Facultad de Ciencias

Process Mining para la mejora del proceso educativo a nivel universitario
(Upgrading university level educational proccess with Process Mining)

Trabajo de Fin de Máster
para acceder al

MÁSTER EN CIENCIA DE DATOS

Autor: María González-Estéfani Bravo

Director\es: Diego García Sainz
Ruth Villalón Molina

Septiembre – 2023

Resumen

A medida que aumenta el uso de las tecnologías en la enseñanza, la organización de los cursos se va adaptando a estas, intentando facilitar el proceso de aprendizaje del alumnado mediante la realización de actividades *online* vía plataformas como *Moodle*. En este proyecto, se aplican técnicas de Minería de Procesos (*Process Mining*) sobre datos de cursos universitarios de la actividad de los estudiantes en *Moodle* para extraer modelos que permitan, analizando los procesos de aprendizaje de los estudiantes, entender mejor cómo realizan las actividades *online* propuestas. A su vez se estudiará el proceso general de aprendizaje de los estudiantes, y cómo este pueden afectar a su rendimiento.

Palabras clave: Minería de Procesos (*process mining*), educación, modelo de proceso, modelo BPMN, ciencia de datos, rendimiento educativo, enseñanza presencial, enseñanza a distancia.

Abstract

As the use of technologies in the education field increases, the way of organizing the courses evolves to fit them, trying to ease the students' learning process by carrying out online activities via platforms such as Moodle. In this project, Process Mining techniques are applied to data on university student activity in Moodle to extract models that allow, by analyzing students' learning processes, to better understand how they execute proposed online activities. At the same time, the general learning process of the students will also be studied, and how it can affect their performance.

Key words: *Process Mining, education, process model, BPMN model, data science, educational performance, offline teaching, online teaching.*

Índice

Índice de figuras **4**

Índice de cuadros **5**

1. Introducción **6**
1.1. Objetivos 7

2. *Process Mining* **9**
2.1. *Process Discovery* 9
2.2. *Conformance checking* 14

3. Aplicación de PM en la educación **15**
3.1. Gestión del proyecto 15
3.2. Implementación en código: *pm4py* 17
3.2.1. Carga de archivos 17
3.2.2. Obtención de actividades de inicio y final y variantes de procesos 17
3.2.3. Filtros sobre los datos 18
3.2.4. Obtención del modelo BPMN 19
3.2.5. Obtención de la red heurística 19
3.2.6. Obtención de la red Petri 20
3.2.7. Visualización y guardado de modelos y gráficas 20
3.2.8. *Conformance checking*: evaluación de los modelos 20

4. Datos utilizados **22**
4.1. Procesado de datos 23
4.1.1. *Scripts* utilizados 24

5. Resultados y análisis **27**
5.1. Modelo general inicial 27
5.2. Modificación de parámetros de las funciones 28
5.2.1. *Red heurística* 29
5.3. Modelos generales 33
5.3.1. Modelo general del curso A1 15-16 de la universidad presencial . 33
5.3.2. Curso A2 15-16 de la universidad presencial 34
5.3.3. Curso 16-17 de la universidad presencial 34
5.3.4. Modelo de la universidad a distancia 36
5.4. Modelos según las calificaciones obtenidas 37
5.4.1. Modelo por calificaciones de la universidad presencial 37
5.4.2. Modelo por calificaciones de la universidad a distancia 39

6. Conclusiones **41**

Referencias **44**

Índice de figuras

1. Varios ejemplos de diferentes representaciones de los procesos que se pueden obtener con PM. 13
2. Diagrama de Gantt de este proyecto. 16
3. Figura del modelo general del curso A1 2015-2016 eliminando algunos eventos menores. 27
4. Gráfica que permite comparar los 4 valores de evaluación de un modelo para diferentes valores del *noise_threshold*. *Fitness* en azul, precisión en rojo, generalización en verde y simplicidad en amarillo. 28
5. Comparación de las medidas de evaluación de las redes Petri variando los parámetros *dependency_threshold* y *loop_two_threshold*. 31
6. Red heurística del modelo general del curso A1 2015-2016 con los parámetros óptimos. 32
7. Recorte del modelo 18 en el que se pueden distinguir tanto el nodo *AND* como la separación entre los cuestionarios. 33
8. 34
9. Recorte de las 2 últimas partes del modelo 20. 35
10. Recorte del subproceso esperado en los cuestionarios del modelo 21. . . 36
11. Recorte de la parte final del modelo 21 con los varios eventos de visualización y revisión. 37
12. Red heurística de los modelos obtenidos según los aprobados de cada estudiante. Arriba los aprobados con buena nota, a la izquierda los aprobados con nota justa y a la derecha los suspensos. 38
13. Recorte de la parte final del modelo 22 en el que se puede ver los varios eventos agrupados bajo el nodo *AND*. 39
14. Modelos de los estudiantes no presentados en los cuestionarios. 39
15. Extracto de una parte del modelo de los aprobados 24 en el que se puede ver los varios eventos agrupados bajo varios nodo *AND*. 40
16. 40
17. Figura del modelo general del curso A1 de la universidad presencial 2015-2016. 46
18. Figura del modelo general del curso A1 2015-2016 eliminando algunos eventos menores y con umbral de ruido *noise_threshold*= 0,2. 47
19. Figura del modelo general de la asignatura A2 del curso 15-16 de la universidad presencial. 47
20. Figura del modelo general sin grupos del curso 16-17 de la universidad presencial. 47
21. Figura del modelo general del curso de la universidad a distancia. . . . 48
22. Figuras de los modelos obtenidos según los aprobados de cada estudiante en la universidad presencial. Arriba los que han aprobado ambos, abajo los que han suspendido ambos y en medio los que han aprobado uno. . 48

23. Figuras de los modelos obtenidos según la nota media de cada estudiante en la universidad presencial. Arriba el modelo de los estudiantes con una calificación media de más de 1,5, abajo el modelo de los estudiantes con una calificación media aprobada de menos de 1,5. 49
24. Figuras de los modelos obtenidos según los resultados del alumnado de la universidad a distancia. Arriba el modelo de los aprobados y abajo el modelo de los suspensos. 49

Índice de cuadros

1. Tabla que ilustra el formato necesario para aplicar PM. 23

1. Introducción

En los últimos años, y con los grandes avances que se están dando en ciencia y tecnología, el uso de las tecnologías en la vida diaria se ha expandido a la mayor parte de sectores de trabajo, incluida la enseñanza. En este campo, actualmente es común apoyarse en recursos informáticos tales como plataformas interactivas o herramientas de comunicación, que ayudan a mejorar las condiciones de estudio y facilitan el trabajo tanto de docentes como del alumnado [8].

Una de las herramientas más utilizadas es la plataforma de *Moodle* [1], que proporciona la posibilidad de acceso directo e ininterrumpido al material del curso y una forma sencilla y flexible de hacer entregas y evaluaciones, entre otras cosas. El uso de esta herramienta está pensado para mejorar y facilitar el proceso de aprendizaje del alumnado, buscando ayudarles a aprender y entender mejor los contenidos de cada curso y asignatura; pero la gran flexibilidad que ofrece la plataforma a la hora de realizar y completar las actividades del curso puede hacer que organizar una asignatura con este objetivo sea una tarea complicada para los docentes.

Por ello, a los docentes les interesa conocer la forma en la que los estudiantes interactúan con la plataforma y el proceso seguido para sacar el curso adelante. Es sencillo pensar que basta mostrar en la plataforma los contenidos y tareas del curso de forma secuencial para que los estudiantes sigan el orden esperado para realizarlas, pero debido en parte a la flexibilidad de los horarios, al acceso ilimitado a los materiales y a la naturaleza impredecible de los humanos, el verdadero proceso seguido por los estudiantes no suele ser el esperado.

Para poder observar y analizar estos procesos desconocidos y poder mejorar los resultados obtenidos con ellos, surgió la disciplina de *Process Mining* [13], que permite extraer el proceso subyacente en el conjunto de actividades y pequeñas acciones realizadas a diario almacenadas en un registro de datos y eventos. Esta disciplina empezó desarrollándose en base a la ciencia de datos y otras técnicas similares ya existentes, con el objetivo de optimizar los procesos realizados en las empresas y negocios.

Puesto que los procesos de enseñanza en un curso académico se asemejan en cierta medida a los de los negocios en cuanto a procesos ocultos y potencial de optimización, esta disciplina se ha llevado para aplicar también en la enseñanza, permitiendo extraer y analizar el proceso de aprendizaje de los estudiantes de un curso a partir de la actividad almacenada por plataformas como *Moodle*. De esta forma, es posible conocer hasta cierto punto la forma que tienen los estudiantes de trabajar para poder organizar el curso de la mejor forma posible.

Por ello, en este trabajo se busca aplicar técnicas de *Process Mining* para poder extraer estos procesos subyacentes de los datos recabados y, así, otorgar a los docentes un apoyo para poder mejorar y optimizar la organización de sus cursos, en vistas de ayudar a los estudiantes a mejorar también.

1.1. Objetivos

La finalidad de extraer el proceso de estudio del alumnado es, como ya se ha mencionado anteriormente, poder encontrar el método más beneficioso para los estudiantes e intentar hacer que aprendan lo máximo posible, por lo que es importante para los docentes optimizar la programación del curso y apoyarse en el buen uso de las herramientas tecnológicas como recurso, pues hay veces que lo que se cree más beneficioso resulta dar malos resultados.

Por ejemplo, tener la libertad para realizar y entregar los ejercicios y cuestionarios en una amplia ventana de tiempo puede desembocar en una dejadez por parte de algunos estudiantes, quienes en vez de hacer las actividades poco a poco, lo dejan todo para el final y procuran completarlo todo en un tiempo insuficiente, incluso abandonando la idea de intentarlo todo y decidiendo sólo realizar lo mínimo para salir adelante, lo que generalmente lleva a malos resultados. Si el número de alumnos y alumnas que toman esta actitud es demasiado alto en el modelo generado, entonces dejar esa libertad de tiempo de entrega no sería una acción recomendable para el profesorado.

Estas cuestiones es el tipo de cosas que se busca identificar y corregir o mantener en los procesos de aprendizaje, y con este fin, se han establecido algunos objetivos tanto a nivel general como más concretos de cara a intentar resolver estos problemas.

Primero de todo, el objetivo principal de este trabajo va a ser **extraer el proceso de aprendizaje y estudio de los alumnos y alumnas de un curso académico universitario a partir de los modelos generados con *Process Mining***. Una vez obtenido este proceso y tras comprobar que es sólido, se pasará a abarcar los siguientes objetivos más concretos:

1. Comprobar si los estudiantes siguen el proceso de aprendizaje indicado por los docentes, comparando el proceso real obtenido con PM con el proceso esperado.

2. Buscar si existe una correlación entre el modelo obtenido y el rendimiento del alumnado.

3. En relación con el objetivo anterior, encontrar las diferencias entre los estudiantes con buen rendimiento y los estudiantes con mal rendimiento.

4. Detectar los puntos en común y las diferencias, si existen, entre los procesos de aprendizaje del alumnado de una universidad presencial y una universidad a distancia.

Estos objetivos nos permitirá dar con los pequeños detalles mencionados anteriormente, permitiendo discernir qué acciones y métodos es positivo mantener y cuáles es mejor eliminar.

Para poder llevar todo esto a cabo, se cuenta con datos de un mismo curso en una universidad presencial y en una a distancia que han realizado la misma actividad, lo que nos da la posibilidad de poder comparar los procesos de ambos lugares para de comprobar si existen diferencias entre los procesos de aprendizaje de los dos grupos de estudiantes o si el modo de enseñanza (presencial o virtual) es irrelevante en el proceso.

2. *Process Mining*

Los conjuntos de datos y tablas generados por las plataformas de aprendizaje utilizadas en la enseñanza pueden ser utilizados para obtener un modelo sobre el proceso de aprendizaje de los estudiantes mediante *Process Mining*.

Process Mining (abreviado como PM de ahora en adelante) es un conjunto de técnicas relacionadas con la ciencia de datos y la gestión de procesos que permite, extrayendo información de registros de eventos, obtener, visualizar y mejorar procesos no triviales ni asumidos.

Esta disciplina surgió inicialmente alrededor del año 2000 de la mano de Wil van der Aalst [12], y a partir de ese año empezó a evolucionar, desarrollándose diferentes algoritmos tales como *Alpha miner* (el primero de todos, presentado por van der Aalst en 2004 en el artículo [11]) o *Heuristic miner*, también presentado por van der Aalst posteriormente en [14]. Cada nuevo algoritmo cubría y corregía los mayores problemas que estaban presentes en los anteriores, como la tolerancia al ruido o las limitaciones sobre las relaciones entre los eventos.

Dentro del concepto de PM, se pueden considerar diferentes categorías según el tipo de técnicas a utilizar y finalidad, que según queda establecido en el Manifiesto sobre Minería de Procesos [13] son las siguientes:

- ***Process Discovery*** o descubrimiento: permite descubrir un modelo a partir de los datos de un registro. Este es el tipo en el que nos centraremos posteriormente para obtener la información que nos interesa.
- ***Conformance Checking*** o verificación de conformidad: permite validar los modelos generados, comparando los datos de los registros de datos existentes con los modelos y determinando si el proceso generado en los modelos se ajusta o no al real.
- ***Enhancement*** o mejora del modelo: adapta, extiende y mejora los modelos originales a partir de la información extraída de los registros de datos, generando un nuevo modelo mejorado.

En vista de los objetivos de este trabajo y como ya se ha mencionado, se va a tratar principalmente la primera categoría, *Process Discovery*, utilizando de apoyo la segunda, *Conformance Checking*, para escoger los mejores modelos generados. Por ello, pasamos a expandir los *inputs* y salidas requeridos en la categoría de *Process Discovery*.

2.1. *Process Discovery*

Para empezar, se requiere un registro de eventos en el que se hayan guardado las diferentes acciones tomadas por los usuarios, Los registros de eventos deben tener al menos 3 columnas distinguibles que contengan la siguiente información:

1. **Eventos o actividades**: lista de los diferentes eventos y acciones realizadas en el proceso a extraer.

2. **Usuarios o casos**: ID que permita identificar diferentes ejecuciones del mismo proceso.

3. **Tiempos**: marcas de tiempo que permitan identificar el orden en el que ocurren los eventos.

Los registros también pueden contener otras columnas con información complementaria sobre los eventos o los casos, con el objetivo de poder filtrar el registro según interese y poder generar diferentes modelos sobre ciertos conjuntos de datos condicionados.

Por otro lado, los modelos de proceso generados se pueden visualizar de diferentes formas, según qué estructura nos interese más. Para este trabajo se destacan 5 representaciones principales:

- **Modelos BPMN**: basado en la notación gráfica del mismo nombre (siglas de *Business Process Modeling Notation*, ver [2]), estos modelos se introdujeron precisamente para visualizar sencillamente los procesos generados en los negocios.

  En estos modelos, los diferentes eventos del proceso están representados cronológicamente de forma similar a un grafo dirigido, unidos mediante nodos con símbolos que indican diferentes acciones en el proceso.

  Una X en el nodo indica una elección, es decir, que en el proceso sólo se puede escoger uno de los caminos y acciones que salen de esta (equivalente al operador *XOR*). Por otro lado, un nodo + indica que los eventos sucesivos tras este nodo pueden ocurrir al mismo tiempo, sin un orden determinado (es decir, un operador *AND*). Ambos nodos aparecen por parejas en el modelo, indicando el comienzo y el final de la elección o conjunto de acciones paralelas.

  En la figura 1a se puede ver un ejemplo de este tipo de modelos.

- **Árboles inductivos**: la representación en árbol de la que se extrae el modelo BPMN, gegenerado con un algoritmo inductivo.

  En estos árboles, que se recorren en inorden, además de los dos nodos presentes en los modelos BPMN *XOR* y *AND* se les añade dos tipos más, representados por los símbolos $\rightarrow$, que indica secuencia, y $*$, que indica un bucle.

  La figura 1b muestra un ejemplo de árbol del mismo proceso que el ilustrado con el modelo BPMN.

- **Grafos dirigidos (DFG)**: como su nombre indica, los procesos también se pueden representar directamente con grafos dirigidos, en los cuales los nodos se corresponden con los eventos. En este caso, los pesos de las aristas representan el número de veces que ocurre una relación entre eventos en el registro.

Un ejemplo de este tipo de representación se puede ver en la figura 1c, siguiendo el mismo proceso que los dos anteriores.

- **Redes heurísticas**: modelos obtenidos mediante el algoritmo *Heuristic Miner*, que trabajan sobre el DFG revisando los posibles eventos concurrentes y las dependencias entre ellos.

  Su representación es similar a la de los grafos dirigidos, siendo su mayor diferencia que, en estas redes, los eventos que se detecten que suceden al tiempo (el equivalente a los nodos *AND* en los modelos BPMN) no están relacionados entre sí, lo que hace que la representación de las redes heurísticas sea algo más simple que los DFG.

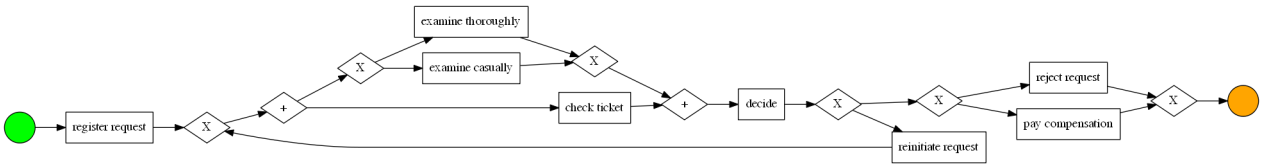
  En la figura 1d está representado un ejemplo de red heurística en base al proceso común mostrado en los anteriores .

- **Redes Petri**: un tipo de red ya existente de forma previa a la introducción de PM, inicialmente utilizadas para modelar reacciones químicas. Estas redes permiten no sólo representar el proceso completo, sino que también permiten simularlo de forma gráfica, añadiendo y moviendo puntos (denominados formalmente como *tokens*) sobre este.

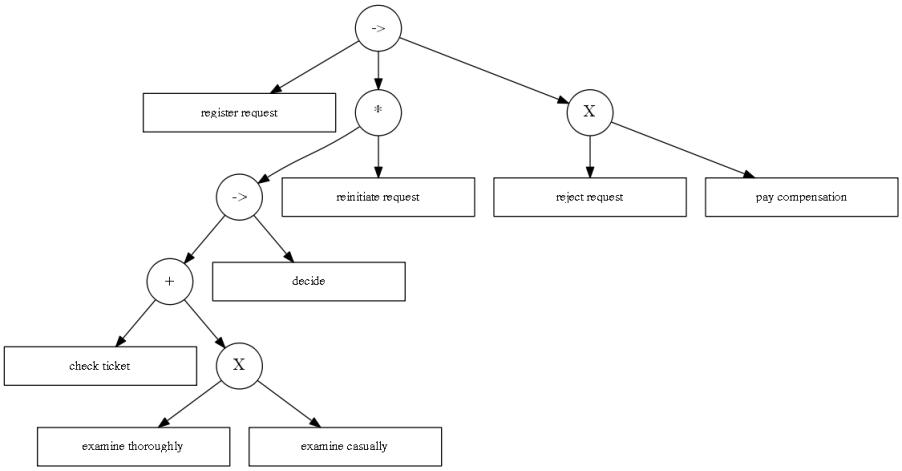
  En la representación gráfica, se tiene que los nodos rectangulares indican transiciones y en el caso de PM también representan los eventos del proceso, mientras que los círculos, que es donde se colocan los *tokens*, indican lugares. Las aristas con pesos indican tanto el número de *tokens* generados tras la transición como el número de *tokens* requeridos para realizar una transición, según la arista llegue a un lugar o a una transición respectivamente.

  Siguiendo estas definiciones, para poder realizar una acción en el proceso se requiere que los lugares previos a la transición contengan al menos el número de *tokens* requeridos en las aristas. Debido a esta posibilidad de simular el proceso, las redes Petri son las que se utilizan por ejemplo para evaluar los procesos generados.

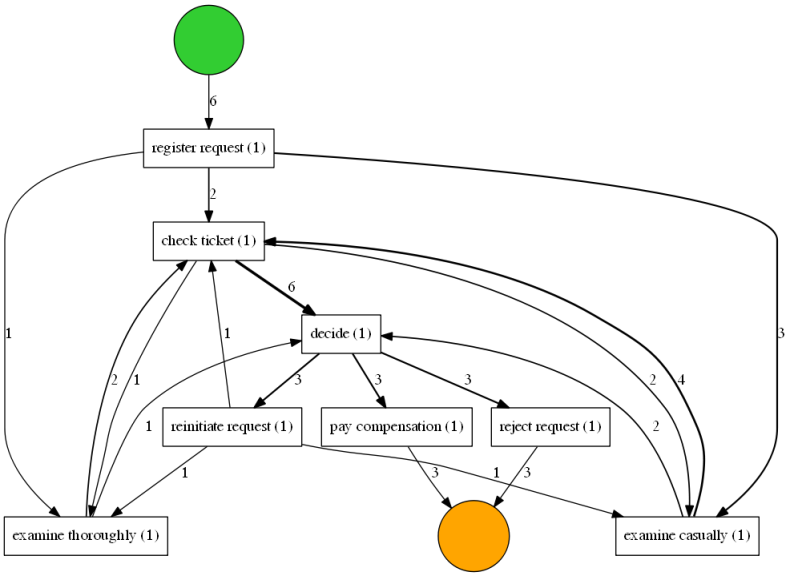
  La figura 1e ilustra un ejemplo de este tipo de red.



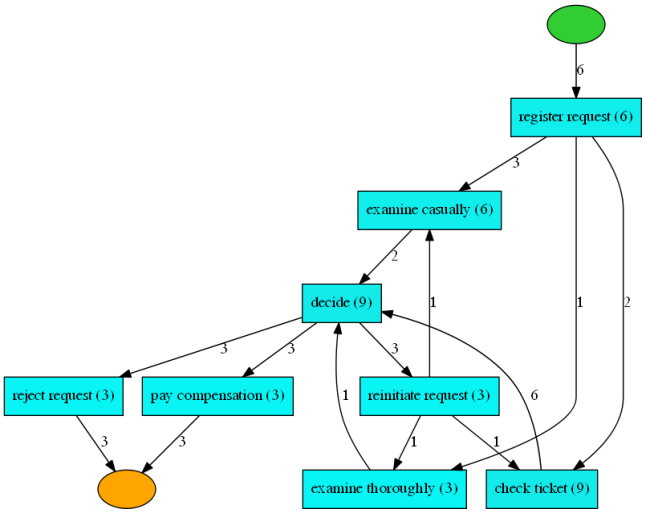
(a) Ejemplo de un modelo BPMN obtenido de la documentación de [6].



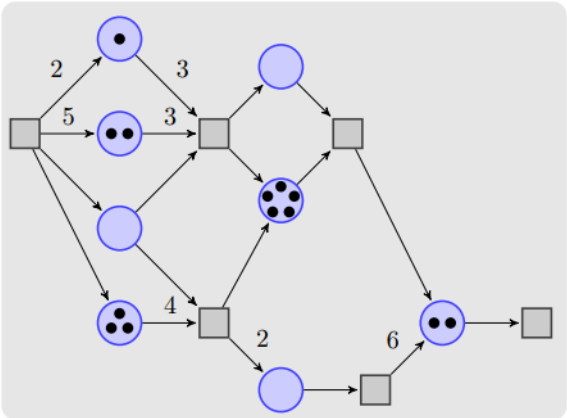
(b) Ejemplo de un árbol de proceso inductivo obtenido de la documentación de [6].



(c) Ejemplo de un grafo dirigido obtenido de la documentación de [6].



(d) Ejemplo de una red heurística obtenido de la documentación de [6].



(e) Ejemplo de una red Petri obtenido de [7].

Figura 1: Varios ejemplos de diferentes representaciones de los procesos que se pueden obtener con PM.

2.2. *Conformance checking*

Tras haber extraído los procesos requeridos, interesa saber si el modelo o red obtenida se ajusta correctamente al conjunto de datos que teníamos originalmente. Para ello, se puede recurrir a la segunda categoría de PM, *Conformance checking*, para evaluar el modelo comparándolo contra el *dataset* original, pudiendo utilizar 4 medidas diferentes:

- ***Fitness***: calcula cuánto del comportamiento del conjunto de datos es admitida por el modelo.
- **Precisión**: compara los procesos de los registros de datos que compartan los primeros pasos (acciones iniciales) con el proceso del modelo.
- **Capacidad de generalización**: la aptitud que tiene el modelo de generalizar, analizando cómo coinciden el modelo y el conjunto de datos. Dicho de otro modo, un modelo es general (o tiene alta capacidad de generalización) cuando todos los posibles procesos que ofrece el modelo ocurren múltiples veces al extraer procesos de este.
- **Simplicidad**: cómo de sencilla es la red Petri del modelo, en base al número de arcos que tenga.

Las 4 medidas señaladas son independientes entre sí, por lo que escoger la medida que más interesa es cuestión del objetivo con el que se generen los modelos, aunque en general es recomendable revisarlas todas para poder obtener los mejores resultados.

Una vez introducidos los conceptos básicos tanto de *Process Discovery* como de *Conformance checking*, pasamos a ver su aplicación en el problema que nos interesa en específico, en el campo de la educación.

3. Aplicación de PM en la educación

En nuestro caso, en este proyecto nos interesa aplicar PM sobre datos en el campo de la enseñanza, extrayendo los datos de las interacciones de los estudiantes con la plataforma de *Moodle* para poder generar modelos sobre los procesos de aprendizaje de estos, y así poder aprender y optimizarlos para intentar mejorar el rendimiento del alumnado.

No es la primera vez que se utiliza PM en el campo de la educación, pues esta disciplina ya ha sido utilizada anteriormente en otros estudios tales como [3], en donde específicamente se centran en obtener modelos mediante el algoritmo *heuristic miner*, o [5], en donde los procesos educativos se extrapolan a partir de redes Petri. Otros estudios deciden tomar otra ruta y, en vez de estudiar el proceso de aprendizaje en general, se centran en detalles más específicos, como en [9], focalizado en estrategias de aprendizaje de textos; o directamente utilizan PM educativo para mejorar la propia disciplina, como se ve en [4] en donde se buscan otras formas de agrupar o dividir los subconjuntos de estudiantes.

La línea de este proyecto se queda en el nivel general de análisis del proceso educativo, similar al artículo [10], y no centrándose en ninguna técnica en especial, salvo en las diferencias según el rendimiento.

3.1. Gestión del proyecto

Como todo proyecto de de análisis de datos, es necesaria una buena planificación para poder cumplir los objetivos en el tiempo establecido. Por ello, para este proyecto se han seguido las etapas fundamentales de un proyecto de análisis de datos, partiendo de los propios objetivos y obtención de datos hasta las conclusiones finales y presentación de resultados. En la figura 2 se muestra el Diagrama de Gantt seguido para este proyecto.

El primer paso realizado ha sido investigar sobre el campo de *Data Science* que se va a utilizar, *Process Mining*, para conocer qué es, sus aplicaciones, posibilidades y objetivos. Esta investigación empezó como una introducción general, y a medida que se iba avanzando en el proyecto, se ha ido investigando sobre partes más concretas, útiles para la parte sobre la que se estaba trabajando mientras.

Dentro de la primera introducción y para irse familiarizando, también se ha ido aprendiendo sobre diferentes métodos de implementación de esta disciplina en código; aunque no es hasta después de haber procesado los datos que se investiga a fondo el paquete y las funciones necesarias para trabajar. A su vez, con la introducción de PM y al tiempo que se realizaba la obtención de datos, se han ido estableciendo los objetivos de este proyecto, empezando con algunos generales y posteriormente pasando a algunos más concretos. Al final del proyecto, teniendo ya todos los datos y modelos que interesaban y han sido posible conseguir, se revisan y reescriben correctamente y de forma más concreta estos objetivos de cara a extraer las conclusiones.

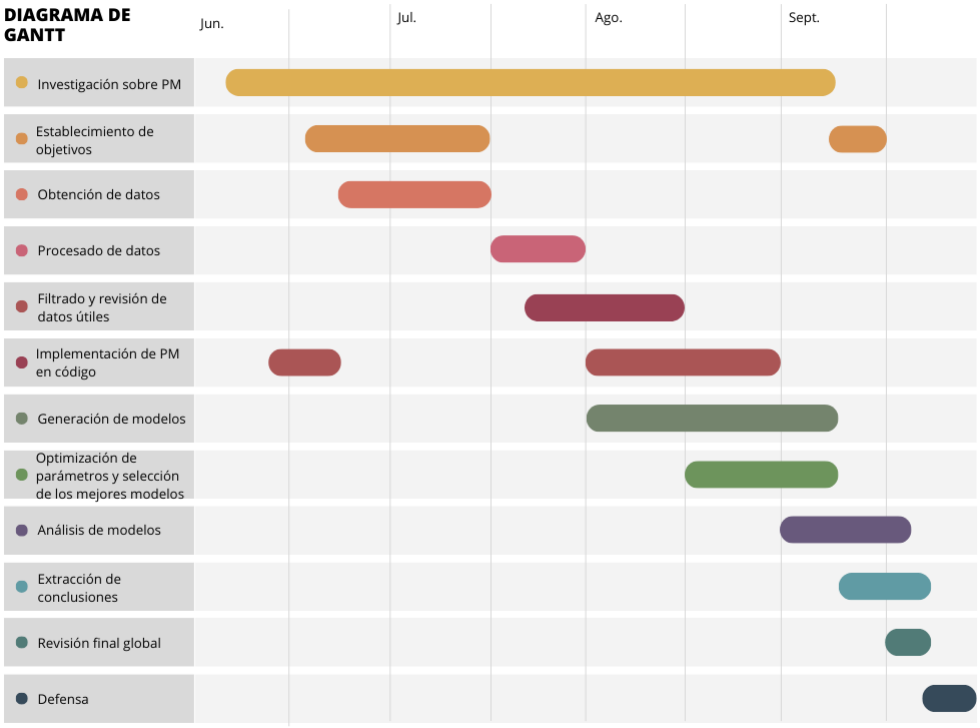


Figura 2: Diagrama de Gantt de este proyecto.

Pasando a la obtención y procesado de datos, lo primero de todo ha sido solicitar permiso a las partes e instituciones involucradas para el uso de los datos, junto a una comprobación de que los datos estén en perfecto estado y no corrompidos. Tras obtenerlos y tener los permisos imprescindibles, se pasa a su procesado, que en el caso de este proyecto, los *scripts* para ello venían incluidos con los propios datos, por lo que este tiempo se ha invertido en ejecutar estos *scripts*, entendiendo lo que se estaba haciendo, y revisando los nuevos datos generados. Una vez procesados los datos, se analizan por encima para descubrir aquellos que son útiles para el proyecto y aplicación de PM y descartar aquellos que se hayan generado de paso o tengan datos repetidos.

Tras haber procesado los datos, también se inicia el proceso de generación de modelos, que ayuda a hacerse una idea del material con el que se está trabajando. Sólo después de haber generado múltiples modelos diferentes y de haber seleccionado los mejores optimizado las funciones correspondientes es cuando se inicia el análisis de estos y la extracción de conclusiones, para finalmente revisarlo todo en conjunto y preparar su presentación al público.

De esta forma y habiendo ya introducido tanto los objetivos como PM en apartados anteriores, se continúa con el resto de actividades del proyecto, empezando con la implementación en código.

3.2. Implementación en código: *pm4py*

Aprovechando que los *scripts* que se van a utilizar para el procesado de datos están hechos con *Python*, se ha buscado una librería de *Process Mining* para *Python*. Así, la librería que se va a utilizar para obtener los correspondientes modelos es *pm4py* [6].

Esta librería permite calcular y generar modelos a partir de un registro de eventos, en el que se recogen los sujetos, acciones y tiempo de una serie de eventos de los que interesa extraer y conocer un proceso. Una vez obtenido el modelo o equivalente, se tienen diferentes opciones para representar los procesos gráficamente, desde árboles o grafos hasta modelos más específicos como las redes Petri.

A continuación, se introducen diferentes funciones que permiten obtener los modelos, gráficas y otros datos interesantes.

3.2.1. Carga de archivos

El tipo de archivos que utiliza este paquete para calcular los modelos son conjuntos de datos con 3 columnas identificadas (usuario/caso, actividad y tiempo) y posibles atributos extra. Por ello, aunque se pueda cargar el *dataset* con librerías externas (como *pandas*), se requiere formatear los nombres de las columnas para que coincidan con los que utilizan las funciones. Para ello, existe la función *format_dataframe*, que toma como *inputs* el *dataset* y los nombres de las 3 columnas requeridas, como se puede ver a continuación.

```
import pandas
import pm4py

event_log = pandas.read_csv('nombre_archivo.csv')
event_log = pm4py.format_dataframe(event_log, case_id='Idusuario',
    activity_key='NombreEvento', timestamp_key='Hora')
```

Así, basta con que el archivo en el que se hayan guardado los datos se pueda leer como un *dataframe* y tenga las 3 columnas indicadas definidas.

3.2.2. Obtención de actividades de inicio y final y variantes de procesos

Puesto que estamos tratando con procesos secuenciales con inicio y final, tener una forma de obtener los eventos de inicio y final es algo imprescindible, al igual que es interesante tener un medio para obtener las diferentes secuencias de eventos presentes en el registro. Por ello, este paquete contiene tres funciones para todo ello:

```
# Actividades iniciales
start_activities = pm4py.get_start_activities(event_log)

# Actividades finales
end_activities = pm4py.get_end_activities(event_log)

```

```
# Variantes
variants = pm4py.get_variants(event_log)
```

3.2.3. Filtros sobre los datos

Puesto que en la mayor parte de los casos se busca comparar modelos entre dos o más conjuntos disjuntos de usuarios u otras particiones, antes de obtener los modelos suele ser necesario filtrar los datos. Para ello, en este paquete existen diferentes funciones que permiten filtrar tanto sobre los atributos extra como sobre las propias actividades o tiempo, como por ejemplo:

- Obtener los casos con actividades iniciales.

```
event_log = pm4py.filtering.filter_start_activities(event_log)
```

- Obtener los casos que verifican las relaciones directas especificadas. Si por ejemplo se quieren filtrar las relaciones (a,b) y (a,c) y las relaciones presentes en el conjunto de datos son <a,b,c>,<a,c,b>,<a,d,b>, el registro filtrado contendrá los casos <a,b,c>,<a,c,b>.

```
event_log = pm4py.filter_directly_follows_relation(event_log, [('
    A','B'),('A','C')])
```

- Filtros sobre los valores de los atributos o datos complementarios.

```
event_log = pm4py.filter_event_attribute_values(event_log,
    attribute_key='NombreAtributo', values=valores)
```

- Filtros sobre los tiempos.

```
event_log = pm4py.filter_time_range(event_log, '2015-01-01
    00:00:00', '2016-01-01 00:00:00')
```

- Filtros sobre el numero de veces que han ocurrido las actividades.

```
filtered_dataframe = pm4py.filter_case_size(dataframe, 5, 10)
```

En la mayor parte de estas funciones, existe un parámetro que permite especificar si lo que se quiere es quedarse o excluir los datos especificados, variando su nombre de función a función.

Como el conjunto de datos se puede filtrar como un dataset normal con *pandas* y no se va a requerir filtrar sobre datos específicos de PM, no se utilizarán ninguna de estas funciones, incluyéndolas sólo como dato informativo.

3.2.4. Obtención del modelo BPMN

Una vez se tiene el conjunto de datos que interesa, se puede calcular directamente el modelo de PM para analizarlo posteriormente. Para ello, basta con utilizar las siguientes funciones:

```
process_tree = pm4py.discover_process_tree_inductive(event_log)
bpmn_model = pm4py.convert_to_bpmn(process_tree)
```

Se observa que no existe una función para obtener directamente el modelo BPMN, sino que hay que obtenerlo a partir del árbol inductivo.

Como toda función, se le puede pasar algunos parámetros para mejorar y controlar el tipo de modelo que se quiere obtener. En este caso (al descubrir el árbol), tenemos el siguiente parámetro aparte del registro de datos:

- *noise_threshold*: un número real entre 0 y 1 que sirve como filtro para eliminar procesos infrecuentes. Se inicializa a 0.

Para generar el modelo BPMN, se ha visto que es necesario primero generar el árbol correspondiente, por lo que la función para calcularlo ya ha sido introducida. Este tipo de modelos, junto a los grafos dirigidos, muestran la misma información que los modelos BPMN, por lo que no se van ni a utilizar ni generar en el análisis de los datos.

3.2.5. Obtención de la red heurística

Otra forma de representar los procesos de los datos es mediante una red heurística, que se puede generar con la siguiente función.

```
map1 = pm4py.discover_heuristics_net(event_log)
```

A diferencia de las anteriores funciones, esta función no tiene como parámetro *noise_threshold*, si no que se le pueden pasar los siguientes 3 parámetros:

- *dependency_threshold*: umbral que permite determinar cuándo 2 actividades o acciones son dependientes o están relacionadas entre sí.
- *and_threshold*: determina la frecuencia con la que una acción o actividad tiene que ocurrir para ser considerada relevante.
- *loop_two_threshold*: límite para determinar cuántas veces una actividad o acción puede ocurrir dentro de un bucle antes de ser considerada un bucle de tamaño 2.

Aunque para representar los modelos se van a utilizar los modelos BPMN, las redes heurísticas quedan como apoyo visual en caso de que las representaciones de los modelos salgan altamente complicados.

3.2.6. Obtención de la red Petri

Por último, la forma más característica de representar los procesos es mediante las redes Petri, obtenibles con la siguiente función:

```
net, im, fm = pm4py.discover_petri_net_inductive(event_log)
```

Al igual que en las funciones para obtener el árbol inductivo y el modelo BPMN, a esta función se la puede pasar como parámetro *noise_threshold*.

Puesto que muestran exactamente la misma información que se muestra con los modelos BPMN, en un principio no haría falta calcularlas; pero como se va a ver en los siguientes apartados, las redes Petri son necesarias para poder evaluar el proceso obtenido, por lo que se generan como recurso de apoyo.

3.2.7. Visualización y guardado de modelos y gráficas

Si se quiere visualizar los modelos y redes obtenidos tras generarlos, hay que utilizar funciones específicas que permiten mostrar el resultado apoyándose en el paquete *GraphViz*. Cada tipo de red o modelo requiere una función específica, por lo que si se quiere por ejemplo visualizar un modelo BPMN, la función sería la siguiente:

```
pm4py.view_bpmn(bpmn_model)
```

De esta forma, se pueden visualizar directamente todas las redes o modelos que se hayan obtenido con las funciones anteriores.

Por otro lado, si lo que interesa es guardar los modelos o redes generados como imágenes en vez de visualizarlos, la función que habría que utilizar sería la siguiente (en este caso ejemplificado para un modelo BPMN).

```
pm4py.save_vis_bpmn(bpmn_model, 'ruta/nombre_archivo.png')
```

3.2.8. *Conformance checking*: evaluación de los modelos

Una vez generados los modelos y redes necesarias, es importante saber cómo de útiles son y si se ajustan correctamente a los datos originales, para lo que se recurre a la segunda categoría de PM para evaluar los modelos.

Las funciones que se utilizan para calcular las 4 medidas de evaluación son las siguientes:

```
print(pm4py.fitness_token_based_replay(event_log, net, im, fm))
print(pm4py.precision_token_based_replay(event_log, net, im, fm))
print(generalization_evaluator.apply(event_log, net, im, fm))
print(simplicity_evaluator.apply(net))
```

Puesto que las 4 miden valores completamente disjuntos, se van a utilizar todas para evaluar los modelos y redes obtenidos, y principalmente para deducir los parámetros óptimos a utilizar en las funciones.

Aunque el paquete *pm4py* cuenta con muchas más funciones sobre PM y existen múltiples formas más de generar y operar con modelos y redes de procesos que no se han introducido (dado que no va a ser necesario utilizarlas), estas funciones son suficientes para generar y analizar los modelos que nos interesa.

4. Datos utilizados

El conjunto de datos utilizado originalmente proviene de dos asignaturas durante los cursos 2015-2016 y 2016-2017 de una universidad presencial y un curso del 2015-2016 de una universidad a distancia, quienes han dado permiso y cedido los datos para su uso y análisis. Las dos asignaturas de la universidad presencial son la misma asignatura ofertada en dos grados diferentes (pero similares), lo que hace esperar que ambas clases sean parecidas entre sí.

Estos datos contienen los registros de parte de la actividad del alumnado en *Moodle* durante los cursos indicados. Con el fin de tener la posibilidad de comparar los datos entre cursos y para evitar que estos sean altamente dispares (puesto que es común que cursos diferentes tengan programaciones diferentes), los cursos de los que se han tomado los datos han participado en una actividad conjunta, con el fin precisamente de obtener datos para analizarlos y sacar conclusiones sobre el proceso de aprendizaje del alumnado.

Esta actividad consistió en un plan formativo sobre la escritura y redacción de textos de opinión y contraste de posiciones, en el que los estudiantes del curso debían llevar a cabo las siguientes tareas que se listan a continuación:

1. Introducción e instrucciones: lectura de las instrucciones de la actividad y visualización de un vídeo introductorio sobre los contenidos de la actividad.

2. Cuestionario 1: cuestionario previo antes de empezar la actividad en sí, para comprobar el grado de conocimiento inicial del alumnado.

3. Lectura de textos: se adjuntaron dos textos de opiniones opuestas para que los estudiantes leyesen y extrajeran conclusiones.

4. Entrega inicial: a los alumnos y alumnas se les pidió redactar un texto respondiendo a ciertas preguntas sobre los textos leídos anteriormente, de forma autónoma y sin ayuda.

5. Revisión y conclusión: se plantean otras preguntas más para mejorar el texto redactado anteriormente.

6. Cuestionario 2 y entrega final: por último, al acabar la actividad, los estudiantes debían responder a un nuevo cuestionario evaluable y redactar un nuevo texto apoyándose en el material otorgado, incluyendo su conclusión sobre el problema expuesto en los textos.

Aunque la actividad estaba preparada para ser realizada de una sola vez en un mismo día, se mantuvo abierta durante todo el curso, pudiendo los estudiantes realizarla y posteriormente revisarla en cualquier momento del curso.

Uno de los cursos del 16-17 de la universidad presencial no realizó la actividad al final, por lo que aunque se cuente con sus datos, estos no se utilizan ni procesan. Respecto al resto, la primera asignatura de la universidad presencial contó con 199 y 215 estudiantes para cada año, la segunda asignatura de la misma universidad contó con 64 estudiantes (en el año en el que se realizó la actividad); y el curso de la universidad a distancia tuvo 94 estudiantes.

Además de los *logs* que produce *Moodle* a partir de la actividad del alumnado, entre los datos de los que se dispone se encuentran calificaciones, encuestas y otros datos sobre los estudiantes que han juntado los profesores de las asignaturas. Todos estos datos, al proceder de diferentes cursos y universidades, tienen formatos diferentes, por lo que es necesario procesarlos antes de trabajar con ellos.

4.1. Procesado de datos

Para unificar el formato de los datos y reducirlos a un par de archivos, se procesan las diferentes fuentes de datos mencionadas anteriormente. Para ello, se han utilizado diferentes *scripts* creados en *notebooks* con *Python* 2.7, en los que se van tomando los diferentes archivos de datos de las asignaturas y se van transformando, modificando y unificando para generar nuevos archivos con los datos listados de forma más sencilla, ordenada y útil.

Como en todo proyecto de ciencia de datos, el procesado de los datos se puede considerar el paso más importante del proyecto, pues los resultados obtenidos pueden cambiar completamente según cómo de bien se hayan formateado los datos, o simplemente según qué conjunto de datos se escoja para analizar. Entre las acciones de procesado a aplicar, es necesario empezar con una limpieza profunda de los datos crudos, pues la plataforma *Moodle* almacena una gran cantidad de datos y acciones en sus registros. Otras acciones a aplicar también incluyen unificar las diferentes fuentes en una sola, renombrar las columnas y datos de forma clara, o unificar el formato de los datos de salida.

Puesto que se busca aplicar *Process Mining* a estos datos para obtener modelos o redes que analizar y comparar, se necesita que los datos utilizados para este fin contengan las 3 columnas requeridas para aplicar PM. Así, como objetivo principal se busca generar registros en los que se almacenen en lista las acciones o actividades efectuadas por los estudiantes o participantes, el usuario que la ha realizado y las horas en las que se llevaron a cabo, como se ilustra en el cuadro 1.

Hora	Idusuario	NombreEvento	Atributo1
31/12/2015 00:00	10000	NombreEvento1	ValorAtributo1

Cuadro 1: Tabla que ilustra el formato necesario para aplicar PM.

Por otro lado, para poder realizar un análisis sobre las correlaciones entre procesos

y modelos de diferentes grupos de estudiantes, es necesario dividir al alumnado en subconjuntos, lo que implica la necesidad de contar con más datos y columnas que las mencionadas anteriormente, como notas, grupos o número de sesiones del alumnado. Así, otro de los objetivos de nuestro procesado de datos va a ser obtener uno o varios *datasets* que almacenen esta información de forma ordenada y unificada.

Puesto que se están tratando datos de diferentes cursos, para distinguirlos bien se dejan por separado los datos de cada curso, generando al menos un registro y un *dataset* completo para cada uno.

4.1.1. *Scripts* utilizados

Debido a la gran cantidad de *scripts* que se han utilizado para procesar los datos, en vez de ir explicando en profundidad lo que hace cada uno individualmente y mostrar el código utilizado, se agrupan según el curso al que pertenezcan los datos que se han procesado, aprovechando que cada *script* coge únicamente datos de uno solo a la vez.

1. **Asignatura 1 (A1) de la universidad presencial del curso 2015-2016**: esta asignatura es la que más se ha procesado, y por ello, es la que tiene tanto los datos más completos como mejor estandarizados. Por esta razón, es la que en más profundidad y con más detalle se va a explicar.

   Primero de todo, se ha empezado por generar un registro inicial con la lista de actividades, usuarios y horas correspondientes con el formato requerido para poder aplicar PM. Tomando los datos completos y tras eliminar los datos que no estén bien formateados, se cruzan con la lista oficial de estudiantes por IDs para generar un primer archivo con los datos en crudo de los estudiantes presentes en el curso.

   Juntando este archivo con los datos de los cuestionarios (intentos, respuestas enviadas, revisiones), se genera primero un *dataset* reducido con la lista de las acciones relativas sólo a los cuestionarios y, posteriormente, el registro inicial con la lista completa de acciones, usuarios y tiempos en el formato que se busca.

   A continuación, se genera un archivo con los datos por estudiante del curso. De todos los archivos que se han generado en el proceso anterior, el archivo con los datos de los estudiantes se utiliza para ayudar a organizar y completar estos datos complementarios.

   Partiendo de nuevos datos sobre los cuestionarios, se genera un nuevo archivo con una lista de acciones, usuarios y tiempos similar al que se ha generado anteriormente, pero con un cálculo más acertado para los segundos de las acciones que se han realizado en el mismo minuto. Con este archivo y siguiendo el mismo procedimiento que la primera vez, se genera otro registro con la lista completa de acciones y demás. A partir de este nuevo archivo y cruzando con la cronología, se calculan las duraciones de las actividades y si realmente se han podido realizar en ese tiempo.

En paralelo y cruzando con la lista de participantes y datos completos, se toman las notas de los estudiantes, generando un archivo con las notas correspondientes por estudiante.

A su vez, también se calculan datos extra como el número de sesiones, eventos por sesión o componente y eventos al mes para cada estudiante. Unificando todos estos archivos y datos, se genera el dataset definitivo con todos los datos por estudiante.

2. **Curso de la universidad a distancia (D1) del curso 2015-2016**: de forma similar al curso A1, se obtienen las notas de los estudiantes para generar un archivo con las calificaciones de cada uno, para posteriormente generar el registro con la lista completa de acciones, estudiantes y tiempos al cruzarlo con otro archivo con los datos completos del curso.

   Esta lista de actividades se procesa para obtener de nuevo datos interesantes del alumnado (tales como eventos por mes y sesión, número de sesiones o eventos por componente) y se unifican en un solo archivo que contiene los datos completos por estudiante.

3. **Asignatura 2 (A2) de la universidad presencial del curso 2015-2016**: para este curso, lo único que se llega a generar es el registro con las acciones, estudiantes y tiempos del curso, tanto la completa como la limitada a los cuestionarios.

4. **Asignatura 1 (A1) de la universidad presencial del curso 16-17**: de forma similar al curso D1, se genera el registro de eventos para aplicar PM cruzando los datos completos crudos con los datos de las calificaciones de los estudiantes, para posteriormente generar el *dataset* con los datos complementarios de los estudiantes.

Como apunte extra sobre el procedimiento del procesado de los datos comunes a todos los cursos, se puede destacar el hecho de que tanto los usuarios como los diferentes eventos realizados se han distinguido mediante los IDs que aparecen en las descripciones de cada acción presentes en los datos en crudo, en vez de obtenerse de los nombres completos y de los nombres originales respectivamente. Esto hace que, al ser cursos diferentes en *Moodle*, la plataforma genere IDs diferentes para cada uno, lo que hace complicado el unificar los nombres de estos datos correctamente. Es por esto por lo que los nombres normalizados de los eventos no coinciden entre los diferentes cursos.

Dado que para algunos cursos no se disponía de todos los datos que se quieren utilizar, no se han podido generar todos los archivos con los datos complementarios de los estudiantes, por lo que no se van a poder obtener modelos según los resultados del alumnado para todos los cursos. Además, aunque se hayan podido procesar y generar registros para todos los cursos, al haberse realizado mediante diferentes *scripts* y tener los datos diferentes procedencias, no todos los datos se han normalizado de la misma

forma, o incluso en algunos casos, no se han llegado a normalizar siquiera, lo que hace imposible unificarlos todos en un par de archivos y generar un modelo completo con todos los datos.

A pesar de todo, se tienen datos y registros suficientes como para proceder con el análisis y la generación de modelos con PM, por lo que procedemos a la aplicación de PM sobre los datos para obtener los procesos subyacentes en los datos.

5. Resultados y análisis

Una vez procesados los datos iniciales y hechas las introducciones a PM y a las diferentes funciones del paquete *pm4py* de *Python*, se procede a aplicar estas funciones para obtener diferentes modelos sobre los procesos de aprendizaje presentes en los datos. La mayor parte de los modelos se adjuntan en los anexos, debido al gran tamaño que tienen; mostrando directamente las partes más relevantes de los modelos.

5.1. Modelo general inicial

A partir de los datos del curso A1 del 15-16 de la universidad presencial, generamos un modelo inicial con los parámetros por defecto de las funciones (figura 17 en el anexo).

A simple vista, se observa que el modelo es altamente complejo y el proceso cuenta con una gran cantidad de nodos, lo que dificulta su visualización. Puesto que en estas condiciones realizar un análisis es técnicamente imposible, se buscan diferentes formas de reducir tanto los propios datos de origen como la representación del modelo.

Para ello, empezamos echando un vistazo a los eventos presentes en los datos, para ver si se puede eliminar o combinar alguno y reducir el tamaño del registro de eventos. Puesto que entre estos eventos se encuentra la acción de contestar y revisar a las preguntas de los 2 cuestionarios de forma individual, aprovechamos que también está como eventos el comienzo, la navegación y la entrega de cuestionarios para eliminar los eventos anteriores y obtener un modelo algo más sencillo, como se puede observar en la figura 3.

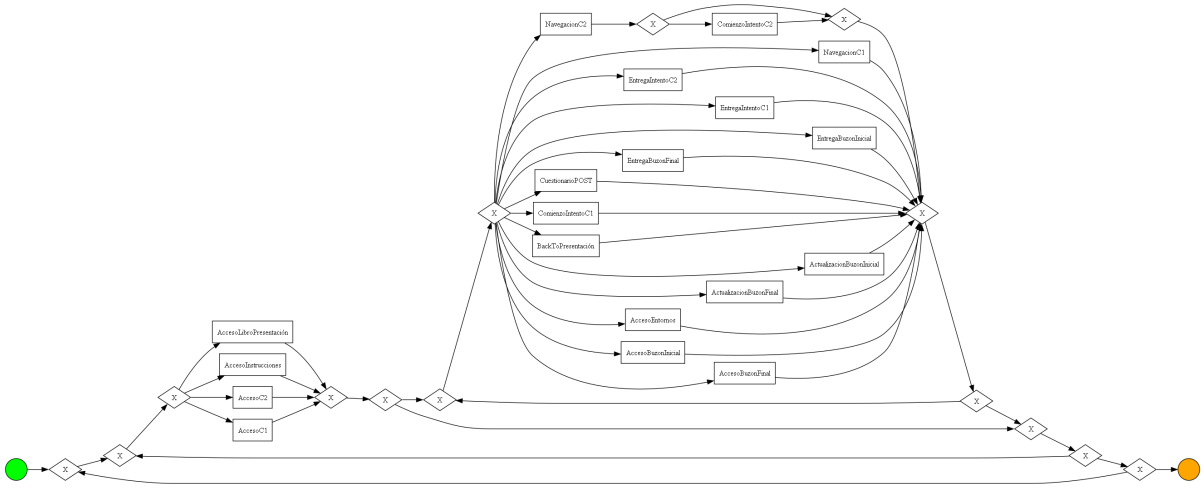


Figura 3: Figura del modelo general del curso A1 2015-2016 eliminando algunos eventos menores.

Esta vez, se pueden distinguir dos grupos de eventos principales, ambos correspondientes a nodos de elección secuenciales. Este modelo muestra que una de las primeras acciones

que se realiza al inicio de una sesión siempre es acceder o a un cuestionario, a las instrucciones o a la presentación, y que ya posteriormente se procede a realizar el resto de acciones y eventos, pudiendo repetir cualquiera de las elecciones. Este hecho era esperable, pues para contestar a los cuestionarios o hacer entregas es necesario primero acceder a ellas o leerse las instrucciones correspondientes.

Aparte de este hecho, no es sencillo sacar más información de este modelo, por lo que para intentar reducir algo más la complejidad del modelo, se prueba a modificar los parámetros de la función utilizada para obtener el modelo.

5.2. Modificación de parámetros de las funciones

El único parámetro que se puede modificar en la función para generar el modelo es *noise_threshold*, que permite filtrar y reducir parte del ruido presente en el log con los datos, permitiendo eliminar casos raros o desviaciones y dejar los más importantes/-representativos. Para encontrar un valor fijo con el que probar los siguientes modelos, se comparan los valores de *fitness*, precisión, capacidad de generalización y simplicidad del modelo anterior para diferentes valores de *noise_threshold* (de 0 a 1 con paso 0.05).

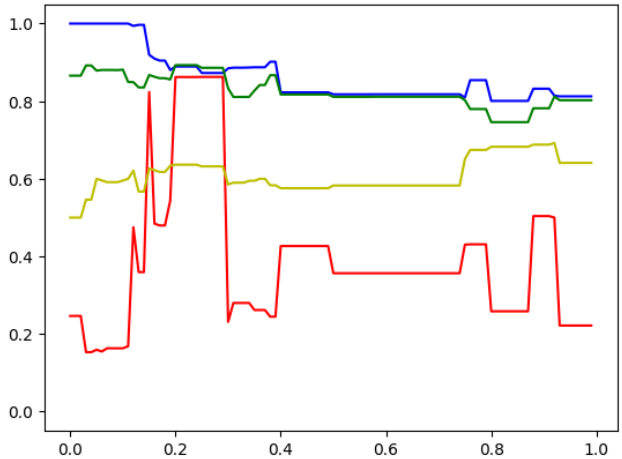


Figura 4: Gráfica que permite comparar los 4 valores de evaluación de un modelo para diferentes valores del *noise_threshold*. *Fitness* en azul, precisión en rojo, generalización en verde y simplicidad en amarillo.

En la figura 4, se puede observar que a medida que aumenta el umbral de ruido, la capacidad de generalización y el valor de *fitness* disminuyen, mientras que la simplicidad del modelo aumenta ligeramente. Estos comportamientos son esperables teniendo en cuenta que aumentar el nivel del límite de ruido filtra y reduce la cantidad de eventos realizados, por lo que es más sencillo que el modelo no se ajuste correctamente al proceso real y que a su vez sea más sencillo (pues hay menos procesos realizados y por tanto menos combinaciones que satisfacer).

Por otro lado, si el umbral de ruido es demasiado bajo, se tienen en cuenta todas las acciones, tanto las comunes como las poco frecuentes; lo que hace que sea mucho más sencillo que el modelo sobreajuste y, por tanto, sea más complejo y se ajuste mejor a nuestros datos. Además, la gran cantidad de posibles procesos presentes en el registro hace que el modelo tenga que tenerlos en cuenta, lo que lleva a que este modelo sea bastante general para cubrir la mayor cantidad de combinaciones posibles.

Respecto a la precisión del modelo, se puede ver que es baja tanto para valores bajos del límite de ruido como para valores altos, sólo superando el 50 % entre los valores 0.1 y 0.3. Esto puede deberse a que, con valores bajos, no se filtran los datos; por lo que el modelo puede presentar procesos contradictorios entre sí, lo que hace que se complique obtener una alta precisión. Por otro lado, valores altos para el límite de ruido hacen que se filtre una mayor cantidad de datos (incluyendo posibles eventos representativos), por lo que obtener un modelo de alta precisión es más complicado al contar con menos datos.

Al final, se ha llegado a la conclusión de que el valor óptimo de *noise_threshold* se encuentra entre el 0,2 y el 0,25, pues en ese umbral es en donde se filtra la cantidad de datos y registros justa para obtener modelos de calidad.

5.2.1. *Red heurística*

Aunque con los modelos BPMN y el análisis del valor óptimo del parámetro *noise_threshold* ya son suficientes para poder observar y analizar los procesos presentes en los datos, se prueba a modificar también los parámetros de la red heurística para tener una segunda representación del modelo, pues como se puede ver en la figura 17, los modelos BPMN pueden ser tan extensos que su representación no es la ideal.

Como primer acercamiento, al haber tres parámetros diferentes en vez de uno, se prueba a modificar uno cada vez tomando valores de 0.1 en 0.1 entre 0 y 1, dejando los otros 2 parámetros con sus valores por defecto. Analizando las múltiples redes obtenidas, se pueden extraer las siguientes conclusiones para cada parámetro:

- *dependency_threshold*: a medida que se aumenta este valor, la representación de la red se vuelve más simple, aunque las 10 redes generadas mantienen el mismo número de eventos (es decir, todos) relacionados entre sí, sin tener en ningún momento un segundo grafo aparte. Puesto que este parámetro determina la dependencia entre los eventos, este hecho implica que todos los eventos guardan una alta relación entre sí.

- *and_threshold*: independientemente del umbral escogido para este parámetro, las 10 redes generadas son idénticas entre sí. Recordando lo que permitía medir este parámetro, podemos concluir que todas las actividades presentes en los datos son igual de relevantes en el proceso. Este resultado guarda relación con el concluido con el parámetro *dependency_threshold*, pues que todos los eventos sean relevantes en el proceso obliga a que todos ellos aparezcan en el grafo principal.

- *loop_two_threshold*: para valores bajos de este parámetro (por debajo del valor por defecto), las redes generadas son idénticas entre sí de nuevo, pero al aumentar el umbral por encima del valor por defecto, se empiezan a notar pequeñas diferencias de representación entre las redes, disminuyendo el número de bucles dobles entre los eventos.

A continuación se pasa a generar nuevas redes heurísticas modificando los 3 parámetros al tiempo para compararlas. Siguiendo los mismos valores con los que se ha probado anteriormente, lo que en principio sería generar unas 1000 redes se puede reducir a unas 25 teniendo en cuenta las conclusiones que se acaban de obtener con las pruebas por separado.

Primero de todo, puesto que modificar el parámetro *and_threshold* no aporta nada a la red generada con nuestros datos, lo dejamos con su valor por defecto, lo que hace que el número de redes a generar se reduzca a la décima parte de lo que era inicialmente. Por otro lado, como se ha podido observar (más descaradamente en el caso del parámetro *loop_two_threshold*), seleccionar valores por debajo de los valores por defecto tampoco aporta nada a las redes generadas, por lo que puesto que en ambos casos este valor es 0,5, reducimos tanto el número de valores a probar para cada parámetro de 10 a 5 como el número de redes a generar a 25.

Para poder comparar más cómodamente estas 25 redes, se generan en vez de las redes heurísticas las redes Petri correspondientes, con las que se pueden comparar numéricamente los valores de evaluación de cada una según los parámetros. Puesto que esta vez estos valores dependen de 2 parámetros y no de uno como en 4, en vez de utilizar una gráfica lineal para representar los resultados, se utilizan 4 mapas de calor para cada medida de evaluación, mostrados en la figura 5.

Puesto que nos interesa que la red escogida tenga el mayor valor posible en las cuatro medidas, los parámetros óptimos serán aquellos en los que coincida un color más amarillo para los 4 mapas.

Por el lado del parámetro *dependency_threshold*, salvo el caso de la precisión, el resto aumentan a medida que lo hace el parámetro, por lo que podemos concluir que los valores más óptimos son los altos, entre 0,8 y 0,9. En el caso de la precisión, esta se mantiene más o menos constante entre los diferentes valores de este parámetro, variando más según el valor de *loop_two_threshold* y llegando a dar un bajón para *loop_two_threshold*=0,9 en valores altos de *dependency_threshold*, lo que hace indicar que valores altos en ambos parámetros no es lo óptimo y recomendable.

Este hecho se refuerza en el mapa que mide el *fitness*, pues los mejores valores se obtienen para valores bajos de *loop_two_threshold*. Por otro lado y en contraste, los mejores valores para la capacidad de generalización y simplicidad se obtienen para valores altos de este parámetro, por lo que podemos concluir que el valor óptimo para *loop_two_threshold* está alrededor de 0,7.

De esta forma, la red heurística generada del modelo general con los parámetros óptimos se muestra en la figura 6.

Aunque en este caso general los valores óptimos sean $d = 0{,}9$ y $l = 0{,}7$, es preferible tomar un valor menor del umbral de dependencia d para los modelos que sean de menor tamaño, pues una menor cantidad de datos hace que sea más complicado llegar al umbral de dependencia, pudiendo provocar que ningún par de eventos sea considere dependiente y que las redes heurísticas generadas salgan vacías.

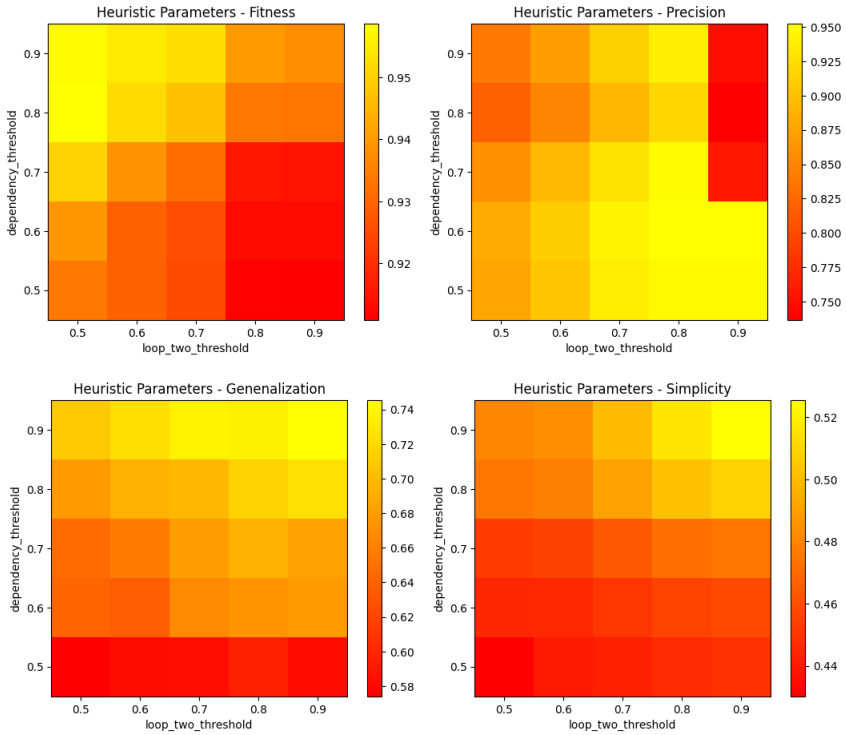


Figura 5: Comparación de las medidas de evaluación de las redes Petri variando los parámetros *dependency_threshold* y *loop_two_threshold*.

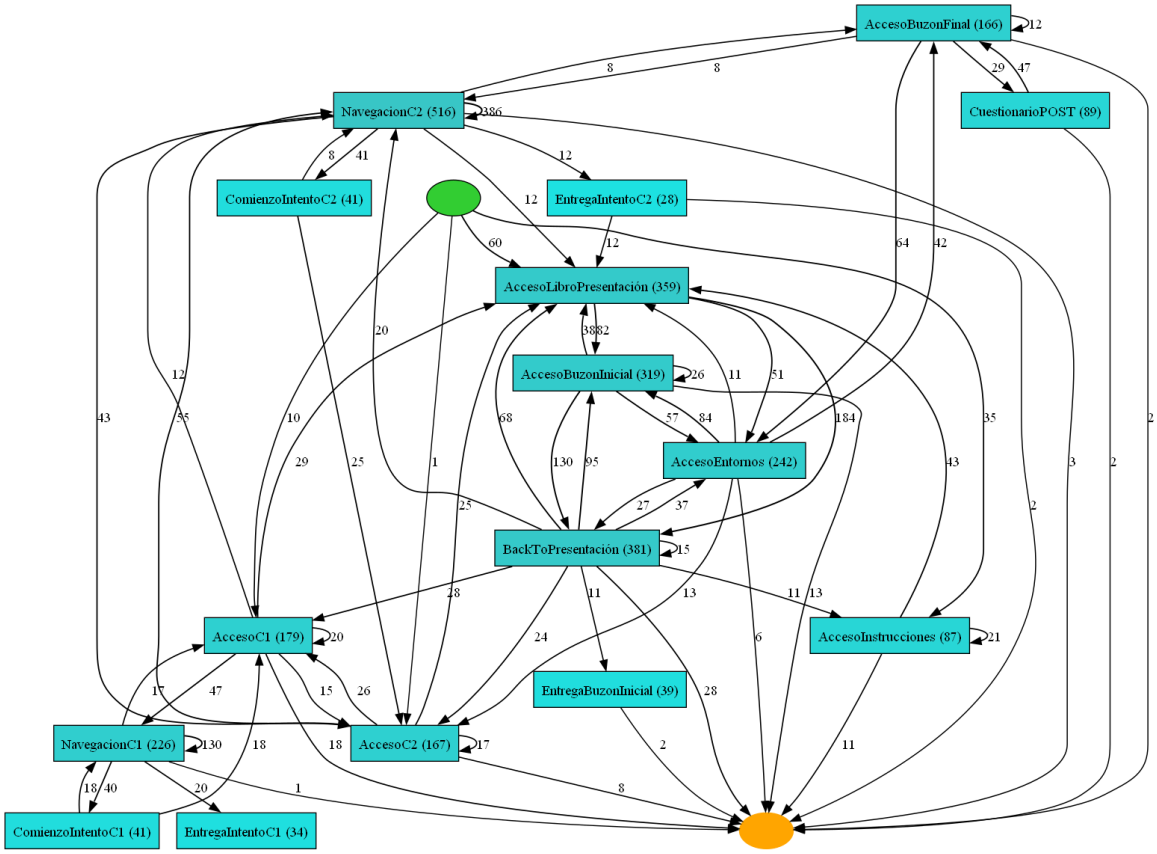


Figura 6: Red heurística del modelo general del curso A1 2015-2016 con los parámetros óptimos.

5.3. Modelos generales

En vista de los objetivos y datos que se tiene, es trivial pensar en combinar todos los datos de las dos universidades y cursos para generar un modelo completo con una gran cantidad de eventos; pero esto no va a ser posible debido a la gran cantidad de diferencias presentes en los datos de los diferentes cursos, destacando mayoritariamente los nombres de los eventos. Por ello, se procede a obtener los modelos generales de cada curso por separado para compararlos a posteriori.

5.3.1. Modelo general del curso A1 15-16 de la universidad presencial

Tras haber encontrado el valor óptimo para el umbral del ruido, el modelo que se ha generado inicialmente (figura 3) pasa a tener la forma mostrada en la figura 18.

En este nuevo modelo se puede observar una mayor segmentación del proceso, pasando de haber 2 elecciones principales a múltiples elecciones, quedándose la primera exactamente igual y habiéndose dividido la segunda elección del primer modelo en varias con menos eventos.

Entre estas nuevas elecciones, se ha separado por un lado los eventos relacionados con el primer cuestionario, los relacionados con el segundo cuestionario y entregas por otro, y el resto de eventos con un nodo *AND*, visibles en la figura 7. Este hecho, al igual que el que se concluyó con el modelo completo, es esperable si pensamos que es imposible realizar y entregar un cuestionario al tiempo, pero en cambio eventos como volver a la presentación o acceder a los entornos se pueden realizar en cualquier momento y orden, siendo independientes del resto.

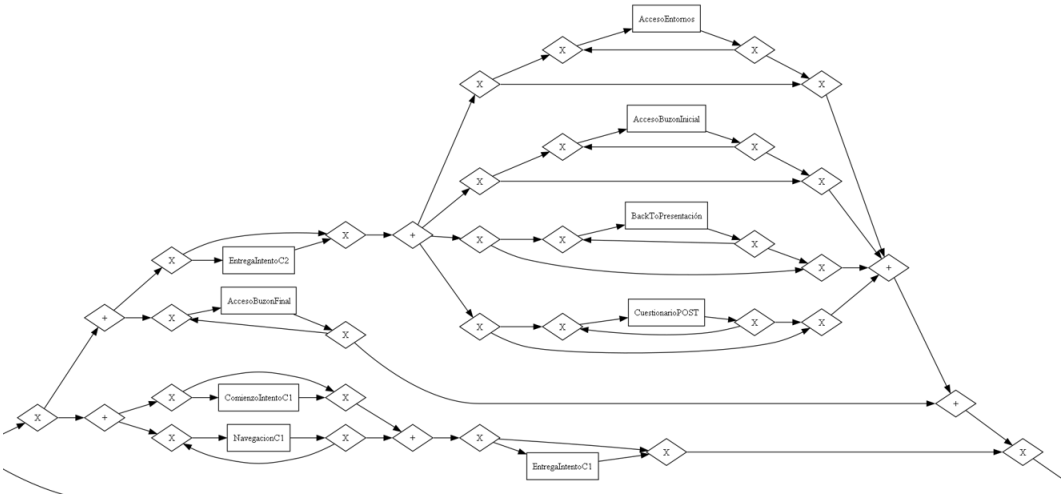
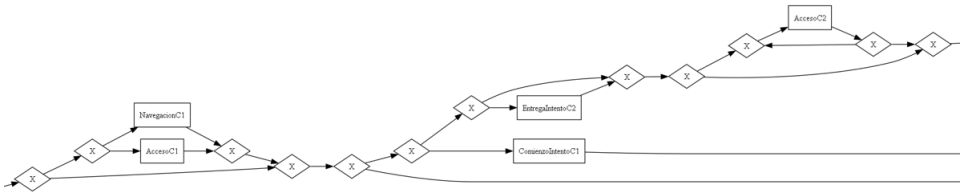


Figura 7: Recorte del modelo 18 en el que se pueden distinguir tanto el nodo *AND* como la separación entre los cuestionarios.

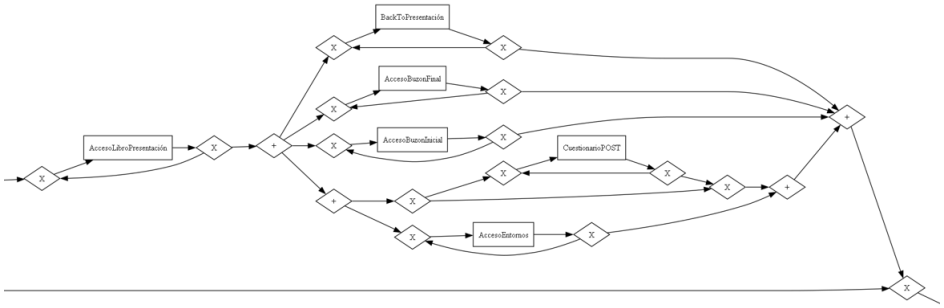
5.3.2. Curso A2 15-16 de la universidad presencial

Los registros de esta asignatura comparten tanto los nombres de los eventos con la primera como el uso de IDs para diferenciar a los estudiantes, por lo que se utiliza el mismo procedimiento para generar el modelo (figura 19).

Este modelo es más complejo que el obtenido con la asignatura A1 (Fig 18), pues el número de nodos de elección presentes en el modelo es mayor. De forma similar al modelo mencionado anteriormente, los eventos relacionados con los cuestionarios y entregas van por separado (aunque esta vez no en paralelo, véase figura 8a), mientras que los eventos agrupados en el nodo *AND* vuelven a ser los eventos de acceso y retorno, que se pueden realizar en cualquier momento (figura 8b).



(a) Recorte del modelo 19 en el que se vuelve a distinguir la separación de los eventos de los cuestionarios.



(b) Recorte del modelo 19 en el que se destaca el nodo *AND*.

Figura 8

Aunque existen pequeñas diferencias en cuanto al orden en el que se realizan las elecciones, la estructura principal de los procesos de los dos cursos son similares, por lo que podemos concluir que los estudiantes se comportan de forma similar entre cursos.

5.3.3. Curso 16-17 de la universidad presencial

En este caso y a diferencia de las asignaturas del curso anterior, el registro de este curso carece de IDs para distinguir a los estudiantes, por lo que su identificación se tiene que realizar con los nombres completos de los estudiantes.

Dado que en ni los modelos generados ni en este trabajo se hace mención alguna de los usuarios individuales, en principio no va a suponer un problema, pero al estar trabajando con datos no anonimizados, estos datos no se pueden mostrar directamente ni se puede dar acceso abierto a ellos, lo que complica en cierta medida el poder repetir el experimento de este trabajo. Para resolverlo, bastaría con asignar a cada estudiante un ID o coger el asignado por la plataforma, de forma similar a como se ha hecho con los datos de la universidad presencial.

Por otro lado, los nombres de eventos no coinciden exactamente con los de cursos anteriores, destacando por encima de todo la división de ciertos eventos en dos grupos AB y CD. Puesto que hay estudiantes de los grupos AB que han realizado acciones de los grupos CD y viceversa, no es posible separarlos limpiamente, por lo que se han unificado los nombres de los eventos para obtener un único modelo que se muestra en la figura 20.

Este modelo, en el que se pueden distinguir 4 partes, parece ser algo más sencillo que los anteriores, probablemente debido a la menor cantidad de datos disponibles para este curso. Se destaca el hecho de que los eventos de navegación se encuentran al final del proceso, y que los eventos de la tercera parte guardan cierto orden lógico, estando el acceso al libro como precedente del acceso al ejercicio y pudiendo realizar al tiempo (o con orden variante) otros eventos como el acceso a la explicación. Todo esto se muestra en la siguiente figura 9.

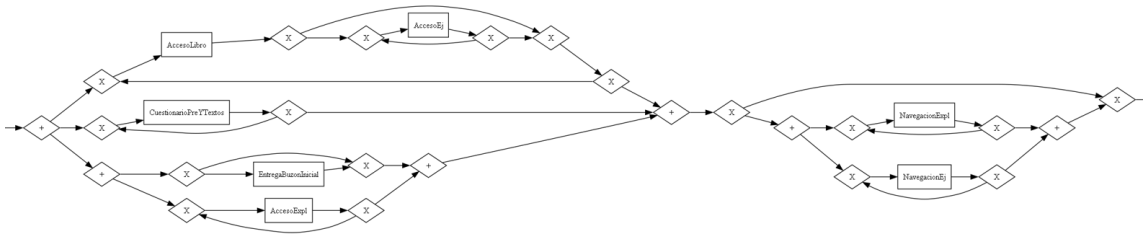


Figura 9: Recorte de las 2 últimas partes del modelo 20.

Puesto que eventos como la entrega del ejercicio y la visualización del segundo cuestionario se encuentran antes que eventos como el acceso al libro, el primer cuestionario o las navegaciones, se puede pensar que el proceso que se ha generado con PM carece de sentido, pero en realidad esto puede deberse simplemente a que una parte del alumnado han ido directamente a realizar la entrega final, saltándose o realizando por encima el resto de eventos.

Ignorando este hecho, el proceso en general no se aleja mucho de los generados con el resto de cursos.

5.3.4. Modelo de la universidad a distancia

En los datos del curso de la universidad a distancia, los nombres de eventos no se han normalizado ni unificado de la forma en la que se ha hecho con los datos de la universidad presencial, por lo que no se pueden comparar directamente los modelos en base a los eventos realizados. El modelo generado se muestra en la figura 21 del anexo.

Al igual que sucedió con el modelo de la figura 19, la gran cantidad de eventos y nodos presentes en el modelo hace que sea más complicado que el resto de modelos generados hasta ahora.

En este modelo se puede destacar el subproceso secuencial presente en la parte media del modelo respecto de los cuestionarios, pues muestra claramente el proceso que uno se espera a la hora de realizar uno (*intento de cuestionario* → *intento enviado* → *calificado* → *resumen visualizado*), destacado en la figura 10.

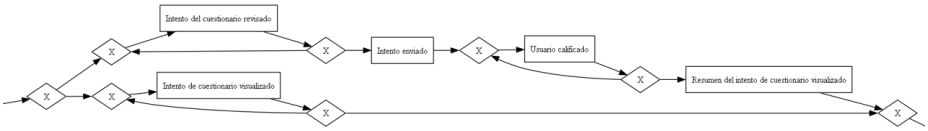


Figura 10: Recorte del subproceso esperado en los cuestionarios del modelo 21.

En general, podemos considerar a este modelo como un proceso sensato, dentro de lo que uno se puede esperar a priori. Al diferir los eventos de este curso con los de los modelos anteriormente generados, no se puede realizar una comparación directa de los procesos, pero teniendo en cuenta mayoritariamente los eventos coincidentes, los procesos vuelven a ser similares.

Respecto a alguna diferencia que se pueda destacar entre los modelos de ambas universidades, está por ejemplo el hecho de que los estudiantes de la universidad a distancia hacen un par de revisiones o visualizaciones al inicio y antes de desconectarse; mientras que en la universidad presencial esto no sucede, pues los estudiantes se conectan sólo para realizar la o las tareas y entregas que querían hacer. Esto se visualiza muy bien en la última parte del modelo de la universidad a distancia, parte que se muestra en la figura 11.

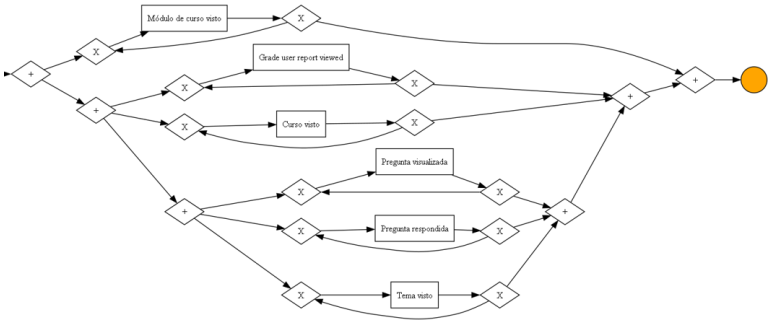


Figura 11: Recorte de la parte final del modelo 21 con los varios eventos de visualización y revisión.

5.4. Modelos según las calificaciones obtenidas

Para los siguientes modelos, al registro de eventos utilizado se le unen por los IDs de los estudiantes los datos del *dataset* completo; es decir, notas por cuestionario, grupo, horario, número de eventos por sesión, mes y componente y otros datos extra. Aunque se cuente con una gran cantidad de atributos, nos vamos a centrar en dividir a los estudiantes según las notas que hayan obtenido en los cursos.

5.4.1. Modelo por calificaciones de la universidad presencial

A partir de las calificaciones binarias (*Pass* y *Fail*) de los dos cuestionarios de los datos de la universidad presencial, se forman 3 grupos según se hayan aprobado ambos, suspendido uno o suspendido ambos, generando los 3 modelos mostrados en el anexo 22.
A simple vista, se puede ver que la mayor diferencia entre los tres modelos es la secuencialidad de los eventos y elecciones, además de la clara diferencia en la cantidad de eventos al visualizar las redes heurísticas, en la figura 12.

En el modelo de los aprobados, el proceso y modelo es más lineal, indicando que los estudiantes realizan todas (o la mayor parte de) las acciones a lo largo del curso. Además, el orden en el que se realizan estos eventos sigue un orden lógico respecto a la realización del ejercicio.

Por otro lado, en el modelo de los suspensos, la mitad de los eventos se encuentran agrupados bajo un mismo nodo *AND* que se indica en la figura 13, lo que implica que estos estudiantes o no siguen el orden lógico para realizar las actividades o sólo hacen un par de actividades del ejercicio.

Puesto que el modelo intermedio es una combinación de los otros dos modelos (la primera parte se asemeja al modelo con actividades paralelas de los suspensos, mientras que la segunda parte tiene una estructura más secuencial, similar al modelo de los aprobados), podemos concluir que estos tres modelos se ajustan perfectamente a lo

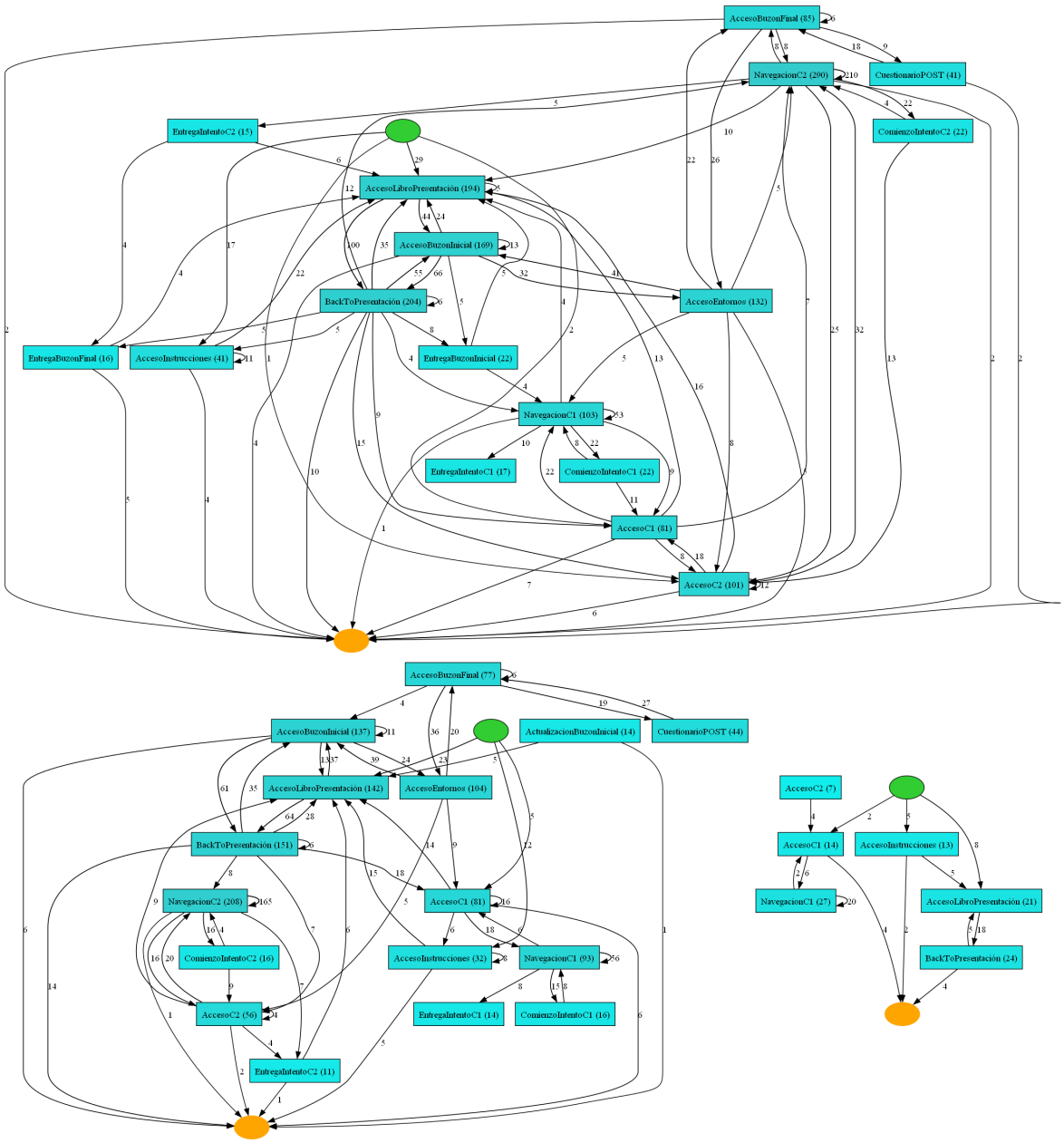


Figura 12: Red heurística de los modelos obtenidos según los aprobados de cada estudiante. Arriba los aprobados con buena nota, a la izquierda los aprobados con nota justa y a la derecha los suspensos.

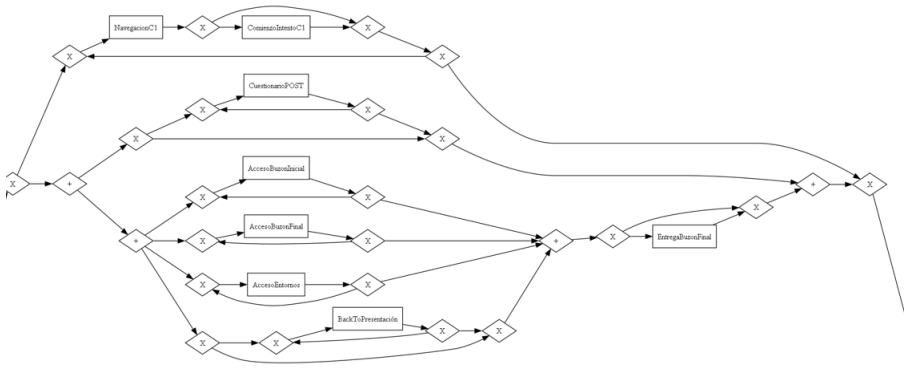


Figura 13: Recorte de la parte final del modelo 22 en el que se puede ver los varios eventos agrupados bajo el nodo *AND*.

que se esperaba de cada uno, mostrando cómo los estudiantes que realizan todas las actividades y siguen un orden lógico obtienen mejores resultados que aquellos que no.

Para averiguar si, además de las claras diferencias que se han detectado en los modelos de la figura 22, existen diferencias en los procesos de los estudiantes que aprueban con buena nota frente a los que aprueban con la nota justa, se ha añadido una nueva columna al conjunto de datos con la media de ambos cuestionarios, que nos sirve para separar a los estudiantes con media aprobada en dos subconjuntos según su nota media. Los dos nuevos modelos generados se muestran en el anexo 23.

Con estos modelos, se puede ver que la estructura lineal del modelo de los aprobados anterior está más presente en el modelo de los estudiantes con notas altas, mientras que la agrupación de eventos bajo un mismo par de nodos es más recurrente en el modelo de los estudiantes con notas bajas, teniendo este una estructura similar al modelo intermedio de 12.

Como último apunte, también se ha generado un modelo con los datos de los no presentados (ver figura 14), en el que se puede observar claramente que el interés de estos estudiantes por el ejercicio es casi nulo, pues la interacción con la plataforma es mínimo.

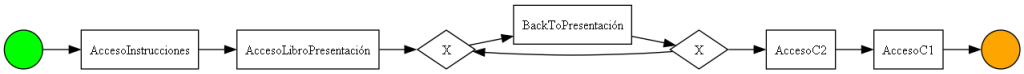


Figura 14: Modelos de los estudiantes no presentados en los cuestionarios.

5.4.2. Modelo por calificaciones de la universidad a distancia

Para el caso de la universidad a distancia, se vuelve a recurrir a la clasificación binaria para generar 2 modelos, que se muestran en el anexo 24.

Esta vez y al contrario que en el modelo 23 de la universidad presencial, la estructura lineal está presente en el modelo de los suspensos en vez de en el de los aprobados, que contiene una mayor cantidad de nodos *AND* . Aún así, es importante observar que, en el modelo de los suspensos, en la mayor parte de los eventos (o directamente en todos) hay un nodo de elección que permite omitir ese evento durante el proceso, lo que implica que, de nuevo, estos estudiantes no realizan todas las actividades que deberían.

En el caso del modelo de los aprobados, no sucede lo mismo que en el modelo de los suspensos, pues una gran cantidad de los nodos que agrupan a los eventos son nodos *AND*, lo que indica la realización en paralelo de los diferentes eventos o simplemente una alta variación en el orden en el que estos estudiantes han hecho las actividades. Esto se puede visualizar claramente en la figura 15. Visto de esta forma, estos modelos y los de la universidad presencial no se diferencian en tanto como aparentaba al inicio.

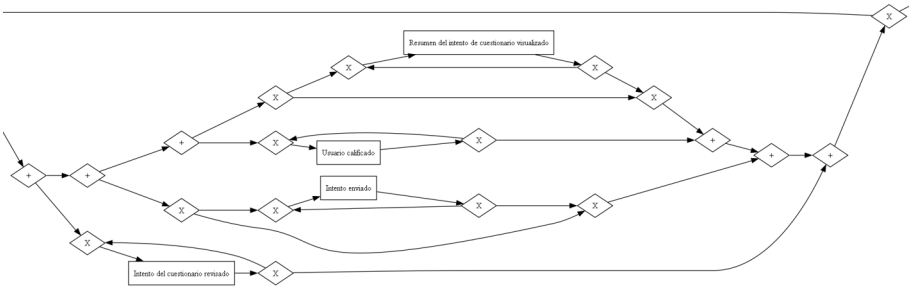
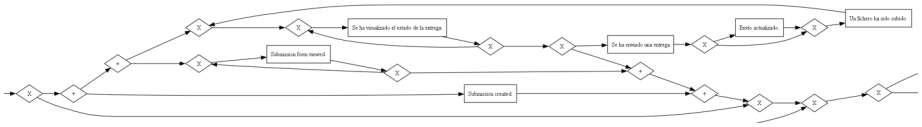
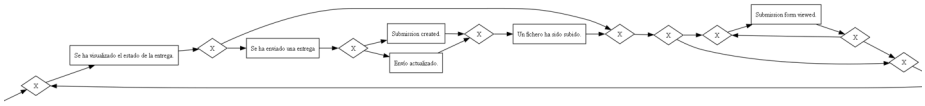


Figura 15: Extracto de una parte del modelo de los aprobados 24 en el que se puede ver los varios eventos agrupados bajo varios nodo *AND*.

Por último, en cuanto al orden en el que aparecen los eventos en el proceso, de nuevo se sigue el orden esperado de *lección visualizada* → *cuestionarios realizados y entregas hechas* → *revisión y actualización de entregas y cuestionarios*, lo que concuerda con los modelos generales obtenidos anteriormente. Estas secuencias se destacan para ambos modelos en las figuras 16a y 16b respectivamente.



(a) Recorte del modelo de los aprobados 24 en el que se distingue la secuencia de las entregas.



(b) Recorte del modelo de los suspensos 24 en el que se distingue la secuencia de las entregas.

Figura 16

6. Conclusiones

En este proyecto, se han aplicado técnicas de ciencia de datos y de *Process Mining* para, partiendo de los registros que genera la plataforma de *Moodle* sobre la actividad los estudiantes al interactuar con ella, obtener los procesos de estudio del alumnado; con el objetivo de encontrar diferencias con el proceso esperado por los docentes y así dar pie a mejorar y optimizar estos procesos de aprendizaje. Una vez finalizado el procesado y análisis de los datos y modelos generados, se procede a introducir las conclusiones extraídas.

Primero de todo, los modelos que se han generado han resultado ser sólidos y tener sentido. El orden en el que se ejecutan los eventos sigue un orden lógico, no muy alejado del proceso que se esperaba que siguiesen los estudiantes; exceptuando casos como el paralelismo a la hora de realizar los dos cuestionarios (que parecen realizarse al tiempo en vez de ir un detrás del otro, como se ha podido ver en el extracto 7) o porque parece ser que no en todos los casos se revisa el material antes de hacer las entregas y cuestionarios. En vista del objetivo general para este proyecto y del primer objetivo concreto, se puede decir que se han podido completar correctamente, pues ninguno de los modelos generados se aleja de la idea de proceso de aprendizaje que se esperaba.

Respecto al primer objetivo concreto establecido, con el que se buscaba conocer cómo de cercano o lejano es el proceso de aprendizaje real del esperado, se puede concluir satisfactoriamente y como ya se ha mencionado que son similares. Puesto que no son exactamente idénticos, a continuación se pasa a repasar las mayores diferencias detectadas.

- En el curso A1 15-16 de la universidad presencial, ya se ha mencionado el paralelismo al realizar los cuestionarios, y el hecho de que no todos los estudiantes parecen revisar las instrucciones y material necesarios para hacer las entregas y cuestionarios.
- Para la segunda asignatura A2 de la misma universidad, se tiene que los cuestionarios sí parecen realizarse en el orden indicado, pero estos se completan antes de las entregas que deberían hacerse previamente (el extracto 8a ocurre previo al extracto 8b).
- En el caso de la asignatura A1 del curso del año posterior 16-17, se vuelve a destacar que los eventos de navegación se realizan al final, tras los ejercicios y cuestionarios.
- Por último, el modelo del curso de la universidad a distancia es el que menos diferencias tiene con en proceso esperado, pudiendo mencionar como diferencia la visualización del tema al final del proceso (visible en la extracción 11).

En conjunto, los procesos de aprendizaje seguidos por los estudiantes al realizar la actividad indicada son los esperados, por lo que bastaría averiguar una solución a las

diferencias que se acaban de indicar para optimizar el modelo; aunque esto es algo que muy probablemente se tenga que conseguir con otro tipo de datos y más análisis de nuevos modelos de proceso.

Por otro lado, los modelos generados según el rendimiento del alumnado también guardan relación entre sí, notándose similitudes entre los procesos seguidos por estudiantes de igual rendimiento en cursos diferentes; pero a su vez viéndose diferencias en los procesos seguidos por los estudiantes con buen rendimiento con respecto a los que no. Siendo más específicos, para ambos tipos de enseñanza se ha podido ver una clara diferencia en su estructura, siendo en unos casos más lineal y secuencial y, en otros, habiendo una mayor agrupación bajo nodos tanto *XOR* como *AND*.

A pesar de todo, resulta desconcertante el hecho de que estas estructuras no se mantienen según los resultados obtenidos, siendo completamente opuesta entre modelos de estudiantes con resultados similares, pero de universidades diferentes. Esta diferencia es altamente probable que derive de las propias diferencias entre los modos de enseñanza presencial y a distancia, pues el segundo requiere una mayor interacción con la plataforma, haciendo que las diferentes sesiones de cada estudiante difieran más entre sí en cuanto a eventos realizados (lo que explica la agrupación de eventos bajo un mismo par de nodos). En cambio, en la modalidad presencial, la interacción con la plataforma es menor, por lo que la linealidad y secuencialidad en los eventos no es sorprendente para aquellos estudiantes que revisan todas las actividades planteadas.

En conclusión, las posibles optimizaciones de los procesos para mejorar el rendimiento de los estudiantes con peores notas depende completamente de la modalidad de enseñanza (presencial o a distancia), pudiendo apoyarse otra vez en nuevos datos y modelos de procesos para observar nuevas excepciones o desviaciones de lo esperado.

Por último, centrándonos en el último objetivo concreto y volviendo a los modelos generales, los procesos que se han obtenido de las dos universidades según el tipo de enseñanza (presencial o virtual) son, ignorando las diferencias en la cantidad y los nombres de los eventos, más similares entre sí de lo que se podía inicialmente esperar. De hecho, los modelos generales obtenidos de los diferentes cursos son parecidos entre sí, compartiendo orden de eventos generales en los procesos (*acceso o visualización de actividades* $\rightarrow$ *cuestionarios* $\rightarrow$ *revisiones varias*).

En el caso de la universidad a distancia, los eventos de ambos cuestionarios están unificados, por lo que no se puede comentar sobre el paralelismo o secuencialidad de su realización. En cuanto a las revisiones y visualizaciones finales e iniciales presentes en el modelo a distancia (figura 11) que no están presentes en el modelo presencial (como se mencionó en la sección anterior), se puede deber a la misma razón mencionada en los modelos según los resultados, pues debido a dar el curso a distancia, se interactúa más con la plataforma, y es más probable conectarse para realizar más actividades que las entregas y cuestionarios (como sucede en la modalidad presencial).

Con los múltiples modelos generados, queda claro que hay una diferencia en la forma de interactuar con la plataforma de *Moodle* según el tipo de modalidad cursada. Es por esto por lo que, para poder optimizar del todo los procesos de aprendizaje, sería necesario generar modelos extra a partir de otros subconjuntos de estudiantes, por ejemplo según el grupo en el que hayan estado o el tiempo invertido en cada tarea o actividad, al igual que como se ha concluido en las ocasiones anteriores.

La obtención de otros modelos en base a otros datos además del rendimiento aportan un nuevo punto de vista a los procesos de aprendizaje, pudiendo detectar nuevas diferencias no visibles en los procesos según el rendimiento del alumnado. Debido a la inmensa cantidad de posibilidades y datos disponibles, esta ampliación y continuación es apta para trabajo futuro, como continuación de lo expuesto en este proyecto.

Aún así, existen por ejemplo estudiantes capaces de entender y sacar el curso sencillamente sin interactuar demasiado con los materiales disponibles en la plataforma; y al mismo tiempo, hay algunos estudiantes a los que les cuesta obtener buena nota aún siendo trabajadores y revisando todo el material. Por ello, la división de los estudiantes por rendimiento no es necesariamente un indicador del proceso óptimo a seguir. Es por esto por lo que optimizar el proceso de aprendizaje completamente no es del todo posible basándose únicamente en PM y la extracción de procesos, quedando esto como una herramienta más de apoyo y quedando en manos de los docentes las decisiones y la organización final del curso.

Referencias

[1] Accessed: 17/9/2023. [Online]. Available: https://moodle.org/

[2] "Bpmn specification - business process model and notation," 1997, accessed: 1/9/2023. [Online]. Available: https://www.bpmn.org/

[3] N. S. N. Ayutaya, P. Palungsuntikul, and W. Premchaiswadi, "Heuristic mining: Adaptive process simplification in education," in *2012 Tenth International Conference on ICT and Knowledge Engineering*, 2012, pp. 221–227.

[4] A. Bogarín, C. Romero, R. Cerezo, and M. Sánchez-Santillán, "Clustering for improving educational process mining," in *Proceedings of the Fourth International Conference on Learning Analytics And Knowledge*, ser. LAK '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 11–15. [Online]. Available: https://doi.org/10.1145/2567574.2567604

[5] R. Campos-Rebelo, A. Costa, and L. Gomes, "Finding learning paths using petri nets modeling applicable to e-learning platforms," in *Technological Innovation for Value Creation*, L. M. Camarinha-Matos, E. Shahamatnia, and G. Nunes, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 151–160.

[6] F. I. for Applied Information Technology, "State-of-the-art-process mining in python," accessed: 1/9/2023. [Online]. Available: https://pm4py.fit.fraunhofer.de/

[7] F. Genovese and J. Herold, "The mathematical specification of the statebox language," *CoRR*, vol. abs/1906.07629, 2019. [Online]. Available: http://arxiv.org/abs/1906.07629

[8] D. Kinshuk, N.-S. Chen, I.-L. Cheng, and S. W. Chew, "Evolution is not enough: Revolutionizing current learning environments to smart learning environments," *International Journal of Artificial Intelligence in Education*, vol. 26, 02 2016.

[9] A. Roegiers, E. Merchie, and H. Van Keer, "Opening the black box of students' text-learning processes: A process mining perspective," *Frontline Learning Research*, p. 40–62, 2020.

[10] N. Trc˘ka, M. Pechenizkiy, and W. van der Aalst, "Process mining from educational data," *Handbook of educational data mining*, pp. 123–142, 2010.

[11] W. van der Aalst, T. Weijters, and L. Maruster, "Workflow mining: Discovering process models from event logs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 9, p. 1128–1142, 2004.

[12] W. M. VAN DER AALST, "The application of petri nets to workflow management," *Journal of Circuits, Systems and Computers*, vol. 08, no. 01, p. 21–66, 1998.

[13] W. van der Aalst, A. Adriansyah, A. K. de Medeiros, F. Arcieri, T. Baier, T. Blickle, J. C. Bose, P. van den Brand, R. Brandtjen, J. Buijs, and et al., "Process mining manifesto," *Business Process Management Workshops*, p. 169–194, 2012.

[14] A. Weijters, W. Aalst, and A. Medeiros, *Process Mining with the Heuristics Miner-algorithm*, 01 2006, vol. 166.

Anexo: modelos completos

Código utilizado para la parte de PM: `https://github.com/DarkShayminDii/TFM_PM_Education.git`

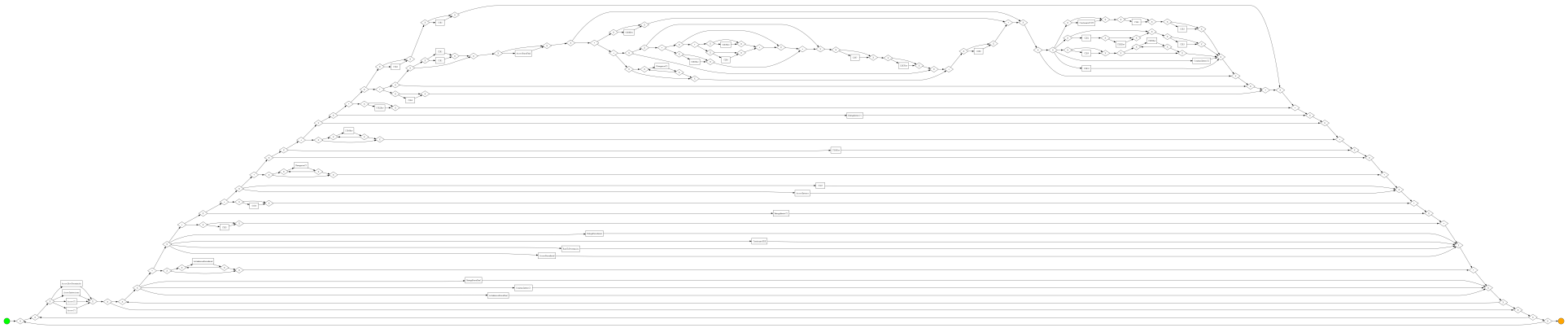


Figura 17: Figura del modelo general del curso A1 de la universidad presencial 2015-2016.

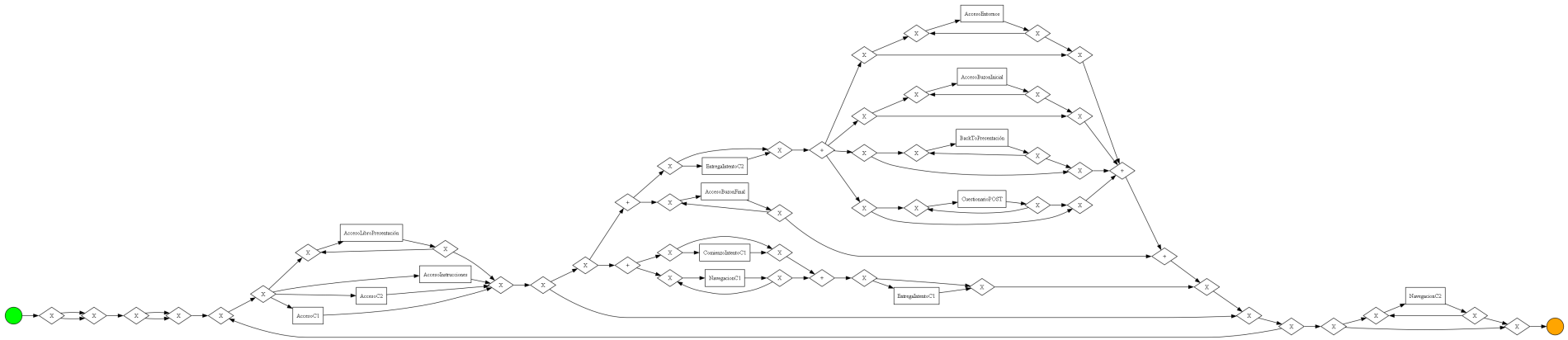


Figura 18: Figura del modelo general del curso A1 2015-2016 eliminando algunos eventos menores y con umbral de ruido *noise_threshold*= 0,2.

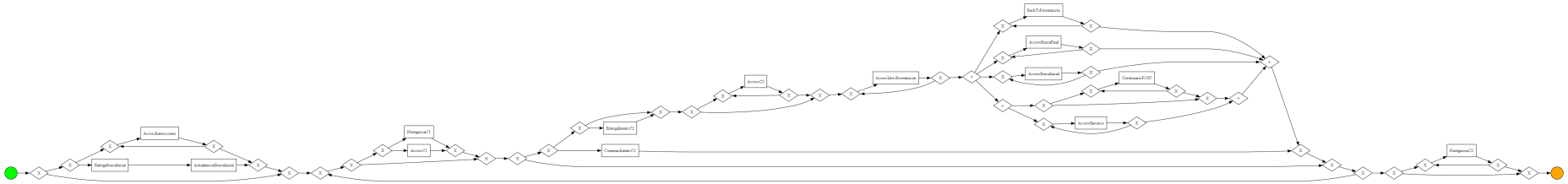


Figura 19: Figura del modelo general de la asignatura A2 del curso 15-16 de la universidad presencial.

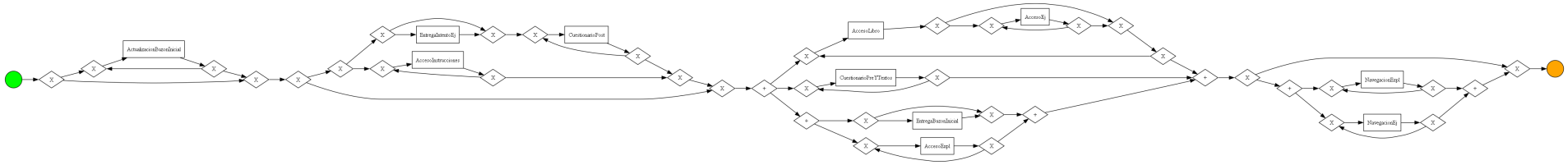


Figura 20: Figura del modelo general sin grupos del curso 16-17 de la universidad presencial.

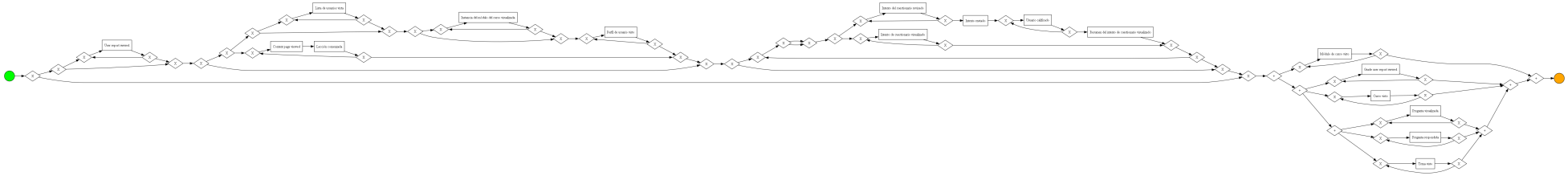


Figura 21: Figura del modelo general del curso de la universidad a distancia.

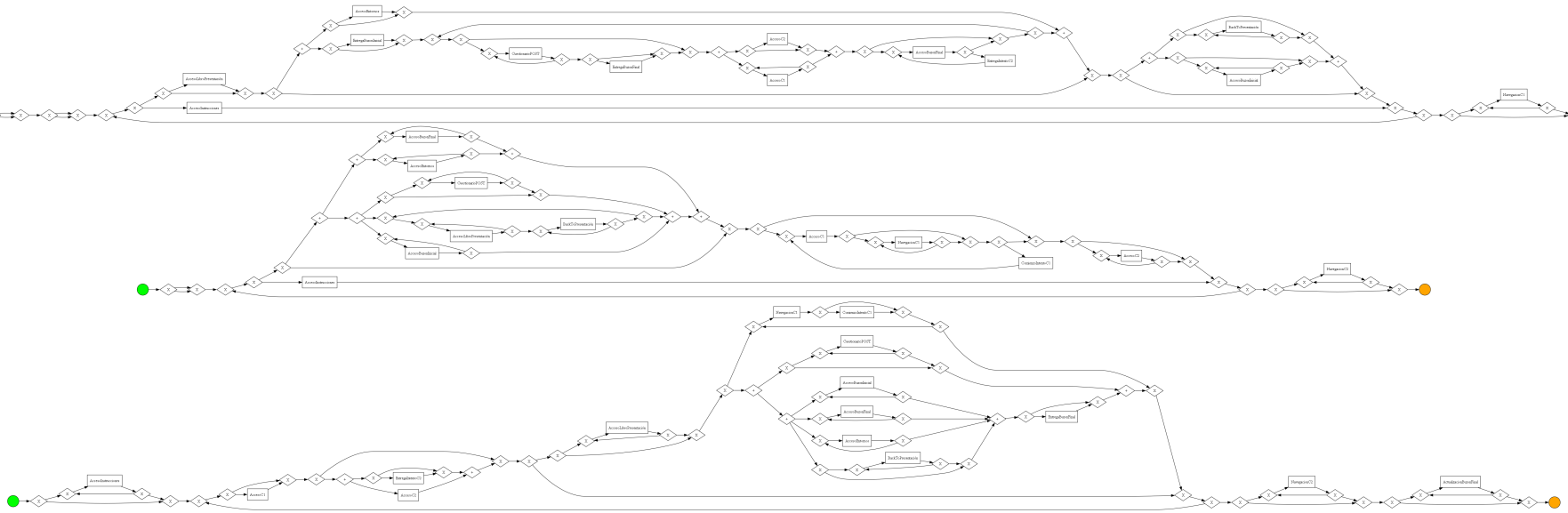


Figura 22: Figuras de los modelos obtenidos según los aprobados de cada estudiante en la universidad presencial. Arriba los que han aprobado ambos, abajo los que han suspendido ambos y en medio los que han aprobado uno.

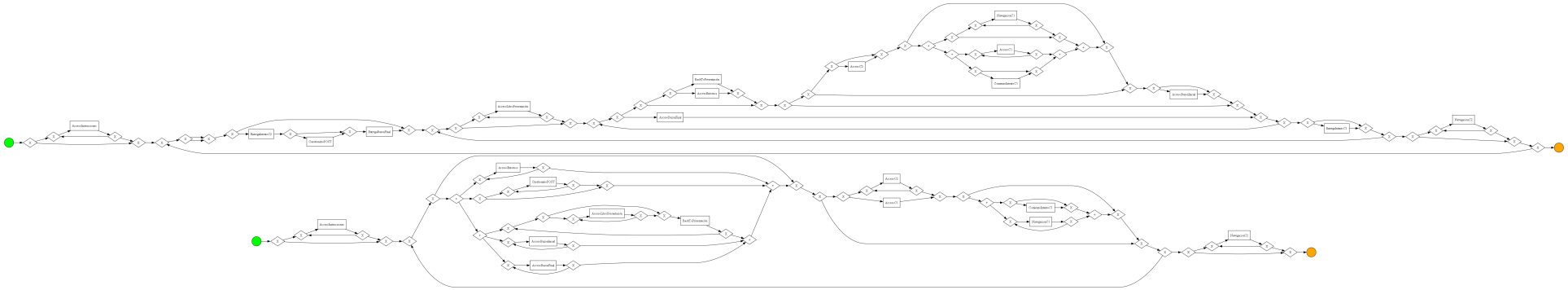


Figura 23: Figuras de los modelos obtenidos según la nota media de cada estudiante en la universidad presencial. Arriba el modelo de los estudiantes con una calificación media de más de 1,5, abajo el modelo de los estudiantes con una calificación media aprobada de menos de 1,5.

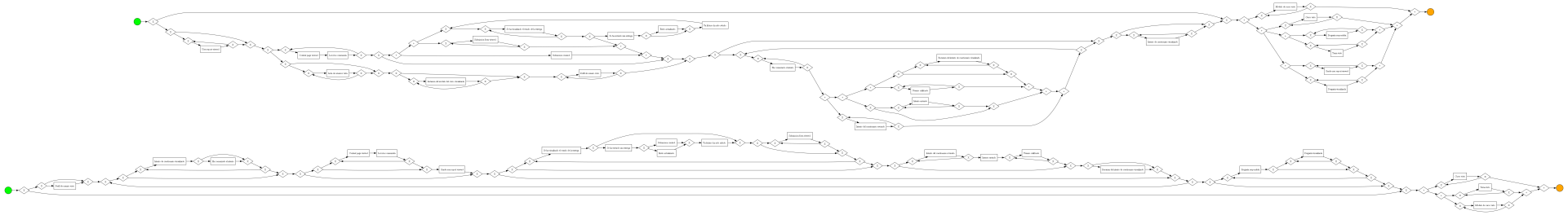


Figura 24: Figuras de los modelos obtenidos según los resultados del alumnado de la universidad a distancia. Arriba el modelo de los aprobados y abajo el modelo de los suspensos.