Sistema de Inferencia Bayesiana para Detección de Fraude en Transacciones

Javier Monzón (20121248)
Daniel Sánchez (20130495)

21 de junio de 2025

Resumen

Este informe documenta el desarrollo de una aplicación interactiva para la detección de fraude en transacciones con tarjetas de crédito, basada en un clasificador bayesiano. El sistema permite la selección de hipótesis causales, la inferencia probabilística basada en evidencia categórica, y la visualización de grafos de dependencia entre variables.

Índice

1. Introduccion

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.

2. Conceptos aplicados

2.1. Estimación de Distribución Condicional

Para una variable X con valor x, dada una condición (contexto) $\text{Pa}(X) = u$, se estima la probabilidad condicional:

$$P(X = x \mid \text{Pa}(X) = u) = \frac{\text{M}(X = x, \text{Pa}(X) = u) + \alpha}{\text{M}(\text{Pa}(X) = u) + \alpha \cdot k}$$

donde:

- $\text{Pa}(X)$: conjunto de variables padres de X según la hipótesis estructural,
- k: cardinalidad (número de valores posibles) de la variable X,
- α: hiperparámetro de suavizado,

2.2. Inferencia Bayesiana por Probabilidad Conjunta

La inferencia de la clase se basa en la maximización de la probabilidad conjunta:

$$\hat{C} = \arg\max_{c} P(C = c) \cdot \prod_{i=1}^{n} P(X_i = x_i \mid \text{Pa}(X_i) = u_i)$$

donde cada X_i es una variable categórica observada, y $\text{Pa}(X_i)$ es el conjunto de sus variables padres.

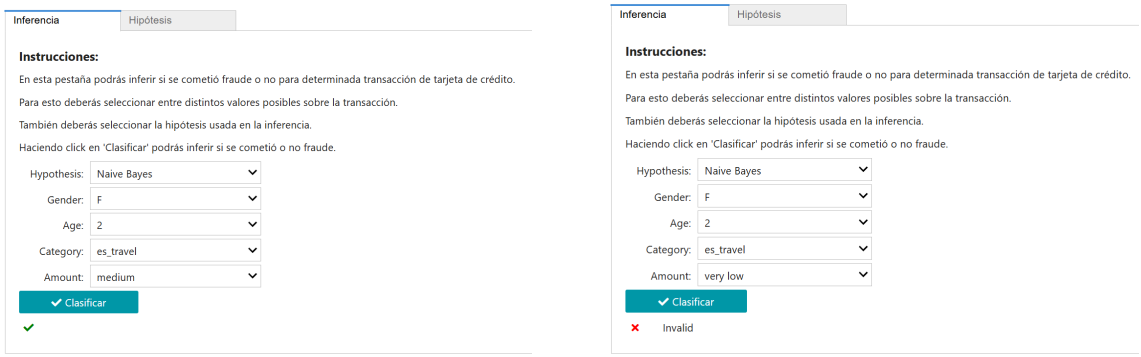
3. Librerías utilizadas

Este proyecto utiliza las siguientes bibliotecas de Python:

- `pandas`: Preprocesar el dataset original en formato CSV.
- `pymongo`: Cliente oficial de MongoDB para Python.
- `python-dotenv`: Cargar variables de entorno en archivo `.env` (cadena de conexión a MongoDB).
- `ipywidgets`: Permite construir interfaces gráficas interactivas dentro de notebooks.
- `networkx`: Permite trabajar con grafos para modelar la estructura de las hipótesis bayesianas.
- `matplotlib`: Librería de visualización para mostrar resultados gráficos en la interfaz.

4. Interfaz del clasificador

4.1. Pestaña Inferencia

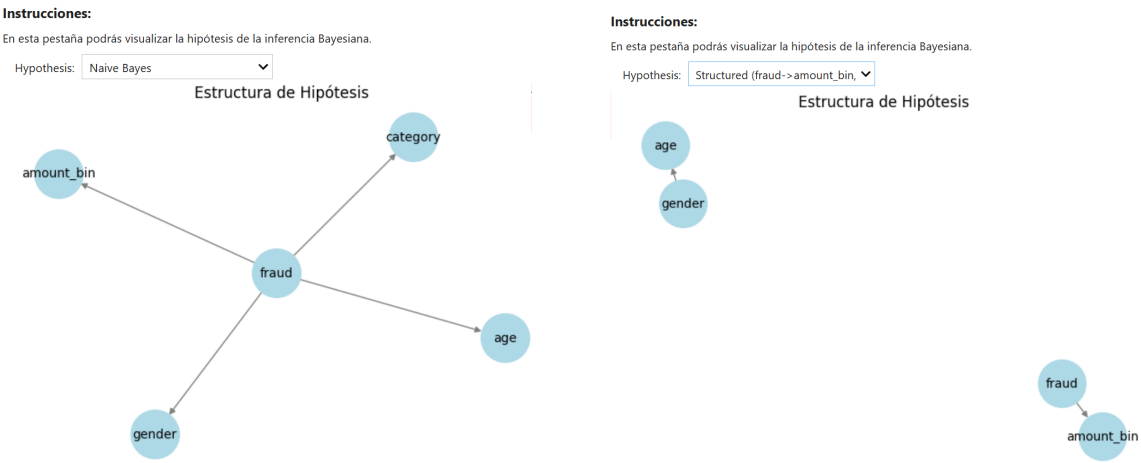


(a) Visualización de hipótesis 1 (b) Visualización de hipótesis 2

Figura 1: Comparación de estructuras de hipótesis

4.2. Pestaña Hipótesis

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



(a) Visualización de hipótesis 1 (b) Visualización de hipótesis 2

Figura 2: Comparación de estructuras de hipótesis

5. Experimentación

5.1. Optimizacion

Explicacion del cambio: Se precalculan los conteos que se hacian en base de datos.

```
def precompute_and_store(cardinalities):
    # precomputing Naive Bayes counts
    target_variable = "fraud"
    target_values = list(cardinalities[target_variable].values())
    for var, cardinality in cardinalities.items():
        for val in cardinality.values():
```

Figura 3: Entorno de ejecución

```
            for target_val in target_values:
                count = indexed.count_documents({var: val, target_variable:
    target_val})
                data = {var: val, target_variable: target_val, "count": count}
                precomputed.insert_one(data)
```

Listing 1: Precomputación de conteos para Naive Bayes

5.2. Evaluación y Resultados

Se discute el tiempo de respuesta del sistema.

Cuadro 1: Tiempos de Inferencia en milisegundos (ms)

Workflow 1: Indexación	**Workflow 2: Pre-calculo de resultados**
2952.08	1365.81
2924.33	1365.28
3004.51	1407.69
2953.00	1389.60
2938.52	1401.49
2929.27	1397.70
3086.87	1392.79
3057.46	1394.48
3055.50	1399.89
3057.30	1408.84
Promedio: 2995.88	**Promedio: 1393.35**

5.3. Entorno de ejecución

Se ejecuto los calculos en el siguiente ordenador:

6. Conclusiones

El sistema demuestra cómo un enfoque bayesiano, combinado con una interfaz intuitiva y un backend eficiente, puede asistir en tareas de detección de fraude de manera interactiva y explicativa.

Asimismo, tener resultados precalculados ayuda a optimizar el tiempo de inferencia.