**##Libraries to install**

pip install Flask

pip install neo4j

pip install pandas

pip install sklearn

**##Add output\_university.csv to import folder**

**##Create a neo4j database and create nodes**

create database with:

username : "neo4j"

password : "root@12345"

**## Change the .csv file path in the python file:**

uni = pd.read\_csv('/Users/yukthapriya/Documents/University/university\_data.csv')

Add your path here, wherever you have stored the file in your local.

**## Run the Python file from the command prompt**

source myenv/bin/activate

python /Users/yukthapriya/Documents/University/main.py

**## Navigate to the URL: http://127.0.0.1:8080**

Enter the university name and click on Get Recommendations

The top 5 recommended universities will be displayed:

**##Create nodes for university:**

LOAD CSV WITH HEADERS FROM 'file:///university\_data.csv' AS row

CREATE (u:University {

name: row.displayName,

actAvg: toFloat(row['act-avg']),

satAvg: toFloat(row['sat-avg']),

hsGpaAvg: toFloat(row['hs-gpa-avg']),

enrollment: toInteger(row.enrollment),

city: row.city,

zip: row.zip,

acceptanceRate: toFloat(row['acceptance-rate']),

rankingDisplayScore: toFloat(row['rankingDisplayScore']),

percentReceivingAid: toFloat(row['percent-receiving-aid']),

costAfterAid: toFloat(row['cost-after-aid']),

state: row.state,

rankingDisplayRank: toInteger(row['rankingDisplayRank']),

businessRepScore: toFloat(row['businessRepScore']),

tuition: toFloat(row.tuition),

engineeringRepScore: toFloat(row['engineeringRepScore']),

institutionalControl: row.institutionalControl

})

RETURN u;

**## Run Similarity and Graph Creation Query in Neo4j Browser, so it stays in the memory:**

**# Similarity Query:**

Euclidean Distance

MATCH (u1:University), (u2:University)

WHERE id(u1) < id(u2)

WITH u1, u2,

gds.similarity.euclidean(

[u1.actAvg, u1.satAvg, u1.acceptanceRate, u1.hsGpaAvg, u1.rankingDisplayRank, u1.businessRepScore, u1.engineeringRepScore],

[u2.actAvg, u2.satAvg, u2.acceptanceRate, u2.hsGpaAvg, u2.rankingDisplayRank, u2.businessRepScore, u2.engineeringRepScore]

) AS euclideanDistance

MERGE (u1)-[similarity:SIMILARITY\_EDGE]->(u2)

ON CREATE SET similarity.euclideanDistance = euclideanDistance;

**MyGraph Query:**

CALL gds.graph.exists('myGraph')

YIELD exists AS graphExists

// If the graph does not exist, create it

WITH 'myGraph' AS graphToCreate, graphExists

WHERE NOT graphExists

CALL gds.graph.project(graphToCreate, 'University', 'SIMILARITY\_EDGE', {

nodeProperties: ['actAvg', 'satAvg', 'acceptanceRate', 'hsGpaAvg', 'rankingDisplayRank', 'businessRepScore', 'engineeringRepScore']

}) YIELD graphName, nodeCount, relationshipCount

RETURN graphName, nodeCount, relationshipCount;

MATCH (u:University)-[r:SIMILARITY\_EDGE]->(similar:University)

WHERE u.name = 'Yale University' // Replace with the university name for which you need recommendations

RETURN similar.name, r.euclideanDistance

ORDER BY r.euclideanDistance ASC // Assuming lower Euclidean distance means higher similarity

LIMIT 5;

#Ten universities with states

MATCH (u:University)

RETURN u.name, u.city, u.state, u.enrollment, u.satAvg, u.actAvg, u.tuition

LIMIT 10;

Similarity edge between two universities

MATCH (u1:University {name: 'Johns Hopkins University'})-[r:SIMILARITY\_EDGE]-(u2:University {name: 'Northwestern University'})

RETURN u1.name AS University1, u2.name AS University2, r.euclideanDistance AS SimilarityScore

LIMIT 1;

Graph

MATCH (u1:University {name: 'Johns Hopkins University'})-[r:SIMILARITY\_EDGE]-(u2:University {name: 'Northwestern University'})

RETURN u1, u2, r

LIMIT 1;

# #Similarities between two universities all comparison

MATCH (u1:University)-[r:SIMILARITY\_EDGE]->(u2:University)

RETURN u1.name AS University1, u2.name AS University2, r.euclideanDistance

LIMIT 20;

MATCH (u1:University)-[r:SIMILARITY\_EDGE]->(u2:University)

RETURN u1.name AS University1, u2.name AS University2, r.euclideanDistance

LIMIT 100;

Good Engineering reputation

MATCH (u:University)

WHERE u.engineeringRepScore > 4.5

RETURN DISTINCT u.name, u.engineeringRepScore, u.city, u.state

ORDER BY u.engineeringRepScore DESC

LIMIT 10;

Total Universities

MATCH (u:University)

RETURN COUNT(u) AS totalUniversities;

Graph

MATCH (u:University)-[r:SIMILARITY\_EDGE]->(similar:University)

RETURN u, r, similar

LIMIT 50;

State and Universities list

MATCH (u:University)

RETURN u.state AS State, collect(u.name) AS Universities

LIMIT 20;

Acceptance rates

MATCH (u:University)

RETURN u.name, u.acceptanceRate

ORDER BY u.acceptanceRate ASC

LIMIT 10;

Business score and Engineering Reputation score

MATCH (u:University)

WHERE u.businessRepScore > 4 OR u.engineeringRepScore > 4

RETURN u.name, u.businessRepScore, u.engineeringRepScore

ORDER BY u.businessRepScore DESC, u.engineeringRepScore DESC

LIMIT 20;

similarities

MATCH (u1:University {name: 'Yale University'}), (u2:University {name: 'Harvard University'})

RETURN Distinct u1.name, u1.satAvg, u1.actAvg, u1.hsGpaAvg, u1.businessRepScore,

u2.name, u2.satAvg, u2.actAvg, u2.hsGpaAvg, u2.businessRepScore;

graph

MATCH (u:University {name: 'Yale University'})-[r:SIMILARITY\_EDGE]->(similar:University)

RETURN u, similar, r

ORDER BY r.euclideanDistance ASC

LIMIT 10;

(optional queries)

MATCH (u:University)-[r:SIMILARITY\_EDGE]->(similar:University)

WHERE u.name = 'Yale University' // Replace with a university name of interest

RETURN similar.name, r.euclideanDistance

ORDER BY r.euclideanDistance ASC

LIMIT 5;

Tuition fee and acceptance rate

MATCH (u:University)

WHERE u.costAfterAid < 20000 AND u.acceptanceRate > 50

RETURN u.name, u.costAfterAid, u.acceptanceRate

ORDER BY u.acceptanceRate DESC

LIMIT 5;

MATCH (u:University)

RETURN u

LIMIT 50;