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 row
CREATE (a:Auto)
SET a.class = row[1],
 a.features = row[2..];
```

```
neo4j$ LOAD CSV FROM "file:///C:/Users/narcisa/Documents/9no/deberesquisi/DEBERES-SE-IA…
```

Table

Code

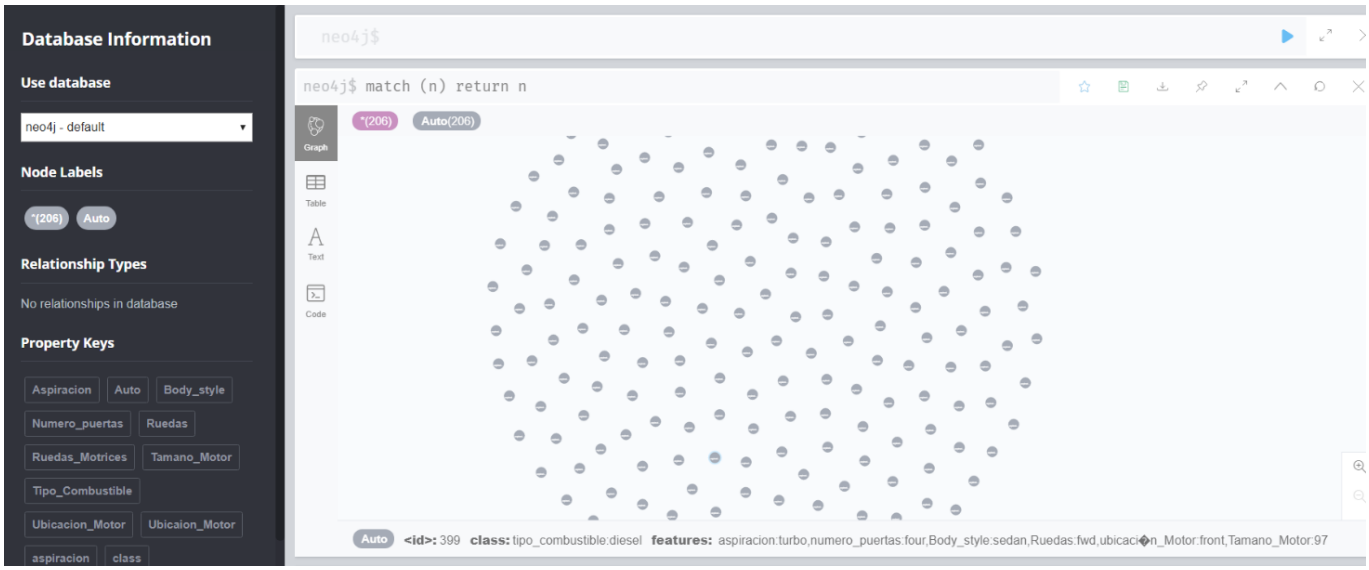
Added 206 labels, created 206 nodes, set 411 properties, completed after 33 ms.

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



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.

Marcamos datos de entrenamiento 70%

MATCH (a:Auto)
WITH a LIMIT 143
SET a:Training;

Marcamos datos de prueba 30%

MATCH (a:Auto)
WITH a SKIP 143
SET a:Test;

```
neo4j$ MATCH (a:Auto) WITH a SKIP 143 SET a:Test;
```

Table

Code

Added 63 labels, completed after 6 ms.

Added 63 labels, completed after 6 ms.

```
neo4j$ MATCH (a:Auto) WITH a LIMIT 143 SET a:Training;
```

Table

Added 143 labels, completed after 10 ms.

Vector de Características

```
1  MATCH (a:Auto)
2  UNWIND a.features as feature
3  WITH a,collect(CASE feature
4  WHEN a.features[0] THEN toInteger(a.features[0])
5  WHEN a.features[1] THEN toInteger(a.features[1])
6  WHEN a.features[2] THEN toInteger(a.features[2])
7  WHEN a.features[3] THEN toInteger(a.features[3])
8  WHEN a.features[4] THEN toInteger(a.features[4])
9  WHEN a.features[5] THEN toInteger(a.features[5])
10 WHEN a.features[6] THEN toInteger(a.features[6])
11 WHEN a.features[7] THEN toInteger(a.features[7])
12 WHEN a.features[8] THEN toInteger(a.features[8])
```

```
neo4j$ MATCH (a:Auto) UNWIND a.features as feature WITH a,coll
```

Table

Set 205 properties, completed after 30 ms.

```
MATCH (test:Test)
WITH test,test.feature_vector as feature_vector
CALL apoc.cypher.run('MATCH (training:Training)
WITH training,gds.alpha.similarity.euclideanDistance($feature_vector, training.feature_vector)
AS similarity
ORDER BY similarity ASC LIMIT 3
RETURN collect(training.class) as classes',
{feature_vector:feature_vector}) YIELD value
WITH test.class as class, apoc.coll.sortMaps(apoc.coll.frequencies(value.classes), '^count')[-
1].item as predicted_class
WITH sum(CASE when class = predicted_class THEN 1 ELSE 0 END) as correct_predictions,
count(*) as total_predictions
RETURN correct_predictions,total_predictions, correct_predictions / toFloat(total_predictions) as
ratio;
```