



UniversidadeVigo

Trabajo Fin de Máster

Análisis, Optimización y Simulación de Procesos de Negocio mediante Técnicas de Process Mining y Modelos Predictivos Avanzados

Pedro Tarrío Mallou

Máster en Técnicas Estadísticas

Curso 2025-2026

Propuesta de Trabajo Fin de Máster

Título en galego: Análise, Optimización e Simulación de Procesos de Negocio mediante Técnicas de Process Mining e Modelos Predictivos Avanzados
Título en español: Análisis, Optimización y Simulación de Procesos de Negocio mediante Técnicas de Process Mining y Modelos Predictivos Avanzados
English title: Analysis, Optimization, and Simulation of Business Processes using Process Mining Techniques and Advanced Predictive Models
Modalidad: Modalidad B
Autor: Pedro Tarrío Mallou, Universidade de Santiago de Compostela
Directora: María José Ginzo Villamayor, Universidade de Santiago de Compostela
Tutor: Javier Soria Esponera, SDG Group
Breve resumen del trabajo: Este proyecto aplica técnicas de Process Mining sobre logs de eventos para analizar, modelar y optimizar los procesos de negocio. Su objetivo es doble: extraer conocimiento accionable sobre rendimiento y conformidad actual, y desarrollar modelos predictivos avanzados (Transformers) para anticipar el comportamiento futuro.
Recomendaciones:
Otras observaciones:

Doña María José Ginzo Villamayor, Profesora Ayudante Doctora de la Universidade de Santiago de Compostela, y don Javier Soria Esponera, Specialist Lead de SDG Group, informan que el Trabajo Fin de Máster titulado

Análisis, Optimización y Simulación de Procesos de Negocio mediante Técnicas de Process Mining y Modelos Predictivos Avanzados

fue realizado bajo su dirección por don Pedro Tarrío Mallou para el Máster en Técnicas Estadísticas. Estimando que el trabajo está terminado, dan su conformidad para su presentación y defensa ante un tribunal. Además, doña María José Ginzo Villamayor y don Pedro Tarrío Mallou

☒ sí

☐ no

autorizan a la publicación de la memoria en el repositorio de acceso público asociado al Máster en Técnicas Estadísticas.

En Santiago de Compostela, a 17 de Julio de 2026.

La directora:

Doña María José Ginzo Villamayor

Firmado por GINZO VILLAMAYOR MARIA
JOSE - ***4013** el día 17/07/2026
con un certificado emitido por AC

El tutor:

Don Javier Soria Esponera



El autor:

Don Pedro Tarrío Mallou



Declaración responsable. Para dar cumplimiento a la Ley 3/2022, de 24 de febrero, de convivencia universitaria, referente al plagio en el Trabajo Fin de Máster (Artículo 11, Disposición 2978 del BOE núm. 48 de 2022), **el autor declara** que el Trabajo Fin de Máster presentado es un documento original en el que se han tenido en cuenta las siguientes consideraciones relativas al uso de material de apoyo desarrollado por otros/as autores/as:

- Todas las fuentes usadas para la elaboración de este trabajo han sido citadas convenientemente (libros, artículos, apuntes de profesorado, páginas web, programas, . . .)

- Cualquier contenido copiado o traducido textualmente se ha puesto entre comillas, citando su procedencia.
- Se ha hecho constar explícitamente cuando un capítulo, sección, demostración, . . . sea una adaptación casi literal de alguna fuente existente.

Y, acepta que, si se demostrara lo contrario, se le apliquen las medidas disciplinarias que correspondan.

Índice general

Resumen	XI
Introducción	XIII
1. Fundamentos de <i>Process Mining</i> clásico	1
1.1. Formalización matemática de un proceso	1
1.1.1. <i>Process Mining</i> como puente entre la minería de datos y el BPM	1
1.1.2. Universos, eventos y <i>data payload</i>	2
1.1.3. Trazas como secuencias ordenadas	3
1.1.4. El registro de actividades y el registro de eventos	4
1.1.5. Ejemplos de procesos reales y sus componentes	5
1.2. Redes de Petri como lenguaje de representación	7
1.3. Descubrimiento de procesos (<i>Process Discovery</i>)	9
1.3.1. Formulación del problema de descubrimiento	9
1.3.2. El <i>Directly-Follows Graph</i>	10
1.3.3. El α - <i>Miner</i> : algoritmo fundacional	11
1.3.4. El <i>Heuristic Miner</i> : robustez frente al ruido	12
1.3.5. El <i>Inductive Miner</i> : garantías estructurales	13
1.3.6. Alcance de los algoritmos de descubrimiento	13
1.4. Verificación de conformidad y métricas de calidad	13
1.4.1. Motivación del problema	13
1.4.2. Las cuatro dimensiones de calidad de un modelo	14
1.4.3. <i>Token Replay</i>	15
1.4.4. <i>Alignments</i> : alineamientos óptimos	16
1.5. Limitaciones del enfoque clásico	17

1.5.1.	Explosión combinatoria y modelos ininterpretables	18
1.5.2.	Exclusión del <i>data payload</i>	18
1.5.3.	Naturaleza retrospectiva y ausencia de capacidad predictiva	18
1.5.4.	Puente hacia el aprendizaje profundo	19
2.	Aprendizaje profundo aplicado a <i>Process Mining</i>	21
2.1.	Justificación del aprendizaje profundo en procesos secuenciales	21
2.2.	Componentes matemáticos y transformaciones base	22
2.2.1.	<i>Embeddings</i> categóricos	22
2.2.2.	Funciones de activación y la ruptura de la linealidad	23
2.2.3.	<i>Layer Normalization</i> y <i>Dropout</i> : estabilización y regularización	25
2.3.	Aprendizaje profundo sobre grafos (GNNs)	27
2.3.1.	Motivación: datos con estructura relacional	27
2.3.2.	El paradigma de <i>Message Passing</i>	28
2.3.3.	Graph Attention Networks (GAT)	29
2.3.4.	GATv2: atención dinámica	30
2.3.5.	Inyección de atributos de arista	31
2.3.6.	Puente computacional: PyTorch y PyTorch Geometric	32
2.4.	Arquitecturas basadas en atención y Transformers	33
2.4.1.	Del lenguaje natural a los procesos	33
2.4.2.	Autoatención escalada por producto escalar	34
2.4.3.	<i>Multi-Head Attention</i>	35
2.4.4.	El bloque Transformer Encoder	36
2.4.5.	Codificación posicional	36
2.4.6.	Adaptación a <i>Process Mining</i> : prefijos y <i>pooling</i> indexado	37
2.5.	<i>Autoencoders</i> y errores de reconstrucción	38
2.5.1.	Definición formal del <i>Autoencoder</i>	39
2.5.2.	Funciones de pérdida para la reconstrucción	40
2.5.3.	Pérdida total multi-tarea	42
2.5.4.	Del error de reconstrucción a la detección de anomalías	42
3.	Caso de Uso: Proceso de Solicitud de Préstamos	45
3.1.	Planteamiento del problema	46
3.2.	Conocimiento de los datos	48

3.2.1.	Origen y Contexto del Conjunto de Datos	48
3.2.2.	Diccionario de Datos	49
3.2.3.	Formalización del registro de eventos	53
3.2.4.	Análisis Exploratorio de Datos (EDA)	55
3.3.	Análisis descriptivo mediante <i>Process Mining</i> clásico	61
3.3.1.	Exploración temporal mediante <i>Dotted Charts</i>	61
3.3.2.	Descubrimiento del proceso	62
3.4.	Detección de anomalías mediante Deep Learning	64
3.4.1.	Limitaciones de las técnicas clásicas	65
3.4.2.	Inyección de anomalías sintéticas	66
3.4.3.	Visión global de la solución	68
3.4.4.	Preparación de los datos	69
3.4.5.	Formalización y modelado como grafo	71
3.4.6.	Arquitectura del autoencoder	73
3.4.7.	Entrenamiento del modelo	80
3.4.8.	Evaluación y resultados	80
3.4.9.	Entrega visual	84
3.5.	Análisis predictivo mediante Transformers	85
3.5.1.	Formulación del problema	85
3.5.2.	Preparación de los datos y generación de secuencias truncadas	86
3.5.3.	Arquitectura del Transformer Multitarea	87
3.5.4.	Entrenamiento y evaluación	89
3.5.5.	Resultados y visualización	90
3.5.6.	Breve comparativa	93
3.6.	Conclusiones	94
4.	Síntesis y extensiones	95
4.1.	Síntesis del trabajo	95
4.2.	Limitaciones	96
4.3.	Extensiones futuras	97
4.4.	Consideraciones finales	98

Resumen

Resumen en español

Las organizaciones modernas generan grandes volúmenes de datos operacionales cuyos registros de eventos permiten, mediante el *Process Mining* clásico, obtener una visión objetiva de sus procesos. Sin embargo, las limitaciones de este enfoque para manejar datos contextuales, alta variabilidad y predicciones motivan la adopción del aprendizaje profundo (*Deep Learning*).

Este trabajo, enmarcado en las prácticas realizadas en SDG Group, explora la aplicabilidad de dos arquitecturas avanzadas aprovechando la naturaleza dual de los datos de procesos. Por un lado, la arquitectura Transformer modela los registros como secuencias para realizar predicciones precisas a largo plazo. Por otro lado, las Redes Neuronales sobre Grafos (GNN), específicamente mediante capas de atención dinámica (GATv2), representan los procesos como estructuras relacionales para la detección no supervisada de anomalías.

Utilizando como caso de estudio el proceso de concesión de préstamos personales del conjunto de datos *BPI Challenge 2017*, se desarrolla una solución integral bajo la metodología CRISP-DM. Esta abarca un análisis descriptivo clásico, un autoencoder basado en GATv2 para detectar anomalías contextuales y un modelo predictivo multitarea con Transformer que anticipa la siguiente actividad, el tiempo restante y la resolución del caso. Los resultados, desplegados en paneles interactivos, demuestran cómo la combinación de estas tecnologías supera las restricciones del *Process Mining* tradicional, aportando un conocimiento operativo altamente accionable y avanzado.

English abstract

Modern organizations generate large volumes of operational data. Through classical Process Mining, the event logs from this data allow them to obtain an objective view of their processes. However, the limitations of this approach in handling contextual data, high variability, and predictive tasks motivate the adoption of Deep Learning.

This work, framed within an internship at SDG Group, explores the applicability of two advanced architectures by leveraging the dual nature of process data. First, the Transformer architecture models event logs as sequences to achieve accurate, long-term predictions. Second, Graph Neural Networks (GNNs) — specifically utilizing dynamic attention layers (GATv2) — represent processes as relational structures for unsupervised anomaly detection.

Using the personal loan application process from the *BPI Challenge 2017* dataset as a case study, a comprehensive solution is developed following the CRISP-DM methodology. This solution encompasses a classical descriptive analysis, a GATv2-based autoencoder to detect contextual anomalies, and a multi-task predictive model using a Transformer to anticipate the next activity, the remaining time, and the ultimate case resolution. The results, deployed via interactive dashboards, demonstrate how combining these technologies overcomes the constraints of traditional process mining, delivering highly actionable and advanced operational insights.

Introducción

Las organizaciones modernas generan continuamente grandes volúmenes de información operacional como resultado de la ejecución de sus procesos de negocio. Cada solicitud tramitada, cada reclamación gestionada o cada orden de compra procesada deja un rastro digital en los sistemas de información corporativos en forma de registros de eventos. El *Process Mining* (minería de procesos) constituye una disciplina analítica que se sitúa en la intersección entre la gestión de procesos de negocio (*Business Process Management*, BPM) y la minería de datos (*Data Mining*), con el propósito de extraer conocimiento accionable a partir de dichos registros. Estas capacidades permiten a las organizaciones obtener una visión objetiva y basada en datos de sus operaciones, identificar ineficiencias y fundamentar decisiones de optimización.

Sin embargo, los algoritmos clásicos de *Process Mining* presentan limitaciones estructurales que restringen su aplicabilidad en escenarios complejos. En particular, descartan la información contextual asociada a cada evento (importes, recursos, puntuaciones crediticias, tiempos de transición); sufren una explosión combinatoria ante procesos de alta variabilidad que genera modelos ininterpretables; y carecen de mecanismos para realizar inferencia predictiva sobre casos en curso. Estas restricciones motivan la adopción de técnicas de aprendizaje profundo (*Deep Learning*) como complemento al marco clásico.

En el ámbito del aprendizaje profundo, se estudiarán dos familias de arquitecturas para el análisis de procesos de negocio. Por un lado, las Redes Neuronales sobre Grafos (*Graph Neural Networks*, GNNs) están diseñadas para operar sobre datos con estructura relacional. A diferencia de las redes neuronales convencionales, que presuponen un dominio regular (como rejillas de píxeles), las GNNs procesan entidades conectadas por relaciones arbitrarias mediante un mecanismo de paso de mensajes (*message passing*), donde cada nodo actualiza iterativamente su representación integrando información de sus vecinos. Las variantes con mecanismos de atención, como las Graph Attention Networks (GAT y GATv2), aprenden además a ponderar dinámicamente la relevancia de cada conexión en función de su contenido, permitiendo que el modelo se especialice en las relaciones más informativas para la tarea. Por otro lado, la arquitectura Transformer, originalmente propuesta en el campo del Procesamiento de Lenguaje Natural (NLP), introduce el mecanismo de autoatención (*self-attention*) que permite modelar dependencias directas entre cualquier par de elementos de una secuencia. Esta capacidad para capturar dependencias de largo alcance y procesar secuencias de forma paralela ha convertido al Transformer en la arquitectura dominante tanto en lenguaje natural como en múltiples dominios secuenciales.

La conexión entre estas arquitecturas y el *Process Mining* surge de forma natural al considerar la estructura intrínseca de los registros de eventos. Cada traza de un proceso puede

interpretarse simultáneamente desde dos perspectivas complementarias. Como secuencia ordenada de eventos, constituye el dominio natural de los Transformers, que pueden aprender representaciones contextuales de cada evento integrando información de toda la historia del caso para realizar predicciones sobre su evolución futura. Como grafo dirigido donde los nodos son eventos y las aristas representan transiciones temporales, define el escenario idóneo para las GNNs, que pueden propagar información a lo largo de la estructura del proceso y aprender representaciones sensibles tanto al contenido de cada evento como a su contexto relacional. Esta dualidad permite abordar de forma conjunta las limitaciones de *Process Mining* clásico: los Transformers habilitan la predicción sobre prefijos parciales (siguiente actividad, tiempo restante, resolución del caso), mientras que los autoencoders basados en GNNs permiten detectar anomalías contextuales de forma no supervisada al cuantificar las desviaciones respecto a los patrones de normalidad aprendidos.

El presente trabajo se enmarca en el contexto de las prácticas realizadas en SDG Group. El objetivo principal es explorar la aplicabilidad de las técnicas de aprendizaje profundo al análisis de procesos de negocio, utilizando como caso de estudio el proceso de concesión de préstamos personales de una entidad financiera holandesa, registrado en el conjunto de datos del *BPI Challenge 2017*. La solución desarrollada abarca tres pilares analíticos complementarios: un análisis descriptivo del proceso mediante técnicas clásicas, un sistema de detección no supervisada de anomalías basado en un autoencoder con capas GATv2, y un modelo predictivo multitarea basado en Transformer capaz de anticipar simultáneamente la resolución del caso, la siguiente actividad, el tiempo restante y la aceptación de ofertas.

La estructura de esta memoria se organiza de la siguiente forma. El Capítulo 1 establece los fundamentos formales del *Process Mining* clásico, incluyendo la formalización de eventos, trazas y registros, los principales algoritmos de descubrimiento de procesos y las técnicas de verificación de conformidad, concluyendo con un análisis de las limitaciones que motivan la transición hacia métodos basados en aprendizaje profundo. El Capítulo 2 desarrolla el marco teórico de las arquitecturas de *Deep Learning* empleadas, formalizando los componentes matemáticos fundamentales (embeddings, funciones de activación, normalización, regularización), las redes neuronales sobre grafos con atención dinámica (GATv2), la arquitectura Transformer basada en autoatención multi-cabeza y el formalismo de autoencoders con funciones de pérdida multi-tarea. El Capítulo 3 presenta el caso de estudio completo, siguiendo la metodología CRISP-DM desde la comprensión del problema y los datos hasta el diseño, entrenamiento y evaluación de las soluciones propuestas, incluyendo la entrega de los resultados mediante paneles interactivos. Finalmente, el Capítulo 4 ofrece una síntesis global del trabajo realizado, analiza las limitaciones encontradas y propone extensiones futuras.

Capítulo 1

Fundamentos de *Process Mining* clásico

El *Process Mining* comenzó a desarrollarse a finales de los años noventa como respuesta a la necesidad de analizar los procesos reales ejecutados por las organizaciones a partir de los registros generados por sus sistemas de información. Durante las décadas posteriores se desarrollaron numerosos algoritmos para descubrir modelos de proceso, verificar su conformidad y analizar su rendimiento.

Tras presentar el contexto general del trabajo, este capítulo introduce los fundamentos teóricos y matemáticos de *Process Mining* clásico. Para ello se formalizan las estructuras de datos utilizadas, se describen los principales métodos de descubrimiento y conformidad, y se analizan sus limitaciones, motivando posteriormente el uso de técnicas basadas en aprendizaje profundo.

1.1. Formalización matemática de un proceso

1.1.1. *Process Mining* como puente entre la minería de datos y el BPM

La gestión de procesos de negocio (*Business Process Management*, BPM) proporciona métodos y notaciones para diseñar, documentar y optimizar los flujos de trabajo corporativos. Su foco principal es comprender cómo debería funcionar una organización desde una perspectiva de negocio, definiendo modelos que describen el comportamiento esperado de dichas operativas (van der Aalst, 2016).

Por otro lado, la minería de datos (*Data Mining*) se basa en técnicas estadísticas y computacionales orientadas a extraer conocimiento a partir de grandes volúmenes de información. Su objetivo es identificar patrones, relaciones y regularidades ocultas en los datos mediante tareas como clasificación, regresión, agrupamiento o detección de anomalías.

El *Process Mining* (PM) surge precisamente como un puente entre ambas disciplinas. Del BPM hereda el interés por comprender y mejorar los procedimientos organizativos; de la minería de datos, adopta los métodos analíticos necesarios para extraer conocimiento de los

registros generados por los sistemas de información. Ejemplos habituales son la tramitación de solicitudes de préstamo, la gestión de reclamaciones, los ciclos de compra o los circuitos asistenciales en hospitales, donde cada instancia corresponde a la ejecución de una actividad en un momento temporal determinado, generando una secuencia de eventos almacenada digitalmente.

Tradicionalmente, las capacidades de *Process Mining* se agrupan en tres grandes categorías:

- **Descubrimiento de procesos** (*Process Discovery*): obtención automática de un modelo a partir de un registro de eventos, capaz de explicar el proceso subyacente.
- **Verificación de conformidad** (*Conformance Checking*): comparación entre la ejecución observada y un modelo de referencia para identificar desviaciones.
- **Mejora de procesos** (*Process Enhancement*): incorporación de información adicional sobre rendimiento, recursos o tiempos para enriquecer un modelo existente.

Para analizar las técnicas orientadas a las capacidades mencionadas, es indispensable describir matemáticamente la estructura de los datos sobre los que operan. Las siguientes subsecciones presentan los conceptos de evento, traza y registro de eventos, elementos que configuran la base de los algoritmos clásicos de *Process Mining*, siguiendo a van der Aalst (2016).

1.1.2. Universos, eventos y *data payload*

Para analizar un proceso a partir de los registros generados por un sistema de información es necesario formalizar primero qué entendemos por evento, actividad y caso. Esta formalización permite describir los *logs* de eventos mediante estructuras matemáticas bien definidas sobre las que posteriormente se construirán los algoritmos de descubrimiento y verificación de conformidad.

Siguiendo la notación propuesta por van der Aalst (2016), se introducen los universos fundamentales que intervienen en la descripción de un log de eventos.

Definición 1.1.1 (Universos básicos). Se definen los siguientes conjuntos:

- $\mathcal{U}_E$: universo de eventos. Cada $e \in \mathcal{U}_E$ representa una ocurrencia atómica e irreplicable dentro de un proceso.
- $\mathcal{U}_A$: universo de actividades. Cada $a \in \mathcal{U}_A$ es una etiqueta que describe el tipo de acción ejecutada.
- $\mathcal{U}_C$: universo de identificadores de caso. Cada $c \in \mathcal{U}_C$ identifica de forma unívoca una instancia de ejecución del proceso.
- $(\mathcal{T}, \leq)$: dominio temporal, totalmente ordenado, con $\mathcal{T} \subseteq \mathbb{R}$. Cada elemento $t \in \mathcal{T}$ representa una posible marca temporal (*timestamp*) de ejecución de un evento.
- $\mathcal{U}_D$: universo de valores de atributo adicionales, incluyendo características asociadas a un evento: valores reales como importes, variables categóricas como recursos, etc.

Sobre estos universos se define la estructura interna de un evento mediante funciones de proyección.

Definición 1.1.2 (Evento y proyecciones). Un evento $e \in \mathcal{U}_E$ es una entidad observable a la que se asocian las siguientes funciones de proyección:

$$\begin{aligned}\pi_{\text{case}} : \mathcal{U}_E &\rightarrow \mathcal{U}_C, \\ \pi_{\text{act}} : \mathcal{U}_E &\rightarrow \mathcal{U}_A, \\ \pi_{\text{time}} : \mathcal{U}_E &\rightarrow \mathcal{T}.\end{aligned}$$

Adicionalmente, se define la función de *data payload*:

$$\mathcal{D} : \mathcal{U}_E \rightarrow \bigcup_{k=0}^{\infty} \mathcal{U}_D^k,$$

donde $\mathcal{U}_D^k$ denota el producto cartesiano de k atributos pertenecientes al universo $\mathcal{U}_D$.

La función $\mathcal{D}$ asocia a cada evento una tupla de valores de longitud variable. Dado que en un proceso real la cantidad de información contextual depende de la actividad ejecutada, esta formulación elimina la restricción de asumir una dimensión estática global.

En base a las definiciones anteriores, en su representación más compacta, un evento se describe como:

$$e = (\pi_{\text{case}}(e), \pi_{\text{act}}(e), \pi_{\text{time}}(e), \mathcal{D}(e)). \quad (1.1)$$

Esta formulación pone de manifiesto que un evento no es simplemente una etiqueta de actividad, sino un objeto multidimensional definido en un espacio producto.

La componente $\mathcal{D}(e)$, el *data payload*, cobra especial relevancia para aplicaciones avanzadas de análisis predictivo y detección de anomalías, pues contiene información contextual que distingue entre ejecuciones estructuralmente idénticas pero operativamente diferentes. No obstante, la teoría clásica de *Process Mining* opera primordialmente sobre la proyección de actividades. Esta reducción constituye una de las principales causas de las limitaciones que se discutirán al final del capítulo.

1.1.3. Trazas como secuencias ordenadas

Dado un evento $e \in \mathcal{U}_E$, la agrupación por caso y la ordenación temporal inducen la noción de traza. Distinguiremos entre traza enriquecida y traza clásica en función de la información mostrada de cada evento.

Definición 1.1.3 (Traza enriquecida). Sea $c \in \mathcal{U}_C$ un identificador de caso y sea $E_c = \{e \in \mathcal{U}_E : \pi_{\text{case}}(e) = c\}$ el conjunto finito de eventos asociados a c , con $|E_c| = n_c < \infty$. Consideremos la secuencia única $\langle e_1, e_2, \dots, e_{n_c} \rangle$ conformada por los elementos de E_c construida por la condición

$$\pi_{\text{time}}(e_i) \leq \pi_{\text{time}}(e_{i+1}), \quad \forall i = 1, \dots, n_c - 1.$$

En caso de igualdad temporal ($\pi_{\text{time}}(e_i) = \pi_{\text{time}}(e_{i+1})$), se supone definido un orden total sobre el conjunto de eventos refinando el orden temporal mediante el identificador del evento,

lo que garantiza la unicidad de la secuencia. Bajo estas premisas, la traza enriquecida de c se define como

$$\sigma_c^E = \langle e_1, e_2, \dots, e_{n_c} \rangle \in \mathcal{U}_E^*, \quad (1.2)$$

donde cada elemento $e_i \in E_c$ se descompone como la tupla $(\pi_{\text{case}}(e_i), \pi_{\text{act}}(e_i), \pi_{\text{time}}(e_i), \mathcal{D}(e_i))$. El conjunto $\mathcal{U}_E^*$ representa el monoide libre generado por el universo de eventos $\mathcal{U}_E$ bajo la operación de concatenación (Lothaire, 1997). Es decir, el conjunto de todas las posibles secuencias generables a partir de dicho universo de eventos.

Definición 1.1.4 (Traza clásica). Dada la traza enriquecida $\sigma_c^E = \langle e_1, e_2, \dots, e_{n_c} \rangle$ de un caso $c \in \mathcal{U}_C$, la traza clásica correspondiente se define como la secuencia de actividades

$$\sigma_c = \langle a_1, a_2, \dots, a_{n_c} \rangle \in \mathcal{U}_A^*,$$

donde $a_i = \pi_{\text{act}}(e_i)$ para cada posición $i \in \{1, \dots, n_c\}$. El conjunto $\mathcal{U}_A^*$ representa el monoide libre generado por $\mathcal{U}_A$ bajo la operación de concatenación.

La proyección $e \mapsto \pi_{\text{act}}(e)$ constituye una pérdida de información: se descartan las marcas temporales, los recursos y el *data payload*. Todo análisis basado exclusivamente en σ_c hereda la pérdida de información inherente a esta proyección.

La función *variant* : $\mathcal{U}_E^* \rightarrow \mathcal{U}_A^*$ definida por

$$\text{variant}(\sigma_c^E) = \sigma_c \quad (1.3)$$

se introduce de forma inmediata. Este operador será empleado en el Capítulo 3 para conocer la variabilidad del proceso estudiado, a partir del número de trazas clásicas distintas, las cuales representan variantes de ejecución de actividades.

1.1.4. El registro de actividades y el registro de eventos

La agregación de las trazas permite construir los registros globales del proceso. Dependiendo de si se opera con secuencias de eventos completos (trazas enriquecidas) o de actividades puras (trazas clásicas), se distingue entre el registro de eventos (*event log*) y el registro de actividades (*activity log*).

Definición 1.1.5 (Registro de eventos). Un registro de eventos L_E es un conjunto de trazas enriquecidas sobre el monoide libre $\mathcal{U}_E^*$:

$$L_E = \{\sigma_1^E, \sigma_2^E, \dots, \sigma_N^E\} \subset \mathcal{U}_E^*, \quad (1.4)$$

donde N es el número total de casos independientes analizados. Debido a que cada evento es único, L_E es un conjunto y se tiene que $|L_E| = N$.

Un multiconjunto M sobre un conjunto base X constituye una generalización de la noción de conjunto clásico, definida matemáticamente mediante una función de asignación multiplicidad $M : X \rightarrow \mathbb{N}_0$. Para cada elemento $x \in X$, el valor $M(x)$ representa el número de veces (multiplicidad) que dicho elemento pertenece al multiconjunto.

Bajo esta formalización, denotamos por $\mathcal{B}(X) = \{M \mid M : X \rightarrow \mathbb{N}_0\}$ el espacio de todos los posibles multiconjuntos generables sobre X . Para cualquier multiconjunto $M \in \mathcal{B}(X)$, se define su soporte como el conjunto clásico de aquellos elementos que tienen una multiplicidad estrictamente mayor que cero, denotado por $\text{supp}(M) = \{x \in X \mid M(x) > 0\}$.

Definición 1.1.6 (Registro de actividades). Un registro de actividades L_A es un multiconjunto de trazas clásicas sobre el monoide libre $\mathcal{U}_A^*$:

$$L_A \in \mathcal{B}(\mathcal{U}_A^*).$$

Las trazas clásicas con idéntica secuencia lógica de actividades colapsan en variantes con multiplicidades. En la notación de multiconjuntos, el registro se expresa como:

$$L_A = [\sigma_1^{m_1}, \sigma_2^{m_2}, \dots, \sigma_K^{m_K}],$$

con $m_k \in \mathbb{N}^+$ la multiplicidad de σ_k , $K = |\text{supp}(L_A)|$ el número de variantes distintas y donde la suma de las multiplicidades cumple que $|L_A| = \sum_{k=1}^K m_k = N$.

1.1.5. Ejemplos de procesos reales y sus componentes

Las definiciones anteriores adquieren significado pleno al confrontarlas con procesos organizativos reales. A continuación se presentan tres ejemplos del mundo real que ilustran cómo se materializan las nociones de caso, actividad, traza, *data payload* y registro de eventos en dominios heterogéneos.

Proceso de concesión de un préstamo bancario

En una entidad financiera, cada solicitud de préstamo personal constituye un caso $c \in \mathcal{U}_C$ identificado de forma unívoca por un código interno. Desde el momento en que un cliente solicita financiación hasta la resolución final del expediente, el sistema de información registra cada acción ejecutada sobre la solicitud. Las actividades $a \in \mathcal{U}_A$ pueden incluir la creación de la solicitud, la evaluación crediticia, la generación de una oferta, el envío de documentación, la validación de requisitos o la emisión de una resolución definitiva.

La traza clásica de un caso particular podría expresarse como:

$$\sigma_c = \langle \text{Crear solicitud}, \text{Evaluar}, \text{Crear oferta}, \text{Enviar oferta}, \text{Documentación}, \text{Aprobar} \rangle.$$

Sin embargo, esta representación omite información esencial para comprender el verdadero comportamiento del proceso. Cada evento porta un *data payload* $\mathcal{D}(e)$ que puede incluir el recurso responsable de ejecutar la actividad, el importe solicitado, la puntuación crediticia del cliente, el canal de comunicación o el importe ofertado por la entidad. Asimismo, las marcas temporales $\pi_{\text{time}}(e)$ permiten calcular duraciones entre actividades y detectar cuellos de botella. El identificador de caso lleva asociados atributos globales estables a lo largo de toda la traza, como la finalidad declarada del préstamo o el tipo de solicitud, que permanecen invariantes para todos los eventos del mismo caso.

Cabe señalar que dos solicitudes con secuencias de actividades idénticas pueden diferir radicalmente en su contexto: un préstamo de 5 000 € aprobado en tres días y otro de 50 000 € aprobado tras varias semanas de negociación comparten la misma traza clásica, pero sus trazas enriquecidas revelan realidades operativas incomparables.

Flujo de atención en urgencias médicas

En un servicio de urgencias hospitalario, cada paciente que ingresa genera un caso. El identificador de caso corresponde al código de admisión asignado en el triaje inicial. Las actividades registradas abarcan desde el registro administrativo y la clasificación por gravedad hasta las pruebas diagnósticas, intervenciones terapéuticas, interconsultas con especialistas y la resolución final del episodio asistencial.

Una posible traza clásica sería:

$$\sigma_c = \langle \text{Triage, Exploración médica, Analítica, Radiografía, Diagnóstico, Tratamiento, Alta} \rangle.$$

El *data payload* de cada evento recoge el profesional sanitario que ejecuta la actividad, la categoría de prioridad asignada, los valores de constantes vitales en el momento de la exploración, los resultados de las pruebas complementarias o el código diagnóstico. A nivel de caso, los atributos estáticos incluyen la edad del paciente, su historial de alergias, el motivo de consulta declarado en el triaje y la cobertura asistencial.

La riqueza del *data payload* adquiere especial importancia en este dominio. Un paciente clasificado como urgente cuya analítica muestra marcadores alterados seguirá una trayectoria radicalmente distinta a otro con idéntica secuencia de actividades pero valores dentro de la normalidad. La proyección sobre actividades puras es insuficiente para explicar la variabilidad clínica observada.

Proceso de compras en una organización industrial

En una empresa manufacturera, cada orden de compra emitida a un proveedor constituye un caso. El proceso abarca la identificación de la necesidad, la solicitud interna, la aprobación presupuestaria, la emisión del pedido, la recepción del material, la verificación de calidad y el pago al proveedor.

Una traza representativa podría ser:

$$\sigma_c = \langle \text{Solicitar material, Aprobar presupuesto, Emitir pedido, Recibir mercancía, Pago} \rangle.$$

Los atributos dinámicos del evento incluyen el responsable de cada aprobación, el almacén de destino, la fecha de recepción efectiva o el resultado de la inspección de calidad. Los atributos estáticos del caso comprenden el código del proveedor, la categoría del material, el importe total del pedido y el centro de coste solicitante.

Los tres ejemplos anteriores comparten una estructura común que puede representarse mediante las definiciones formales de la Sección 1.1. En todos ellos, la traza clásica $\sigma_c \in \mathcal{U}_A^*$

proporciona una visión esquemática del flujo de control, mientras que la traza enriquecida $\sigma_c^E \in \mathcal{U}_E^*$ preserva la totalidad de la información contextual. La agregación de los casos individuales conforma el registro de eventos L_E o, tras la proyección sobre actividades, el registro de actividades L_A . Las técnicas clásicas que se describen en las secciones siguientes operan exclusivamente sobre L_A , heredando por construcción la pérdida de información del *data payload*.

1.2. Redes de Petri como lenguaje de representación

Los algoritmos de descubrimiento de procesos, que se presentarán en la Sección 1.3, producen modelos expresados en distintos formalismos. El α -Miner y el *Heuristic Miner* generan directamente redes de Petri; el *Inductive Miner* produce árboles de proceso que admiten traducción inmediata a este formalismo. La red de Petri constituye la representación canónica en *Process Mining* por su semántica operacional precisa, que permite definir rigurosamente la noción de ejecución válida y, en consecuencia, formalizar los problemas de verificación de conformidad. En esta sección se introduce su definición matemática y las nociones asociadas de marcado y regla de disparo.

Definición 1.2.1 (Red de Petri etiquetada). Una red de Petri etiquetada es una tupla $\mathcal{N} = (P, T, F, \ell)$ donde:

- P es un conjunto finito de **lugares** (*places*), que representan condiciones o estados intermedios del proceso.
- T es un conjunto finito de **transiciones** (*transitions*), que representan actividades ejecutables, con $P \cap T = \emptyset$.
- $F \subseteq (P \times T) \cup (T \times P)$ es la **relación de flujo** (*flow relation*), que conecta lugares con transiciones y transiciones con lugares mediante arcos dirigidos.
- $\ell : T \rightarrow \mathcal{U}_A \cup \{\tau\}$ es la **función de etiquetado**, que asigna a cada transición el nombre de una actividad observable o el símbolo τ para transiciones silenciosas.

Los lugares se representan gráficamente como círculos y las transiciones como rectángulos. Los arcos de la relación de flujo F se dibujan como flechas dirigidas. En el ámbito de este estudio se consideran exclusivamente redes de Petri ordinarias, donde la relación F carece de funciones de multiplicidad y todos los arcos poseen un peso unitario implícito.

Para cada nodo $x \in P \cup T$, se definen el conjunto de predecesores $\bullet x = \{y : (y, x) \in F\}$ y el conjunto de sucesores $x \bullet = \{y : (x, y) \in F\}$.

Definición 1.2.2 (Marcado). Un **marcado** de una red de Petri $\mathcal{N} = (P, T, F, \ell)$ es una función $M : P \rightarrow \mathbb{N}_0$ que asigna a cada lugar un número natural de **tokens**. El marcado describe el estado global de ejecución del proceso en un instante determinado. Se denota como un multiconjunto sobre P y se representa gráficamente mediante puntos negros en el interior de los lugares.

Definición 1.2.3 (Sistema de red marcado). Un **sistema de red marcado** es una tupla $(\mathcal{N}, M_0, M_f)$ donde $\mathcal{N}$ es una red de Petri etiquetada, M_0 es el marcado inicial que describe el estado del proceso antes de comenzar su ejecución y M_f es el marcado final que representa la terminación correcta del proceso.

La dinámica del sistema se rige por la regla de disparo, que determina cuándo una transición puede ejecutarse y cómo se modifica el estado global tras dicha ejecución.

Definición 1.2.4 (Habilitación y disparo). Sea $(\mathcal{N}, M_0, M_f)$ un sistema de red marcado y sea M un marcado alcanzable. Una transición $t \in T$ está *habilitada* en M , denotado $M[t]$, si y solo si:

$$\forall p \in \bullet t : M(p) \geq 1.$$

Es decir, todos los lugares de entrada de t contienen al menos un token. El *disparo* de t produce un nuevo marcado M' , denotado $M[t]M'$, definido como:

$$M'(p) = M(p) - \mathbf{1}_{p \in \bullet t} + \mathbf{1}_{p \in t\bullet}, \quad \forall p \in P,$$

donde $\mathbf{1}_{(\cdot)}$ denota la función indicadora. El disparo consume un token de cada lugar de entrada y deposita un token en cada lugar de salida.

Una secuencia de disparos $\rho = \langle t_1, t_2, \dots, t_n \rangle$ es *factible* desde M_0 si existe una sucesión de marcados $M_0[t_1]M_1[t_2]M_2 \cdots [t_n]M_n$. Si además $M_n = M_f$, la secuencia conduce del marcado inicial al final y se considera una ejecución completa del proceso. La traza observable asociada a ρ se obtiene proyectando sobre las etiquetas no silenciosas:

$$\sigma(\rho) = \langle \ell(t_i) : t_i \in \rho, \ell(t_i) \neq \tau \rangle.$$

El *lenguaje* de la red es el conjunto de todas las trazas observables generadas por secuencias de disparo factibles que conectan M_0 con M_f :

$$\mathcal{L}(\mathcal{N}) = \{\sigma(\rho) : M_0[\rho]M_f\} \subseteq \mathcal{U}_A^*.$$

Por convención terminológica y notacional en *Process Mining*, se empleará el término red de Petri (y el símbolo $\mathcal{N}$) para denotar indistintamente tanto la estructura estática como el sistema de red marcado $(\mathcal{N}, M_0, M_f)$ siempre que el contexto no dé lugar a ambigüedad.

Este lenguaje formaliza el comportamiento permitido por el modelo: una traza pertenece a $\mathcal{L}(\mathcal{N})$ si y solo si puede ser reproducida íntegramente mediante una secuencia válida de disparos. La verificación de conformidad, objeto de la Sección 1.4, se reduce esencialmente a evaluar en qué medida las trazas del registro empírico pertenecen al lenguaje del modelo descubierto.

Nótese que la red de Petri opera exclusivamente sobre etiquetas de actividades. Los tokens no portan información adicional: no codifican importes, recursos ni atributos temporales. La semántica del modelo describe relaciones de precedencia, elección y concurrencia entre actividades, pero permanece completamente ajena al contenido de los eventos que las instancian.

La propiedad de *Soundness*

Un sistema de red marcado $(\mathcal{N}, M_0, M_f)$ satisface la propiedad de *soundness* si cumple tres condiciones fundamentales (van der Aalst, 2016):

- **Terminación garantizada** (*option to complete*): para todo marcado M alcanzable desde M_0 , existe una secuencia de disparo que conduce a M_f . Es decir, desde cualquier estado intermedio es siempre posible alcanzar la finalización correcta del proceso.
- **Marcado final propio**: cuando se alcanza M_f , todos los demás lugares de la red están vacíos. Esto garantiza que no quedan tokens residuales que representen trabajo pendiente o actividades sin concluir.
- **Ausencia de transiciones muertas**: toda transición $t \in T$ es disparable en al menos un marcado alcanzable desde M_0 . No existen partes del modelo que sean estructuralmente inalcanzables.

La propiedad de *soundness* garantiza que el modelo de proceso carece de *deadlocks* (estados sin salida donde ninguna transición está habilitada y no se ha alcanzado M_f) y de fragmentos inalcanzables. Esta propiedad constituye un requisito de calidad estructural que los algoritmos de descubrimiento como el *Inductive Miner* garantizan por construcción, mientras que otros métodos como el α -Miner pueden generar redes que la violan.

1.3. Descubrimiento de procesos (*Process Discovery*)

Las secciones que siguen no pretenden desarrollar de forma exhaustiva la construcción matemática y algorítmica de cada método de descubrimiento y evaluación. Su propósito es proporcionar al lector una comprensión conceptual sólida del funcionamiento de las técnicas clásicas, suficiente para ilustrar los fundamentos de la disciplina y, sobre todo, para identificar con rigor las limitaciones estructurales que motivarán, en el Capítulo 2, la adopción de arquitecturas basadas en aprendizaje profundo.

1.3.1. Formulación del problema de descubrimiento

El descubrimiento de procesos constituye el problema inverso canónico de esta disciplina. Dado un registro de actividades L_A , se busca construir un modelo formal capaz de describir el comportamiento subyacente del proceso. En el contexto de este capítulo, el universo de modelos $\mathcal{M}$ se restringe a los sistemas de red marcados formalizados en la Sección 1.2, donde cada modelo define un lenguaje formal $\mathcal{L}(\mathcal{N}) \subseteq \mathcal{U}_A^*$. Esta noción de lenguaje se interpreta en su sentido algebraico estricto como el subconjunto de todas las trazas secuenciales de actividades permitidas por las reglas del modelo.

Establecidos estos espacios algebraicos, el problema de descubrimiento se formaliza mediante la búsqueda de un operador:

$$\Delta : \mathcal{B}(\mathcal{U}_A^*) \longrightarrow \mathcal{M},$$

donde $\mathcal{N} = \Delta(L_A)$ representa el modelo extraído para un registro de actividades L_A concreto.

El modelo resultante debe encontrar un equilibrio entre la fidelidad a la evidencia empírica y la capacidad de abstracción. Un modelo que replica todas las variantes observadas sin generalizar incurre en sobreajuste, mientras que uno excesivamente simplificado pierde capacidad explicativa. Este compromiso se materializa en las métricas de calidad del modelo formalizadas en la Sección 1.4 (van der Aalst, 2016; Carmona et al., 2018).

Es importante destacar que el operador Δ recibe como entrada un registro de actividades, es decir, un multiconjunto de secuencias de etiquetas de actividades. Toda la información contextual contenida en las trazas enriquecidas queda descartada antes de que el algoritmo de descubrimiento comience su ejecución. Esta observación resulta crucial para comprender el alcance y las limitaciones inherentes a los métodos clásicos. Las relaciones descubiertas reflejan exclusivamente patrones de precedencia entre etiquetas de actividades, sin capacidad alguna para capturar dependencias sobre atributos numéricos, recursos o temporalidad.

1.3.2. El *Directly-Follows Graph*

Antes de describir los algoritmos que instancian el operador Δ , se introduce el *Directly-Follows Graph* (DFG) como herramienta exploratoria. Si bien el DFG no constituye un modelo de proceso con semántica operacional y, por tanto, no pertenece a $\mathcal{M}$, sintetiza la información de sucesión directa contenida en el log en forma de un grafo dirigido ponderado, donde el peso de cada arista representa la frecuencia empírica con la que una actividad es seguida directamente por otra. De este modo, proporciona la base empírica sobre la que operan los métodos formales.

Definición 1.3.1 (Directly-Follows Graph). Sea $L_A \in \mathcal{B}(\mathcal{U}_A^*)$ un registro de actividades y sea $\Sigma(L_A) = \{a \in \mathcal{U}_A : \exists \sigma = \langle a_1, \dots, a_n \rangle \in \text{supp}(L_A), \exists i \in \{1, \dots, n\} \text{ tal que } a = a_i\}$ el conjunto de actividades observadas. El DFG inducido por L_A es el grafo:

$$G_{\text{DFG}}(L_A) = (V, E, w),$$

con $V = \Sigma(L_A)$, aristas

$$E = \{(a, b) \in V \times V : \exists \sigma = \langle a_1, \dots, a_n \rangle \in \text{supp}(L_A), \exists i < n : a_i = a \wedge a_{i+1} = b\},$$

y función de peso

$$w(a, b) = \sum_{\sigma \in \mathcal{U}_A^*} L_A(\sigma) \cdot \#_{\sigma}(a, b),$$

donde $\#_{\sigma}(a, b)$ cuenta cuántas veces la actividad a es seguida inmediatamente por b en la secuencia σ .

El DFG admite una representación matricial $\mathbf{W} \in \mathbb{N}^{d \times d}$ con $d = |\Sigma(L_A)|$ y $\mathbf{W}_{ij} = w(a_i, a_j)$. Si se normaliza cada fila por su suma, se obtiene una estimación empírica de la matriz de transición de una cadena de Markov de primer orden sobre el espacio de actividades:

$$\hat{P}_{ij} = \frac{\mathbf{W}_{ij}}{\sum_{k=1}^d \mathbf{W}_{ik}},$$

siempre que $\sum_{k=1}^d \mathbf{W}_{ik} > 0$. Si una actividad a_i nunca tiene sucesores, se establece $\hat{P}_{ij} = 0 \forall j \in \{1, \dots, d\}$

Aunque esta representación recuerda a una cadena de Markov de primer orden, no debe interpretarse como un modelo probabilístico completo del proceso. La normalización de la matriz proporciona únicamente una aproximación empírica de las transiciones observadas en el log. En los procesos reales, el supuesto de independencia condicional implícito en las cadenas de Markov rara vez se cumple, ya que la probabilidad de ejecutar una actividad futura suele depender del historial completo del caso y no únicamente de la actividad inmediatamente anterior.

Filtrado por frecuencias

En registros con elevada variabilidad, el DFG completo puede contener numerosas aristas de baja frecuencia que dificultan la interpretación visual. Para obtener representaciones más compactas se aplican mecanismos de filtrado. Sea $\delta_w > 0$ un umbral de frecuencia; el DFG filtrado se define como:

$$G_{\text{DFG}}^\delta(L_A) = (V, E^\delta, w), \quad \text{con} \quad E^\delta = \{(a, b) \in E : w(a, b) \geq \delta_w\}.$$

El filtrado elimina transiciones infrecuentes, potencialmente debidas a ruido o comportamiento excepcional, preservando las relaciones con mayor soporte empírico. El ajuste del umbral δ_w permite navegar entre la visión completa del proceso y una representación resumida que prioriza la interpretabilidad.

Pese a su utilidad como herramienta exploratoria, el DFG carece de semántica operacional propia: no distingue la elección exclusiva de la concurrencia, no detecta sincronizaciones entre ramas paralelas y no posee noción de estado de ejecución. Los algoritmos de descubrimiento que buscan modelos con semántica rica deben superar las limitaciones inherentes a la mera contabilización de sucesiones directas.

1.3.3. El α -Miner: algoritmo fundacional

El α -Miner, propuesto por van der Aalst et al. (van der Aalst, 2016), está ampliamente aceptado como el primer algoritmo formal de descubrimiento de procesos y posee un interés fundamentalmente didáctico e histórico. Su lógica se sustenta en relaciones algebraicas binarias derivadas del registro de actividades.

Dados $a, b \in \Sigma(L_A)$, se definen las siguientes relaciones sobre la base del log:

- **Sucesión directa:** $a >_L b$ si y solo si existe al menos una traza en $\text{supp}(L_A)$ donde a precede inmediatamente a b .
- **Causalidad empírica:** $a \rightarrow_L b$ si y solo si $a >_L b$ y no se observa $b >_L a$.
- **Paralelismo:** $a \parallel_L b$ si y solo si se observan simultáneamente $a >_L b$ y $b >_L a$.
- **Independencia:** $a \#_L b$ si y solo si no se observa ni $a >_L b$ ni $b >_L a$.

A partir de estas relaciones, el algoritmo identifica conjuntos maximales de actividades causalmente conectadas y construye un modelo formal que recoge las condiciones intermedias entre dichas actividades.

El α -Miner presenta numerosas limitaciones, detalladas por van der Aalst (2016), que restringen su aplicabilidad en entornos reales: extrema sensibilidad al ruido, incapacidad para manejar bucles de longitud uno y dos, etc. Estas carencias estructurales provocan que el algoritmo genere modelos con trayectorias de ejecución sin salida, impidiendo que el proceso pueda alcanzar su estado final correctamente.

1.3.4. El *Heuristic Miner*: robustez frente al ruido

El *Heuristic Miner* (HM), introducido por Weijters et al. (2006) y formalizado posteriormente en Weijters and Ribeiro (2011), destaca por su capacidad para operar sobre registros ruidosos. Su principio rector consiste en reemplazar la lógica binaria de presencia o ausencia del α -Miner por una lógica cuantitativa basada en frecuencias relativas. De forma que la confianza depositada en una relación causal es proporcional a la evidencia empírica que la respalda.

Definición 1.3.2 (Medida de dependencia). Dados $a, b \in \Sigma(L_A)$, sea $w(a, b)$ la frecuencia de la sucesión directa $a >_L b$. La medida de dependencia se define como:

$$\text{dep}(a, b) = \frac{w(a, b) - w(b, a)}{w(a, b) + w(b, a) + 1}.$$

Los valores $\text{dep}(a, b) \in (-1, 1)$ capturan la asimetría direccional de la relación entre dos actividades. Valores cercanos a 1 indican una fuerte dependencia causal de a hacia b ; valores próximos a 0 sugieren ausencia de dirección dominante, ya que ambas direcciones se observan con frecuencia comparable; y valores cercanos a -1 señalan la relación causal inversa. La constante aditiva $+1$ en el denominador evita la división por cero y actúa como suavizado laplaciano, penalizando relaciones basadas en escasa evidencia absoluta.

El algoritmo construye el modelo de proceso aplicando dos umbrales configurables:

- **Umbral de dependencia** δ_{dep} : una arista causal $a \rightarrow b$ se incluye en el modelo solo si $\text{dep}(a, b) \geq \delta_{\text{dep}}$.
- **Umbral de frecuencia absoluta** δ_{freq} : se exige adicionalmente que $w(a, b)$ supere un mínimo absoluto para ser considerada.

Aumentar δ_{dep} genera modelos más simples con el riesgo de excluir secuencias legítimas presentes en el registro; disminuirlo produce modelos más detallados que pueden incorporar ruido. Esta regulación constituye un mecanismo implícito de control del compromiso sesgo-varianza (Weijters and Ribeiro, 2011). La diferencia principal respecto al DFG es la capacidad para discriminar entre señal y ruido, lo que hace al HM adecuado para registros reales con variantes excepcionales y errores de registro.

1.3.5. El *Inductive Miner*: garantías estructurales

El *Inductive Miner* (Leemans et al., 2013a) adopta una estrategia de divide y vencerás recursiva que garantiza por construcción la *soundness* del modelo resultante. En cada paso recursivo, el algoritmo busca un corte en el log que permita descomponer el conjunto de actividades en subconjuntos con una relación bien definida. Los tipos de corte posibles son: secuencial, paralelo, de elección exclusiva y de bucle. La descomposición recursiva genera un árbol de proceso cuyas hojas son actividades individuales y cuyos nodos internos representan operadores de composición.

Su variante *Infrequent* (IMf) (Leemans et al., 2013b) incorpora filtrado de aristas de baja frecuencia sin comprometer las garantías estructurales. Al eliminar comportamiento infrecuente antes de buscar los cortes, IMf produce modelos que toleran el ruido sin sacrificar la propiedad de *soundness*. El *Inductive Miner* constituye uno de los estándares contemporáneos para el descubrimiento automatizado, produciendo modelos bloque-estructurados, interpretables y libres de *deadlocks*.

1.3.6. Alcance de los algoritmos de descubrimiento

Todos los algoritmos descritos en esta sección comparten una característica fundamental que conviene explicitar. La entrada de cada uno de ellos es un registro de actividades L_A , es decir, un multiconjunto de secuencias de etiquetas de actividades. Ninguno de estos métodos incorpora en su lógica interna las marcas temporales, los recursos implicados, los atributos financieros ni cualquier otra dimensión del *data payload*. El modelo de proceso resultante es un objeto puramente estructural que describe relaciones de precedencia, concurrencia y elección entre actividades, sin capacidad para representar las condiciones bajo las cuales una u otra rama del proceso se ejecuta.

La ausencia de contexto del evento no es un defecto de implementación, sino una consecuencia directa del planteamiento formal del problema de descubrimiento. La exposición de estos algoritmos en el presente capítulo cumple un propósito exclusivamente fundacional: establecer la semántica procesual y las representaciones formales sobre las que se construye la disciplina. Las técnicas tradicionales no serán empleadas como solución a los problemas de predicción y detección de anomalías abordados en este trabajo, dado que la información relevante para dichas tareas reside precisamente en las dimensiones que el *Process Mining* clásico descarta.

1.4. Verificación de conformidad y métricas de calidad

1.4.1. Motivación del problema

Una vez que se dispone de un modelo de proceso $\mathcal{N}$, el interés se desplaza hacia la evaluación cuantitativa de la consistencia entre el comportamiento empíricamente registrado y el comportamiento formalmente permitido por el modelo. Sea $\mathcal{L}(\mathcal{N}) \subseteq \mathcal{U}_A^*$ el lenguaje de la red de Petri y sea $\text{supp}(L_A) = \{\sigma : L_A(\sigma) > 0\}$ el soporte del registro de actividades. En la práctica, la intersección entre ambos conjuntos rara vez es completa: existen trazas observadas

que el modelo no reproduce y caminos del modelo que nunca se ejecutan. La verificación de conformidad cuantifica esta discrepancia (Rozinat and van der Aalst, 2008; Carmona et al., 2018).

Las dos técnicas principales difieren en su planteamiento algorítmico. El *token replay* adopta un enfoque que intenta reproducir cada traza de forma secuencial, registrando las desviaciones locales. Los *alignments* reformulan el problema como una búsqueda de camino óptimo que minimiza globalmente el coste de las desviaciones. Ambas técnicas operan directamente sobre la semántica de la red de Petri formalizada en la Sección 1.2.

1.4.2. Las cuatro dimensiones de calidad de un modelo

La comunidad de *Process Mining* evalúa la calidad de un modelo descubierto mediante cuatro dimensiones complementarias e interdependientes (van der Aalst, 2016; Carmona et al., 2018; van der Aalst, 2012). Su comprensión resulta necesaria para contextualizar las decisiones metodológicas de los algoritmos de descubrimiento y para interpretar los resultados de las técnicas de conformidad.

Fitness

Cuantifica la capacidad del modelo para reproducir el comportamiento observado en el registro. Un valor de 1 indica que todas las trazas del log pueden alinearse con coste cero. Un modelo universal cuyo lenguaje sea $\mathcal{L}(\mathcal{N}) = \mathcal{U}_A^*$ obtendría fitness perfecto pero carecería de valor descriptivo. El fitness es, por tanto, condición necesaria pero no suficiente para la calidad.

Precision

La dimensión de precisión penaliza el exceso de permisividad del modelo, evaluando si la topología descubierta admite trayectorias legítimas o comportamientos que carecen de respaldo empírico. Una posible métrica es la desarrollada por Muñoz-Gama and Carmona (2010). El cálculo se define de manera global mediante la siguiente ecuación:

$$\text{Precision}(L_A, \mathcal{N}) = \frac{\sum_{\sigma \in L_A} \sum_{k=1}^{|\sigma|} |\text{Obs}(\sigma \upharpoonright_{k-1}, L_A)|}{\sum_{\sigma \in L_A} \sum_{k=1}^{|\sigma|} |\text{En}(\sigma \upharpoonright_{k-1}, \mathcal{N})|},$$

donde L_A representa el registro de actividades global y σ corresponde a una traza individual de longitud $|\sigma|$. El término $\sigma \upharpoonright_{k-1}$ denota el prefijo o secuencia parcial recorrida dentro de la traza hasta el paso anterior al evento k .

Sobre esta base, $\text{En}(\sigma \upharpoonright_{k-1}, \mathcal{N})$ define el conjunto de transiciones que el modelo de proceso $\mathcal{N}$ mantiene teóricamente habilitadas tras la ejecución de dicho prefijo, mientras que $\text{Obs}(\sigma \upharpoonright_{k-1}, L_A)$ representa el conjunto de actividades que efectivamente se han observado en los datos reales sucediendo de forma inmediata a esa misma secuencia. La diferencia matemática entre el denominador y el numerador cuantifica las transiciones permitidas por el modelo

que la realidad ignora; por tanto, un valor de precisión bajo indica que el modelo permite numerosos comportamientos que nunca aparecen en los datos.

Generalization

Evalúa si el modelo acepta variaciones plausibles no observadas en el registro. Un modelo con baja generalización ha memorizado el log sin abstraer sus regularidades subyacentes. Una posible aproximación propuesta por van der Aalst (2016) es:

$$\text{Gen}(L_A, \mathcal{N}) = 1 - \frac{1}{|T|} \sum_{t \in T} \sqrt{\frac{1}{n_t}},$$

donde n_t es el número de trazas del log que disparan la transición t al menos una vez.

Simplicity

Implementa el principio de parsimonia: entre modelos con capacidad explicativa similar, se prefiere el más compacto. Los modelos excesivamente complejos deterioran la interpretabilidad y dificultan la comunicación con los responsables de negocio. Para cuantificar esta dimensión en modelos representados mediante redes de Petri $N = (P, T, F)$, se recurre a métricas de tamaño estructural (Mendling, 2008).

En concreto, se define el *tamaño del modelo* (*Size*) como la suma de sus nodos constituyentes:

$$\text{Size}(N) = |P| + |T|.$$

Bajo esta métrica, un menor valor de *Size* se asocia de forma directa con una mayor simplicidad estructural y, por ende, con una mejor facilidad de comprensión.

Relación entre dimensiones

Estas cuatro dimensiones presentan compromisos inevitables, por lo que no es posible maximizar todas simultáneamente. Incrementar el *fitness* puede reducir la *precision* si el modelo se vuelve excesivamente permisivo; maximizar la *simplicity* puede sacrificar *fitness* al eliminar caminos necesarios; y perseguir la *generalization* sin control genera modelos laxos que admiten comportamiento arbitrario. La evaluación de un modelo de proceso es, en consecuencia, un problema multicriterio donde el equilibrio entre dimensiones depende del objetivo analítico perseguido.

De las cuatro dimensiones expuestas, el *fitness* constituye la más fundamental y la primera en ser evaluada. Las dos subsecciones siguientes formalizan varios procedimientos algorítmicos que permiten su cálculo riguroso.

1.4.3. Token Replay

El *token replay* constituye la técnica de conformidad más intuitiva y computacionalmente eficiente. Su mecánica consiste en intentar reproducir cada traza $\sigma = \langle a_1, \dots, a_n \rangle$ del registro

sobre la red de Petri $(\mathcal{N}, M_0, M_f)$, disparando secuencialmente las transiciones correspondientes a cada actividad observada. Durante la reproducción se mantienen cuatro contadores acumulativos:

- p : tokens **producidos**, es decir, depositados en lugares de salida tras cada disparo.
- c : tokens **consumidos**, retirados de lugares de entrada para habilitar los disparos.
- m : tokens **faltantes** (*missing*), creados artificialmente cuando una transición no está habilitada en el marcado actual pero la traza exige su disparo.
- r : tokens **restantes** (*remaining*), presentes en la red al finalizar la reproducción que no corresponden al marcado final M_f .

La medida de fitness propuesta por Rozinat and van der Aalst (2008) combina ambos tipos de desviación:

$$\text{fitness}_{\text{TR}}(\sigma, \mathcal{N}) = \frac{1}{2} \left(1 - \frac{m}{c} \right) + \frac{1}{2} \left(1 - \frac{r}{p} \right).$$

Dado que los tokens producidos actúan como cota superior para los restantes ($r \leq p$), de igual manera que los tokens consumidos para los faltantes ($m \leq c$), se sigue que esta medida toma valores en el intervalo $[0, 1]$. Un valor de 1 indica reproducción perfecta, donde todos los disparos están habilitados y el marcado final coincide con M_f .

Para el registro completo, la métrica se agrega ponderando por la multiplicidad de cada variante:

$$\text{fitness}_{\text{TR}}(L_A, \mathcal{N}) = \frac{1}{2} \left(1 - \frac{\sum_{\sigma} L_A(\sigma) \cdot m_{\sigma}}{\sum_{\sigma} L_A(\sigma) \cdot c_{\sigma}} \right) + \frac{1}{2} \left(1 - \frac{\sum_{\sigma} L_A(\sigma) \cdot r_{\sigma}}{\sum_{\sigma} L_A(\sigma) \cdot p_{\sigma}} \right).$$

La principal limitación del *token replay* reside en su carácter voraz (*greedy*). Al reproducir la traza de izquierda a derecha sin considerar el alineamiento global óptimo, puede sobreestimar el coste de las desviaciones en situaciones donde una reordenación o un camino alternativo a través de la red minimizaría los tokens faltantes.

1.4.4. *Alignments*: alineamientos óptimos

Los *alignments* resuelven la limitación dimensional de la reproducción voraz reformulando la verificación de conformidad como un problema de optimización global (Carmona et al., 2018). En lugar de reproducir la traza de forma secuencial, se busca la correspondencia óptima entre la traza observada y una ejecución válida del modelo que minimice el coste total de las desviaciones.

Sea $\gg$ el símbolo de movimiento nulo. Un alineamiento entre una traza σ y un sistema de red $(\mathcal{N}, M_0, M_f)$ se define formalmente como una secuencia de pares:

$$\gamma = \left\langle \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}, \begin{pmatrix} x_2 \\ y_2 \end{pmatrix}, \dots, \begin{pmatrix} x_m \\ y_m \end{pmatrix} \right\rangle,$$

donde la fila superior representa los eventos del registro ($x_i \in \mathcal{U}_A \cup \{\gg\}$) y la fila inferior las transiciones del modelo ($y_i \in T \cup \{\gg\}$). Cada par se clasifica en uno de los tres siguientes tipos de movimientos:

- **Movimiento síncrono:** $x_i = \ell(y_i)$. La actividad del registro coincide con la transición disparada en el modelo, representando concordancia perfecta.
- **Movimiento en log** (*move on log*): $y_i = \gg$. La actividad se observó en el registro pero no tiene correspondencia en el modelo, denotando un comportamiento no previsto.
- **Movimiento en modelo** (*move on model*): $x_i = \gg$. La transición se dispara en el modelo sin que exista una actividad correspondiente en la traza, indicando un paso omitido en la realidad.

Dada una función de coste estándar κ que asigna coste 0 a los movimientos síncronos y coste 1 a las desviaciones (movimientos en log y en modelo), el alineamiento óptimo se formula como:

$$\gamma^* \in \arg \min_{\gamma \in \Gamma(\sigma, \mathcal{N})} \sum_{i=1}^m \kappa(x_i, y_i),$$

donde $\Gamma(\sigma, \mathcal{N})$ representa el conjunto de todos los alineamientos factibles, cuya proyección inferior constituye una secuencia de disparo válida desde M_0 hasta M_f .

La métrica de ajuste se normaliza dividiendo el coste del alineamiento óptimo entre el coste máximo posible, $\text{cost}_{\text{máx}}(\sigma)$. Este peor escenario teórico asume una discrepancia absoluta entre los datos y la estructura, calculándose formalmente como la suma de dos costes independientes: el coste de procesar la traza σ al completo mediante movimientos en log exclusivos (lo que equivale a su longitud, $|\sigma|$), más el coste de recorrer la ruta síncrona más corta y viable del modelo $\mathcal{N}$ desde el marcado inicial M_0 hasta el final M_f empleando únicamente movimientos en modelo:

$$\text{fitness}_{\text{AL}}(\sigma, \mathcal{N}) = 1 - \frac{\sum_{i=1}^m \kappa(x_i, y_i) |_{\gamma^*}}{\text{cost}_{\text{máx}}(\sigma)}$$

donde $\text{cost}_{\text{máx}}(\sigma) = |\sigma| + \min_{\gamma_m \in \Gamma(\langle \rangle, \mathcal{N})} \text{cost}(\gamma_m)$. Esta aproximación proporciona una noción de distancia semánticamente rigurosa, determinista y robusta frente al ruido, delegando la complejidad del proceso a la penalización por coste, aunque introduce una carga computacional superior a la del *token replay*.

1.5. Limitaciones del enfoque clásico

La exposición realizada a lo largo de este capítulo ha presentado los fundamentos conceptuales y algorítmicos del *Process Mining* clásico. Resulta ahora imprescindible sintetizar las limitaciones estructurales de este paradigma para motivar su extensión hacia enfoques complementarios basados en aprendizaje profundo, los cuales se desarrollarán en el Capítulo 2.

1.5.1. Explosión combinatoria y modelos ininterpretables

La primera limitación se manifiesta ante procesos de alta variabilidad. El espacio potencial de configuraciones crece exponencialmente con el número de actividades. Cuando $K = |\text{supp}(L_A)|$ es elevado y las variantes comparten regularidades dispersas, el algoritmo de descubrimiento intenta comprimir toda esa variabilidad en un único objeto gráfico global. El resultado es un modelo de tipo “*Spaghetti*” cuya densidad de arcos y lugares lo hace ilegible, no auditable y carente de valor diagnóstico para los responsables del proceso.

1.5.2. Exclusión del *data payload*

Esta limitación constituye la más relevante para los objetivos del presente trabajo. Todos los algoritmos clásicos operan exclusivamente sobre la proyección de actividades $\sigma_c = \langle \pi_{\text{act}}(e_1), \dots, \pi_{\text{act}}(e_n) \rangle$, descartando la totalidad del *data payload*. Esta reducción proyecta un objeto de alta dimensionalidad sobre una única variable categórica de resolución limitada. Dos trazas con secuencias de actividades idénticas pueden corresponder a contextos completamente distintos en términos de riesgo crediticio, temporalidad, recursos implicados o probabilidad de resultado adverso. El *Process Mining* clásico las trata como instancias equivalentes de la misma variante.

Recuperando los ejemplos de la Sección 1.1, en el proceso de préstamo bancario variables como el importe solicitado, la puntuación crediticia del cliente o las condiciones de la oferta pueden determinar completamente el desenlace del caso sin que se produzca diferencia alguna en la secuencia de actividades. En el proceso hospitalario, los valores de constantes vitales y los resultados analíticos condicionan la trayectoria clínica sin necesidad de que varíe el flujo de control. Las redes de Petri, los DFGs y las métricas de conformidad describen relaciones de precedencia entre etiquetas simbólicas, pero no aprenden representaciones latentes que integren conjuntamente la estructura secuencial y el contenido informacional de los eventos.

1.5.3. Naturaleza retrospectiva y ausencia de capacidad predictiva

El *Process Mining* clásico es intrínsecamente retrospectivo ya que opera sobre trazas completadas. Sin embargo, en aplicaciones operativas el interés reside en responder cuestiones proactivas sobre casos en curso. Dado un prefijo $\sigma \upharpoonright_k$ de un caso cuya ejecución aún no ha finalizado, las preguntas de mayor valor para la organización son: ¿cuál es la siguiente actividad más probable? ¿Cuánto tiempo restante se estima hasta la resolución? ¿Cuál será el resultado final del caso? ¿Está evolucionando hacia un comportamiento anómalo?

Las estructuras clásicas carecen de mecanismos para estimar distribuciones predictivas condicionadas a prefijos parciales. Las dependencias de largo alcance, donde la probabilidad de un evento futuro depende de decisiones tomadas muchas actividades atrás, de combinaciones no lineales entre atributos o de patrones temporales complejos, quedan fuera del alcance de las representaciones simbólicas.

1.5.4. Puente hacia el aprendizaje profundo

Las limitaciones descritas configuran un escenario donde el *Process Mining* clásico proporciona los cimientos conceptuales y la semántica procesual, pero no las herramientas necesarias para abordar los problemas de mayor interés operativo. La transición hacia el aprendizaje profundo, desarrollada en el Capítulo 2, no pretende sustituir la semántica procesual, sino complementarla. Allí donde el enfoque clásico formaliza estructura simbólica sobre secuencias de actividades, las arquitecturas profundas aprenden representaciones distribuidas que integran simultáneamente secuencias, atributos y temporalidad, capturando dependencias de largo alcance y optimizando objetivos predictivos sobre trazas no finalizadas.

No obstante, el marco clásico formalizado en este capítulo dispone de una materialización computacional consolidada que resulta de gran utilidad para el análisis descriptivo de procesos. La librería `pm4py` (Berti et al., 2023) constituye el estándar de referencia en Python para la implementación de algoritmos clásicos. Desarrollada por investigadores de la Universidad Tecnológica de Eindhoven, proporciona implementaciones eficientes del DFG, α -Miner, *Heuristic Miner*, *Inductive Miner*, técnicas de conformidad y métricas de calidad. Su arquitectura modular permite la integración directa con el ecosistema de ciencia de datos de Python, facilitando la construcción de flujos de trabajo que combinan *Process Mining* clásico con técnicas de aprendizaje automático. En el contexto de este trabajo, `pm4py` se empleará en el Capítulo 3 como herramienta para el análisis descriptivo del proceso de concesión de préstamos, operativizando los conceptos teóricos aquí formalizados.

Capítulo 2

Aprendizaje profundo aplicado a *Process Mining*

2.1. Justificación del aprendizaje profundo en procesos secuenciales

El Capítulo 1 estableció los fundamentos formales del *Process Mining* clásico y concluyó identificando tres limitaciones estructurales inherentes al marco clásico: la explosión combinatoria de modelos ante procesos de alta variabilidad, la incapacidad para integrar el *data payload* (atributos asociados a un evento) y la naturaleza retrospectiva que impide la inferencia predictiva sobre casos en curso. Estas restricciones justifican la transición hacia un enfoque complementario.

El aprendizaje profundo (*Deep Learning*) ofrece una respuesta natural a estas limitaciones. A diferencia de los modelos simbólicos del *Process Mining* clásico, las redes neuronales profundas aprenden representaciones latentes que integran simultáneamente la estructura secuencial, la información temporal y los atributos heterogéneos del evento, sin requerir un modelo normativo previo del proceso. Formalmente, dada una traza enriquecida $\sigma_c^E = \langle e_1, \dots, e_n \rangle$ con $e_i = (\pi_{\text{act}}(e_i), \pi_{\text{time}}(e_i), \mathcal{D}(e_i))$, una red neuronal f_θ produce un vector latente:

$$\mathbf{z} = f_\theta(\sigma_c^E) \in \mathbb{R}^d,$$

que condensa la información de toda la traza en un espacio geométrico continuo donde operaciones como proximidad, proyección y regresión adquieren significado semántico.

El *significado semántico* del espacio latente se fundamenta en el aprendizaje de representaciones, mediante el cual el modelo aprende una codificación de las trazas que captura las regularidades y patrones relevantes del proceso (Bishop, 2006; Goodfellow et al., 2016). La geometría del espacio latente adquiere una interpretación funcional: la distancia entre vectores refleja el grado de similitud entre los comportamientos representados, determinadas direcciones o proyecciones permiten aislar factores de variación asociados a características específicas del proceso, y la continuidad del espacio favorece que pequeñas modificaciones en la secuencia de eventos produzcan cambios graduales en la representación latente $\mathbf{z}$ (Murphy, 2022).

Esta capacidad resulta especialmente relevante para las dos familias de problemas que motivan este trabajo:

- **Predicción sobre prefijos** (*Predictive Process Monitoring*): dado un prefijo parcial $\sigma^E \upharpoonright_k = \langle e_1, \dots, e_k \rangle$ con $1 \leq k \leq n$, estimar la distribución condicional $P(Y \mid \sigma \upharpoonright_k, \mathbf{x}_1, \dots, \mathbf{x}_k)$ para variables de interés Y (siguiente actividad, tiempo restante, actividad final). Los trabajos de Tax et al. (2017) y Teinmaa et al. (2019) establecen el marco formal de este problema.
- **Detección no supervisada de anomalías**: aprender los patrones de normalidad del proceso y detectar instancias cuyo comportamiento de tipo estructural, temporal o contextual se desvía significativamente de los patrones aprendidos, sin requerir etiquetas de anomalía ni un modelo prescriptivo externo (Nolle et al., 2022).

En las secciones siguientes se desarrollan los componentes matemáticos fundamentales que constituyen la base teórica de ambas arquitecturas: las transformaciones y normalizaciones de entrada (Sección 2.2), las redes neuronales sobre grafos con mecanismos de atención (Sección 2.3), las arquitecturas Transformer basadas en autoatención (Sección 2.4) y el formalismo de autoencoders y funciones de pérdida (Sección 2.5).

El objetivo no es explicar con el máximo rigor y nivel de detalle cada uno de los conceptos presentados, en el sentido de una revisión bibliográfica, sino proporcionar suficiente contexto para comprender los algoritmos implementados en el Capítulo 3, intentando que el trabajo sea autocontenido en la mayor medida posible.

2.2. Componentes matemáticos y transformaciones base

Antes de abordar las arquitecturas profundas, es necesario formalizar las técnicas que permiten transformar los datos crudos de un registro de eventos en representaciones numéricas aptas para el procesamiento neuronal. Cada uno de estos componentes cumple un papel específico en el flujo de información de la red, y es relevante respecto a las decisiones de diseño del Capítulo 3.

2.2.1. *Embeddings* categóricos

Las variables categóricas del evento como la actividad ejecutada, el recurso responsable u otros atributos específicos de cada proceso son nominales: no admiten operaciones aritméticas ni poseen una noción natural de distancia. Un recurso denominado “Recurso_045” no es intrínsecamente más que “Recurso_012”, ni una actividad “Validar documentación” está algebraicamente más cerca de “Enviar oferta” que de “Cancelar solicitud”. Sin embargo, las redes neuronales operan exclusivamente sobre vectores numéricos y requieren que las categorías se representen en un espacio donde la aritmética tenga significado.

La solución habitual es la codificación *one-hot*, que representa cada categoría como un vector binario de dimensión igual a la cardinalidad. Esta resulta inadecuada por dos razones.

Primero, produce vectores extremadamente dispersos y de alta dimensión (un recurso entre 200 posibles genera un vector de 200 componentes con un solo 1). Segundo, todos los pares de categorías son equidistantes y ortogonales entre sí, dificultando que el modelo descubra relaciones de similitud funcional.

La técnica estándar en aprendizaje profundo consiste en proyectar cada categoría hacia un espacio vectorial continuo de dimensión reducida mediante matrices de *embeddings* entrenables.

Definición 2.2.1 (*Embedding* categórico). Sea $\mathcal{C}_m$ el conjunto de categorías posibles de la variable m -ésima, con $|\mathcal{C}_m| = C_m$. Un *embedding* categórico es una función parametrizada:

$$\mathbf{E}_m : \mathcal{C}_m \rightarrow \mathbb{R}^{d_e}, \quad \mathbf{E}_m(c) = (\mathbf{E}_m)_{c,\bullet},$$

donde $\mathbf{E}_m \in \mathbb{R}^{C_m \times d_e}$ es una matriz de parámetros entrenables y d_e es la dimensión del *embedding*.

Computacionalmente, esta operación es simplemente una consulta (*lookup*) a una tabla: dada la categoría c , se extrae la fila c -ésima de la matriz $\mathbf{E}_m$. Los valores de esta tabla no están fijos, sino que se aprenden durante el entrenamiento de forma conjunta con el resto de la red mediante retropropagación del gradiente. Si el modelo descubre que dos actividades aparecen en contextos similares, sus respectivos *embeddings* se encontrarán en regiones cercanas del espacio $\mathbb{R}^{d_e}$. Por ejemplo, en un proceso de solicitud de préstamos bancarios “Validar documentación” y “Revisar solicitud” suelen preceder a decisiones de aprobación. De igual forma, dos recursos con el mismo puesto de trabajo que acostumbran realizar tareas similares deberían ser parecidos a los ojos de un modelo.

Esta idea tiene un precedente clásico en el procesamiento de lenguaje natural. Los modelos Word2Vec y GloVe aprenden representaciones vectoriales de palabras donde relaciones semánticas se traducen en relaciones geométricas (por ejemplo, $\text{vec}(\text{“rey”}) - \text{vec}(\text{“hombre”}) + \text{vec}(\text{“mujer”}) \approx \text{vec}(\text{“reina”})$). En nuestro contexto, los *embeddings* de actividades capturan relaciones análogas: actividades que cumplen roles procesales similares ocuparán regiones cercanas del espacio latente.

Dado un evento con M variables categóricas y un vector de variables numéricas $\mathbf{x}_i^{num} \in \mathbb{R}^p$, la representación unificada se construye por concatenación:

$$\mathbf{x}_i^{full} = \left[\mathbf{x}_i^{num} \parallel \mathbf{E}_1(cat_1^{(i)}) \parallel \mathbf{E}_2(cat_2^{(i)}) \parallel \cdots \parallel \mathbf{E}_M(cat_M^{(i)}) \right] \in \mathbb{R}^{p + \sum_{j=1}^M d_{e,j}}, \quad (2.1)$$

donde $d_{e,j}$ denota la dimensión del espacio de *embedding* asociado a la j -ésima variable categórica.

Esta concatenación produce un vector denso que fusiona toda la información disponible del evento, numérica y categórica, en una representación unificada apta para las capas subsiguientes de la red.

2.2.2. Funciones de activación y la ruptura de la linealidad

Una red neuronal sin funciones de activación no lineal es, independientemente de su profundidad, equivalente a una única transformación lineal. Si cada capa computa $\mathbf{h}^{(l+1)} =$

$W^{(l)}\mathbf{h}^{(l)} + \mathbf{b}^{(l)}$, la composición de L capas lineales produce:

$$\mathbf{h}^{(L)} = W^{(L-1)} \dots W^{(1)} W^{(0)} \mathbf{x} + \mathbf{b}_{\text{eff}} = W_{\text{eff}} \mathbf{x} + \mathbf{b}_{\text{eff}},$$

donde $W_{\text{eff}} = \prod_{l=0}^{L-1} W^{(L-1-l)}$ representa la matriz de pesos efectiva y $\mathbf{b}_{\text{eff}}$ constituye el vector de sesgo efectivo resultante, el cual colapsa como una combinación lineal de los sesgos (término independiente) de cada capa transformados por las matrices de pesos subsecuentes:

$$\mathbf{b}_{\text{eff}} = \mathbf{b}^{(L-1)} + \sum_{l=0}^{L-2} \left(\prod_{j=l+1}^{L-1} W^{(j)} \right) \mathbf{b}^{(l)}.$$

De este modo, el vector resultante es una sola transformación afín del vector original. Esto limita drásticamente la capacidad del modelo: sólo puede aprender fronteras de decisión lineales y relaciones proporcionales, cuando los procesos empresariales presentan habitualmente interacciones no lineales. Por ejemplo, el impacto del importe solicitado sobre la probabilidad de aprobación de un préstamo bancario no es proporcional, puede existir un umbral a partir del cual el riesgo crece de forma mucho más pronunciada.

Las funciones de activación resuelven esta limitación introduciendo no linealidades entre las capas lineales, dotando a la red de capacidad para aproximar funciones arbitrariamente complejas.

Definición 2.2.2 (ReLU (*Rectified Linear Unit*)). La función de activación ReLU se define como:

$$\text{ReLU}(x) = \max(0, x) = \begin{cases} x, & \text{si } x > 0, \\ 0, & \text{si } x \leq 0. \end{cases}$$

ReLU actúa como una compuerta: deja pasar la señal intacta si es positiva (la neurona está “activa”) y la anula si es negativa (la neurona está “apagada”). Esta operación, aparentemente trivial, tiene dos propiedades fundamentales. En primer lugar, es no lineal; la composición de capas lineales con ReLU intercaladas puede aproximar cualquier función continua definida sobre dominio compacto con precisión arbitraria (teorema de aproximación universal (Cybenko, 1989; Hornik et al., 1989)). Segundo, para valores positivos el gradiente es exactamente 1, lo que permite que la señal de error se propague eficientemente durante el entrenamiento sin atenuarse (*vanishing gradients*). Sin embargo, las neuronas cuya entrada es siempre negativa producen gradiente cero permanentemente, fenómeno conocido como *dying neurons*.

Definición 2.2.3 (GELU (*Gaussian Error Linear Unit*)). La función GELU suaviza la transición alrededor del origen:

$$\text{GELU}(x) = x \cdot \Phi(x),$$

donde Φ es la función de distribución acumulada de la normal estándar. En la práctica se emplean aproximaciones para reducir el coste computacional.

A diferencia de ReLU, que produce una transición abrupta en $x = 0$, GELU permite un flujo de gradiente no nulo para valores ligeramente negativos, ponderando cada entrada por su probabilidad de ser positiva bajo una distribución gaussiana. Esta activación ha mostrado un

rendimiento competitivo en arquitecturas Transformer para tareas de procesamiento del lenguaje natural y ha sido adoptada en modelos como BERT y, posteriormente, GPT (Hendrycks and Gimpel, 2016; Devlin et al., 2019).

Definición 2.2.4 (LeakyReLU). La función de activación LeakyReLU introduce una pequeña pendiente constante para los valores negativos y se define matemáticamente como:

$$\text{LeakyReLU}(x) = \max(\alpha x, x) = \begin{cases} x, & \text{si } x > 0, \\ \alpha x, & \text{si } x \leq 0, \end{cases}$$

donde $\alpha \in (0, 1)$ es un hiperparámetro fijo que determina la inclinación en la región negativa.

Esta variante surge como una solución directa al problema de las neuronas muertas (*dying neurons*) intrínseco a la ReLU convencional. Al asignar una pendiente no nula α a las entradas negativas, LeakyReLU garantiza que el gradiente nunca se sature ni se anule por completo en esa región, manteniendo un valor constante igual a α . De este modo, aquellas neuronas que reciben activaciones negativas retienen la capacidad de transmitir la señal de error y actualizar sus parámetros durante la fase de retropropagación.

2.2.3. *Layer Normalization y Dropout*: estabilización y regularización

El entrenamiento de redes profundas se enfrenta a dos problemas complementarios: la inestabilidad de las activaciones internas, que dificulta la convergencia, y el sobreajuste, que degrada la capacidad de generalización. Layer Normalization y Dropout abordan respectivamente cada uno de estos problemas, y su uso conjunto constituye un patrón arquitectónico estándar en todas las redes profundas modernas.

Layer Normalization

A medida que la señal se propaga por las capas de una red profunda, las distribuciones de las activaciones internas pueden desplazarse y cambiar de escala de forma impredecible (fenómeno conocido como *internal covariate shift*). Esto obliga al optimizador a ajustar continuamente los parámetros para compensar estos cambios, ralentizando la convergencia. Tradicionalmente, las técnicas de normalización se han interpretado como un mecanismo para mitigar este fenómeno y mejorar el entrenamiento, si bien esta explicación ha sido posteriormente matizada en la literatura (Santurkar et al., 2018).

Definición 2.2.5 (Layer Normalization). Dado un vector de activaciones $\mathbf{h} \in \mathbb{R}^d$, Layer Normalization computa:

$$\text{LN}(\mathbf{h}) = \gamma \odot \frac{\mathbf{h} - \mu_{\mathbf{h}} \mathbf{1}_d}{\sqrt{\sigma_{\mathbf{h}}^2 + \epsilon}} + \beta,$$

donde el símbolo $\odot$ denota la multiplicación elemento a elemento entre dos vectores y $\mathbf{1}_d = [1, 1, \dots, 1] \in \mathbb{R}^d$. Los términos $\mu_{\mathbf{h}} \in \mathbb{R}$ y $\sigma_{\mathbf{h}}^2 \in \mathbb{R}_{\geq 0}$ son escalares puros que definen la media y la varianza muestral calculadas a través de las d componentes del propio vector $\mathbf{h}$:

$$\mu_{\mathbf{h}} = \frac{1}{d} \sum_{i=1}^d h_i, \quad \sigma_{\mathbf{h}}^2 = \frac{1}{d} \sum_{i=1}^d (h_i - \mu_{\mathbf{h}})^2.$$

El escalar $\epsilon > 0$ es una constante de estabilidad numérica, y $\gamma, \beta \in \mathbb{R}^d$ son parámetros de escala y desplazamiento aprendibles.

La operación tiene dos fases. Como primera operación, se centra y reescala el vector para que tenga media cero y varianza unitaria, garantizando que las activaciones permanezcan en un rango numérico estable independientemente de la capa o del instante del entrenamiento. Después, los parámetros aprendibles γ y β permiten que la red recupere, si es necesario, la escala y el desplazamiento óptimos para la tarea. La normalización no elimina capacidad expresiva, sino que reparametriza la representación de forma más favorable para la optimización.

A diferencia de *Batch Normalization* (otra técnica de normalización que normaliza a lo largo de la dimensión de batch y requiere lotes de tamaño suficiente), Layer Normalization opera dentro de cada instancia individual, normalizando a lo largo de la dimensión de características. Esto la hace independiente del tamaño del batch y especialmente adecuada para datos secuenciales de longitud variable, donde los batches pueden tener composiciones heterogéneas.

Dropout como regularización estocástica

La regularización ataca el problema del sobreajuste (*overfitting*), un fenómeno crítico en el aprendizaje profundo donde los modelos, debido a su elevada capacidad matemática y densidad de parámetros, tienden a memorizar el conjunto de datos de entrenamiento en lugar de abstraer sus patrones generales subyacentes. Al carecer de restricciones, la red neuronal absorbe por igual tanto la señal útil del proceso como el ruido, si su capacidad lo permite. Como consecuencia directa, el modelo exhibe un rendimiento óptimo en la fase de entrenamiento, pero sufre una severa degradación en su capacidad de generalización, siendo incapaz de predecir correctamente al enfrentarse a datos nuevos. Las técnicas de regularización actúan introduciendo penalizaciones matemáticas o perturbaciones estocásticas (como el Dropout) para conducir al modelo hacia la comprensión las relaciones profundas en los datos y evitar la memorización.

Definición 2.2.6 (Dropout). Durante el entrenamiento, Dropout anula aleatoriamente una fracción p de las componentes del vector de activaciones:

$$\text{Dropout}(\mathbf{h}, p) = \frac{1}{1-p} \mathbf{h} \odot \mathbf{m}, \quad m_i \sim \text{Bernoulli}(1-p),$$

donde $\mathbf{m} \in \{0, 1\}^d$ es una máscara binaria aleatoria y el factor $\frac{1}{1-p}$ compensa la reducción de magnitud esperada. Durante la inferencia, Dropout se desactiva y la red opera con todas sus neuronas.

El fundamento de Dropout es el siguiente, al apagar aleatoriamente neuronas en cada paso de entrenamiento, se impide que la red desarrolle co-adaptaciones excesivas. Estas son patrones donde un grupo de neuronas solo funciona correctamente si todas están activas simultáneamente. De este modo, se fuerza a cada neurona a aprender representaciones útiles por sí misma, sin depender de la presencia de compañeras específicas. Este proceso fuerza a la red a encontrar una configuración de pesos óptima y consensuada que funcione bajo múltiples escenarios de omisión de neuronas. En la fase de inferencia se desactiva el Dropout y el resultado es una red con mayor capacidad de generalización que resiste mejor las particularidades de los datos de entrenamiento.

Justificación de su uso conjunto

El patrón estándar **Linear** → **Activación** → **LayerNorm** → **Dropout** (o variantes del mismo) aparece repetidamente en las arquitecturas de este trabajo. La razón es que estos componentes se complementan:

- **La activación** (ReLU/GELU) rompe la linealidad, permitiendo modelar relaciones complejas.
- **Layer Normalization** estabiliza las distribuciones internas, acelerando la convergencia y permitiendo tasas de aprendizaje más agresivas.
- **Dropout** regulariza, previniendo el sobreajuste al introducir ruido estocástico durante el entrenamiento.

Juntos, estos tres componentes permiten entrenar redes profundas que convergen de forma estable, aprenden representaciones expresivas y generalizan correctamente a datos no vistos.

2.3. Aprendizaje profundo sobre grafos (GNNs)

Las secciones anteriores han formalizado los bloques elementales que transforman datos crudos en representaciones numéricas. Sin embargo, estas transformaciones operan sobre cada evento de forma aislada, sin considerar las relaciones entre eventos. En esta sección se introduce la familia de arquitecturas capaz de explotar la estructura relacional de los datos: las redes neuronales sobre grafos. Se presenta primero la motivación intuitiva mediante dominios donde los grafos son la representación natural, después el formalismo de *message passing* inherente a esta familia, y finalmente las variantes con atención (GAT/GATv2).

2.3.1. Motivación: datos con estructura relacional

Muchos problemas del mundo real se modelan de forma natural como grafos, entidades (nodos) conectadas por relaciones (aristas). Consideremos como ejemplo una red social, donde cada persona es un nodo con atributos (edad, intereses, ubicación), y las amistades son aristas que conectan pares de personas. Si queremos predecir si un usuario se unirá a un grupo o si una cuenta es fraudulenta, la respuesta no depende solo de los atributos del usuario aislado, sino de con quién está conectado y de las propiedades de esos vecinos. Un usuario con tres amigos identificados como fraudulentos es más sospechoso que otro con conexiones legítimas, aunque ambos tengan atributos individuales idénticos.

De forma análoga, en una red de interacciones proteicas, los nodos representan proteínas y las aristas codifican interacciones bioquímicas. En una red de citas académicas, los nodos son artículos y las aristas representan referencias bibliográficas. En todos estos dominios, la información relevante no reside solo en el nodo aislado, sino en su contexto relacional formado por el vecindario, los vecinos de los vecinos y los patrones estructurales que emergen de la topología del grafo.

Las redes neuronales convencionales (*fully-connected*, convolucionales, recurrentes, etc.) no pueden operar directamente sobre grafos porque presuponen una estructura regular del dominio, por ejemplo, una rejilla de píxeles para imágenes. Un grafo, en cambio, tiene vecindarios de tamaño variable, no posee un sistema global de coordenadas y es invariante al etiquetado arbitrario de los nodos. Las *Graph Neural Networks* (GNNs) se diseñaron precisamente para resolver esta limitación, son arquitecturas capaces de operar sobre dominios relacionales irregulares, aprendiendo representaciones nodales que integran tanto los atributos propios como la estructura del entorno.

La conexión con Process Mining. Un registro de eventos puede representarse de forma natural como una colección de grafos. Cada traza enriquecida se modela como un grafo dirigido secuencial donde los nodos son eventos y las aristas son transiciones temporales entre eventos consecutivos. Formalmente, cada traza enriquecida $\sigma_c^E = \langle e_1, \dots, e_n \rangle$ se transforma en un grafo $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{E}_{\text{attr}})$. La matriz de características nodales $\mathbf{X} \in \mathbb{R}^{n \times F}$ materializa computacionalmente el *data payload* formalizado en el Capítulo 1, recogiendo los atributos asociados a cada evento. De este modo, el paso de la formalización teórica del evento a su representación computacional es:

$$e_i = (\pi_{\text{case}}, \pi_{\text{act}}, \pi_{\text{time}}, \mathcal{D}(e_i)) \longrightarrow \mathbf{x}_i \in \mathbb{R}^F,$$

que llamaremos vector de características del nodo i .

Los atributos de arista $\mathbf{E}_{\text{attr}}$ codifican la información de cada transición, como el tiempo entre eventos consecutivos. De esta forma, el grafo construido recoge los atributos del nodo, el flujo estructural del proceso e información sobre las transiciones entre eventos.

2.3.2. El paradigma de *Message Passing*

El paradigma unificador de las GNNs modernas es el *Message Passing Neural Network* (MPNN), que formaliza la propagación de información en grafos mediante una metáfora de comunicación: cada nodo “envía mensajes” a sus vecinos, “recibe mensajes” de ellos, y “actualiza” su estado basándose en la información agregada (Wu et al., 2021).

Sea $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{E}_{\text{attr}})$ un grafo atribuido con $N = |\mathcal{V}|$ nodos, aristas $\mathcal{E}$, una matriz de características nodales $\mathbf{X} \in \mathbb{R}^{N \times F}$ (donde la fila i -ésima contiene los F atributos del nodo i) y atributos de arista $\mathbf{E}_{\text{attr}}$.

Definición 2.3.1 (Marco de Message Passing). Sea $\mathbf{h}_i^{(0)} \in \mathbb{R}^{d_0}$ la representación inicial de cada nodo (habitualmente construida a partir de $\mathbf{x}_i$ vector de características del nodo mediante las transformaciones presentadas en la Sección 2.2). En la capa l -ésima del modelo la representación $\mathbf{h}_i^{(l)} \in \mathbb{R}^{d_l}$ se actualiza mediante tres operaciones:

$$\begin{aligned} \mathbf{m}_{ij}^{(l)} &= \phi^{(l)}(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}, \mathbf{a}_{ij}), & j \in \mathcal{N}(i), & \text{(Mensaje)} \\ \bar{\mathbf{m}}_i^{(l)} &= \bigoplus_{j \in \mathcal{N}(i)} \mathbf{m}_{ij}^{(l)}, & & \text{(Agregación)} \\ \mathbf{h}_i^{(l+1)} &= \psi^{(l)}(\mathbf{h}_i^{(l)}, \bar{\mathbf{m}}_i^{(l)}), & & \text{(Actualización)} \end{aligned}$$

donde $\phi^{(l)}$ es la función de mensaje que computa la información transmitida desde el vecino j al nodo i (incorporando opcionalmente el atributo de arista $\mathbf{a}_{ij}$), $\oplus$ es una operación de agregación invariante a permutaciones (suma, media o máximo), y $\psi^{(l)}$ es la función de actualización que combina la representación actual del nodo con la información agregada de su vecindario. El vecindario $\mathcal{N}(i)$ denota el conjunto de nodos directamente conectados a i . Véase que la elección construir un grafo dirigido o bidireccional afecta directamente al vecindario de los nodos, y por tanto al pase de mensajes.

Para ilustrar la mecánica del message passing, regresemos al ejemplo de la red social. Supongamos que queremos clasificar a los usuarios según su riesgo de fraude. Inicialmente, cada usuario i posee una representación $\mathbf{h}_i^{(0)}$ compuesta por sus atributos individuales (edad, frecuencia de transacciones, antigüedad). En la primera capa de message passing, el usuario i “recibe” los atributos de todos sus amigos directos, los agrega (por ejemplo, sumándolos o promediándolos) y combina esa información con su propia representación para producir $\mathbf{h}_i^{(1)}$. Tras esta operación, $\mathbf{h}_i^{(1)}$ ya no describe solo al usuario i , sino que incorpora información sobre quiénes son sus contactos. Si apilamos una segunda capa, $\mathbf{h}_i^{(2)}$ incorporará información de los amigos de sus amigos (vecinos a distancia 2). En general, L capas permiten capturar dependencias de hasta L saltos de distancia.

En el contexto de *Process Mining*, con $L = 2$ capas sobre un grafo secuencial, cada evento incorpora información de sus dos predecesores inmediatos: sabe qué ocurrió uno y dos pasos antes de él, lo que le permite contextualizar su propia situación dentro del flujo del caso.

Equivarianza a permutaciones. Una restricción fundamental que justifica la forma del operador de agregación $\oplus$ es la equivarianza a permutaciones. En un grafo, el etiquetado de los nodos es arbitrario: si $P \in \mathbb{R}^{N \times N}$ es una matriz de permutación que reordena los índices nodales, las entradas se transforman como $\mathbf{X} \mapsto P\mathbf{X}$ y la conectividad (matriz de adyacencia) como $A \mapsto PAP^T$. Para que el modelo produzca representaciones consistentes con independencia de la numeración escogida, toda capa Φ debe satisfacer:

$$\Phi(PAP^T, P\mathbf{X}) = P\Phi(A, \mathbf{X}).$$

Esta restricción algebraica implica que $\oplus$ no puede depender del orden de enumeración de los vecinos, restringiendo las opciones a funciones simétricas como la suma, media o máximo. Es una consecuencia formal, no una decisión heurística.

2.3.3. Graph Attention Networks (GAT)

Las *Graph Convolutional Networks* (GCN) de Kipf and Welling (2017) asignan la importancia de cada vecino mediante factores fijos que dependen exclusivamente de la topología (los grados de los nodos). En la metáfora de la red social, sería como ponderar la opinión de cada amigo por el número de amigos que tiene, independientemente de lo que diga. Las *Graph Attention Networks* (GAT) de Veličković et al. (2018) resuelven esta limitación introduciendo coeficientes de atención aprendidos. La red aprende a qué vecinos “prestar más atención” en función de su contenido.

Sea $\mathbf{h}_i \in \mathbb{R}^F$ la representación del nodo i . Dado un peso compartido $\Theta \in \mathbb{R}^{F \times F'}$ y un vector de atención $\mathbf{a} \in \mathbb{R}^{2F'}$, GAT computa un *score* de relevancia para cada par vecino:

$$e_{ij} = \text{LeakyReLU}(\mathbf{a}^T [\Theta \mathbf{h}_i \parallel \Theta \mathbf{h}_j]), \quad j \in \mathcal{N}(i) \cup \{i\},$$

donde $\parallel$ denota concatenación. Los scores se normalizan mediante la función *softmax*, usualmente empleada para scores arbitrarios en probabilidades, para obtener coeficientes de atención:

$$\alpha_{ij} = \text{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}(i) \cup \{i\}} \exp(e_{ik})},$$

y la representación actualizada es la media ponderada de las representaciones proyectadas de los vecinos junto a una función de activación ς :

$$\mathbf{h}'_i = \varsigma \left(\sum_{j \in \mathcal{N}(i) \cup \{i\}} \alpha_{ij} \Theta \mathbf{h}_j \right).$$

2.3.4. GATv2: atención dinámica

Brody et al. (2022) demuestran que el mecanismo GAT original sufre una limitación fundamental que denominan *atención estática*. Al aplicar la no linealidad (LeakyReLU) después de la concatenación de las proyecciones lineales y del producto escalar con $\mathbf{a}$, la función de scoring resultante produce rankings de atención que son independientes del nodo que consulta. En la metáfora de la red social, si un usuario es intrínsecamente “popular” (tiene un score alto), todos los nodos le prestan la máxima atención, independientemente de quiénes sean. La atención no es verdaderamente contextual.

GATv2 resuelve esta limitación invirtiendo el orden de operaciones, aplicando primero la no linealidad y después el producto escalar con el vector de atención:

$$e_{ij} = \mathbf{a}^T \text{LeakyReLU}(\Theta [\mathbf{h}_i \parallel \mathbf{h}_j]).$$

Este cambio aparentemente sutil tiene consecuencias teóricas profundas. Al aplicar la transformación lineal Θ sobre la concatenación completa $[\mathbf{h}_i \parallel \mathbf{h}_j]$ y luego la no linealidad, se introduce una interacción no lineal entre las representaciones de ambos nodos *antes* del cómputo de relevancia. Esto permite que la función de scoring pueda aprender rankings de atención arbitrarios que dependen tanto del contenido del nodo consultor como del vecino, modelando así una *atención dinámica*.

La normalización y agregación proceden de forma idéntica:

$$\alpha_{ij} = \text{softmax}_j(e_{ij}), \quad \mathbf{h}'_i = \varsigma \left(\sum_{j \in \mathcal{N}(i) \cup \{i\}} \alpha_{ij} \Theta_V \mathbf{h}_j \right).$$

Sin embargo, utilizar un único mecanismo de atención presenta una limitación práctica. Se obliga a la red a comprimir todas las posibles dimensiones de relación entre dos nodos en un solo número (α_{ij}), lo que puede promediar y diluir señales contradictorias. Para superar este

efecto, se introduce el concepto de múltiples cabezas de atención (*multi-head*). Al operar en paralelo, cada cabeza puede aprender de forma independiente un patrón de atención diferente, permitiendo al modelo capturar simultáneamente distintos tipos de relaciones entre los nodos.

Con H cabezas de atención paralelas (*multi-head*), la salida se obtiene por promediado:

$$\mathbf{h}'_i = \varsigma \left(\frac{1}{H} \sum_{h=1}^H \sum_{j \in \mathcal{N}(i) \cup \{i\}} \alpha_{ij}^{(h)} \Theta_V^{(h)} \mathbf{h}_j \right),$$

manteniendo constante la dimensión latente d y evitando el crecimiento innecesario de parámetros. Cada cabeza aprende un criterio de atención diferente: una cabeza puede atender a la actividad del vecino, otra al recurso, otra al patrón temporal.

En el contexto de *Process Mining*, la atención dinámica resulta especialmente ventajosa. Considérese un grafo secuencial donde cada evento es un nodo: la relevancia del predecesor inmediato puede variar drásticamente según el evento actual. Una actividad de “Validar documentación” puede atender fuertemente al evento de “Enviar oferta” que la precede (necesita saber qué oferta valida), mientras que una actividad administrativa rutinaria como “Registrar hora” puede no requerir contexto del predecesor. GATv2 modela esta variabilidad de forma nativa.

2.3.5. Inyección de atributos de arista

Cuando las aristas del grafo portan información, como el tiempo de transición entre eventos consecutivos, es necesario incorporar estos atributos en el cómputo de atención. En la metáfora de la red social, esto equivale a considerar no solo quién es tu amigo, sino cuánto tiempo hace que sois amigos o con qué frecuencia os comunicáis. La extensión natural de GATv2 con atributos de arista $\mathbf{a}_{ij} \in \mathbb{R}^{D_e}$ modifica la función de scoring:

$$e_{ij} = \mathbf{a}^T \text{LeakyReLU}(\Theta_s \mathbf{h}_i + \Theta_t \mathbf{h}_j + \Theta_e \mathbf{a}_{ij}), \quad (2.2)$$

donde $\Theta_s, \Theta_t, \Theta_e$ son matrices de pesos que proyectan respectivamente el nodo fuente, el nodo destino y el atributo de arista a un espacio común donde se computa la relevancia.

En *Process Mining*, un posible atributo de arista es el *Z-score* del tiempo de transición entre eventos consecutivos. Dada la arista (v_i, v_{i+1}) con diferencia temporal $\Delta t_i = \pi_{\text{time}}(e_{i+1}) - \pi_{\text{time}}(e_i)$, se considera la normalización por el par de actividades:

$$z_i = \frac{\Delta t_i - \mu_{(a,b)}}{\sigma_{(a,b)}},$$

donde $\mu_{(a,b)}$ y $\sigma_{(a,b)}$ son la media y desviación típica históricas para la transición desde la actividad a hasta b . En la práctica, cuando $\sigma_{(a,b)} = 0$ (por ejemplo, si todas las observaciones históricas presentan el mismo tiempo de transición), la normalización anterior deja de estar definida. Para abordar esta situación, pueden emplearse distintas estrategias, como añadir una pequeña constante $\varepsilon > 0$ al denominador o utilizar una estimación de la desviación típica en base a otro criterio.

El atributo resultante cuantifica cuánto se desvía temporalmente cada transición respecto del comportamiento habitual de ese par específico de actividades. Una transición “Enviar oferta $\rightarrow$ Validar documentación” que habitualmente toma 2 horas pero en este caso tomó 15 minutos producirá un Z-score fuertemente negativo. De este modo, se dota a la red neuronal de un contexto temporal estandarizado que indica si el ritmo de ejecución es el esperado o si presenta una desviación significativa, facilitando su procesamiento en las capas posteriores sin depender de magnitudes temporales absolutas.

2.3.6. Puente computacional: PyTorch y PyTorch Geometric

La materialización computacional de las arquitecturas de grafos descritas se realiza mediante el ecosistema **PyTorch** y su extensión **PyTorch Geometric** (PyG) (Fey and Lenssen, 2019). PyTorch proporciona el motor de tensores y diferenciación automática que permite la optimización por gradiente de los parámetros de la red. PyG extiende este marco con abstracciones específicas para grafos: la clase **MessagePassing** materializa directamente el formalismo de la Definición 2.3.1, ofreciendo métodos `message()`, `aggregate()` y `update()` que corresponden a las tres operaciones del marco.

La representación de la conectividad en PyG se codifica mediante un tensor `edge_index` $\in \mathbb{N}^{2 \times E}$, donde E es el número de aristas del grafo. Bajo la convención `source_to_target` (flujo por defecto), la primera fila contiene los nodos origen y la segunda los nodos destino:

$$\text{edge_index} = \begin{bmatrix} j_1 & j_2 & \cdots & j_E \\ i_1 & i_2 & \cdots & i_E \end{bmatrix},$$

de modo que la e -ésima columna codifica la arista dirigida $j_e \rightarrow i_e$. El vecindario de agregación del nodo i se recupera como $\mathcal{N}(i) = \{j : \exists e \text{ con } (j_e, i_e) = (j, i)\}$. En un registro de eventos representado como grafo secuencial, si los nodos corresponden a eventos ordenados temporalmente dentro de un caso, entonces `edge_index` materializa la relación de precedencia directa:

$$\text{edge_index} = \begin{bmatrix} 0 & 1 & \cdots & n-2 \\ 1 & 2 & \cdots & n-1 \end{bmatrix} \in \mathbb{N}^{2 \times (n-1)},$$

representando las $n-1$ transiciones consecutivas de la traza. Los atributos de arista se almacenan en un tensor `edge_attr` $\in \mathbb{R}^{E \times D_e}$ alineado columna a columna con `edge_index`, de forma que `edge_attr[e]` corresponde a la arista $j_e \rightarrow i_e$. En nuestro caso, $D_e = 1$ y el atributo será el Z-score temporal.

La implementación de GATv2 se encuentra disponible como `GATv2Conv`, que acepta directamente matrices de características nodales, `edge_index` y `edge_attr`, encapsulando toda la mecánica de propagación descrita. La estructura de datos `Data` permite representar cada caso del registro de eventos como un grafo individual, agrupando `(X, edge_index, edge_attr)` junto con targets y metadatos auxiliares. El `DataLoader` de PyG gestiona automáticamente el agrupamiento (*batching*) de grafos de diferente tamaño mediante reindexación de nodos, lo

que permite procesar múltiples casos simultáneamente en una sola pasada de GPU (PyTorch Geometric Team, 2026).

2.4. Arquitecturas basadas en atención y Transformers

Consideremos ahora el problema del monitoreo predictivo sobre prefijos en curso de un proceso puramente secuencial. Para ello, buscamos obtener una representación del caso completo, atendiendo a sus eventos. En este contexto, las redes neuronales de grafos (GNN) presentan limitaciones para capturar dependencias a largo plazo, ya que la propagación de información entre eventos distantes requiere múltiples saltos a través de la topología. Para superar esta barrera, la arquitectura Transformer ofrece un mecanismo de atención global que permite a cada evento del prefijo contextualizar su significado consultando directamente a todos los demás eventos históricos, capturando dependencias a largo plazo en un solo paso computacional. En esta sección se introduce primero la analogía con el procesamiento de lenguaje natural que motivó su creación; después se formalizan los mecanismos de autoatención (*self-attention*) y atención múltiple (*multi-head attention*); y, finalmente, se describe su adaptación específica al dominio del *Process Mining* mediante la codificación de prefijos, técnicas de enmascaramiento (*masking*) y agrupamiento indexado (*pooling*).

2.4.1. Del lenguaje natural a los procesos

La arquitectura Transformer fue propuesta originalmente por Vaswani et al. (2017) para el procesamiento de lenguaje natural (NLP), donde la tarea fundamental es modelar secuencias de palabras. En NLP, una frase como “El banco aprobó el préstamo porque la solicitud cumplía los requisitos” es una secuencia de tokens (palabras), y el significado de cada palabra depende del contexto: “banco” puede significar una entidad financiera o un asiento, y solo el contexto de la frase lo desambigua. El Transformer resuelve esto permitiendo que cada palabra “mire” directamente a todas las demás palabras de la frase para decidir cuáles son relevantes para su interpretación.

La analogía con *Process Mining* es directa:

Concepto NLP	Concepto PM
Token/Palabra	Evento individual
Frase/Oración	Traza (caso)
<i>embedding</i> de palabra	<i>embedding</i> de evento
Posición en la frase	Posición en la traza

Al igual que una palabra aislada es ambigua sin contexto, un evento atómico resulta insufi-

ciente para la inferencia predictiva. Por ejemplo, conocer que se ha ejecutado una determinada tarea de verificación o un hito de aprobación carece de valor operativo si se ignora qué actividades la precedieron, cuánto tiempo transcurrió en las transiciones previas o qué recursos específicos intervinieron en el flujo. La arquitectura Transformer resuelve esta limitación al permitir que cada evento atienda dinámicamente a todos los estados anteriores del prefijo, logrando contextualizar su verdadero significado procesal en función de la ruta exacta recorrida por el caso. Como veremos a continuación, el concepto de atención en Transformers es mucho más complejo que el de GAT/GATv2, donde eran simples coeficientes entrenables.

2.4.2. Autoatención escalada por producto escalar

El mecanismo de autoatención (*self-attention*) permite que cada posición de una secuencia atienda directamente a todas las demás posiciones, sin intermediación recurrente. Sea $(\mathbf{x}_1, \dots, \mathbf{x}_n)$ una secuencia de entrada con $\mathbf{x}_t \in \mathbb{R}^{d_{model}}$, organizada como una matriz $X \in \mathbb{R}^{n \times d_{model}}$ donde cada fila es un evento. Se definen tres transformaciones lineales que producen tres “roles” diferentes para cada posición:

$$Q = XW^Q \in \mathbb{R}^{n \times d_k}, \quad K = XW^K \in \mathbb{R}^{n \times d_k}, \quad V = XW^V \in \mathbb{R}^{n \times d_v},$$

con $W^Q, W^K \in \mathbb{R}^{d_{model} \times d_k}$ y $W^V \in \mathbb{R}^{d_{model} \times d_v}$. Estas matrices de pesos constituyen los parámetros entrenables del mecanismo.

La intuición es la siguiente. Pensemos en una biblioteca:

- Q (*queries*) representa “qué estoy buscando”: cada evento formula una pregunta sobre qué información del contexto le sería útil.
- K (*keys*) representa “qué ofrezco”: cada evento anuncia qué tipo de información contiene.
- V (*values*) representa “qué transmito”: es la información que efectivamente se transmite si la consulta tiene éxito.

La función de atención escalada por producto escalar (*Scaled Dot-Product Attention*) se define como:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V.$$

Desglosemos esta operación paso a paso:

1. $QK^T \in \mathbb{R}^{n \times n}$: el producto matricial genera una *matriz de afinidades* donde la entrada (i, j) mide cuánto la “pregunta” de la posición i coincide con la “oferta” de la posición j . Valores altos indican alta relevancia.
2. $\frac{QK^T}{\sqrt{d_k}}$: el factor de escala $\sqrt{d_k}$ previene que los productos escalares crezcan excesivamente con la dimensión. Sin este factor, asumiendo el escenario teórico ideal donde las componentes son variables aleatorias independientes con media cero y varianza unitaria (condiciones que se fuerzan mediante técnicas de inicialización), la varianza del producto

escalar crecería linealmente como d_k . Aunque empíricamente estas distribuciones fluctúan durante el entrenamiento, el factor de escala sigue siendo crítico para desacoplar la dispersión de los logits respecto a la dimensión d_k , evitando así la saturación de la función *softmax* y la consiguiente desaparición de los gradientes. Con la escala teórica:

$$\text{Var}\left(\frac{q_i^T k_j}{\sqrt{d_k}}\right) = \frac{d_k}{d_k} = 1,$$

manteniendo los gradientes en un rango estable.

3. *softmax*($\cdot$): aplicada por filas, convierte cada fila de afinidades en una distribución de probabilidad sobre las posiciones. El resultado es una *matriz de pesos de atención* $\alpha_{ij} \geq 0$ con $\sum_j \alpha_{ij} = 1$.
4. Multiplicación por V : la salida para la posición i es una media ponderada de los valores de todas las posiciones, donde los pesos son los coeficientes de atención:

$$\text{out}_i = \sum_{j=1}^n \alpha_{ij} v_j \in \mathbb{R}^{d_v}.$$

Las posiciones más relevantes contribuyen más a la representación contextualizada.

En Process Mining, esto significa que al procesar el evento 5º de un caso para redefinirse contextualmente, el mecanismo de atención puede asignar alta atención al evento 2º y baja atención al evento 3º (una actividad rutinaria), independientemente de la distancia temporal entre ellos. Siguiendo la analogía del NLP, un sustantivo se redefine en base a su adjetivo aunque haya palabras sin aporte semántico de por medio. Esta capacidad de capturar dependencias a largo alcance de forma eficiente es una de las principales ventajas del mecanismo de autoatención frente a las redes recurrentes. Aunque las arquitecturas LSTM mitigan el problema del desvanecimiento del gradiente, la propagación secuencial de la información sigue dificultando el aprendizaje de dependencias muy distantes (Bengio et al., 1994; Vaswani et al., 2017).

2.4.3. *Multi-Head Attention*

Para capturar distintos tipos de relaciones simultáneamente, Vaswani et al. (2017) proponen la *Multi-Head Attention*. La idea es que una sola función de atención puede modelar un solo tipo de relación (por ejemplo, “qué actividad precede”), pero un proceso tiene múltiples dimensiones relevantes (actividad, recurso, temporalidad, importe). Se definen H cabezas paralelas, cada una con proyecciones propias:

$$\text{head}_h = \text{Attention}(XW_h^Q, XW_h^K, XW_h^V), \quad h = 1, \dots, H, \quad (2.3)$$

con $W_h^Q, W_h^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_h^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ y típicamente $d_k = d_v = d_{\text{model}}/H$, de modo que el coste total es comparable al de una sola cabeza con la dimensión completa. Las cabezas se concatenan y se proyectan:

$$\text{MultiHead}(X) = \text{Concat}(\text{head}_1, \dots, \text{head}_H) W^O,$$

con $W^O \in \mathbb{R}^{H d_v \times d_{model}}$. Cada cabeza puede especializarse en un aspecto diferente: una cabeza puede aprender relaciones de flujo de control (qué actividades tienden a seguirse), otra relaciones temporales (qué eventos están temporalmente próximos), otra relaciones de recurso (qué eventos comparten recurso).

2.4.4. El bloque Transformer Encoder

Un bloque *Transformer Encoder* combina multi-head attention con una red feed-forward posicional, conexiones residuales y Layer Normalization:

$$\tilde{H} = \text{LN}(X + \text{MultiHead}(X)), \quad (\text{Sub-capas de Atención})$$

$$H' = \text{LN}(\tilde{H} + \text{FFN}(\tilde{H})), \quad (\text{Sub-capas Feed-Forward})$$

donde $\text{FFN}(\mathbf{x}) = \text{GELU}(\mathbf{x}W_1 + b_1)W_2 + b_2$ es una red de dos capas con activación GELU.

Cada componente de este bloque cumple un papel preciso:

- La **Multi-Head Attention** permite la comunicación entre posiciones, cada evento “consulta” al resto para contextualizarse y agrega la información que le resulta útil.
- La **FFN** introduce capacidad de modelado no lineal, procesando la información recién agregada de cada posición de forma estrictamente independiente, aplicando transformaciones profundas para extraer reglas lógicas complejas y enriquecer la representación latente del evento.
- Las **conexiones residuales** ($+X$ y $+\tilde{H}$) permiten el flujo directo del gradiente a través de las capas, facilitando el entrenamiento de redes profundas al crear atajos que evitan la degradación de la señal.
- La **Layer Normalization** estabiliza las activaciones entre capas, como se explicó en la Sección 2.2.

El apilamiento de L_T bloques produce un Transformer Encoder de profundidad L_T , capaz de capturar dependencias a múltiples escalas de abstracción (Vaswani et al., 2017). Es importante señalar que la arquitectura utilizada en este trabajo es exclusivamente el *Encoder* del Transformer, con atención bidireccional sobre toda la secuencia de entrada. Existen también arquitecturas *decoder-only* (como GPT), que emplean atención causal (cada posición solo puede atender a posiciones anteriores) para generación autorregresiva; sin embargo, para las tareas predictivas sobre prefijos abordadas aquí, la atención bidireccional del encoder resulta más adecuada al permitir que cada posición del prefijo se contextualice con toda la información disponible.

2.4.5. Codificación posicional

El mecanismo de self-attention es inherentemente invariante a permutaciones, la operación $\text{softmax}(QK^T/\sqrt{d_k})V$ no depende del orden en que se presenten las filas de X . Si reordenamos

los eventos de un caso, la salida de la atención sería la misma (salvo la reordenación correspondiente). Sin embargo, en *Process Mining*, el orden temporal de los eventos es fundamental: “Crear solicitud → Aprobar → Enviar” es un caso completamente diferente de “Enviar → Crear solicitud → Aprobar”.

Para inyectar información posicional, se suma a la entrada un vector de codificación posicional. Aunque existen diversas estrategias en la literatura para generar estos tensores, como el uso de matrices de parámetros aprendibles (*learnable* embeddings) que se optimizan durante el entrenamiento, la arquitectura predictiva desarrollada en este trabajo emplea la estrategia original del Transformer basada en funciones armónicas fijas.

Definición 2.4.1 (Codificación sinusoidal fija). Propuesta por Vaswani et al. (2017), define las componentes del vector posicional mediante funciones armónicas de frecuencia decreciente:

$$\begin{aligned} \text{PE}(i, 2k) &= \sin\left(\frac{i}{10000^{2k/d_{\text{model}}}}\right), \\ \text{PE}(i, 2k+1) &= \cos\left(\frac{i}{10000^{2k/d_{\text{model}}}}\right), \end{aligned}$$

con $k = 0, \dots, d_{\text{model}}/2 - 1$.

La intuición es que cada dimensión del vector posicional oscila a una frecuencia diferente: las primeras dimensiones oscilan rápidamente (distinguen posiciones cercanas) y las últimas oscilan lentamente (capturan la posición global). La combinación de senos y cosenos permite representar desplazamientos relativos como transformaciones lineales, lo que facilita al modelo aprender a atender “al evento de hace 3 posiciones” sin importar la posición absoluta en caso de que fuese útil. Esta codificación no añade parámetros entrenables y generaliza naturalmente a longitudes de secuencia no vistas durante el entrenamiento.

La inyección posicional se realiza mediante suma:

$$\mathbf{x}'_i = \mathbf{x}_i + \text{PE}_{(i)} \in \mathbb{R}^{d_{\text{model}}},$$

donde $\text{PE}_{(i)}$ denota el vector de codificación completo para la posición i , siguiendo el estándar de Vaswani et al. (2017).

2.4.6. Adaptación a *Process Mining*: prefijos y *pooling* indexado

En *Predictive Process Monitoring*, el modelo opera sobre *prefijos* de trazas: secuencias parciales de casos aún en ejecución. Dada una traza enriquecida completa $\sigma^E = \langle e_1, \dots, e_n \rangle$, se generan n prefijos truncados:

$$\sigma_c^E \upharpoonright_t = \langle e_1, \dots, e_t \rangle, \quad t = 1, \dots, n,$$

cada uno asociado a sus respectivos objetivos de predicción (siguiente actividad, tiempo restante, resultado final, clasificación).

Manejo de longitudes variables: padding y masking. Los prefijos de longitud heterogénea dentro de un mismo batch se rellenan (*padding*) hasta la longitud máxima del batch $n_{\text{máx}}$. Para que las posiciones de relleno no contaminen el cómputo de atención, se define una máscara binaria $M \in \{0, 1\}^{B \times n_{\text{máx}}}$ (con B el tamaño del batch) donde:

$$M_{b,j} = \begin{cases} 1, & \text{si } j \leq t_b \text{ (posición real),} \\ 0, & \text{si } j > t_b \text{ (padding),} \end{cases}$$

siendo t_b la longitud efectiva del prefijo b -ésimo. Las posiciones con $M_{b,j} = 0$ se excluyen del cálculo de softmax asignándoles afinidad $-\infty$ antes de la normalización, lo que equivale a anular su contribución ($\exp(-\infty) = 0$), por lo que las posiciones de padding no reciben ni emiten atención.

Pooling indexado. Tras el procesamiento por L_T bloques encoder, se extrae la representación del caso mediante un mecanismo de *pooling indexado*. Se emplea directamente el vector latente correspondiente al último evento real del prefijo:

$$\mathbf{z}_{last} = \mathbf{h}_t^{(L_T)} \in \mathbb{R}^{d_{model}},$$

donde t es la posición del último token activo (no-padding) de la secuencia. Debido a la naturaleza contextual del Transformer, en la que cada posición recibe información de toda la secuencia mediante autoatención, el vector $\mathbf{z}_{last}$ condensa el historial completo del caso hasta el instante t . Este método se conoce como *last token pooling*. Sin embargo, existen otras alternativas en la literatura como el *max pooling*, que extrae el valor máximo de cada dimensión a lo largo de toda la secuencia temporal, o el *mean pooling*, que calcula el promedio aritmético de las representaciones de todos los eventos del prefijo.

Alcanzado este punto, se consigue generar una representación latente $\mathbf{z}_{last}$ que condensa la dinámica del proceso en un único vector denso. Es fundamental recalcar que este vector no constituye la predicción final en sí misma, sino el mapa de características abstractas (*features*) que sintetiza el estado del caso. Para materializar las labores de predicción deseadas, este espacio latente alimenta en una etapa posterior a arquitecturas más simples y desacopladas denominadas cabezales de predicción (*prediction heads*). Es crucial destacar que todo este proceso se realiza bajo un enfoque de entrenamiento de extremo a extremo (*end-to-end*); esto implica que la señal de error combinada de todas las tareas predictivas fluye hacia atrás mediante retropropagación, optimizando los parámetros del Transformer. Como resultado, la representación latente $\mathbf{z}_{last}$ se aprende conjuntamente con el resto de parámetros del modelo, para codificar las características del proceso más relevantes para las tareas predictivas consideradas.

2.5. Autoencoders y errores de reconstrucción

Las secciones anteriores han presentado dos familias de arquitecturas, GNNs y Transformers, capaces de producir representaciones latentes ricas a partir de datos de procesos. Queda por formalizar cómo estas representaciones se emplean para las tareas objetivo: la detección

no supervisada de anomalías y la predicción multitarea. Ambos problemas comparten un elemento común, este es la necesidad de funciones de pérdida que cuantifiquen la discrepancia entre lo que el modelo produce y lo que debería producir. En esta sección se introduce primero el paradigma del autoencoder (que transforma la detección de anomalías en un problema de reconstrucción), después se formalizan las funciones de pérdida adaptadas a cada tipo de variable, y finalmente se describe cómo los errores de reconstrucción se agregan para producir scores de anomalía interpretables.

2.5.1. Definición formal del *Autoencoder*

Un autoencoder es una arquitectura compuesta por dos funciones parametrizadas —un encoder y un decoder— cuyo objetivo es aprender una representación de los datos de entrada que preserve la información esencial para su reconstrucción.

Definición 2.5.1 (Autoencoder). Sea $\mathbf{X} \in \mathbb{R}^{N \times F}$ la matriz de características de entrada. Un autoencoder se compone de:

$$\begin{aligned} \text{Encoder: } \mathbf{Z} &= f_{\phi}(\mathbf{X}) \in \mathbb{R}^{N \times d}, & (\text{Codificación}) \\ \text{Decoder: } \hat{\mathbf{X}} &= g_{\theta}(\mathbf{Z}) \in \mathbb{R}^{N \times F}, & (\text{Reconstrucción}) \end{aligned}$$

donde ϕ y θ son los parámetros entrenables del encoder y decoder respectivamente, d es la dimensión del espacio latente y $\hat{\mathbf{X}}$ es la reconstrucción de la entrada.

En la formulación clásica de autoencoders, se impone $d \ll F$ para crear una restricción de capacidad que fuerce al modelo a comprimir la información, descartando ruido y reteniendo solo las regularidades esenciales. Sin embargo, la reducción de dimensionalidad no es la única forma de generar ese efecto de cuello de botella sobre la información. Un enfoque alternativo, también adoptado en este trabajo, es el *feature masking*. Durante el entrenamiento, se oculta aleatoriamente una fracción de las características de entrada (asignándoles valor cero), obligando al encoder a reconstruir la señal completa a partir de información parcial. En este caso puede mantenerse $d \geq F$ o incluso $d \gg F$ (cuando los *embeddings* categóricos aprendidos expanden la dimensionalidad original), porque la dificultad de la tarea no proviene de la compresión dimensional sino de la corrupción estocástica de la entrada. El modelo debe aprender redundancias y correlaciones entre atributos para completar los datos faltantes, lo que fuerza representaciones latentes ricas sin necesidad de reducir la dimensión.

Independientemente del mecanismo de cuello de botella empleado, el principio de detección de anomalías es el mismo. Cuando el autoencoder se entrena exclusivamente sobre datos normales, aprende a reconstruir fielmente las instancias que siguen los patrones de normalidad. Ante una instancia anómala, una traza con comportamiento inusual, la red carece de los mecanismos para procesarla correctamente (nunca vio patrones similares durante el entrenamiento), produciendo errores de reconstrucción elevados que sirven como indicador de anomalía. Es decir, lo que el modelo no puede reconstruir bien, es probablemente anómalo.

Sin embargo, en aplicaciones prácticas de *Process Mining*, la suposición de disponer de un conjunto de entrenamiento estrictamente puro y libre de contaminación resulta idealista. En escenarios reales, se opera bajo un paradigma no supervisado donde se asume que la

gran mayoría de las trazas exhiben el comportamiento estándar del negocio, mientras que las anomalías se presentan como desviaciones infrecuentes. Es bajo esta premisa donde la presencia de un cuello de botella (*bottleneck*), ya sea mediante la reducción dimensional del espacio latente o mediante técnicas de enmascaramiento agresivo, se vuelve indispensable. Al limitar deliberadamente la capacidad del modelo para memorizar la totalidad de los datos de entrada, la red se ve obligada a optimizar su función de pérdida concentrando sus recursos en comprimir y reconstruir únicamente las dinámicas y correlaciones estadísticas más robustas, repetitivas y dominantes del proceso. Como consecuencia, las anomalías, al ser raras y dispersas, no tienen el peso suficiente en el gradiente global para justificar el uso de la limitada capacidad de almacenamiento de la red, siendo descartadas por el cuello de botella como si fuesen ruido de fondo. Esto garantiza que, incluso ante un *dataset* sutilmente contaminado, el autoencoder se especialice exclusivamente en la estructura dominante de la normalidad.

En el contexto de grafos de procesos, el encoder puede ser una GATv2 que produce $\mathbf{Z} = \text{GATv2}_\phi(\mathbf{X}, \mathbf{E}_{\text{attr}}, \text{edge_attr})$, integrando información topológica, nodal y de aristas en la representación latente. Esta arquitectura resulta idónea al fusionar la topología rígida del grafo con la flexibilidad de la atención. Esta sinergia faculta al modelo para detectar simultáneamente anomalías estructurales (conexiones o caminos inválidos en el grafo), lógicas (rupturas en la secuencia causal de las actividades) y de atributos (valores de recursos, tiempos o importes que resultan inconsistentes con el contexto local del evento).

2.5.2. Funciones de pérdida para la reconstrucción

La optimización del autoencoder requiere funciones de pérdida adecuadas a la naturaleza de las variables reconstruidas. Dado que un registro de eventos contiene atributos de naturaleza heterogénea (numéricas continuas, categóricas nominales, binarias) se emplea una pérdida multi-eje donde cada componente se adapta al dominio de la variable.

Mean Squared Error (MSE) para variables numéricas

Definición 2.5.2 (Mean Squared Error). Para la reconstrucción de atributos continuos se emplea el error cuadrático medio:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N} \sum_{i=1}^N \frac{1}{p} \sum_{j=1}^p (\hat{x}_{ij}^{\text{num}} - x_{ij}^{\text{num}})^2,$$

donde p es el número de variables numéricas.

El MSE penaliza cuadráticamente las desviaciones, esto otorga mayor peso a los errores grandes y proporciona gradientes proporcionales al error, favoreciendo una convergencia estable.

Cross-Entropy para variables categóricas

Definición 2.5.3 (Cross-Entropy categórica). Para la reconstrucción de atributos categóricos:

$$\mathcal{L}_{\text{CE}} = -\frac{1}{N} \sum_{i=1}^N \sum_{c=1}^C y_{ic} \log \hat{p}_{ic}, \quad (2.4)$$

donde $y_{ic} \in \{0, 1\}$ es el indicador de la clase verdadera, C es el número de categorías y $\hat{p}_{ic}$ es la probabilidad estimada por el modelo para la clase c .

La cross-entropy mide la divergencia entre la distribución empírica, toda la masa de probabilidad sobre la clase correcta, y la distribución predicha por el modelo. Es mínima ($= 0$) cuando el modelo concentra toda la probabilidad en la clase correcta, y crece cuando el modelo asigna masa de probabilidad a clases incorrectas. Desde la perspectiva de teoría de la información, minimizar la cross-entropy equivale a minimizar el número esperado de bits adicionales necesarios para codificar la realidad usando la distribución predicha como modelo.

Las probabilidades $\hat{p}_{ic}$ se obtienen aplicando la función softmax sobre los *logits* de salida del modelo. Los logits son los valores numéricos crudos, potencialmente negativos o de cualquier magnitud, generados por la última capa lineal de la red:

$$\hat{p}_{ic} = \text{softmax}(l_i)_c = \frac{\exp(l_{ic})}{\sum_{c'=1}^C \exp(l_{ic'})},$$

donde $l_i \in \mathbb{R}^C$ es el vector de logits para la instancia i . A mayor valor de un logit, mayor es la afinidad del modelo hacia esa categoría. La softmax garantiza que estos valores se transformen en una distribución de probabilidad válida.

Label smoothing. En tareas de clasificación predictiva, se emplea una variante regularizada denominada *label smoothing*, introducida por Szegedy et al. (2016) y posteriormente adoptada en el entrenamiento del Transformer original (Vaswani et al., 2017). En lugar de usar la distribución one-hot $y_{ic} \in \{0, 1\}$ como objetivo (que exige al modelo asignar probabilidad 1 a una clase y 0 al resto), se suaviza la distribución target:

$$y_{ic}^{\text{smooth}} = (1 - \varepsilon) y_{ic} + \frac{\varepsilon}{C}, \quad (2.5)$$

donde $\varepsilon \in [0, 1)$ es el factor de suavizado (típicamente $\varepsilon = 0.1$). Esta modificación impide que el modelo genere logits excesivamente grandes (que saturan gradientes) y previene la sobreconfianza en predicciones, fenómeno frecuente en registros de eventos donde la variabilidad intrínseca del proceso genera incertidumbre legítima: cuando una solicitud puede ser seguida tanto por “Validar” como por “Derivar”, forzar probabilidad 1 en una de ellas no refleja la realidad del proceso.

Huber Loss como alternativa al MSE

Definición 2.5.4 (Huber Loss). La pérdida de Huber combina las ventajas del MSE (sensibilidad a errores pequeños) con la robustez del error absoluto medio (MAE) frente a valores

extremos:

$$\mathcal{L}_{\text{Huber}}(\hat{y}, y) = \begin{cases} \frac{1}{2}(\hat{y} - y)^2, & \text{si } |\hat{y} - y| \leq \delta, \\ \delta \left(|\hat{y} - y| - \frac{\delta}{2} \right), & \text{si } |\hat{y} - y| > \delta, \end{cases}$$

donde $\delta > 0$ es un hiperparámetro que controla la transición entre comportamiento cuadrático y lineal.

Para errores menores que δ , Huber Loss se comporta como MSE (gradientes proporcionales al error); para errores mayores, crece linealmente como MAE (gradientes constantes). Esto resulta especialmente adecuado para la predicción de tiempos restantes en procesos empresariales, donde existen casos atípicamente largos, una secuencia que se demora meses por circunstancias excepcionales, cuyo error cuadrático dominaría la pérdida total bajo MSE, distorsionando el entrenamiento del modelo hacia la predicción de estos outliers.

2.5.3. Pérdida total multi-tarea

Tanto para el autoencoder de detección de anomalías como para el Transformer predictivo, la función de pérdida total es una combinación ponderada de las pérdidas individuales.

Definición 2.5.5 (Pérdida multi-tarea ponderada).

$$\mathcal{L}_{\text{total}} = \sum_{t=1}^T \mu_t \mathcal{L}_t, \quad \mu_t \geq 0, \quad (2.6)$$

donde $\mu_1, \dots, \mu_T$ son hiperparámetros que equilibran la contribución de cada tarea. Es habitual normalizarlos para satisfacer $\sum_{t=1}^T \mu_t = 1$, aunque esta condición no resulta estrictamente necesaria.

El entrenamiento multi-tarea para la predicción ofrece ventajas frente a modelos independientes (Caruana, 1997; Ruder, 2017). Las capas compartidas de la arquitectura aprenden representaciones más ricas al optimizar simultáneamente objetivos complementarios. Por ejemplo, la tarea de predecir la siguiente actividad y la de estimar el tiempo restante fuerzan al encoder a capturar tanto la dinámica de control de flujo como los patrones temporales, produciendo una representación latente más completa que la obtenida optimizando una sola tarea.

2.5.4. Del error de reconstrucción a la detección de anomalías

En el paradigma de autoencoders de grafos para la detección de anomalías, el modelo asume que las instancias normales dominan el conjunto de entrenamiento. Durante la fase de inferencia, la red intenta reconstruir cada nodo del grafo, el cual representa un evento individual, junto con el conjunto de sus atributos asociados y sus transiciones o aristas. El grado de desviación de un evento respecto a la normalidad aprendida se cuantifica mediante un vector de errores parciales de reconstrucción:

$$\mathbf{r}_i = \left(r_i^{\text{num}}, r_i^{\text{cat}}, r_i^{\text{edge}} \right) \in \mathbb{R}_{\geq 0}^3,$$

donde las componentes representan los errores desacoplados en las dimensiones numéricas, categóricas y de aristas, las cuales abarcan la dinámica temporal, de atributos y de flujo.

A la hora de explotar esta información para determinar si un nodo es anómalo, la arquitectura permite transitar con flexibilidad desde una evaluación holística hasta análisis sumamente específicos. Por un lado, las desviaciones pueden integrarse en un score global ponderado mediante la combinación lineal

$$s_i = w_{\text{num}} r_i^{\text{num}} + w_{\text{cat}} r_i^{\text{cat}} + w_{\text{edge}} r_i^{\text{edge}},$$

donde el vector de pesos $\mathbf{w} \geq \mathbf{0}$ se configura para obtener una métrica de riesgo única que prioriza los objetivos del negocio. Por otro lado, este mismo vector de errores puede someterse a un filtrado dimensional aislado, evaluando cada componente de forma independiente bajo umbrales específicos τ_{num} , τ_{cat} y τ_{edge} .

Para elevar el análisis desde el nivel de nodo hasta la escala macroscópica de caso, las señales locales calculadas se agregan utilizando dos operadores complementarios:

$$S_c^{\text{max}} = \max_i s_i, \quad S_c^{\text{mean}} = \frac{1}{n_c} \sum_{i=1}^{n_c} s_i.$$

La agregación mediante el máximo, S_c^{max} , está diseñada para aislar la presencia de nodos anómalos individuales. Basta con que un único evento crítico del grafo supere el umbral para que el caso entero sea marcado como sospechoso, una propiedad de gran utilidad para detectar fraudes o violaciones puntuales de políticas. Por el contrario, la agregación mediante la media, S_c^{mean} , evalúa la degradación del caso de forma global, por lo que resulta idónea para detectar ineficiencias sistémicas o una acumulación de sutiles retrasos temporales a lo largo de la traza.

Aunque estas herramientas nos dan pistas muy útiles para diagnosticar el problema, no son infalibles. El error tiende a propagarse entre las variables (*error smearing*), lo que significa que un fallo en un atributo puede arrastrar al resto y hacer que parezcan anómalos cuando en realidad están bien. Dado que el autoencoder procesa y comprime todas las variables del nodo a través de un espacio latente reducido y común para capturar sus correlaciones, una anomalía extrema en una sola dimensión, tal como un importe desproporcionado, puede desestabilizar la representación interna completa de ese evento.

Como consecuencia, es perfectamente factible que el sistema active un filtro de anomalía categórica, r_i^{cat} , o devuelva un máximo local alarmante en un nodo cuyos atributos categóricos o secuenciales sean intrínsecamente válidos. El fallo observado en una rama del decodificador puede ser simplemente el reflejo colateral de una distorsión severa en otra dimensión, por lo que tanto el score global como los filtros individuales deben interpretarse siempre como una guía de sospecha y nunca como una herramienta de atribución causal infalible.

Capítulo 3

Caso de Uso: Proceso de Solicitud de Préstamos

El objetivo de este capítulo es aplicar técnicas clásicas de Process Mining y modelos de *Deep Learning* al análisis de un proceso de solicitud de préstamos, tomando como referencia los conceptos y métodos presentados en los capítulos previos.

Durante la realización de las prácticas en SDG Group, se realizó una fase de familiarización con las herramientas estudiadas, evaluando su aplicabilidad en distintos escenarios.

Tras ello, se desarrolló un caso de uso completo que reproduce el ciclo de vida de un proyecto real de Ciencia de Datos, desde la comprensión del problema hasta la evaluación de los modelos, siguiendo la metodología CRISP-DM (*Cross-Industry Standard Process for Data Mining*). Debido al tiempo disponible, el estudio se centra principalmente en las arquitecturas basadas en aprendizaje profundo, utilizando las técnicas clásicas de *Process Mining* como apoyo para el análisis descriptivo y exploratorio.

A continuación, se describen las seis fases que servirán de guía para el desarrollo del caso de estudio:

1. **Conocimiento del Negocio:** definición de los objetivos del proyecto y de los requisitos desde la perspectiva de negocio.
2. **Conocimiento de los Datos:** exploración, comprensión y evaluación de la calidad de los datos disponibles.
3. **Preparación de Datos:** limpieza y transformación de los datos para su explotación.
4. **Modelado:** aplicación de modelos y calibración de parámetros.
5. **Evaluación:** validación de métricas y análisis de resultados.
6. **Entrega:** creación de un sistema para el consumo de la solución.

Aunque las fases se muestran de forma ordenada, el proceso de desarrollo no es estrictamente secuencial, ya que la metodología contempla iteraciones entre etapas en función de los hallazgos obtenidos y requerimientos del proyecto.

En este capítulo se describe cada uno de los puntos anteriores con el fin de ofrecer una comprensión global del proyecto. El volumen de código desarrollado durante este caso de estudio es considerable; sin embargo, su inclusión en esta memoria no se alinea con los objetivos del trabajo. La finalidad de este capítulo es ilustrar la aplicación de las técnicas presentadas en los capítulos anteriores sobre un caso real. Por ello, se hará especial énfasis en la descripción de la arquitectura de las soluciones propuestas, así como en el análisis de los resultados obtenidos y las conclusiones derivadas, evitando profundizar en los detalles de implementación. No obstante, el lector interesado en los *scripts* desarrollados puede consultar el repositorio asociado al proyecto que contiene las arquitecturas de aprendizaje profundo desarrolladas: TFM_Repo_ProcessMining (2026).

3.1. Planteamiento del problema

Las entidades financieras gestionan diariamente un elevado volumen de operaciones asociadas a la concesión de productos y servicios bancarios, tales como préstamos, hipotecas, tarjetas de crédito o aperturas de cuentas. En un sector tan regulado, las organizaciones generan grandes cantidades de información operacional que reflejan el detalle de la ejecución de sus actividades. El registro de datos se convierte en una fuente de alto potencial para la mejora continua y la toma de decisiones. Además, en un mundo que avanza hacia la digitalización y automatización de los procesos, la calidad de la información tiende a aumentar.

Entre los procesos más relevantes se encuentran aquellos relacionados con la evaluación y aprobación de solicitudes de financiación. Estos procesos involucran múltiples participantes, flujos y reglas de negocio, generando secuencias complejas de eventos. Su monitorización resulta clave para garantizar la eficiencia operativa de la compañía y la calidad del servicio recibido por el cliente.

El caso de uso implementado entra precisamente en este ámbito. Aunque los datos utilizados son reales y proceden de una entidad financiera existente, el contexto de negocio descrito ha sido construido con fines académicos. Con el objetivo de facilitar la comprensión del estudio, dicho escenario se expone como si se tratase de una situación real. A continuación se presenta el problema objeto de estudio.

Como caso de estudio se considera una entidad financiera holandesa dedicada a la concesión de préstamos personales. Existe un proceso subyacente que describe la secuencia de actividades necesarias para gestionar una solicitud de préstamo, desde su recepción hasta la emisión de una resolución final. Este proceso involucra distintas tareas, recursos y puntos de decisión, cuya ejecución queda registrada en los sistemas de información.

El banco ha trasladado una serie de fenómenos operativos que generan ineficiencias sistémicas y desea solucionar:

- **Variabilidad extrema de caminos:** cada caso es único, la ausencia de un flujo secuencial rígido de las tareas permite numerosas combinaciones de actividades. Por tanto, se genera una inmensa cantidad de variantes de ejecución. Este hecho, sumado a la cardinalidad de recursos, tiempos de ejecución y otros factores asociados a cada tarea, aumenta la complejidad del proceso e imposibilita su análisis descriptivo de forma manual.

- **Anomalías operativas:** la entidad ha identificado comportamientos puntuales que generan ineficiencias y tienen un impacto económico directo sobre el negocio. Entre ellos destacan cuellos de botella en determinadas actividades, repeticiones de tareas, bucles de ejecución y desviaciones respecto al comportamiento esperado del proceso. La detección de estas situaciones ayudaría a mejorar la eficiencia operativa.
- **Incertidumbre en la resolución:** algunas solicitudes permanecen abiertas durante periodos de tiempo superiores a los esperados debido a retrasos en determinadas etapas del proceso o a la falta de actuación oportuna. Como consecuencia, ciertos casos pueden experimentar degradaciones en la calidad del servicio, incumplimientos de acuerdos de nivel de servicio o incluso la pérdida de oportunidades de negocio derivada del abandono de la solicitud por parte del cliente.

Aunque estos problemas se manifiestan de formas distintas, podrían estar relacionados con la falta de visibilidad sobre el comportamiento real del proceso y en la dificultad para explotar la información contenida en los registros de eventos generados.

De este modo, se genera la necesidad de dotar a la organización de mecanismos que permitan comprender, monitorizar y anticipar el comportamiento del proceso de concesión de préstamos. Con este propósito, la entidad plantea como objetivo del proyecto el desarrollo de una solución integrada, implementada mediante un *dashboard* interactivo, capaz de integrar toda la información disponible y proporcionar apoyo a la toma de decisiones en los distintos niveles operativos.

Formalmente, el problema se puede dividir en tres tareas: caracterización de procesos, detección de anomalías y predicción de resultados. Estas tareas se pueden clasificar en los siguientes ámbitos complementarios de análisis:

1. **Análisis descriptivo:** proporcionar una visión detallada del comportamiento histórico del proceso mediante visualizaciones, métricas de impacto y diagramas que permitan comprender cómo se ejecutan realmente las solicitudes de préstamo, identificar variantes frecuentes y detectar oportunidades de mejora.
2. **Análisis proactivo:** identificar comportamientos anómalos o potencialmente problemáticos, tales como cuellos de botella excesivos, desviaciones respecto al flujo esperado o secuencias de actividades consideradas no deseables desde la perspectiva de negocio.
3. **Análisis predictivo:** realizar predicciones a partir de casos en curso de variables críticas del proceso, como el tiempo restante de la solicitud o resolución definitiva del préstamo, para anticipar posibles incidencias y facilitar una actuación temprana.

Los requisitos planteados presentan una estrecha relación con las capacidades ofrecidas por las técnicas estudiadas en los capítulos anteriores. La naturaleza del problema sugiere abordarlo mediante *Process Mining*. En particular, el conocimiento adquirido en capítulos anteriores permite anticipar el papel que desempeñará cada una de las herramientas en la resolución del problema; en principio, el análisis descriptivo podrá realizarse con algoritmos clásicos, mientras que se deberá recurrir a modelos más sofisticados para completar con éxito el análisis proactivo y predictivo.

Con esto, la primera de las fases del marco metodológico, *Conocimiento de Negocio*, queda cubierta en gran medida. Sin embargo, no se puede dar por finalizada, ya que podría ser necesario redefinir objetivos, realizar consultas adicionales con los responsables de negocio o surgir nuevos requisitos. El carácter iterativo de la metodología CRISP-DM contempla de forma natural este tipo de saltos entre fases.

Antes de abordar el desarrollo de las distintas líneas de análisis, resulta imprescindible comprender en profundidad la naturaleza de los datos disponibles. La calidad, estructura y riqueza de la información registrada condicionan la viabilidad de las técnicas a aplicar y el valor de los resultados obtenidos. Por este motivo, la siguiente fase del proyecto se centra en el estudio exhaustivo del conjunto de datos, analizando sus características, evaluando posibles limitaciones y verificando si es adecuado para dar respuesta a los objetivos definidos por la entidad financiera. Además, este análisis permitirá identificar los preprocesamientos necesarios y establecer las bases sobre las que se construirán las soluciones descriptivas, proactivas y predictivas desarrolladas posteriormente.

3.2. Conocimiento de los datos

La presente sección tiene como objetivo caracterizar el conjunto de datos que servirá de base para el desarrollo del proyecto. Para ello, se describirá el origen de la información utilizada, se analizará el significado de las variables y actividades registradas, se formalizará el registro de eventos desde la perspectiva del *Process Mining* y, finalmente, se realizará un análisis exploratorio orientado a comprender las principales características del proceso y de los casos observados.

3.2.1. Origen y Contexto del Conjunto de Datos

El conjunto de datos empleado en este proyecto corresponde al publicado en el marco del *BPI Challenge 2017* y es uno de los *datasets* de referencia más utilizados en la literatura de *Process Mining*. El *BPI Challenge 2017* es una iniciativa impulsada anualmente por la comunidad de *Process Mining* de la Universidad Tecnológica de Eindhoven (TU/e), cuyo objetivo es proporcionar registros de eventos reales procedentes de organizaciones de distintos sectores para fomentar la investigación, comparación y validación de nuevas técnicas sobre problemas empresariales reales (BPI Challenge 2017 Organizers, 2017).

Los datos fueron publicados a través del repositorio del *4TU Centre for Research Data* (van Dongen, 2017) y describen el proceso de concesión de préstamos personales de una entidad financiera holandesa. El registro recoge la ejecución real de miles de solicitudes de préstamo, incluyendo tanto las actividades realizadas durante la tramitación de cada caso como diversa información contextual asociada a los clientes y a las decisiones tomadas a lo largo del proceso.

La edición de 2017 planteaba originalmente diversas cuestiones de análisis relacionadas con los tiempos de ejecución de tareas, la influencia de solicitudes incompletas sobre la conversión de clientes y el comportamiento de aquellos solicitantes que reciben múltiples ofertas. Sin embargo, el presente trabajo no pretende reproducir las tareas propuestas en el desafío original. En su lugar, se utilizará el conjunto de datos como base para desarrollar una solución integral

orientada al análisis descriptivo, proactivo y predictivo del proceso de concesión de préstamos, alineada con los objetivos de negocio definidos en la Sección 3.1.

La organización responsable de los datos ya había participado previamente en el *BPI Challenge 2012*, también ampliamente estudiado en el ámbito de investigación en *Process Mining*. Mientras que el desafío de 2012 proporcionaba una representación más simplificada del proceso de gestión de solicitudes, la edición de 2017 amplía significativamente el nivel de detalle disponible, incorporando información adicional sobre ofertas, estados intermedios de cada tarea, recursos involucrados y atributos de carácter económico. Esta mayor riqueza informativa ha convertido al BPI 2017 en un escenario especialmente atractivo para el desarrollo de técnicas de análisis descriptivo, detección de anomalías y monitorización predictiva.

Entre los trabajos desarrollados en el contexto del propio desafío pueden destacarse los realizados por equipos académicos y profesionales, como los presentados por Blevi et al. (2017) y Jeong et al. (2017), donde se exploran diferentes estrategias para descubrir conocimiento de negocio a partir de los registros de eventos y comprender los factores que influyen en el comportamiento del proceso de concesión de préstamos. Estos estudios constituyen ejemplos representativos del tipo de análisis que pueden desarrollarse sobre el conjunto de datos y han servido como apoyo durante las etapas iniciales de exploración y comprensión de los datos.

El interés de la comunidad investigadora por este registro de eventos se debe, en gran medida, a que reúne muchas de las características presentes en procesos empresariales reales: elevada variabilidad de ejecución, múltiples puntos de decisión, participación de numerosos recursos, información temporal detallada y una gran cantidad de atributos complementarios asociados a cada caso. Estas propiedades lo convierten en un banco de pruebas idóneo para evaluar técnicas avanzadas de *Process Mining* y justifican su elección como caso de estudio en este trabajo.

Siguiendo la narrativa introducida en la sección anterior, los datos serán tratados como si correspondiesen a la entidad financiera descrita en la Sección 3.1, responsable de plantear el problema de negocio considerado. No obstante, conviene recordar que tanto el escenario planteado como los objetivos específicos del proyecto constituyen una simulación construida con fines académicos, mientras que los registros utilizados proceden de un conjunto de datos público y real.

3.2.2. Diccionario de Datos

En su formato original, el *dataset* empleado se presenta en formato tabular y contiene 1 202 267 registros y 19 variables. El nivel mínimo de granularidad corresponde a un evento individual registrado durante la ejecución del proceso de concesión de préstamos. Estos eventos se agrupan mediante un identificador común de caso, lo que permite reconstruir cronológicamente la secuencia completa asociada a cada solicitud.

A continuación, se listan las 19 variables junto a una breve explicación de la información que recogen:

Variable	Descripción	Tipo
<code>case:concept:name</code>	Identificador único de la solicitud de préstamo.	ID
<code>concept:name</code>	Actividad ejecutada dentro del proceso.	Categórica
<code>time:timestamp</code>	Marca temporal asociada al evento.	Fecha
<code>org:resource</code>	Recurso responsable de la ejecución de la actividad.	Categórica
<code>lifecycle:transition</code>	Estado del evento dentro de su ciclo de vida.	Categórica
<code>Action</code>	Acción específica registrada por el sistema para el evento.	Categórica
<code>EventID</code>	Identificador único del evento registrado.	ID
<code>EventOrigin</code>	Segmentación del tipo de actividad.	Categórica
<code>case:LoanGoal</code>	Finalidad declarada por el cliente para la solicitud del préstamo.	Categórica
<code>case:ApplicationType</code>	Tipo de solicitud realizada.	Categórica
<code>case:RequestedAmount</code>	Importe solicitado por el cliente.	Numérica
<code>OfferID</code>	Identificador único de la oferta generada.	ID
<code>OfferedAmount</code>	Importe ofrecido por la entidad financiera.	Numérica
<code>FirstWithdrawalAmount</code>	Importe inicialmente dispuesto por el cliente.	Numérica
<code>NumberOfTerms</code>	Número de cuotas o plazos asociados a la oferta.	Numérica
<code>MonthlyCost</code>	Cuota mensual asociada a la oferta.	Numérica
<code>Selected</code>	Indica si la oferta fue seleccionada por el cliente.	Binaria
<code>Accepted</code>	Indica si la oferta es apta para ser aceptada.	Binaria
<code>CreditScore</code>	Puntuación crediticia asociada al solicitante.	Numérica

Tabla 3.1: Variables disponibles en el conjunto de datos.

Véase que no todas las variables corresponden a atributos del evento, unidad mínima de información registrada en el proceso. Algunas variables se encuentran definidas a nivel de caso, por lo que mantienen un valor invariante para todos los eventos pertenecientes a una misma solicitud de préstamo. Otras, en cambio, están asociadas a las ofertas generadas durante la tramitación, permaneciendo constantes para un mismo identificador de oferta.

Además, los campos presentes abarcan distintas tipologías de datos, incluyendo identificadores y variables categóricas, numéricas y binarias. Aunque la Tabla 3.1 resume las variables disponibles en el conjunto de datos, profundizaremos sobre aquellas que desempeñan un papel especialmente relevante para la comprensión del proceso. Si bien todas las variables aportan información potencialmente útil, su relevancia en este trabajo varía según la tarea y se detallarán las empleadas en cada caso.

La variable `concept:name` es categórica y representa el nombre de la actividad ejecutada dentro del proceso. Conocer la lista de actividades es fundamental para familiarizarse con las etapas por las que puede pasar una solicitud de préstamo, desde su creación hasta su resolución. Siguen una nomenclatura de prefijos estructurada en tres grandes grupos:

- **A_**: actividades relacionadas con el estado de la solicitud (*Application*).
- **O_**: actividades asociadas a la creación, envío y gestión de ofertas (*Offer*).
- **W_**: tareas operativas realizadas por empleados de la entidad (*Workflow*).

Existe una correspondencia exacta entre estos prefijos y el campo `EventOrigin`, que toma precisamente uno de los siguientes valores: $\{Application, Offer, Workflow\}$. La siguiente lista recoge los distintos estados por los que puede transitar una solicitud de préstamo a lo largo de su ciclo de vida.

Actividad	Descripción
A_Create Application	Creación de una nueva solicitud de préstamo.
A_Submitted	La solicitud ha sido enviada por el cliente para iniciar su tramitación.
A_Concept	Estado inicial tras una primera evaluación automática.
A_Accepted	La solicitud supera la evaluación inicial y puede recibir ofertas.
A_Complete	Se han enviado ofertas y se espera la documentación del cliente.
A_Validating	La documentación recibida está siendo revisada.
A_Incomplete	La solicitud requiere documentación o información adicional.
A_Pending	La solicitud es aceptada y el préstamo se concede.
A_Denied	La solicitud es rechazada por la entidad financiera.
A_Cancelled	La solicitud es cancelada por parte del cliente.

Tabla 3.2: Estados de la solicitud (A_)

Una vez que una solicitud supera la evaluación inicial y resulta apta para recibir propuestas por parte de la entidad financiera, aparecen actividades asociadas al ciclo de vida de las ofertas. Estas actividades registran la creación, envío, devolución, aceptación o cancelación de las distintas propuestas realizadas al cliente. Cabe destacar que una misma solicitud puede generar varias ofertas con importes, plazos o condiciones diferentes, lo que permite analizar

estrategias de negociación y estudiar el comportamiento de clientes que reciben múltiples propuestas.

Actividad	Descripción
O_Create Offer	Creación de una nueva oferta.
O_Created	La oferta ha sido generada correctamente.
O_Sent (mail and online)	La oferta es enviada al cliente mediante correo postal y/o canales digitales.
O_Sent (online only)	La oferta es enviada exclusivamente por medios digitales.
O_Returned	El cliente devuelve la oferta firmada junto con la documentación requerida.
O_Accepted	La oferta es aceptada por el cliente.
O_Refused	La oferta es rechazada por el cliente.
O_Cancelled	La oferta es cancelada o deja de estar vigente.

Tabla 3.3: Estados asociados a las ofertas (O_)

Por último, a lo largo de todo el proceso, los empleados de la entidad financiera ejecutan diversas tareas operativas. La principal particularidad de estas actividades es su relación con la variable categórica `lifecycle:transition`. Mientras que en las actividades con prefijo A_ y O_, este campo siempre toma el valor *complete*; en las actividades operativas (W_) se registran diferentes estados del ciclo de vida de una tarea, tales como *complete* (completada), *suspend* (suspendida), *schedule* (programada), *start* (empezada) o *resume* (reanudada). Esta característica da la opción de realizar análisis temporales más complejos, aunque se sale de los objetivos establecidos.

Actividad	Descripción
W_Complete application	Actividad orientada a completar la información de una solicitud.
W_Validate application	Revisión y validación de la solicitud por parte de un empleado.
W_Call incomplete files	Contacto con clientes que presentan documentación incompleta.
W_Call after offers	Seguimiento realizado tras el envío de ofertas.
W_Handle leads	Gestión inicial de potenciales clientes o solicitudes entrantes.
W_Assess potential fraud	Análisis orientado a detectar posibles indicios de fraude.
W_Shortened completion	Procedimiento abreviado para completar una solicitud.
W_Personal Loan collection	Actividad residual relacionada con la gestión de préstamos personales.

Tabla 3.4: Tareas operativas realizadas por empleados (W_)

A partir de las descripciones anteriores, se puede apreciar la lógica general del proceso. Una solicitud de préstamo comienza con la creación y registro de la petición del cliente, avanzando hacia una evaluación inicial para determinar si la entidad puede formular una oferta. En caso afirmativo, la oferta con ciertas condiciones reflejadas en algunas columnas del *dataset* se envía al solicitante, existiendo la posibilidad de que rechace dicha oferta y el banco proponga nuevas condiciones. Finalmente, tras las correspondientes validaciones, el proceso concluye en un de tres estados posibles:

- Concedido (*Granted*): la solicitud se aprueba y el préstamo se concede, corresponde al registro del evento **A_Pending**.
- Denegado (*Denied*): el banco rechaza la solicitud, queda marcado por el evento **A_Denied**.
- Cancelado: el proceso se cancela debido al cliente, se produce cuando se registra el evento **A_Cancelled**.

3.2.3. Formalización del registro de eventos

Aunque el conjunto de datos original se distribuye en formato tabular, las técnicas de *Process Mining* requieren interpretar dicha información como un registro de eventos. Esta transformación permite reconstruir las secuencias ordenadas temporalmente de actividades asociadas a cada caso.

Recordemos que los tres componentes imprescindibles de un registro de eventos son el identificador del caso, la actividad y el instante temporal; permitiendo definir el evento i -ésimo de un caso como $e_i = (c, a_i, t_i)$. Puesto que se dispone de variables adicionales V_i como el recurso, el importe solicitado o el motivo del préstamo, un evento del proceso estudiado puede representarse como

$$e_i = (c, a_i, t_i, V_i).$$

Conviene recalcar que no todos estos atributos se encuentran definidos al mismo nivel de granularidad. Como consecuencia, determinados campos pueden aparecer vacíos en parte de los registros o no reflejar información específica del instante temporal en el que se produce el evento, sino características persistentes del caso o de la oferta a la que pertenecen.

Obsérvese que la representación anterior mantiene una correspondencia exacta con la estructura formalizada en la ecuación (1.1) de la Definición 1.1.2:

$$e_i = (\pi_{\text{case}}(e_i), \pi_{\text{act}}(e_i), \pi_{\text{time}}(e_i), \mathcal{D}(e_i))$$

Siguiendo dicho formalismo, una vez definido el evento, se construye la traza enriquecida asociada al caso c bajo la estructura de la ecuación (1.2). Por último, el conjunto de todas las trazas de la muestra da lugar al registro global de eventos L_E , modelado según la ecuación (1.4):

$$\sigma_c^E = \langle e_1, e_2, \dots, e_n \rangle,$$

la cual representa una solicitud de préstamo completa. Por último, el conjunto de todas las trazas enriquecidas da lugar al registro de eventos:

$$L_E = \{\sigma_1^E, \sigma_2^E, \dots, \sigma_N^E\}.$$

Aplicando las definiciones anteriores al conjunto de datos empleado, cada valor distinto de la variable `case:concept:name` identifica una traza del registro de eventos, mientras que cada fila de la tabla representa un evento individual.

Tras la construcción del *log* de eventos se obtienen:

- $|L_E| = 31\,509$ casos.
- $|\mathcal{U}_E| = 1\,202\,267$ eventos, con $\mathcal{U}_E$ el conjunto de eventos.
- $|\mathcal{U}_A| = 26$ actividades distintas, con $\mathcal{U}_A$ el conjunto de actividades.

Los resultados anteriores suponen una media de 38.15 eventos por caso. Desde un punto de vista temporal, los registros abarcan el periodo comprendido entre el 1 de enero de 2016 y el 1 de febrero de 2017. El atributo de mayor cardinalidad es el de los recursos, con un total de 149 trabajadores diferentes.

Más allá de estas métricas agregadas, se analiza la diversidad de comportamientos observados en el proceso. Para ello se emplea el concepto de *variante de ejecución*. Dos trazas enriquecidas pertenecen a la misma variante cuando sus trazas clásicas son iguales. La transformación de una traza enriquecida σ_c^E en su correspondiente traza clásica σ_c se ejecuta aplicando el operador *variant* formalizado en la ecuación (1.3):

$$\text{variant}(\sigma_c^E) = \langle a_1, a_2, \dots, a_n \rangle = \sigma_c$$

donde $a_i = \pi_{\text{act}}(e_i)$ es la actividad asociada al evento e_i . De este modo, dos trazas enriquecidas σ_i^E y σ_j^E pertenecen a la misma variante de ejecución si y solo si

$$\text{variant}(\sigma_i^E) = \text{variant}(\sigma_j^E).$$

Nótese que el concepto de variante es puramente estructural. La función anterior conserva únicamente la secuencia ordenada de actividades tras eliminar toda la información temporal y contextual.

Aplicando la definición anterior sobre las 31 509 trazas enriquecidas del registro, se obtienen 15 930 variantes distintas, de las cuales 13 907 variantes son únicas. Las 46 variantes más frecuentes, todas ellas con frecuencia mayor a 50, explican el 25 % de los casos (la más común se repite en 1 056 ocasiones). Sin embargo, son necesarias más de 9 500 variantes para explicar el 80 % de las observaciones.

Este resultado pone de manifiesto la elevada complejidad y flexibilidad del proceso analizado. Esta elevada dispersión es habitual en procesos reales de concesión de préstamos, donde factores como la situación financiera del cliente, la necesidad de aportar documentación adicional o la generación de múltiples ofertas pueden dar lugar a recorridos significativamente distintos dentro del mismo proceso.

La existencia de un número tan elevado de variantes constituye además uno de los principales retos, ya que dificulta la obtención de modelos fácilmente interpretables y justifica la necesidad de emplear técnicas más sofisticadas.

Por último, se estudia la presencia de múltiples ofertas dentro de una misma solicitud. El registro contiene 42 995 ofertas distintas, lo que equivale a una media de 1.36 ofertas por solicitud.

Una vez formalizado el *log* y descritas sus principales características estructurales, es necesario realizar un análisis exploratorio de los datos. La siguiente sección presenta un estudio descriptivo del conjunto de datos con el objetivo de comprender mejor la distribución de los casos, identificar patrones relevantes y detectar posibles particularidades que deban tenerse en cuenta en las etapas posteriores del proyecto.

3.2.4. Análisis Exploratorio de Datos (EDA)

A diferencia de otros escenarios empresariales, el conjunto de datos empleado en este trabajo fue publicado específicamente con fines de investigación dentro del *BPI Challenge 2017*. Como consecuencia, los registros han sido previamente anonimizados, estructurados y sometidos a distintos procesos de validación, reduciendo significativamente la presencia de problemas habituales en entornos industriales como valores ausentes, inconsistencias estructurales o errores evidentes de registro.

No obstante, la disponibilidad de un conjunto de datos ampliamente utilizado no exime de la necesidad de verificar su calidad antes de abordar cualquier análisis. Por este motivo, se llevaron a cabo diversas comprobaciones orientadas a garantizar la integridad, consistencia y adecuación del registro para los objetivos planteados. Estas verificaciones se inspiraron en las metodologías de evaluación de calidad propuestas por la comunidad *BupaR*, una de las referencias más consolidadas para el análisis de registros de eventos, adaptando a Python los principios descritos en su guía de evaluación de calidad de datos (BupaR Team, 2024).

En esta sección se presentan únicamente los resultados más relevantes para caracterizar el conjunto de datos y facilitar la comprensión del proceso objeto de estudio. No se pretende

realizar un análisis exploratorio y estadístico exhaustivo, ya que este no constituye el objetivo principal del capítulo. En su lugar, se ofrece una visión general del registro de eventos que sirva de contexto para las etapas posteriores.

Control de calidad

Como primera etapa del análisis exploratorio, se llevaron a cabo diversas comprobaciones orientadas a evaluar la calidad del registro. Entre otros aspectos, se revisó la presencia de valores ausentes, la consistencia de los identificadores de caso y evento, así como la integridad de los atributos necesarios para la construcción del registro de eventos. Los resultados obtenidos confirman que todas las variables esenciales del proceso presentan completitud total, incluyendo el identificador del caso (`case:concept:name`), la actividad (`concept:name`), la marca temporal (`time:timestamp`) y el recurso (`org:resource`).

Los únicos valores ausentes detectados aparecen en atributos asociados a las ofertas, tales como `OfferedAmount`, `CreditScore`, `MonthlyCost`, `NumberOfTerms`, `FirstWithdrawalAmount`, `Accepted` y `Selected`. Este comportamiento es coherente con la semántica del proceso, ya que dichos atributos únicamente se encuentran disponibles cuando existe una oferta asociada a la solicitud, concretamente durante los eventos relacionados con su creación (actividad `O_Create Offer`). De forma similar, la variable `OfferID` solo se registra en actividades pertenecientes al ciclo de vida de las ofertas, permaneciendo vacía en el resto de eventos. Por tanto, estos valores ausentes no responden a problemas de calidad del dato, sino a la propia estructura del proceso modelado.

No se detectaron inconsistencias temporales ni eventos duplicados que impidiesen reconstruir correctamente las trazas del proceso. En consecuencia, no fue necesario aplicar procedimientos de limpieza adicionales.

Longitud y duración de las trazas

La Figura 3.1 muestra la distribución del número de eventos por solicitud. Aunque la longitud media se sitúa en torno a 38 eventos por caso, existe una elevada dispersión. Se observan solicitudes que superan ampliamente los 100 eventos.

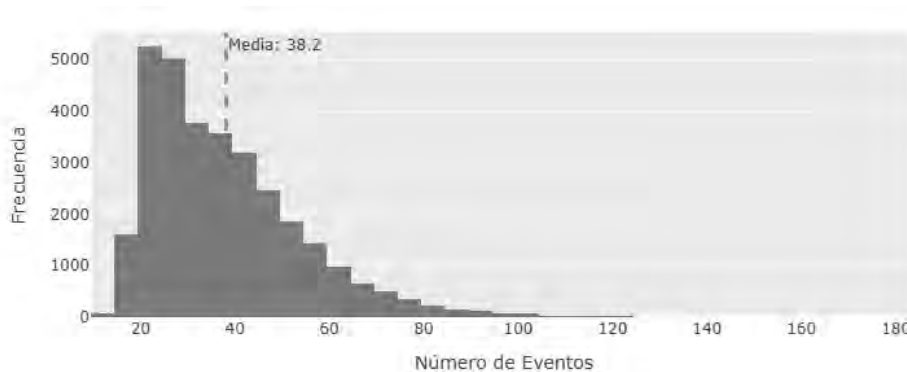


Figura 3.1: Histograma de la longitud de las secuencias junto a su valor medio (línea roja).

El caso más largo corresponde a una secuencia de 180 eventos, en contraste con las numerosas solicitudes de menos de 30 actividades. Esto evidencia la elevada variabilidad del proceso y resulta coherente con el alto porcentaje de variantes identificadas durante la formalización del registro de eventos.

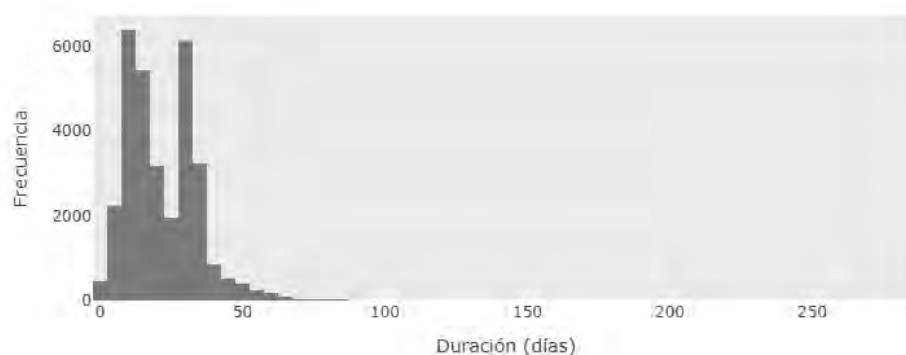


Figura 3.2: Histograma de la duración de las secuencias.

La distribución parece presentar un comportamiento compatible con una estructura bimodal, de acuerdo con la Figura 3.2. La duración media de los casos es de aproximadamente 22 días, mientras que la mediana se sitúa en 19 días. El diagrama de caja de la Figura 3.3 muestra que el 50 % central de las observaciones se concentra aproximadamente entre los 10 y los 30 días, lo que indica que la mayoría de las solicitudes se resuelven en un plazo relativamente reducido. Aunque existen expedientes cuya tramitación alcanza los 286 días, estas duraciones corresponden a un conjunto minoritario de casos.

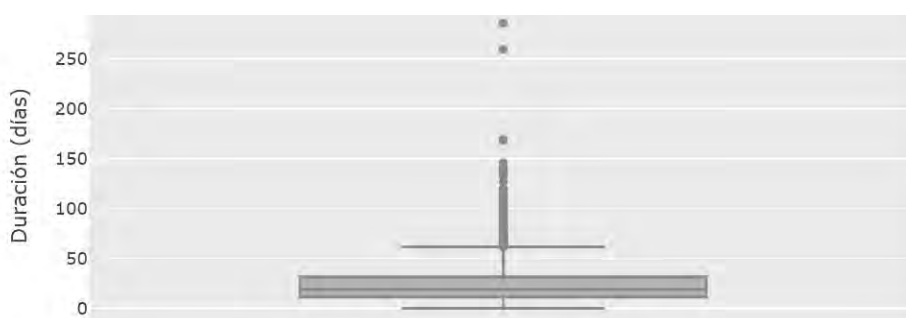


Figura 3.3: Boxplot duración de las secuencias.

Por último, tras analizar la relación entre duración y longitudes de los casos se observó el comportamiento esperado, ambas magnitudes aumentan en conjunto. Un aspecto a destacar es la diferencia de estos valores según el estado final del préstamo. Las solicitudes canceladas presentan un ratio duración/longitud más alto ($\mu = 1.21$), posiblemente asociado a la presencia de cuellos de botella o esperas largas. Por otro lado, los préstamos concedidos tienden a dispersarse a longitudes y duraciones más largas respecto a los denegados, ofreciendo medias

similares: 0.41 para concedidos y 0.44 para denegados.

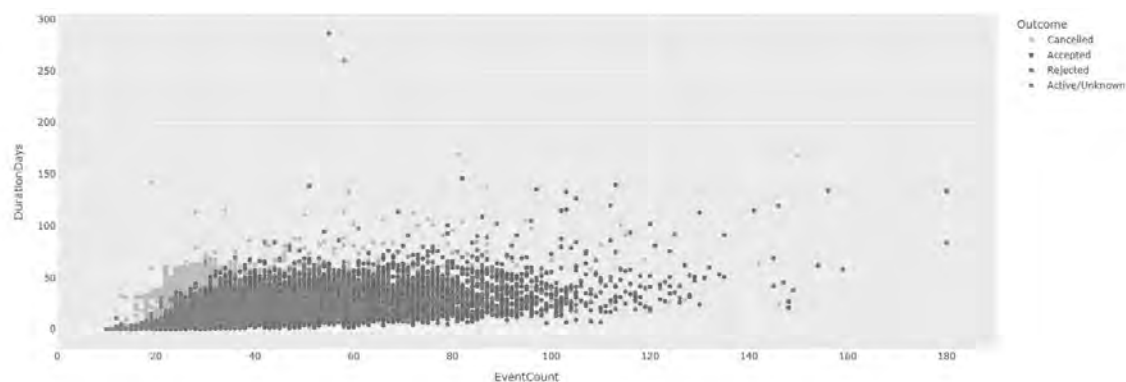


Figura 3.4: Relación Duración - Número de eventos por estado final: Concedido (verde), Denegado (rojo), Cancelado (amarillo), Ninguno de los anteriores (violeta).

Generación de ofertas

Una de las características más interesantes del proceso analizado es la multiplicidad de ofertas dentro de una misma solicitud. La posibilidad de negociaciones por mejores condiciones ante una oferta podría constituir una fuente relevante de variabilidad en la duración y longitud de las trazas. En la Figura 3.5 se puede ver como las solicitudes con una única oferta son las más frecuentes.

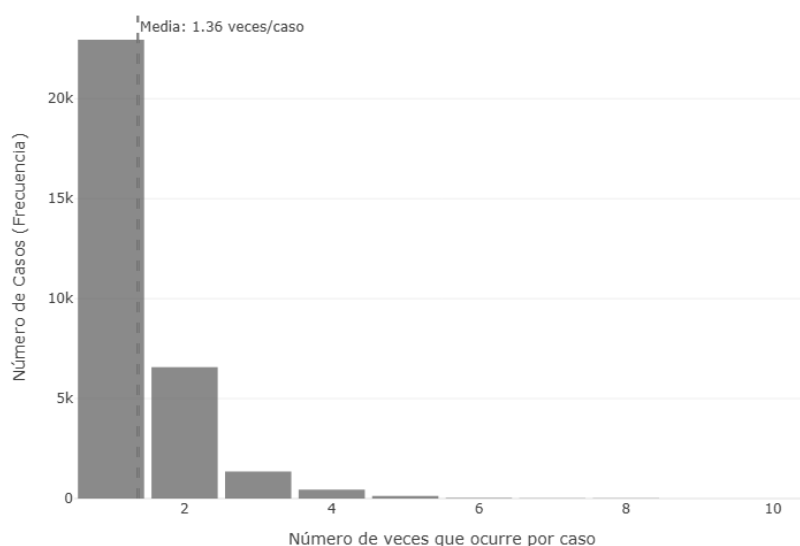


Figura 3.5: Histograma de ofertas por caso y media de ofertas por caso (línea roja).

Sin embargo, aproximadamente el 27% de las solicitudes reciben más de una oferta, al-

canzándose un máximo de 10 ofertas para un mismo caso. Este comportamiento podría estar asociado a mayores tiempos de tramitación y merece un análisis más detallado. Negociaciones tan largas podrían repercutir negativamente en la duración y longitud, motivo por el cual este aspecto ha sido identificado como uno de los puntos de mejora en el proceso.

Concretamente, se plantea el objetivo de que la solución sea capaz de detectar el impacto en la solicitud, prediciendo magnitudes como el tiempo restante o la resolución final en base a las condiciones de la oferta propuesta.

Actividades y recursos

La Figura 3.6 muestra un claro predominio de las actividades operativas (W_), especialmente aquellas relacionadas con la validación de solicitudes y el contacto con los clientes. Esto sugiere que una parte importante del esfuerzo del proceso se concentra en tareas de seguimiento y gestión documental.

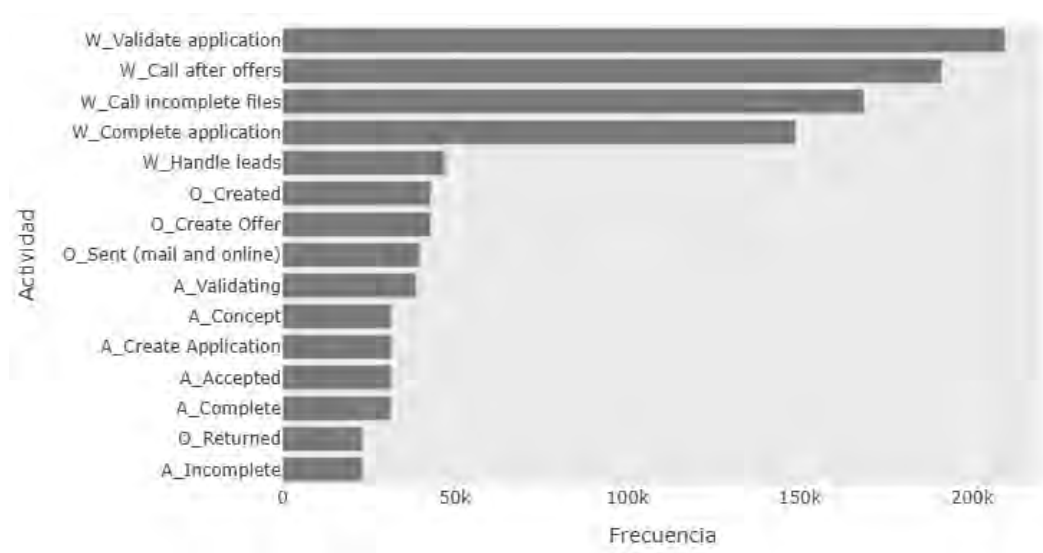


Figura 3.6: Histograma de actividades más frecuentes.

Véase la elevada frecuencia de actividades como *W_Call after offers* y *W_Call incomplete files*. Ambas reflejan situaciones en las que la entidad financiera debe realizar acciones adicionales de seguimiento, bien para obtener documentación pendiente o bien para gestionar la respuesta del cliente ante una oferta.

Tras analizar la frecuencia de tareas realizadas por los recursos, conviene señalar que el recurso *User_1* es el responsable de una parte considerable de los eventos registrados, multiplicando por siete la carga de trabajo media del resto de empleados. El motivo es que no representa un empleado real, sino un usuario ficticio al que se le asignan todas las tareas automáticas ejecutadas por el sistema online.

Estados finales

La Figura 3.7 muestra la distribución de los estados finales observados en las solicitudes. Más de la mitad de los casos (54.7 %) finalizan con la concesión del préstamo, mientras que un 33.1 % terminan siendo cancelados por el cliente y un 11.9 % son denegados por la entidad financiera.

Aunque la mayoría de las solicitudes terminan siendo concedidas, cerca del 45 % no alcanzan una resolución favorable, ya sea por cancelación o por denegación. Además, la distribución observada evidencia un cierto desbalanceo entre las distintas clases de resultado, aspecto que deberá tenerse en cuenta durante la construcción y evaluación de modelos predictivos.

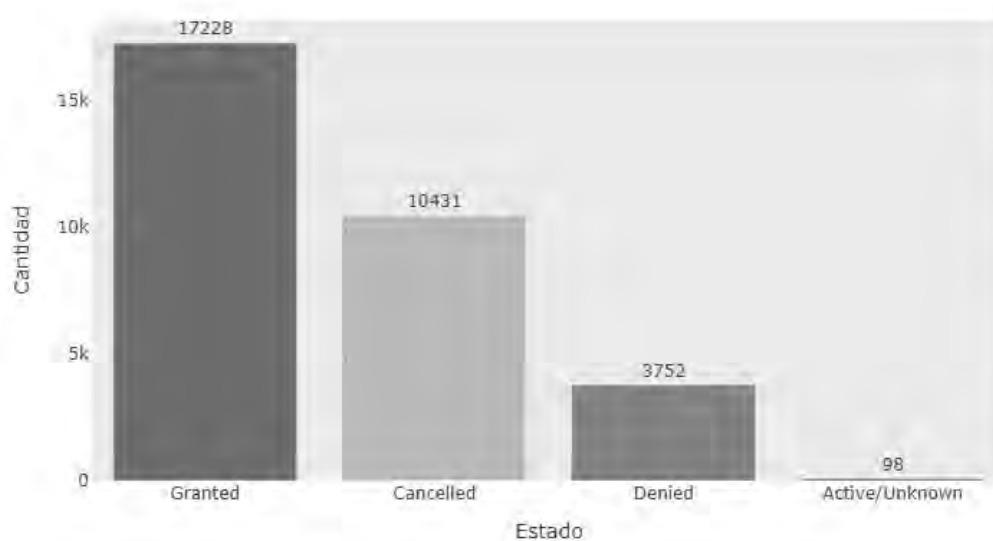


Figura 3.7: Distribución del estado final de la solicitud.

Por otro lado, únicamente 98 solicitudes permanecen activas al final del periodo de observación. Dado que estos casos no disponen todavía de una resolución definitiva, su tratamiento requiere especial atención en tareas posteriores de modelado predictivo para evitar introducir información incompleta o sesgos en el entrenamiento.

Se observó una patrón determinista entre el estado final del préstamo y la variable **CreditScore**: un *score* estrictamente positivo garantiza la concesión del préstamo.

Tras analizar la matriz de correlaciones mediante el coeficiente de Pearson, se obtuvo un valor de 0.73 entre **CreditScore** y la variable binaria **Selected** (tratada numéricamente como 0/1 para el cálculo), indicador de si el cliente selecciona la oferta propuesta. Este resultado pone de manifiesto una asociación positiva elevada entre ambas variables.

En definitiva, los resultados obtenidos en esta sección reflejan la complejidad y variabilidad del proceso. Además, refuerzan la utilidad potencial del registro y de las variables disponibles para abordar los objetivos fijados por la entidad.

3.3. Análisis descriptivo mediante *Process Mining* clásico

Una vez comprendidas las características del registro y verificada su calidad, el siguiente paso consiste en explotar el histórico de solicitudes mediante técnicas clásicas de *Process Mining*. Desde la perspectiva de la entidad financiera, el objetivo de esta fase es proporcionar una herramienta que permita explorar el comportamiento real del proceso por parte de los analistas de la compañía. Esta debe facilitar la identificación de patrones recurrentes, cuellos de botella y oportunidades de mejora en el proceso a partir de los datos históricos disponibles.

Para ello se empleó la librería **PM4Py** (Berti et al., 2023), una de las herramientas de referencia para el análisis de procesos en Python. En este contexto, no fue necesario aplicar transformaciones adicionales sobre los datos. La fase de preparación se limitó a adaptaciones de formato orientadas a garantizar la compatibilidad con la librería utilizada.

A diferencia de los enfoques predictivos o de detección de anomalías, el análisis descriptivo se basa en la exploración guiada del registro. Su verdadero potencial reside en la capacidad del usuario para formular hipótesis, aplicar filtros adecuados e interpretar las visualizaciones generadas con el fin de extraer conclusiones relevantes.

Los patrones identificados en esta fase servirán posteriormente para definir anomalías con impacto en negocio, así como las variables potencialmente relevantes para los modelos predictivos. En las siguientes subsecciones se presentan algunos ejemplos representativos del valor de estas técnicas.

3.3.1. Exploración temporal mediante *Dotted Charts*

Una de las funcionalidades de la herramienta desarrollada es la posibilidad de analizar visualmente la evolución temporal de las solicitudes mediante *Dotted Charts*. En este tipo de representación, cada evento se representa como un punto situado en función de su instante de ocurrencia, ya sea de forma absoluta como fecha o de manera relativa respecto al inicio de la solicitud; mientras que cada fila corresponde a un caso distinto. Permite observar la secuencia de actividades de múltiples casos de forma simultánea, facilitando la identificación de patrones temporales, periodos de inactividad y comportamientos atípicos.

El verdadero valor de estas visualizaciones aparece al combinarse con mecanismos de filtrado. Se selecciona como ejemplo el conjunto de solicitudes con una duración superior a 100 días, para estudiar los 40 casos más largos del registro.

La Figura 3.8 muestra que las solicitudes más largas presentan una elevada concentración de actividades relacionadas con la gestión de ofertas, destacando las numerosas repeticiones de la actividad **W_Call incomplete files** en un gran porcentaje de los casos. Esto sugiere que los problemas relacionados con la documentación requerida podrían estar asociados a una prolongación del proceso, ya que presenta una media de más de 34 apariciones por caso en las solicitudes de más de 100 días, frente a una frecuencia media de 5.35 en la totalidad de las secuencias.

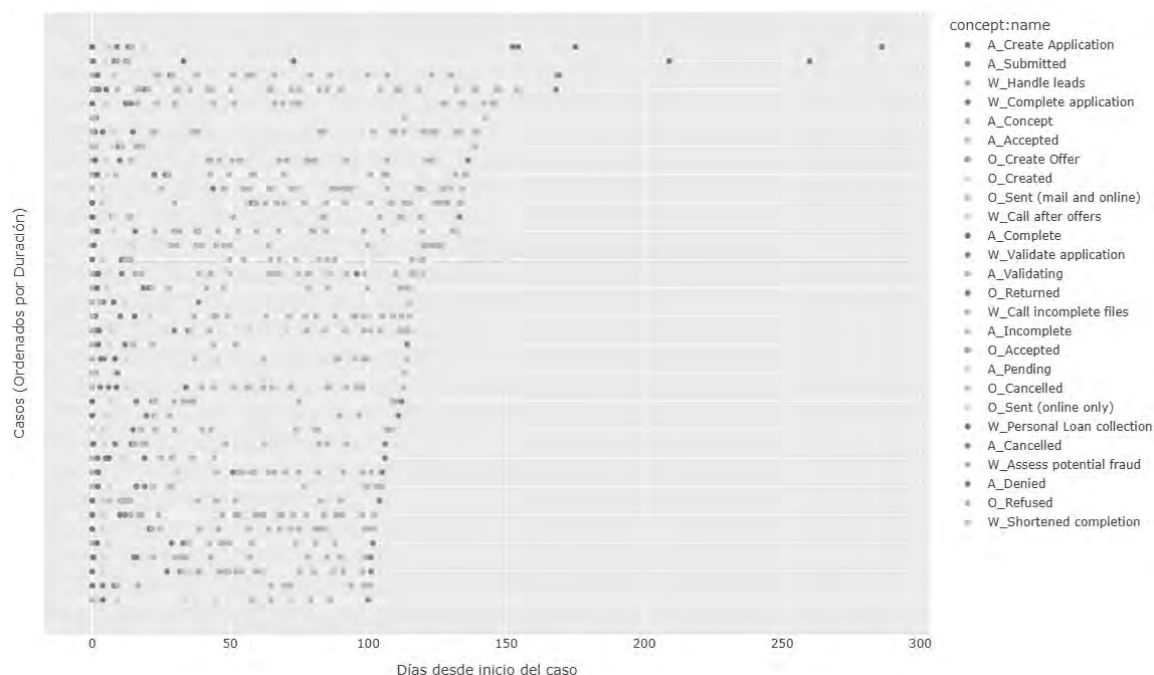


Figura 3.8: *Dotted Chart* relativo para solicitudes con duración superior a 100 días.

Otro aspecto relevante es la existencia de amplios intervalos temporales entre determinados eventos. A diferencia de otros casos del registro, donde las actividades se concentran en periodos relativamente cortos, las solicitudes de larga duración muestran extensas fases de espera antes de alcanzar una resolución final. Este comportamiento sugiere que los retrasos no se deben únicamente a la ejecución o repetición de tareas internas, sino también a tiempos de espera entre actividades.

Cabe destacar que el gráfico desarrollado es interactivo, permitiendo seleccionar el punto deseado para consultar el identificador del caso al que corresponde, el nombre de la actividad y otros atributos asociados; así como aplicar filtros dinámicos con agilidad.

3.3.2. Descubrimiento del proceso

Otra de las funcionalidades incluidas en la herramienta es la generación automática de mapas de proceso a partir del registro histórico. Estas visualizaciones permiten representar las relaciones observadas entre actividades, mostrando las rutas seguidas por las solicitudes y facilitando la identificación de ciclos, bifurcaciones y comportamientos recurrentes.

La Figura 3.9 muestra el modelo descubierto sobre la totalidad del registro (DFG), representando todas las trazas existentes sin ningún filtro. Como puede observarse, la elevada cantidad de actividades y transiciones genera una representación imposible de interpretar de forma directa. Este comportamiento resulta coherente con la alta variabilidad ya observada durante el análisis exploratorio, donde se constató la existencia de numerosas variantes y diferencias significativas entre casos.

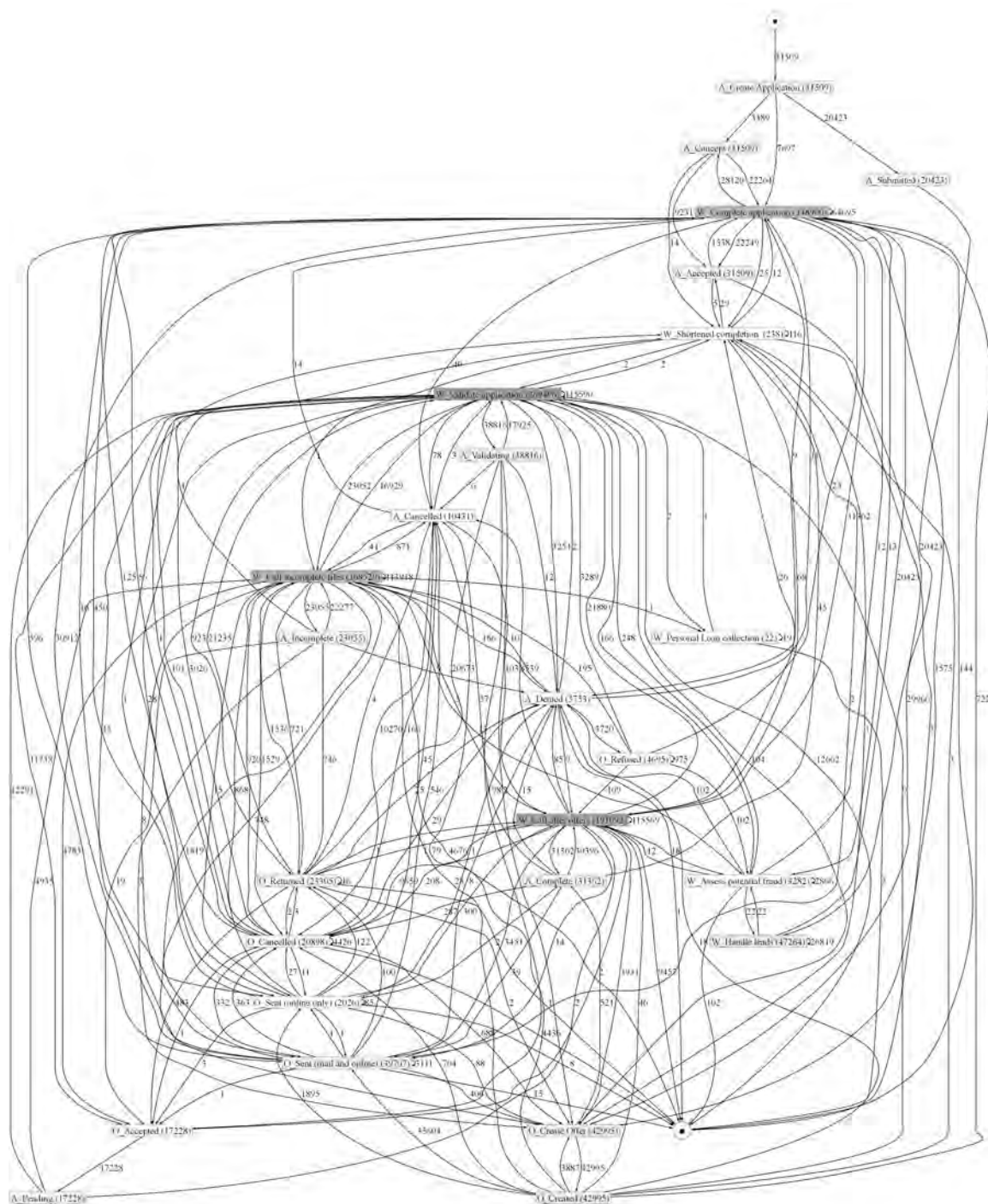


Figura 3.9: DFG calculado sobre el registro completo. Se muestra el nombre de la actividad y su frecuencia de aparición (recuadros); transiciones y frecuencias de transición (flechas).

Ante esta situación, el valor de la herramienta reside en la posibilidad de aplicar filtros que

comportamientos poco frecuentes que podrían pasar desapercibidos durante una exploración manual del registro. Una vez identificados estos casos, las herramientas descriptivas descritas anteriormente pueden emplearse para profundizar en sus causas y comprender mejor los factores que los originan.

No obstante, el problema presenta una dificultad adicional: la ausencia de un conjunto de datos etiquetado que permita distinguir de forma objetiva entre comportamientos normales y anómalos. En consecuencia, la detección debe abordarse desde una perspectiva no supervisada, aprendiendo primero los patrones habituales del proceso para posteriormente identificar aquellas solicitudes que se desvían significativamente de ellos.

3.4.1. Limitaciones de las técnicas clásicas

Tal y como se expuso en el Capítulo 1, en el ámbito tradicional del *Process Mining*, la detección de desviaciones suele abordarse mediante técnicas de Verificación de Conformidad (*Conformance Checking*), como los alineamientos óptimos (*Alignments*) o la reproducción de *tokens* (*Token Replay*). Estas metodologías evalúan el grado de ajuste entre el comportamiento observado en el registro de eventos y un modelo de proceso de referencia, habitualmente representado mediante una Red de Petri.

No obstante, la aplicación de estas técnicas al conjunto de datos del *BPI Challenge 2017* presenta diversas limitaciones para los objetivos establecidos:

- **Dependencia de un modelo de referencia.** Las técnicas de *Conformance Checking* requieren la existencia de un modelo normativo frente al cual comparar las trazas observadas. Sin embargo, el proceso estudiado presenta una elevada variabilidad, tal y como se vio durante el análisis exploratorio y descriptivo. En este contexto, la obtención de un modelo suficientemente representativo no es trivial. Un modelo excesivamente general puede perder capacidad para discriminar comportamientos anómalos, mientras que un modelo demasiado específico puede considerar como desviaciones comportamientos legítimos presentes en el registro.
- **Ceguera ante atributos.** Estas técnicas centran su análisis principalmente en la secuencia de actividades ejecutadas, evaluando el orden y las transiciones observadas entre ellas. Así, su capacidad de detección se limita exclusivamente a anomalías estructurales. Sin embargo, gran parte de la información relevante de la traza se encuentra en los atributos asociados a cada evento, como importes económicos, puntuaciones crediticias, recursos involucrados o tiempos de ejecución. Muchas anomalías de interés para el negocio no se manifiestan mediante secuencias incorrectas de actividades, sino a través de combinaciones atípicas de estos atributos dentro de un flujo aparentemente válido.

Estas limitaciones motivan la búsqueda de enfoques capaces de integrar simultáneamente la información estructural del proceso y los atributos asociados a cada evento. En este contexto, se plantea una solución basada en una las herramientas de aprendizaje profundo estudiadas en el Capítulo 2.

3.4.2. Inyección de anomalías sintéticas

Uno de los principales retos de los problemas de detección de anomalías es la ausencia de un criterio objetivo que permita evaluar la calidad de las detecciones realizadas por el modelo. En el caso estudiado, la entidad financiera no disponía de un histórico etiquetado de incidencias, errores operativos o comportamientos anómalos sobre el que validar las predicciones obtenidas. Se descarta por tanto un enfoque de aprendizaje supervisado.

Idealmente, el entrenamiento no supervisado de un sistema para detección de anomalías debería realizarse sobre un conjunto de datos representativo exclusivamente del comportamiento normal del proceso o, alternativamente, sobre un registro que disponga de etiquetas de anomalía que permitan excluir dichos casos del conjunto de entrenamiento. Sin embargo, ninguna de estas condiciones se cumple en el conjunto de datos empleado, ya que no existe información que permita identificar de forma explícita solicitudes anómalas, como posibles fraudes, errores operativos u otras desviaciones del funcionamiento esperado.

En consecuencia, el modelo se entrena bajo la hipótesis incierta de que el registro histórico refleja mayoritariamente la operativa habitual de la entidad. Esta constituye una limitación inherente al enfoque adoptado, pues, si el conjunto de entrenamiento contiene una proporción significativa de anomalías, el modelo podría llegar a incorporarlas como parte del comportamiento normal, reduciendo su capacidad para detectarlas posteriormente.

Esta estrategia carece de un mecanismo objetivo para evaluar la capacidad del detector. Para solventar este problema, se acordó conjuntamente con la entidad la definición de un conjunto de anomalías sintéticas representativas desde el punto de vista operativo. Estas anomalías fueron inyectadas artificialmente sobre una muestra controlada de solicitudes, generando así un conjunto de referencia (*Ground Truth*) que permitiera cuantificar el rendimiento del sistema.

Las solicitudes modificadas se reservan exclusivamente para la fase de evaluación, evitando cualquier fuga de información (*Data Leakage*) durante el entrenamiento. De este modo, el modelo aprende únicamente la normalidad del proceso y posteriormente se evalúa sobre un conjunto contaminado que combina solicitudes originales y anomalías sintéticas.

A continuación se listan las anomalías generadas manualmente:

Anomalía	Descripción
Anomaly1	Incremento extremo del atributo RequestedAmount , simulando una solicitud con un importe desproporcionado respecto al comportamiento habitual.
Anomaly2	Asignación de un importe de préstamo muy elevado para un objetivo de tipo coche (LoanGoal = Car), pero promedio entre todos los objetivos posibles.
Anomaly3	Modificación de FirstWithdrawalAmount para que supere ampliamente el importe ofertado, creando una incoherencia financiera.
Anomaly4	Modificación de OfferedAmount y MonthlyCost para una misma oferta con valores aleatorios (debería mantenerse constante).

Anomalia	Descripción
Anomaly5	Secuencia completa ejecutada por el mismo recurso (org:resource), nunca observado en el histórico.
Anomaly6	Préstamo concedido con nuevo valor del atributo Selected = <i>Unknown</i> y alteración del atributo Accepted entre <i>True</i> y <i>False</i> .
Anomaly7	Alteración puntual de la marca temporal de un evento, generando una transición casi inmediata entre dos actividades que suelen tener tiempos de transición más elevados.
Anomaly8	Modificación de la fecha de un evento para producir un intervalo temporal significativamente diferente del esperado.
Anomaly9	Desplazamiento completo de la traza completa un día hacia atrás, provocando que las actividades ocurran en domingo en lugar de un día laborable.
Anomaly10	Eliminación de todos los eventos de origen Workflow de una solicitud, alterando la secuencia habitual del proceso.
Anomaly11	Intercambio del orden temporal entre A_Submitted y A_Validating , invirtiendo la lógica natural de ejecución, la solicitud se valida antes de ser recibida.
Anomaly12	Cambio del resultado final de la solicitud a Denied , modificando el desenlace del caso, para un cliente con puntuación crediticia alta.
Anomaly13	Alteración del atributo Selected para generar un préstamo concedido con oferta no seleccionada por el cliente.
Anomaly14	Intercambio temporal entre las actividades O_Create Offer y O_Created , alterando el orden lógico de generación de ofertas.
Anomaly15	Sustitución de la actividad original A_Cancelled por A_Pending , simulando una solicitud concedida que originalmente fue cancelada.

Tabla 3.5: Anomalías sintéticas inyectadas.

Las anomalías definidas abarcan modificaciones de naturaleza diversa y corresponden a comportamientos nunca observados en el histórico. En particular, se incluyen alteraciones sobre variables numéricas de negocio, atributos categóricos, marcas temporales y estructura de las trazas, evitando centrar la evaluación exclusivamente en cambios sobre la secuencia de actividades. De este modo, el conjunto de referencia permite comprobar si el sistema es capaz de identificar comportamientos anómalos independientemente del tipo de información afectada.

Por último, antes de adentrarnos en los detalles de la solución planteada, puntualizar que las anomalías anteriores se han construido modificando secuencias reales del histórico que el modelo no identificó como anómalas en una primera ejecución.

3.4.3. Visión global de la solución

A la vista de las limitaciones de las técnicas clásicas y de la ausencia de ejemplos etiquetados de comportamiento anómalo, se propone una estrategia basada en aprendizaje profundo no supervisado utilizando un autoencoder sobre grafos. La elección de esta arquitectura se fundamenta en los conceptos introducidos en el Capítulo 2, donde se mostró el potencial de las Redes Neuronales sobre Grafos (GNN) en combinación con los autoencoders para aprender representaciones compactas de la normalidad sin necesidad de supervisión teniendo en cuenta la estructura secuencial del proceso.

Frente a alternativas tabulares como *Isolation Forest* (Liu et al., 2008), esta formulación presenta ventajas al operar directamente sobre la estructura del proceso. Al modelar cada solicitud como un grafo y reconstruir cada evento en su contexto secuencial, el modelo es capaz de evaluar la coherencia de un evento respecto a sus vecinos, detectando no solo valores atípicos aislados sino también combinaciones de atributos que resultan incoherentes dentro de la traza. Además, la representación mediante grafos admite de forma natural secuencias de longitud variable sin necesidad de agregación ni truncamiento. Esta capacidad para integrar simultáneamente información estructural, temporal y de atributos heterogéneos es consistente con las investigaciones previas, donde se evidencia la eficacia general de los autoencoders neuronales (Nolle et al., 2018) y, de manera específica, la superioridad de los *Graph Autoencoders* (Huo et al., 2021) frente a métodos clásicos en tareas de detección de anomalías sobre registros de eventos.

La lógica subyacente consiste en entrenar el modelo para reconstruir fielmente las solicitudes históricas consideradas normales. Durante este proceso, la red aprende una representación latente que condensa las relaciones entre actividades, la evolución temporal y las dependencias existentes entre los distintos atributos de negocio. Cuando posteriormente se presenta una solicitud cuyo comportamiento difiere significativamente de los patrones aprendidos, el error de reconstrucción aumenta, permitiendo utilizarlo como indicador de anomalía.

A grandes rasgos, el procedimiento completo puede dividirse en cuatro etapas principales:

1. Preparación y transformación del registro de eventos, incorporando las transformaciones necesarias sobre variables temporales y categóricas.
2. Modelado de cada solicitud como un grafo secuencial, donde los eventos constituyen los nodos y las transiciones temporales las aristas.
3. Entrenamiento de un autoencoder basado en Graph Neural Networks, encargado de aprender una representación latente y compacta del comportamiento normal del proceso y reconstruir tanto la información estructural como los atributos asociados a cada evento.
4. Cálculo de una puntuación de anomalía a partir de los errores obtenidos durante la reconstrucción, generando finalmente un ranking que prioriza las solicitudes cuyo comportamiento resulta más alejado de la normalidad aprendida.

La Figura 3.11 resume de forma esquemática la arquitectura general propuesta. En las siguientes subsecciones se describen con detalle cada una de las etapas del *pipeline*, desde la preparación del registro hasta el cálculo final y visualización de la puntuación de anomalía.

La implementación se ha desarrollado íntegramente en *Python* utilizando las librerías presentadas en el Capítulo 2, remitiéndose al lector al código fuente para los aspectos específicos de implementación.

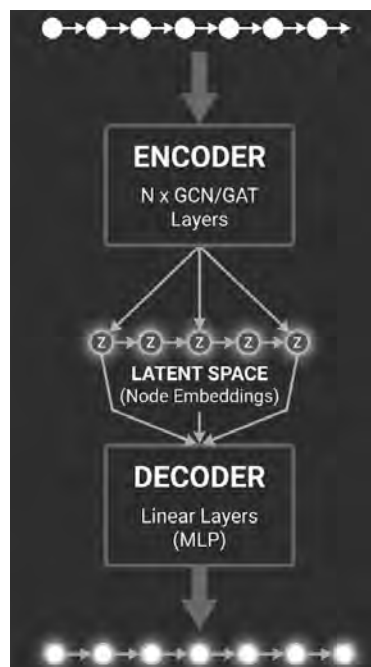


Figura 3.11: Esquema simplificado del autoencoder de secuencias para la detección de anomalías.

3.4.4. Preparación de los datos

En primer lugar, se filtra el conjunto de datos para quedarse únicamente con secuencias completas para las cuales se conoce la resolución final de la solicitud. De este modo, se aplica un filtro para eliminar aquellas trazas que no contengan ninguno de los siguientes eventos: `A_Pending`, `A_Cancelled` o `A_Denied`.

Como se vio en el análisis exploratorio de los datos, muchos atributos no se registran en todos los eventos de una solicitud, como los asociados a las ofertas. Para que los datos sean óptimos para su consumo por el modelo, es necesario rellenar estos nulos de forma coherente. Además, se lleva a cabo un proceso de ingeniería de características (*feature engineering*) mediante el cual se generan nuevas variables derivadas y se transforman atributos existentes con el objetivo de proporcionar una representación más informativa para el modelo. Se realizaron las siguientes operaciones:

1. Propagación del OfferID

Mediante una estrategia de *Forward Fill* se asume que los eventos posteriores a la creación de una oferta (`A_` y `W_`) permanecen asociados a la última oferta generada dentro de la misma solicitud hasta que aparece una nueva oferta. Esta aproximación permite

extender la información de la oferta a eventos que originalmente no la incluyen de forma explícita, preservando el contexto del caso a lo largo de la secuencia.

Debe señalarse que esta hipótesis no está garantizada por el registro de eventos y puede no cumplirse en situaciones particulares, por ejemplo cuando un evento esté realmente vinculado a una oferta anterior distinta de la última creada. No obstante, dada la ausencia de dicha información en el histórico, se adopta esta simplificación al considerarse una aproximación razonable para la reconstrucción del contexto de cada evento.

A partir del `OfferID`, los eventos se agrupan por dicho identificador y se propagan los atributos asociados

2. Imputación de nulos

Para los nulos restantes, que corresponden a columnas asociadas a una oferta en eventos previos a la creación de la primera de ellas, se rellenan con el valor 0 en el caso de las variables numéricas y con *Unknown* para las categóricas. Estos valores son coherentes con la propia semántica del proceso.

3. Generación de variables derivadas

El nuevo atributo `offer_rank` sustituye el identificador de la oferta por su orden de emisión dentro de cada solicitud. Esta variable toma el valor 0 para los eventos previos a la primera oferta y valores crecientes (1, 2, 3, ...) para las sucesivas ofertas generadas.

Se crea la variable categórica `Outcome_case` que contiene el estado final de la solicitud, es decir, *Granted*, *Cancelled* o *Denied*. Esta se incorpora a todos los eventos de la solicitud

4. Transformación de variables temporales

Con el objetivo de proporcionar al modelo una representación más adecuada de la información temporal, se generan nuevas variables a partir de la marca de tiempo de cada evento.

En primer lugar, se calcula el tiempo acumulado transcurrido desde el inicio de la solicitud (`cumulative_time`) y posteriormente se aplica una transformación logarítmica (`log_cumulative_time`). Esta transformación comprime la distribución de los tiempos, buscando reducir su elevada asimetría (Skewness `cumulative_time` (antes): 2.410156, Skewness `log1p(cumulative_time)`: -1.430959) y homogeneizar la variabilidad.

Las componentes periódicas del tiempo se representan mediante proyecciones trigonométricas. En concreto, se generan las variables `hour_sin`, `hour_cos`, `wday_sin`, `wday_cos`, `month_sin` y `month_cos`, obtenidas como la transformación seno y coseno de la hora del día, el día de la semana y el mes del año, respectivamente.

Esta codificación evita las discontinuidades artificiales presentes en la representación original de dichas variables (por ejemplo, entre las 23:00 y las 00:00 horas) y permite que el modelo capture de forma natural patrones estacionales y comportamientos periódicos.

Como resultado de este proceso, se obtiene un *dataset* tabular donde cada evento queda representado por un conjunto completo de atributos numéricos y categóricos, libres de valores ausentes y enriquecidos con información temporal. Esta representación constituye la base sobre la que, en la siguiente subsección, cada solicitud será transformada en un grafo dirigido para su posterior procesamiento mediante el *autoencoder*.

3.4.5. Formalización y modelado como grafo

Una vez completada la preparación del conjunto de datos, cada solicitud de préstamo se transforma en una estructura de grafo que pueda ser procesada por la red neuronal. En este trabajo, cada traza se modela de forma independiente como un grafo dirigido y atribuido,

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{E}_{\text{attr}}),$$

donde $\mathcal{V}$ representa el conjunto de nodos, $\mathcal{E}$ el conjunto de aristas dirigidas, $\mathbf{X}$ las características asociadas a cada nodo y $\mathbf{E}_{\text{attr}}$ los atributos asociados a las aristas.

Esta representación permite trasladar el problema de detección de anomalías desde un registro tabular de eventos hacia una estructura relacional sobre la que pueden operar las Redes Neuronales sobre Grafos. De esta forma, el modelo no solo dispone de la información propia de cada evento, sino también de su contexto secuencial y temporal dentro del proceso.

Topología del grafo

Cada nodo representa un evento individual perteneciente a una solicitud,

$$\mathcal{V} = \{v_1, v_2, \dots, v_n\},$$

ordenado cronológicamente según su marca temporal.

La estructura de grafo se obtiene conectando únicamente eventos consecutivos mediante aristas dirigidas,

$$\mathcal{E} = \{(v_1, v_2), (v_2, v_3), \dots, (v_{n-1}, v_n)\}.$$

La elección de un grafo dirigido responde a la propia naturaleza del proceso de negocio modelado. Las solicitudes de préstamo constituyen secuencias temporales donde existe una relación causal entre eventos consecutivos: una actividad ocurre como consecuencia de la anterior y condiciona la siguiente. Al mantener las aristas dirigidas, la información se transmite únicamente en el sentido natural de ejecución de la solicitud, preservando la causalidad temporal y permitiendo que la representación latente refleje fielmente la evolución del caso. Además, la dirección de las aristas servirá posteriormente para incorporar información asociada a cada transición, como el tiempo transcurrido entre eventos.

$$\mathbf{v}_1 \longrightarrow \mathbf{v}_2 \longrightarrow \mathbf{v}_3 \longrightarrow \dots \longrightarrow \mathbf{v}_n$$

Cada solicitud genera exactamente un grafo independiente, por lo que el conjunto de entrenamiento queda finalmente constituido por una colección de grafos de distinta longitud.

Características de los nodos

Cada nodo almacena el estado completo del evento mediante un vector de características

$$\mathbf{x}_i = [\mathbf{x}_i^{\text{num}}, \mathbf{x}_i^{\text{cat}}],$$

en este vector se omiten identificadores, los cuales carecen de información semántica. Concretamente, las variables empleadas son las siguientes:

Variables categóricas	Variables numéricas
concept:name	case:RequestedAmount
lifecycle:transition	OfferedAmount
EventOrigin	MonthlyCost
Action	NumberOfTerms
org:resource	FirstWithdrawalAmount
case:LoanGoal	CreditScore
case:ApplicationType	log_cumulative_time
Accepted	hour_sin
Selected	hour_cos
offer_rank	wday_sin
Outcome_case	wday_cos
	month_sin
	month_cos

Tabla 3.6: Variables utilizadas como características de los nodos del grafo.

Véase que la variable `offer_rank` aparece en la lista de categóricas. Si bien este atributo representa un orden, modelar cada valor como una categoría independiente evita imponer relaciones lineales artificiales entre sus valores y permite preservar el carácter semántico de cada posición dentro del proceso. Cada una de estas variables categóricas se codifica mediante un *Label Encoder*, obteniendo un identificador entero único, y reservando un valor especial (UNK) para garantizar la robustez del modelo frente a categorías no observadas en el entrenamiento.

Antes de construir el grafo, todas las variables numéricas se normalizan utilizando un *Robust Scaler*. A diferencia de una estandarización convencional basada en media y desviación típica, este método utiliza la mediana y el rango intercuartílico, reduciendo significativamente la influencia de valores extremos frecuentes en variables financieras como los importes solicitados o las duraciones de las solicitudes.

Además del vector de atributos, cada nodo incorpora un índice posicional entero correspondiente a su orden dentro de la secuencia,

$$p_i = i,$$

que posteriormente servirá como entrada para una codificación posicional aprendible por la red neuronal. Esto permite incorporar de forma explícita la información sobre la posición que ocupa cada evento dentro de la traza y distinguir actividades iguales que aparecen en diferentes etapas del proceso.

Atributos de las aristas

Las aristas no representan únicamente la existencia de una transición entre dos eventos consecutivos, sino también el tiempo transcurrido entre ellos. Dada la arista

$$\varepsilon_i = (v_i, v_{i+1}),$$

el atributo asociado a dicha arista se obtiene calculando inicialmente

$$\Delta t_i = t_{i+1} - t_i.$$

Sobre esta diferencia temporal expresada en segundos se aplica una transformación logarítmica

$$\Delta t'_i = \log(1 + \Delta t_i),$$

con el objetivo de reducir la elevada asimetría existente entre transiciones de distinta duración.

Posteriormente, cada transición se normaliza mediante un Z-Score específico para el par de actividades implicado,

$$z_i = \frac{\Delta t'_i - \mu_{(a,b)}}{\sigma_{(a,b)}},$$

donde $\mu_{(a,b)}$ y $\sigma_{(a,b)}$ representan la media y desviación típica históricas observadas para la transición desde la actividad a hasta la actividad b , tras la transformación logarítmica. En el caso de transiciones no presentes en el conjunto de entrenamiento, se utilizan las estadísticas globales del conjunto de transiciones como mecanismo de respaldo.

Finalmente, cada arista queda caracterizada por un único atributo

$$\mathbf{a}_i = [z_i],$$

que cuantifica cuánto se desvía temporalmente dicha transición respecto al comportamiento habitual observado para ese mismo cambio de actividad.

Véase que la decisión inicial de una topología dirigida resulta coherente con la definición de los atributos de las aristas. Cada conexión representa una transición concreta entre dos actividades consecutivas y almacena el tiempo transcurrido entre ellas, de modo que la relación (A, B) no es equivalente a (B, A) . En consecuencia, la dirección de la arista no solo preserva la secuencia lógica del proceso, sino que es necesaria para dotar de significado al propio atributo temporal asociado.

Esta representación es especialmente relevante para la detección de anomalías temporales, ya que una transición extremadamente rápida o lenta generará automáticamente valores alejados de cero incluso aunque la secuencia de actividades permanezca inalterada.

3.4.6. Arquitectura del autoencoder

Una vez contruidos los grafos descritos en la subsección anterior, el siguiente paso consiste en definir una arquitectura capaz de procesarlos y aprender representaciones útiles para la detección de anomalías.

Para ello, se propone un autoencoder basado en Graph Attention Networks, diseñado específicamente para explotar de forma conjunta la información estructural, temporal y de negocio contenida en cada solicitud.

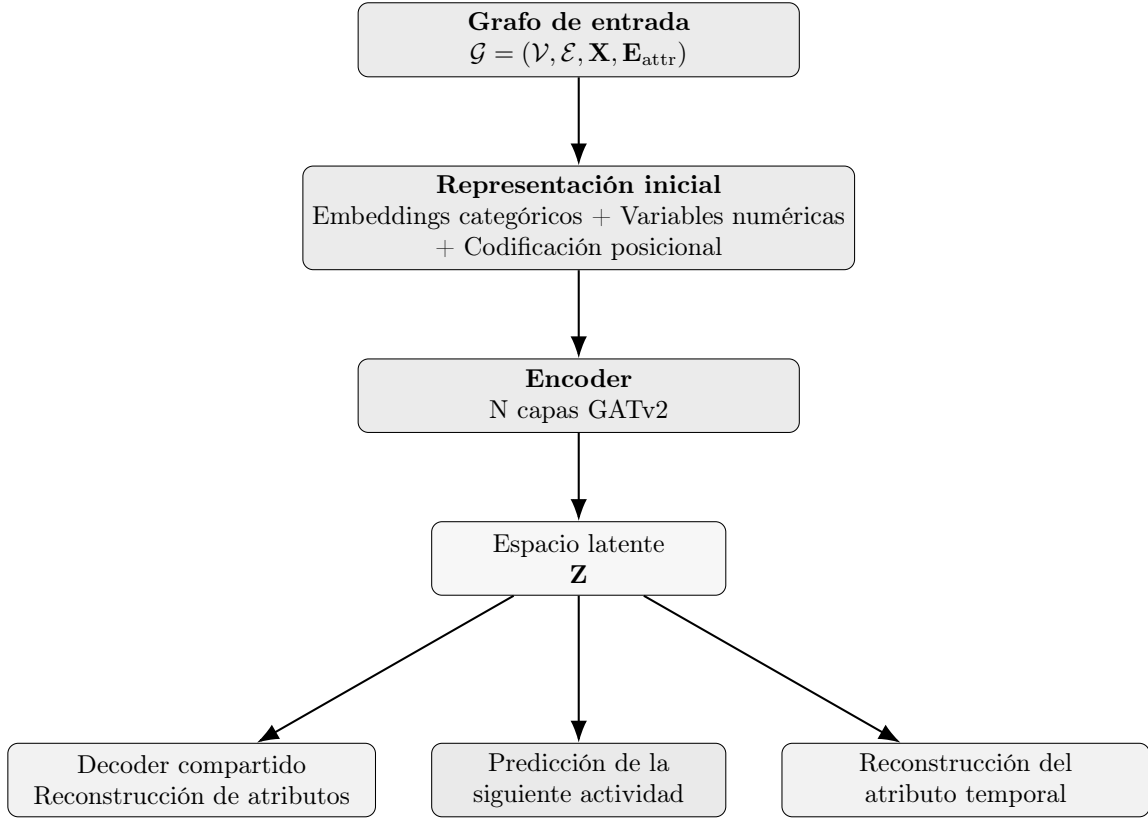


Figura 3.12: Arquitectura general del autoencoder propuesto.

La arquitectura puede dividirse en cuatro bloques principales:

1. **Representación inicial de los nodos.** Su objetivo es transformar las variables heterogéneas de entrada (numéricas y categóricas) en una representación vectorial homogénea que pueda ser procesada por la red neuronal, incorporando además información sobre la posición secuencial de cada evento dentro de la traza.
2. **Encoder basado en capas GATv2.** Encargado de propagar información entre eventos vecinos según la estructura de grafo para obtener representaciones latentes.
3. **Decoder para la reconstrucción de atributos.** Su finalidad es reconstruir las características originales de los eventos a partir del espacio latente aprendido.
4. **Cabezas auxiliares para tareas complementarias de aprendizaje.** Estas tareas adicionales, como la predicción de la siguiente actividad o la reconstrucción de la información temporal de las transiciones, enriquecen la supervisión del entrenamiento y favorecen la obtención de vectores latentes más informativos.

El objetivo primordial de este diseño es que el autoencoder actúe como un filtro selectivo de la normalidad operativa: ante un caso que presente desviaciones estructurales, lógicas o temporales, el modelo carecerá de los mecanismos necesarios para procesarlo correctamente. Esta incapacidad se traducirá en un incremento en los errores de reconstrucción, proporcionando la métrica cuantitativa para priorizar las solicitudes potencialmente anómalas.

En los apartados siguientes se describe detalladamente el funcionamiento de cada uno de los bloques anteriores y las decisiones de diseño adoptadas durante su implementación. La Figura 3.12 muestra una visión global del flujo de información a través del modelo.

Representación inicial de los nodos

Los nodos del grafo contienen información de naturaleza muy diferente: variables numéricas continuas, variables categóricas codificadas mediante enteros y un índice posicional que representa el orden del evento dentro de la traza. Estas magnitudes no pueden ser utilizadas directamente por una red neuronal, ya que presentan escalas y semánticas completamente distintas. Por ello, antes de aplicar el encoder, es necesario transformarlas en una representación vectorial homogénea.

Para lograrlo, cada una de las variables categóricas del nodo i se proyecta hacia un espacio continuo mediante matrices de *embeddings* entrenables, $\mathbf{E}_m$. De este modo, cada valor de la variable categórica m en el evento i , $cat_m^{(i)}$, se transforma en un vector denso de dimensión constante.

La dimensión de los *embeddings* es un hiperparámetro para el que no existe un criterio universal de selección. En versiones preliminares de la implementación se exploró la posibilidad de ajustar dinámicamente dicha dimensión en función de la cardinalidad de cada variable categórica. Sin embargo, las diferencias observadas respecto al empleo de una dimensión fija no resultaron sustanciales en los experimentos realizados. Por este motivo, se optó por fijar empíricamente una dimensión de 8 para todas las variables categóricas, simplificando el diseño de la arquitectura y reduciendo el número de hiperparámetros sin apreciar cambios en el rendimiento.

A continuación, se concatenan los vectores correspondientes a todas las variables categóricas con el vector original de variables numéricas normalizadas $\mathbf{x}_i^{num}$, obteniendo el vector unificado $\mathbf{x}_i^{full}$ de la ecuación (2.1):

$$\mathbf{x}_i^{full} = \left[\mathbf{x}_i^{num} \parallel \mathbf{E}_1(cat_1^{(i)}) \parallel \mathbf{E}_2(cat_2^{(i)}) \parallel \cdots \parallel \mathbf{E}_M(cat_M^{(i)}) \right],$$

el cual cuenta con $13 + 8 \cdot 11 = 101$ dimensiones.

Cabe destacar que estas matrices paramétricas de *embeddings* no provienen de un modelo pre-entrenado externo, sino que se inicializan de forma estocástica y se optimizan de manera conjunta con el resto de la red. Durante la fase de retropropagación del entrenamiento, los gradientes ajustan los pesos de las matrices $\mathbf{E}_m$, permitiendo que el modelo aprenda de forma autónoma representaciones vectoriales donde categorías semánticamente similares ocupen regiones cercanas en el espacio vectorial.

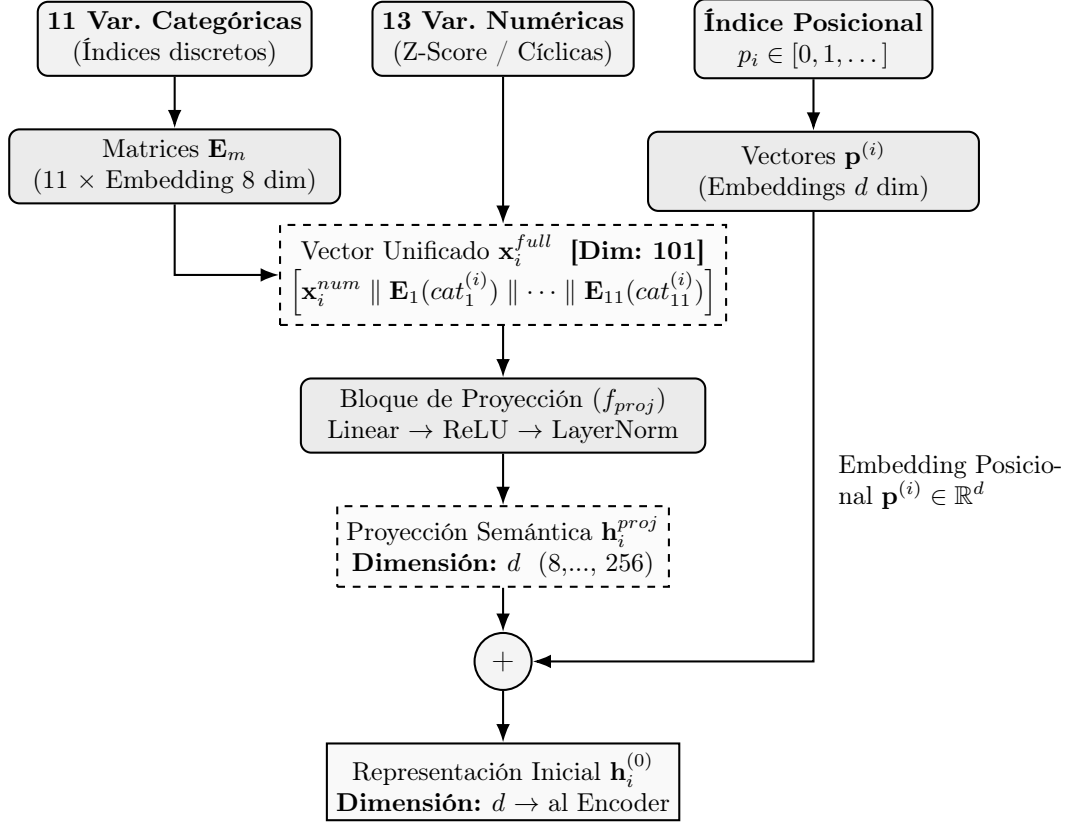


Figura 3.13: Flujo de transformaciones y evolución de dimensiones en la fase de representación inicial del nodo.

Posteriormente, este vector concatenado atraviesa un bloque de proyección inicial. Esta transformación aplica una proyección matemática no lineal para mapear el vector hacia la dimensión latente configurada para el experimento ($d \in \{8, 16, 32, 64, 128, 256\}$), estabilizando simultáneamente las magnitudes de las características antes de iniciar el paso de mensajes en el grafo:

$$\mathbf{h}_i^{proj} = f_{proj}(\mathbf{x}_i^{full}),$$

de forma simplificada, esta proyección se basa en aplicar secuencialmente una transformación lineal, función de activación no lineal y capa de normalización.

Finalmente, es útil inyectar información explícita sobre la ubicación cronológica del evento para que el modelo comprenda la secuencialidad de la traza. Para ello, se entrena un *embedding* posicional $\mathbf{p}^{(i)}$ de dimensión d , basándose en el índice temporal p_i , y se suma directamente a la proyección:

$$\mathbf{h}_i^{(0)} = \mathbf{h}_i^{proj} + \mathbf{p}^{(i)}$$

La elección de una operación de suma en lugar de una concatenación se fundamenta en el estándar metodológico introducido por las arquitecturas de atención de referencia como el *Transformer* (Vaswani et al., 2017).

Esta representación resultante $\mathbf{h}_i^{(0)} \in \mathbb{R}^d$ constituye la representación vectorial inicial del nodo. En este punto, el vector integra exitosamente la semántica del negocio, las magnitudes financieras y su posición absoluta en la secuencia temporal, quedando preparado para interactuar con su vecindario a través del mecanismo de atención.

Encoder basado en capas GATv2

Una vez obtenida la representación vectorial inicial $\mathbf{h}_i^{(0)} \in \mathbb{R}^d$ para cada evento, el siguiente bloque de la arquitectura se encarga de propagar y agregar la información contextual a lo largo de la topología del grafo. Para ello, se emplea un codificador compuesto por dos capas sucesivas de Redes de Atención en Grafos de segunda generación (*Graph Attention Networks v2*, GATv2), cuyo mecanismo calcula los coeficientes de relevancia locales gobernado por la ecuación de atención dinámica (2.2).

Esta propagación iterativa de la información se asienta sobre el paradigma algebraico de paso de mensajes (*message passing*) formalizado en la Definición 2.3.1, donde el algoritmo encadena el cálculo de mensajes inter-nodo (Mensaje), su posterior agregación (Agregación) sobre el vecindario, y la actualización final del estado latente (Actualización).

Dado que el grafo construido presenta una estructura lineal y dirigida donde cada nodo se conecta únicamente con el evento inmediatamente anterior y consigo mismo (mediante *self-loop*), el empleo de dos capas consecutivas permite expandir este flujo de difusión relacional hasta dos saltos de distancia, logrando que cada hito de la solicitud se contextualice en base al historial operativo exacto que lo precede.

Aunque una implementación basada en GAT original también habría sido válida para la arquitectura propuesta, se optó por GATv2 por ser la evolución recomendada del mecanismo original, sin penalización apreciable en términos de complejidad. Si bien un estudio comparativo entre ambas opciones queda fuera del alcance de este trabajo, dicha evaluación constituye una línea natural de investigación futura.

El objetivo del encoder es, por tanto, que cada evento incorpore de forma selectiva la información procedente de su entorno operativo. Dado que el grafo construido presenta una estructura lineal y dirigida, cada nodo se conecta únicamente con el evento inmediatamente anterior y consigo mismo mediante autoconexión (*self loop*). Al encadenar dos capas GATv2 consecutivas, el rango se amplía hasta dos saltos, permitiendo capturar dependencias secuenciales de mayor alcance y patrones locales dentro del flujo del proceso.

La implementación empleada incorpora además varias decisiones de diseño relevantes:

- **Inclusión de atributos de arista** (`edge_dim = 1`). Cada conexión incorpora como atributo el *z-score* del tiempo de transición entre eventos, calculado sobre su transformación logarítmica. De este modo, la atención puede modularse no solo en función de las características de los nodos, sino también de la normalidad temporal de la transición observada.
- **Atención multi-cabeza** (`heads = 4`, `concat = False`). Se utilizan cuatro cabezas de atención independientes que permiten aprender simultáneamente diferentes criterios de

agregación. Sus salidas se promedian, manteniendo constante la dimensión latente d y evitando un crecimiento innecesario del número de parámetros.

- **No linealidad y regularización.** Tras cada capa se aplica una función de activación ReLU para dotar al espacio de propiedades no lineales. Adicionalmente, de forma previa a la segunda capa, se intercala un *Dropout* del 20 %. Esta técnica opera únicamente durante la fase de entrenamiento, poniendo a cero una quinta parte de los elementos del vector latente de forma aleatoria con el fin de favorecer la generalización y evitar el sobreajuste.

Como resultado, el encoder genera para cada grafo una matriz de representaciones latentes $\mathbf{Z} \in \mathbb{R}^{|\mathcal{V}| \times d}$, donde recordemos que $\mathcal{V}$ es el conjunto de eventos (vértices) de una solicitud (grafo). De este modo, cada vector $\mathbf{z}_i \in \mathbf{Z}$ sintetiza la información del propio evento junto con el contexto estructural, temporal y semántico de su vecindario, constituyendo la base para las tareas de reconstrucción y predicción descritas en los bloques posteriores.

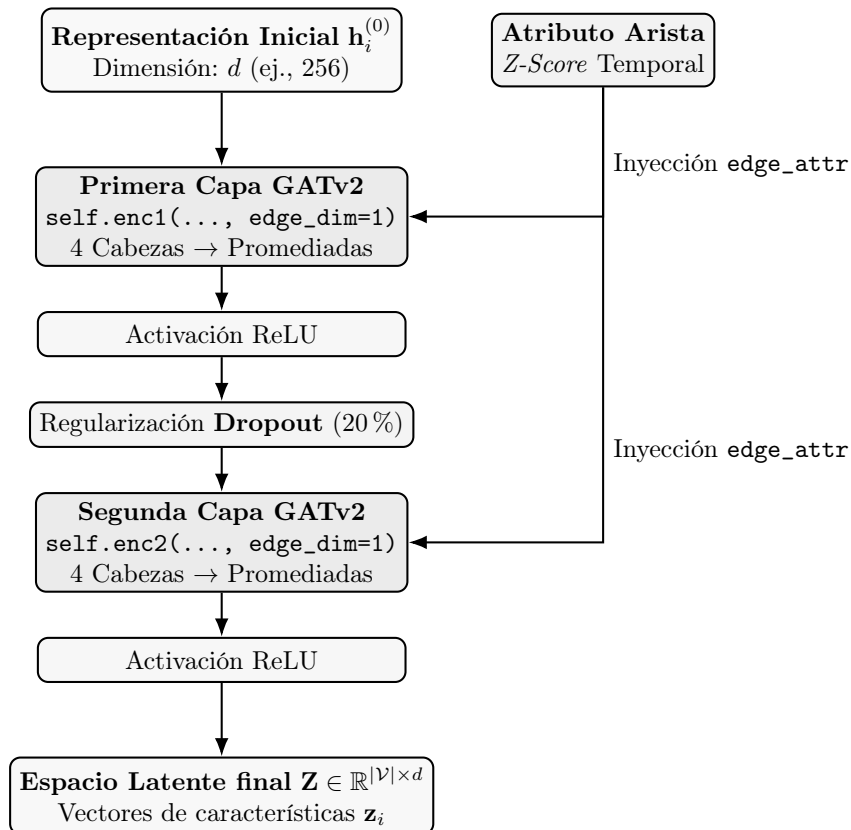


Figura 3.14: Flujo de mensajes y propagación de características a través de las capas del encoder GATv2.

Decoder unificado y cabezas auxiliares

El bloque final de la arquitectura propuesta corresponde al decodificador unificado y sus cabezas auxiliares. Su objetivo es revertir la compresión realizada por el *encoder* y proyectar las representaciones densas de la matriz del espacio latente $\mathbf{Z} \in \mathbb{R}^{|\mathcal{V}| \times d}$ de vuelta a los espacios de características y objetivos originales.

El flujo de procesamiento de la información se estructura como en la Figura 3.15. Como se observa en el esquema, el modelo divide la tarea de descompresión en tres ramas diferenciadas que operan en paralelo a partir de las representaciones latentes $\mathbf{z}_i$:

- **Reconstrucción de atributos de nodo:** Los vectores latentes atraviesan el decodificador compartido donde, mediante un MLP de dos capas, se recuperan las correlaciones complejas del negocio. El vector unificado resultante se segrega finalmente en las variables numéricas estimadas ($\hat{\mathbf{x}}_i^{num}$) y en los vectores de *logits* de las variables categóricas ($\hat{\mathbf{x}}_{m,i}^{cat}$).
- **Predicción estructural:** Se encarga de evaluar la secuencia lógica del proceso proyectando linealmente el vector del nodo central para predecir la siguiente actividad del flujo. Esta tarea busca obligar al modelo a anticipar el siguiente paso para detectar anomalías de flujo como la omisión de controles obligatorios o la ejecución de actividades fuera del orden establecido.
- **Reconstrucción de aristas :** Se especializa en la dimensión temporal. Para evaluar la transición entre dos eventos consecutivos, concatena los vectores latentes del par de nodos implicados ($[\mathbf{z}_i \parallel \mathbf{z}_j]$) y proyecta dicha información hacia un único valor escalar que estima el *Z-score* de la duración.

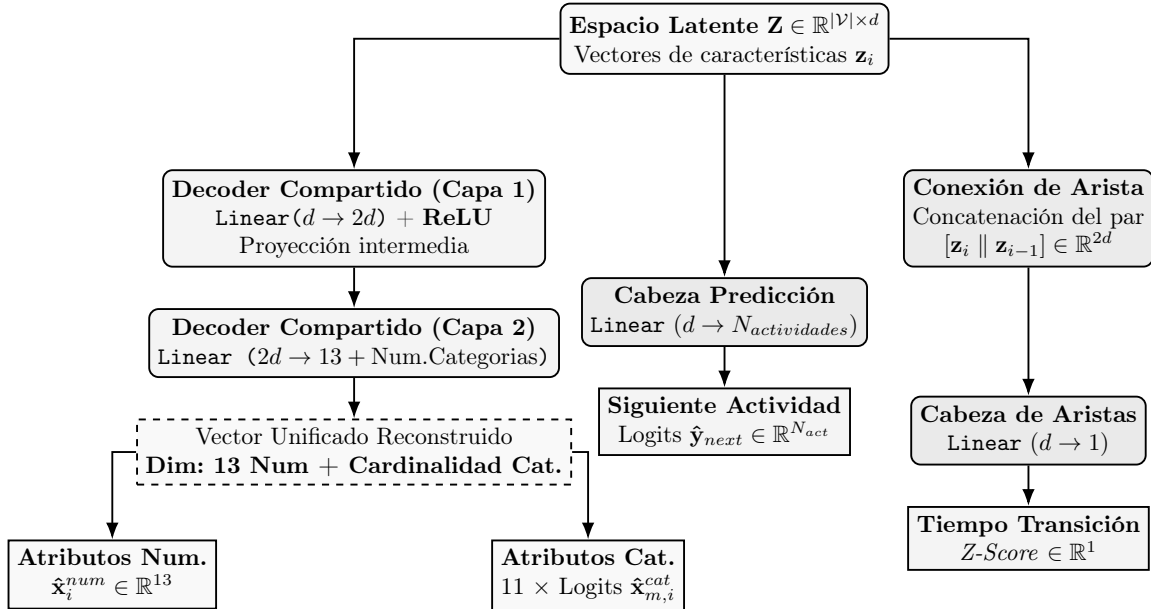


Figura 3.15: Arquitectura del decoder para la reconstrucción del grafo.

Un aspecto importante de las salidas categóricas y de siguiente actividad es el uso de *logits*. En el ámbito del aprendizaje profundo, los *logits* son los valores numéricos crudos y sin normalizar generados directamente por la última capa lineal de una red antes de aplicar cualquier función de activación que los convierta en probabilidades (como la función *Softmax*). Matemáticamente, representan puntuaciones en un rango continuo; a mayor valor de un *logit*, mayor es la afinidad o confianza del modelo hacia esa categoría concreta.

La decisión de que el modelo devuelva directamente *logits* en lugar de probabilidades normalizadas se debe a que la función de pérdida empleada en el entrenamiento está diseñada para recibir estos valores en bruto.

3.4.7. Entrenamiento del modelo

La estrategia de entrenamiento de la arquitectura propuesta se centra en optimizar simultáneamente las cuatro tareas del decodificador: reconstrucción numérica (L_{num} , mediante MSE), reconstrucción categórica (L_{cat} , mediante Entropía Cruzada), predicción estructural (L_{pred} , mediante Entropía Cruzada) y reconstrucción temporal de las aristas (L_{edge} , mediante MSE).

Durante el desarrollo de la solución, se constató el riesgo de que la red aprendiese la función de identidad trivial si la dimensión del espacio latente es suficientemente grande. Para mitigarlo, se incorporó un mecanismo de enmascaramiento estocástico (*masking*) sobre un porcentaje de las entradas, forzando al modelo a inferir la información faltante a partir del contexto topológico. La optimización de hiperparámetros de entrenamiento se llevó a cabo utilizando técnicas y regularizadores estándar de aprendizaje profundo.

Finalmente, para capturar distintos niveles de profundidad semántica y representaciones de variada complejidad, el proceso de entrenamiento no se limitó a una única configuración. Se entrenó un conjunto de modelos independientes variando la dimensión del espacio latente ($d \in \{8, 16, 32, 64, 128, 256\}$). La dimensión del espacio latente tiene un impacto crucial en la detección de anomalías, como se verá en la siguiente subsección.

3.4.8. Evaluación y resultados

Una vez entrenados los modelos, la fase de inferencia y detección de anomalías se basa en evaluar cada secuencia del conjunto de *test* por el modelo.

Para cada grafo de entrada $\mathcal{G}$ evaluado en modo de inferencia, el decodificador calcula los errores de reconstrucción. Esto permite aislar los errores de distinto tipo:

- **Error Numérico** ($e_{num,i}$): Error Cuadrático Medio (*MSE*) sobre el vector de 13 variables continuas:

$$e_{num,i} = \frac{1}{13} \sum_{k=1}^{13} (\hat{x}_{i,k}^{num} - x_{i,k}^{num})^2$$

- **Error Categórico** ($e_{cat,i}$): Promedio de la Entropía Cruzada (*CE*) de las 11 variables

de negocio, según la ecuación (2.4):

$$e_{cat,i} = \frac{1}{11} \sum_{m=1}^{11} \text{CE} \left(\hat{\mathbf{x}}_{m,i}^{cat}, cat_m^{(i)} \right)$$

- **Error Estructural** ($e_{pred,i}$): Entropía Cruzada sobre los *logits* de la predicción del flujo secuencial:

$$e_{pred,i} = \text{CE} (\hat{\mathbf{y}}_{next,i}, y_{next,i})$$

- **Error Temporal** ($e_{edge,ij}$): Error Cuadrático Medio sobre la reconstrucción del *Z-score* de la arista:

$$e_{edge,i} = (\hat{a}_i - a_i)^2$$

Para obtener una métrica representativa a nivel de solicitud completa, se extraen simultáneamente el valor máximo (máx) y el valor promedio (mean) de cada tipo de error a lo largo de todos los componentes del grafo. El uso del máximo es crítico para identificar anomalías puntuales, mientras que el promedio evalúa la degradación general de la traza.

Para cada solicitud evaluada, el modelo genera un vector de características de error **Error** $\in \mathbb{R}^8$, compuesto por los 4 tipos de pérdida del decodificador evaluados bajo 2 niveles de agregación (el error máximo y el error medio de la secuencia):

$$\mathbf{Error} = [e_{num}^{max}, e_{num}^{avg}, e_{cat}^{max}, e_{cat}^{avg}, e_{pred}^{max}, e_{pred}^{avg}, e_{edge}^{max}, e_{edge}^{avg}]^T$$

Por último, se establece un umbral para detectar anomalías de forma muy simple. Para cada uno de los 4 tipos de errores y 2 niveles de agregación (máximo y media), se calcula el cuantil 0.998. El valor del cuantil se seleccionó de forma empírica tras evaluar distintos umbrales, al proporcionar un compromiso adecuado entre la sensibilidad para detectar comportamientos anómalos y la reducción de falsos positivos. Un caso será considerado anómalo si alguna de sus componentes del error total supera este umbral, para alguna de las dimensiones latentes consideradas.

Anomalía	Categórico		Numérico	
	Max	Mean	Max	Mean
Anomaly1	16, 32, 64, 128, 256	16, 32, 64, 128, 256	8, 16, 32, 64, 128, 256	8, 16, 32, 64, 128, 256
Anomaly2	×	×	8, 16, 32, 64	8, 16, 32, 64
Anomaly3	×	×	8, 16, 32, 64, 128, 256	8, 16, 32, 64, 128, 256
Anomaly4	×	×	×	×
Anomaly5	64, 128	128	×	×
Anomaly6	64, 128	8, 16, 32, 64, 128, 256	×	×
Anomaly7	×	×	×	×
Anomaly8	×	×	×	×
Anomaly9	×	×	×	×
Anomaly10	8, 64, 128, 256	8, 64, 128, 256	×	×
Anomaly11	8, 16, 32, 64, 128, 256	128, 256	×	×
Anomaly12	×	×	×	×
Anomaly13	×	128	×	×
Anomaly14	×	×	×	×
Anomaly15	128, 256	128, 256	×	×

Tabla 3.7: Detección de anomalías en base a la frontera de decisión cuantílica ($Q = 0.998$) para las variables de atributos. Los valores numéricos indican la dimensión del espacio latente (d) en la que el caso logró atravesar el escudo de la métrica correspondiente. El símbolo $\times$ indica que no fue detectada.

Anomalía	Estructural		Temporal	
	Max	Mean	Max	Mean
Anomaly1	16, 32	16, 32, 64, 256	×	×
Anomaly2	×	×	×	×
Anomaly3	×	×	×	×
Anomaly4	×	×	×	×
Anomaly5	16, 64, 128, 256	16, 64, 128, 256	×	×
Anomaly6	×	16	×	×
Anomaly7	×	×	8, 16, 32, 64, 128, 256	8, 16, 32, 64, 128, 256
Anomaly8	×	×	×	×
Anomaly9	×	×	×	×
Anomaly10	8, 16, 32, 64, 128, 256	8, 16, 32, 64, 128, 256	×	×
Anomaly11	8, 16, 32, 64, 128, 256	8, 32, 64, 128, 256	×	×
Anomaly12	×	×	×	×
Anomaly13	×	×	×	×
Anomaly14	64, 128, 256	×	×	×
Anomaly15	16, 64	×	×	×

Tabla 3.8: Detección de anomalías en base a la frontera de decisión cuantílica ($Q = 0.998$) para las variables topológicas y de flujo. Los valores numéricos indican la dimensión del espacio latente (d) en la que el caso logró atravesar el escudo de la métrica correspondiente. El símbolo $\times$ indica que no fue detectada.

Los resultados presentados en las Tablas 3.7 y 3.8 permiten extraer tres conclusiones sobre el comportamiento del modelo.

Bajo el umbral estricto del cuantil 0.998, la arquitectura no logró clasificar como anómalos los casos *Anomaly4*, *Anomaly8*, *Anomaly9* y *Anomaly12* en ninguna de sus dimensiones. Su detección requeriría rebajar el umbral de alerta, asumiendo un mayor coste de falsos positivos, o incorporar variables de negocio adicionales.

En segundo lugar, se aprecia el papel de la dimensión del espacio latente (d). Mientras que los fraudes más groseros o evidentes (como *Anomaly1* o *Anomaly10*) logran romper el escudo cuantílico en la práctica totalidad de las configuraciones, las desviaciones más complejas son altamente dependientes del nivel de compresión. Un ejemplo es *Anomaly13*, que únicamente es detectado por la métrica categórica media cuando el espacio latente posee una dimensionalidad holgada ($d = 128$). Esto valida la propuesta metodológica de evaluar el proceso bajo distintos niveles de profundidad semántica.

Finalmente, el diseño multi-tarea otorga al modelo cierto grado de explicabilidad. Observar qué rama específica del decodificador supera el umbral de normalidad alerta al auditor no solo de la existencia de una anomalía, sino de su naturaleza probable (por ejemplo, el carácter estrictamente temporal de *Anomaly7* o la desviación exclusivamente numérica de *Anomaly2*).

No obstante, esta explicabilidad analítica es orientativa. Debido a la naturaleza del modelo y al mecanismo de paso de mensajes de las capas GATv2, la información fluye iterativamente

a través de las aristas. Esto implica que una alteración severa en el atributo categórico de un evento temprano puede propagarse y distorsionar el espacio latente de sus vecinos sucesores, disparando de forma indirecta el error estructural o numérico. Por lo tanto, el tipo de error reportado ofrece una pista del motivo de la anomalía, pero el análisis definitivo se debe abordar con la herramienta descriptiva.

3.4.9. Entrega visual

Volviendo al punto de vista de la entidad financiera, la herramienta de detección de anomalías se entrega como un panel visual e interactivo, como el de la Figura 3.16. Esta permite alternar entre el nivel de agregación y las distintas dimensiones latentes estudiadas. Además, se representa en un espacio 3D, ofreciendo la posibilidad de cambiar el eje Z entre el error categórico y de aristas, de forma que cada eje representa una dimensión del error.

En este caso, las anomalías inyectadas se muestran con rombos, mostrando el identificador del caso al pasar el cursor sobre ellas. Por último, el escudo azul tiene fines únicamente visuales, construido como un elipsoide a partir de los cuantiles 0.998 de cada eje.

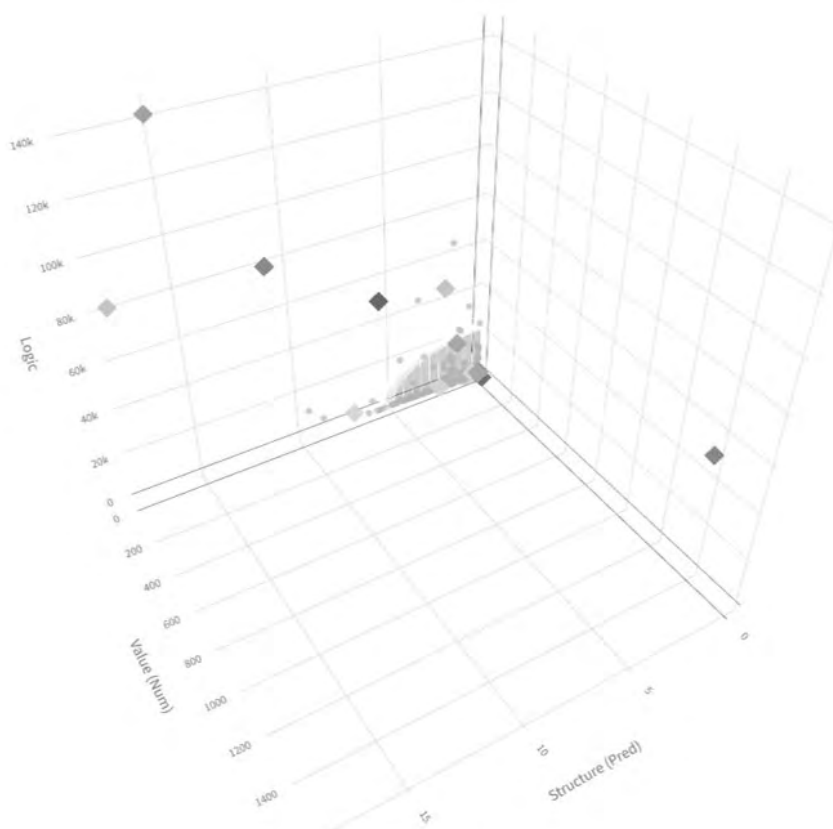


Figura 3.16: Panel interactivo para la detección de anomalías.

3.5. Análisis predictivo mediante Transformers

En la sección anterior se abordó la detección de desviaciones respecto al comportamiento normal del proceso. Ahora, en lugar de identificar lo que ya ocurrió de forma anómala, se pretende anticipar lo que aún no ha sucedido. En concreto, dado el historial parcial de una solicitud de préstamo en curso, se busca inferir simultáneamente cuál será su resolución final, qué actividad se ejecutará a continuación, cuánto tiempo restará hasta la finalización del caso y si una oferta concreta será aceptada por el cliente.

Desde la perspectiva de la entidad financiera, la capacidad de predecir estos aspectos en tiempo real permite optimizar la asignación de recursos humanos, priorizar la atención sobre solicitudes con alta probabilidad de éxito comercial y detectar de forma temprana aquellas que se encaminan hacia un desenlace desfavorable. Un sistema de este tipo complementa las herramientas descriptivas del *Process Mining* clásico, transformando el análisis retrospectivo en una herramienta de soporte a la decisión operativa.

3.5.1. Formulación del problema

Dada una solicitud incompleta, se considera la secuencia de actividades desde su comienzo hasta el momento actual. El objetivo es producir simultáneamente cuatro predicciones:

1. **Resultado del caso** (*Outcome*): $\hat{y}^{out} \in \{\text{Granted, Cancelled, Denied}\}$.
2. **Siguiente actividad** (*Next Activity*): $\hat{y}^{next} \in \mathcal{A} \cup \{\text{FIN}\}$, donde $\mathcal{A}$ es el vocabulario de actividades.
3. **Tiempo restante** (*Remaining Time*): $\hat{y}^{time} \in \mathbb{R}^+$, expresado en escala logarítmica.
4. **Selección de oferta** (*Selected*): $\hat{y}^{sel} \in [0, 1]$, solo aplicable cuando el evento es del tipo oferta (0_-).

Las tareas seleccionadas responden a distintos objetivos de interés para la gestión del proceso de concesión de préstamos. La predicción del resultado final permite anticipar el desenlace esperado de cada solicitud y facilitar la priorización de recursos. La predicción de la siguiente actividad y del tiempo restante proporcionan soporte a la monitorización del proceso, permitiendo detectar desviaciones respecto al comportamiento habitual y anticipar posibles retrasos. Finalmente, la predicción de la aceptación de una oferta aporta una herramienta de apoyo a la decisión, al estimar la probabilidad de aceptación por parte del cliente y facilitar la adaptación de las condiciones ofertadas.

Todas estas tareas comparten una característica común: dependen del historial completo del caso y no únicamente de los eventos más recientes. En consecuencia, requieren modelos capaces de capturar dependencias de largo alcance entre eventos y combinar información procedente de distintos momentos de la traza.

En este contexto, las arquitecturas Transformer constituyen una opción adecuada gracias a sus mecanismos de autoatención, que permiten modelar de forma directa las relaciones entre cualquier par de eventos de la secuencia y procesar eficientemente grandes volúmenes de datos

mediante entrenamiento paralelo. Por ello, en este trabajo se propone un modelo basado en un *Transformer Encoder* capaz de procesar el prefijo de una traza y generar simultáneamente las cuatro predicciones descritas.

3.5.2. Preparación de los datos y generación de secuencias truncadas

Al igual que para la herramienta de detección de anomalías, es necesario realizar una preparación previa de los datos. Aunque la información de partida es la misma, la representación utilizada difiere de la empleada en la detección de anomalías, ya que el modelo opera directamente sobre secuencias de eventos en lugar de grafos. Partiendo del *dataset* tabular sin nulos construido en la anterior sección, cada evento se representa mediante la concatenación de dos vectores: uno categórico $\mathbf{x}_i^{cat}$ y otro numérico $\mathbf{x}_i^{num}$. Las variables empleadas son:

Variables categóricas (9)	Variables numéricas (22)
concept:name	case:RequestedAmount
lifecycle:transition	OfferedAmount
org:resource	MonthlyCost
case:LoanGoal	NumberOfTerms
case:ApplicationType	FirstWithdrawalAmount
Action	devolution_amount
EventOrigin	interest_amount, interest_rate
Accepted	log_delta_time, log_elapsed_time
offer_rank	sin/cos_hour, sin/cos_weekday sin/cos_day, sin/cos_month, year ratio_offer_request diff_monthly_cost log_daily_workload

Tabla 3.9: Variables utilizadas como entrada del modelo predictivo.

Nótese la exclusión deliberada de la variable **CreditScore**, presente en el sistema de detección de anomalías. Esta decisión se adopta para prevenir la fuga de datos (*data leakage*), ya que durante el análisis exploratorio se observó que ninguna solicitud con **CreditScore** >0 finalizaba en un estado distinto al de concesión. Este patrón sugiere que esta variable podría incorporar información futura del caso. Al no conocer el proceso exacto de cálculo de la puntuación, se decide omitirla.

Por otra parte, mediante ingeniería de variables, se generan algunos atributos derivados

con el fin de mejorar el rendimiento del modelo. El cociente entre el importe ofrecido y el solicitado (**ratio_offer_request**) resume el grado de ajuste entre la oferta realizada y las expectativas iniciales del cliente, por lo que puede resultar informativo tanto para predecir la aceptación de la oferta como el resultado final de la solicitud. Asimismo, la diferencia entre la cuota mensual de la oferta y una aproximación de la cuota teórica obtenida al dividir el importe solicitado entre el número de plazos (**diff_monthly_cost**) proporciona una medida del coste relativo de las condiciones ofertadas, pudiendo influir en la decisión del cliente. La variable **log_daily_workload**, calculada como el número acumulado de eventos de todos los casos registrados durante la jornada hasta el instante considerado, se incorpora como un *proxy* de la carga operativa de la entidad.

A diferencia de las particiones aleatorias convencionales (usada en la anterior sección), se emplea un split cronológico que respeta la causalidad temporal del proceso. Los casos se ordenan por su fecha de inicio y se asignan secuencialmente: los más antiguos para entrenamiento, los intermedios para validación y los más recientes para test. Esta estrategia garantiza que la evaluación simula un despliegue real donde el modelo debe predecir sobre solicitudes temporalmente posteriores a las empleadas para su ajuste.

El procedimiento de generación de secuencias incompletas corresponde al ilustrado en la Figura 3.17. Para una traza completa de N eventos, se generan $N - 1$ secuencias truncadas llamadas prefijos (no se considera la secuencia completa). El resultado es un total de 1 166 908 prefijos construidos, sumando 27 805 801 eventos.

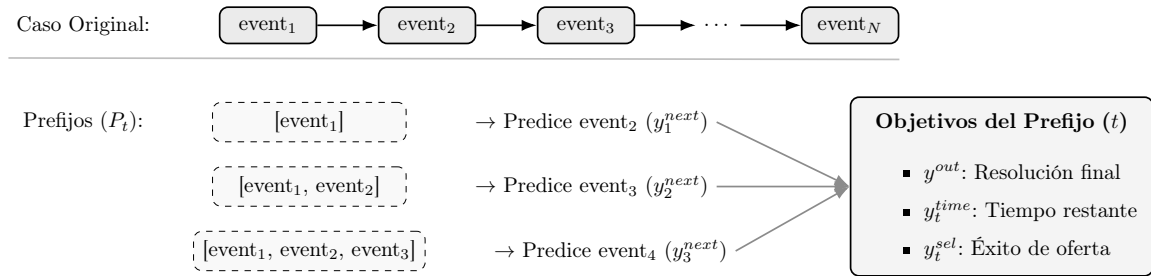


Figura 3.17: Mecanismo de extracción de prefijos secuenciales y su mapeo hacia objetivos inmediatos (y^{next}) e históricos (y^{out} , y^{time} , y^{sel}).

3.5.3. Arquitectura del Transformer Multitarea

El objetivo de la arquitectura a montar es clara: aprovechar la potencia de los Transformers para aprender representaciones vectoriales de la secuencia útiles para la predicción. Esta representación debe capturar y entender muy bien la esencia de la solicitud a partir de sus eventos y atributos, para posteriormente alimentar una red neuronal que tome este vector como entrada para predecir aquello que nos interesa.

A diferencia del enfoque adoptado en la detección de anomalías (Sección 3.4), las dimensiones de los *embeddings* categóricos se establecen de forma dinámica según la ecuación:

$$d_{\text{embedding}} = \min \left(50, \left\lfloor \frac{C + 1}{2} \right\rfloor \right)$$

donde C denota la cardinalidad de la variable. Esta expresión constituye una heurística ampliamente utilizada en aprendizaje profundo, busca asignar una mayor dimensionalidad a las variables con alta variabilidad para explotar al máximo su capacidad predictiva.

El flujo detallado de dimensiones, transformaciones y operaciones matemáticas se condensa visualmente en el diagrama de la Figura 3.18. Como fases de esta arquitectura, se destacan:

- **Proyección de atributos:** Las 22 variables numéricas escaladas atraviesan un MLP (GELU y *LayerNorm*) para extraer relaciones complejas, y se concatenan con los *embeddings* categóricos. El vector combinado se proyecta linealmente hacia un espacio latente de dimensión $d_{model} = 64$. Este valor concreto corresponde a la dimensión del espacio latente que mejor resultado ofreció según el rendimiento obtenido en el conjunto de validación.
- **Inyección Posicional Secuencial:** Se añade una codificación posicional aprendible (p_i) por suma directa para capturar el orden cronológico estricto de los prefijos.
- **Mecanismo de Pooling Indexado:** Tras procesar la traza con dos capas basadas en la atención multi-cabeza ($nhead = 4$) definida en la ecuación (2.3), se extrae exclusivamente el vector latente del último token real activo de la secuencia ($\mathbf{z}_{last}$). Debido a la naturaleza contextual del Transformer, este vector integra información contextual procedente de toda la secuencia.

En nuestros experimentos dos capas proporcionaron un equilibrio adecuado entre capacidad de representación y coste computacional.

- **Decodificación en Cabezas Lineales:** El vector $\mathbf{z}_{last}$ se bifurca en paralelo hacia cuatro capas lineales independientes encargadas de emitir de forma simultánea los *logits* y escalares de las tareas de predicción.

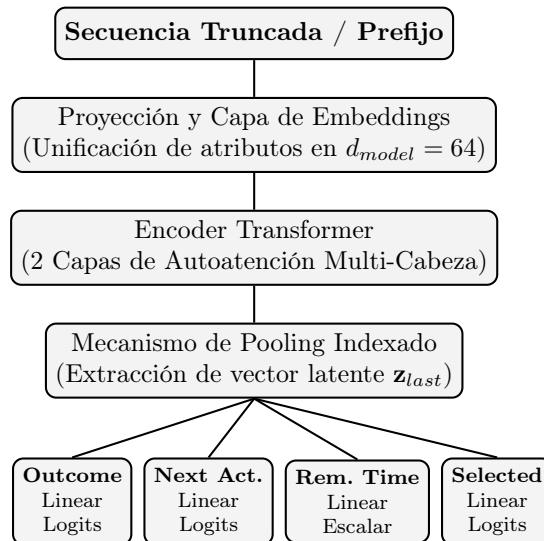


Figura 3.18: Flujo arquitectónico del Transformer Multitarea desde la recepción del prefijo secuencial hasta las cuatro cabezas predictoras.

Una cuestión natural que puede surgir es el porqué de un único transformer multitarea. Es decir, el vector latente generado $\mathbf{z}_{last}$ contiene el mejor resumen del caso para alimentar las 4 cabezas de predicción que vienen después.

Una alternativa válida, sería realizar la división en cuatro tareas desde el comienzo, entrenando cuatro transformers independientes cada uno especializado en una tarea. Sin embargo, una arquitectura multitarea puede enriquecer el modelo mediante la transferencia inductiva cuando las tareas están suficientemente relacionadas, permitiendo que objetivos como la predicción de la siguiente actividad y del tiempo restante, compartan conocimiento en el espacio latente y establezcan mutuamente su aprendizaje. Por otra parte, el coste computacional se multiplica por cuatro en caso de entrenar todas las ramas.

3.5.4. Entrenamiento y evaluación

Para el ajuste y optimización de la arquitectura, las cuatro tareas predictivas se entrenan de manera simultánea mediante una única función de pérdida global fundamentada en la ecuación (2.6). Los errores individuales evaluados durante el entrenamiento se estructuran de la siguiente manera:

- **Error de Resolución ($\mathcal{L}_{out}$) y Siguiente Actividad ($\mathcal{L}_{next}$):** Se calculan mediante funciones de Entropía Cruzada (*Cross-Entropy*, ec. (2.4)) con *label smoothing* (2.5), midiendo la desviación en las predicciones categóricas del flujo y la resolución final del caso.
- **Error de tiempo restante ($\mathcal{L}_{time}$):** Se calcula a través de la función (*Huber Loss*), una métrica de regresión más robusta a valores extremos en comparación con el MSE.
- **Error de selección de oferta ($\mathcal{L}_{sel}$):** Se evalúa mediante Entropía Cruzada Binaria con Logits (*BCEWithLogitsLoss*), penalizando los fallos al anticipar el éxito comercial de una propuesta.

Los coeficientes de ponderación de la función de pérdida total se ajustan empíricamente con el fin de equilibrar la contribución de cada tarea durante el entrenamiento. En este trabajo se utilizan los valores $\mu_{out} = 1$, $\mu_{next} = 1$, $\mu_{time} = 0,5$ y $\mu_{sel} = 1,5$, otorgando un mayor peso a la predicción de la aceptación de ofertas y reduciendo la influencia de la tarea de regresión. Además, esta parametrización proporciona un mecanismo sencillo para adaptar el modelo a distintos escenarios de aplicación, permitiendo priorizar unas tareas frente a otras.

Puesto que la aceptación o rechazo de una oferta solo tiene sentido cuando se emite una propuesta económica, la rama de selección de oferta requiere un mecanismo de control. Para ello, se introduce una máscara *booleana* basada en el atributo de origen del evento. Esta máscara actúa como un filtro estricto durante la retropropagación: anula por completo el error en los eventos de tramitación ordinaria y obliga al Transformer a optimizar la pérdida comercial exclusivamente sobre aquellos tokens identificados como ofertas reales (`EventOrigin = 'Offer'`).

Una vez completado el entrenamiento, la evaluación final se ejecuta sobre el conjunto de test

aislado cronológicamente, garantizando que el modelo prediga sobre solicitudes temporalmente posteriores para simular un despliegue real.

El rendimiento final en la fase de test se cuantifica mediante las siguientes métricas sectorizadas:

- **Clasificación:** Se extrae el *Accuracy* global y reportes de clasificación detallados (Precisión, *Recall* y *F1-Score*) tanto para el desenlace del caso como para la predicción de la siguiente actividad.
- **Regresión Temporal:** El error del tiempo restante se revierte de la escala logarítmica y se transforma a días reales, mostrando el Error Absoluto Medio (MAE).

3.5.5. Resultados y visualización

Una vez completada la fase de inferencia sobre el conjunto de test cronológico, se procede a analizar el rendimiento de las distintas configuraciones del Transformer Multitarea. En la Tabla 3.10 se expone el comportamiento global del sistema comparando las dimensiones de espacio latente estudiadas ($d_{model} \in \{64, 128\}$) tanto bajo un enfoque general de traza completa (*Global*) como evaluando únicamente en el momento de creación de la oferta (*Snapshot: O_Create Offer*), algo interesante desde el punto de vista de negocio.

Configuración	Outcome Acc.	Next Act. Acc.	Time MAE (días)	Selected AUC	Selected <i>F1</i>
$d_{model} = 64$	0.6863	0.9082	7.06	0.8263	0.7958
$d_{model} = 64$ Offer	0.5836	1.0000	9.34	0.6538	0.6941
$d_{model} = 128$	0.6858	0.9039	7.55	0.8120	0.7909
$d_{model} = 128$ Offer	0.5811	1.0000	10.18	0.6301	0.6910

Tabla 3.10: Tabla resumen comparativa de rendimiento multitarea en el conjunto de test.

A partir de los resultados, se extrae una primera conclusión: la arquitectura compacta ($d_{model} = 64$) supera al modelo extendido ($d_{model} = 128$) en todas las métricas evaluadas. Este comportamiento indica que un espacio latente excesivamente holgado induce un ligero sobreajuste (*overfitting*) sobre las dinámicas del conjunto de entrenamiento, mientras que la dimensión restrictiva actúa como un regularizador natural que favorece la generalización.

Para comprender las fortalezas y los límites del modelo definitivo ($d_{model} = 64$), se desglosan los resultados analíticos indexados por sus respectivas cabezas de predicción:

1. **Predicción de Siguiente Actividad:** La cabeza de flujo registra una precisión global sobresaliente del 90.82 % a nivel global. Al evaluar el escenario de *Snapshot* en **O_Create Offer**, el *Accuracy* converge a un perfecto 1.0000 (100 %). Este comportamiento es completamente coherente con la semántica del registro de eventos donde, tras la actividad de creación de oferta, el flujo operativo del banco exige de forma determinista la ejecución inmediata de **O_Created**, validando la capacidad de la red para detectar este tipo de transiciones obligatorias.

2. **Resolución del Caso (Outcome) y Sesgo de Datos:** La clasificación del desenlace decae desde un 68.63 % global hasta un 58.36 % en el punto de la oferta. De hecho, si calculamos la Accuracy balanceada esta métrica cae hasta cerca del 43 %. Al inspeccionar el reporte detallado de clasificación expuesto en la Tabla 3.11, se evidencia un fuerte desbalanceo en el aprendizaje.

Dado que las solicitudes concedidas (**Granted**) constituyen la clase mayoritaria (aproximadamente el 65 % del conjunto de test), el Transformer desarrolla un sesgo adaptativo hacia esta categoría, alcanzando un excelente *Recall* de hasta el 96 %. Sin embargo, esta inercia penaliza drásticamente la detección de las clases minoritarias (**Cancelled** y **Denied**), cuyas tasas de recuperación caen a mínimos operativos debido a la sutil diferencia estructural que presentan estos casos en etapas tempranas.

3. **Regresión del Tiempo Restante:** El decaimiento del error absoluto medio (MAE) refleja la acumulación de contexto a lo largo del ciclo de vida del proceso. En la fase inicial del prefijo (*Snapshot*), la incertidumbre temporal es elevada, registrando una desviación media de 9.34 días. A nivel global, a medida que la traza consume eventos y estabiliza su flujo, el MAE se reduce significativamente hasta los 7.06 días de margen de error, consolidando una herramienta valiosa para la estimación de plazos.
4. **Selección de la oferta (Selected Offer):** A nivel global, el modelo predice la aceptación de ofertas con un notable AUC-ROC de 0.8263 y un *F1* de 0.7958. En contraste, en el instante exacto de la creación (*Snapshot*), el AUC-ROC desciende a 0.6538. Esta degradación se debe a que, en el momento de emitir la oferta, el Transformer solo dispone de los atributos estáticos de la solicitud, careciendo de contexto futuro (interacciones del cliente, renegociaciones o tiempos de respuesta) que se genera en los pasos posteriores y que es crítico para anticipar el éxito comercial.

Clase	GLOBAL (175,611 muestras)			Snapshot: 0_Create Offer (6,355 muestras)		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Cancelled	0.56	0.22	0.32	0.44	0.21	0.29
Denied	0.72	0.12	0.21	0.57	0.04	0.07
Granted	0.70	0.96	0.81	0.61	0.90	0.73

Tabla 3.11: Desglose detallado de precisión, recall y F1-score para la resolución del caso ($d_{model} = 64$).

A partir de las métricas anteriores, se puede concluir que la herramienta no debe ser empleada como un algoritmo de decisión, sino de recomendación para la entidad financiera. En particular, preocupa la toma de decisiones sobre la resolución del préstamo. Es por ello que se establece un sistema de predicción basado en margen para reflejar el nivel de seguridad que tiene el modelo al hacer la predicción.

Fijado un umbral $\tau \in [0, 1]$, la herramienta solo mostrará la predicción del **Outcome_case** si la distancia entre la probabilidad otorgada a la clase más probable y la segunda más probable

supera dicho umbral. Las consecuencias de aumentar el umbral tiene un impacto positivo sobre la accuracy, a coste de disminuir el número de predicciones, tal y como se ve en la Figura

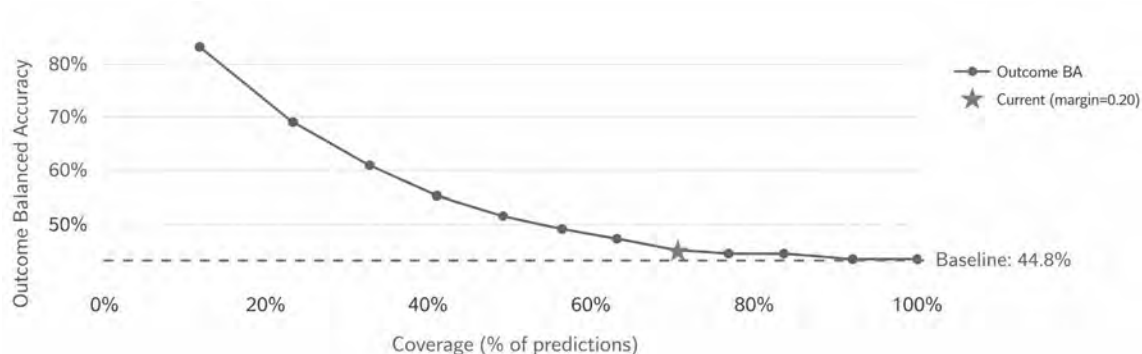


Figura 3.19: Relación entre porcentaje de predicciones realizadas y *accuracy* balanceada para distintos umbrales τ .

Finalmente, la herramienta de predicción se presenta mediante un panel interactivo que permite monitorizar en tiempo real la evolución temporal y probabilística de cada solicitud individual. Como se observa en la Figura 3.20, la interfaz muestra los cuatro valores a predecir.

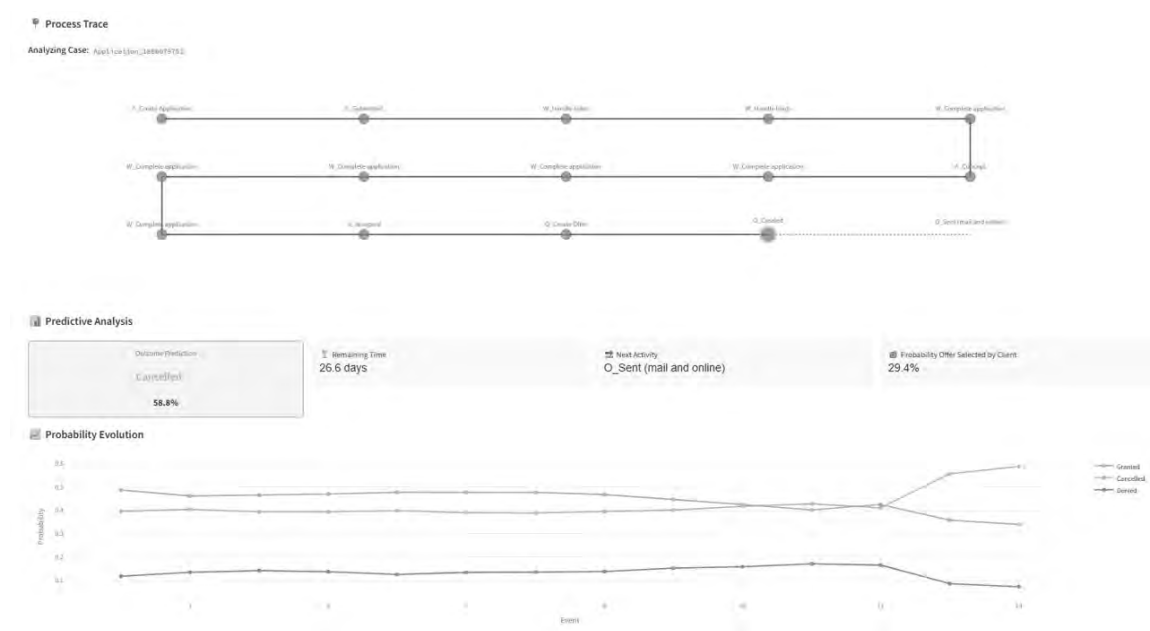


Figura 3.20: Panel interactivo de analítica predictiva multitarea para la monitorización en tiempo real e inspección del histórico de probabilidades de una solicitud de préstamo (Application_1806079752).

En el bloque superior, se representa de forma gráfica la traza del proceso operativo recorrido

por el expediente hasta el momento actual (`Application_1806079752`), destacando el último hito ejecutado (`0_Created`) y proyectando mediante una línea discontinua el paso inmediato estimado por la cabeza de flujo (`0_Sent (mail and online)`). El bloque central arroja las predicciones del modelo para el evento actual: la resolución más probable (un desenlace de cancelación con el 58.8%), el tiempo restante estimado para el fin del caso (26.6 días) y la probabilidad de éxito comercial de la oferta generada (29.4%).

El componente inferior, centrado en la evolución histórica de las probabilidades, constituye el núcleo de mayor valor operativo para la toma de decisiones proactivas dentro de la entidad financiera. En lugar de ofrecer una predicción estática en el instante presente, el gráfico de líneas permite auditar la trayectoria de convicción del Transformer a lo largo de los 14 eventos del ciclo de vida del caso.

Esta perspectiva dinámica revela un cambio de tendencia crítico para el negocio. Durante las primeras etapas del expediente (eventos 1 al 12), el modelo mantenía un diagnóstico optimista, situando la probabilidad de concesión (`Granted`) de forma dominante por encima de las demás opciones. Sin embargo, al alcanzar los eventos asociados a la oferta económica (`0_Create Offer` y `0_Created`), las curvas sufren un punto de inflexión y se cruzan drásticamente: la probabilidad de cancelación se dispara hasta el 58.8%, mientras que la de éxito se desploma.

Para la entidad bancaria, visualizar este declive en tiempo real proporciona una ventana de oportunidad analítica inestimable. Un auditor puede identificar exactamente en qué momento y ante qué condiciones financieras el caso comenzó a deteriorarse, permitiendo intervenir de forma proactiva, como contactar inmediatamente al cliente para renegociar las condiciones de la propuesta, mucho antes de que se formalice el desenlace desfavorable.

3.5.6. Breve comparativa

En el trabajo de Teinmaa et al. (2019) sobre monitorización predictiva de procesos, se evalúa el rendimiento de diversas técnicas utilizando, entre otros, el *BPI Challenge 2017*. En dicho estudio, el problema se simplifica dividiéndolo en múltiples tareas de clasificación binaria; es decir, se entrena un modelo independiente para predecir exclusivamente si un préstamo es concedido o no, aislando este estado del resto de posibles resultados finales.

En contraste, en la presente memoria se ha optado por abordar el problema mediante un enfoque de clasificación multiclase, entrenando un único modelo capaz de discriminar simultáneamente entre los tres posibles estados finales del proceso. Esta formulación exige al algoritmo optimizar las fronteras de decisión considerando todas las clases a la vez. Adicionalmente, en dicho *benchmark* no eliminan deliberadamente variables como el *credit score* por riesgo a elevar las métricas a través de fuga de datos.

El modelo propuesto en este trabajo alcanza un F-score de 0.73 para el estado concedido. Si comparamos este resultado con las métricas del estudio de referencia expuestas en la Tabla 3.12, se observa que los mejores modelos para la tarea binaria apenas logran un F-score máximo de 0.72.

Superar estas métricas de referencia bajo condiciones más estrictas pone de manifiesto la robustez del enfoque propuesto. Este hecho motiva la realización de un estudio comparativo

más riguroso replicando las mismas condiciones iniciales de la referencia.

Método / Codificación	XGBoost	Random Forest	Logística	SVM
single_laststate	0.66	0.70	0.67	0.14
single_agg	0.71	0.71	0.67	0.67
knn_laststate	0.66	0.65	0.59	0.28
knn_agg	0.59	0.65	0.61	0.06
state_laststate	0.70	0.70	0.67	0.11
state_agg	0.70	0.72	0.68	0.58
cluster_laststate	0.69	0.71	0.67	0.07
cluster_agg	0.70	0.71	0.67	0.13
prefix_index	0.72	0.72	0.68	0.00
prefix_laststate	0.70	0.70	0.67	0.34
prefix_agg	0.71	0.71	0.69	0.01

Tabla 3.12: Valores F-score para el conjunto de datos bpic2017_1 sobre la predicción binaria de si el estado final corresponde a préstamo concedido

3.6. Conclusiones

Con la ejecución de este caso de estudio se consolida el desarrollo de una solución orientada a la optimización operativa en el sector financiero, estructurada bajo el marco estructurado e iterativo de la metodología CRISP-DM. A través de este enfoque, se ha completado con éxito el diseño y análisis de los tres pilares analíticos fijados originalmente como requisitos esenciales por la entidad: el análisis descriptivo clásico, el análisis proactivo mediante la detección no supervisada de anomalías y el análisis predictivo multitarea basado en arquitecturas Transformer.

Es importante recalcar que el propósito primordial de este trabajo no ha sido realizar una comparación exhaustiva ni un *benchmarking* formal frente a la literatura existente sobre el conjunto de datos del *BPI Challenge 2017*, ni tampoco reproducir los objetivos del desafío original. No obstante, contrastar de manera rigurosa los rendimientos obtenidos con otros modelos y variaciones se plantea como una continuación natural para este trabajo.

Finalmente, aunque esta sección cierra satisfactoriamente los bloques descriptivo, proactivo y predictivo del proceso de gestión de préstamos, la investigación deja abierta la transición hacia el análisis prescriptivo como la extensión futura idónea. El uso de técnicas de simulación avanzada permitiría evaluar cuantitativamente las consecuencias de tomar ciertas decisiones operativas antes de su ejecución real. De este modo, el sistema no solo describe, detecta y predice la realidad del proceso, sino que evoluciona hacia una herramienta activa capaz de proponer vías de acción óptimas y guiar la toma de decisiones estratégicas de la entidad financiera, encontrándose plenamente respaldada por la evidencia empírica de los datos.

Capítulo 4

Síntesis y extensiones

4.1. Síntesis del trabajo

La principal aportación de este trabajo consiste en integrar técnicas clásicas de Process Mining con arquitecturas modernas de aprendizaje profundo dentro de un flujo analítico único, capaz de combinar análisis descriptivo, detección de anomalías y monitorización predictiva.

La materialización práctica de estos fundamentos se ha llevado a cabo sobre el conjunto de datos del *BPI Challenge 2017*, un registro real de 31 509 solicitudes de préstamo con más de 1.2 millones de eventos y 15 930 variantes de ejecución. Siguiendo la metodología CRISP-DM, se han desarrollado tres pilares analíticos complementarios. Estos tres enfoques cubren las principales necesidades analíticas de un proceso de negocio: comprender su comportamiento, detectar desviaciones y anticipar su evolución.

El análisis descriptivo, implementado mediante algoritmos clásicos de *Process Mining*, ha permitido caracterizar el comportamiento del proceso e identificar sus principales patrones y fuentes de variabilidad. El análisis proactivo, basado en un autoencoder con capas GATv2 que modela cada solicitud como un grafo dirigido, ha demostrado la capacidad de detectar anomalías de naturaleza diversa (estructurales, temporales, de atributos) sin requerir etiquetas de supervisión, evaluándose sobre un catálogo de 15 anomalías sintéticas diseñadas específicamente para cubrir distintas dimensiones del comportamiento anómalo. Finalmente, los resultados obtenidos en el análisis predictivo, articulado mediante un Transformer Encoder multitarea, muestran el potencial de estas arquitecturas para la monitorización predictiva de procesos reales.

Más allá de su contribución académica, los artefactos metodológicos desarrollados en este TFM han servido directamente para construir propuestas de valor reales dentro del portfolio de innovación de SDG Group. Específicamente, los modelos predictivos aquí validados se han estructurado como potenciales soluciones comerciales para dos sectores estratégicos:

- **Mantenimiento Predictivo en Aeronáutica:** Aplicando la arquitectura a la secuencia de eventos de vuelo en flotas de aviones, se ha diseñado una propuesta para predecir el tiempo y número de vuelos restantes hasta el desgaste crítico de componentes (como las ruedas de las aeronaves).

- **Eficiencia Operativa en Telecomunicaciones:** En este sector, se ha valorado la adaptación del modelo para un cliente dedicado a la instalación de líneas de internet, desde la solicitud hasta el alta física. El procesamiento en tiempo real de los prefijos permite anticiparse a cuellos de botella en las contrataciones de instalación y reducir la tasa de cancelación de clientes.

4.2. Limitaciones

La evaluación del sistema de detección de anomalías ha puesto de manifiesto las dificultades inherentes a los enfoques no supervisados en este dominio. Al carecer de etiquetas de anomalía reales, la validación se ha realizado exclusivamente sobre casos sintéticos inyectados de forma controlada. Bajo el umbral cuantílico estricto ($Q = 0.998$), cuatro de las quince anomalías diseñadas no lograron ser detectadas en ninguna configuración del espacio latente, revelando que las desviaciones sutiles, como alteraciones de atributos financieros que no rompen la estructura secuencial, requieren umbrales menos conservadores o la incorporación de variables de negocio adicionales. Además, el fenómeno de propagación de errores dificulta la atribución causal precisa, ya que una anomalía severa en una dimensión puede distorsionar la representación latente completa y disparar alertas en ramas del decodificador que no corresponden a la fuente real del problema. Estas limitaciones no invalidan la herramienta como sistema de alerta, pero evidencian la necesidad de interpretarla como un mecanismo de priorización que debe complementarse con análisis experto.

Desde una perspectiva metodológica, tres limitaciones adicionales han condicionado el desarrollo del trabajo. En primer lugar, la ausencia de un estudio de *benchmarking* formal y riguroso frente a la literatura impide establecer con precisión el empleo de arquitecturas de aprendizaje profundo frente a algoritmos clásicos en todas las tareas realizadas.

En segundo lugar, la optimización de hiperparámetros ha constituido un cuello de botella operativo significativo. La exploración del espacio de configuraciones (dimensión del espacio latente, número de capas, tasas de *dropout*, pesos de la función de pérdida multi-tarea) se ha llevado a cabo de forma manual y guiada por experimentación iterativa, dado que la ejecución sobre CPU del entrenamiento completo de ambas arquitecturas requiere tiempos del orden de horas por configuración. Esta restricción computacional ha impedido la aplicación de técnicas sistemáticas de búsqueda como *Grid Search* sobre la totalidad del espacio de hiperparámetros. En consecuencia, las configuraciones seleccionadas deben interpretarse como soluciones razonables, aunque no necesariamente óptimas.

Por último, debe señalarse como una limitación el empleo de un único conjunto de datos (*BPI Challenge 2017*). Si bien este registro real posee una escala, riqueza de atributos y complejidad estructural idóneas para validar las metodologías propuestas, la falta de experimentación sobre otros procesos con lógicas operativas distintas o pertenecientes a sectores industriales diferentes restringe la generalización inmediata de las conclusiones obtenidas, requiriendo validaciones adicionales en trabajos futuros.

4.3. Extensiones futuras

Una primera línea de trabajo futuro consiste en la realización de una comparación rigurosa y en igualdad de condiciones con los modelos de referencia del estado del arte. Esto requiere reproducir las condiciones experimentales exactas de los trabajos de referencia (incluyendo el mismo conjunto de variables, la misma estrategia de partición y las mismas métricas de evaluación) para obtener una comparación numéricamente válida. En particular, contrastar el rendimiento del Transformer multitarea con modelos como los propuestos por Teinmaa et al. (2019), quienes evalúan de forma sistemática métodos de *Predictive Process Monitoring* sobre múltiples registros de eventos, o con las arquitecturas LSTM propuestas por Tax et al. (2017), permitiría situar con precisión la contribución de las arquitecturas desarrolladas en este trabajo dentro del panorama actual de la investigación. Del mismo modo, sería de interés la comparación del autoencoder GATv2 con métodos de detección de anomalías específicos para registros de eventos.

Una segunda extensión de especial interés se refiere a la explotación de las Graph Attention Networks para el modelado de procesos con actividades concurrentes. En el caso de estudio desarrollado, el proceso de concesión de préstamos presenta una estructura fundamentalmente secuencial, por lo que cada traza se ha modelado como un grafo dirigido lineal. Sin embargo, numerosos procesos empresariales reales incorporan relaciones de concurrencia, donde múltiples actividades pueden ejecutarse simultáneamente sin un orden predeterminado entre ellas. En estos escenarios, la topología del grafo deja de ser una cadena y se convierte en una estructura con bifurcaciones y convergencias, donde un nodo puede tener múltiples predecesores y sucesores simultáneos. Las GATs están diseñadas precisamente para este tipo de estructura: el mecanismo de atención dinámica permite a cada nodo ponderar de forma diferenciada la información procedente de cada uno de sus vecinos, independientemente de su número. Esta propiedad las convierte en una arquitectura idónea para procesos con paralelismo masivo, donde la relevancia de cada actividad concurrente puede variar en función del contexto.

Como tercera línea, la arquitectura multitarea adoptada asume que un único espacio latente compartido es óptimo para las cuatro cabezas de predicción simultáneamente. Esta hipótesis, si bien ofrece ventajas computacionales y de transferencia inductiva, no ha sido contrastada empíricamente frente a alternativas. Sería valioso comparar de forma sistemática el rendimiento de la arquitectura multitarea actual con variantes de cabeza única (un Transformer especializado por tarea) y con configuraciones intermedias (dos o tres Transformers agrupando tareas afines), evaluando el compromiso entre coste computacional, calidad predictiva por tarea y capacidad de generalización.

En cuarto lugar, se plantea la necesidad de desarrollar procedimientos de calibración del umbral de anomalía y de incorporar mecanismos de explicabilidad en ambas arquitecturas. En relación con la detección de anomalías, el umbral cuantílico ($Q = 0,998$) se ha fijado de forma empírica sin un procedimiento formal de calibración. En cuanto a la explicabilidad, para el autoencoder GATv2, los coeficientes de atención aprendidos por cada cabeza podrían visualizarse para identificar qué eventos vecinos contribuyen en mayor medida a la representación latente de cada nodo, proporcionando indicios sobre las relaciones que el modelo considera relevantes para determinar la normalidad de un caso. Para el Transformer predictivo, se ha obviado en gran medida esta faceta, dado que la herramienta se concibe como un sistema de recomen-

dación para la entidad financiera. La integración de métodos como SHAP o la investigación sobre técnicas específicas en Transformers permitiría identificar qué eventos y atributos del historial de la solicitud fundamentan cada predicción, dotando al sistema de la transparencia necesaria para su uso en contextos de decisión regulados.

Adicionalmente, la incorporación de análisis prescriptivo mediante técnicas de simulación permitiría evaluar cuantitativamente las consecuencias de distintas decisiones operativas antes de su ejecución, evolucionando el sistema desde una herramienta de soporte a la decisión hacia un mecanismo activo de recomendación; cerrando así el ciclo completo de analítica descriptiva, predictiva y prescriptiva.

4.4. Consideraciones finales

En conjunto, los objetivos planteados al inicio del trabajo se han alcanzado satisfactoriamente. En primer lugar, se ha establecido un marco teórico que integra el Process Mining clásico con técnicas de aprendizaje profundo. En segundo lugar, se ha desarrollado y evaluado un sistema de detección de anomalías basado en GATv2. Finalmente, se ha implementado un modelo predictivo multitarea basado en Transformers, validando su aplicabilidad al caso de estudio seleccionado.

Es necesario enfatizar que el presente trabajo constituye, ante todo, una prueba de concepto desarrollada con fines de aprendizaje y exploración tecnológica. Su propósito primordial no ha sido la obtención de un sistema con garantías de rendimiento para su despliegue en producción, sino la comprensión práctica del funcionamiento interno de las arquitecturas de aprendizaje profundo aplicadas al análisis de procesos, la evaluación de su viabilidad sobre datos reales y la acumulación de experiencia en el diseño de soluciones extremo a extremo para registros de eventos.

La integración de Process Mining y aprendizaje profundo permite superar algunas de las limitaciones tradicionales del análisis basado exclusivamente en el flujo de control. Aunque todavía existen desafíos relacionados con la interpretabilidad, la disponibilidad de datos y la generalización entre dominios, los resultados obtenidos muestran que estas técnicas constituyen una línea prometedora para el análisis predictivo de procesos empresariales.

Bibliografía

- Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning Long-Term Dependencies with Gradient Descent Is Difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166. DOI: 10.1109/72.279181.
- Berti, A., van Zelst, S., and Schuster, D. (2023). Pm4py: A process mining library for python. *Software Impacts*, 17:100556. DOI: 10.1016/j.simpa.2023.100556.
- Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer, New York.
- Blevi, L., Delporte, L., and Robbrecht, J. (2017). Process mining on the loan application process of a dutch financial institute: Bpi challenge 2017. Winner of the Professional Track, BPI Challenge 2017.
- BPI Challenge 2017 Organizers (2017). Bpi challenge 2017. Accessed: 2026-06-06. Available at: <https://ais.win.tue.nl/bpi/2017/challenge.html>.
- Brody, S., Alon, U., and Yahav, E. (2022). How attentive are graph attention networks? In *International Conference on Learning Representations*. arXiv:2105.14491.
- BupaR Team (2024). Data quality assessment. Accessed: 2026-06-07. Available at: https://bupaverse.github.io/docs/data_quality.html.
- Carmona, J., van Dongen, B., Solti, A., and Weidlich, M. (2018). *Conformance Checking: Relating Processes and Models*. Springer.
- Caruana, R. (1997). Multitask Learning. *Machine Learning*, 28(1):41–75. DOI: 10.1023/A:1007379606734.
- Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314.
- Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bi-directional Transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, pages 4171–4186. Association for Computational Linguistics. DOI: 10.18653/v1/N19-1423.
- Fey, M. and Lenssen, J. E. (2019). Fast graph representation learning with pytorch geometric. *ICLR Workshop on Representation Learning on Graphs and Manifolds*. arXiv:1903.02428.

- Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press, Cambridge.
- Hendrycks, D. and Gimpel, K. (2016). Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*. arXiv:1606.08415.
- Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*.
- Huo, S., Völzer, H., Reddy, P., Agarwal, P., Isahagian, V., and Muthusamy, V. (2021). Graph autoencoders for business process anomaly detection. In *International Conference on Business Process Management (BPM)*, pages 284–301. Springer.
- Jeong, D., Lim, J., and Bae, Y. (2017). Bpic 2017: Business process mining – a loan process application.
- Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*. arXiv:1609.02907.
- Leemans, S. J. J., Fahland, D., and van der Aalst, W. M. P. (2013a). Discovering block-structured process models from event logs - a constructive approach. In *Application and Theory of Petri Nets and Concurrency*.
- Leemans, S. J. J., Fahland, D., and van der Aalst, W. M. P. (2013b). Discovering block-structured process models from event logs containing infrequent behaviour. In *Business Process Management Workshops*.
- Liu, F. T., Ting, K. M., and Zhou, Z.-H. (2008). Isolation forest. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM)*, pages 413–422. IEEE. DOI: 10.1109/ICDM.2008.17.
- Lothaire, M. (1997). *Combinatorics on Words*. Cambridge Mathematical Library. Cambridge University Press.
- Mendling, J. (2008). *Metrics for Process Models: Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness*. Springer.
- Muñoz-Gama, J. and Carmona, J. (2010). A fresh look at precision in process conformance. In *International Conference on Business Process Management*, pages 211–226. Springer.
- Murphy, K. P. (2022). *Probabilistic Machine Learning: An Introduction*. MIT Press, Cambridge.
- Nolle, T., Luetzgen, S., Seeliger, A., and Mühlhäuser, M. (2018). Analyzing business process anomalies using autoencoders. *Machine Learning*, 107(11):1875–1893.
- Nolle, T., Seeliger, A., and Mühlhäuser, M. (2022). BINet: Multivariate business process anomaly detection using deep learning. *ACM Transactions on Knowledge Discovery from Data*, 16(1):1–37. DOI: 10.1145/3461474.
- PyTorch Geometric Team (2026). Pytorch geometric documentation: Messagepassing, gcnconv and gatconv. Official documentation, consulted in April 2026. Available at: <https://pytorch-geometric.readthedocs.io>.

- Rozinat, A. and van der Aalst, W. M. P. (2008). Conformance checking of processes based on monitoring real behavior. *Information Systems*, 33(1):64–95.
- Ruder, S. (2017). An Overview of Multi-Task Learning in Deep Neural Networks. arXiv:1706.05098.
- Santurkar, S., Tsipras, D., Ilyas, A., and Madry, A. (2018). How does Batch Normalization help optimization? (No, it is not about Internal Covariate Shift). In *Advances in Neural Information Processing Systems*, volume 31.
- Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826. DOI: 10.1109/CVPR.2016.308.
- Tax, N., Verenich, I., La Rosa, M., and Dumas, M. (2017). Predictive business process monitoring with LSTM neural networks. *Proceedings of the International Conference on Advanced Information Systems Engineering*, pages 477–492. DOI: 10.1007/978-3-319-59536-8_30.
- Teinemaa, I., Dumas, M., La Rosa, M., and Maggi, F. M. (2019). Outcome-oriented predictive process monitoring: Review and benchmark. *ACM Transactions on Knowledge Discovery from Data*, 13(2):1–57. DOI: 10.1145/3301300.
- TFM_Repo_ProcessMining (2026). https://github.com/PedroTM7/TFM_Repo_ProcessMining.
- van der Aalst, W. M. P. (2012). Process mining. *Communications of the ACM*, 55(8):76–83.
- van der Aalst, W. M. P. (2016). *Process Mining: Data Science in Action*. Springer, 2 edition. DOI: 10.1007/978-3-662-49851-4.
- van Dongen, B. F. (2017). BPI Challenge 2017. 4TU Centre for Research Data. Dataset. DOI: 10.4121/uuid:5f3067df-f10b-45da-b98b-86ae4c7a310b.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008. arXiv:1706.03762.
- Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018). Graph attention networks. In *International Conference on Learning Representations*. arXiv:1710.10903.
- Weijters, A. J. M. M. and Ribeiro, J. T. S. (2011). Flexible heuristics miner (fhm). In *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining*.
- Weijters, A. J. M. M., van der Aalst, W. M. P., and de Medeiros, A. K. A. (2006). *Process mining with the HeuristicsMiner algorithm*. BETA publicatie : working papers. Technische Universiteit Eindhoven. ISBN: 978-90-386-0813-6.
- Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. (2021). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24. DOI: 10.1109/TNNLS.2020.2978386.