
David Israel Leon

Objetivo:

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

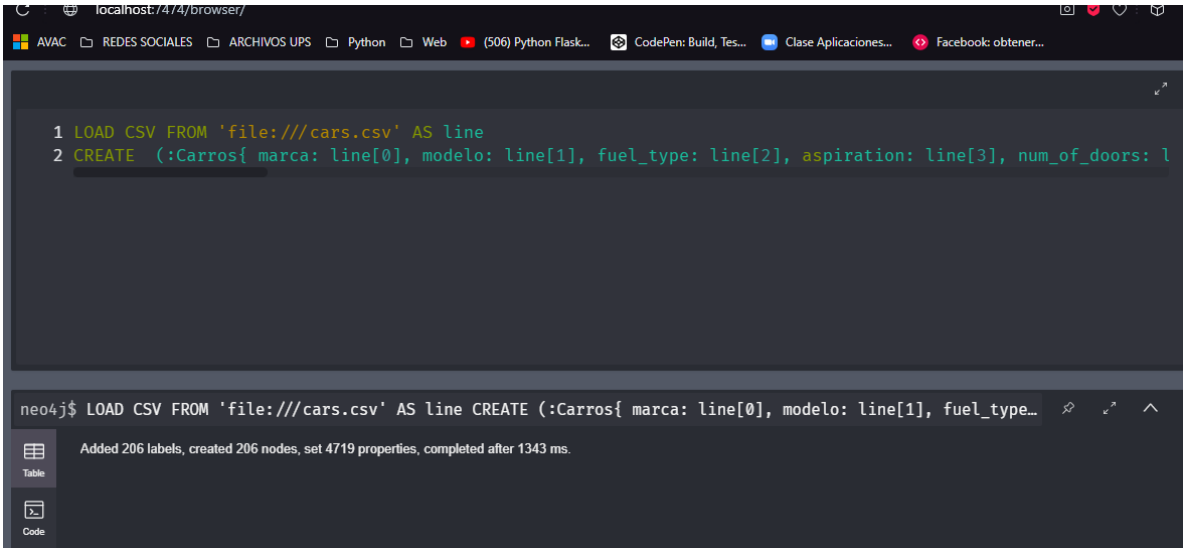
Enunciado:

- 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]

Se cargan los datos

LOAD CSV FROM 'file:///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])})




**Se diseña con Eucliades**

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

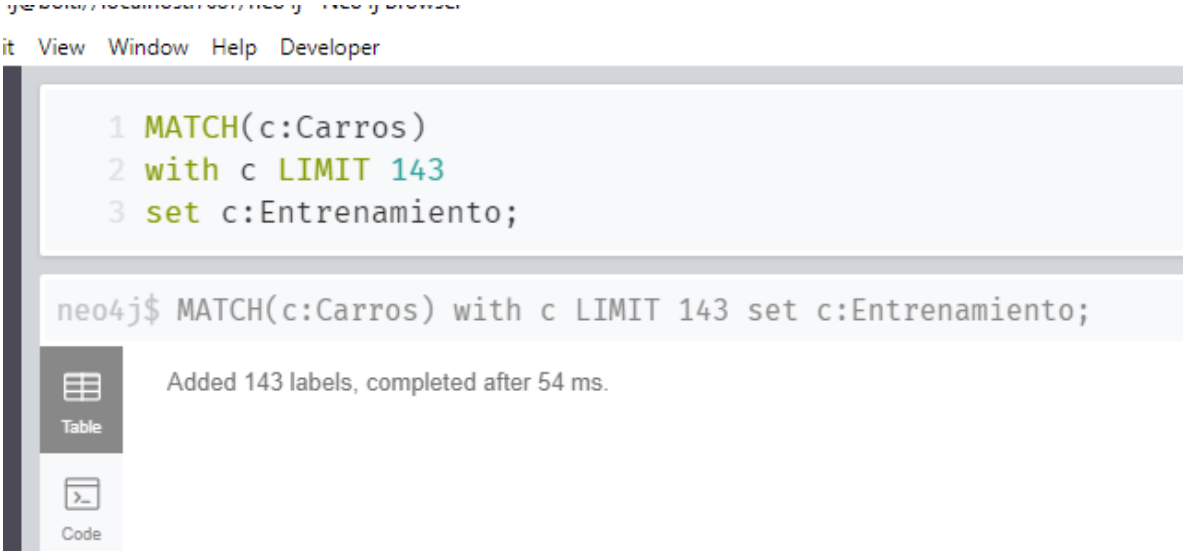
ORDER BY similarity DESC

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

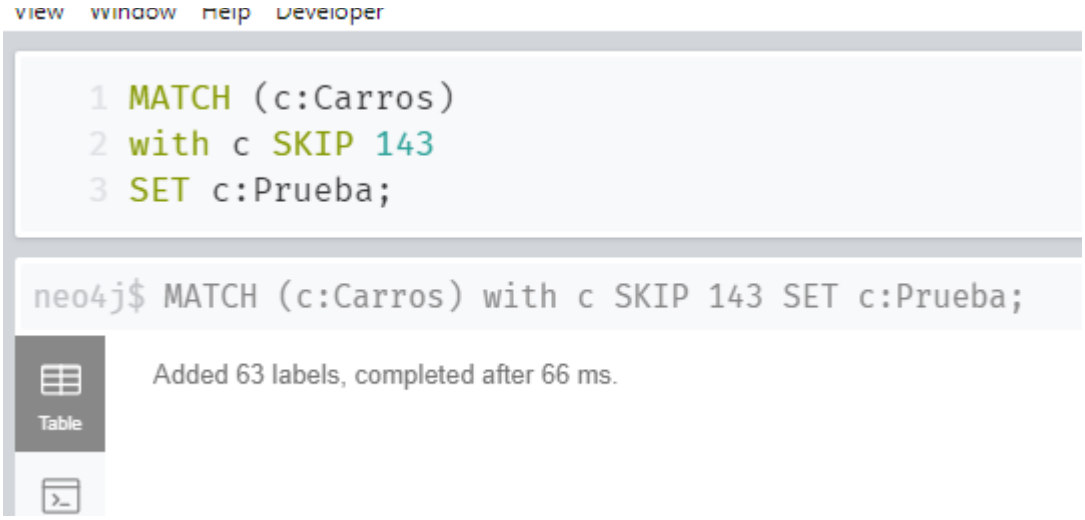
```
neo4j$ MATCH (c:Carros) WITH {item:id(c), weights: apoc.convert.toIntList(c.price)} AS userData W…
```

| | 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 |

Se divine el conjunto de datos, el uno debe tener el 70% y el otro debe tener el 30%, se calcula en porcentaje con el total de nuestros nodos: entonces el 70% es 143 nodos y el resto es el 30%

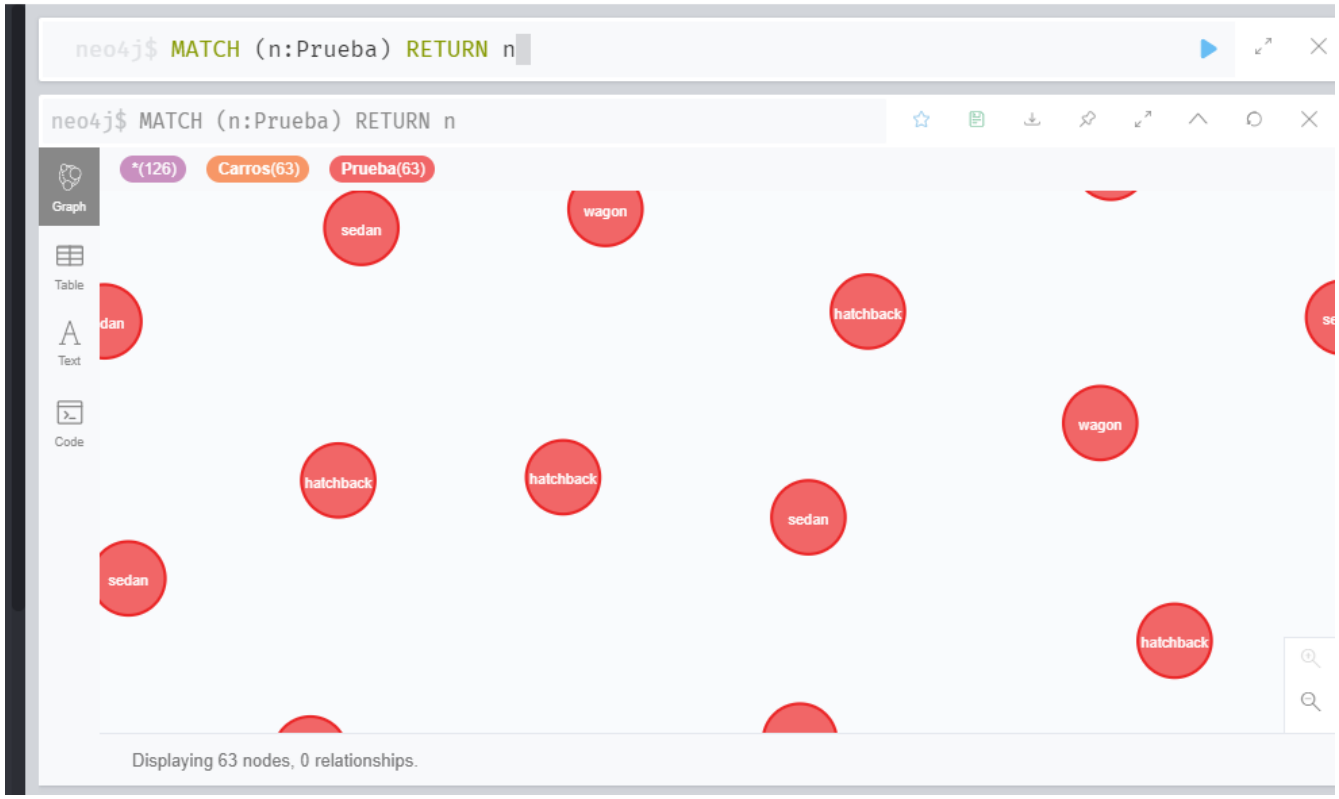


Sacamos los datos de prueba




Se crea la base con el 70%

Se crea la base con el 30%



Creamos el algoritmo knn comparado con los ambos datos

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

        Prueba: {
            label: 'Prueba',
            properties: 'price'
        }
```

```
neo4j$ CALL gds.graph.create( 'Carros', { Entrenamiento: { label:…
```

Table | Text | Code

| nodeProjection | relationshipProjection | graphName | nodeCount | relationshipCount | creat |
|---|---|---|---|---|---|
| { "Entrenamiento": { "properties": { "price": { "property": "price", "defaultValue": null | { "__ALL__": { "orientation": "NATURAL", "aggregation": "DEFAULT", "type": "*", "properties": { } | "Carros" | 206 | 0 | 26 |

Hacemos la comparación entre los dos que creamos

```
1 CALL gds.beta.knn.stream('Carros', {topK: 2, randomSeed: 42,
  nodeWeightProperty: 'price'})
2 YIELD node1, node2, similarity
3 RETURN gds.util.asNode(node1).marca AS Marca1, gds.util.asNode(node2).marca
  AS Marca2, similarity AS Similitud
4 ORDER BY Similitud ASCENDING, Marca1, Marca2
```

neo4j$ CALL gds.beta.knn.stream('Carros', {topK: 2, randomSeed: 4…

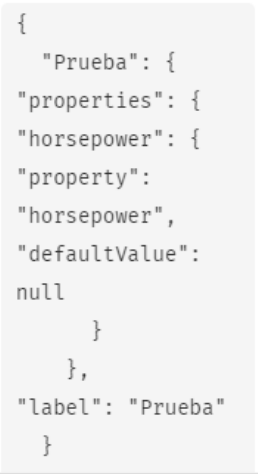
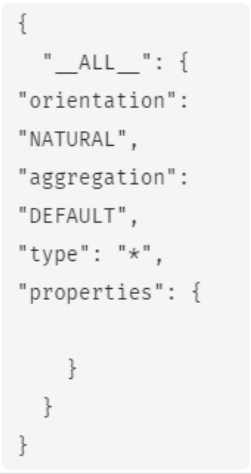
Table | Text | Code

| | Marca1 | Marca2 | Similitud |
|---|---|---|---|
| 19 | "porsche" | "jaguar" | 0.0009718172983479105 |
| 20 | "porsche" | "mercedes-benz" | 0.0009718172983479105 |
| 21 | "porsche" | "mercedes-benz" | 0.001076426264800861 |
| 22 | "bmw" | "jaguar" | 0.0011350737797956867 |
| 23 | "jaguar" | "bmw" | 0.0011350737797956867 |
| 24 | "mercedes-benz" | "mercedes-benz" | 0.001145475372279496 |

```
CALL gds.graph.create(
    'creacion2',
    {
        Prueba : {
            label: 'Prueba',
            properties:'horsepower'
        }
    },
    '*'
);
```

```
neo4j$ CALL gds.graph.create( 'creacion2', { Prueba : { label: 'P…
```

Table | Text | Code

| | nodeProjection | relationshipProjection | graphName | nodeCount | relati |
|---|---|---|---|---|---|
| 1 |  |  | "creacion2" | 63 | 0 |


```
{
  "Prueba": {
"properties": {
"horsepower": {
"property":
"horsepower",
"defaultValue":
null
      }
    },
"label": "Prueba"
  }
```

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

    }
  }
}
```

```
CALL gds.beta.knn.stream('k1Localidadd', {topK: 2, randomSeed: 42,
nodeWeightProperty: 'price'})
YIELD node1, node2, similarity
RETURN gds.util.asNode(node1).marca AS Marca1, gds.util.asNode(node2).marca
AS Marca2, similarity AS Similitud
ORDER BY Similitud ASCENDING, Marca1, Marca2
```

eo4j$ CALL gds.beta.knn.stream('k1Localidadd', {topK: 2, randomS…

| | Marca1 | Marca2 | Similitud |
|---|---|---|---|
| 6 | "mercedes-benz" | "mercedes-benz" | 0.00022517451024544022 |
| 7 | "bmw" | "mercedes-benz" | 0.0002447381302006853 |
| 8 | "mercedes-benz" | "bmw" | 0.0002447381302006853 |
| 9 | "mercedes-benz" | "porsche" | 0.0002542588354945334 |
| 10 | "mercedes-benz" | "volvo" | 0.00034153005464480874 |