



**NOMBRES:**

NICOLÁS AÑAZCO

**CARRERA:**

INGENIERÍA EN SISTEMAS

**MATERIA:**

SISTEMAS EXPERTOS

**FECHA:**

22/05/2020

## Creación de Nodos

La información sobre el conjunto de datos y el conjunto de datos en sí, disponible por Jeff Schlimmer, se puede encontrar en el [repositorio de aprendizaje automático UCI](http://archive.ics.uci.edu/ml/machine-learning-databases/voting-records/house-votes-84.data).

Aquí cargamos todos los datos y creamos los nodos de las Personas (Person)

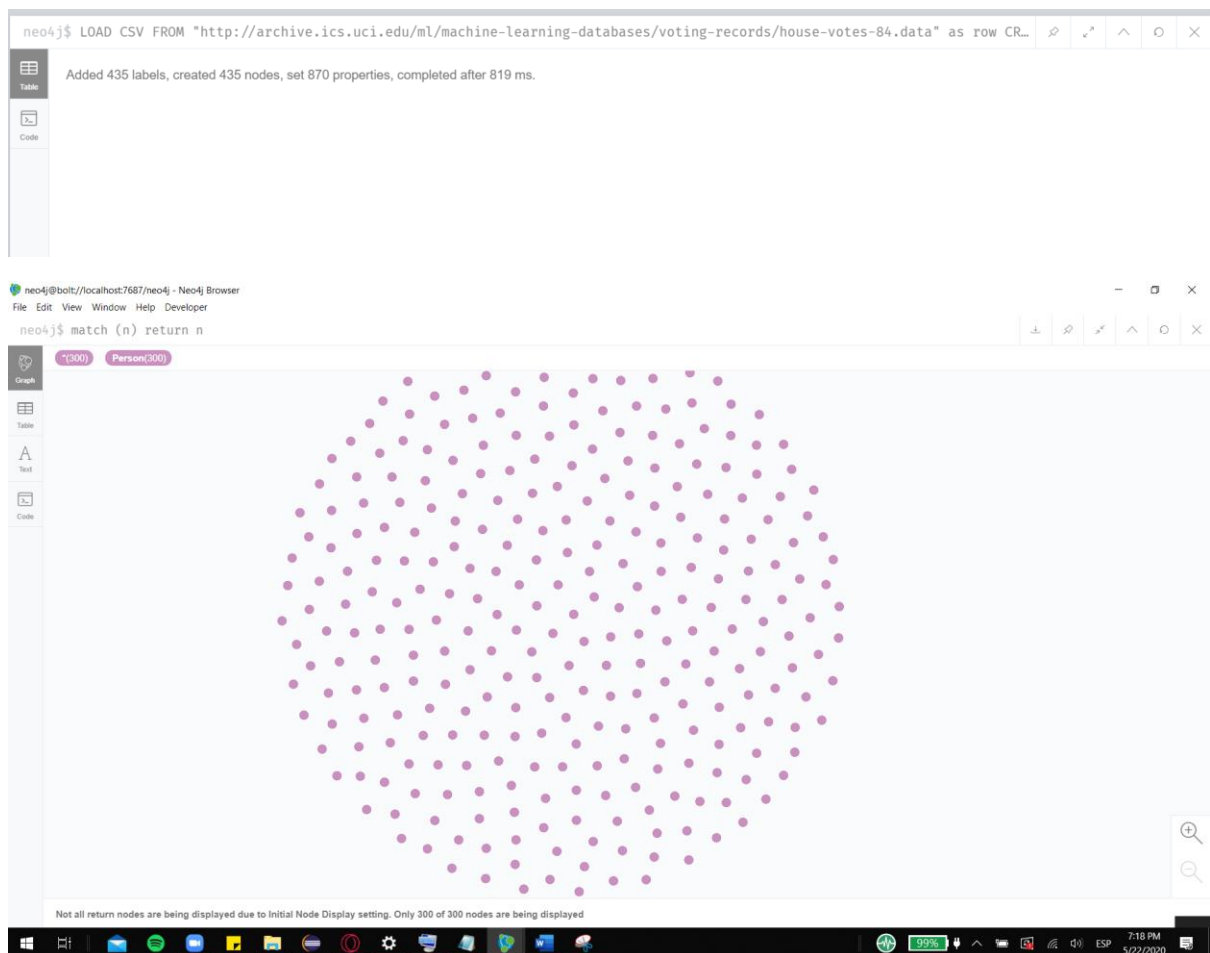
Código:

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

```
CREATE (p:Person)
```

```
SET p.class = row[0],
```

```
    p.features = row[1..];
```



## Votos faltantes

Veamos cuántos miembros del congreso tienen al menos un voto faltante.

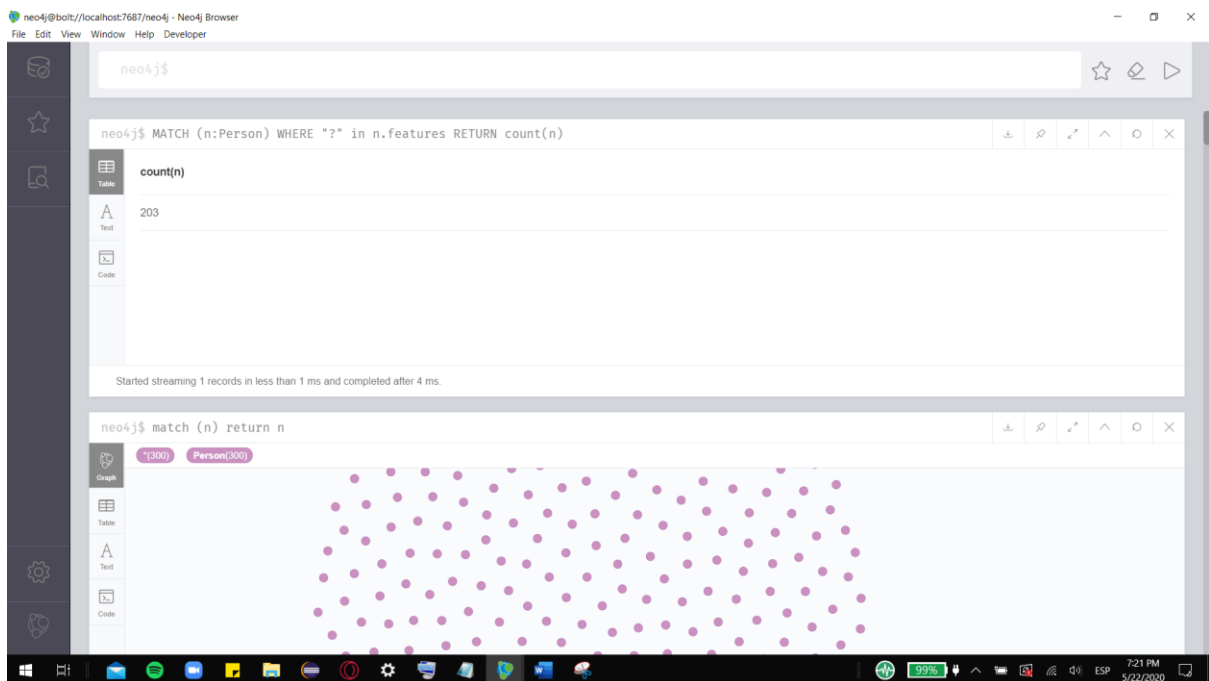
Código:

**MATCH (n:Person)**

**WHERE "?" in n.features**

**RETURN count(n)**

Resultado:



Casi la mitad de los miembros del congreso tienen votos faltantes. Eso es bastante significativo, así que profundicemos más. Revisaremos cuál es la distribución de los votos faltantes por miembro.

Código:

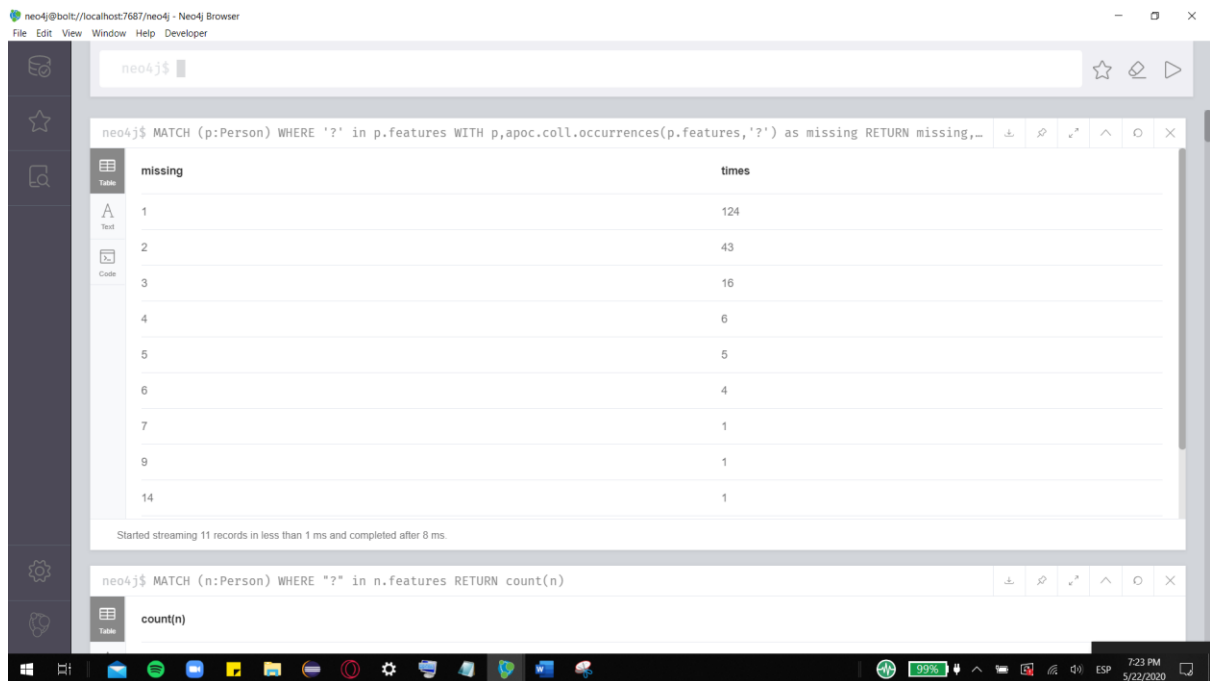
**MATCH (p:Person)**

**WHERE '?' in p.features**

**WITH p,apoc.coll.occurrences(p.features,'?') as missing**

**RETURN missing,count(\*) as times ORDER BY missing ASC**

## Resultado:



The screenshot shows the Neo4j Browser interface. The top bar indicates the connection to 'neo4j@bolt://localhost:7687/neo4j - Neo4j Browser'. The main area displays a Cypher query: `neo4j$ MATCH (p:Person) WHERE '?' in p.features WITH p,apoc.coll.occurrences(p.features,'?') as missing RETURN missing,...`. Below the query, a table titled 'missing' shows the results. The table has two columns: 'missing' and 'times'. The data rows are as follows:

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

Below the table, a status message reads: 'Started streaming 11 records in less than 1 ms and completed after 8 ms.' At the bottom, another query is visible: `neo4j$ MATCH (n:Person) WHERE "?" in n.features RETURN count(n)`, with a table titled 'count(n)' below it. The Windows taskbar at the bottom shows the time as 7:23 PM on 5/22/2020.

Tres miembros casi nunca votaron (14,15,16 votos faltantes) y dos de ellos (7,8 votos faltantes) tienen más del 50% de votos faltantes. Los excluirémos de nuestro análisis posterior para intentar reducir el ruido.

Código:

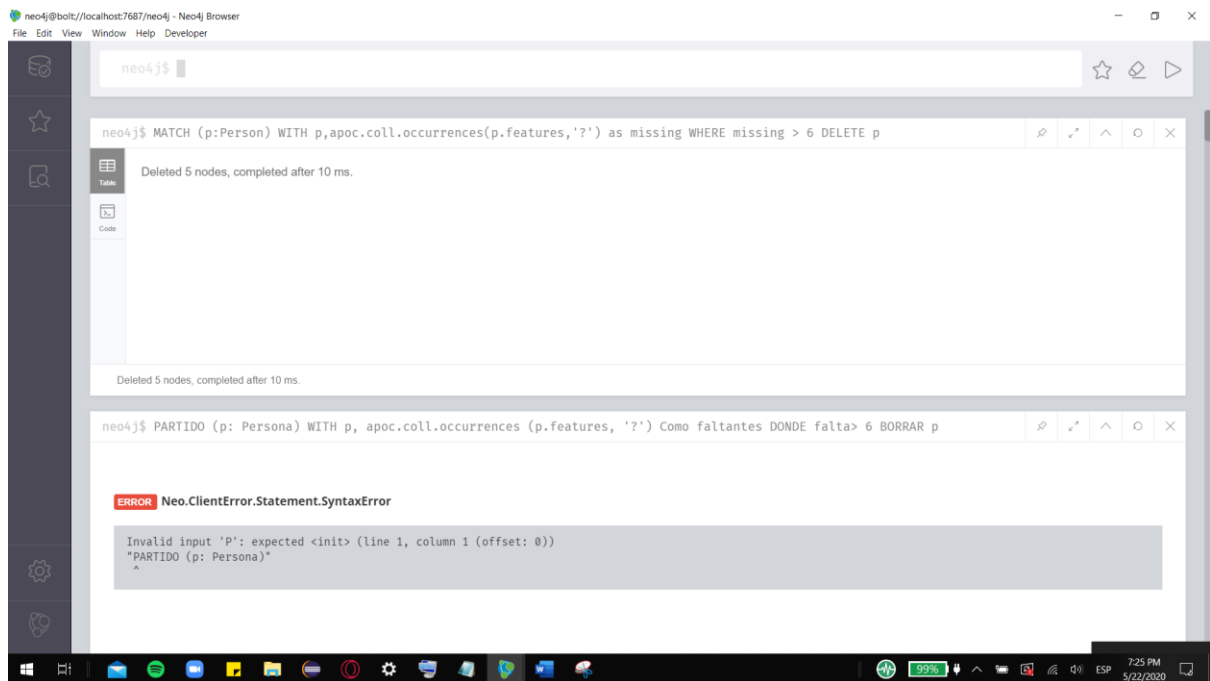
**MATCH (p:Person)**

**WITH p,apoc.coll.occurrences(p.features,'?') as missing**

**WHERE missing > 6**

**DELETE p**

## Resultado:



## Marcar datos de entrenamiento:

**MATCH (p:Person)**

**WITH p LIMIT 344**

**SET p:Training;**

## Marcar datos de prueba:

**PARTIDO (p: Persona)**

**CON p SKIP 344**

**SET p: Prueba;**

**Transformar a vector de características:**

**MATCH (n:Person)**

**UNWIND n.features as feature**

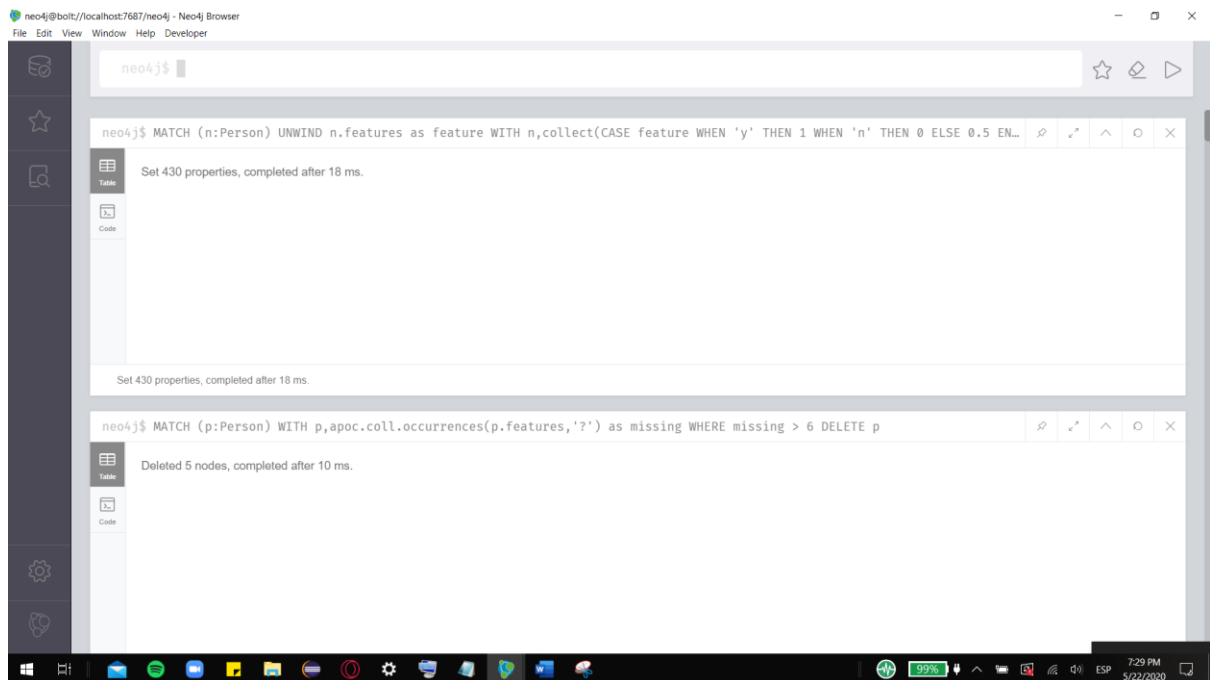
**WITH n,collect(CASE feature WHEN 'y' THEN 1**

**WHEN 'n' THEN 0**

**ELSE 0.5 END) as feature\_vector**

**SET n.feature\_vector = feature\_vector**

**Resultado:**



## **Algoritmo clasificador kNN**

Usaremos la distancia euclideana 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.

Consulta:

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;
```

Resultado:

neo4j\$ MATCH (test:Test) WITH test,test.feature\_vector as feature\_vector CALL apoc.cypher.run('MATCH (training:Training...)

correct_predictions	total_predictions	ratio
84	86	0.9767441860465116

Started streaming 1 records after 53 ms and completed after 615 ms.

neo4j\$ MATCH (n:Person) UNWIND n.features as feature WITH n,collect(CASE feature WHEN 'y' THEN 1 WHEN 'n' THEN 0 ELSE 0.5 EN...

Set 430 properties, completed after 11 ms.