

# NoSQL - Bayesian Classifier

## Integrantes

- Javier Monzón (20121248, javier.monzon@pucp.edu.pe)
- Daniel Sanchez (20130495, hdsanchez@pucp.edu.pe)

## Introducción

La detección de fraude en sistemas financieros es una tarea crítica.

Este proyecto presenta un enfoque probabilístico para inferir si una transacción fue fraudulenta, usando técnicas de inferencia bayesiana y estructuras causales definidas por el usuario.

## README

[https://github.com/daneelsan/PUCP\\_NoSQL\\_BayesianClassifier/blob/main/Classifier/README.md](https://github.com/daneelsan/PUCP_NoSQL_BayesianClassifier/blob/main/Classifier/README.md)

## Entorno de Ejecución



Figura 3: Entorno de ejecución

## Dataset

[fraud\\_credit\\_card.csv](#)

```
SystemOpen["fraud_credit_card.csv"]
```

La data representa transacciones financieras.

Cada fila es una transacción individual y contiene información detallada sobre ella:

- **step:** Un paso o índice de tiempo.
- **customer:** Un identificador único para el cliente que realiza la transacción.
- **age:** La edad del cliente, probablemente en categorías o rangos ('0', '1', '2', etc., 'U' para desconocido).

- **gender:** El género del cliente ('M' para masculino, 'F' para femenino, y posiblemente 'E' o 'U' para otros/desconocido).
- **zipcodeOri:** El código postal de origen de la transacción.
- **merchant:** Un identificador único para el comercio donde se realizó la transacción.
- **zipMerchant:** El código postal del comercio.
- **category:** La categoría o tipo de la transacción (ej. 'es\_transportation' para transporte, 'es\_health' para salud).
- **amount:** El valor monetario de la transacción.
  - Importante: Utiliza coma (,) como separador decimal (ej. "4,55" es 4.55).
- **fraud:** La variable objetivo. Es un indicador binario (0 o 1) que señala si la transacción fue fraudulenta (1) o no (0).

En resumen, son registros de transacciones con características del cliente, del comercio y de la propia operación, etiquetadas como **fraudulentas** o **no fraudulentas**.

El propósito principal de este código es identificar automáticamente si una transacción es fraudulenta o no, utilizando un Clasificador Bayesiano.

Es una herramienta para detectar y prevenir el fraude en transacciones financieras de manera automatizada.

```

1 step, customer, age, gender, zipcodeOri, merchant, zipMerchant, category, amount, fraud
2 0, 'C1093826151', '4', 'M', '28007', 'M348934600', '28007', 'es_transportation', "4,55", 0
3 0, 'C352968107', '2', 'M', '28007', 'M348934600', '28007', 'es_transportation', "39,68", 0
4 0, 'C2054744914', '4', 'F', '28007', 'M1823072687', '28007', 'es_transportation', "26,89", 0
5 0, 'C1760612790', '3', 'M', '28007', 'M348934600', '28007', 'es_transportation', "17,25", 0
6 0, 'C757503768', '5', 'M', '28007', 'M348934600', '28007', 'es_transportation', "35,72", 0
7 0, 'C1315400589', '3', 'F', '28007', 'M348934600', '28007', 'es_transportation', "25,81", 0
8 0, 'C765155274', '1', 'F', '28007', 'M348934600', '28007', 'es_transportation', "9,1", 0
9 0, 'C202531238', '4', 'F', '28007', 'M348934600', '28007', 'es_transportation', "21,17", 0
10 0, 'C105845174', '3', 'M', '28007', 'M348934600', '28007', 'es_transportation', "32,4", 0
11 0, 'C39858251', '5', 'F', '28007', 'M348934600', '28007', 'es_transportation', "35,4", 0
12 0, 'C98707741', '4', 'F', '28007', 'M348934600', '28007', 'es_transportation', "14,95", 0
13 0, 'C1551465414', '1', 'M', '28007', 'M1823072687', '28007', 'es_transportation', "1,51", 0
14 0, 'C623601481', '3', 'M', '28007', 'M50039827', '28007', 'es_health', "68,79", 0
15 0, 'C1865204568', '5', 'M', '28007', 'M1823072687', '28007', 'es_transportation', "20,32", 0
16 0, 'C490238464', '3', 'M', '28007', 'M348934600', '28007', 'es_transportation', "13,56", 0
17 0, 'C194016923', '3', 'F', '28007', 'M348934600', '28007', 'es_transportation', "30,19", 0
18 0, 'C1207205377', '4', 'M', '28007', 'M1823072687', '28007', 'es_transportation', "17,54", 0
19 0, 'C834963773', '5', 'F', '28007', 'M348934600', '28007', 'es_transportation', "40,69", 0
20 0, 'C1897705669', '2', 'M', '28007', 'M348934600', '28007', 'es_transportation', "21,21", 0
21 0, 'C124539163', '2', 'F', '28007', 'M348934600', '28007', 'es_transportation', "10,09", 0
22 0, 'C1687101094', '2', 'F', '28007', 'M348934600', '28007', 'es_transportation', "19,31", 0
23 0, 'C1695454092', '2', 'M', '28007', 'M348934600', '28007', 'es_transportation', "44,22", 0
24 0, 'C986553990', '5', 'F', '28007', 'M348934600', '28007', 'es_transportation', "44,39", 0
25 0, 'C819690995', '5', 'F', '28007', 'M348934600', '28007', 'es_transportation', "30,72", 0
26 0, 'C1622124632', '2', 'M', '28007', 'M348934600', '28007', 'es_transportation', "29,84", 0
27 0, 'C187514477', '3', 'M', '28007', 'M348934600', '28007', 'es_transportation', "12,1", 0
28 0, 'C272748313', '4', 'M', '28007', 'M348934600', '28007', 'es_transportation', "24,84", 0
29 0, 'C490092965', '2', 'M', '28007', 'M348934600', '28007', 'es_transportation', "16,42", 0
30 0, 'C546957379', '5', 'M', '28007', 'M348934600', '28007', 'es_transportation', "2,19", 0
31 0, 'C1563705147', '5', 'F', '28007', 'M348934600', '28007', 'es_transportation', "32,27", 0
32 0, 'C1273110804', '4', 'M', '28007', 'M348934600', '28007', 'es_transportation', "28,09", 0
33 0, 'C1582366224', '5', 'M', '28007', 'M1823072687', '28007', 'es_transportation', "16,13", 0
34 0, 'C998987490', '2', 'F', '28007', 'M348934600', '28007', 'es_transportation', "32,7", 0
35 0, 'C1413412440', '2', 'F', '28007', 'M348934600', '28007', 'es_transportation', "25,53", 0
36 0, 'C1166355595', '4', 'F', '28007', 'M348934600', '28007', 'es_transportation', "39,22", 0
37 0, 'C719598710', '5', 'F', '28007', 'M348934600', '28007', 'es_transportation', "46,14", 0
38 0, 'C1161949399', '2', 'M', '28007', 'M348934600', '28007', 'es_transportation', "29,55", 0
39 0, 'C60351691', '2', 'F', '28007', 'M1823072687', '28007', 'es_transportation', "36,88", 0
40 0, 'C1769927077', '2', 'F', '28007', 'M348934600', '28007', 'es_transportation', "36,15", 0
41 0, 'C995844287', '5', 'F', '28007', 'M348934600', '28007', 'es_transportation', "2,81", 0

```

```
In[6]:= original = Import["./fraud_credit_card.csv", "Tabular"];
original = TransformColumns[original, "amount" → Function[
  ColumnwiseThread[ToExpression@StringReplace[#, ",," → "."]]]];
original = TransformColumns[original, {
  "age" → Function[ColumnwiseThread[ToExpression@StringTake[#, {2, -2}]]],
  "gender" → Function[ColumnwiseThread[StringTake[#, {2, -2}]]]
}]
}

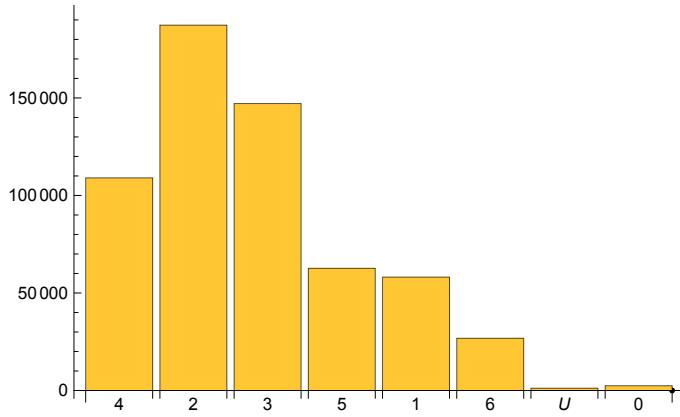
Out[6]=
```

	step	customer	age	gender	zipcodeOri	merchant	zipMerchant	cate
6	0	'C1315400589'	3	F	'28007'	'M348934600'	'28007'	'es_
7	0	'C765155274'	1	F	'28007'	"M348934600"	'28007'	'es_
8	0	'C202531238'	4	F	'28007'	'M348934600'	'28007'	'es_
9	0	'C105845174'	3	M	'28007'	'M348934600'	'28007'	'es_
10	0	'C39858251'	5	F	'28007'	'M348934600'	'28007'	'es_
11	0	'C98707741'	4	F	'28007'	'M348934600'	'28007'	'es_
12	0	'C1551465414'	1	M	'28007'	"M1823072687"	'28007'	'es_
13	0	'C623601481'	3	M	'28007'	"M50039827"	'28007'	'es_
14	0	'C1865204568'	5	M	'28007'	"M1823072687"	'28007'	'es_

### age

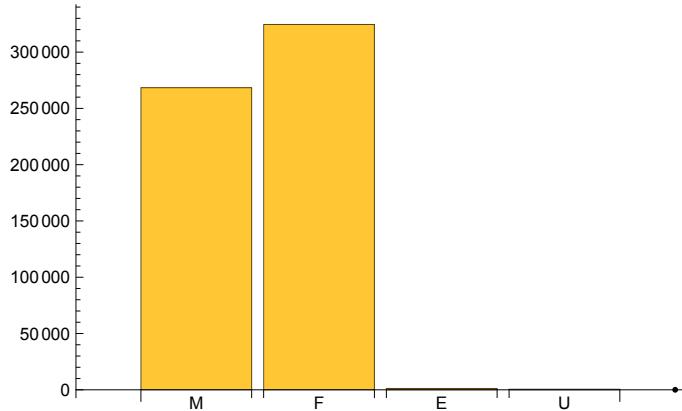
```
In[7]:= BarChart[Counts[original[[All, "age"]]], ChartLabels → Automatic]
```

Out[7]=

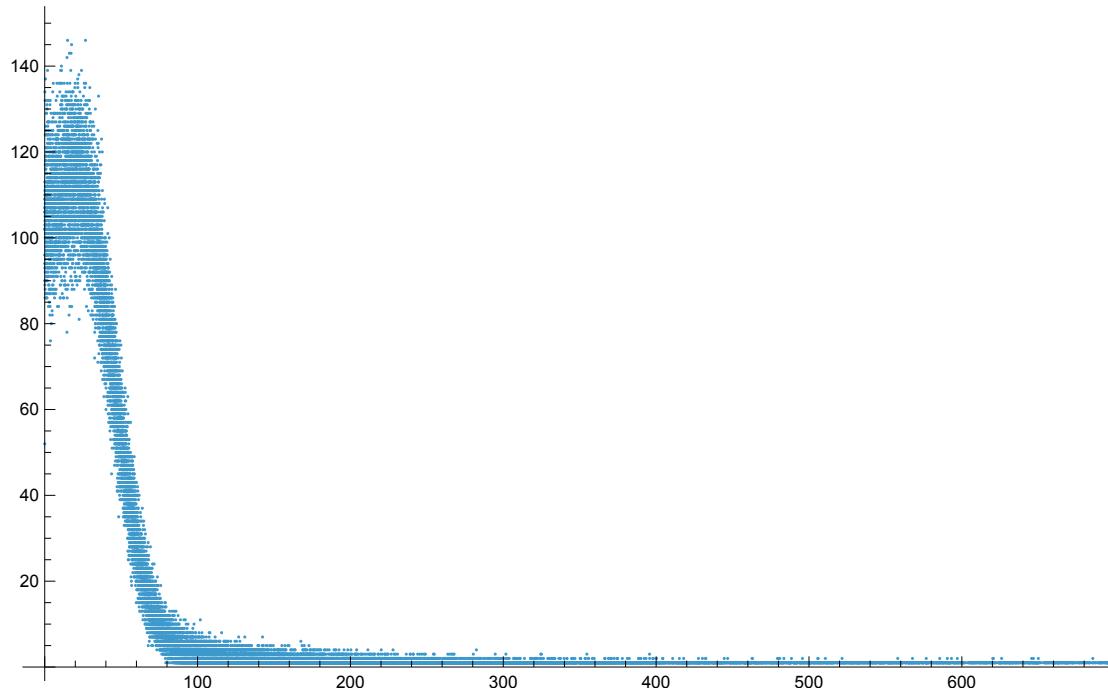


**gender**

```
In[6]:= BarChart[Counts[original[[All, "gender"]]], ChartLabels -> Automatic]  
Out[6]=
```

**amount**

```
In[7]:= ListPlot[Counts[original[[All, "amount"]]], ImageSize -> Large]  
Out[7]=
```

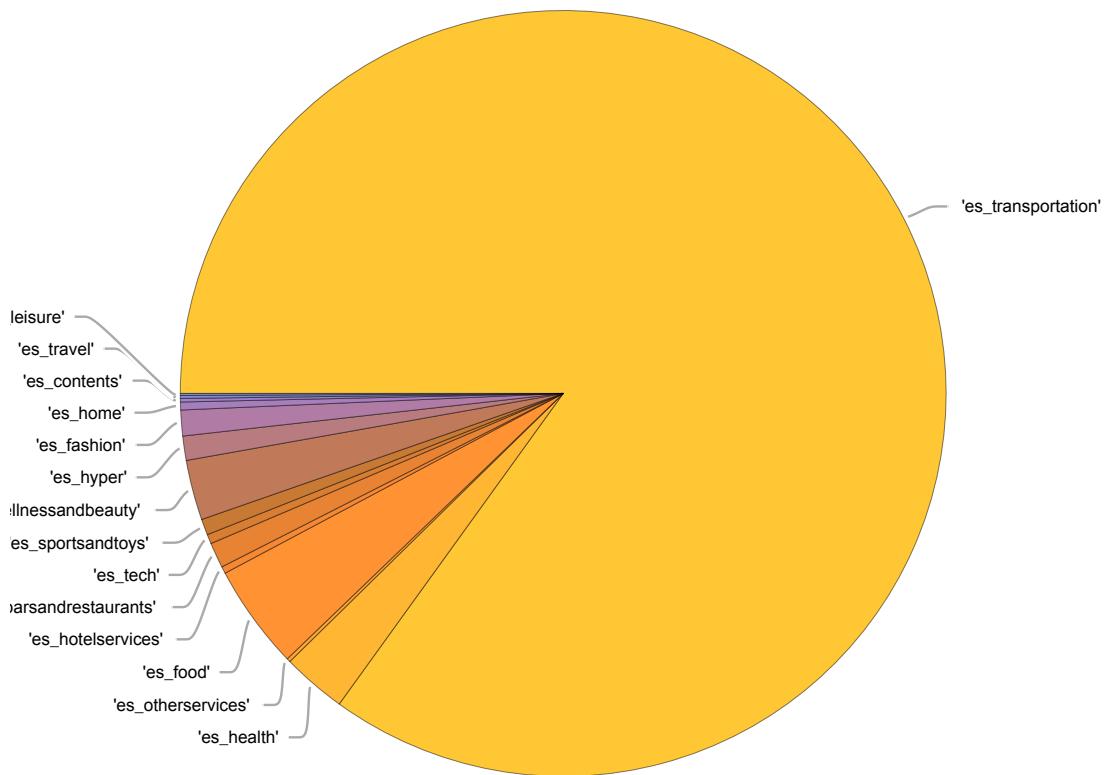


## category

```
In[®]:= ReverseSort[Dataset[Counts[original[[All, "category"]]]]]  
Out[®]=
```

'es_transportation'	505119
'es_food'	26254
'es_health'	16133
'es_wellnessandbeauty'	15086
'es_fashion'	6454
'es_barsandrestaurants'	6373
'es_hyper'	6098
'es_sportsandtoys'	4002
'es_tech'	2370
'es_home'	1986
'es_hotservices'	1744
'es_otherservices'	912
'es_contents'	885
'es_travel'	728
'es_leisure'	499

```
In[8]:= PieChart[Counts[original[[All, "category"]]],
ChartLabels → Callout[Automatic], ImageSize → Large]
Out[8]=
```



## fraud

- There are way more “no” fraud than “yes” fraud:

```
In[9]:= Counts[original[[All, "fraud"]]]
```

```
Out[9]= <| 0 → 587443, 1 → 7200 |>
```

```
Out[10]//TableForm=
```

	count	percentage
no fraud	587443	98.7892%
fraud	7200	1.21081%

## upload\_dataset.py

```
In[11]:= SystemOpen["upload_dataset.py"]
```

Este script tiene como objetivo principal **preparar y cargar datos de transacciones financieras en**

## una base de datos MongoDB.

Realiza los siguientes pasos clave:

**1. Carga de Datos:** Lee un archivo CSV llamado `fraud\_credit\_card.csv` en un DataFrame de Pandas.

### 2. Preprocesamiento y Limpieza de Datos:

- Convierte el campo `amount` (monto) a formato numérico (reemplazando comas por puntos).
- Discretiza\*\* el `amount` en categorías como 'very low', 'low', 'medium' y 'high'.
- Estandariza los campos `age`, `gender` y `category` (remueve comillas y maneja valores 'U' o desconocidos).
- Convierte la columna `fraud` (que es 0 o 1) a etiquetas 'yes' o 'no'.

**3. Selección de Características:** Elimina columnas que se consideran no relevantes para el análisis posterior (como `step`, `customer`, `zipcodeOri`, `merchant`, `zipMerchant`).

### 4. Carga a MongoDB:

- Establece una conexión a una base de datos MongoDB Atlas (usando credenciales de variables de entorno).
- Limpia (borra) la colección `transactions` en la base de datos `fraud\_db` .
- Inserta los datos preprocesados del DataFrame en la colección `transactions` de MongoDB, realizando la carga en \*\*lotes (batches)\*\* para optimizar el rendimiento y mostrando una barra de progreso.

## index\_dataset.py

```
SystemOpen["index_dataset.py"]
```

Este script tiene como objetivo principal **preparar, indexar y optimizar un dataset de transacciones almacenado en MongoDB** para su uso eficiente en un clasificador bayesiano.

Sus funciones principales son:

**1. Calcular y Almacenar Cardinalidades:** Identifica todos los valores únicos para las variables clave (`age`, `gender`, `category`, `amount\_bin`, `fraud`) y les asigna un índice numérico.

**1.1.** Estas correspondencias (mapeos) se guardan en la colección `cardinalities` .

**2. Indexar el Dataset:** Crea una nueva colección (`transactions\_indexed`) donde los valores categóricos originales de las transacciones se reemplazan por sus índices numéricos correspondientes, facilitando los cálculos probabilísticos.

**3. Precalcular y Almacenar Conteos:** Calcula y guarda en la colección `precomputed` las frecuencias de aparición de valores individuales y ciertas combinaciones de valores (especialmente con la variable `fraud` ).

**3.1.** Esto actúa como una caché para acelerar significativamente las consultas de probabilidad realizadas por el clasificador.

En resumen, es una herramienta de **preparación y optimización de datos** que transforma los datos brutos en un formato más adecuado para el análisis probabilístico y la inferencia rápida, reduciendo la carga de consultas en la base de datos para el clasificador.

## MongoDB

A continuación se describirán los *collections* disponibles en la base de datos:

### transactions

La data del archivo **fraud\_credit\_card.xlsx**:

---

```
self.client["fraud_db"]["transactions"]
```

---

The screenshot shows the MongoDB Compass interface with the following details:

- Database Path:** ClusterPUCP > fraud\_db > transactions
- Document Count:** 594.6K
- Operations:** Aggregations, Schema, Indexes (1), Validation
- Search Bar:** Type a query: { field: 'value' } or [Generate query](#)
- Buttons:** Explain, Reset, Find, Options
- Result Panel:** Displays 25 documents from 1 - 25 of 594643. Each document card shows fields: \_id, age, gender, category, amount\_bin, and fraud.

```

_id: ObjectId('6854e243e35a50901882de2f')
age : "5"
gender : "F"
category : "es_transportation"
amount_bin : "very_low"
fraud : "no"

_id: ObjectId('6854e243e35a50901882de32')
age : "4"
gender : "F"
category : "es_health"
amount_bin : "high"
fraud : "no"

_id: ObjectId('6854e243e35a50901882de37')
age : "3"
gender : "F"
category : "es_transportation"
amount_bin : "low"
fraud : "no"

_id: ObjectId('6854e243e35a50901882de57')
age : "2"
gender : "M"

```

### cardinalities

Guarda las cardinalidades de los parámetros:

---

```
self.client["fraud_db"]["cardinalities"]
```

---

```

_id: ObjectId('6857582f5b0e094b81cdd54b')
variable : "age"
mapping : Object

_id: ObjectId('6857582f5b0e094b81cdd54c')
variable : "gender"
mapping : Object
M : 0
F : 1
E : 2
U : 3

_id: ObjectId('6857582f5b0e094b81cdd54d')
variable : "category"
mapping : Object

_id: ObjectId('6857582f5b0e094b81cdd54e')
variable : "amount_bin"
mapping : Object

_id: ObjectId('6857582f5b0e094b81cdd54f')
variable : "fraud"
mapping : Object

```

## transactions\_indexed

Versión del collection *transactions*, pero indexado:

---

```
self.client["fraud_db"]["transactions_indexed"]
```

---

```

_id: ObjectId('6854e243e35a50901882de19')
age : 3
gender : 1
category : 0
amount_bin : 1
fraud : 0

_id: ObjectId('6854e243e35a50901882de1f')
age : 3
gender : 1
category : 0
amount_bin : 1
fraud : 0

_id: ObjectId('6854e243e35a50901882de49')
age : 5
gender : 0
category : 0
amount_bin : 1
fraud : 0

_id: ObjectId('6854e243e35a50901882de55')
age : 1
gender : 1

```

## precomputed

Conteos precomputados para optimizar la clasificación:

---

```
self.client["fraud_db"]["precomputed"]
```

---

ClusterPUCP > fraud\_db > precomputed

Documents 101 Aggregations Schema Indexes 1 Validation

Type a query: { field: 'value' } or [Generate query](#)

[ADD DATA](#) [EXPORT DATA](#) [UPDATE](#) [DELETE](#)

25 1 - 25 of 101

```
_id: ObjectId('6857598c5b0e094b81cdd550')
fraud: 0
count: 587443
```

```
_id: ObjectId('6857598d5b0e094b81cdd551')
fraud: 1
count: 7200
```

```
_id: ObjectId('6857598d5b0e094b81cdd552')
age: 0
count: 109025
```

```
_id: ObjectId('6857598d5b0e094b81cdd553')
age: 0
fraud: 0
count: 107615
```

```
_id: ObjectId('6857598e5b0e094b81cdd554')
age: 0
fraud: 1
count: 1410
```

## transacciones\_sampled\_10

Un sample del 10% de la colección “transacciones\_indexed”:

---

```
self.client["fraud_db"]["transacciones_sampled_10"]
```

---

ClusterPUCP > fraud\_db > transactions\_sampled\_10

Documents 59.5K Aggregations Schema Indexes 1 Validation

Type a query: { field: 'value' } or [Generate query](#)

[ADD DATA](#) [EXPORT DATA](#) [UPDATE](#) [DELETE](#)

25 1 - 25 of 59464

```
_id: ObjectId('6854e2e9e35a50901889bd59')
age: 2
gender: 0
category: 0
amount_bin: 1
fraud: 0
```

```
_id: ObjectId('6854e2f2e35a5090188a4c17')
age: 2
gender: 0
category: 3
amount_bin: 1
fraud: 0
```

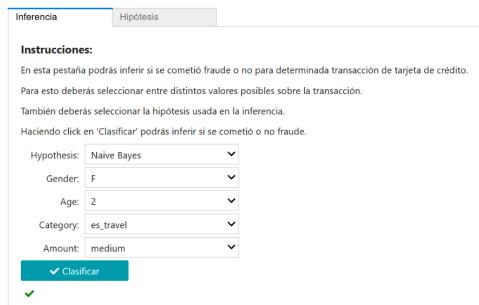
```
_id: ObjectId('6854e2cae35a5090188870da')
age: 1
gender: 0
category: 0
amount_bin: 1
fraud: 0
```

```
_id: ObjectId('6854e280e35a509018857540')
age: 2
gender: 1
```

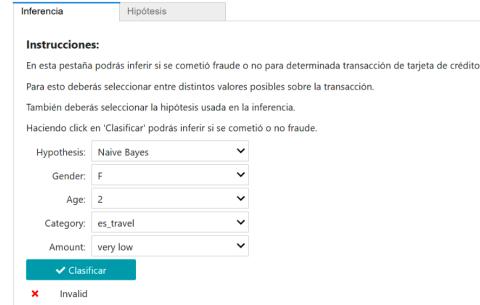
## Interfaz del clasificador

```
SystemOpen["interface.ipynb"]
```

## Pestaña de Inferencia



(a) Visualización de hipótesis 1



(b) Visualización de hipótesis 2

Figura 1: Comparación de estructuras de hipótesis

## Pestaña de Hipótesis

El grafo causal se construye dinámicamente usando **networkx**, permitiendo al usuario interpretar las relaciones causa-efecto establecidas.

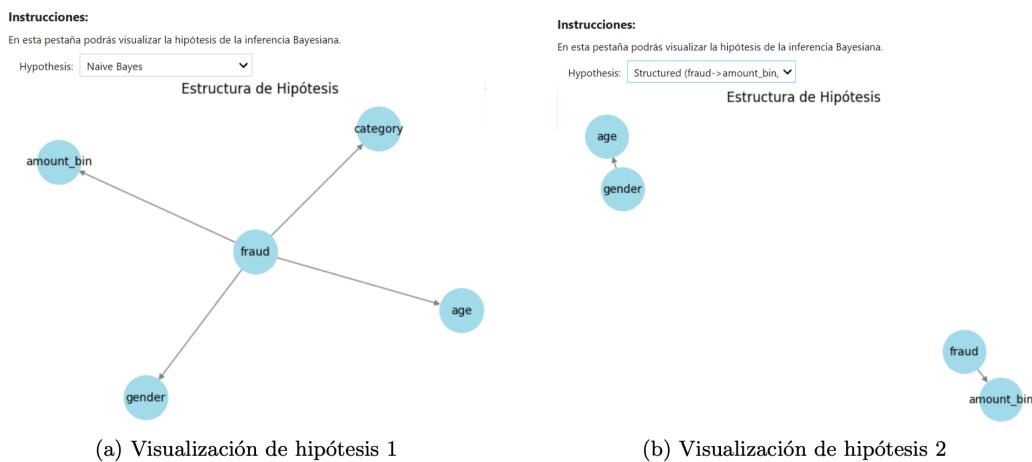


Figura 2: Comparación de estructuras de hipótesis

## Benchmarks de Optimización

SystemOpen["benchmark\_classifier.py"]

### Optimizaciones

**1. Ninguna:** Se refiere a realizar consultas directas a la base de datos para cada conteo necesario, sin optimizaciones específicas a nivel de la consulta o caché. Aunque la data base ha sido preprocesada (ej. binning), cada **count\_documents** aún puede ser costoso.

- 1.1.** En este escenario base, cada vez que el clasificador necesita obtener un conteo de documentos para una combinación de valores (por ejemplo, `count_documents({'gender': 'M', 'age': '2'})`), realiza una consulta completa a la colección en la base de datos.
- 1.2.** De cierta manera, la data ya ha sido optimizada ya que se realizó bins de “amount”, se convirtió los posibles valores de los parámetros a índices, etc.

**2. Indexes:** Son estructuras de datos en la base de datos que permiten búsquedas y recuperación de documentos mucho más rápidas en campos específicos. Mejoran el rendimiento al permitir que la base de datos salte directamente a los datos relevantes en lugar de escanear toda la colección, reduciendo el tiempo de I/O y procesamiento.

---

```
self.data_collection.create_index([(parent, 1), (var, 1)])
```

**3. Precomputed database:** Consiste en almacenar conteos o agregaciones frecuentes en una colección separada (**precomputed**) dentro de la base de datos. Mejora el rendimiento al evitar cálculos repetitivos en vivo; en su lugar, el sistema realiza una búsqueda rápida en esta tabla de resultados ya calculados.

---

```
res = self.precomputed.find_one(evidence, {"count": 1})
if res is not None:
    count = res["count"]
else:
    count = self.data_collection.count_documents(evidence)
return count
```

**4. LRU cache:** Es un caché en **memoria RAM a nivel de aplicación** que guarda los resultados de las llamadas a funciones. La primera vez que se ejecuta una consulta de conteo, el resultado se almacena; las veces subsiguientes con los mismos argumentos, el resultado se devuelve instantáneamente desde la RAM, eliminando la necesidad de consultar la base de datos y ofreciendo la mayor ganancia de velocidad.

---

```
@lru_cache(maxsize=10000) # You can adjust maxsize based on expected unique
queries
def _cached_compute_counts(self, evidence_tuple):
    evidence = dict(evidence_tuple) # Convert tuple back to dict
    ...
    return count

def compute_counts(self, evidence):
    # Convert the dictionary (which is not hashable) to a sorted tuple of (key,
    value) pairs
    # so it can be used as a cache key.
    hashable_evidence = tuple(sorted(evidence.items()))
    return self._cached_compute_counts(hashable_evidence)
```

---

## Análisis

### code

### Import benchmarks

- **benchmark0:** No optimizations

```
In[6]:= benchmark0 = importOptimizationBenchmark[
  "./benchmarks/benchmark_results-no_index_no_precompute_no_cache.csv"]
```

Out[6]=

	test_id	repeat	category	gender	age	amount_bin	fraud_pred
1	0	0	es_fashion	M	0	medium	
2	0	1	es_fashion	M	0	medium	
3	0	2	es_fashion	M	0	medium	
4	0	3	es_fashion	M	0	medium	
5	0	4	es_fashion	M	0	medium	
6	0	5	es_fashion	M	0	medium	

■ **benchmark1:** Just mongodb index optimization

```
In[7]:= benchmark1 = importOptimizationBenchmark[
  "./benchmarks/benchmark_results-no_precompute_no_cache.csv"]
```

Out[7]=

	test_id	repeat	category	gender	age	amount_bin	fraud_pred
1	0	0	es_fashion	M	0	medium	
2	0	1	es_fashion	M	0	medium	
3	0	2	es_fashion	M	0	medium	
4	0	3	es_fashion	M	0	medium	
5	0	4	es_fashion	M	0	medium	
6	0	5	es_fashion	M	0	medium	

■ **benchmark2:** mongodb index optimization + precomputed counts

```
In[6]:= benchmark2 =
importOptimizationBenchmark["./benchmarks/benchmark_results-no_cache.csv"]
```

Out[6]=

	test_id	repeat	category	gender	age	amount_bin	fraud_pred
1	0	0	es_fashion	M	0	medium	
2	0	1	es_fashion	M	0	medium	
3	0	2	es_fashion	M	0	medium	
4	0	3	es_fashion	M	0	medium	
5	0	4	es_fashion	M	0	medium	
6	0	5	es_fashion	M	0	medium	

#### ■ **benchmark3:** mongodb index optimization + precomputed counts + LRU cache

```
In[7]:= benchmark3 = importOptimizationBenchmark[
  "./benchmarks/benchmark_results-indexes_precompute_cache.csv"]
```

Out[7]=

	test_id	repeat	category	gender	age	amount_bin	fraud_pred
1	0	0	es_fashion	M	0	medium	
2	0	1	es_fashion	M	0	medium	
3	0	2	es_fashion	M	0	medium	
4	0	3	es_fashion	M	0	medium	
5	0	4	es_fashion	M	0	medium	
6	0	5	es_fashion	M	0	medium	

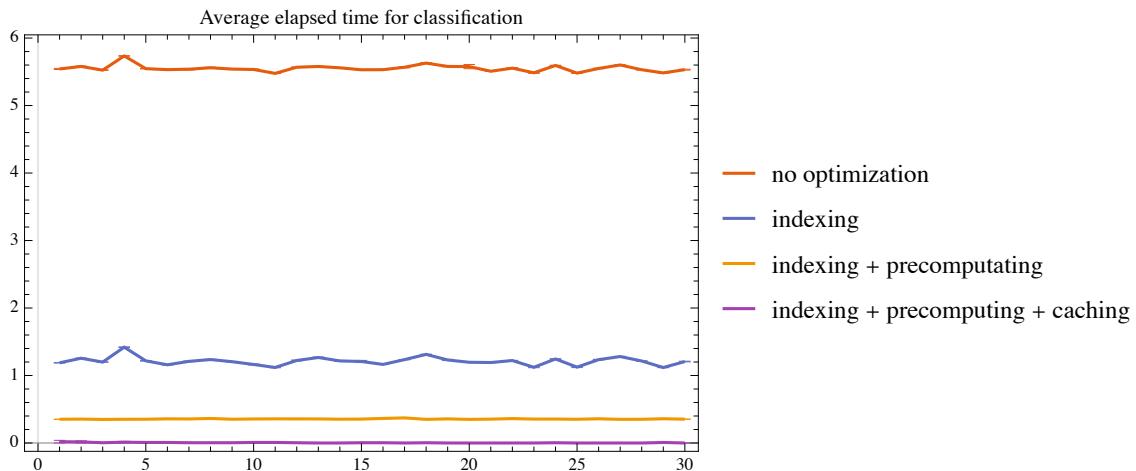
## Statistics Summary

Out[8]=

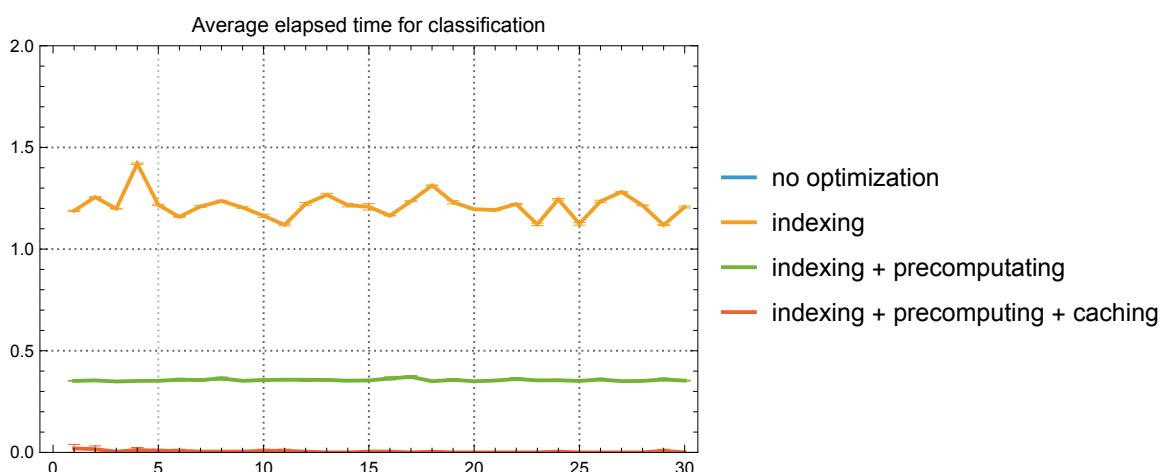
name	Count	NonNumeric	Mean	StandardDeviation	Min
benchmark 0	300	0	5.55147 s	0.0551509 s	5.45389 s
benchmark 1	300	0	1.21282 s	0.0624999 s	1.10361 s
benchmark 2	300	0	0.355424 s	0.0140068 s	0.342133 s
benchmark 3	300	0	0.00411283 s	0.0199202 s	0.0000157350 s

< columns 1-10 of 11 >

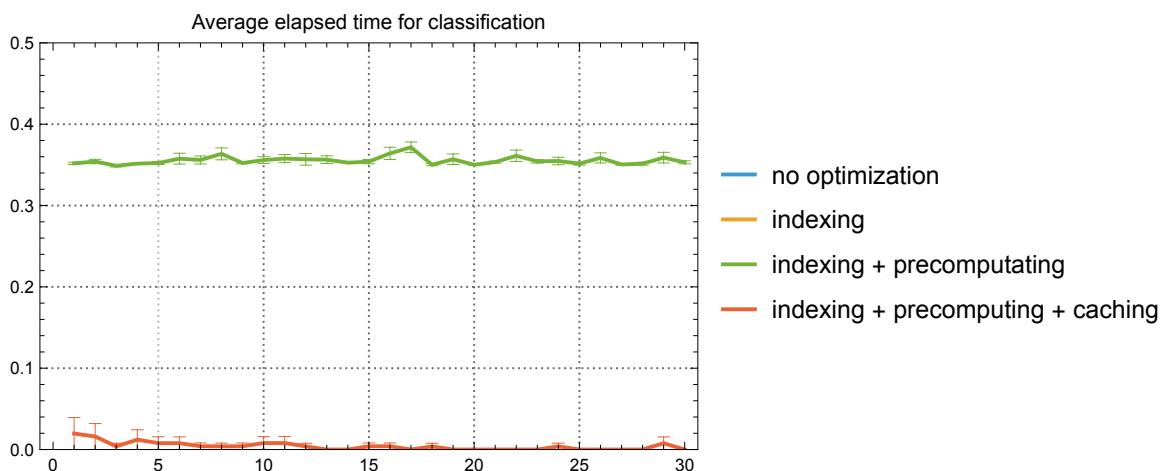
Out[•] =



Out[•] =



Out[•] =



- Caching es realmente rápido:

In[•]:= `UnitConvert[Mean@benchmark3[[All, "elapsed"]], "Milliseconds"]`

Out[•] =

(4.1 ± 1.2) ms

## Benchmarks de Hipótesis

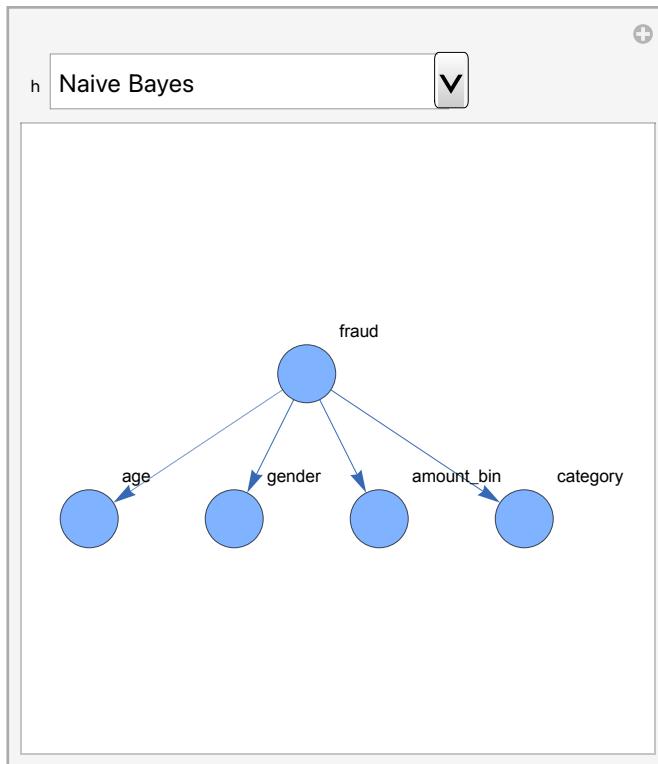
```
SystemOpen["full_benchmark_classifier.py"]
```

### Hipótesis

Hipótesis determinadas a mano (leer comentarios en la sección de K2):

```
In[8]:= hypo = Map[Graph[Flatten[Thread /@ Normal[#]],  
VertexLabels → Automatic, VertexSize → Large, ImageSize → {300, 300}] &,  
Import["./available_hypothesis.json", "RawJSON"]];
```

Out[8]=



### Análisis

```
In[9]:= hypothesisBenchmark =  
Import["./benchmarks/full_benchmark_results.csv", "Tabular"];  
hypothesisBenchmark = TransformColumns[hypothesisBenchmark, "avg_time_sec" →  
Function[ColumnwiseThread @ Quantity[#"avg_time_sec", "Seconds"]]];  
hypothesisBenchmark =  
GroupBy[hypothesisBenchmark, #"hypothesis" &, KeyDrop["hypothesis"]];
```

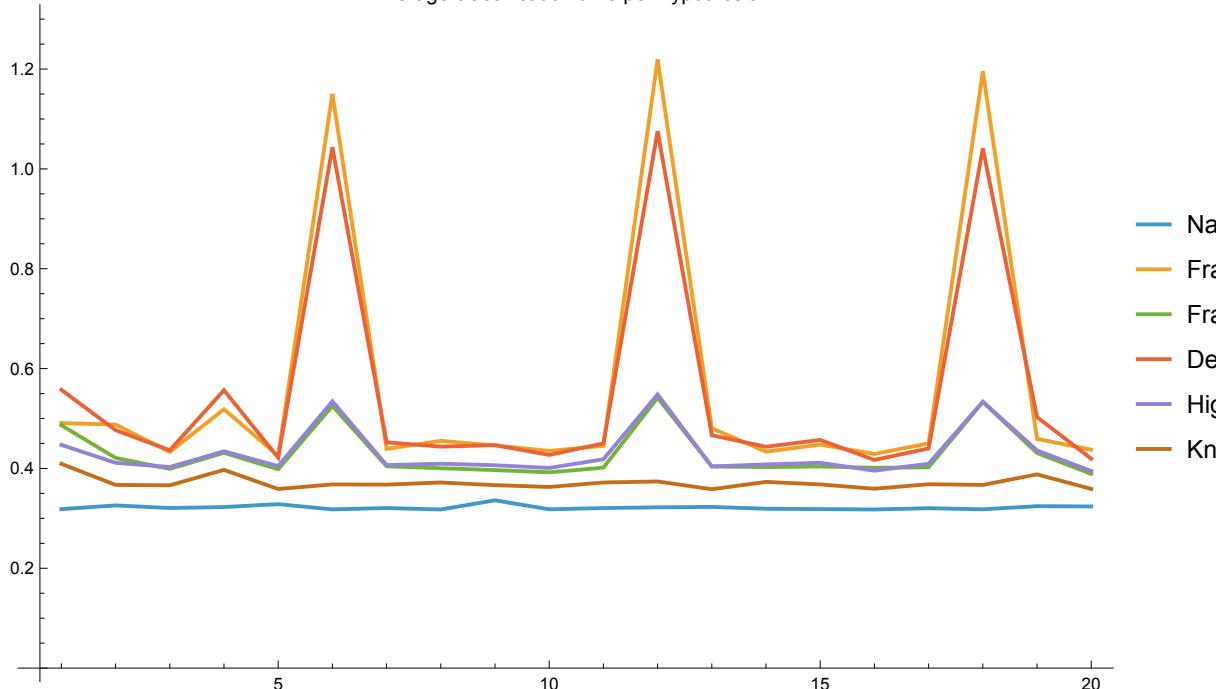
Out[6] =

Naive Bayes

	evidence_id	avg_time_sec (s)	fraud_prediction	probability	age	gender
1	1	0.319	False	0.0018	3	M
2	2	0.326	False	0.0008	4	F
3	3	0.321	False	0.0000	4	M
4	4	0.323	False	0.0029	4	F
5	5	0.328	True	0.0000	5	F
6	6	0.318	False	0.0066	5	F

Out[7] =

Average classification time per hypothesis



## Benchmarks de Tamaño de datos

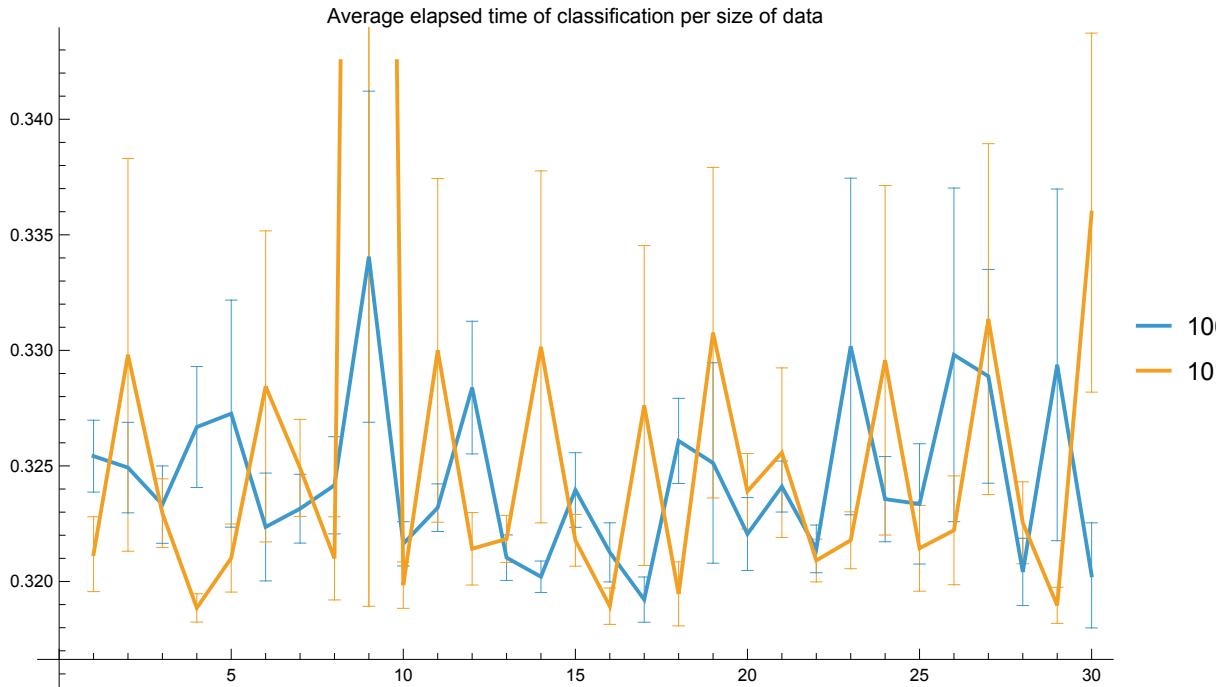
```
In[8]:= fractions =
GroupBy[Import["./benchmarks/benchmark_results_fractions.csv", "Tabular"],
#"transactions_db_name" &, KeyDrop["transactions_db_name"]];
```

Out[=]

f transactions \_indexed transactions \_sampled \_10

	test_id	repeat	category	gender	age	amount_bin
1	0	0	es_fashion	M	0	medium
2	0	1	es_fashion	M	0	medium
3	0	2	es_fashion	M	0	medium
4	0	3	es_fashion	M	0	medium
5	0	4	es_fashion	M	0	medium
6	0	5	es_fashion	M	0	medium

Out[=]



## Métricas de Clasificación

Matriz de confusión

- **Verdadero Positivo (VP):** El modelo predijo que era fraude, y realmente era fraude.
- **Verdadero Negativo (VN):** El modelo predijo que NO era fraude, y realmente NO era fraude.
- **Falso Positivo (FP):** El modelo predijo que era fraude, pero en realidad NO era fraude.
- **Falso Negativo (FN):** El modelo predijo que NO era fraude, pero en realidad SÍ era fraude.

Métricas:

■ **Accuracy:**  $(VP + VN) / (VP + VN + FP + FN)$

- Es la proporción de predicciones correctas (tanto fraudes como no-fraudes) sobre el total de predicciones.

■ **Precision:**  $VP / (VP + FP)$

- De todas las veces que el modelo predijo “fraude”, ¿cuántas veces realmente fue fraude?

■ **Recall (Exhaustividad o Sensibilidad):**  $VP / (VP + FN)$

- De todos los casos que realmente eran fraude, ¿cuántos logró detectar el modelo?

■ **F1-Score:**  $2 \times (\text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall})$

- Es un promedio “armónico” de la Precisión y el Recall. Proporciona un equilibrio entre ambas métricas.

En resumen, para la detección de fraude:

- Recall es a menudo lo más importante porque no quieres que los fraudes se te escapen.
- Precision también es importante para evitar la “fatiga de alertas” o molestar a clientes legítimos.
- F1-Score busca un buen compromiso entre ambos.
- Accuracy puede ser engañosa y debe usarse con precaución, especialmente si el número de fraudes es muy pequeño.

```
In[6]:= metrics = Dataset[Association@Map[#model → KeyDrop[#, "model"] &,
Import["./classification_metrics.json", "RawJSON"]]]
```

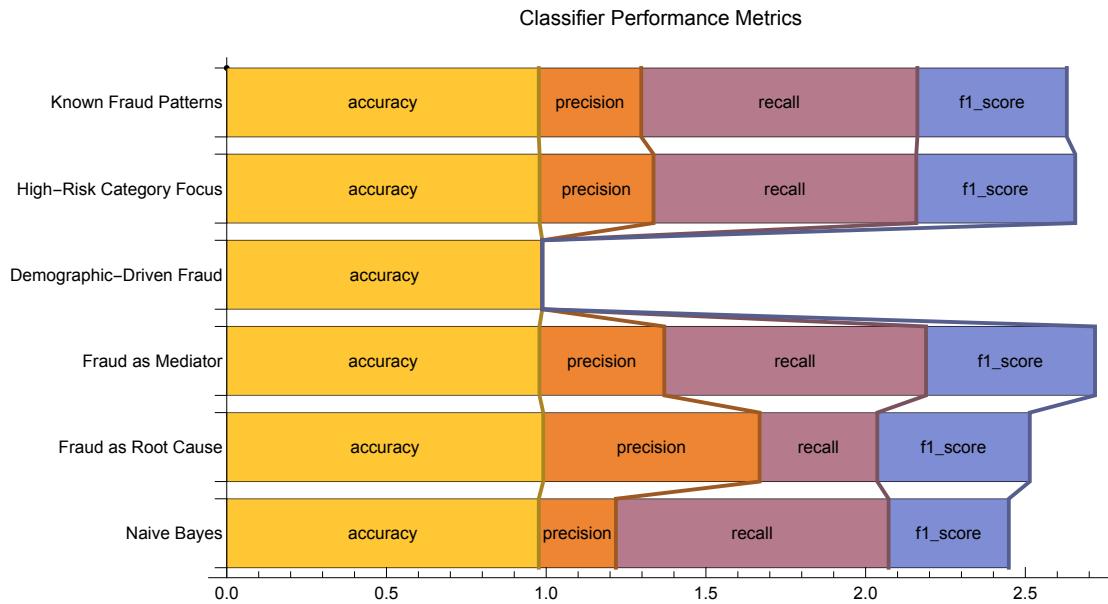
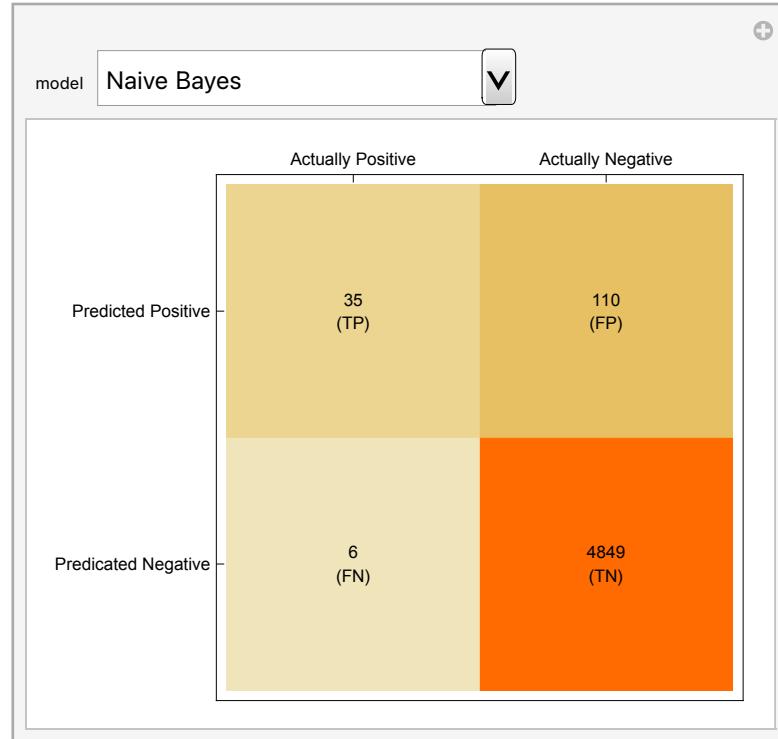
Out[6]=

	dataset_info	
	total_documents_in_db	evaluated_samples
Naive Bayes	594 643	5000
Fraud as Root Cause	594 643	5000
Fraud as Mediator	594 643	5000
Demographic–Driven Fraud	594 643	5000
High–Risk Category Focus	594 643	5000
Known Fraud Patterns	594 643	5000

```
In[7]:= metrics1 = Map[#"metrics" &, metrics]
```

Out[7]=

	accuracy	precision	recall
Naive Bayes	0.9768	0.2414	0.8537
Fraud as Root Cause	0.9908	0.6774	0.3684
Fraud as Mediator	0.979	0.3907	0.8194
Demographic–Driven Fraud	0.9878	0.0	0.0
High–Risk Category Focus	0.9794	0.3566	0.8226
Known Fraud Patterns	0.9768	0.3208	0.8644

Out[<sup>6</sup>] =Out[<sup>6</sup>] =

## Algoritmo K2 (learn\_k2\_structures.py)

```
SystemOpen["learn_k2_structures.py"]
```

El script tiene como propósito descubrir automáticamente cómo se relacionan las diferentes variables en el dataset de transacciones para construir un modelo de red bayesiana.

- Para lograrlo, inicializa un clasificador bayesiano, que ya tiene acceso a tus datos preprocesados.
- Luego, se utiliza el algoritmo **K2** para analizar tus datos y encontrar las conexiones más probables entre las variables

- por ejemplo, determinar si la edad o el género influyen en la categoría de una transacción o en la probabilidad de fraude.
- Este proceso se repite varias veces, probando con diferentes límites para la cantidad de “padres” que cada variable puede tener.
- Una vez que el algoritmo K2 descubre estas relaciones, las guarda en un formato que el clasificador puede utilizar, y las almacena en un archivo JSON.

En resumen, este script actúa como un “entrenador” para la red bayesiana, permitiéndole aprender la arquitectura óptima directamente de los datos en lugar de tener que definirla manualmente.

```
In[5]:= Import["./learned_hypotheses.json", "RawJSON"] // Dataset
```

```
Out[5]=
```

metadata	dataset_size	594 643
	variables	{age, gender, ...}
	cardinalities	<  age → 8, ge ...
	learning_timestamp	1 751 895 502.
hypotheses	Naive Bayes	<  fraud → {ag ...
	Structured (fraud→amount_bin, gender→age)	<  fraud → {an ...
	K2 learned (u=1)	<   >
	K2 learned (u=2)	<   >
	K2 learned (u=3)	<   >
	7 total >	
learning_details	K2 learned (u=1)	<  structure →
	K2 learned (u=2)	<  structure →
	K2 learned (u=3)	<  structure →
	K2 learned (u=4)	<  structure →
	K2 learned (u=5)	<  structure →

Es normal que el algoritmo K2 encuentre que algunas (o todas) las variables no tienen padres. No necesariamente significa que haya un error.

Posibles razones por la cual no se encuentre estructura:

- **No hay dependencias fuertes:** Si en los datos una variable realmente no depende mucho de otras, K2 lo detecta y no le asigna padres
- **Datos limitados:** Con pocos datos, es difícil ver relaciones claras, y el algoritmo prefiere simplicidad
  - Este no es el caso con *fraud\_credit\_card.xlsx*
- **Orden de las variables:** El orden en que le das las variables a K2 es crucial
  - Si un posible parente aparece después de su hijo, K2 nunca lo considerará
- **El parámetro alpha:** Este valor afecta qué tan “convencido” está el algoritmo de que hay una relación
  - Un alpha alto puede hacer que el modelo prefiera no asignar padres

- Se probaron  $\alpha=0.01, \alpha=0.1, \alpha=1.0, \alpha=10.0, \alpha=100.0$  sin resultados
- **Datos dispersos:** Si las variables tienen muchos valores posibles y pocos ejemplos para cada combinación, K2 puede tener dificultades para encontrar patrones