UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

IMPLEMENTAR MÓDULO DE EXPLORACIÓN Y VISUALIZACIÓN DE DATOS PARA SOFTWARE DASHAI

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

NICOLÁS IGNACIO OLGUÍN STEGMAIER

PROFESOR GUÍA:
FELIPE BRAVO MÁRQUEZ

MIEMBROS DE LA COMISIÓN:
Cinthia Sánchez Macías
Valentin Barriere
Claudia López Moncada

SANTIAGO DE CHILE
2024

RESUMEN DE LA MEMORIA PARA OPTAR AL
TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN
POR: NICOLÁS IGNACIO OLGUÍN STEGMAIER
FECHA: 2024
PROF. GUIA: FELIPE BRAVO MÁRQUEZ

IMPLEMENTAR MÓDULO DE EXPLORACIÓN Y VISUALIZACIÓN DE DATOS PARA SOFTWARE DASHAI

El aprendizaje de máquinas es una de las sub disciplinas de la inteligencia artificial que ha tenido un gran auge en los últimos años, debido a su capacidad para generar conocimiento a partir de los datos y facilitar la toma de decisiones.

Como esta sub disciplina es aplicada en una amplia variedad de campos, profesionales de distintas áreas y distintos niveles de experiencia pueden verse interesados en utilizar estas herramientas. Sin embargo, la curva de aprendizaje para utilizar estas herramientas puede ser muy empinada, lo que puede desmotivar a los usuarios menos experimentados.

En este contexto, DashAI busca disminuir las barreras de entrada en el área del aprendizaje de máquinas, proporcionando una plataforma de código abierto que permita a los usuarios, independiente de su nivel de experiencia, experimentar y entrenar diversos modelos de aprendizaje de máquinas de manera sencilla mediante una interfaz gráfica interactiva, sin la necesidad de escribir código.

DashAI es un proyecto en desarrollo, y previo a este trabajo no contaba con las funcionalidades de explorar con mayor detalle los datos cargados en la plataforma. Esta funcionalidad es indispensable para que los usuarios puedan entender los datos con los que están trabajando, y así poder tomar decisiones informadas al momento de entrenar un modelo de aprendizaje de máquinas.

En este trabajo se presenta la implementación de un módulo de exploración y visualización de datos para DashAI, que permite a los usuarios generar «exploraciones» que detallen de manera tabular o gráfica las características de los datos cargados en la plataforma, permitiendo a los usuarios realizar un análisis exploratorio de los datos de manera sencilla y rápida.

Para concluir, la implementación del módulo se realizó con éxito, agregando a las funcionalidades de DashAI una herramienta de exploración de datos extensible y que se alinea a la naturaleza asíncrona que caracteriza a la plataforma.

Even the mightiest hero cannot win a war alone.

Agradecimientos

Me gustaría agradecer a mi familia por su apoyo incondicional durante toda mi vida. En especial a mis padres, por siempre estar ahí para mí, por enseñarme a ser una persona honesta y trabajadora, y por darme la oportunidad de crecer y aprender. Este agradecimiento se extiende también a la familia Jeldes por ser una segunda familia para mí.

También quiero agradecer a todos mis amigos más cercanos, por apoyarme en los momentos difíciles y por celebrar conmigo en los momentos felices. Especialmente a Fran, Dimitri, Noemí, Gian, Tito, Tabo y Chukky, por ser los mejores amigos que uno podría pedir. No hay palabras para expresar lo agradecido que estoy por tenerlos en mi vida y también los considero parte de los que formaron mi forma de ser.

Agradezco a mis amigos más cercanos del DCC, Dani, Vicho, Cristian, David, Bas, Ken y Camilo por que cada uno de ustedes ha aportado en mi vida, ya sea jugando, trabajando o simplemente compartiendo un rato juntos.

Tampoco quiero dejar fuera a mi grupo de amigos, los csm en ventana, cada uno de ustedes me ha hecho pasar muchos buenos momentos durante toda la universidad. Siempre están en mi corazón y son todos unos cracks.

Finalmente, agradezco a Felipe, Cristian, Daniel, Natalia, Isaías y a todo el equipo que participa en el desarrollo de DashAI, por darme la oportunidad de trabajar en un proyecto tan interesante y desafiante, y por su apoyo y guía durante todo el proceso de desarrollo de este trabajo.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1 Contexto

El aprendizaje de máquinas es una sub disciplina de la inteligencia artificial que se enfoca en el desarrollo de algoritmos y técnicas que permiten a los sistemas computacionales aprender de los datos y generar conocimiento a partir de ellos. En la actualidad, el aprendizaje de máquinas se ha vuelto una herramienta fundamental en la toma de decisiones en distintas áreas, como la medicina, la economía, la ingeniería, entre otras[1]. Sin embargo, es una disciplina que requiere de un alto nivel de conocimiento teórico y técnico para poder ser aplicada de manera efectiva, generando una brecha significativa entre los expertos en el área y los usuarios menos experimentados.

En este contexto, DashAI[2] busca disminuir las barreras de entrada en el área de aprendizaje de máquinas, proporcionando una plataforma de código abierto que permita a usuarios, con distintos niveles de experiencia en manejo de datos, experimentar y entrenar diversos modelos de aprendizaje de máquinas de manera sencilla mediante una interfaz gráfica interactiva, sin la necesidad de programar. DashAI busca unificar en una misma plataforma las distintas etapas del proceso de aprendizaje de máquinas, desde la carga de datos, la exploración y preprocesamiento de estos, hasta la evaluación y visualización de los resultados obtenidos.

Hoy en día existen diversas herramientas de software que ofrecen funcionalidades similares a las que se busca implementar en DashAI. La principal diferencia de DashAI con estas herramientas es que busca ser una plataforma capaz de ofrecer todas las funcionalidades necesarias para el aprendizaje de máquinas en un solo lugar, de manera abierta y mantenida por la comunidad, permitiendo a los usuarios contribuir con nuevas funcionalidades y mejoras, manteniendo la plataforma actualizada con el estado del arte.

Este primer capítulo introductorio contiene distintas secciones. En primera instancia se describe el contexto en el que se desarrolla DashAI y el objetivo que busca cumplir en el área de aprendizaje de máquinas.

Luego se describe a grandes rasgos la situación actual de DashAI, detallando las funcionalidades que ofrece la plataforma y los componentes principales que la componen.

En la sección de problema y relevancia se analizan los procesos de aprendizaje de máquinas y se destaca la importancia de la etapa de comprensión de los datos en el éxito de un proyecto de aprendizaje de máquinas. Junto con esto, se expone la falta de funcionalidades en DashAI para lograr esta etapa de manera efectiva y se justifica la necesidad de implementar un módulo de exploración y visualización de datos en la plataforma.

A continuación, en la sección de objetivos se presentan los objetivos generales y específicos planteados para este trabajo, definiendo lo que se busca lograr con la implementación del módulo de exploración y visualización de datos en DashAI.

En la sección de metodología se describe la metodología de desarrollo seguida durante el trabajo, detallando las herramientas y técnicas utilizadas para el diseño e implementación.

Finalmente, en la sección de solución general se presenta un resumen de la solución propuesta, para dar una visión general de como se abordaron los problemas planteados en este trabajo.

1.2 Situación actual

DashAI está compuesta por un back-end desarrollado en Python[3] y un front-end desarrollado en React[4]. Este último consume endpoints del back-end para, mediante la Application Programming Interface (API) de DashAI, realizar las distintas operaciones que ofrece la plataforma.

Actualmente, DashAI permite a los usuarios cargar datos en formato CSV o JSON, entrenar modelos de clasificación, mostrar las métricas relevantes resultantes de estos entrenamientos y finalmente, aplicar métodos de explicabilidad sobre los modelos entrenados. Así, la plataforma ofrece, de manera básica y limitada, las funcionalidades necesarias para el aprendizaje de máquinas. Existen proyectos en desarrollo para agregar funcionalidades de transformación de datos (o «converters»), inferencia de tipos en la carga de datos, aplicaciones de modelos de inteligencia artificial generativa, un sistema de plugins, utilizar los modelos entrenados para funcionalidad de predicciones y la implementación del módulo de este trabajo.

1.2.1 Componentes de DashAI

DashAI está organizado en torno a los siguientes componentes principales:

- **Datasets**: Representan un conjunto de datos cargados en la plataforma. Los datos son almacenados en tablas de *Arrow*[5] en el sistema de archivos del sistema operativo, permitiendo un acceso rápido y eficiente a los datos cargados y pueden estar segmentados en distintos *splits* para facilitar entrenar y evaluar modelos de aprendizaje de máquinas.
- **Experimentos**: Representan un experimento de aprendizaje de máquinas creado por el usuario. Puede aplicar más de un modelo de aprendizaje de máquinas a un mismo *dataset*, pero solo de un tipo de tarea de aprendizaje de máquinas (*Task*). Esto permite, para una misma tarea objetivo, comparar distintos modelos y configuraciones.
- **Runs**: Representan una instancia de un modelo de un *experimento*. Estos se ejecutan de manera asíncrona como *Jobs*. Contienen los resultados de las métricas obtenidas al evaluar el modelo.
- **Objeto Configurable (*ConfigObject*)**: Representan un componente configurable por un usuario mediante parámetros. Estos son definidos por esquemas de *pydantic*[6], lo que permite validar los campos automáticamente. Ofrecen una interfaz unificada que permite a los usuarios configurar los componentes de la plataforma de manera sencilla y eficiente. Además, tienen capacidades recursivas, lo que permite crear componentes compuestos que contienen otros objetos configurables.

1.2.1.1. Componentes extensibles

Los componentes extensibles en la plataforma contienen una clase base abstracta que define los métodos y atributos necesarios para su funcionamiento. Los usuarios pueden extender estas clases base y los objetos configurables para crear nuevos componentes que se integren de manera transparente con la plataforma. Algunos de los componentes extensibles en la plataforma son:

- **Jobs**: Representan un proceso que se ejecuta de manera asíncrona en la plataforma. Esto es logrado mediante una cola de trabajos ejecutada en un thread separado. No extiende de *ConfigObject*, pero las nuevas funcionalidades que utilicen la asincronía de DashAI deberán ser implementadas heredando de su clase base. Actualmente, DashAI cuenta con jobs para entrenar modelos y para aplicar métodos de explicabilidad.
- **Task**: Representa una tarea de aprendizaje de máquinas. No extiende de *ConfigObject*, pero es un componente extensible que permite a los usuarios definir nuevas tareas de aprendizaje de máquinas. Actualmente, DashAI cuenta con tareas de clasificación tabular y de imágenes, además de traducción y clasificación de texto. Definen las cardinalidades, columnas de entrada y salida de los modelos de aprendizaje de máquinas.

- **Dataloaders**: Objetos configurables que se encargan de cargar los datos en la plataforma. Actualmente hay dataloaders habilitados para cargar datos en formato CSV, JSON y Excel, pero también existen dataloaders para cargar imágenes y audio.
- **Modelos**: Representan un modelo de aprendizaje de máquinas. Definen la lógica de entrenamiento, predicción, guardado y carga de los modelos. Actualmente, DashAI cuenta con modelos de aprendizaje de máquinas basados en los de las librerías *Hugging Face: transformers*[7] y *Scikit-Learn*[8] y su implementación es de tipo *wrapper*, es decir, envuelven los modelos de *Hugging Face* y *Scikit-Learn* para adaptarlos a la plataforma.
- **Métricas**: Representan una métrica de evaluación de un modelo de aprendizaje de máquinas. No extienden de *ConfigObject*, pero son componentes extensibles que permiten a los usuarios definir y obtener nuevas métricas de evaluación para las distintas *Tasks* de la plataforma.
- **Optimizers**: Representan un optimizador de entrenamiento de un modelo de aprendizaje de máquinas. Extienden de *ConfigObject* para permitir configurar los hiperparámetros de los optimizadores.
- **Explainer**: Representan una instancia de un método de explicabilidad. Pueden ser globales o locales, y buscan ayudar a los usuarios entender cómo los modelos de aprendizaje de máquinas resultantes de una *Run* toman decisiones. También se ejecutan de manera asíncrona.

Con la arquitectura descrita, DashAI busca ser una plataforma modular y extensible, cuyos componentes sean agnósticos a la implementación, a los datos, y permitan a los usuarios extender y personalizar la plataforma de acuerdo a sus necesidades y requerimientos.

1.3 Problema y relevancia

Debido al auge del aprendizaje de máquinas en los últimos años, los científicos de datos buscan generar el mejor modelo de aprendizaje de máquinas posible para resolver problemas específicos. Esto llevó a formalizar los pasos necesarios para entrenar un modelo de aprendizaje de máquinas, conocidos como el ciclo de vida de un modelo de aprendizaje de máquinas. Este concepto puede variar dependiendo de la metodología.

Tomando como referencia una de las metodologías más utilizadas en la industria, CRISP-DM[9], en la Figura 1[1] podemos ver que el ciclo de vida de un modelo de aprendizaje de máquinas se puede dividir en seis etapas:

1. Comprensión del negocio: Se busca comprender el problema que se quiere resolver y cómo se puede resolver con aprendizaje de máquinas.

[1]Obtenida de www.ibm.com/docs/en/spss-modeler/saas?topic=dm-crisp-help-overview

2. Comprensión de los datos: Se busca comprender los datos con los que se trabajará, identificar patrones, valores atípicos, relaciones entre las distintas variables y evaluar la calidad de los datos.

3. Preparación de los datos: Se busca preparar los datos para ser utilizados en el entrenamiento de un modelo de aprendizaje de máquinas. Esto incluye la limpieza de los datos, la transformación de los datos y la selección de las características más relevantes.

4. Modelado: Se busca entrenar un modelo de aprendizaje de máquinas con los datos preparados. Esto incluye la selección de un modelo, la configuración de los hiperparámetros del modelo y la evaluación del modelo.

5. Evaluación: Se busca evaluar el modelo entrenado con los datos de prueba. Esto incluye el cálculo de las métricas de evaluación del modelo y la interpretación de los resultados obtenidos.

6. Despliegue: Se busca desplegar el modelo entrenado en un entorno de producción. Esto incluye la integración del modelo en un sistema de producción y la monitorización del modelo en producción.

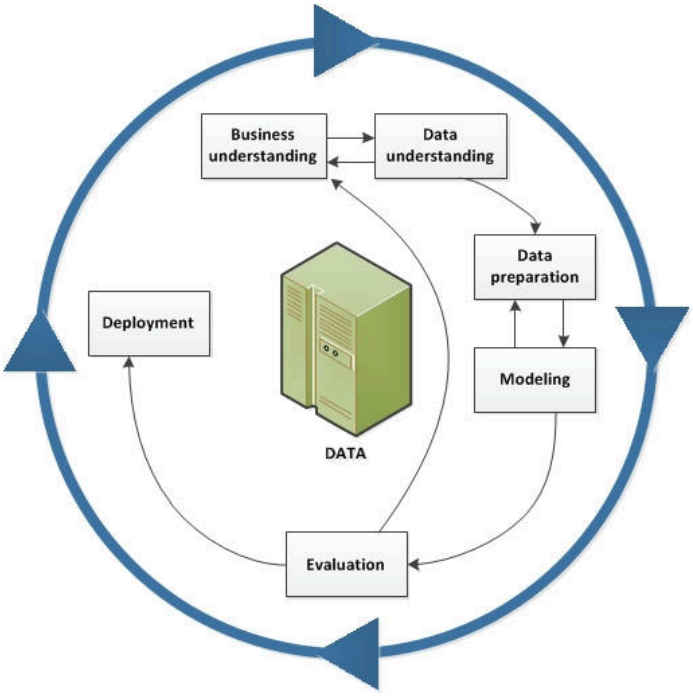

Figura 1: Diagrama del proceso CRISP-DM

Debido a que los datos son el insumo principal para entrenar los modelos de aprendizaje de máquinas, las etapas de comprensión y preparación de los datos son fundamentales para el éxito de un proyecto de aprendizaje de máquinas. Sin una comprensión y preparación adecuada de los datos, los modelos pueden ser entrenados con datos de baja calidad, lo que puede llevar a que tengan poca precisión y sean poco confiables.

Comprendiendo lo anterior, se puede ver que la etapa de comprensión de los datos es fundamental para el éxito de un proyecto de aprendizaje de máquinas, afectando directamente la calidad del modelo entrenado.

Debido a que DashAI busca ser una plataforma completa y funcional que permita a los usuarios realizar todo el proceso de aprendizaje de máquinas en un solo lugar, es necesario que tenga considerada la etapa de comprensión de los datos dentro de sus flujos y capacidades. Sin embargo, en el estado previo a la memoria, DashAI no cuenta con las funcionalidades suficientes para lograr esta etapa de manera efectiva.

1.3.1 Problemáticas en DashAI

Así, las problemáticas que se presentan en DashAI son las siguientes:

- **Falta de funcionalidades para explorar los datos cargados en la plataforma**

  Actualmente, DashAI cuenta únicamente con un resumen básico de los datos cargados. Es un paso inicial para entender los datos, pero no es suficiente para poder realmente comprender los datos con los que se está trabajando. Los usuarios necesitan poder explorar los datos de manera más detallada, identificar patrones, valores atípicos, relaciones entre las distintas variables y evaluar la calidad de los datos.

- **Falta de visualizaciones interactivas**

  Actualmente, DashAI no cuenta con visualizaciones interactivas que permitan a los usuarios explorar los datos de manera visual. Las visualizaciones interactivas son una herramienta fundamental para explorar los datos, ya que permiten a los usuarios seleccionar distintas variables, filtrar los datos, y obtener información detallada de los puntos en las visualizaciones.

Además, se presentan las siguientes oportunidades o limitantes que se deben considerar:

- **Asincronía de la plataforma**

  DashAI es una plataforma asíncrona, lo que permite a los usuarios encolar nuevas tareas mientras se ejecutan otras. La asincronía de la plataforma es una característica clave para que la experiencia de usuario sea fluida y eficiente. Por lo tanto, es necesario que el módulo de exploración y visualización de datos sea capaz de ejecutar sus procesos de manera asíncrona.

- **Extensibilidad**

La plataforma de DashAI busca ser modular y extensible, permitiendo a los usuarios extender y personalizar la plataforma de acuerdo a sus necesidades y requerimientos. Por lo tanto, es necesario que el módulo de exploración y visualización de datos sea fácil de extender y mantener.

1.4 Objetivos

Considerando las problemáticas descritas en la sección anterior, de la falta de funcionalidades para la etapa de comprensión de los datos en DashAI, se plantean los siguientes objetivos generales y específicos para este trabajo.

1.4.1 Objetivo general

El objetivo general es implementar un módulo de exploración y visualización de datos para la plataforma de DashAI, que permita a los usuarios explorar los datos cargados en la plataforma mediante estadísticas de resumen y visualizaciones interactivas. Tal módulo debe estar integrado con la interfaz de usuario de DashAI y debe ser fácil de usar e intuitivo para los usuarios.

1.4.2 Objetivos específicos

Para lograr el objetivo general, se plantearon los siguientes objetivos específicos:

1. Diseñar el «User Journey» del módulo de exploración y visualización de datos, para definir cómo el usuario interactuará con él, y qué funcionalidades se deben implementar. Esto implica también diseñar un flujo de exploración teniendo en cuenta la naturaleza asincrónica de DashAI.
2. Diseñar la interfaz del módulo de exploración y visualización de datos, siendo fácil de usar e intuitiva y que permita a los usuarios explorar los datos de manera fluida.
3. Diseñar la clase «Explorer» en el back-end de DashAI. Su diseño comprende la definición de cómo interactuará con el front-end, cómo ejecutará su lógica y cómo guardará las exploraciones resultantes.
4. Implementar la clase «Explorer» en el back-end de DashAI para obtener los datos, ejecutar la lógica de negocio, guardar los resultados y cargarlos para retornarlos al front-end.
5. Implementar las vistas y componentes en el front-end de DashAI, que permitan a los usuarios interactuar con las funcionalidades del back-end y visualizar los resultados de las exploraciones.

1.5 Metodología

Para diseñar e implementar la solución, se siguió una metodología de desarrollo de software ágil, que permite la iteración rápida y la adaptación a los cambios en los requerimientos del proyecto. La metodología ágil se basa en el desarrollo iterativo e incremental, en el que se realizan ciclos cortos de desarrollo y retroalimentación, permitiendo a los desarrolladores adaptar el software a los cambios en los requerimientos del proyecto.

1.5.1 Control de versiones

Para el control de versiones del código fuente se utilizó Git[10], un sistema de control de versiones distribuido que permite a los desarrolladores colaborar en el desarrollo de un proyecto de manera eficiente. Git permite a los desarrolladores trabajar en ramas independientes del código fuente y fusionar los cambios realizados en estas ramas de manera sencilla.

También se aplicaron *actions* de GitHub[11] para automatizar la ejecución de pruebas unitarias y de estilo de código. Esto permite a los desarrolladores verificar de manera automática que el código cumple con los estándares de calidad definidos en el proyecto y que no se introducen errores en el código fuente.

1.5.2 Reuniones periódicas

Para mantener una comunicación efectiva, las metodologías ágiles promueven la realización de reuniones periódicas entre los miembros del equipo de desarrollo. En este caso, se realizaron reuniones semanales donde cada memorista o tesista compartió sus avances, se discutieron ideas a implementar y problemas mayores que se presentaron durante el desarrollo y que se debían evaluar con todo el equipo.

1.5.3 Presentación de diseños

En las implementaciones de front-end que requieran un diseño previo, se presentan mock-ups o maquetas del diseño de las vistas a implementar. Estos diseños se discuten en las reuniones con el resto del equipo, donde se presentan recomendaciones o comentarios al diseño que aportan usabilidad o estilo y que son útiles para lograr la finalidad de la vista a implementar.

En el caso de que la nueva característica requiera código en el back-end de la aplicación también hay ocasiones que se presentan diagramas de flujo con el proceso a implementar, considerando los métodos utilizados y llamadas a la API. Esto también es comentado en las reuniones con el equipo de desarrolladores para lograr una mejor implementación.

1.5.4 Mejoramiento continuo

Por último, es importante destacar que la metodología aplicada se centra en el concepto de mejora constante tanto a nivel individual como en el desarrollo del proyecto en sí. Cada iteración y reunión planificada se consideran como puntos cruciales para perfeccionar no solo el producto final, sino las habilidades y procesos internos del equipo.

Estas instancias proporcionan oportunidades valiosas para llevar a cabo discusiones significativas y recibir comentarios constructivos. De esta manera, cada miembro del equipo puede aprender de las experiencias pasadas, identificar áreas de mejora y aplicar ajustes relevantes para elevar la calidad del proyecto y el rendimiento del equipo en su conjunto.

1.6 Solución general

La solución general contempla implementación de código en el back-end y front-end de DashAI.

1.6.1 Back-end

- **Extensión de modelos de datos**: Se extienden los modelos de datos existentes para incluir la información necesaria para soportar las exploraciones y sus resultados.
- **Implementación de la clase «Explorer»**: Se realiza una implementación abstracta de la clase «Explorer», incluyendo los métodos necesarios para interactuar con los datos, ejecutar su lógica de negocio, guardar los resultados y cargarlos para retornarlos al front-end. Estos métodos actúan como una interfaz que permite la interacción de los otros componentes de la plataforma con los exploradores, sin importar su implementación específica.
- **Implementación de la API de «Explorer»**: Se implementa una API REST que permite a los usuarios interactuar con las clases «Explorer» desde el front-end. Esta API es agnóstica a la implementación del *Explorer* específico y es compartida por todos los tipos de exploradores.
- **Implementación asincrónica**: Se implementa la clase «ExplorerJob» que permite ejecutar las exploraciones de manera asíncrona. Esta clase es agnóstica a la implementación del *Explorer* específico y es compartida por todos los tipos de exploradores.

Con lo anterior, se busca resolver en el back-end la lógica de negocio necesaria para realizar las exploraciones de datos de manera asíncrona y otorgar extensibilidad a la implementación de nuevos exploradores.

1.6.2 Front-end

- **Diseño de la interfaz**: Se diseña la interfaz del módulo de exploración y visualización de datos, definiendo el flujo de interacción del usuario y las vistas necesarias para realizar las exploraciones de datos. Este diseño debe seguir la línea de diseño de DashAI y ser fácil de usar e intuitivo para los usuarios.
- **Implementación de las vistas administrativas**: Se implementan las vistas necesarias para administrar las exploraciones de datos, como la creación, edición, ejecución y eliminación de exploraciones. Estas vistas deben ser cómodas y permitir a los usuarios gestionar sus exploraciones de manera simple, eficiente e intuitiva.
- **Implementación de vistas de resultados**: Se implementan las vistas necesarias para visualizar los resultados de las exploraciones de datos. Son dependientes del tipo de resultado que se obtenga de la exploración especifica, por lo que inicialmente se definieron resultados de tipo tabla, gráfico e imagen, con sus formatos específicos y con posibilidad de extensiones futuras para otros tipos de resultados.

Con lo anterior, se busca resolver en el front-end la interacción del usuario con el módulo de exploración y visualización de datos, permitiendo a los usuarios explorar los datos de manera sencilla y rápida.

1.6.3 Implementación de nuevos exploradores

Una vez finalizadas las implementaciones de back-end y front-end, se implementan los nuevos exploradores específicos que, por ejemplo, permiten mostrar filas y columnas de los datos, estadísticas de resumen, histogramas, gráficos de dispersión, entre otros. Estos exploradores son los primeros que tendrá la plataforma y actúan como ejemplos de implementación de futuros exploradores.

1.7 Estructura del documento

El presente trabajo se estructura de la siguiente manera:

- En el Capítulo 1 se presenta el contexto en el que se desarrolla DashAI, se describen las funcionalidades actuales de la plataforma y se plantean los problemas y objetivos que se buscan abordar en este trabajo junto con la metodología de desarrollo seguida. Por último, se presenta una solución general de cómo se abordarán los problemas planteados.
- En el Capítulo 2 se presenta el estado del arte, donde se revisan las herramientas y técnicas existentes para la exploración y visualización de datos, así como los conceptos teóricos necesarios para comprender el trabajo realizado.
- En el Capítulo 3 se presenta la solución implementada por el memorista, detallando y justificando los métodos y herramientas utilizadas en la implementación.

- En el Capítulo 4 se presentan los resultados obtenidos, mostrando las funcionalidades implementadas y ejemplos de uso del módulo de exploración y visualización de datos.
- Finalmente, en el Capítulo 5 se presentan las conclusiones del trabajo, destacando los principales hallazgos, aprendizajes y desafíos pendientes.

Capítulo 2

Estado del Arte

2.1 Soluciones existentes

Existen varios softwares y herramientas similares que ofrecen funcionalidades para el aprendizaje de máquinas, la inspiración de DashAI es Weka (Waikato Environment for Knowledge Analysis)[12], una plataforma de código abierto portable desarrollada en Java que permite a sus usuarios aplicar técnicas de minería de datos mediante una interfaz gráfica. Weka ofrece una amplia variedad de algoritmos de aprendizaje de máquinas, herramientas de preprocesamiento de datos, y de exploración y visualizaciones, pero su última versión estable fue lanzada a inicios de 2022. Otras opciones similares más recientes son RapidMiner[13] y KNIME (Konstanz Information Miner)[14], que al igual que Weka, están desarrolladas en Java.

El problema con estas herramientas es que Python se ha vuelto el lenguaje de programación más popular en el área de aprendizaje de máquinas[15] y como tal, la mayoría de las librerías de aprendizaje de máquinas están desarrolladas en este lenguaje, lo que dificulta la integración de estas a las herramientas mencionadas anteriormente.

En el ámbito de la visualización de datos, existen varias librerías y herramientas que permiten a los usuarios visualizar y explorar los datos de manera interactiva. Algunas de las librerías más populares son Matplotlib[16], Seaborn[17] y Plotly[18]. Estas librerías permiten a los usuarios crear visualizaciones estáticas y dinámicas de los datos, pero requieren de conocimientos en programación para poder ser utilizadas. Por otro lado, existen herramientas como Tableau[19] y PowerBI[20] que permiten a los usuarios crear visualizaciones de datos de manera interactiva sin la necesidad de programar, pero son herramientas de pago y no ofrecen la flexibilidad que ofrecen las librerías mencionadas anteriormente.

2.1.1 KNIME

KNIME es una plataforma de código abierto que permite a los usuarios crear flujos de trabajo de análisis de datos mediante una interfaz gráfica. Esta plataforma ofrece una amplia variedad de nodos que permiten a los usuarios realizar diversas tareas como la limpieza, transformación, exploración y visualización de datos. KNIME se encuentra desarrollado en Java, pero su sistema de nodos permite a los usuarios integrar nodos de Python en sus flujos de trabajo, requiriendo conocimientos en programación para poder ser utilizados. KNIME es una herramienta muy completa y con una gran cantidad de funcionalidades, extensibles por su comunidad mediante los nodos, pero su enfoque es más orientado al análisis de datos que al aprendizaje de máquinas.

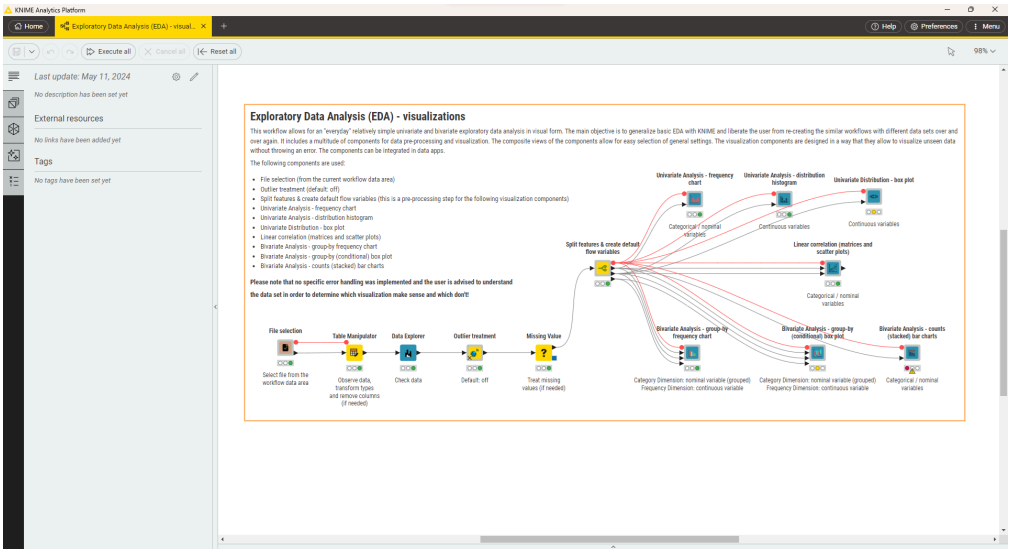

Figura 2: Interfaz de KNIME: Flujo de exploración

2.1.2 Weka

Weka es un software gratuito desarrollado en la Universidad de Waikato en el lenguaje Java. Es una de las herramientas más utilizadas en el área de aprendizaje de máquinas y minería de datos. Weka ofrece una amplia variedad de algoritmos de aprendizaje de máquinas, herramientas de preprocesamiento de datos y visualizaciones de manera modular, como se muestra en la Figura 3.

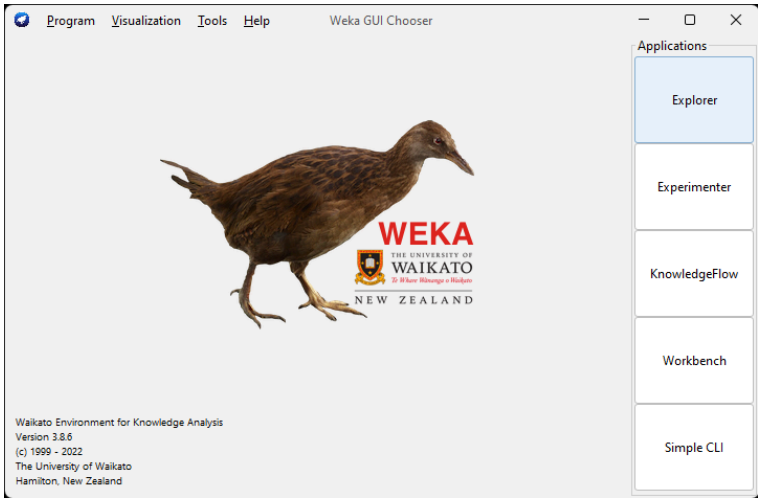

Figura 3: Selector de interfaz de Weka

Para el caso específico de esta memoria lo más relevante de Weka es su capacidad de exploración de datos en el módulo "Explorer", el cual permite a los usuarios indagar en los datos cargados en la plataforma mediante estadísticas de resumen y visualizaciones de datos.

En la Figura 4 y Figura 5 se muestra la vista de preprocesamiento con el dataset "iris" cargado. En esta vista se puede seleccionar un atributo para ver sus estadísticas de resumen (distintas para atributos numéricos y nominales) y la distribución de los valores.

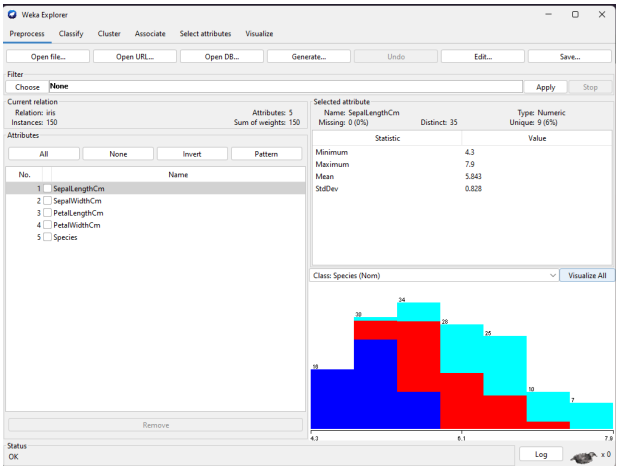

Figura 4: Inspección de atributo numérico en Weka

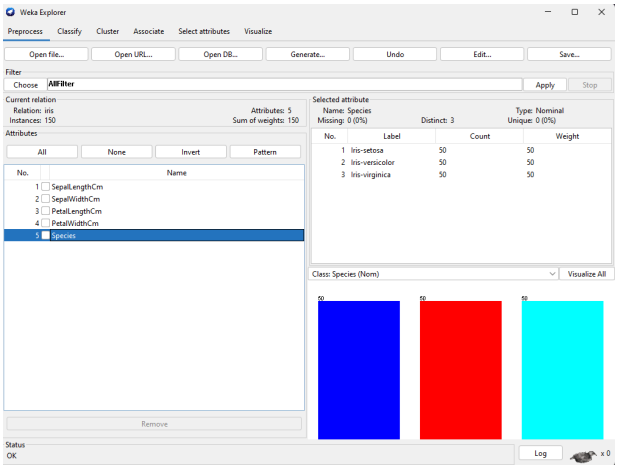

Figura 5: Inspección de atributo nominal en Weka

Por otro lado, la vista de visualización mostrada en la Figura 6 permite a los usuarios explorar los datos mediante visualizaciones matriciales entre sus atributos. Esta técnica llamada "Plot Matrix" permite a los usuarios identificar relaciones entre los dos distintos atributos y detectar valores atípicos.

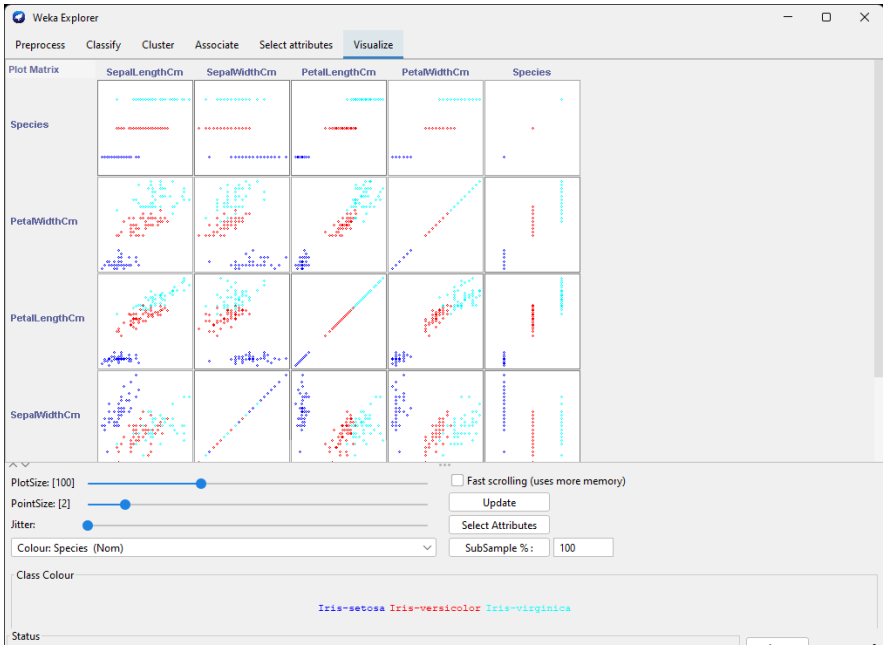

Figura 6: Vista de visualización de Weka

Weka es una herramienta muy completa y con una gran cantidad de funcionalidades, pero como fue mencionado anteriormente, a día de hoy el uso de Python es mucho más popular en el área de aprendizaje de máquinas, lo que dificulta la integración de las librerías de Python a Weka. Aun así, Weka actúa como una referencia e inspiración para el desarrollo de DashAI.

2.2 Herramientas y técnicas

En esta sección se habla de las herramientas y técnicas utilizadas comúnmente para resolver los problemas planteados. Se mencionan las librerías de Python más utilizadas en el área de aprendizaje de máquinas y visualización de datos, así como las técnicas de visualización de datos más comunes.

2.2.1 Técnicas de exploración de datos

La exploración de datos es una etapa fundamental en el proceso de aprendizaje de máquinas, ya que permite a los científicos de datos comprender los datos con los que están trabajando, identificar patrones, valores atípicos, relaciones entre las distintas variables y evaluar la calidad de los datos. Algunas de las técnicas más comunes[21] utilizadas en la exploración de datos son:

- **Estadísticas de resumen**: Las estadísticas de resumen permiten a los científicos de datos obtener una visión general de los datos con los que están trabajando. Algunas

de las estadísticas de resumen más comunes son la media, la mediana, la desviación estándar, el mínimo, el máximo y los cuartiles de los datos.

- **Visualizaciones de datos**: Las visualizaciones de datos permiten a los científicos de datos explorar los datos de manera visual. Las visualizaciones de datos pueden ser estáticas o interactivas, y permiten a los usuarios identificar patrones, valores atípicos y relaciones entre las distintas variables.
- **Análisis de correlación**: El análisis de correlación permite a los científicos de datos identificar relaciones entre las distintas variables de los datos. La correlación puede ser positiva, negativa o nula, y permite a los usuarios identificar qué variables están relacionadas entre sí.
- **Análisis de componentes principales**: El análisis de componentes principales (PCA) es una técnica de reducción de dimensionalidad que permite a los científicos de datos reducir la cantidad de variables de los datos. PCA permite a los usuarios identificar las componentes más importantes de los datos y visualizarlos en un espacio de menor dimensión.
- **Análisis de clusters**: El análisis de clusters permite a los científicos de datos agrupar los datos en clusters o grupos de datos similares. El análisis de clusters permite a los usuarios identificar patrones en los datos y agruparlos en categorías.

En el contexto de DashAI, se busca implementar las técnicas de estadísticas de resumen y visualizaciones de datos para permitir a los usuarios explorar los datos cargados en la plataforma.

2.2.2 Librerías comúnmente utilizadas

En el área de aprendizaje de máquinas y visualización de datos, existen varias librerías de Python que son ampliamente utilizadas por los científicos de datos. Algunas de las librerías más populares son:

- **Pandas**[22]: Pandas es una librería de Python que permite a los científicos de datos manipular y analizar datos de manera sencilla. Pandas ofrece una gran cantidad de funciones y métodos para cargar, guardar, limpiar, transformar y analizar datos.
- **Numpy**[23]: Numpy es una librería de Python que permite realizar operaciones matemáticas y estadísticas de manera eficiente. Numpy ofrece una gran cantidad de funciones y métodos para realizar operaciones matemáticas y estadísticas en matrices y vectores.
- **Matplotlib**[16]: Matplotlib es una librería de Python que permite crear visualizaciones de datos estáticas. Matplotlib ofrece una gran cantidad de funciones y métodos para crear gráficos de barras, gráficos de líneas, gráficos de dispersión, histogramas, entre otros.

- **Seaborn**[17]: Seaborn es una librería de Python que permite crear visualizaciones de datos más avanzadas. Seaborn ofrece una gran cantidad de funciones y métodos para crear gráficos de barras, gráficos de líneas, gráficos de dispersión, histogramas, entre otros, con un estilo más atractivo y moderno con respecto a *Matplotlib*, en el que se basa.
- **Plotly**[18]: Plotly es una librería de Python que permite crear visualizaciones de datos interactivas. Plotly ofrece una gran cantidad de funciones y métodos para crear gráficos de barras, gráficos de líneas, gráficos de dispersión, histogramas, entre otros, que pueden ser interactivos y personalizables.

Capítulo 3

Solución

En este capítulo se presenta la solución implementada por el memorista para abordar los objetivos planteados en la Sección 1.4. Inicialmente, se presenta el diseño de la solución, detallando diagramas de flujo, diagramas de clases y diagramas de secuencia que permiten comprender la arquitectura y la lógica de la solución. Luego se presentan las herramientas y tecnologías utilizadas en el desarrollo de la solución, detallando el uso de las librerías ya existentes en DashAI y las nuevas librerías utilizadas en el desarrollo del módulo de exploración y visualización de datos.

Finalmente, se presenta la implementación de la solución, detallando los cambios realizados en el back-end y front-end de DashAI para integrar el módulo de exploración y visualización de datos. Se describen los nuevos modelos de datos, las nuevas clases y métodos implementados, y las nuevas vistas y componentes creados en el front-end de la plataforma.

3.1 Diseño de la solución

En esta sección se presenta el diseño de la solución, detallando los diagramas de flujo y diagramas de clases que permiten comprender la arquitectura y la lógica de la solución implementada.

3.1.1 Flujo de usuario

Inicialmente, se presentó un diagrama que consideraba que las instancias de exploradores son independientes entre sí, como se puede observar en la Figura 7. En este diagrama, se muestra el flujo de usuario para crear un explorador, ejecutarlo y visualizar los resultados obtenidos.

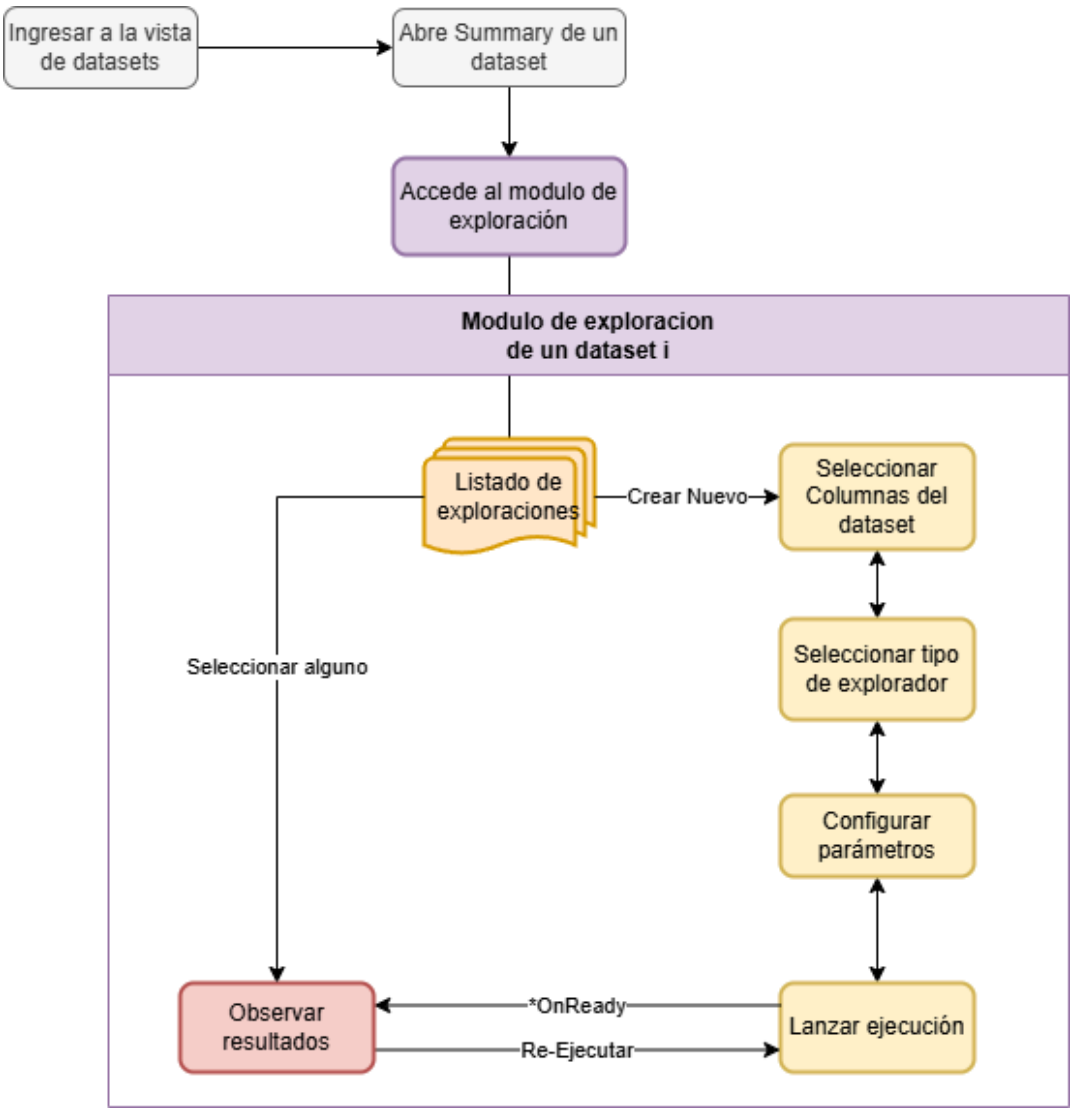

Figura 7: Flujo de usuario para los exploradores

Sin embargo, durante las reuniones semanales con el equipo de desarrollo, se identificó que esta implementación no era la más cómoda para los usuarios, ya que no permitía crear exploradores de manera sencilla y rápida. Para solventar esto, se definió conceptualmente las «Sesiones de exploración» o «Exploraciones». El objetivo de estas es agrupar las instancias de exploradores, permitiendo a los usuarios crear, ejecutar y visualizar los resultados de varios exploradores en conjunto, como se muestra en el Figura 8. Además, se decidió que las configuraciones iniciales de columnas y parámetros de los exploradores se aplicarían automáticamente al agregarlos a la exploración, facilitando y agilizando inmensamente la creación de estas instancias.

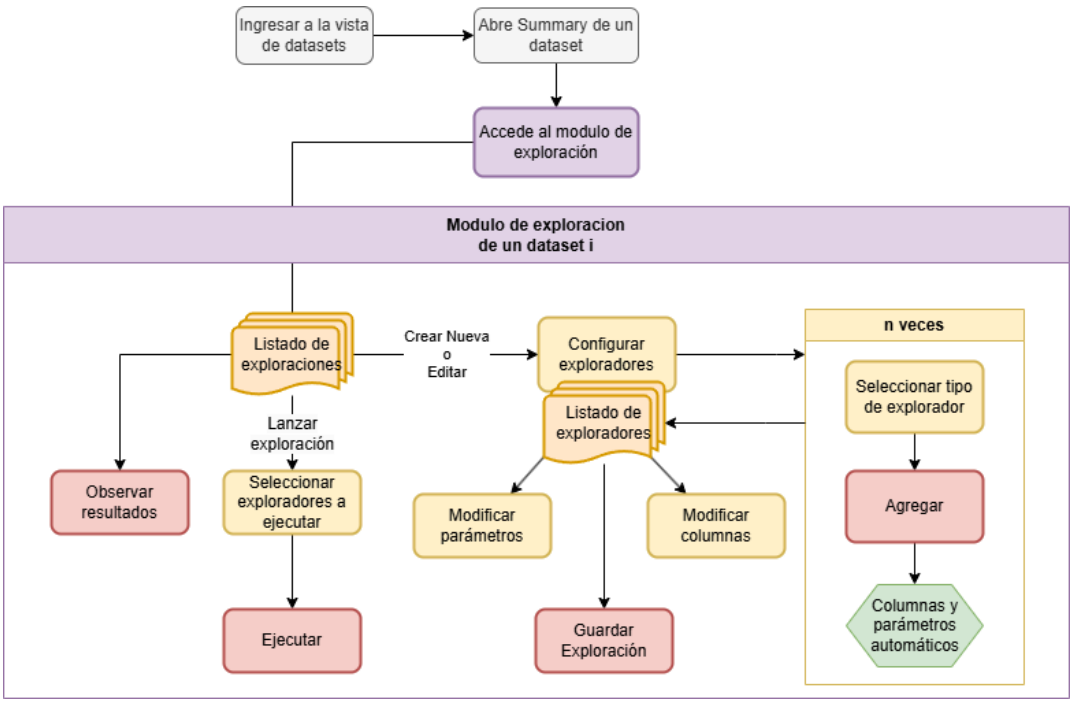

Figura 8: Flujo de usuario para las exploraciones

3.1.2 Flujo de sistema

Habiendo definido el flujo de usuario, se presenta el diagrama de flujo de sistema que muestra cómo interactúa el front-end y el back-end de DashAI para crear, ejecutar y visualizar los resultados de las exploraciones. En la Figura 9 se muestra el flujo de sistema para crear una exploración y sus exploradores (sin configurarlos, pues eso alargaría mucho el diagrama de flujo).

El flujo comienza cuando un usuario usa el botón de crear exploración, luego de escoger un nombre y descripción, pasa al paso 2: «Configurar exploradores». Al llegar al paso 2, el front-end inicia una consulta para obtener los componentes de tipo *Explorer* disponibles en el back-end. Una vez obtenidos, el usuario puede seleccionar uno.

Al seleccionar algún tipo de explorador, el front-end envía una petición al back-end para obtener los parámetros iniciales del tipo de explorador seleccionado. Al agregar ese tipo de explorador a la exploración, se agrega con los parámetros iniciales obtenidos y se toman todas las columnas válidas.

Lo anterior se repite hasta que el usuario agrega todos los exploradores que desea a la exploración. Al finalizar, el usuario guarda la exploración. Al guardarla, el front-end envía una petición al back-end para guardar la exploración y luego de que la exploración está creada, se envían paralelamente las peticiones para crear cada explorador en la exploración.

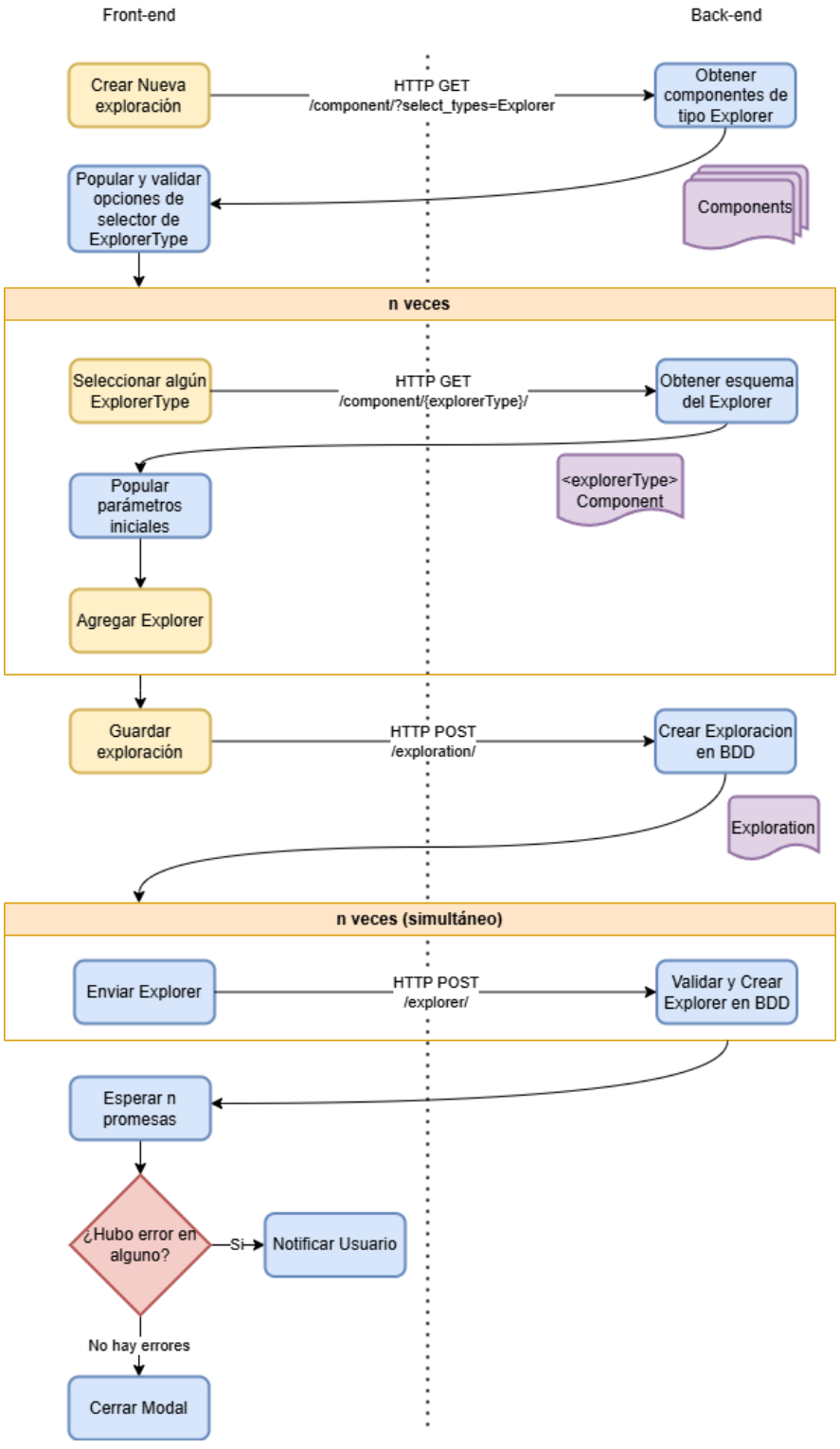

Figura 9: Flujo de sistema para crear exploraciones

Correr las exploraciones es un proceso bastante más simple para el usuario, como se muestra en la Figura 10. El usuario usa el botón «Play» para abrir el modal de ejecución de la exploración, donde puede escoger los exploradores que se ejecutarán. Al confirmar, el front-end envía simultáneamente peticiones al back-end para encolar cada explorador. El

back-end crea una instancia de *ExplorerJob* para cada uno. Luego de encolar todos los exploradores, el front-end cada cierto tiempo refresca el estado de los exploradores mediante una petición hasta que todos terminan.

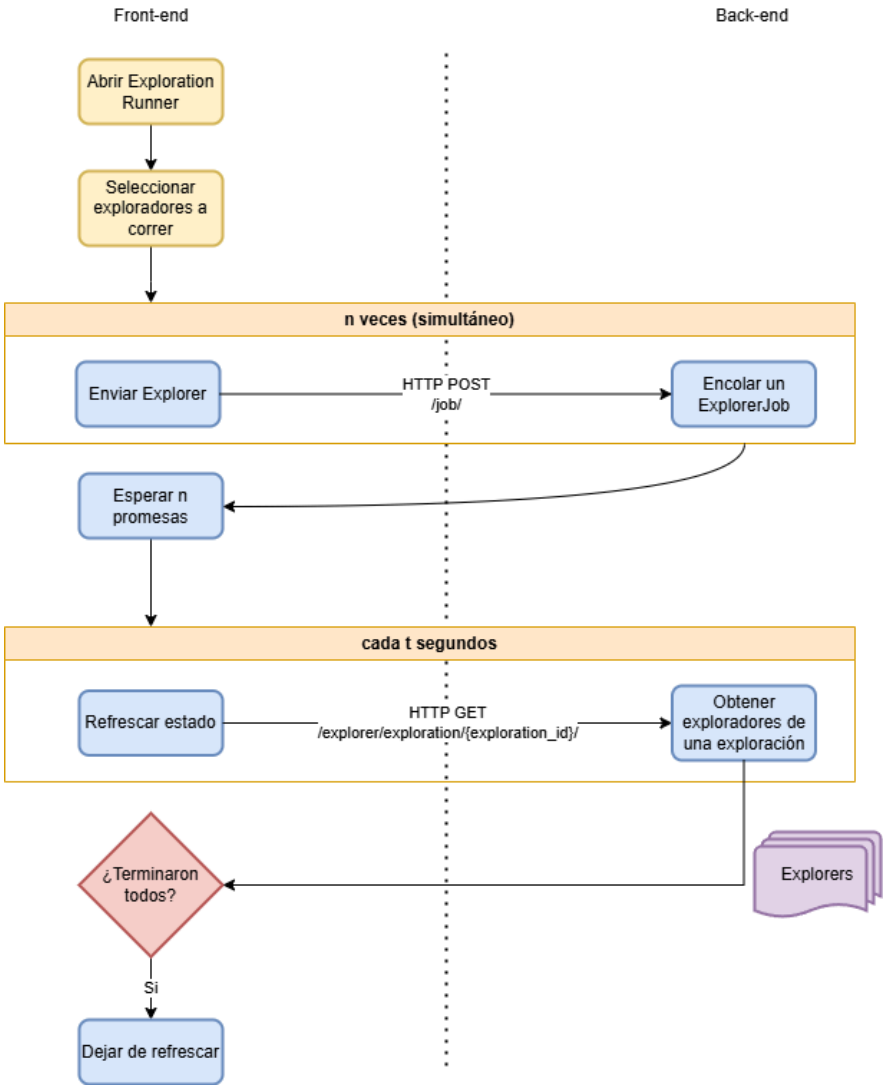

Figura 10: Flujo de sistema para correr exploraciones

Finalmente, para visualizar los resultados de las exploraciones, como se muestra en la Figura 11, el usuario usa el botón de visualizar resultados. Al hacerlo, el front-end envía simultáneamente una petición por cada explorer al back-end, para obtener los resultados de cada uno. El back-end obtiene los resultados del explorer y los envía al front-end para ser visualizados.

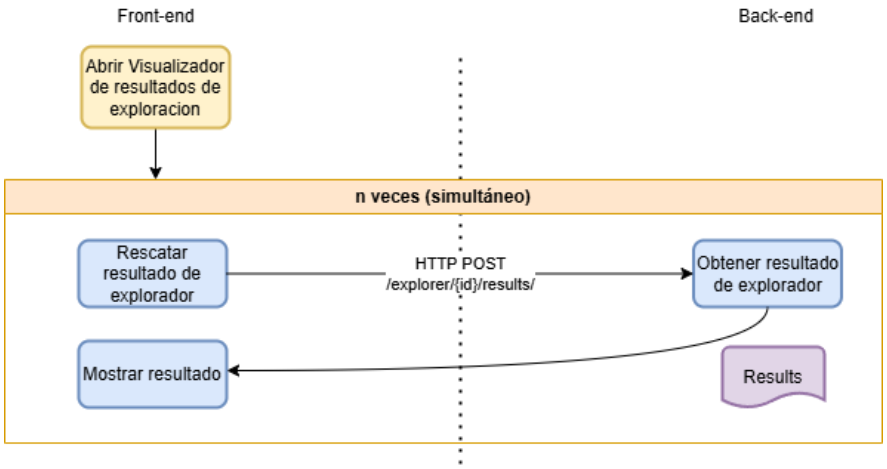

Figura 11: Flujo de sistema para visualizar resultados de exploraciones

3.1.3 Diagrama de clases

Para la implementación de la solución, se diseñó un diagrama de clases que muestra las clases y métodos necesarios en el back-end de DashAI para implementar el módulo de exploración y visualización de datos. En el diagrama de clases se muestran las clases *Exploration*, *Explorer*, *ExplorerJob* y *BaseExplorer*, que son las clases principales de la solución.

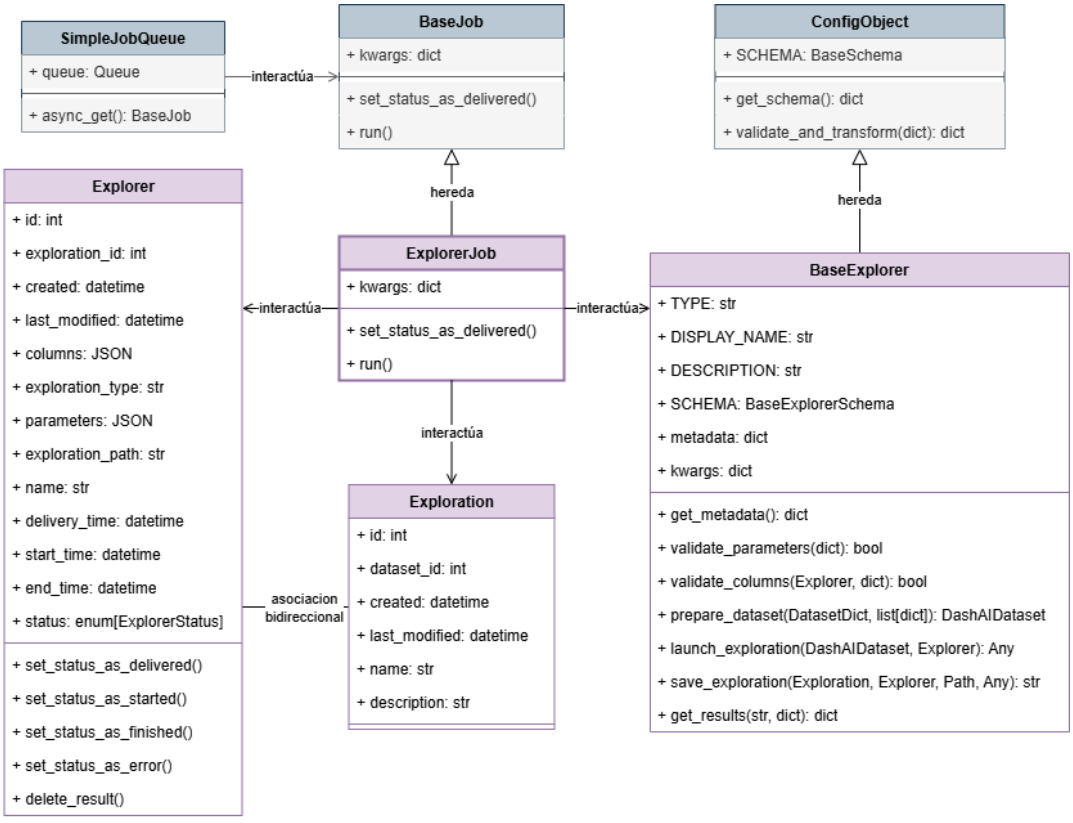

Figura 12: Diagrama de clases del módulo de exploración y visualización de datos

Como se puede observar en la Figura 12, las clases *Exploration* y *Explorer* contienen y guardan la información de las exploraciones y exploradores creados por los usuarios. Por lo tanto, estas clases están definidas como objetos relacionales mapeados (ORM) en la base de datos de DashAI. La clase *ExplorerJob* es una clase auxiliar que permite ejecutar los métodos de *BaseExplorer* de manera asíncrona. Esta última es una clase abstracta que define los métodos necesarios para aplicar la lógica de negocio de los exploradores.

3.2 Herramientas utilizadas

En esta sección se describen las herramientas y tecnologías utilizadas en el desarrollo de la solución. Se detallan las librerías de Python y JavaScript utilizadas en el back-end y front-end de DashAI, así como las tecnologías y técnicas utilizadas en el desarrollo de la plataforma. Cabe destacar, que las herramientas y tecnologías utilizadas en DashAI son muchas y variadas, por ello se detallan solo las más relevantes para el contexto de esta memoria.

3.2.1 Tecnologías utilizadas en DashAI

Back-end

- **Python**

  Python[3] es un lenguaje de programación interpretado y de alto nivel, que se ha vuelto muy popular debido a su simplicidad y facilidad de uso. Python cuenta con una gran cantidad de librerías y frameworks para el aprendizaje de máquinas, como Scikit-Learn, TensorFlow, PyTorch, entre otros. DashAI está desarrollado en Python, lo que permite a los usuarios utilizar las librerías de aprendizaje de máquinas más populares en la plataforma.

- **FastAPI y Pydantic**

  FastAPI[24] es un framework web de Python que permite a los desarrolladores crear APIs REST de manera rápida y eficiente. FastAPI es utilizado en DashAI para crear las APIs REST que permiten a los usuarios interactuar con la plataforma.

  Pydantic[6] es una librería de Python que permite a los desarrolladores definir esquemas de datos y validar los datos de entrada de manera sencilla y eficiente. Pydantic es utilizada en DashAI para definir los esquemas de los datos que se envían y reciben en las APIs REST de la plataforma.

- **SQLAlchemy**

SQLAlchemy[25] es una librería de Python que permite a los desarrolladores hacer mapas de objetos relacionales (ORM) en bases de datos relacionales. Esto simplifica la gestión e interacción con la base de datos de DashAI.

- **Asyncio**

  Asyncio[26] es una librería de Python que permite a los desarrolladores escribir código asíncrono. Asyncio es extendida en DashAI para definir la *JobQueue* que permite ejecutar *Jobs* de manera asíncrona en la plataforma.

- **Pillow**

  Pillow[27] es una librería de Python que permite a los desarrolladores manipular imágenes. Con Pillow, los desarrolladores pueden cargar, guardar, editar y transformar imágenes. Esto es utilizado en DashAI para guardar y cargar imágenes en la plataforma.

- **Hugging Face Datasets**

  Hugging Face Datasets[28] es una librería de Python centrada en la gestión de conjuntos de datos para el aprendizaje de máquinas. Datasets permite a los desarrolladores cargar, procesar y compartir conjuntos de datos de manera sencilla y eficiente. Datasets es extendida en DashAI para cargar y almacenar los datasets.

- **Numpy y Pandas**

  También se utiliza numpy[23] y pandas[22], dos librerías de Python muy populares en el área de aprendizaje de máquinas. Numpy es utilizada para realizar operaciones matemáticas y estadísticas de manera eficiente, mientras que pandas es utilizada para manipular y analizar datos de manera sencilla.

- **Plotly**

  Plotly[29] es una librería de Python que permite a los desarrolladores crear visualizaciones de datos interactivas en el back-end. Estas son enviadas al front-end para ser mostradas al usuario.

Front-end

- **JavaScript, TypeScript y React**

  JavaScript[30] es un lenguaje de programación interpretado que se utiliza principalmente en el desarrollo de aplicaciones web. JavaScript es uno de los lenguajes de programación más populares en la actualidad.

TypeScript[31] es un superconjunto de JavaScript que agrega tipado estático al lenguaje. TypeScript generalmente es utilizado para mejorar la calidad del código y reducir los errores en el desarrollo. En DashAI se utiliza para restringir los tipos de datos que se pueden enviar y recibir en las APIs REST de la plataforma. Su uso es similar al de pydantic pero en el front-end, con ambos, se establece un contrato entre el back-end y el front-end.

React[4] es una librería de JavaScript desarrollada por Facebook que permite a los desarrolladores crear interfaces de usuario interactivas y dinámicas. React es una de las librerías más populares para el desarrollo de interfaces de usuario en la web, y es utilizada en DashAI para el desarrollo del front-end de la plataforma.

- **Mui Material**

Material-UI[32] es una librería de componentes de React que implementa el diseño de Material Design de Google. Material-UI permite a los desarrolladores crear interfaces de usuario atractivas rápidamente. Ofrece una gran cantidad de componentes predefinidos, iconos, temas, estilos y otros elementos que facilitan el desarrollo de interfaces de usuario.

- **Formik**

Formik[33] es una librería de React que permite a los desarrolladores crear formularios. Formik se encarga de manejar el estado de los formularios, la validación de los datos y la interacción con los usuarios. Se usa para automatizar la creación de formularios de los *Objetos Configurables* en DashAI.

- **Plotly**

Plotly[34] es una librería de JavaScript que permite a los desarrolladores crear visualizaciones de datos interactivas. Dado que también se utiliza en el back-end, Plotly permite a los desarrolladores crear visualizaciones de datos de manera consistente en todo el proyecto.

Herramientas de desarrollo

- **Linters y formateadores de código**

Linters y formateadores de código son herramientas que permiten a los desarrolladores mantener un código limpio y consistente. Los linters revisan el código en busca de errores y malas prácticas, mientras que los formateadores de código aplican un estilo de código consistente en todo el proyecto.

En backend se utilizan Flake8[35] y Black[36], mientras que en el front-end se utilizan ESlint[37] y Prettier[38].

- **Git y GitHub**

  Git[10] es un sistema de control de versiones distribuido que permite a los desarrolladores colaborar en el desarrollo de un proyecto de manera eficiente. Git permite a los desarrolladores trabajar en ramas independientes del código fuente y fusionar los cambios realizados en estas ramas de manera sencilla.

  GitHub[39] es una plataforma de desarrollo colaborativo que permite a los desarrolladores alojar, revisar y colaborar en proyectos de software. GitHub ofrece una gran cantidad de herramientas y funcionalidades que facilitan la colaboración en el desarrollo de software.

- **Pre-commit**

  Pre-commit[40] es una herramienta que permite a los desarrolladores ejecutar scripts antes de realizar un commit en un repositorio de Git. Pre-commit ayuda a verificar que el código cumple con los estándares de calidad definidos en el proyecto antes de ser enviado al repositorio.

  En este caso, se utiliza pre-commit para formatear y arreglar el código de back-end con Ruff, y el de front-end con Prettier y ESLint.

- **GitHub Actions**

  GitHub Actions[11] es una herramienta de GitHub que permite a los desarrolladores automatizar acciones en un proyecto. GitHub Actions permite a los desarrolladores ejecutar pruebas unitarias, pruebas de estilo de código, despliegues automáticos y otras tareas de manera automática. En este caso, se utiliza GitHub Actions para, luego de subir un cambio al repositorio, ejecutar el pre-commit, y luego construir y probar la aplicación en variados entornos.

- **Figma**

  Figma[41] es una herramienta de diseño de interfaces de usuario que permite a los desarrolladores crear maquetas y prototipos de interfaces de usuario de manera sencilla y eficiente. Figma ofrece una gran cantidad de herramientas y funcionalidades que facilitan el diseño de interfaces de usuario.

  En este caso, se utiliza Figma para diseñar las interfaces de usuario de DashAI, y para compartir los diseños con el resto del equipo.

3.2.2 Tecnología añadida en esta memoria

Todas las herramientas mencionadas anteriormente en la Sección 3.2.1 ya eran parte de DashAI y son también utilizadas en esta memoria. Sin embargo, para la solución en este trabajo se añade la siguiente librería:

- **Wordcloud**

  Wordcloud[42] es una librería de Python que permite a los desarrolladores crear nubes de palabras a partir de un texto. Esta es una herramienta útil para visualizar la frecuencia de las palabras en un texto y detectar patrones y tendencias. Es la librería más popular para la creación de nubes de palabras en Python. Su ventaja sobre otras librerías es su algoritmo simple y eficiente, y sus dependencias ya existentes en DashAI.

  La inclusión de esta librería aborda un caso de exploración de datos específico para cadenas de texto, permitiendo a los usuarios visualizar la frecuencia de las palabras en un texto y detectar patrones y tendencias.

3.3 Implementación

En esta sección se presenta la implementación de la solución, detallando los cambios realizados en el back-end y front-end de DashAI para integrar el módulo de exploración y visualización de datos. Se describen los nuevos modelos de datos, las nuevas clases y métodos implementados, y las nuevas vistas y componentes creados en el front-end de la plataforma.

3.3.1 Modelos de datos

Para soportar las exploraciones y sus resultados, no se modificaron entidades existentes, pero se agregaron nuevas entidades que permiten almacenar la información necesaria para soportar las exploraciones y exploradores. Se crearon dos nuevas entidades: «Exploration» y «Explorer».

Como se ve en la Figura 13[2], la entidad «Exploration» representa una sesión de exploración de datos en DashAI. Cada exploration tiene un dataset asociado, nombre y descripción. Por otro lado, la entidad «Explorer» representa una instancia de explorador en DashAI. Cada explorer tiene una exploración asociada, un nombre, un tipo de explorador, las columnas escogidas, parámetros, la ubicación de los resultados y el estado de la exploración. Los atributos *delivery_time*, *start_time*, *end_time* y *status* son utilizados para el manejo de la ejecución asíncrona de las exploraciones.

[2]Se omitieron atributos de metadata como *created_at* y *last_modified* del diagrama

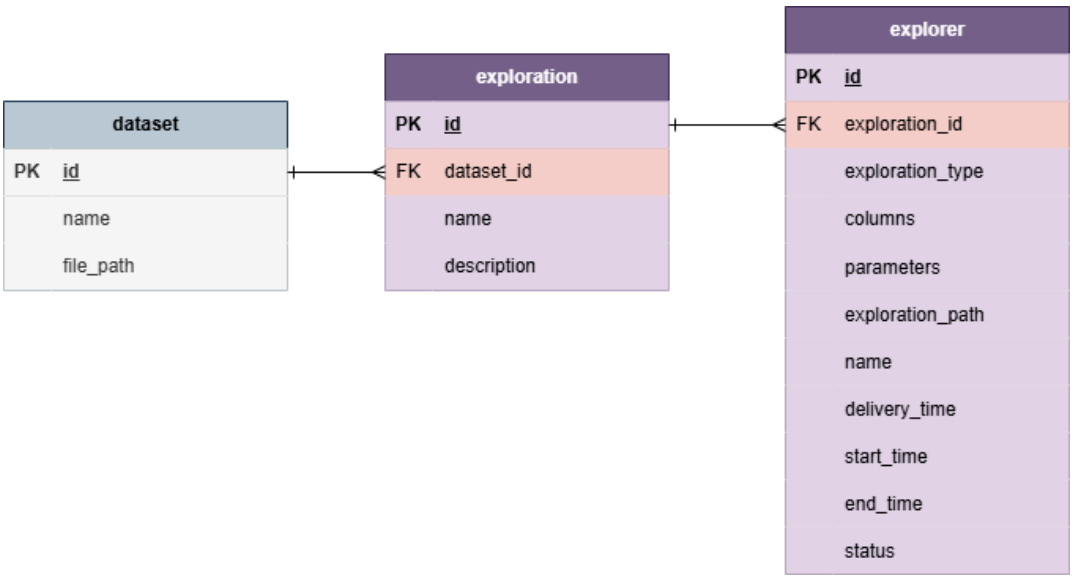

Figura 13: Diagrama relacional de las entidades Exploration y Explorer

El detalle de los atributos se verá con más detalle en la Sección 3.3.2, donde se revisará la implementación de las clases y métodos en el back-end de DashAI.

3.3.2 Clases y métodos

En esta sección se presenta la implementación de las clases y métodos en el back-end de DashAI para soportar las exploraciones y exploradores.

Como fue mencionado anteriormente en la Sección 3.1.3, se crearon las clases *Exploration*, *Explorer*, *ExplorerJob* y *BaseExplorer*.

- **Exploration**

  La clase *Exploration* es una clase ORM que representa una sesión de exploración de datos en DashAI. La clase *Exploration* tiene los atributos:

  - `id`: Identificador de la exploración.
  - `dataset_id`: Identificador del dataset asociado.
  - `name`: Nombre de la exploración.
  - `description`: Descripción de la exploración.
  - `created`: Fecha de creación de la exploración.
  - `last_modified`: Fecha de última modificación de la exploración.

- **Explorer**

  La clase *Explorer* es una clase ORM que representa una instancia de explorador en DashAI. La clase *Explorer* tiene los atributos:

- `id`: Identificador del explorador.
- `exploration_id`: Identificador de la exploración asociada.
- `created`: Fecha de creación del explorador.
- `last_modified`: Fecha de última modificación del explorador.
- `name`: Nombre del explorador.
- `exploration_type`: Tipo de explorador.
- `columns`: Columnas seleccionadas para la exploración con su tipo y orden.
- `parameters`: Parámetros de la exploración.
- `results_path`: Ubicación de los resultados de la exploración.
- `status`: Estado de la exploración.
- `delivery_time`: Tiempo de entrega de los resultados.
- `start_time`: Tiempo de inicio de la exploración.
- `end_time`: Tiempo de fin de la exploración.

La clase *Explorer* está asociada a una exploración mediante una relación muchos a uno. Esto permite a los usuarios crear varias instancias de exploradores en una exploración.

Dado que esta clase almacena el estado de la exploración, se crearon métodos para guardar y actualizar el estado de la exploración, estos métodos son:

- `set_status_as_delivered()`: Cambia el estado de la exploración a «delivered».
- `set_status_as_started()`: Cambia el estado de la exploración a «started».
- `set_status_as_finished()`: Cambia el estado de la exploración a «finished».
- `set_status_as_error()`: Cambia el estado de la exploración a «error».

Por último, para manejar la eliminación de exploradores, se creó el método `delete_result()` que elimina los resultados de la exploración del sistema de archivos.

- **ExplorerJob**

  La clase *ExplorerJob* es una clase auxiliar que permite ejecutar los métodos de *BaseExplorer* de manera asíncrona y actualizar el estado de los exploradores interactuando con la clase *Explorer*. La clase *ExplorerJob* hereda de la clase *BaseJob* de DashAI, que permite ejecutar tareas de manera asíncrona en la plataforma mediante una cola de trabajos.

  La clase *ExplorerJob* tiene un solo atributo `kwargs` que almacena los argumentos necesarios para la ejecución. Entre estos, se utilizan `explorer_id` para identificar el explorador a ejecutar y `db`, que es la sesión de la base de datos con la que se interactúa.

  Por otro lado, la clase debe implementar el método `set_status_as_delivered()` para actualizar el estado del explorador a «delivered» cuando el trabajo ingresa a la cola de trabajos y el método `run()`, al que se le inyectan como dependencia los argumentos:

- `component_registry`: Registro de componentes de DashAI.
- `config`: Configuración de DashAI.

El método `run()` es la parte más importante de la clase, ya que es el que:

1. Obtiene la información del *Explorer*.
2. Cambia el estado del explorador a «started».
3. Obtiene la información de la exploración asociada.
4. Obtiene la información del dataset asociado.
5. Carga el dataset.
6. Busca la clase *BaseExplorer* correspondiente al tipo de explorador.
7. Instancia la clase anterior con los parámetros del explorador.
8. Ejecuta la lógica de negocio del objeto *BaseExplorer*. Esto implica:
   1. Preparar los datos mediante el método `prepare_dataset`.
   2. Ejecutar la exploración mediante el método `launch_exploration`.
   3. Guardar los resultados mediante el método `save_exploration`.
9. Cambia el estado del explorador a «finished» y actualiza la localización de los resultados en el explorador.

Todo lo anterior se realiza de manera asíncrona y se manejan los errores que puedan surgir durante la ejecución, cambiando el estado del explorador a «error» en caso de que ocurra un error.

- **BaseExplorer y BaseExplorerSchema**

  La clase *BaseExplorer* es una clase abstracta que define los métodos necesarios para aplicar la lógica de negocio de los exploradores. Hereda de la clase *ConfigObject* de DashAI, que permite a los desarrolladores definir objetos configurables en la plataforma.

  Los objetos configurables utilizan *pydantic* para definir los esquemas de los parámetros de configuración, asignando como atributo de clase `SCHEMA` una clase que define los atributos y tipos de los parámetros. En este caso, se creó la clase *BaseExplorerSchema* que no tiene atributos, pero se hereda en las clases hijas para definir los parámetros de configuración de cada tipo de explorador.

  Así, la clase *BaseExplorer* presenta los atributos:

  - `TYPE`: Tipo de componente.
  - `DISPLAY_NAME`: Nombre a mostrar en la interfaz.
  - `DESCRIPTION`: Descripción del componente.

- `SCHEMA`: Esquema de los parámetros de configuración.
- `metadata`: Metadatos del componente. Estos metadatos son utilizados para definir restricciones sobre las columnas aceptadas por el tipo de explorador. Definen restricciones de tipo de dato y cantidad de columnas.

Además, la clase *BaseExplorer* define los métodos:

- `get_metadata(cls)`: Método de clase que retorna los metadatos del componente.
- `validate_parameters(cls, params)`: Método de clase que valida los parámetros de configuración del componente.
- `validate_columns(cls, explorer_info, column_spec)`: Método de clase que valida las columnas seleccionadas por el usuario (en explorer_info) con las restricciones definidas en los metadatos del componente. También valida que las columnas seleccionadas estén presentes en la especificación de columnas (column_spec) entregada.
- `prepare_dataset(self, dataset_dict, columns)`: Método que prepara el dataset para la exploración. Este método es llamado antes de ejecutar la exploración y permite a los exploradores realizar tareas de preprocesamiento de los datos, ya sea limpieza, transformación o selección de columnas. Por defecto solamente selecciona las columnas especificadas en el explorador, concatenando los datos de los distintos splits del dataset.
- `launch_exploration(self, dataset, explorer_info)`: Método abstracto, a implementar por las clases hijas, que ejecuta la exploración. Este método es el corazón de la lógica de negocio del explorador, y es el encargado de aplicar las técnicas de exploración y generar los resultados.
- `save_exploration(self, exploration_info, explorer_info, save_path, result)`: Método abstracto, a implementar por las clases hijas, que guarda los resultados de la exploración. Este método es el encargado de guardar los resultados de la exploración en el sistema de archivos de DashAI.
- `get_results(self, save_path, options)`: Método abstracto, a implementar por las clases hijas, que obtiene los resultados de la exploración. Este método es el encargado de cargar los resultados de la exploración desde el sistema de archivos de DashAI.

3.3.3 Exploradores implementados

Habiendo definido las clases y métodos necesarios para soportar las exploraciones y exploradores, se implementaron varios exploradores en DashAI. Estos exploradores permiten a los usuarios explorar los datos cargados en la plataforma y visualizar los resultados de manera interactiva.

Los exploradores implementados son:

- **Explorador de filas**

  El explorador definido como `RowExplorer` permite a los usuarios explorar las filas del dataset, configurando si se muestran las primeras o últimas filas, cuántas filas se muestran y si se desea hacer un muestreo aleatorio de las filas. En la visualización de los resultados, se muestra una tabla con las filas y se permite al usuario buscar, filtrar y ordenar las filas. Este explorador se logra simplemente seleccionando y mostrando las filas del dataset.

- **Explorador descriptor de datos**

  El explorador definido como `DescribeExplorer` permite a los usuarios obtener un resumen estadístico de las columnas del dataset, incluyendo la media, la desviación estándar, el mínimo, el máximo y los cuartiles de las columnas numéricas, y la frecuencia de los valores de las columnas categóricas. En la visualización de los resultados, se muestra una tabla con el resumen estadístico de las columnas y se permite al usuario buscar, filtrar y ordenar las columnas.

  Este explorador se logra mediante el uso de la librería pandas, que permite obtener estadísticas descriptivas de las columnas del dataset con el método `describe()`.

- **Explorador de distribución de puntos**

  El explorador definido como `ScatterPlotExplorer` permite a los usuarios visualizar la distribución de puntos de dos columnas del dataset, pudiendo configurar la agrupación de los puntos por color y símbolos según una tercera o cuarta columna configurables. En la visualización de los resultados, se muestra un gráfico de dispersión con los puntos y se permite al usuario seleccionar y ver información detallada de los puntos.

  Este explorador se logra mediante el uso de la librería Plotly, que permite crear gráficos interactivos de manera sencilla.

- **Explorador de caja y bigotes**

  El explorador definido como `BoxPlotExplorer` permite a los usuarios generar un gráfico de caja y bigotes de un máximo de dos columnas del dataset, una en el eje x y otra en el eje y. En la visualización de los resultados, se muestra un gráfico de caja y bigotes con los valores de las columnas agrupados como cajas. Se permite al usuario seleccionar y ver información detallada de las cajas (cuartiles, mediana, outliers). Además, se permite configurar la orientación de las cajas y si los puntos se muestran.

  Este explorador se logra mediante el uso de la librería Plotly, que permite crear gráficos interactivos de manera sencilla.

- **Explorador de caja de múltiples columnas**

El explorador definido como `MultiColumnBoxPlotExplorer` permite a los usuarios generar un gráfico de caja de varias columnas del dataset. En la visualización de los resultados, se muestra un gráfico de caja con los valores de las columnas esparcidas en el eje x y los valores en el eje y, agrupados como cajas. Se permite al usuario seleccionar y ver información detallada de las cajas (cuartiles, mediana, outliers). Además, se permite configurar la orientación de las cajas y si los puntos se muestran.

Este explorador se logra mediante el uso de la librería Plotly, que permite crear gráficos interactivos de manera sencilla.

- **Explorador de nube de palabras**

  El explorador definido como `WordCloudExplorer` permite a los usuarios generar una nube de palabras a partir de una o más columnas de texto. En la visualización de los resultados, se muestra una imagen de la nube de palabras con las palabras más frecuentes en el texto. Se permite al usuario configurar el máximo de palabras a mostrar y el color de fondo de la nube de palabras.

  Este explorador se logra mediante el uso de la librería wordcloud, que permite crear nubes de palabras a partir de un texto.

3.3.4 API REST

Para interactuar con las clases y métodos implementados en el back-end de DashAI, se crearon nuevas rutas en la API REST de la plataforma. Estas rutas permiten a los usuarios crear, ejecutar y visualizar los resultados de las exploraciones y exploradores.

Las rutas creadas son:

- Para las exploraciones:
  - `GET /exploration/`: Obtiene todas las exploraciones creadas. Soporta paginación.
  - `GET /exploration/{exploration_id}/`: Obtiene una exploración específica.
  - `GET /exploration/dataset/{dataset_id}/`: Obtiene todas las exploraciones asociadas a un dataset. Soporta paginación.
  - `POST /exploration/`: Crea una nueva exploración.
  - `PATCH /exploration/{exploration_id}/`: Actualiza una exploración existente.
  - `DELETE /exploration/{exploration_id}/`: Elimina una exploración existente, eliminando también todos los exploradores asociados y sus resultados.
- Para los exploradores:
  - `GET /explorer/`: Obtiene todos los exploradores creados. Soporta paginación.

- `GET /explorer/{explorer_id}/`: Obtiene un explorador específico.
- `GET /explorer/exploration/{exploration_id}/`: Obtiene todos los exploradores asociados a una exploración. Soporta paginación.
- `POST /explorer/`: Crea un nuevo explorador.
- `PATCH /explorer/{explorer_id}/`: Actualiza un explorador existente.
- `DELETE /explorer/{explorer_id}/`: Elimina un explorador existente y sus resultados.
- `GET /explorer/{explorer_id}/results/`: Obtiene los resultados de un explorador específico.

Todos los métodos de la API REST interactúan con la base de datos de DashAI para crear, leer, actualizar y eliminar exploraciones y exploradores. Algunos, como los de creación y edición, realizan validaciones que requieren cargar el dataset asociado e instanciar el *BaseExplorer* correspondiente al tipo de explorador seleccionado para validar los parámetros y columnas seleccionadas.

Por otro lado, para ejecutar las exploraciones, se reutiliza el endpoint `POST /job/` de DashAI, que permite encolar trabajos en la cola de trabajos de la plataforma. Al encolar un trabajo de tipo *ExplorerJob*, se agrega a la cola de trabajos asíncrona, actualizando el estado del explorador a «delivered».

3.3.5 Front-end

En el front-end de DashAI, se implementaron nuevas vistas y componentes para interactuar con las exploraciones y exploradores. Estas vistas y componentes están basados en las vistas y componentes de la sección de experimentos de DashAI, reutilizando la línea de diseño y su lógica de navegación.

Las vistas y componentes implementados son:

- **Launcher del módulo**

  El launcher del módulo de exploración es un componente que permite a los usuarios lanzar el módulo de exploración y visualización de datos para algún dataset cargado en la plataforma. Como se puede ver en la Figura 14, el launcher se encuentra en la sección de datasets de DashAI, en la vista de resumen de un dataset. Al hacer clic en el botón «EXPLORATIONS», se inicia un modal que contiene el módulo embebido.

  Adicionalmente, también inicia un contexto de React que almacena toda la información que utilizará el módulo, como el dataset seleccionado, las columnas del dataset, las exploraciones y los exploradores, además de los estados que controlan la visibilidad de los componentes y cambios de estado internos del módulo.

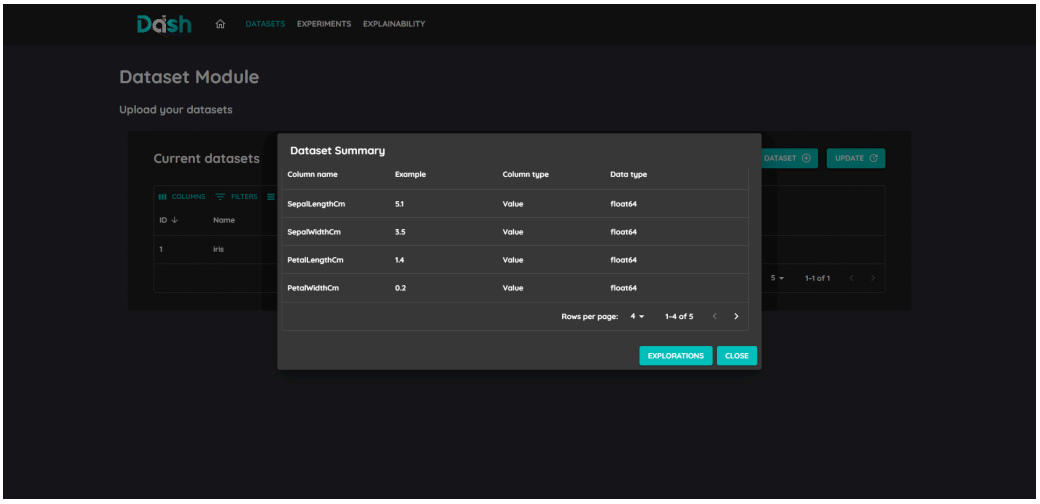

Figura 14: Panel de resumen de un dataset con el botón de exploraciones.

- **Módulo de exploración y visualización de datos**

  El módulo de exploración y visualización de datos es el componente principal de la solución, desde el que se accede a las exploraciones creadas por los usuarios. Como se puede ver en la Figura 15, el módulo inicialmente muestra una lista de exploraciones creadas para el dataset seleccionado.

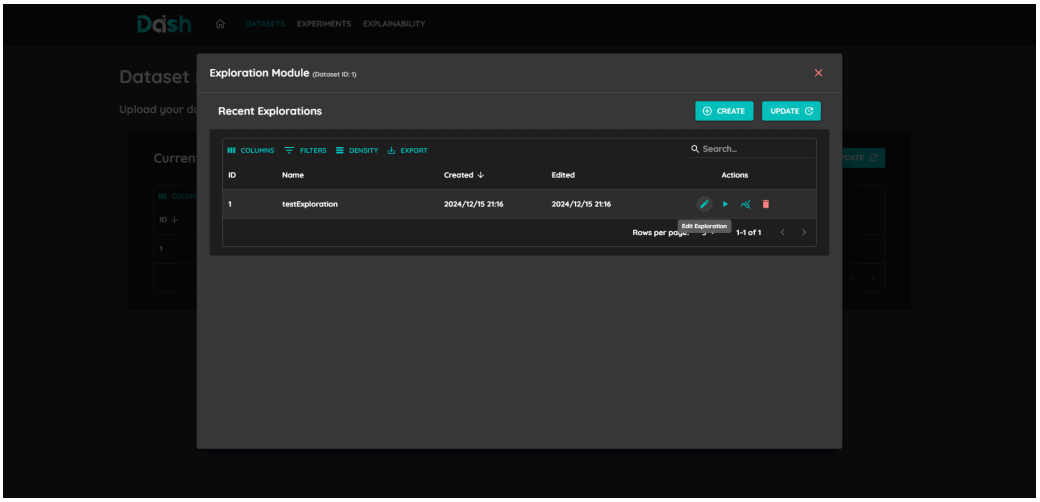

Figura 15: Módulo de exploración y visualización de datos.

  Desde el módulo, los usuarios pueden crear nuevas exploraciones, editar exploraciones existentes, ejecutar exploraciones y visualizar los resultados de las exploraciones. Para cada exploración, se muestra el id, nombre, fechas de creación y modificación, y las acciones disponibles (editar, ejecutar, ver resultados y eliminar).

- **Editor de exploraciones**

El editor de exploraciones es un mismo componente que permite a los usuarios crear y editar exploraciones. Como se puede ver en la Figura 16, al hacer clic en el botón «NEW EXPLORATION» el editor permite a los usuarios configurar el nombre y la descripción de la exploración. Luego de esto, como se muestra en la Figura 17 y la Figura 18, los usuarios deben configurar los exploradores que tendrá la exploración, asignándole opcionalmente un nombre y seleccionando que tipo de explorador se añadirá.

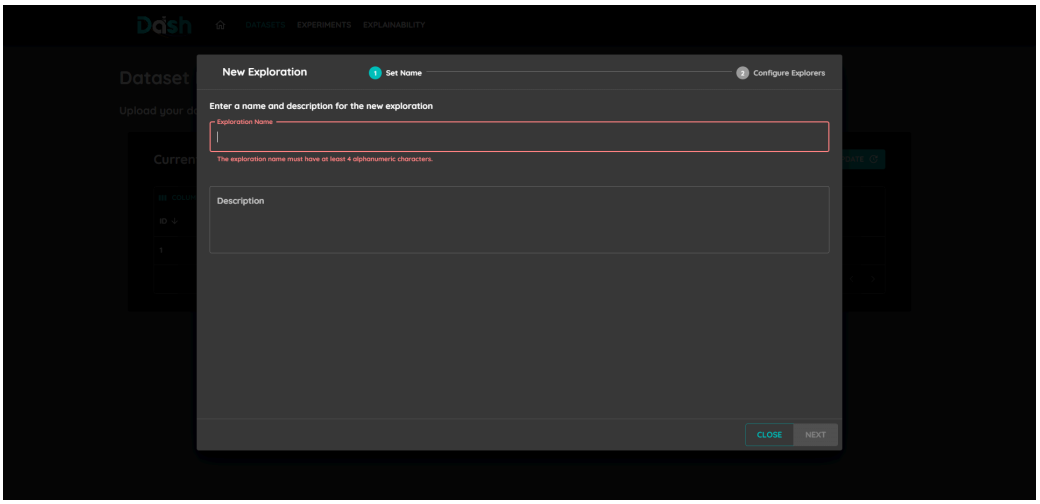

Figura 16: Editor de exploraciones: Asignación de nombre y descripción.

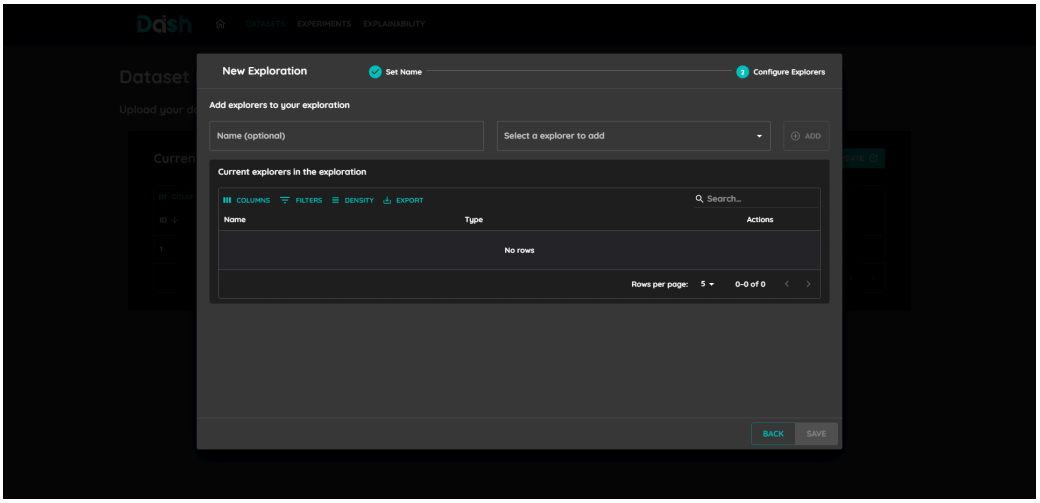

Figura 17: Editor de exploraciones: Asignación de exploradores.

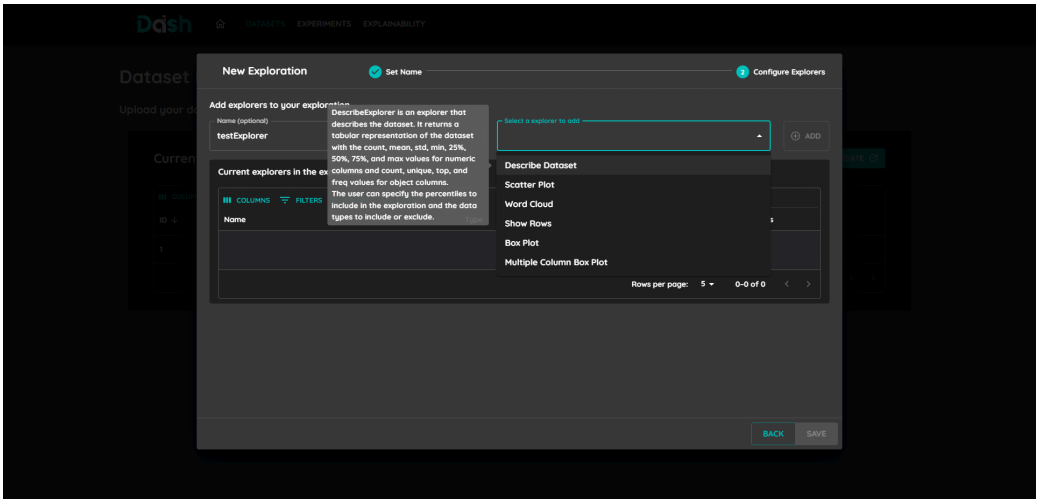

Figura 18: Tooltip de explorador en el editor de exploraciones.

Al seleccionar un explorador, se muestra un tooltip con una descripción del explorador y un botón para añadirlo a la exploración. Al hacer clic en el botón «ADD», se añade el explorador a la exploración y se muestra en la lista de exploradores de la exploración.

Debido a que los exploradores son agregados con una configuración automática de columnas, si estas fueron limitadas por una cardinalidad máxima, se muestra un mensaje para informarlo, como se ve en la Figura 19.

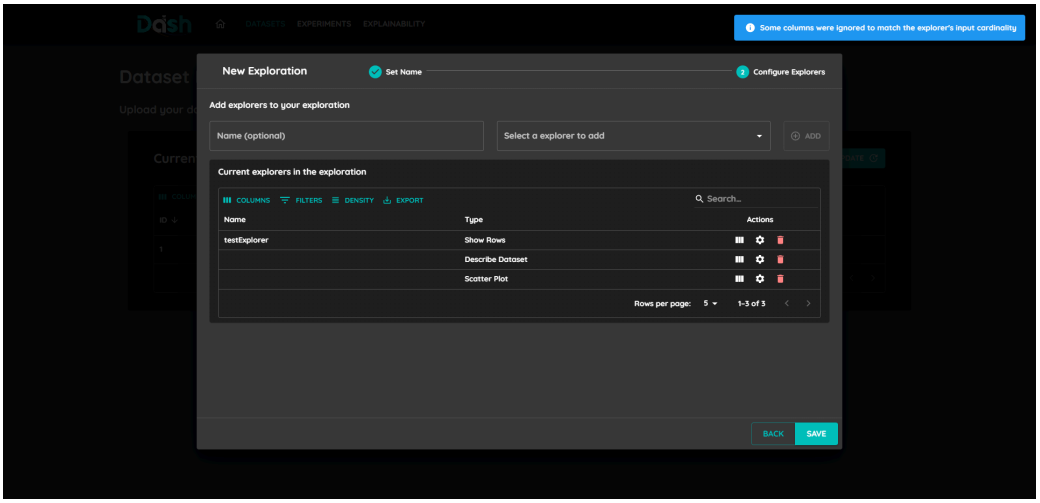

Figura 19: Editor de exploraciones: Columnas ignoradas por cardinalidad máxima.

Cada uno de los exploradores de la lista tiene acciones que permiten editar la selección de columnas, editar los parámetros de configuración y eliminar el explorador de la exploración.

- **Selector de columnas**

El selector de columnas es un componente que permite a los usuarios seleccionar las columnas del dataset que serán utilizadas por los exploradores. Como se puede ver en la Figura 20, el selector de columnas muestra una lista de columnas del dataset con un checkbox para seleccionar las columnas. Este componente tiene autoguardado con validación, si el usuario tiene una selección inválida, se muestra un mensaje de error.

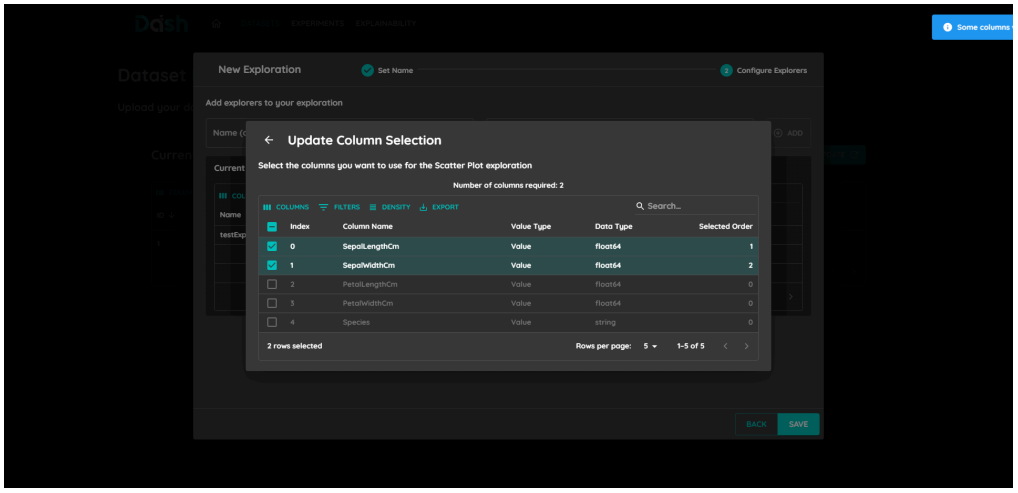

Figura 20: Selector de columnas de explorer (ScatterExplorer).

- **Editor de parámetros de objeto configurable**

El editor de parámetros es un componente que ya estaba implementado en DashAI y permite a los usuarios configurar los parámetros de los objetos configurables de la plataforma. En la Figura 21 y Figura 22 se muestra el editor de parámetros para los exploradores de filas y de distribución de puntos, respectivamente.

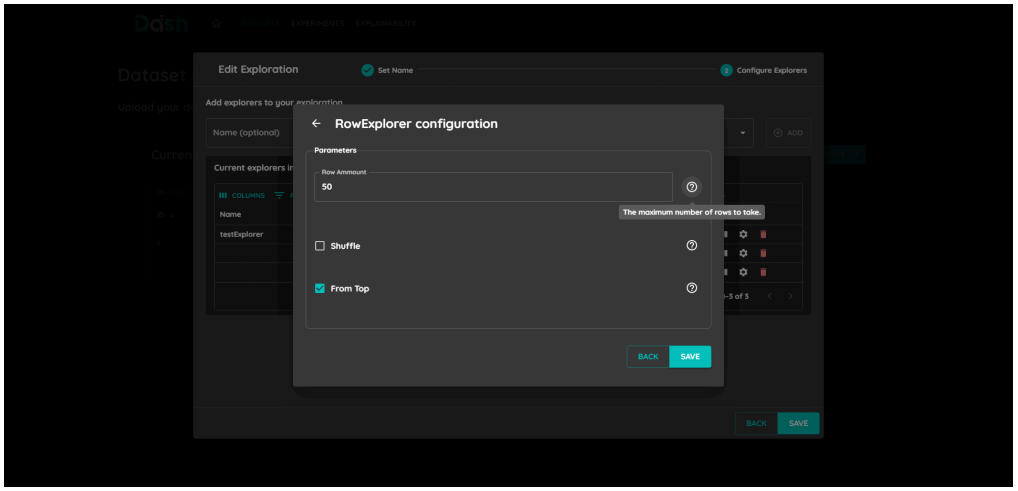

Figura 21: Editor de parámetros de explorer (RowExplorer).

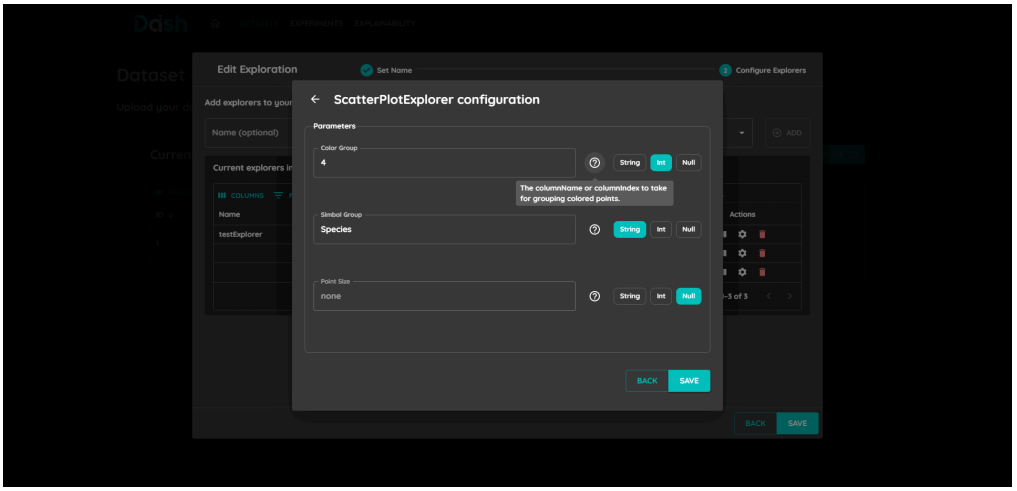

Figura 22: Editor de parámetros de explorer (ScatterPlotExplorer).

- **Ejecutor de exploración**

El ejecutor de exploración es un componente que permite a los usuarios ejecutar las exploraciones creadas. Se accede a él desde la acción en lista de exploraciones.

Como se puede ver en la Figura 23, el ejecutor de exploraciones muestra una lista de los exploradores de la exploración y un botón para ejecutar la exploración. Al hacer clic en el botón «START», se ejecutan los exploradores seleccionados y se inicia un refresco automático de los resultados hasta que todos ellos terminen.

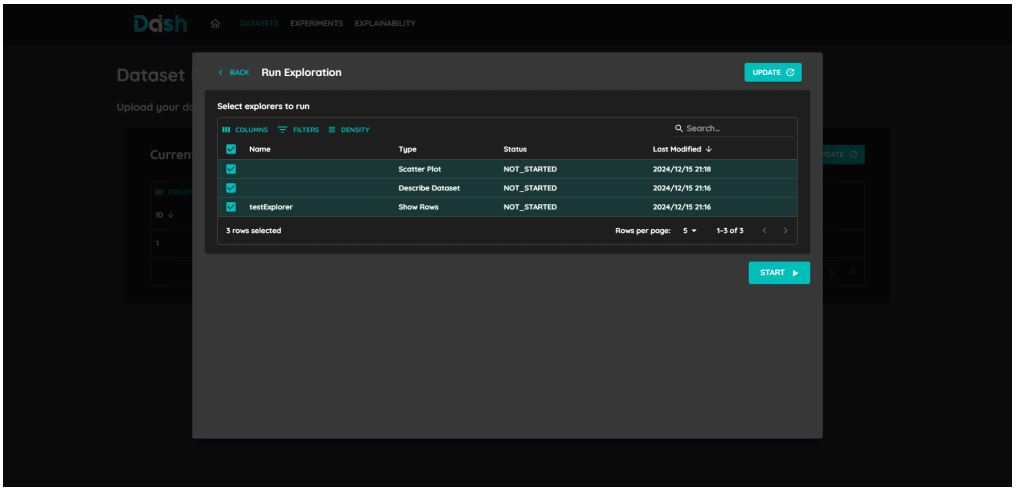

Figura 23: Ejecutor de exploración.

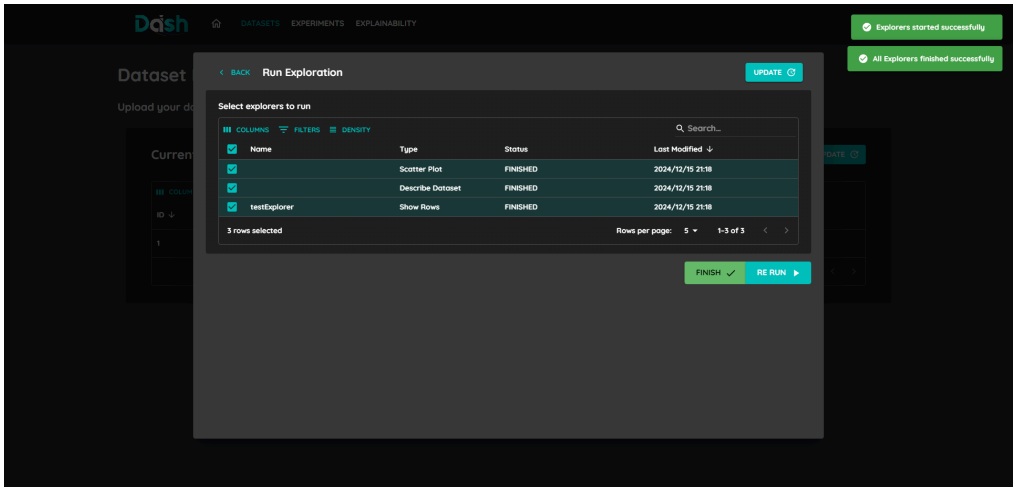

Figura 24: Ejecutor de exploración con todos los exploradores ejecutados.

- **Visualizador de resultados**

El visualizador de resultados es un componente que permite a los usuarios visualizar los resultados de las exploraciones ejecutadas. Se accede a él desde la acción en lista de exploraciones.

El visualizador de resultados tiene dos modos, uno para visualizar todos los resultados de la exploración y otro para un explorador específico. En el modo de visualización de todos los resultados, como se puede ver en la Figura 25 y la Figura 26, se muestran directamente los resultados uno tras otro.

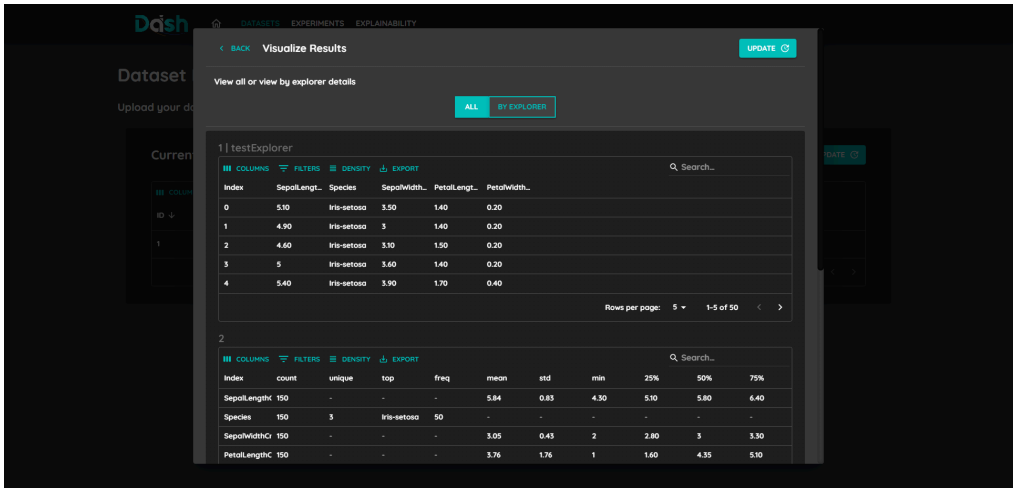

Figura 25: Visualizador de resultados.

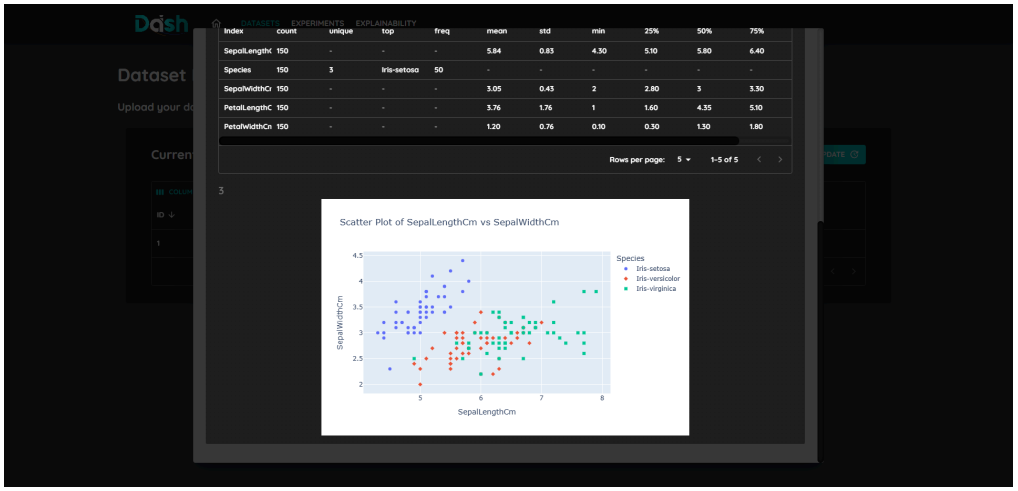

Figura 26: Visualizador de resultados, parte inferior de la vista.

En el modo de visualización de un explorador específico, como se puede ver en la Figura 27, se muestra una lista de exploradores, donde se puede acceder al detalle de estos con un botón de acción.

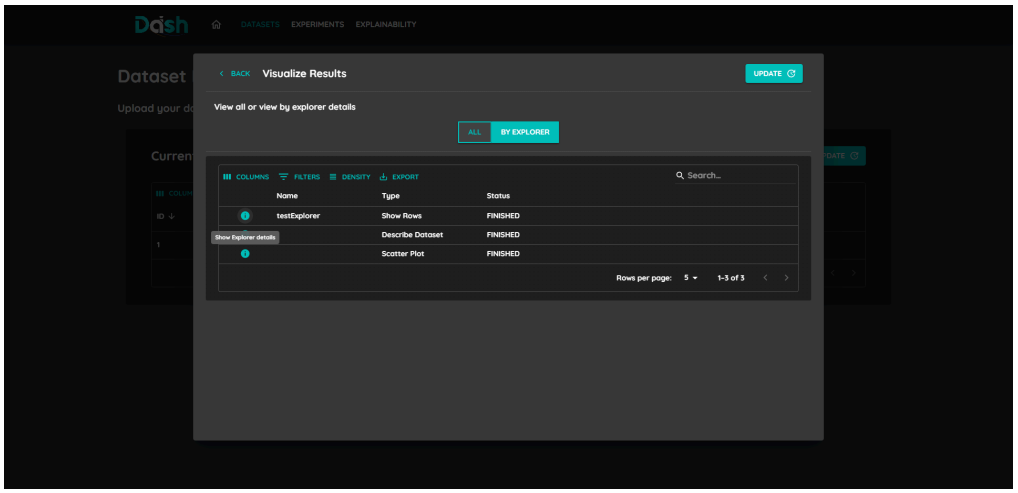

Figura 27: Visualizador de resultados por explorador.

En el detalle de un explorador, como se puede ver en las siguientes figuras, inicialmente se muestra el resultado, pero mediante tabs se puede acceder ver la información, la selección de columnas y los parámetros de configuración del explorador.

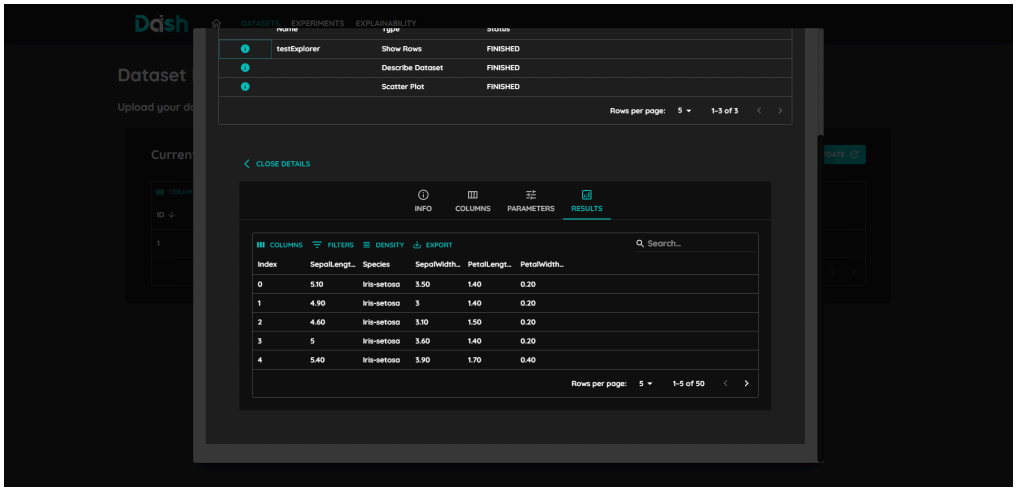

Figura 28: Visualizador de resultados de un explorador: Resultados.

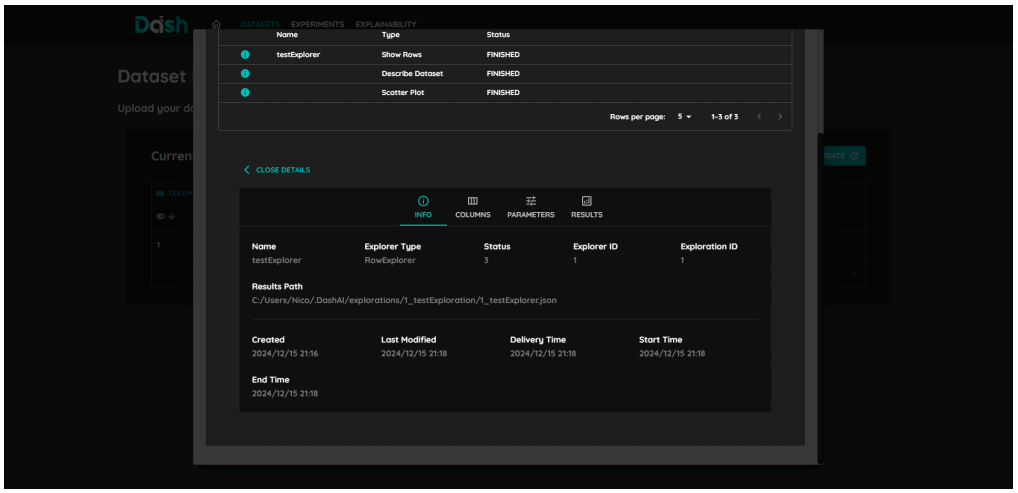

Figura 29: Visualizador de resultados de un explorador: Información.

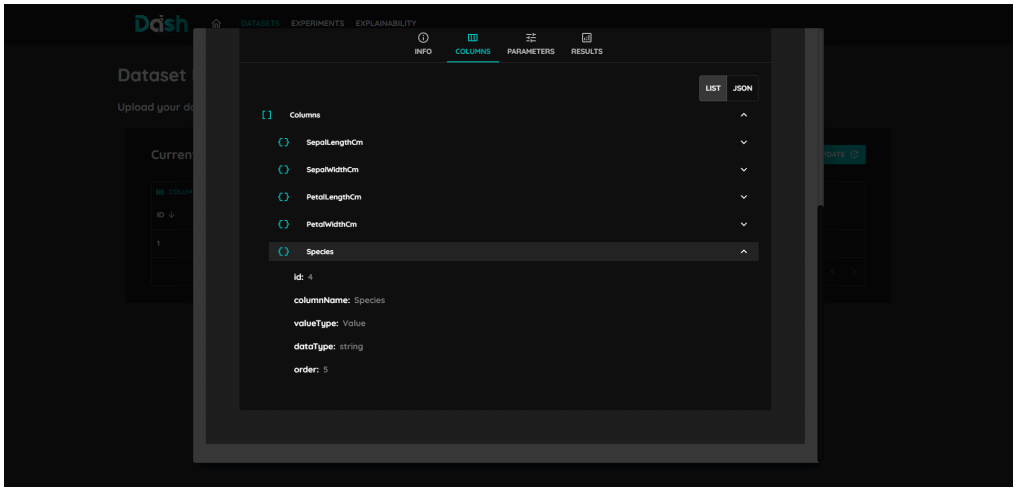

Figura 30: Visualizador de resultados de un explorador: Selección de columnas.

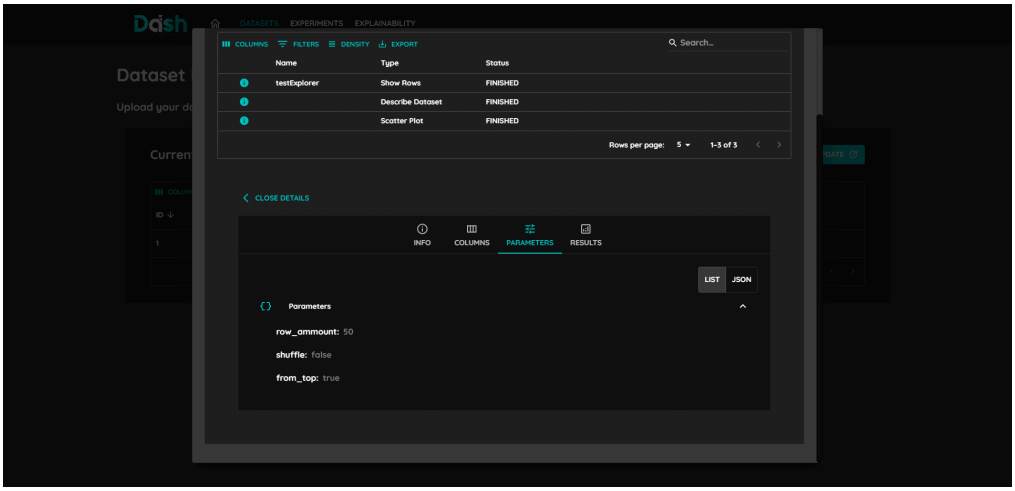

Figura 31: Visualizador de resultados de un explorador: Parámetros de configuración.

Por último, también se agregaron pequeños componentes generales como «JsonDisplayer», «NestedListDisplayer» y «TooltipedCellItem» para mostrar información de manera más amigable y reutilizable.

Capítulo 4

Evaluación

En este capítulo se detalla como se evaluó la solución implementada. Se describen las pruebas de usabilidad realizadas con usuarios, los resultados obtenidos y las conclusiones derivadas de las pruebas.

4.1 Pruebas de usabilidad

Para evaluar la solución implementada, se realizaron pruebas de usabilidad con 6 usuarios, de distintos perfiles, que probaron las nuevas funcionalidades de la aplicación. Los usuarios completaron una serie de tareas y respondieron a una encuesta de satisfacción al finalizar las pruebas. Estas pruebas se realizaron en un entorno controlado y se registraron los tiempos de ejecución de las tareas, las respuestas de los usuarios, los errores encontrados y las sugerencias de mejora, identificando los problemas y oportunidades de mejora en la solución implementada.

En las pruebas de usuario realizadas, se evaluaron dos nuevos trabajos, el de la presente memoria, destinado a la exploración de datos, y el de la memoria del estudiante Isaías Venegas, destinado a la transformación de datos, mediante los llamados «Converters» y su módulo. Dado que ambos trabajos se complementan en funcionalidades, problemas y relevancia, se evaluaron en conjunto para obtener una visión más completa de la plataforma.

4.1.1 Perfiles de usuario

Los perfiles de usuario seleccionados para las pruebas de usabilidad fueron:

ID	Perfil	Experiencia
U1	Tesista DCC	Ha cursado el ramo de minería de datos y utiliza ChatGPT, Dall-E, Jenny.AI.

ID	Perfil	Experiencia
U2	Tesista DIM	Ha implementado redes neuronales con Keras y utilizado Scikit-learn para clasificación.
U3	Tesista DIE	Ha trabajado en proyectos académicos del área y utilizado aprendizaje de máquinas en su tesis.
U4	Titulada Industrias	Solo ha utilizado Transformers de Google, pero no entendió bien como funcionan.
U5	Estudiante DCC	No tiene mucha experiencia en el área, solo lo que recuerda de minería de datos, solamente utiliza ChatGPT.
U6	Tesista DCC	Ha utilizado muchos modelos de ML distintos, principalmente de Scikit-learn y Keras. Ha probado muchas alternativas de modelos LLM

Tabla 1: Perfiles de usuario para las pruebas de usabilidad.

Entre los usuarios anteriores, U3 y U6 fueron los que se destacaron por su experiencia en el área, mientras que los U1, U2 y U5 tenían una experiencia media y U4 tenía muy poca experiencia.

4.1.2 Preparación

Para la realización de las pruebas de usabilidad, se preparó una rama de desarrollo con las nuevas funcionalidades implementadas, y se desplegó localmente para que los usuarios pudieran acceder a la plataforma.

Se prepararon una serie de tareas que los usuarios debían completar. Para facilitar las pruebas, solo se evaluaron los módulos nuevos y no la plataforma en su totalidad. Las tareas se centraron en la exploración y transformación de datos, y se diseñaron de manera que los usuarios pudieran probar las funcionalidades de los módulos en al menos dos flujos.

Así, se preparó con anterioridad dos datasets idénticos en la plataforma, uno para cada flujo, con el fin de que los usuarios pudieran realizar las tareas de manera independiente. El dataset utilizado es «wine»[43], un dataset de clasificación de vinos, que contiene 13 atributos (las características de los vinos) y una clase objetivo.

Además de esto, se preparó un video de presentación de la plataforma, para que los usuarios pudieran familiarizarse ligeramente con ella. Este video no explica las funcionalidades nuevas, sino que muestra el estilo y la navegación de la plataforma.

4.1.3 Tareas

Las tareas que los usuarios debían completar se dividieron en dos flujos, ambos interactuando con los nuevos módulos de la plataforma.

Se diseñaron dos flujos, para disminuir posible sesgo de adaptación que los usuarios pudieran tener al encontrar la plataforma por primera vez. El primer flujo es de menor complejidad, y su objetivo, además de analizar la primera impresión de los usuarios, es que estos se familiaricen con la plataforma. El segundo flujo, de mayor complejidad, busca evaluar la capacidad de los usuarios para explorar y transformar datos de manera más avanzada.

Las tareas de los flujos se presentan en las siguientes tablas:

ID	**Tarea**	**Criterio de éxito**
T1	Abrir el módulo de datos y observar el resumen de datos.	El usuario puede acceder al módulo de datos y comprender la información resumida que se presenta.
T2	Abrir el módulo de exploración para un dataset.	El usuario inicia correctamente el módulo de exploración para un dataset y confirma que se puede acceder a las herramientas disponibles.
T3	Crear una exploración con estadísticas de resumen y ejecutarla.	El usuario configura una nueva exploración con estadísticas de resumen y ejecuta el análisis sin problemas.
T4	Visualizar los parámetros, columnas y resultados de la exploración.	El usuario puede identificar y comprender los resultados generados por una exploración y los parámetros y columnas utilizados.
T5	Abrir el módulo de convertidores para modificar un dataset.	El usuario accede al módulo de convertidores y selecciona un dataset para realizar modificaciones.
T6	Filtrado de convertidores.	El usuario localiza y selecciona un convertidor específico utilizando los filtros disponibles.
T7	Ejecutar convertidor con los hiperparámetros y alcance adecuados.	El usuario edita correctamente los hiperparámetros de un convertidor y lo aplica a una parte especifica del dataset.
T8	Volver a ejecutar una exploración y ver sus cambios	El usuario re-ejecuta la exploración creada y verifica que los cambios realizados en los datos están reflejados en los resultados.
T9	Añadir explorador de tipo box plot.	El usuario modifica la exploración existente y ejecuta el análisis actualizado.

Tabla 2: Tareas del flujo 1 de las pruebas de usabilidad

A partir de este punto, se espera que el usuario haya adquirido un conocimiento básico de la plataforma y pueda realizar tareas más complejas. Por lo que se diseñaron tareas más

avanzadas para el segundo flujo y con instrucciones menos detalladas. Este flujo incumbe a la copia del dataset original, por lo que actúa como un flujo independiente.

ID	Tarea	Criterio de éxito
T10	Aplicar secuencialmente dos convertidores.	El usuario utiliza dos convertidores en secuencia y confirma que las modificaciones esperadas en los datos se han aplicado correctamente.
T11	Crear exploración, ejecutar y visualizar su resultado.	El usuario genera una exploración para los datos modificados en secuencia, la ejecuta y valida los resultados.

Tabla 3: Tareas del flujo 2 de las pruebas de usabilidad

Se debe notar que las tareas tienen como objetivo evaluar una o más funcionalidades de alguno de los dos módulos. En la Tabla 4 se muestra la relación entre las tareas y funcionalidades evaluadas, solamente para el módulo de la presente memoria.

Tarea	Funcionalidad evaluada
T2	Inicio del módulo para un dataset específico.
T3, T11	Creación de una exploración y sus exploradores.
T3, T8, T11	Ejecución de una exploración.
T4, T8, T11	Visualización de resultados de exploración.
T9	Edición de exploraciones.

Tabla 4: Asociación de tareas y funcionalidades evaluadas

Durante y después de la realización de las tareas, se acumulaba la retroalimentación de los usuarios y posibles dificultades y errores encontrados. Al finalizar las pruebas, se realizaron las siguientes preguntas:

- ¿Qué aspectos encontraste más útiles o eficaces durante la realización de las tareas? ¿Por qué consideras que estos aspectos fueron destacados en tu experiencia de uso?
- ¿Alguna tarea te resultó especialmente difícil de completar?. ¿Por qué crees que experimentaste dificultades y cómo crees que podrían abordarse?
- Basado en tu experiencia con DashAI, ¿qué recomendaciones harías a otros usuarios que deseen utilizar este software? ¿Qué crees que hace a DashAI destacarse o ser único en comparación con otras herramientas similares?

4.1.4 Encuesta SUS

Para complementar las pruebas de usabilidad, los usuarios contestaron una breve encuesta de satisfacción para **uno de los dos** módulos evaluados. La encuesta utilizada fue el System Usability Scale (SUS)[44], una encuesta de 10 preguntas que evalúa la usabilidad de un sistema. Las preguntas de la encuesta se presentan a continuación:

1. Creo que me gustaría utilizar este sistema con frecuencia.
2. Encontré el sistema innecesariamente complejo.
3. Creo que el sistema es fácil de usar.
4. Creo que necesitaría el apoyo de una persona técnica para poder utilizar este sistema.
5. Creo que las distintas funciones del sistema están bien integradas.
6. Creo que hay demasiada inconsistencia en el sistema.
7. Imagino que la mayoría de las personas aprenderían a utilizar este sistema muy rápidamente.
8. Creo que el sistema es muy engorroso de utilizar.
9. Me siento confiado al utilizar el sistema.
10. Necesitaría aprender muchas cosas antes de poder utilizar este sistema.

Los usuarios debían responder a cada pregunta en una escala de 1 a 5, donde 1 es «Totalmente en desacuerdo» y 5 es «Totalmente de acuerdo». Los resultados de la encuesta se presentan en la siguiente sección.

4.2 Resultados

A continuación, se presentan los resultados obtenidos para las tareas presentadas en la Tabla 4.

4.2.1 Resultados de las tareas

✓: Completado, ✗: No completado, •: Completado con dificultades.

Tarea	U1	U2	U3	U4	U5	U6	% Éxito
T2	✓	✓	✓	✓	✓	✓	100%
T3	✓	✓	✓	✓	✓	✓	100%
T4	✓	✓	•	•	✓	✓	67%
T8	•	✓	•	✓	•	✓	50%
T9	•	✓	✓	✓	•	•	50%
T11	✓	✓	✓	✓	✓	✓	100%

Tabla 5: Resultados de las tareas de las pruebas de usabilidad

Como se puede observar en la Tabla 5, la mayoría de los usuarios completaron con éxito las tareas, sin embargo, algunas tareas se completaron presentando dificultades para algunos usuarios, lo que indica que existen oportunidades de mejora en la implementación.

Cabe destacar que ninguna tarea quedo incompleta, lo que indica que las funcionalidades implementadas cumplen con su objetivo.

4.2.2 Dificultades encontradas

Durante las pruebas de usabilidad, los usuarios encontraron algunas dificultades al utilizar las nuevas funcionalidades de la plataforma. Las principales dificultades encontradas fueron:

- **Dificultad para identificar los parámetros y columnas de las exploraciones**: Algunos usuarios tuvieron problemas para identificar los parámetros y columnas utilizados en las exploraciones, específicamente encontraban dificultades para localizar la información en la interfaz de usuario. Se recopila al hablar con los usuarios que esto se debe a que la información es fácilmente accesible, pero no se destaca lo suficiente como para ser identificada rápidamente, y por lo tanto, se pasa por alto.
- **Dificultad para Re-ejecutar exploraciones**: Algunos usuarios tuvieron problemas para re-ejecutar exploraciones. Se recopila al hablar con los usuarios que esto se debe a que al re-ejecutar una exploración, se destaca el botón de finalización, pero no se destaca el botón de re-ejecución, lo que no es intuitivo para los usuarios, cerrando el panel de lanzamiento de exploración sin haber ejecutado ningún explorador.
- **Dificultad para editar exploraciones**: Algunos usuarios tuvieron problemas para editar exploraciones y agregarle exploradores. Se observó que algunos de los usuarios no distinguían conceptualmente entre exploradores y exploraciones. Al recitar las instrucciones, algunos de los usuarios no entendían que una exploración podía contener varios exploradores, y que estos podían ser editados de manera independiente. Estos usuarios lograban completar la tarea creando una nueva exploración.

- **Dificultad para reconocer la interactividad de las visualizaciones**: La mayoría de los usuarios no reconocieron que las visualizaciones eran interactivas. Esto se debe a que no se destaca lo suficiente esta característica en la interfaz de usuario. Algunos usuarios no interactuaron con las visualizaciones, y otros lo hicieron de manera accidental.

4.2.3 Tiempos de ejecución

Los tiempos de ejecución de las tareas se presentan en la siguiente tabla:

Tarea	**U1**	**U2**	**U3**	**U4**	**U5**	**U6**	**Tiempo promedio (s)**
T2	13	14	19	29	35	79	32
T3	68	96	94	124	315	105	134
T4	224	75	84	161	244	94	147
T8	96	87	185	210	315	60	159
T9	125	72	145	173	480	297	215
T11	63	103	46	127	115	87	90

Tabla 6: Tiempos de ejecución de las tareas de las pruebas de usabilidad

Como se puede observar en la Tabla 6, los tiempos de ejecución de las tareas varían entre los usuarios.

Para la tarea T2, que es la más sencilla, los tiempos de ejecución son menores, lo que era esperado.

Para la tarea T3, era la primera vez que los usuarios creaban una exploración, por lo que los tiempos de ejecución son mayores, sin embargo, esto era esperado, ya que los usuarios debían familiarizarse con la plataforma. Esto se valida por el usuario con mayor tiempo de ejecución, que expresó que esta primera instancia generaba dudas sobre los resultados que el explorador entregaría con la configuración inicial, dudas que no se presentarían cuando hubiese realizado más exploraciones.

Para las tareas T4, T8 y T9, que son más complejas, los tiempos de ejecución son mayores, sin embargo, estas eran las tareas que presentaban más dificultades para los usuarios, afectando el tiempo de ejecución de las mismas.

Finamente, la tarea T11, que era la más compleja debido a que evaluaba varias funcionalidades anteriores, presentó tiempos de ejecución menores, lo que indica que los usuarios se familiarizaron con la plataforma y pudieron completar las funcionalidades de manera más rápida. Esto valida la hipótesis de que la familiaridad con la plataforma influye beneficiosamente en los tiempos de ejecución.

4.2.4 Encuesta SUS

Como se mencionó en la Sección 4.1.4, con el fin de obtener una evaluación específica para cada módulo en vez de global, los usuarios se dividieron en dos grupos, donde uno contestó la encuesta SUS para el módulo de exploración y el otro para el módulo de transformación. No se pidió a los usuarios que contestaran ambas encuestas, ya que esto podría generar sesgo en las respuestas debido a la repetición de preguntas. Los resultados de la encuesta SUS para el módulo de exploración de datos se presentan en la siguiente tabla:

Usuario	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**	**P9**	**P10**
U1	4	1	4	2	5	1	4	1	4	1
U4	3	3	3	2	4	1	1	3	5	1
U6	3	1	4	1	4	1	5	1	4	2

Tabla 7: Resultados de la encuesta SUS para el módulo de exploración de datos

Recordando que las preguntas impares son positivas (columnas grises) y las pares negativas (columnas blancas), se puede observar en la Tabla 7 que los resultados de la encuesta SUS indican que los usuarios tienen una percepción positiva de la usabilidad del módulo de exploración de datos. La mayoría de los usuarios estuvieron de acuerdo en que el sistema es fácil de usar, que las distintas funciones del sistema están bien integradas y que la mayoría de las personas aprenderían a utilizar el sistema muy rápidamente.

Sin embargo, uno de los usuarios no estuvo de acuerdo en que el sistema es fácil de usar y que necesitarían el apoyo de una persona técnica para poder utilizar el sistema, lo que indica que existen oportunidades de mejora en la usabilidad del sistema. El contexto de esta persona es que no tiene experiencia en el área del análisis de datos, por lo que la curva de aprendizaje es mayor y la percepción de la usabilidad se espera que sea menor.

Finamente, para evaluar los resultados de esta encuesta, se debe aplicar una fórmula que calcula el puntaje de usabilidad del sistema. La fórmula es la siguiente:

$$\text{Puntaje SUS} = \Big(\sum(\text{P\_impares} - 1) + \sum(5 - \text{P\_pares})\Big) * 2.5$$

Donde «P_impares» son las respuestas a las preguntas impares y «P_pares» son las respuestas a las preguntas pares. El puntaje de usabilidad del sistema se calcula en una escala de 0 a 100, donde 0 es el peor puntaje y 100 es el mejor puntaje. Tras un estudio de Jeff Sauro[45], se estableció que un puntaje de 68 es el promedio de usabilidad de un sistema, por lo que cualquier puntaje por encima de 68 indica que el sistema es más usable que la mayoría de los sistemas. Para representarlo de manera más visual, se puede utilizar la escala presentada en la Figura 32.

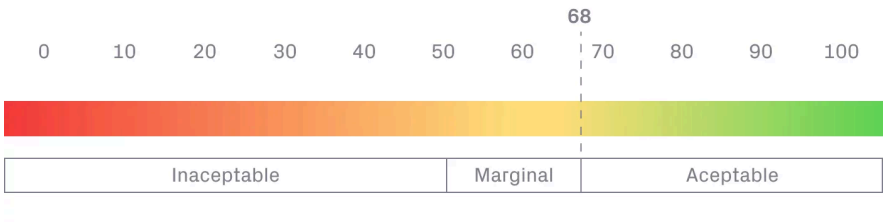

Figura 32: Representación de los resultados de un puntaje SUS

El puntaje de usabilidad del sistema se calcula para cada usuario y se presenta en la siguiente tabla:

Usuario	**Puntaje SUS**	**Resultado**
U1	87,5	Aceptable
U4	65	Marginal
U6	85	Aceptable

Tabla 8: Puntajes de usabilidad del sistema para el módulo de exploración de datos

Finalmente, para concluir, se puede observar en la Tabla 8 que los puntajes de usabilidad del sistema para el módulo de exploración de datos son aceptables, con un puntaje promedio de 79,2. Esto indica que los usuarios tienen una percepción positiva de la usabilidad del sistema, pero que existen oportunidades de mejora en la usabilidad del sistema. Cabe destacar que los resultados no implican buena usabilidad directamente, sino que indican que los usuarios perciben la usabilidad del sistema de manera positiva.

Capítulo 5

Conclusión

En este capítulo se presentan las conclusiones del trabajo, resumiendo las principales contribuciones, reflexionando sobre el aprendizaje obtenido y proponiendo posibles mejoras y trabajos futuros sobre esta memoria y la plataforma DashAI.

5.1 Principales contribuciones

Las funcionalidades nuevas a las que puede acceder el usuario luego del desarrollo de esta memoria son:

- El usuario puede utilizar el módulo de exploración de datos para analizar y visualizar los datos de un dataset.
- El usuario puede crear exploraciones con diferentes exploradores.
- El usuario puede utilizar los siguientes exploradores y visualizar sus resultados:
  - Estadísticas de resumen, que muestra de manera tabular estadísticas descriptivas de las columnas del dataset.
  - Explorador de filas, que muestra de manera tabular una cantidad configurable de filas del dataset.
  - Scatter plot, que muestra un gráfico de dispersión de dos columnas del dataset, con la posibilidad de configurar columnas que definan la agrupación de los puntos en colores y símbolos.
  - Box plot, que muestra un gráfico de caja de dos columnas del dataset.
  - Box plot de múltiples columnas, que muestra un gráfico de caja de múltiples columnas del dataset en un mismo eje.
  - Wordcloud, que muestra una nube de palabras a partir de una o más columnas de texto del dataset.

- El usuario puede editar y re-ejecutar exploraciones, y visualizar los resultados actualizados.
- El usuario puede eliminar exploraciones y exploradores, limpiando los resultados y liberando recursos.
- El usuario o desarrolladores pueden extender la plataforma con nuevos exploradores y visualizaciones.

5.2 Retrospectiva

En soluciones como DashAI, la extensibilidad y modularidad son aspectos fundamentales, y el desarrollo de esta memoria ha permitido al estudiante comprender la importancia de estos aspectos y cómo implementarlos en un proyecto de software real. La implementación de los nuevos módulos ha requerido un análisis detallado de los requisitos y una planificación cuidadosa de la arquitectura y diseño de la solución, lo que ha permitido al estudiante adquirir habilidades en la toma de decisiones y la resolución de problemas en un entorno de desarrollo real.

El contexto y la complejidad de la plataforma fueron los que más aportaron al aprendizaje del estudiante. La plataforma DashAI es un proyecto de software complejo y en constante evolución, lo que ha permitido al estudiante enfrentarse a desafíos técnicos y conceptuales de alto nivel y adquirir habilidades en el diseño y desarrollo de software a gran escala, además de aprender más sobre el área de análisis de datos y aprendizaje de máquinas.

Como reflexión personal, si se tuviera la oportunidad de volver a realizar este trabajo, se dedicaría más tiempo a la evaluación de la solución, ya que esto habría permitido al estudiante no solamente identificar los problemas y oportunidades de mejora en la solución, sino también solucionarlos y mejorar la calidad de la solución final.

Además, se considera que la implementación del nuevo módulo ha sido un éxito, ya que se han cumplido los objetivos del trabajo y se han implementado las funcionalidades propuestas de manera satisfactoria. La solución implementada es funcional y cumple con los requisitos del proyecto, y se espera que los usuarios de la plataforma puedan beneficiarse de las nuevas funcionalidades y mejorar su experiencia de uso.

Finalmente, el estudiante considera que el desarrollo de esta memoria ha sido una experiencia enriquecedora y desafiante, que le ha permitido adquirir nuevas habilidades y conocimientos en el desarrollo de software y análisis de datos, y que le ha preparado para enfrentar desafíos más complejos en el futuro.

5.3 Trabajo futuro

A pesar de los logros obtenidos en esta memoria, existen oportunidades de mejora y trabajos futuros que podrían ser realizados. Algunas de las mejoras y trabajos futuros propuestos son:

- **Soporte a tipos de datos**

  Los exploradores y visualizaciones implementados en esta memoria están diseñados para trabajar con los tipos de datos más comunes, como números y texto, utilizando los tipos de *Value* definidos por la librería «datasets» de huggingFace en sus *Features*[46]. Sobre estos y su presencia en el dataset, los exploradores validan si se pueden ejecutar para los datos presentes.

  Sin embargo, dado que DashAI busca extender su soporte a otros tipos de datos, como imágenes y audio, sería necesario extender los exploradores para soportar estos tipos de datos. Un trabajo en progreso es la implementación de tipos en DashAI, extender los exploradores para soportar cualquier tipo de *Feature* mediante estos tipos sería un trabajo futuro ideal para mejorar la plataforma. Se dejó un espacio en la implementación para que esta extensión sea sencilla y no afecte el funcionamiento actual de los exploradores, además de dejar documentado el proceso de extensión.

- **Confusión entre exploradores y exploraciones**

  Durante las pruebas de usabilidad, se identificó que algunos usuarios tenían dificultades para distinguir entre exploradores y exploraciones, y que esto afectaba su capacidad para editar y re-ejecutar exploraciones. Para abordar este problema, se propone mejorar la interfaz de usuario para que sea más clara y fácil de entender, y proporcionar instrucciones más detalladas y ejemplos de uso para ayudar a los usuarios a comprender la diferencia entre exploradores y exploraciones.

- **Mejoras en la visualización de resultados**

  Durante las pruebas de usabilidad, se identificó que los usuarios tenían dificultades para identificar los parámetros y columnas de las exploraciones. Para abordar este problema, se propone mejorar las vistas de los resultados de las exploraciones para la información del explorador sea más destacada y fácil de identificar.

  Además, se propone mejorar la interactividad de las visualizaciones para que los usuarios puedan interactuar con ellas de manera más intuitiva y descubrir información oculta. Por ejemplo, para los gráficos interactivos se podría destacar la interactividad con un mensaje, icono o mostrando el panel de herramientas de interacción no solo al pasar el mouse por encima de la visualización.

- **Almacenar y mostrar errores de exploradores**

  Debido a la naturaleza asincrónica de las exploraciones, los errores que se producen durante la ejecución de los exploradores no se muestran al usuario. Para mejorar la experiencia del usuario, se propone almacenar y mostrar los errores de los exploradores en la interfaz de usuario, para que los usuarios puedan identificar y corregir los problemas que se producen durante la ejecución de los exploradores.

Con estas mejoras y trabajos futuros, se espera que la plataforma DashAI siga evolucionando y mejorando para satisfacer las necesidades de los usuarios y proporcionar una experiencia de uso más intuitiva y eficiente.

Bibliografía

[1] IBM Data and AI Team, «10 everyday machine learning use cases». [En línea]. Disponible en: https://www.ibm.com/blog/10-everyday-machine-learning-use-cases/

[2] «DashAI». [En línea]. Disponible en: https://www.dash-ai.com/

[3] Guido van Rossum y J. Fred L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. [En línea]. Disponible en: https://dl.acm.org/doi/book/10.5555/1593511

[4] «React». [En línea]. Disponible en: https://reactjs.org/

[5] «Apache Arrow». [En línea]. Disponible en: https://arrow.apache.org/

[6] S. Colvin *et al.*, «Pydantic». [En línea]. Disponible en: https://github.com/pydantic/pydantic

[7] T. Wolf *et al.*, «Transformers: State-of-the-Art Natural Language Processing». octubre de 2020. [En línea]. Disponible en: https://github.com/huggingface/transformers

[8] F. Pedregosa *et al.*, «Scikit-learn: Machine Learning in Python». pp. 2825-2830, 2011.

[9] Nick Hotz, «What is CRISP-DM?». [En línea]. Disponible en: https://www.datascience-pm.com/crisp-dm-2/

[10] «Git». [En línea]. Disponible en: https://git-scm.com/

[11] «GitHub Actions». [En línea]. Disponible en: https://docs.github.com/en/actions

[12] «Weka 3 - Data Mining Software in Java». [En línea]. Disponible en: https://www.cs.waikato.ac.nz/~ml/weka

[13] «RapidMiner». [En línea]. Disponible en: https://rapidminer.com/

[14] «KNIME Analytics Platform». [En línea]. Disponible en: https://www.knime.com/

[15] Jean-François Puget, «The Most Popular Language For Machine Learning Is…». 16 de marzo de 2017. [En línea]. Disponible en: https://medium.com/inside-machine-learning/the-most-popular-language-for-machine-learning-is-46e2084e851b/

[16] J. D. Hunter, «Matplotlib: A 2D graphics environment», n.º 3. IEEE Computer Society, pp. 90-95, 18 de junio de 2007. doi: 10.1109/MCSE.2007.55.

[17] M. L. Waskom, «Seaborn: statistical data visualization», n.º 6. abril de 2021. doi: 10.21105/joss.03021.

[18] «Plotly Open Source Graphing Libraries». [En línea]. Disponible en: https://plotly.com/graphing-libraries/

[19] «Tableau». [En línea]. Disponible en: https://www.tableau.com/

[20] «Power BI». [En línea]. Disponible en: https://powerbi.microsoft.com/

[21] «What is EDA?». [En línea]. Disponible en: https://www.ibm.com/think/topics/exploratory-data-analysis

[22] The pandas development team, «pandas». Zenodo, febrero de 2020. doi: 10.5281/zenodo.3509134.

[23] C. R. Harris *et al.*, «Array programming with NumPy». Springer Science and Business Media LLC, pp. 357-362, septiembre de 2020. doi: 10.1038/s41586-020-2649-2.

[24] Sebastián Ramírez, «FastAPI». [En línea]. Disponible en: https://github.com/fastapi/fastapi

[25] M. Bayer, «SQLAlchemy», *The Architecture of Open Source Applications*, vol. II. aosabook.org, 2012. [En línea]. Disponible en: https://aosabook.org/en/sqlalchemy.html

[26] «asyncio». [En línea]. Disponible en: https://docs.python.org/3/library/asyncio.html

[27] A. Clark, «Pillow (PIL Fork) Documentation». [En línea]. Disponible en: https://pillow.readthedocs.io/en/stable/

[28] Q. Lhoest *et al.*, «huggingface/datasets». doi: 10.5281/zenodo.4817768.

[29] N. Kruchten, A. Seier, y C. Parmer, «An interactive, open-source, and browser-based graphing library for Python». 12 de septiembre de 2024. doi: 10.5281/zenodo.14366348.

[30] «JavaScript (MDN)». [En línea]. Disponible en: https://developer.mozilla.org/en-US/docs/Web/JavaScript

[31] «TypeScript». [En línea]. Disponible en: https://www.typescriptlang.org/

[32] «Material-UI». [En línea]. Disponible en: https://material-ui.com/

[33] «Formik». [En línea]. Disponible en: https://formik.org/

[34] A. Johnson, É. Tétreault-Pinard, y M. Samimi, «Open source Plotly charting library». doi: 10.5281/zenodo.13963606.

[35] «Flake8». [En línea]. Disponible en: https://flake8.pycqa.org/en/latest/

[36] Ł. Langa y contributors to Black, «Black: The uncompromising Python code formatter». [En línea]. Disponible en: https://github.com/psf/black

[37] «ESLint». [En línea]. Disponible en: https://eslint.org/

[38] «Prettier». [En línea]. Disponible en: https://prettier.io/

[39] «GitHub». [En línea]. Disponible en: https://github.com/

[40] «pre-commit». [En línea]. Disponible en: https://pre-commit.com/

[41] «Figma». [En línea]. Disponible en: https://www.figma.com/

[42] A. C. Mueller, «Wordcloud». 27 de abril de 2023. [En línea]. Disponible en: https://github.com/amueller/wordcloud

[43] Stefan Aeberhard y M. Forina, «Wine Dataset». [En línea]. Disponible en: https://archive.ics.uci.edu/ml/datasets/wine

[44] John Brooke, «A 'Quick and Dirty' Usability Scale», *Usability Evaluation in Industry*, vol. 189. noviembre de 1995. [En línea]. Disponible en: https://www.researchgate.net/publication/228593520_SUS_A_quick_and_dirty_usability_scale/

[45] Jeff Sauro, «Measuring Usability with the System Usability Scale (SUS)». [En línea]. Disponible en: https://measuringu.com/sus/

[46] «Hugging Face Datasets: Features». [En línea]. Disponible en: https://huggingface.co/docs/datasets/main/en/package_reference/main_classes#datasets.Features