leave
15634602	619	France	Female	42	2	0	1	101348.88	1
15647311	608	Spain	Female	41	1	83807.86	1	112542.58	0
15619304	502	France	Female	42	8	159660.8	3	113931.57	1
15701354	699	France	Female	39	1	0	2	93826.63	0
15737888	850	Spain	Female	43	2	125510.82	1	79084.1	0
15574012	645	Spain	Male	44	8	113755.78	2	149756.71	1
15592531	822	France	Male	50	7	0	2	10062.8	0
15656148	376	Germany	Female	29	4	115046.74	4	119346.88	1
15792365	501	France	Male	44	4	142051.07	2	74940.5	0
15592389	684	France	Male	27	2	134603.88	1	71725.73	0
15767821	528	France	Male	31	6	102016.72	2	80181.12	0
15737173	497	Spain	Male	24	3	0	2	76390.01	0

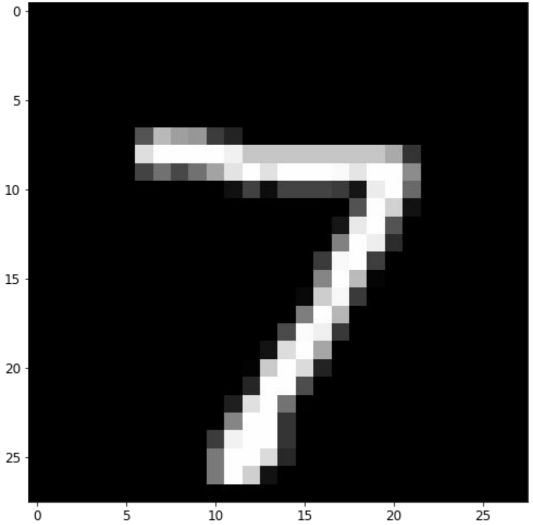
How do you know if you have a graph-y problem?

Rule of thumb:

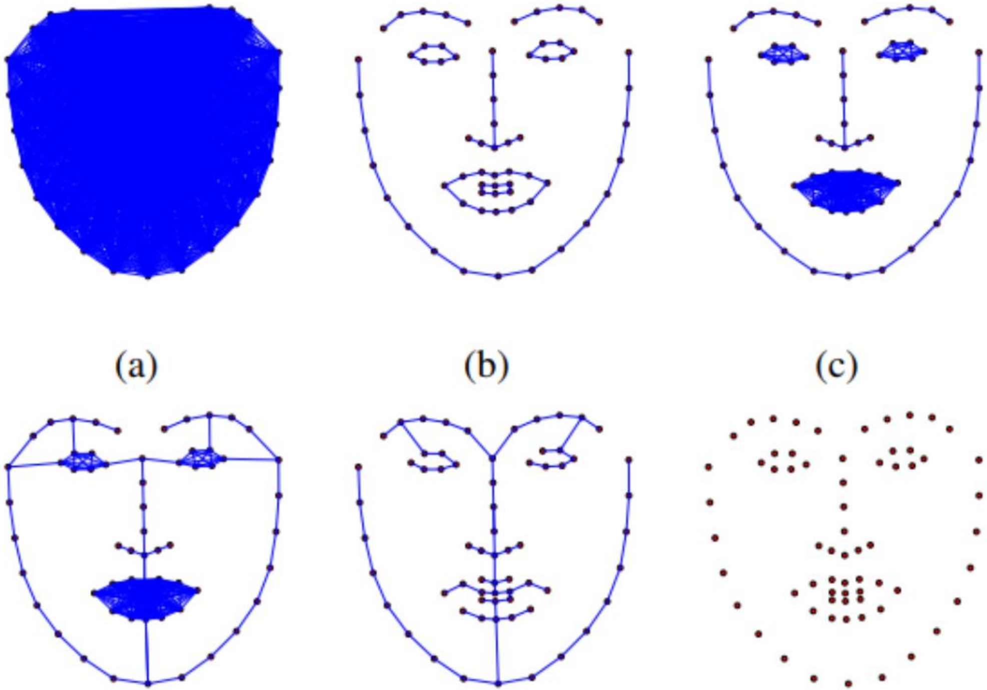
If you have to do more than a couple SQL JOINs then suspect you have a graph-y problem


YOUR SQL QUERIES WON'T FAIL
IF YOU REMOVE ALL YOUR JOINS
memegenerator.net

Will it graph: MNIST

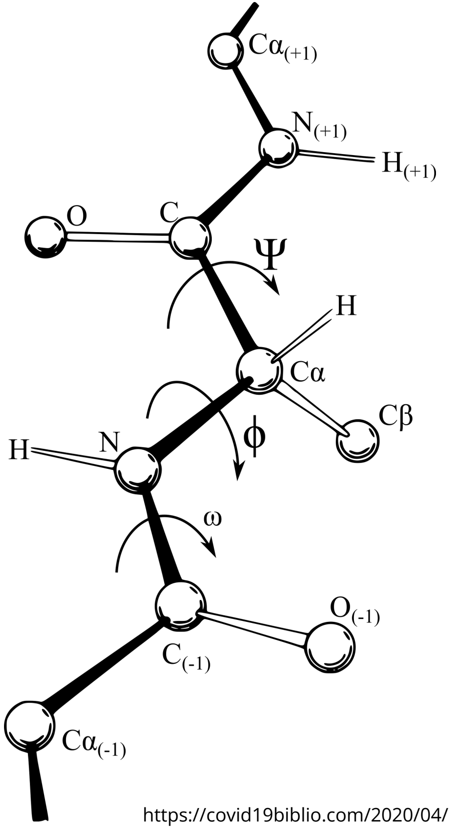


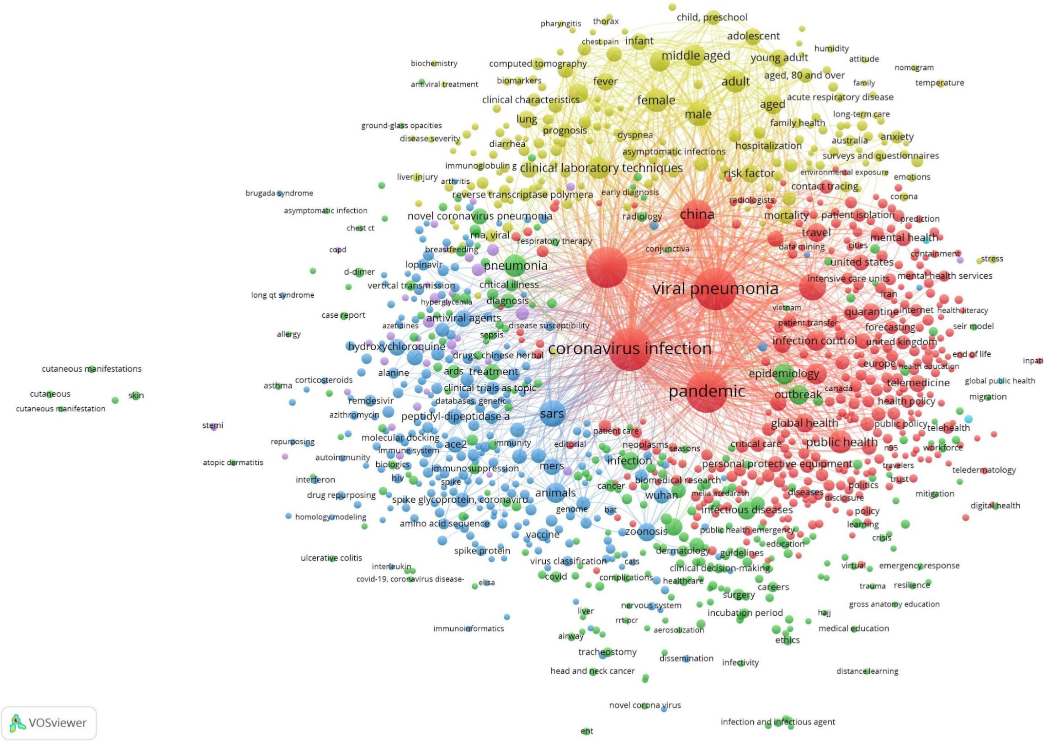
Will it graph: facial recognition



https://medium.com/@BorisAKnyazev/tutorial-on-graph-neural-networks-for-computer-vision-and-beyond-part-1-3d9fada3b80d

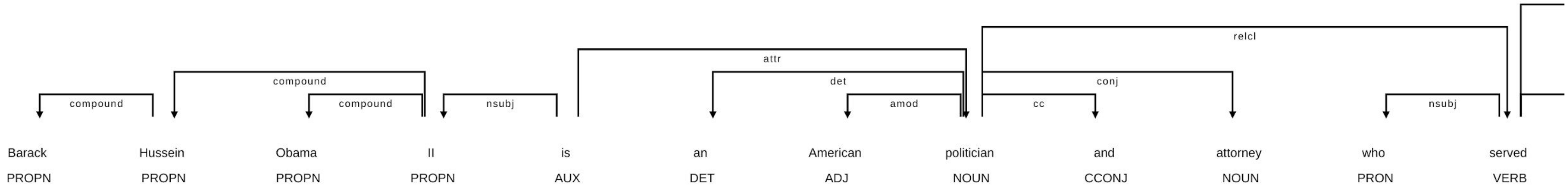
Will it graph: drug discovery





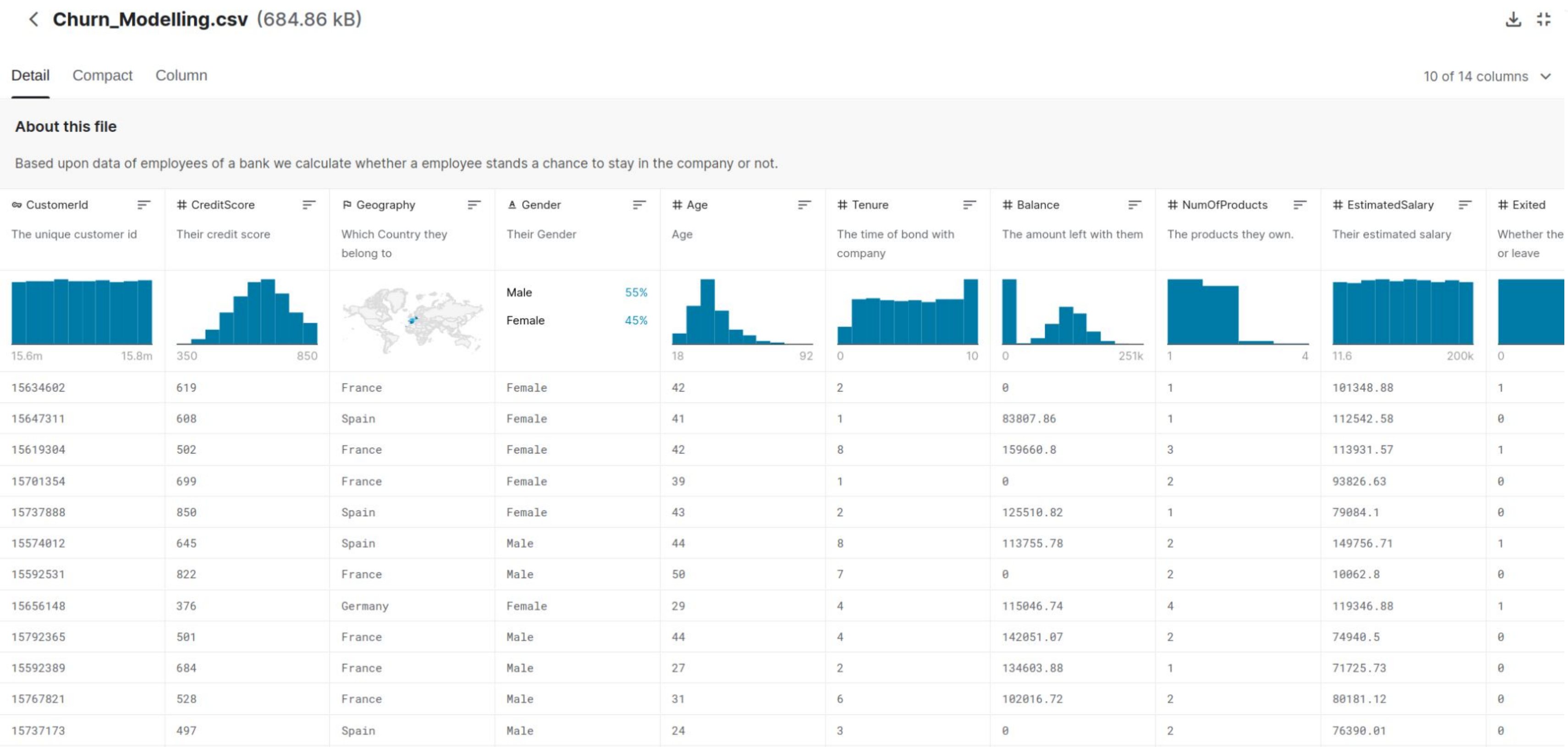
https://covid19biblio.com/2020/04/28/keyword-co-occurrence-network-graph-for-the-overall-research-field-on-covid-19-up-to-april-27th-2020/

Will it graph: natural language processing



Why graphs can be better than tables for graph-y problems

Traditional ML assumes all data points are independent



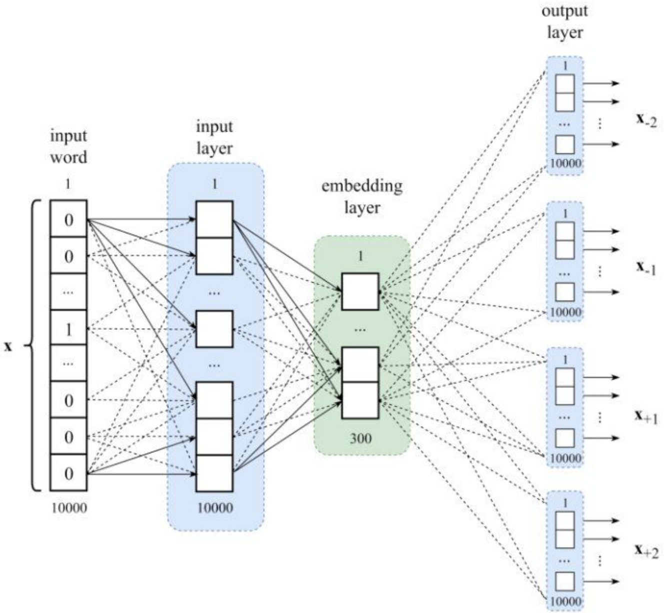
Churn_Modelling.csv (684.86 kB)

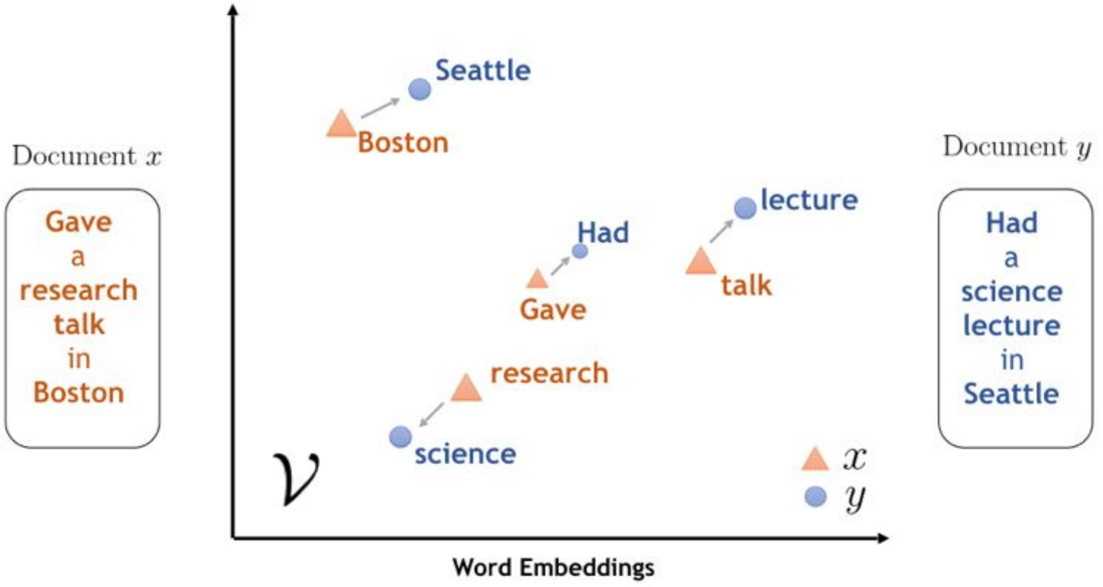
About this file

Based upon data of employees of a bank we calculate whether a employee stands a chance to stay in the company or not.

CustomerId	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	EstimatedSalary	Exited
The unique customer id	Their credit score	Which Country they belong to	Their Gender	Age	The time of bond with company	The amount left with them	The products they own.	Their estimated salary	Whether the or leave
15.6m 15.8m	350 850		Male 55% Female 45%	18 92	0 10	0 251k	1 4	11.6 200k	0
15634602	619	France	Female	42	2	0	1	101348.88	1
15647311	608	Spain	Female	41	1	83807.86	1	112542.58	0
15619304	502	France	Female	42	8	159660.8	3	113931.57	1
15701354	699	France	Female	39	1	0	2	93826.63	0
15737888	850	Spain	Female	43	2	125510.82	1	79084.1	0
15574012	645	Spain	Male	44	8	113755.78	2	149756.71	1
15592531	822	France	Male	50	7	0	2	10062.8	0
15656148	376	Germany	Female	29	4	115046.74	4	119346.88	1
15792365	501	France	Male	44	4	142051.07	2	74940.5	0
15592389	684	France	Male	27	2	134603.88	1	71725.73	0
15767821	528	France	Male	31	6	102016.72	2	80181.12	0
15737173	497	Spain	Male	24	3	0	2	76390.01	0

An example of traditional ML: word vectors (NLP)



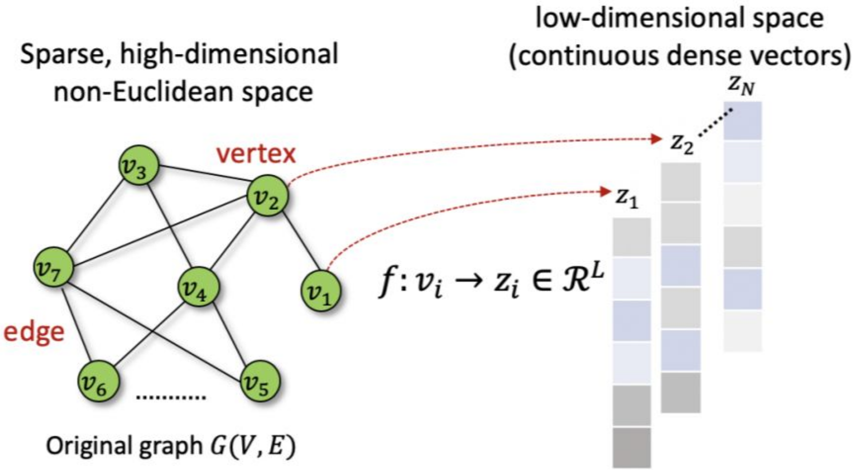


https://www.kdnuggets.com/2019/01/burkov-self-supervised-learning-word-embeddings.html

https://medium.com/swlh/word2vec-in-practice-for-natural-language-processing-a179b3286a21

Graph embeddings

- Transductive
- Inductive
- Matrix factorization
- Methods based on random walks
  - FastRP
  - node2vec
- Methods based on neural networks



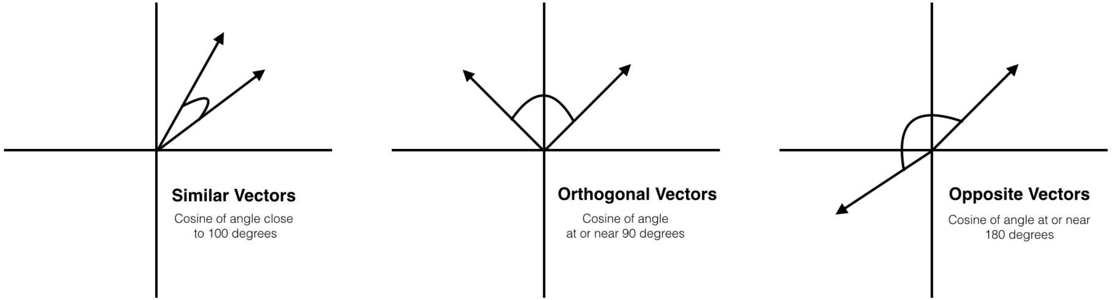
Sparse, high-dimensional non-Euclidean space
low-dimensional space (continuous dense vectors)
vertex
edge
v_1
v_2
v_3
v_4
v_5
v_6
v_7
$f: v_i \rightarrow z_i \in \mathcal{R}^L$
z_1
z_2
z_N
Original graph $G(V, E)$

M. Xu (2020) *arXiv:2012.08019v1*

Node similarity

- Jaccard
- Cosine
- Euclidean distance
- K-Nearest Neighbors (KNN)

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$


Similar Vectors
Cosine of angle close to 100 degrees
Orthogonal Vectors
Cosine of angle at or near 90 degrees
Opposite Vectors
Cosine of angle at or near 180 degrees

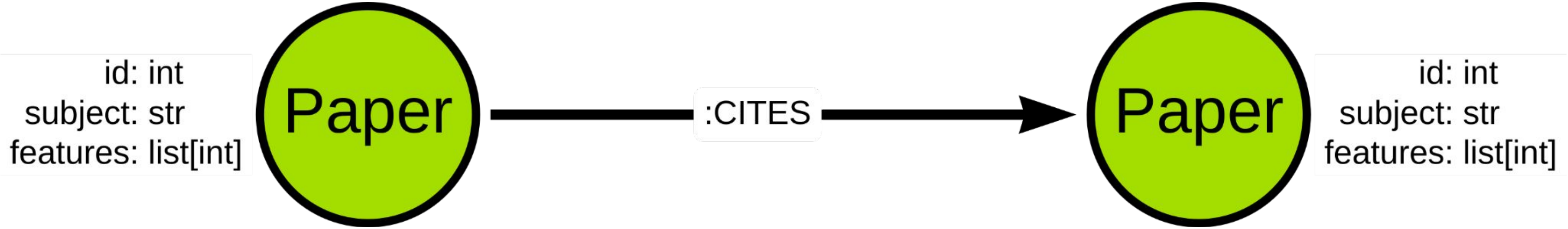
https://learning.oreilly.com/library/view/mastering-machine-learning/9781785283451/ba8bef27-953e-42a4-8180-cea152af8118.xhtml

All of the same ML models can be run using graph embeddings!

- Classification (binary, multi-class, multi-label)
- Regression
- Clustering
- Dimensionality reduction
- Similarity
- Plus more that are unique to graphs!
    - Link prediction
    - (Sub)graph-level structural similarity

CORA Dataset

- 2708 scientific publications in data science
- 7 classes
- 5429 citation relationships
- Abstracts one-hot encoded to a vocabulary of 1433 words

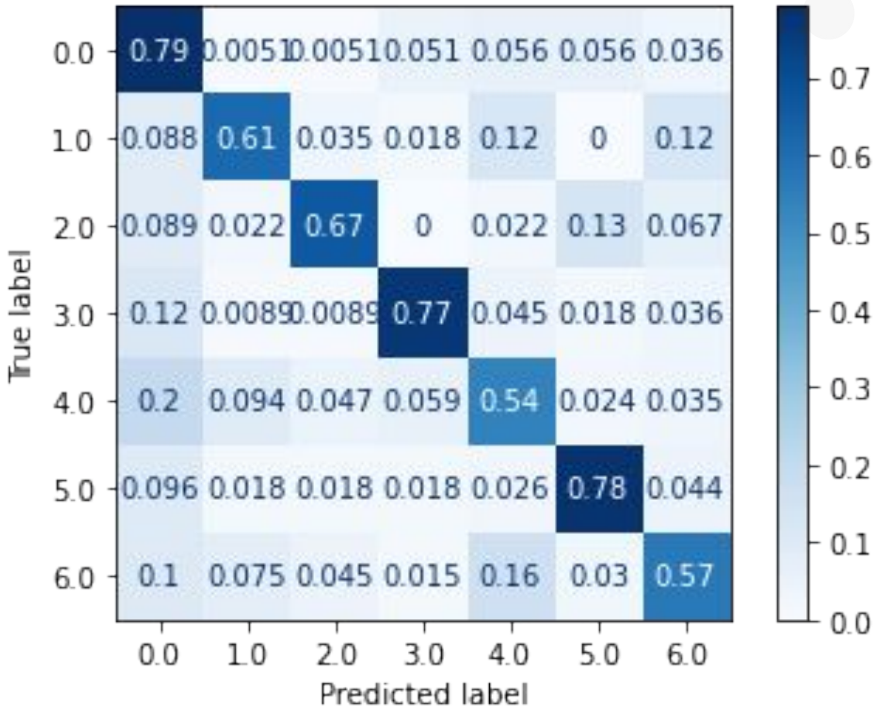


Two “models”

- Goal: compare embeddings
  - Keep ML model identical
  - Keep ML model simple
- Caveat
  - No access to the word vectors, vocabularies for tuning
- Both “models” given the same task
  - Multi-class classification
  - Imbalanced dataset

Results using word vectors

Mean accuracy: 0.726

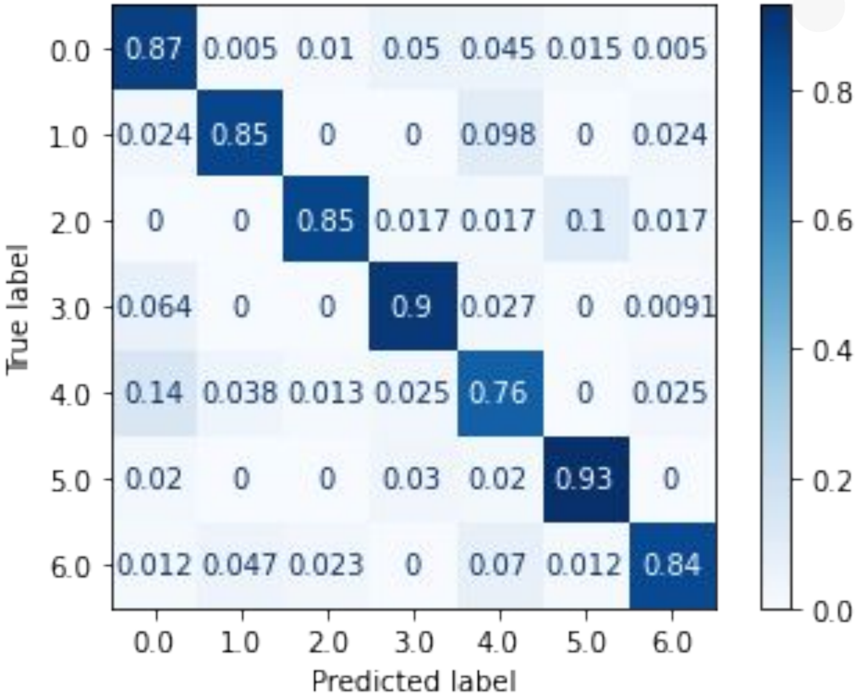


True label \ Predicted label	0.0	1.0	2.0	3.0	4.0	5.0	6.0
0.0	0.79	0.0051	0.0051	0.051	0.056	0.056	0.036
1.0	0.088	0.61	0.035	0.018	0.12	0	0.12
2.0	0.089	0.022	0.67	0	0.022	0.13	0.067
3.0	0.12	0.0089	0.0089	0.77	0.045	0.018	0.036
4.0	0.2	0.094	0.047	0.059	0.54	0.024	0.035
5.0	0.096	0.018	0.018	0.018	0.026	0.78	0.044
6.0	0.1	0.075	0.045	0.015	0.16	0.03	0.57

https://dev.neo4j.com/frp_tuning

FastRP embeddings with default hyperparameters

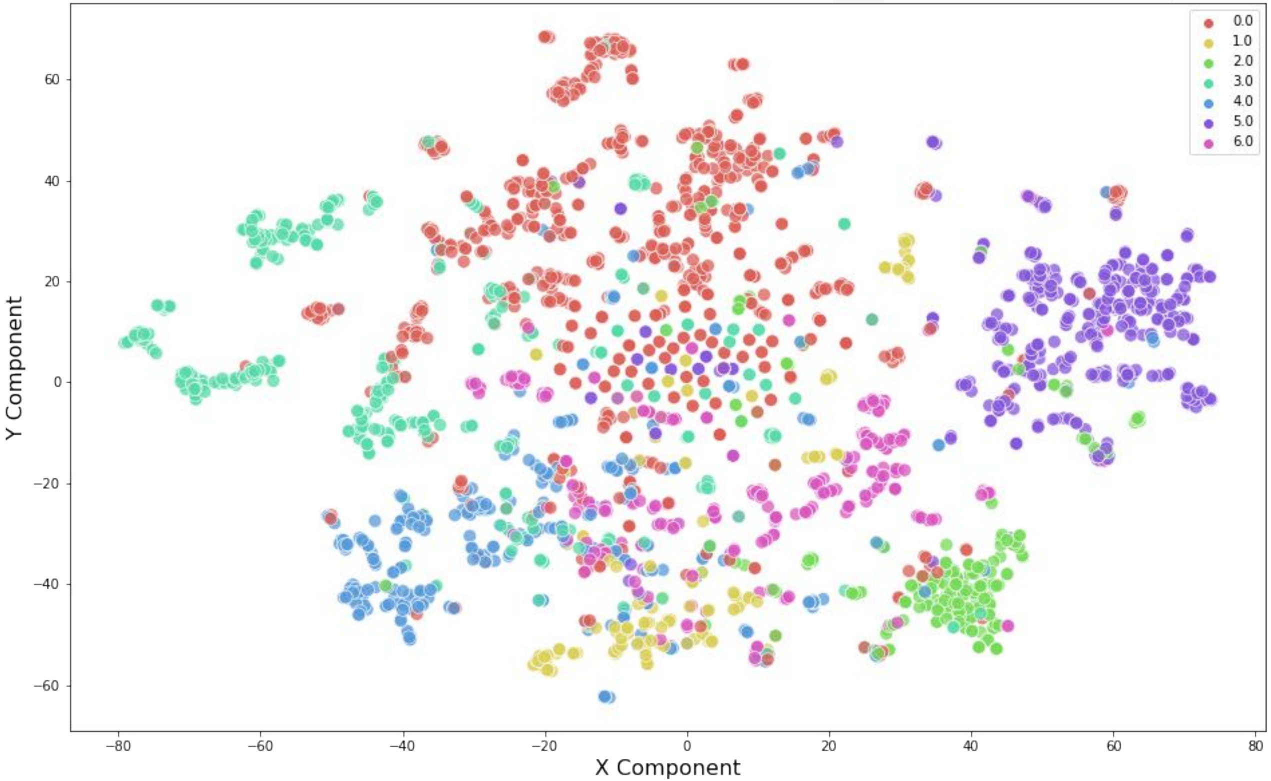
Mean accuracy: 0.850



True label \ Predicted label	0.0	1.0	2.0	3.0	4.0	5.0	6.0
0.0	0.87	0.005	0.01	0.05	0.045	0.015	0.005
1.0	0.024	0.85	0	0	0.098	0	0.024
2.0	0	0	0.85	0.017	0.017	0.1	0.017
3.0	0.064	0	0	0.9	0.027	0	0.0091
4.0	0.14	0.038	0.013	0.025	0.76	0	0.025
5.0	0.02	0	0	0.03	0.02	0.93	0
6.0	0.012	0.047	0.023	0	0.07	0.012	0.84

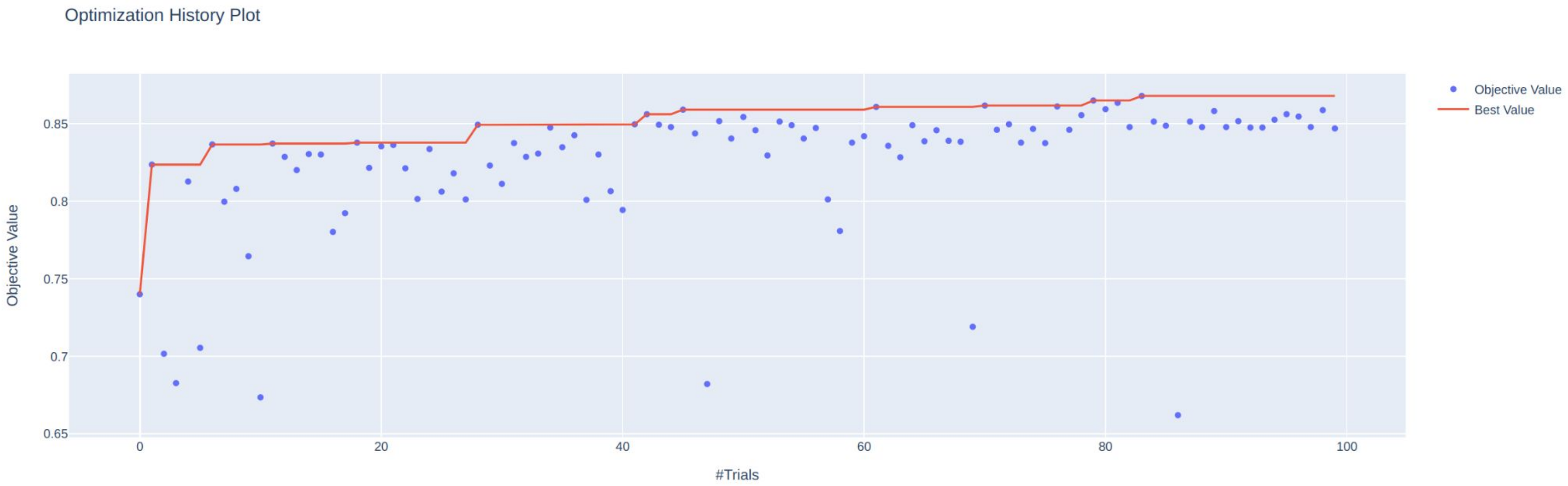
https://dev.neo4j.com/frp_tuning

Neural_Networks: 0.0
Rule_Learning: 1.0
Reinforcement_Learning: 2.0
Probabilistic_Methods: 3.0
Theory: 4.0
Genetic_Algorithms: 5.0
Case_Based: 6.0



https://dev.neo4j.com/frp_tuning

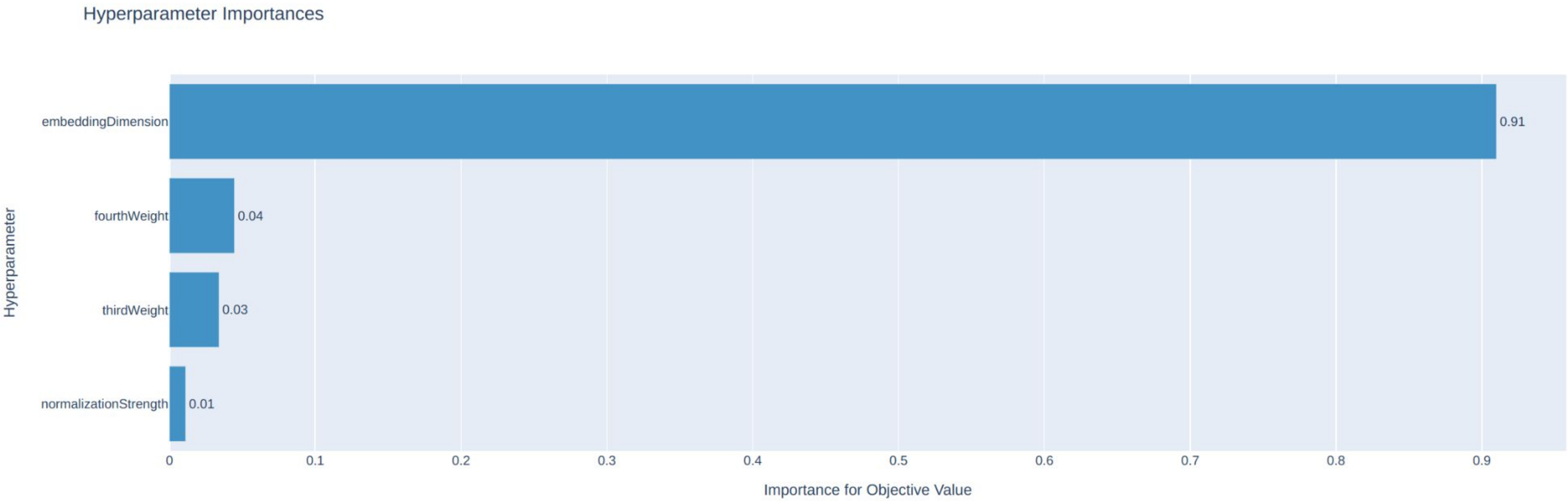
Results of tuning hyperparameters



Mean accuracy: 0.868

https://dev.neo4j.com/frp_tuning

Results of tuning hyperparameters

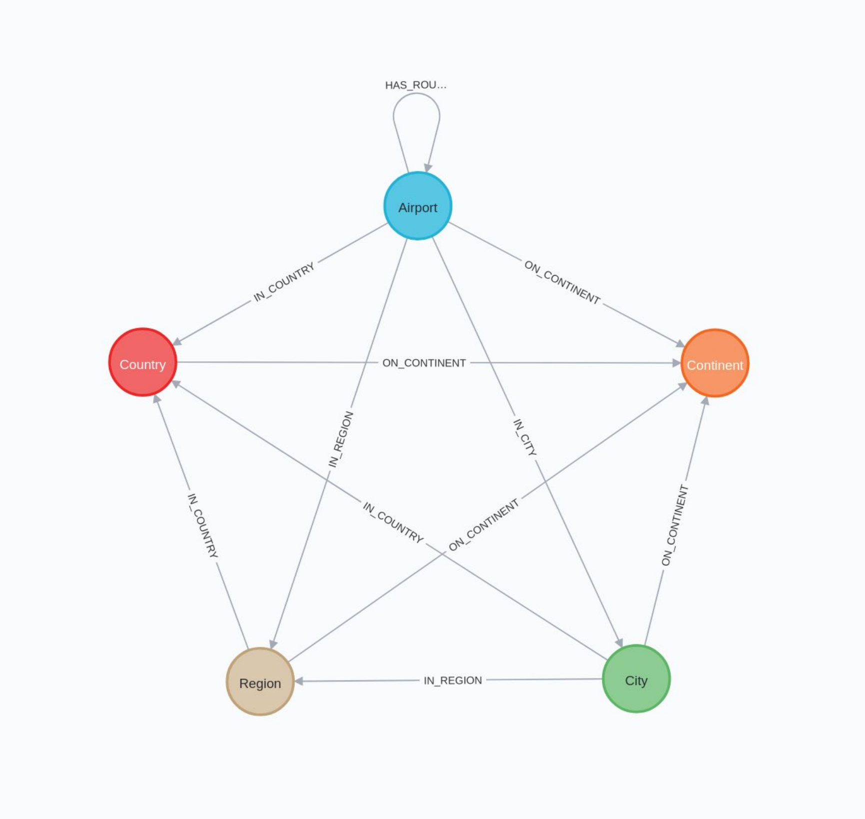


https://dev.neo4j.com/frp_tuning

LIVE CODING

https://dev.neo4j.com/graphy_problems

Graph model


HAS_ROU...
Airport
IN_COUNTRY
ON_CONTINENT
Country
ON_CONTINENT
Continent
IN_REGION
IN_CITY
IN_COUNTRY
IN_COUNTRY
ON_CONTINENT
ON_CONTINENT
Region
IN_REGION
City

Tools you will (maybe) need

- The repository! https://dev.neo4j.com/graphy_problems
- Sandbox: https://sandbox.neo4j.com
- The official Neo4j Python driver (`pip install neo4j`)
- A notebook environment
- A SQL environment
    - There is a Docker container in the repo to create this for you if you don't have one
    - PostgreSQL

Two Key Concepts

1. It isn’t always obvious that you have a graph-y problem, but the importance of relationships between the data or a lot of SQL JOINs are hints

2. It is easier and faster to solve graph-y problems in a graph database rather than an RDBMS

Final thoughts

- SQL (RDBMSs) can be alright when:
    - The queries are basic
    - There is no relationship between the individual data points
- Suspect relationships between the data points when you have more than a few JOINs
- Solve graph-y problems with graph-y solutions because:
    - They are faster
    - They are easier to write
    - They are designed to take advantage of the relationships between the individual data points

Thank you!

@CJLovesData1