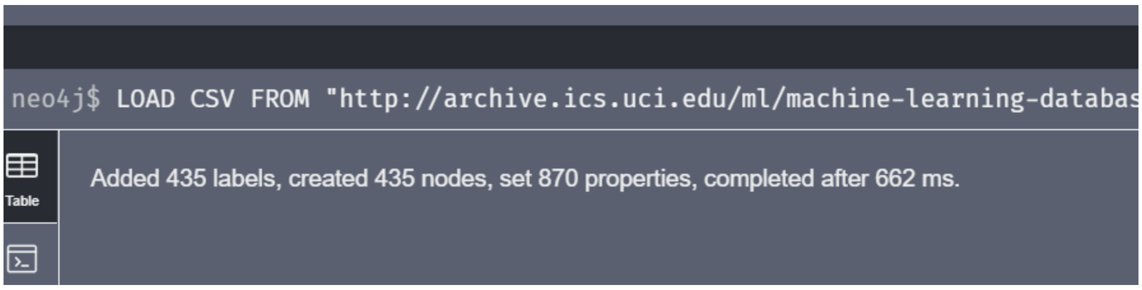
IMPORTAR DATOS A NEO4J

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

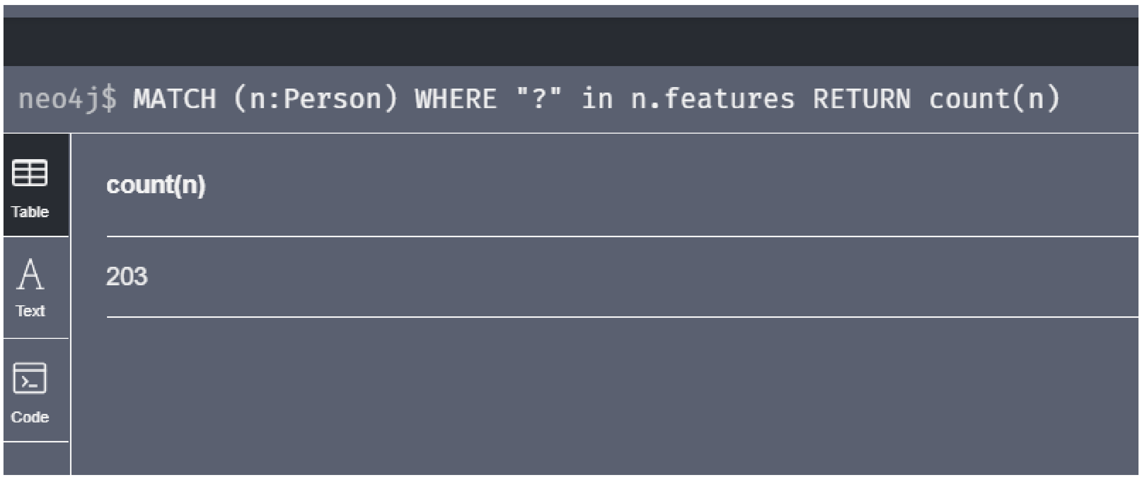


```
neo4j$ LOAD CSV FROM "http://archive.ics.uci.edu/ml/machine-learning-databas
```

Added 435 labels, created 435 nodes, set 870 properties, completed after 662 ms.

Ver votos perdidos

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



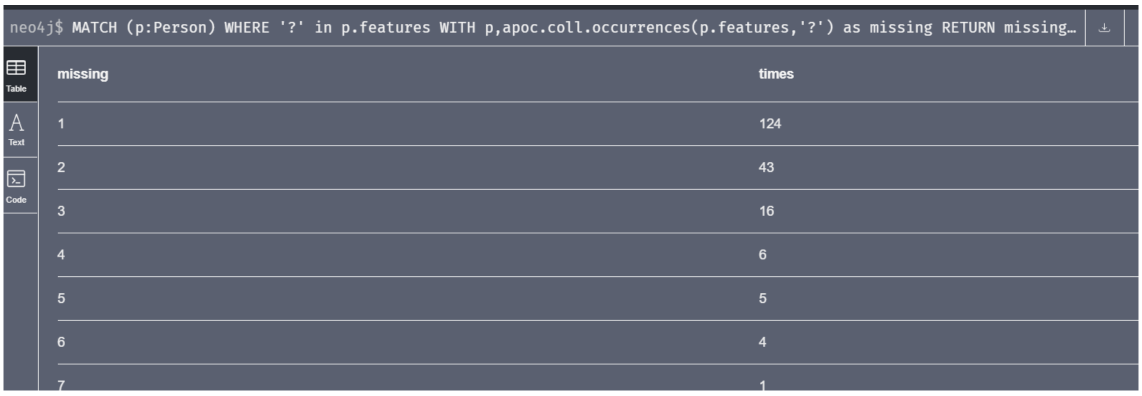
```
neo4j$ MATCH (n:Person) WHERE "?" in n.features RETURN count(n)
```

```
count(n)

203
```

Votos perdidos 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



```
neo4j$ MATCH (p:Person) WHERE '?' in p.features WITH p,apoc.coll.occurrences(p.features,'?') as missing RETURN missing…
```

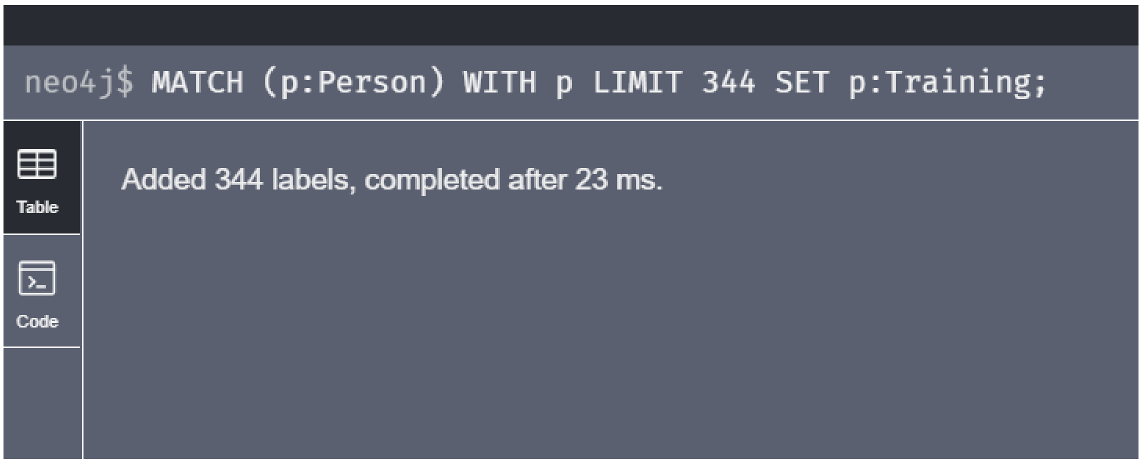
missing	times
1	124
2	43
3	16
4	6
5	5
6	4
7	1

Eliminar votos perdidos mayores a 6

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

Seleccionar datos para entrenamiento que son 80% de los datos

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

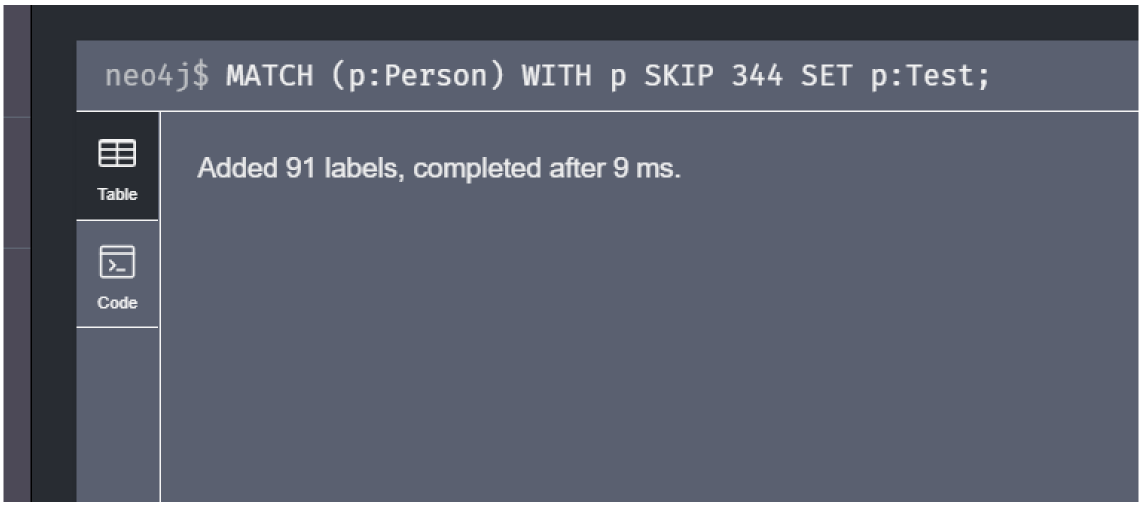


```
neo4j$ MATCH (p:Person) WITH p LIMIT 344 SET p:Training;

Added 344 labels, completed after 23 ms.
```

20%

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

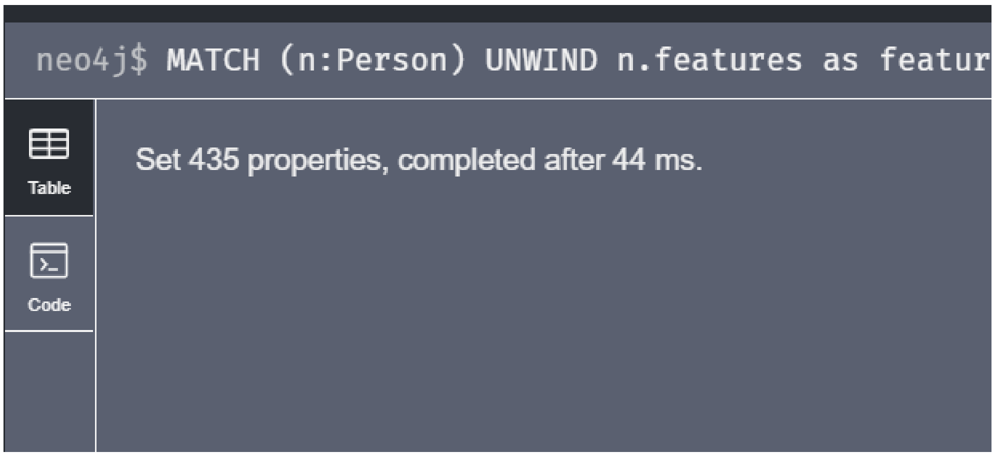


```
neo4j$ MATCH (p:Person) WITH p SKIP 344 SET p:Test;
```

```
Added 91 labels, completed after 9 ms.
```

Creación de vectores para el análisis de la similitud mediante la distancia euclidiana

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

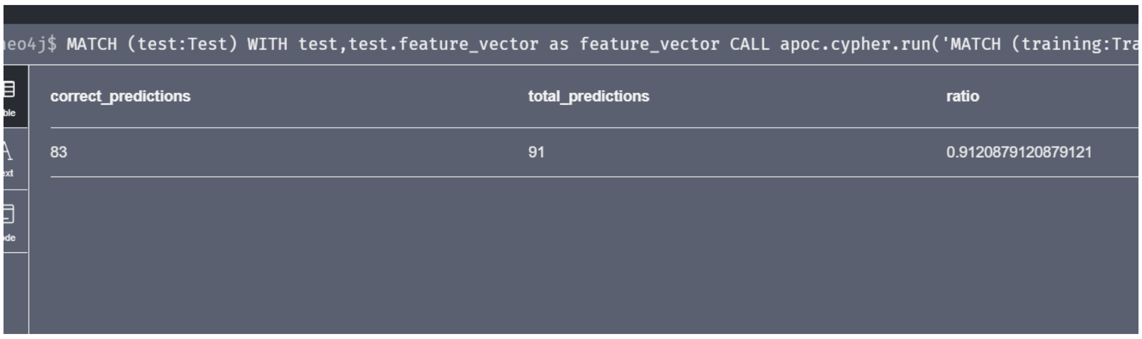


```
neo4j$ MATCH (n:Person) UNWIND n.features as featur
```

```
Set 435 properties, completed after 44 ms.
```

kNN classifier algorithm

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



```
neo4j$ MATCH (test:Test) WITH test,test.feature_vector as feature_vector CALL apoc.cypher.run('MATCH (training:Tra
```

```
correct_predictions          total_predictions          ratio
83                           91                         0.9120879120879121
```