# Índice General

Fase 1 Solución NoSQL MongoDB 2

1.1 Objetivo 2

1.2 Trabajo a realizar 2

1.3 Solución 3

1.3.1 Estructura de colecciones 3

1.3.2 Carga de Información 3

1.3.3 Consultas 5

Fase 2 Aplicaciones Científicas Empresariales (I) 7

2.1 Objetivo 7

2.2 Trabajo a Realizar 7

2.3 Solución 8

2.3.1 Cargar datos 8

2.3.2 Preprocesado de datos 9

2.3.3 Implementación modelo de clasificación 10

# Solución NoSQL MongoDB

## Objetivo

El objetivo de esta práctica es afianzar los conocimientos impartidos en la parte teórica de la asignatura sobre el uso de la solución NoSQL MongoDB.

## Trabajo a realizar

A partir del siguiente esquema entidad-interrelación:

Diagrama

El contenido generado por IA puede ser incorrecto.

1. Proponga una estructura de colecciones de documentos MongoDB para almacenar la información.
2. Realizar una carga de información lo suficientemente completa para que le sirva como prueba y validación.
3. Realice varias consultas generando informes de salida que le permitan comprobar el acceso a la información.

## Solución

### Estructura de colecciones

Haré una colección para cada tipo de entidad del esquema, una para DEPT y para EMP.

#### Colección DEPT

**{**

"\_id"**:** ObjectId**(**"..."**),**

"DEPTNO"**:** "Identificador del departamento"**,**

"DNAME"**:** "Título del departamento"**,**

"LOC"**:** "Ubicación del departamento"**,**

"MANAGER"**:** "Número de empleado (EMPNO) que lidera el departamento"

**}**

#### Colección EMP

**{**

"\_id"**:** ObjectId**(**"..."**),**

"EMPNO"**:** "Número identificador del empleado"**,**

"ENAME"**:** "Nombre completo del empleado"**,**

"JOB"**:** "Cargo que desempeña el empleado"**,**

"HIREDATE"**:** "Fecha en la que el empleado fue contratado"**,**

"SAL"**:** "Ingreso salarial del empleado"**,**

"COMM"**:** "Bonificación o comisión del empleado"**,**

"DEPTNO"**:** "Número identificador del departamento al que pertenece"**,**

"ISJEFE"**:** "Indica si el empleado es el jefe"**,**

"JEFE"**:** "Si no es jefe, identifica a su jefe "

**}**

### Carga de Información

**from** pymongo **import** MongoClient

cliente **=** MongoClient**(**"mongodb://localhost:27017/"**)**

db **=** cliente**[**"Organizacion"**]**

colecciones **=** **[**"DEPT"**,** "EMP"**]**

**for** coleccion **in** colecciones**:**

db**.***create\_collection***(**coleccion**)**

Texto

El contenido generado por IA puede ser incorrecto.

**Figura 1:** Colecciones creadas

**from** pymongo **import** MongoClient

**from** bson**.***objectid* **import** ObjectId

**from** datetime **import** datetime

# Conexión a MongoDB

cliente **=** MongoClient**(**"mongodb://localhost:27017/"**)**

db **=** cliente**[**"Organizacion"**]**

db**.***DEPT***.***insert\_many***([**

**{**"\_id"**:** ObjectId**(),** "DEPTNO"**:** 10**,** "DNAME"**:** "Contabilidad"**,** "LOC"**:** "New York"**,** "MANAGER"**:** 7839**},**

**{**"\_id"**:** ObjectId**(),** "DEPTNO"**:** 20**,** "DNAME"**:** "Investigación"**,** "LOC"**:** "Dallas"**,** "MANAGER"**:** 7566**},**

**{**"\_id"**:** ObjectId**(),** "DEPTNO"**:** 30**,** "DNAME"**:** "Ventas"**,** "LOC"**:** "Chicago"**,** "MANAGER"**:** 7698**},**

**{**"\_id"**:** ObjectId**(),** "DEPTNO"**:** 40**,** "DNAME"**:** "Operaciones"**,** "LOC"**:** "Boston"**,** "MANAGER"**:** 7782**},**

**{**"\_id"**:** ObjectId**(),** "DEPTNO"**:** 50**,** "DNAME"**:** "Recursos Humanos"**,** "LOC"**:** "Seattle"**,** "MANAGER"**:** 7902**},**

**{**"\_id"**:** ObjectId**(),** "DEPTNO"**:** 60**,** "DNAME"**:** "Legal"**,** "LOC"**:** "San Francisco"**,** "MANAGER"**:** 7654**},**

**{**"\_id"**:** ObjectId**(),** "DEPTNO"**:** 70**,** "DNAME"**:** "Marketing"**,** "LOC"**:** "Los Angeles"**,** "MANAGER"**:** 7499**},**

**{**"\_id"**:** ObjectId**(),** "DEPTNO"**:** 80**,** "DNAME"**:** "IT"**,** "LOC"**:** "Miami"**,** "MANAGER"**:** 7788**},**

**{**"\_id"**:** ObjectId**(),** "DEPTNO"**:** 90**,** "DNAME"**:** "Producción"**,** "LOC"**:** "Denver"**,** "MANAGER"**:** 7521**},**

**{**"\_id"**:** ObjectId**(),** "DEPTNO"**:** 100**,** "DNAME"**:** "Compras"**,** "LOC"**:** "Houston"**,** "MANAGER"**:** 7844**},**

**{**"\_id"**:** ObjectId**(),** "DEPTNO"**:** 110**,** "DNAME"**:** "Calidad"**,** "LOC"**:** "Austin"**,** "MANAGER"**:** 7934**},**

**{**"\_id"**:** ObjectId**(),** "DEPTNO"**:** 120**,** "DNAME"**:** "Logística"**,** "LOC"**:** "Phoenix"**,** "MANAGER"**:** 7900**},**

**{**"\_id"**:** ObjectId**(),** "DEPTNO"**:** 130**,** "DNAME"**:** "Relaciones Públicas"**,** "LOC"**:** "Las Vegas"**,** "MANAGER"**:** 7782**},**

**{**"\_id"**:** ObjectId**(),** "DEPTNO"**:** 140**,** "DNAME"**:** "Desarrollo"**,** "LOC"**:** "San Diego"**,** "MANAGER"**:** 7839**},**

**{**"\_id"**:** ObjectId**(),** "DEPTNO"**:** 150**,** "DNAME"**:** "Soporte"**,** "LOC"**:** "Philadelphia"**,** "MANAGER"**:** 7566**},**

**{**"\_id"**:** ObjectId**(),** "DEPTNO"**:** 160**,** "DNAME"**:** "Finanzas"**,** "LOC"**:** "Portland"**,** "MANAGER"**:** 7698**},**

**{**"\_id"**:** ObjectId**(),** "DEPTNO"**:** 170**,** "DNAME"**:** "Proyectos"**,** "LOC"**:** "Detroit"**,** "MANAGER"**:** 7782**},**

**{**"\_id"**:** ObjectId**(),** "DEPTNO"**:** 180**,** "DNAME"**:** "Ingeniería"**,** "LOC"**:** "Columbus"**,** "MANAGER"**:** 7839**},**

**{**"\_id"**:** ObjectId**(),** "DEPTNO"**:** 190**,** "DNAME"**:** "Seguridad"**,** "LOC"**:** "Charlotte"**,** "MANAGER"**:** 7566**},**

**{**"\_id"**:** ObjectId**(),** "DEPTNO"**:** 200**,** "DNAME"**:** "Investigación y Desarrollo"**,** "LOC"**:** "Indianapolis"**,** "MANAGER"**:** 7698**}**

**])**

db**.***EMP***.***insert\_many***([**

**{**"\_id"**:** ObjectId**(),** "EMPNO"**:** 7839**,** "ENAME"**:** "Rey"**,** "JOB"**:** "Presidente"**,** "HIREDATE"**:** datetime**(**2020**,** 6**,** 9**),** "SAL"**:** 5000**,** "COMM"**:** **None,** "DEPTNO"**:** 10**,** "ISJEFE"**:** **True,** "JEFE"**:** **None},**

**{**"\_id"**:** ObjectId**(),** "EMPNO"**:** 7566**,** "ENAME"**:** "Martinez"**,** "JOB"**:** "Gerente"**,** "HIREDATE"**:** datetime**(**2021**,** 5**,** 19**),** "SAL"**:** 3500**,** "COMM"**:** **None,** "DEPTNO"**:** 20**,** "ISJEFE"**:** **True,** "JEFE"**:** 7839**},**

**{**"\_id"**:** ObjectId**(),** "EMPNO"**:** 7698**,** "ENAME"**:** "Gomez"**,** "JOB"**:** "Gerente"**,** "HIREDATE"**:** datetime**(**2022**,** 3**,** 15**),** "SAL"**:** 3200**,** "COMM"**:** **None,** "DEPTNO"**:** 30**,** "ISJEFE"**:** **True,** "JEFE"**:** 7839**},**

**{**"\_id"**:** ObjectId**(),** "EMPNO"**:** 7782**,** "ENAME"**:** "Perez"**,** "JOB"**:** "Gerente"**,** "HIREDATE"**:** datetime**(**2022**,** 8**,** 5**),** "SAL"**:** 3100**,** "COMM"**:** **None,** "DEPTNO"**:** 40**,** "ISJEFE"**:** **True,** "JEFE"**:** 7839**},**

**{**"\_id"**:** ObjectId**(),** "EMPNO"**:** 7844**,** "ENAME"**:** "Lopez"**,** "JOB"**:** "Vendedor"**,** "HIREDATE"**:** datetime**(**2023**,** 1**,** 20**),** "SAL"**:** 1500**,** "COMM"**:** 300**,** "DEPTNO"**:** 30**,** "ISJEFE"**:** **False,** "JEFE"**:** 7698**},**

**{**"\_id"**:** ObjectId**(),** "EMPNO"**:** 7876**,** "ENAME"**:** "Rodriguez"**,** "JOB"**:** "Analista"**,** "HIREDATE"**:** datetime**(**2023**,** 4**,** 10**),** "SAL"**:** 2800**,** "COMM"**:** **None,** "DEPTNO"**:** 20**,** "ISJEFE"**:** **False,** "JEFE"**:** 7566**}**

**])**

### Consultas

#### Consulta 1: Número empleados

El departamento con más empleados es: 20 con 2 empleados.

*pipeline* **=** **[**

**{**"$group"**:** **{**"\_id"**:** "$DEPTNO"**,** "total\_empleados"**:** **{**"$sum"**:** 1**}}},**

**{**"$sort"**:** **{**"total\_empleados"**:** **-**1**}},**

**{**"$limit"**:** 1**}**

**]**

resultado **=** **list(**db**.***EMP***.***aggregate***(**pipeline**))**

**print(**f"El departamento con más empleados es: {resultado**[**0**][**'\_id'**]**} con {resultado**[**0**][**'total\_empleados'**]**} empleados."**)**

#### Consulta 2: Salario máximo

El empleado con el salario más alto es Rey con un salario de 5000.

*pipeline* **=** **[**

**{**"$sort"**:** **{**"SAL"**:** **-**1**}},**

**{**"$limit"**:** 1**}**

**]**

resultado **=** **list(**db**.***EMP***.***aggregate***(**pipeline**))**

**if** resultado**:**

empleado **=** resultado**[**0**]**

**print(**f"El empleado con el salario más alto es {empleado**[**'ENAME'**]**} con un salario de {empleado**[**'SAL'**]**}."**)**

#### Consulta 3: Número empleados por departamentos

Número de empleados por departamento:

* Departamento 20: 2 empleados
* Departamento 30: 2 empleados
* Departamento 10: 1 empleados
* Departamento 40: 1 empleados

*pipeline* **=** **[**

**{**"$group"**:** **{**"\_id"**:** "$DEPTNO"**,** "total\_empleados"**:** **{**"$sum"**:** 1**}}},**

**{**"$sort"**:** **{**"total\_empleados"**:** **-**1**}}**

**]**

resultado **=** **list(**db**.***EMP***.***aggregate***(**pipeline**))**

**print(**"Número de empleados por departamento:"**)**

**for** departamento **in** resultado**:**

**print(**f"Departamento {departamento**[**'\_id'**]**}: {departamento**[**'total\_empleados'**]**} empleados"**)**

# Aplicaciones Científicas Empresariales (I)

## Objetivo

El objetivo de esta práctica es afianzar los conocimientos impartidos en la parte teórica de la asignatura sobre la caracterización y predicción de actividades/propiedades de compuestos.

## Trabajo a Realizar

Utilizando las bases de datos de compuestos del ejemplo práctico desarrollado en la parte teórica de la asignatura, implementar el código python necesario para predecir (clasificación) el tipo de actividad biológica (Active “1”; Inactive ”0”). Recomendaciones para la implementación:

1. Utilice alguno de los modelos de clasificación implementados en la biblioteca scikit-learn.
2. El tipo de actividad de cada compuesto (variable respuesta) está almacenado en el campo “class” de la colección “molecules”.
3. Como variables predictoras se utilizará el fingerprint molecular de cada compuesto. Recuerde que en la base de datos sólo están almacenadas las posiciones de los bits con valor “1”.

Las variables predictoras y la respuesta quedarían representadas de siguiente forma:

Tabla

El contenido generado por IA puede ser incorrecto.

1. Las columnas donde todos los valores sean ‘0’ pueden ser eliminadas.
2. Para evaluar el desempeño de los clasificadores se deben utilizar métricas clásicas como GMean, Kappa, Accuracy, AURoc, etc.

## Solución

### Cargar datos

Primero, cargaré las bases de datos:

**import** seaborn **as** sns

client **=** MongoClient**(**"mongodb://localhost:27017/"**);**

database\_names **=** client**.***list\_database\_names***();**

**def** load\_database**(**db\_name**):**

**if** db\_name **in** database\_names**:**

**print(**'The database ' **+** db\_name **+** ' exists'**);**

db**=**client**.***get\_database***(**db\_name**);**

**else:**

**print(**'&&&&&& The database ' **+** db\_name **+** ' must be loaded &&&&&.'**);**

**print()**

sys**.exit()**

**return** db

db\_CDS29**=**load\_database**(**'CDS29'**);**

db\_CDS16**=**load\_database**(**'CDS16'**);**

**print(**"\nCDS29: "**);**

**print(**"- "**,** db\_CDS29**.***list\_collection\_names***());**

**print(**"- "**,** db\_CDS29**.***CDS29***.***molecules***);**

**print(**"- "**,** db\_CDS29**.***CDS29***.***mfp\_counts***);**

**print(**"\nCDS16"**);**

**print(**"- "**,** db\_CDS16**.***list\_collection\_names***());**

**print(**"- "**,** db\_CDS16**.***CDS29***.***molecules***);**

**print(**"- "**,** db\_CDS16**.***CDS29***.***mfp\_counts***);**

Obtenemos la siguiente salida:

*CDS29***:**

**-** **[**'molecules'**,** 'mfp\_counts'**]**

**-** Collection**(**Database**(**MongoClient**(**host**=[**'localhost:27017'**],** document\_class**=dict,** tz\_aware**=False,** connect**=True),** 'CDS29'**),** 'CDS29.molecules'**)**

**-** Collection**(**Database**(**MongoClient**(**host**=[**'localhost:27017'**],** document\_class**=dict,** tz\_aware**=False,** connect**=True),** 'CDS29'**),** 'CDS29.mfp\_counts'**)**

*CDS16***:**

**-** **[**'mfp\_counts'**,** 'molecules'**]**

**-** Collection**(**Database**(**MongoClient**(**host**=[**'localhost:27017'**],** document\_class**=dict,** tz\_aware**=False,** connect**=True),** 'CDS16'**),** 'CDS29.molecules'**)**

**-** Collection**(**Database**(**MongoClient**(**host**=[**'localhost:27017'**],** document\_class**=dict,** tz\_aware**=False,** connect**=True),** 'CDS16'**),** 'CDS29.mfp\_counts'**)**

Y cargo los datos de las moléculas de CDS16 y CDS29.

**import** pandas **as** pd

**def** load\_data**(**db**,** collection\_name**):**

collection **=** db**[**collection\_name**]**

cursor **=** collection**.***find***()**

data **=** pd**.***DataFrame***(list(**cursor**))**

**return** data

data\_CDS29 **=** load\_data**(**db\_CDS29**,** 'molecules'**)**

data\_CDS16 **=** load\_data**(**db\_CDS16**,** 'molecules'**)**

Captura de pantalla con letras y números

El contenido generado por IA puede ser incorrecto.

**Figura 2:** Head de data\_CDS16

Texto

El contenido generado por IA puede ser incorrecto.

**Figura 3:** Head de data\_CDS19

### Preprocesado de datos

Aplicaré un RandomForest. Para esta tarea convierto mfp en una matriz binaria para poder utilizarla como características de nuestro modelo.

**import** pandas **as** pd

**import** numpy **as** np

**from** sklearn**.***model\_selection* **import** train\_test\_split

**from** sklearn**.***ensemble* **import** RandomForestClassifier

**from** sklearn**.***metrics* **import** accuracy\_score**,** roc\_auc\_score**,** cohen\_kappa\_score**,** balanced\_accuracy\_score

# Función para transformar los fingerprints en un DataFrame binario

**def** transform\_fingerprints**(**data**,** num\_bits**=**1024**):**

fingerprints **=** np**.***zeros***((len(**data**),** num\_bits**))**

**for** i**,** fp **in** **enumerate(**data**[**'mfp'**]):**

**for** bit **in** fp**[**'bits'**]:**

**if** bit **<** num\_bits**:**

fingerprints**[**i**,** bit**]** **=** 1

**return** pd**.***DataFrame***(**fingerprints**,** columns**=[**f'FP{i}' **for** i **in** **range(**num\_bits**)])**

**for** dataset**,** name **in** **zip([**data\_CDS29**,** data\_CDS16**],** **[**'CDS29'**,** 'CDS16'**]):**

**print(**f'Procesando dataset {name}'**)**

X **=** transform\_fingerprints**(**dataset**)**

# Convertir la variable de respuesta a binario

dataset**[**'class'**]** **=** dataset**[**'class'**].map({**'Active'**:** 1**,** 'Inactive'**:** 0**})**

y **=** dataset**[**'class'**]**

# Eliminar columnas con solo ceros

X **=** X**.***loc***[:,** **(**X **!=** 0**).any(**axis**=**0**)]**

# Dividir en conjunto de entrenamiento y prueba

X\_train**,** X\_test**,** y\_train**,** y\_test **=** train\_test\_split**(**X**,** y**,** test\_size**=**0.2**,** random\_state**=**42**,** stratify**=**y**)**

### Implementación modelo de clasificación

# Entrenar con Random Forest

clf **=** RandomForestClassifier**(**n\_estimators**=**100**,** random\_state**=**42**)**

clf**.***fit***(**X\_train**,** y\_train**)**

# Predicciones

y\_pred **=** clf**.***predict***(**X\_test**)**

y\_prob **=** clf**.***predict\_proba***(**X\_test**)[:,** 1**]**

# Evaluación

accuracy **=** accuracy\_score**(**y\_test**,** y\_pred**)**

roc\_auc **=** roc\_auc\_score**(**y\_test**,** y\_prob**)**

kappa **=** cohen\_kappa\_score**(**y\_test**,** y\_pred**)**

gmean **=** balanced\_accuracy\_score**(**y\_test**,** y\_pred**)**

# Métricas

**print(**f'Resultados para {name}:'**)**

**print(**f'Accuracy: {accuracy:.4f}'**)**

**print(**f'ROC AUC: {roc\_auc:.4f}'**)**

**print(**f'Cohen Kappa: {kappa:.4f}'**)**

**print(**f'G-Mean: {gmean:.4f}\n'**)**

#### Resultados

Obtengo los siguientes resultados:

* CDS29:
* Accuracy: 0.8061
* ROC AUC: 0.7021
* Cohen Kappa: 0.1418
* G-Mean: 0.5508
* CDS16:
* Accuracy: 0.8235
* ROC AUC: 0.9653
* Cohen Kappa: 0.6483
* G-Mean: 0.8264

#### Interpretación

En cuanto al **Accuracy**, CDS16 alcanza un 82.35%, superando el 80.61% de CDS29, lo que indica un mejor rendimiento general del modelo en la predicción de la clase correcta. Sin embargo, la diferencia más significativa se observa en la métrica **ROC AUC**, donde CDS16 obtiene un valor de 0.9653 frente a 0.7021 en CDS29. Esto sugiere que el modelo en CDS16 es mucho más efectivo en la discriminación entre clases activas e inactivas.

**Cohen Kappa** ajusta el efecto del [azar](https://es.wikipedia.org/wiki/Azar) en la proporción de la concordancia observada. En CDS29, el valor de 0.1418 indica una concordancia baja. Por otra parte, en CDS16 tenemos 0.6483 que refleja una concordancia moderada, sugiriendo un mejor desempeño del modelo en CDS16.

**G-Mean** evalúa el equilibrio entre la sensibilidad y la especificidad, útil en conjuntos de datos desbalanceados. En CDS29, el valor de 0.5508 sugiere dificultades en la clasificación de ambas clases, mientras que en CDS16, el 0.8264 indica que el modelo distingue mejor entre compuestos activos e inactivos.