Sistemas Expertos
Tema: Basados en casos.

Prueba 2 17/01/2021

Objetivo:

- Consolidar los conocimientos adquiridos en clase de los sistemas expertos basados en casos utilizando Neo4J.

Enunciado:

- Diseñe y desarrolle un algoritmo Knn en Neo4j para:
  - **Fila A - 0**: Usemos el ejemplo de conjunto de datos de Kaggle.com. Elegír el conjunto de datos del automóvil para este ejemplo y permite predecir el tipo de carro o automóvil, para ello el siguiente link de datos https://github.com/yfujieda/tech-cookbook/blob/master/python/knn-example2/data/car_dataset.csv [1]

1) Primero vamos a cargar los datos de tipo csv al Neo4j:

```
LOAD CSV FROM "file:///C:/Users/narcisa/Documents/9no/deberesquisi/DEBERES-SE-IA/SE/PruebaKnn/cars.csv" as line
CREATE  (:Carros{ marca: line[0], modelo: line[1], fuel_type: line[2], aspiration: line[3], num_of_doors: line[4], estilo:
line[5], drive_wheels: line[6], engine_location: line[7], wheel_base: toFloat(line[8]), largo: toFloat(line[9]), ancho:
toFloat(line[10]), alto: toFloat(line[11]), curb_weight: line[12], engine_type: line[13], num_of_cylinders: line[14],
engine_size: toFloat(line[15]), fuel_system: line[16], compression_ratio: toFloat(line[17]), horsepower:
toFloat(line[18]), peak_rpm: toFloat(line[19]), city_mpg: toFloat(line[20]), highway_mpg: toFloat(line[21]), price:
toFloat(line[22])})
```

```
neo4j$ LOAD CSV FROM "file:///C:/Users/narcisa/Documents/9no/deberesquisi/DEBERES-SE-IA/SE/PruebaKnn/cars.csv" as line …
Table
Added 206 labels, created 206 nodes, set 1641 properties, completed after 20 ms.
```

2) Hacemos un match(n) return n para ver los nodos generados en Neo4j

Nodos

Sistemas Expertos
Tema: Basados en casos.

Prueba 2 17/01/2021

Diseño del algoritmo Eucladiana

```
MATCH (c:Carros)
 WITH {item:id(c), weights: apoc.convert.toIntList(c.price)} AS userData
 WITH collect(userData) AS valorPrecio
 CALL gds.alpha.similarity.euclidean.stream({
  data: valorPrecio,
  skipValue: null
 })
 YIELD item1, item2,count1, count2, similarity
 RETURN gds.util.asNode(item1).marca AS from, gds.util.asNode(item2).marca AS to, similarity
 ORDER BY similarity DESC
```

```
$ MATCH (c:Carros) WITH {item:id(c), weights: apoc.convert.toIntList(c.price)} AS userData WITH collect(userDa…
```

	from	to	similarity
1	"mercedes-benz"	"porsche"	8372.0
2	"mercedes-benz"	"mercedes-benz"	4440.0
3	"bmw"	"porsche"	4287.0
4	"bmw"	"mercedes-benz"	4085.0
5	"mercedes-benz"	"bmw"	4085.0

Datos de Entrenamiento y Prueba

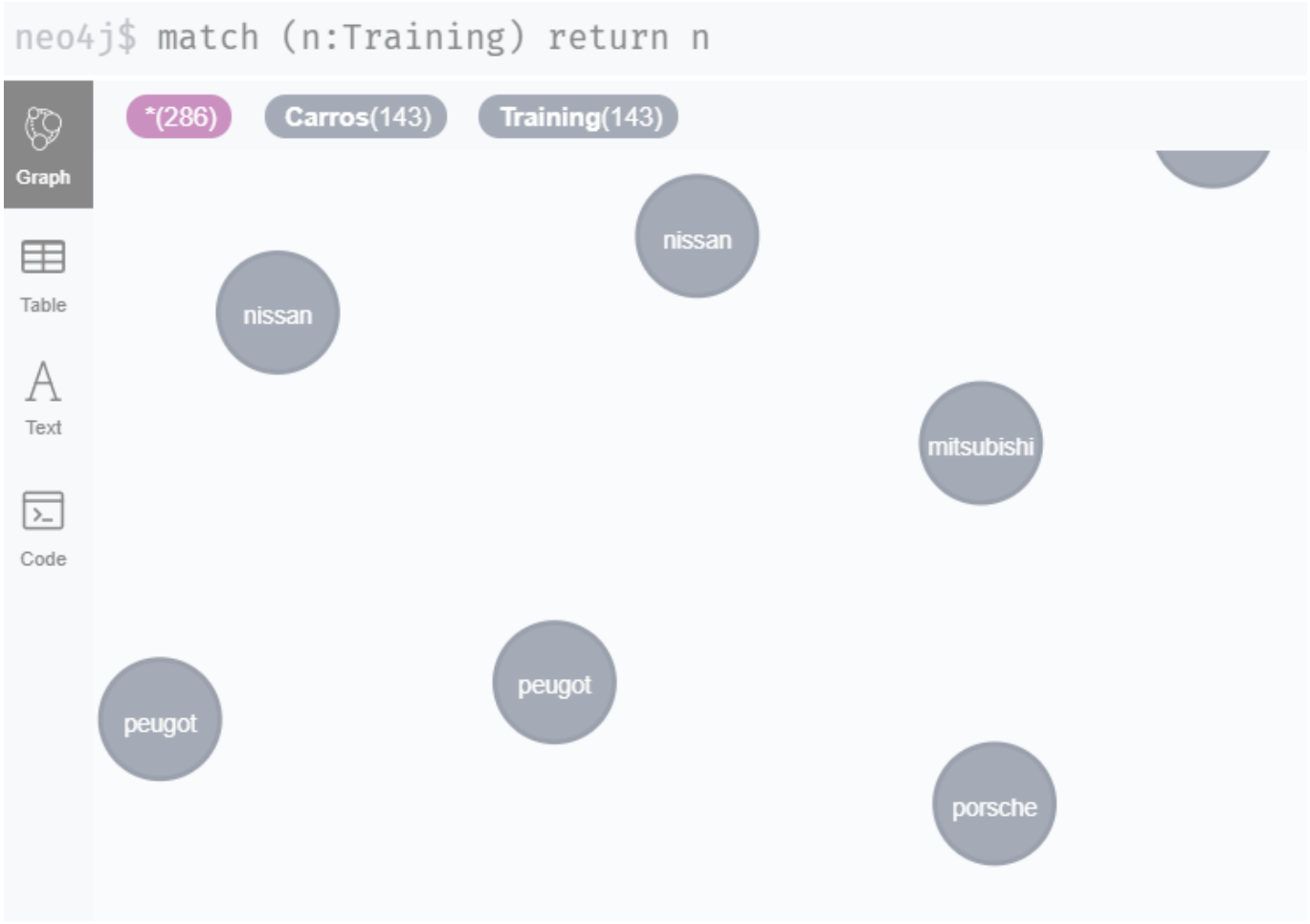
Dividimos el conjunto de datos en dos subconjuntos, donde el 70% de los nodos se marcarán como datos de entrenamiento y el 30% restante como datos de prueba. Hay un total de 206 nodos en nuestro gráfico. Total Nodos del 70% es 143 Marcaremos 143 nodos como subconjunto de entrenamiento y el resto como prueba.

Prueba 2

17/01/2021

Marcamos datos de entrenamiento 70%

MATCH (c:Carros)
WITH c LIMIT 143
SET c:Training;



Marcamos datos de prueba 30%

MATCH (c:Carros)
WITH c SKIP 143
SET c:Test;

Prueba 2 17/01/2021

Ahora creamos el grafo para hacer comprobar la similitud de los carros con respecto al precio y la marca.

nodeProjection

```
{
  "Carros": {
"properties": {
"price": {
"property": "price",
"defaultValue": null
      }
    },
"label": "Carros"
  }
}
```

relationshipProjection

```
{
  "__ALL__": {
"orientation": "NATURAL",
"aggregation": "DEFAULT",
"type": "*",
"properties": {

    }
  }
}
```

Ahora podemos ver la similitud que hay entre los carros.

Prueba 2 17/01/2021

```
j$ CALL gds.beta.knn.stream('k1Localidadd', {topK: 2, randomSeed: 42, nodeWeigh…
```

	Marca1	Modelo	Similitud
4	"porsche"	"porsche 127"	0.012195121951219513
5	"porsche"	"porsche 129"	0.012195121951219513
6	"jaguar"	"porsche 128"	0.017857142857142856
7	"jaguar"	"porsche 129"	0.017857142857142856
8	"chevrolet"	"toyota 159"	0.1111111111111111
9	"mitsubishi"	"nissan 102"	0.125

```
CALL gds.graph.create(
    'k1Localidadd',
    {
        Carros : {
            label: 'Carros',
            properties: 'price'

        }
    },
    '*'
);

```

Prueba 2

17/01/2021

Marca1	Modelo	Similitud
"bmw"	"mercedes-benz 75"	0.0002447381302006853
"mercedes-benz"	"bmw 17"	0.0002447381302006853
"mercedes-benz"	"porsche 129"	0.0002542588354945334
"mercedes-benz"	"mercedes-benz 70"	0.00038095238095238096
"mercedes-benz"	"bmw 16"	0.00038684719535783365
"mercedes-benz"	"bmw 16"	0.0003979307600477517

```
CALL gds.beta.knn.stream('k1Localidadd', {topK: 2,sampleRate: 1 , randomSeed: -1,
nodeWeightProperty: 'price'})
YIELD node1, node2, similarity
RETURN gds.util.asNode(node1).modelo AS Modelo, gds.util.asNode(node2).modelo AS Modelo2,
similarity AS Similitud
ORDER BY Similitud ASCENDING, Modelo, Modelo2
```

	Marca1	Modelo	Similitud
8	"mercedes-benz"	"bmw 17"	0.0002447381302006853
9	"mercedes-benz"	"porsche 129"	0.0002542588354945334
10	"mercedes-benz"	"mercedes-benz 70"	0.00038095238095238096
11	"mercedes-benz"	"bmw 16"	0.00038684719535783365

Prueba 2 17/01/2021

```
CALL gds.beta.knn.stream('k1Localidadd', {topK: 5,sampleRate: 1 , randomSeed: -1,
nodeWeightProperty: 'horsepower'})
YIELD node1, node2, similarity
RETURN gds.util.asNode(node1).marca AS Modelo, gds.util.asNode(node2).marca AS Modelo2,
similarity AS Similitud
ORDER BY Similitud ASCENDING, Modelo, Modelo2
```

j$ CALL gds.beta.knn.stream('k1Localidadd', {topK: 5,sampleRate: 1 , randomSeed…

	Modelo	Modelo2	Similitud
5	"porsche"	"porsche"	0.012195121951219513
6	"porsche"	"porsche"	0.012195121951219513
7	"porsche"	"porsche"	0.012195121951219513
8	"jaguar"	"nissan"	0.015873015873015872