



**Alumno:**

Juan Cañar

**Docente:**

Ing. Diego Quisi.

**Materia:**

Sistemas Expertos

**Ciclo:**

9no

**Fecha:**

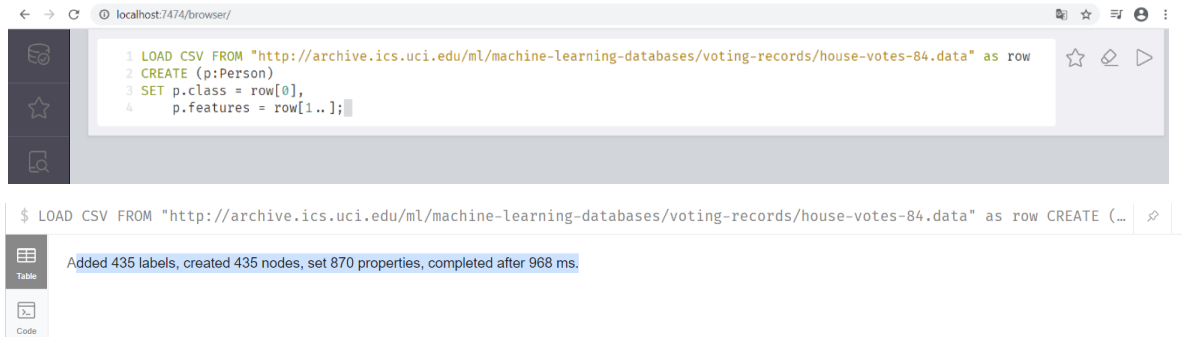
21/05/2020

---

## *kNN Clasificación de miembros del congreso utilizando algoritmos de similitud en Neo4j*

---

1. Se carga el conjunto de datos y se ejecuta en neo4j de la siguiente manera.

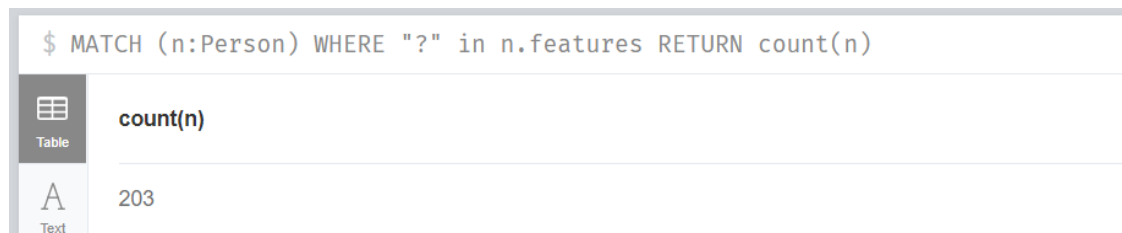


```
1 LOAD CSV FROM "http://archive.ics.uci.edu/ml/machine-learning-databases/voting-records/house-votes-84.data" as row
2 CREATE (p:Person)
3 SET p.class = row[0],
4     p.features = row[1..];
```

\$ LOAD CSV FROM "http://archive.ics.uci.edu/ml/machine-learning-databases/voting-records/house-votes-84.data" as row CREATE (...)

Added 435 labels, created 435 nodes, set 870 properties, completed after 968 ms.

2. Con la siguiente sentencia se ve cuántos miembros del congreso tienen al menos un voto faltante.



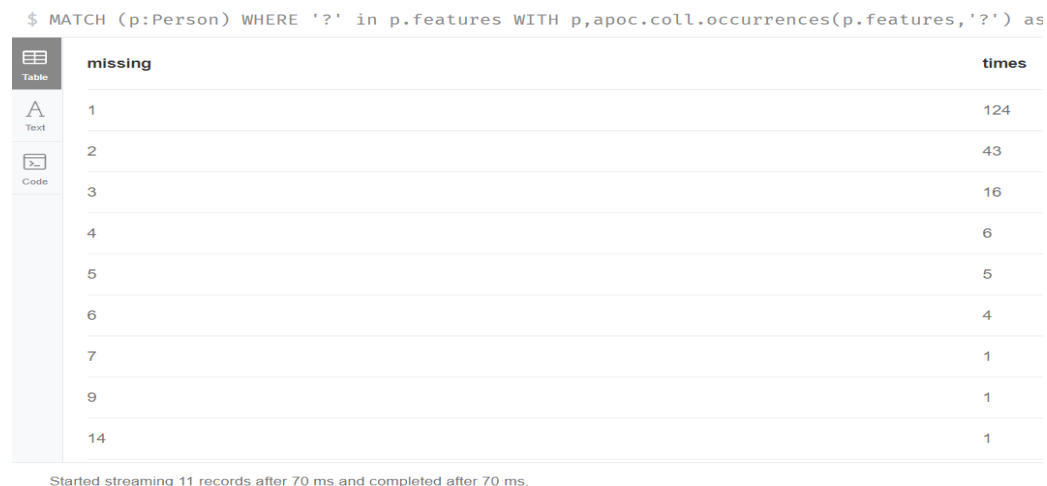
```
$ MATCH (n:Person) WHERE "?" in n.features RETURN count(n)
```

count(n)
203

3. Se ve cual es la distribución de los votos faltantes por miembro.

```
MATCH (p:Person)
WHERE '?' in p.features
WITH p,apoc.coll.occurrences(p.features,'?') as missing
RETURN missing,count(*) as times ORDER BY missing ASC
```

### Resultado:



```
$ MATCH (p:Person) WHERE '?' in p.features WITH p,apoc.coll.occurrences(p.features,'?') as
```

missing	times
1	124
2	43
3	16
4	6
5	5
6	4
7	1
9	1
14	1

Started streaming 11 records after 70 ms and completed after 70 ms.

**4. Se excluye los votos faltantes:**

```
MATCH (p:Person)
WITH p,apoc.coll.occurrences(p.features,'?') as missing
WHERE missing > 6
DELETE p
```

**Resultado:**

```
$ MATCH (p:Person) WITH p,apoc.coll.occurrences(p.features,'?') as missing WHERE missing > 6 DELETE p
```



Deleted 5 nodes, completed after 30 ms.



**5. Marcaremos 344 nodos como subconjunto de entrenamiento y el resto como prueba.**

```
MATCH (p:Person)
WITH p LIMIT 344
SET p:Training;
```

**Resultado:**

```
$ MATCH (p:Person) WITH p LIMIT 344 SET p:Training
```



Added 344 labels, completed after 24 ms.

```
MATCH (p:Person)
WITH p SKIP 344
SET p:Test;
```

**Resultado:**

```
$ MATCH (p:Person) WITH p SKIP 344 SET p:Test
```



Added 86 labels, completed after 4 ms.

**6. Se transforma a vector de características.**

```
MATCH (n:Person)
UNWIND n.features as feature
WITH n,collect(CASE feature WHEN 'y' THEN 1
                WHEN 'n' THEN 0
                ELSE 0.5 END) as feature_vector
SET n.feature_vector = feature_vector
```

```
$ MATCH (n:Person) UNWIND n.features as feature WITH n
```



Set 430 properties, completed after 52 ms.

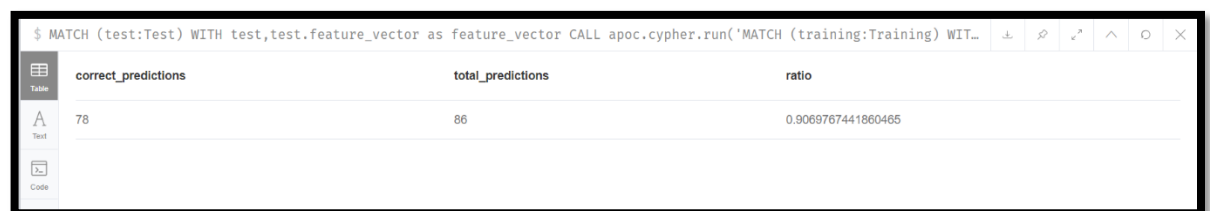
7. Tendremos que usar `apoc.cypher.run` para limitar los resultados de las coincidencias por fila en lugar de por consulta. Esto nos ayudará a recuperar los 3 primeros vecinos por nodo.

### Algoritmo clasificador kNN

Usaremos la distancia euclidiana como la función de distancia y el valor topK de 3. Es aconsejable usar un número impar como K para evitar producir casos extremos, donde, por ejemplo, con los dos vecinos superiores y cada uno con una clase diferente, terminamos sin clase mayoritaria, pero una división 50/50 entre los dos. Tenemos una situación específica en la que queremos comenzar con todos los nodos etiquetados como Prueba y encontrar los tres nodos vecinos principales solo del subconjunto de Entrenamiento. De lo contrario, todos los nodos etiquetados para Prueba también se considerarían parte de los datos de Entrenamiento, que es algo que queremos evitar. Esta es exactamente la razón por la cual no podemos usar una consulta simple como:

#### Código:

```
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
```



The screenshot shows a query execution interface with a toolbar at the top containing icons for undo, redo, copy, paste, and other actions. Below the toolbar is a table with three columns: `correct_predictions`, `total_predictions`, and `ratio`. The table contains one row of data.

correct_predictions	total_predictions	ratio
78	86	0.9069767441860465