Sistemas Expertos
Tema: Basados en casos.

Prueba 2

17/01/2021

Sistemas Expertos

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 C – 2:** Predicción de "género" mediante el valor de "Compra" y el tipo de "Ocupación", para descargar los datos del siguiente link: https://www.kaggle.com/alllexander/blackfriday [3]

- Ingresar cada uno de los datos en un nodo y obtener el grado de similitud se recomienda utilizar la distancia Euclidiana o Person, una vez obtenido la similitud ingresar datos de prueba para validar (Máximo 3 datos).
- Generar otro entorno en donde solo ingrese el 70% de los datos y validar con el 30%.
- Agregar el grafico con los nodos conformados.
- El proceso de programación desarrollado deberá considerar los siguientes aspectos:
  - Se deberá tener un archivo que tenga todos los procesos o código de búsqueda y datos de Neo4j (https://neo4j.com/docs/labs/apoc/current/export/cypher/).

Bibliografía

- [1] https://tech-cookbook.com/2019/11/11/python-machine-learning-knn-example-from-csv-data/
- [2] https://smalldatabrains.com/python-knn/
- [3] https://www.kaggle.com/cengizeralp/practice-1-gender-prediction-with-knn/data

Entrega: Enviar el documento y código hasta las ***23:55*** del domingo ***31 de Enero del 2021***.

Prueba 2

17/01/2021

DESARROLLO

1) **Primero vamos a cargar los datos de tipo csv al Neo4j con el siguiente código:**

```
LOAD CSV FROM "file:/ / /BlackFriday2.csv" AS fila FIELDTERMINATOR ";"
CREATE (p:Persona)
SET p.class = fila[2],
p.features = fila[3..];
```

2) **Hacemos un match(n) return n para ver los nodos generados en Neo4j**

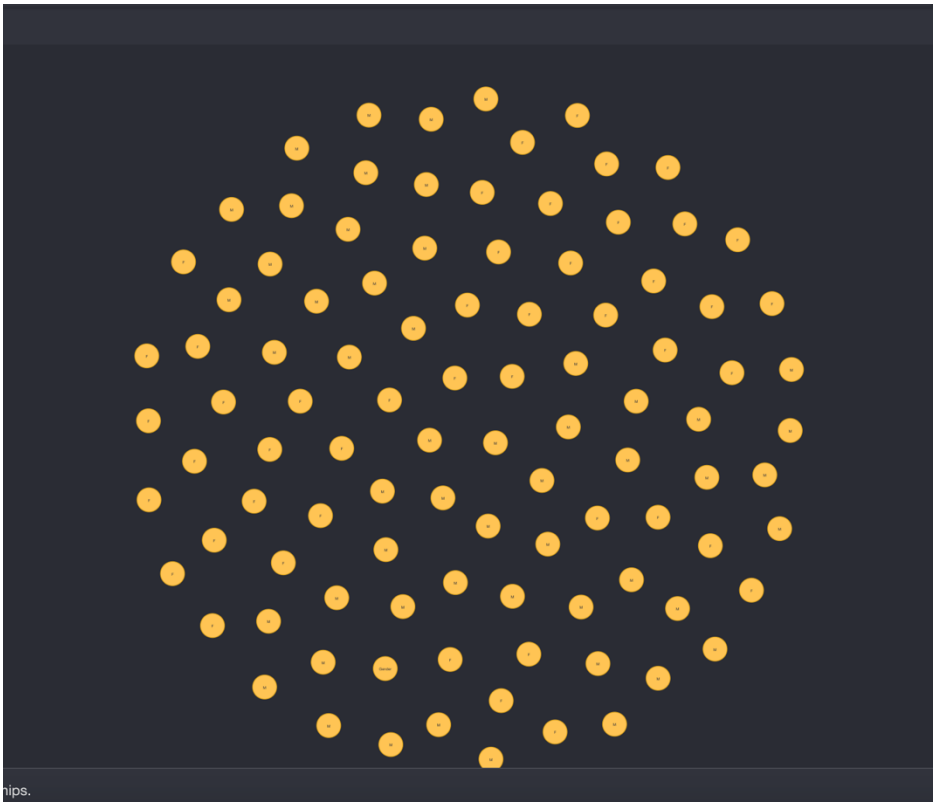


Fig. 1: Gráfico de los nodos creados en Neo4j

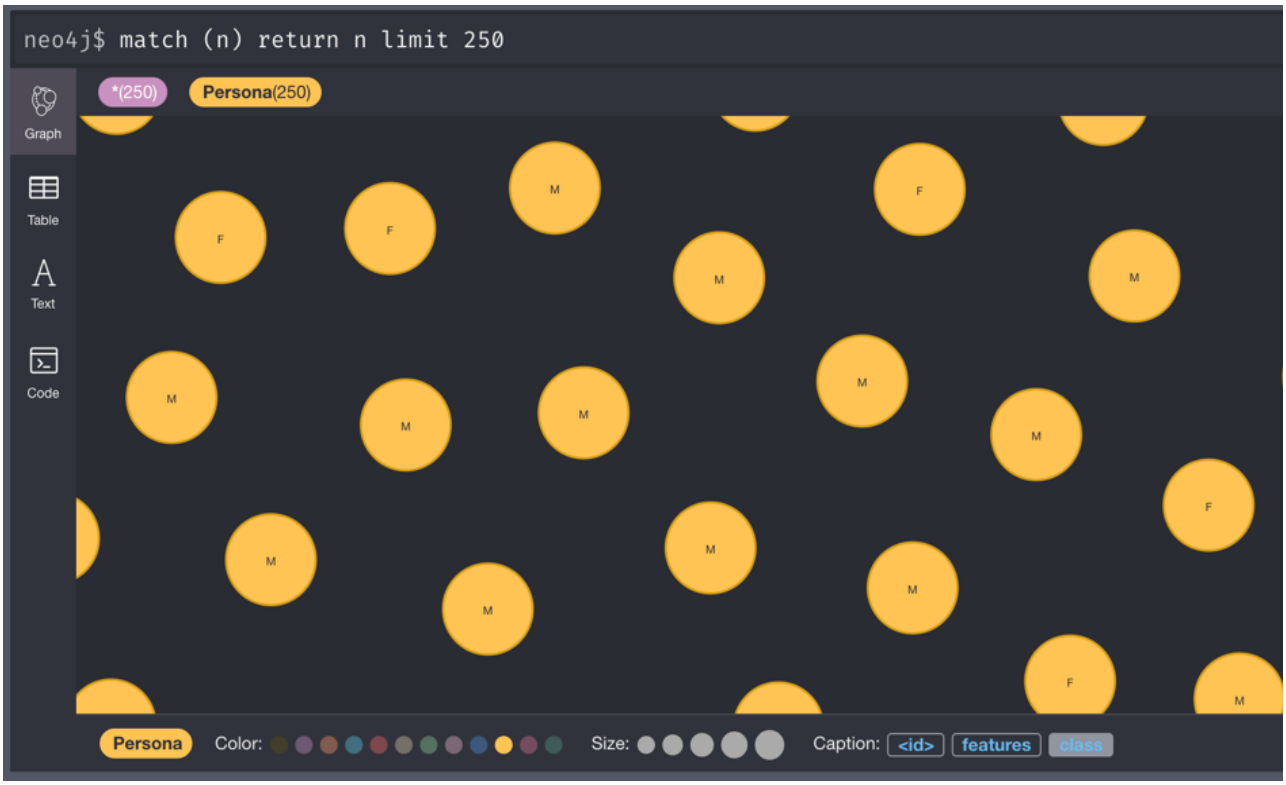


Fig. 2: Representacion de los nodos con M - F

3) Realizamos una conteo de los Nodos que representan a los hombres "M"

- **Codigo Implementado:**

```
MATCH (n)
WHERE "M" in n.class RETURN count(n)
```

- **Resultado Obtenido:**

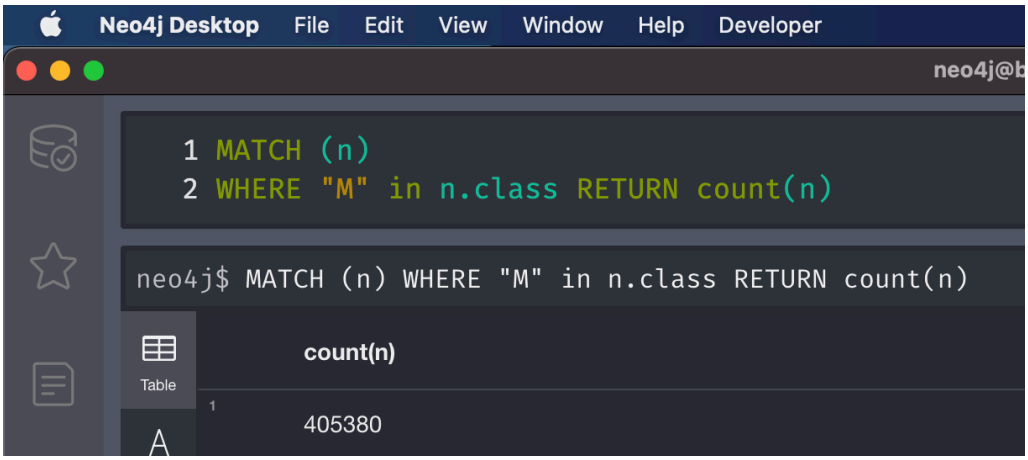


Fig. 3: Gráfico del total de Nodos 'M'

4) **Realizamos una cuenta de los Nodos que representan a las mujeres "F"**

- **Codigo Implementado:**

```
MATCH (n)
WHERE "F" in n.class RETURN count(n)
```

- **Resultado Obtenido:**

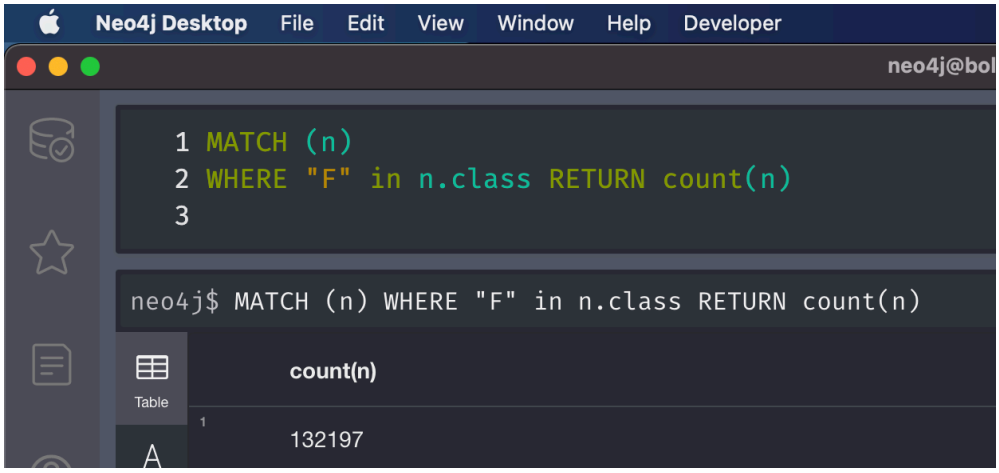


Fig. 4: Gráfico del total de Nodos 'F'

5) **Dividamos nuestro 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 537578 nodos en nuestro gráfico. TotalNodos del 70% es 376304,6. Marcaremos 376304,6 nodos como subconjunto de entrenamiento y el resto como prueba.**

- Trainning

```
MATCH (p:Persona)
WITH p LIMIT 376305
SET p:Training;
```

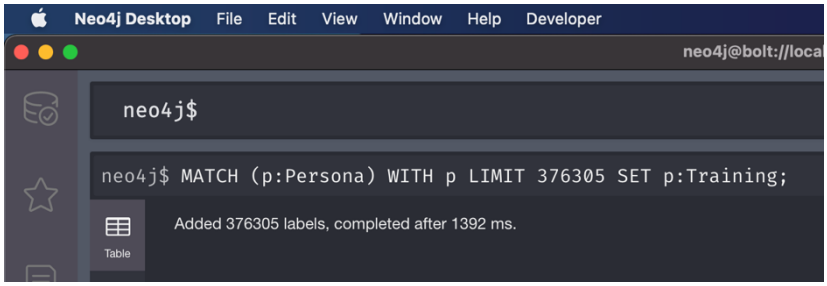
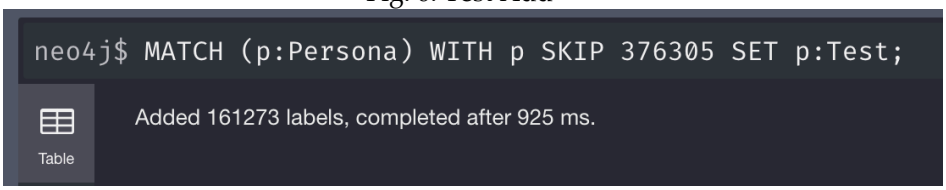


Fig. 5: Training Add

- Test

```
MATCH (p:Persona)
WITH p SKIP 376305
SET p:Test;
```

Fig. 6: Test Add



6) **Transformar a vector de caracteristicas, utilizando el valor de "Compra" y el tipo de "Ocupación".**

```
MATCH (n:Persona)
UNWIND n.features as feature
WITH n,collect(CASE feature
WHEN n.features[1] THEN toInteger(n.features[1])
WHEN n.features[8] THEN toInteger(n.features[8])
END) as feature_vector
SET n.feature_vector = feature_vector
```

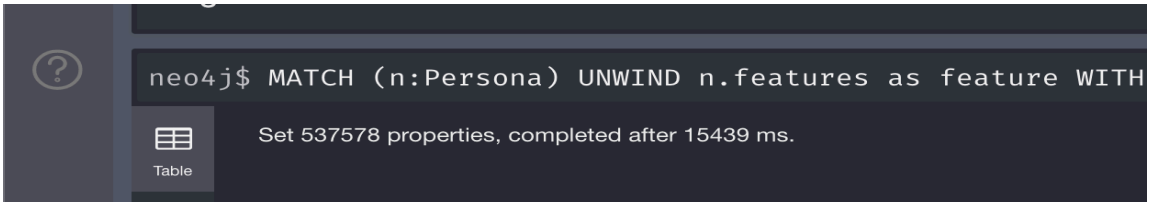


Fig. 7: Vector de caracteristicas

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,gds.alpha.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
```

8) Conclusiones

Necesitaremos usar apoc.cypher.run para limitar los resultados de coincidencia por fila en lugar de por consulta. Esto nos ayudará a recuperar los 3 vecinos principales por nodo.