kNN Classification of members of congress using similarity algorithms in Neo4j.

Tenemos que instalar los siguientes plugins.

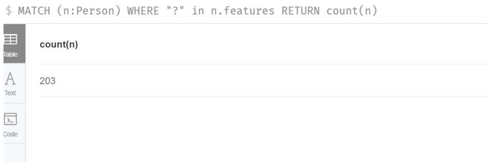
- Neo4j graph algorithms plugin
- Neo4j APOC plugin

1. Código para importar a Neo4j.

```
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..];
```

2. Verificamos el numero de miembros del congreso que tienen menos un voto.

```
MATCH (n:Person)

WHERE "?" in n.features

RETURN count(n)
```

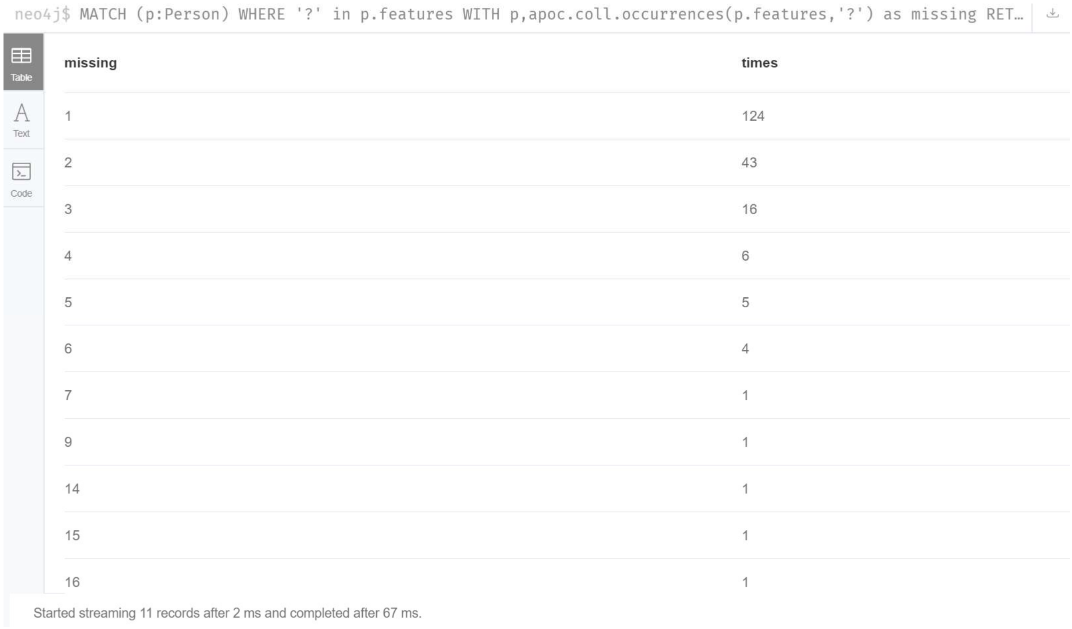


```
$ MATCH (n:Person) WHERE "?" in n.features RETURN count(n)
```

count(n)
203

3. En el resultado anterior, se pude ver que casi la mitad tienen votos faltantes, por lo que con el siguiente codigo se procede a revisar cual es la distribución de los votos faltantes.

```
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 RET…
```

missing	times
1	124
2	43
3	16
4	6
5	5
6	4
7	1
9	1
14	1
15	1
16	1

Started streaming 11 records after 2 ms and completed after 67 ms.

4. Datos de entrenamiento.

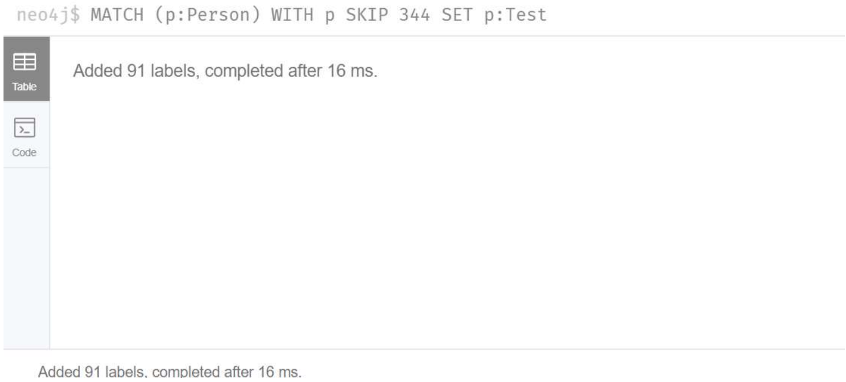
80% datos de entrenamiento.

20% datos de prueba.

Total 344 nodos.

```
MATCH (p:Person)

WITH p LIMIT 344

SET p:Training;
```

5. Marcar datos de entrenamiento.



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

Added 91 labels, completed after 16 ms.

Added 91 labels, completed after 16 ms.

6. Vector de caracteristicas.

Vamos a mapear el vector de las siguientes características; ‘y’ para 1, ‘n’ para 0, y ‘?’ para 0.5

Transformamos el 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
```

7. Algoritmo clasificador KNN.

```
MATCH (test:Test)

WITH test,test.feature_vector as feature_vector

CALL apoc.cypher.run('MATCH (training:Training)

   // calculate euclidian distance between each test node and all training nodes

   WITH training,algo.similarity.euclideanDistance($feature_vector, training.feature_vector) AS similarity

   // return only top 3 nodes

   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
```