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

DISEÑO E IMPLEMENTACIÓN DE INTERFAZ DE USUARIO PARA SOFTWARE DE INTELIGENCIA ARTIFICIAL SIN PROGRAMAR

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

MAXIMILIANO ELÍAS AGUILAR SANHUEZA

PROFESOR GUÍA:
FELIPE BRAVO MÁRQUEZ

PROFESORA CO-GUÍA:
CLAUDIA LÓPEZ MONCADA

MIEMBROS DE LA COMISIÓN:
CLAUDIO GUTIÉRREZ GALLARDO
JOSÉ SAAVEDRA RONDO

SANTIAGO DE CHILE
2023

Resumen

DashAI es un proyecto de código abierto como parte de la iniciativa *OpenCENIA* que busca disminuir las barreras de entrada que tiene un área tan compleja como la inteligencia artificial mediante la generalización de las tareas (*tasks*). La extensibilidad del *software*, que permitirá adaptarse a los cambios. Una configuración flexible que se adapte al conocimiento de los usuarios. Y en lo que se enfoca el presente trabajo, una interfaz gráfica usable que permita a sus usuarios experimentar con la inteligencia artificial sin tener la necesidad de usar la programación.

El problema a resolver en este trabajo corresponde al diseño e implementación de una interfaz de usuario para *DashAI*, que sea usable y capaz de representar las ventajas de la herramienta. La interfaz diseñada y desarrollada cuenta con un diseño vertical y cuatro componentes principales que permiten a los usuarios: cargar datos, diseñar un experimento agregando modelos y configurando sus parámetros, visualizar resultados del experimento y comparar modelos entre sí, así como también probar modelos con un *input* decidido por el mismo usuario.

La interfaz diseñada e implementada se evalúa mediante la aplicación de una prueba de usabilidad cualitativa en la que han participado 5 usuarios y cuyo principal objetivo es encontrar errores en la interfaz. Para resumir la opinión de los usuarios se han tomado indicadores objetivos como eficacia y eficiencia. Además de indicadores sobre la percepción subjetiva de los usuarios a través del cuestionario *NASA-TLX*. Estos indicadores se utilizan para dar la conclusión de los participantes sobre la usabilidad de la herramienta.

En general, el balance de la evaluación es positivo y las opiniones de los usuarios que participaron en dicha prueba dan cuenta del gran potencial que tiene esta herramienta para mejorar y, en posteriores versiones, generar un impacto en el área de la inteligencia artificial.

A mi familia, quienes me han permitido llegar hasta aquí.

Agradecimientos

Me gustaría agradecer al equipo de *DashAI*, a mis profesores guía: Felipe y Claudia, que me dieron muy buenos consejos y conocimientos para poder concretar un proyecto en un área que nunca había desarrollado con la seriedad que requiere. A los desarrolladores: Cristian, Rodrigo, Ignacio y Oziel. Por la buena comunicación, la disposición a resolver dudas y a ayudar en la medida de lo posible a pesar de tener semestres pesados.

Y principalmente, agradecer a mi familia que me han permitido llegar a ser lo que soy hoy.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Contexto y problema

La inteligencia artificial (IA)[18], y en particular los modelos de aprendizaje automático (*machine learning*[1])[15], es un área que ocasiona un gran impacto en la sociedad. Esto puede evidenciarse por la influencia económica que tiene. Según un estudio de la empresa *PwC* para el año 2030 se espera que genere un aumento del 14 % del PIB mundial[17].

Estas proyecciones presentan grandes oportunidades que es necesario aprovechar. Sin embargo en Chile, de acuerdo a la XI encuesta nacional de innovación, tan solo un 14.1 % de las empresas ha realizado algún tipo de innovación[21]. Una de las principales razones de esto según las empresas encuestadas es la "falta de personal calificado". Es decir, existe una brecha entre el progreso tecnológico y el capital humano. Por esto mismo, el problema es planteado en la Política de Inteligencia Artificial de Chile, la cual tiene entre sus objetivos el desarrollo de talentos en IA[22].

Una de las principales razones de esta brecha en el capital humano con respecto a los avances, enfocándose en el área de IA, se debe a las grandes barreras de entrada que existen:

- Conocimientos de programación y matemáticas.
- Rápido avance, ya que constantemente surgen nuevos modelos o tareas que aplicar.
- El gran número y heterogeneidad de tareas (*tasks*) que existen. Entendiendo como *task* a un problema que se aborda a través de la inteligencia artificial, por ejemplo: clasificación de imágenes, generación de texto, traducción de texto, entre otras.

Para resolver estas barreras de entrada se ha creado *DashAI* una herramienta de *software* de

[1]Inteligencia artificial es la disciplina que crea sistemas que resuelven problemas que normalmente requieren de un humano. *Machine learning* es una subárea de la IA que construye sistemas que "aprenden" utilizando datos. Para evitar ambigüedades, se utiliza IA para referirse al área en general, mientras que ML para modelos que aplican este paradigma. Sus definiciones y la distinción entre ellos es profundizada en la sección 2.5.2 "inteligencia artificial y machine learning".

código abierto, la cual permite que usuarios, sin tener la necesidad de programar, puedan: configurar, entrenar, comparar y utilizar modelos de *machine learning* tanto tradicionales como del estado del arte para tareas como: clasificación de texto, clasificación numérica o traducción automática. *DashAI* está inspirado en otros *software* de IA sin programar como:

- *WEKA* (Waikato Environment for Knowledge Analysis)[5][3]
- *RapidMiner*[7]

Estos software permiten entrenar modelos de clasificación, regresión y *clustering* utilizando exclusivamente una interfaz gráfica, es decir, sin tener que programar. Pero los software anteriormente nombrados, con el paso del tiempo, no han sido capaces de adaptarse a las nuevas tendencias de los modelos del estado del arte.

Como se mencionó anteriormente *DashAI* está aún en desarrollo.Tiene actualmente una componente inicial de programación que fue el resultado del trabajo realizado en prácticas profesionales por estudiantes y que luego fue continuado en un trabajo dirigido. A pesar de esto aún queda mucho trabajo por realizar en el proyecto y es específicamente en la interfaz de usuario en lo que se centra el problema abordado como trabajo de título.

Para un software como *DashAI*, la interfaz de usuario es muy importante puesto que gran parte de su valor radica en la interacción con el usuario. El usuario objetivo de *DashAI* puede ser caracterizado como: analistas de datos, personas del mundo académico (científicos, estudiantes) que estén interesados en agilizar su experimentación con inteligencia artificial.

DashAI es un proyecto ambicioso que se proyecta a largo plazo, por lo que no se espera que este trabajo sea capaz de cumplir con todos los objetivos del proyecto. Por ejemplo, es conocida la necesidad de contar con inteligencia artificial explicable. Sin embargo, debido a la complejidad de agregarla se ha decidido dejar este tema como trabajo futuro y reducir los peligros de no entender los fundamentos de los modelos con recomendaciones como tutoriales, guías sobre las limitaciones de la plataforma, canales de comunicación eficiente con usuarios y el incentivo de su módulo para probar los modelos entrenados.

Así, se aborda como trabajo de título el desafío de diseñar e implementar una interfaz gráfica usable, que sea capaz de representar la mayor parte de las fortalezas con las que cuenta *DashAI* con respecto a otros *software*.

1.2. Relevancia

DashAI posee varias ventajas con respecto a otros *software* que tienen objetivos similares:

- *DashAI* está enfocado en *tasks* (tareas de inteligencia artificial) en vez de modelos. Esto hace que la herramienta sea independiente de la biblioteca utilizada y, con esto, que se pueda adaptar a los rápidos cambios en el área de IA.

- A través de su idea de "objetos configurables" permite a los usuarios una configuración flexible que se adapta a sus conocimientos.
- Es de código abierto.
- Otra de las ventajas a las que apunta *DashAI* es ser extensible, es decir, que los usuarios con habilidades de programación puedan crear sus propias tareas o modelos y utilizarlos en la herramienta.
- *DashAI* tiene como objetivo integrar bibliotecas científicas para explicar modelos y así facilitar la comprensión de sus fundamentos a los usuarios.

De estas ventajas la interfaz diseñada e implementada en el presente trabajo se plantea abordar todos ellos, con la excepción de explicabilidad, la cual se deja como trabajo futuro.

Se piensa que producir una primera versión de *DashAI* , aún sin cumplir todos sus objetivos, es relevante, en específico su interfaz de usuario. Pues esto podría dar un gran apoyo al proyecto mientras se está iniciando. Con una interfaz gráfica capaz de comunicar las fortalezas de la herramienta se podría presentar esta a potenciales instituciones interesadas en colaborar en el proyecto o realizar pruebas para verificar que es algo de interés para su público objetivo.

1.3. Objetivos

1.3.1. Objetivo General

La meta principal del trabajo es: **Diseñar e implementar una interfaz gráfica para interactuar con el usuario y que esta sea usable.**

1.3.2. Objetivos Específicos

Se enumeran en lo que sigue objetivos que apuntan hacia el objetivo general:

1. **Diseñar y crear prototipos con alternativas de diseño de interfaz**: Crear prototipos con diferentes formas de interactuar con el usuario. Estos prototipos incluirán diferentes alternativas para configurar modelos, presentar resultados, comparar modelos, etc.
2. **Validar prototipos**: Presentar prototipos al equipo de desarrollo y que estos proporcionen retroalimentación sobre ellos con el fin de encontrar las formas de presentar la información que sean más simples para cada tarea.
3. **Implementar la interfaz** De acuerdo a los prototipos seleccionados se implementarán en más detalle las interfaces en el *software*.

4. **Validación final con los usuarios**: Realizar una prueba de usabilidad con la interfaz implementada.

1.3.3. Evaluación

Como se ha mencionado anteriormente, para evaluar el trabajo se realiza una prueba de usabilidad cuyo principal objetivo es encontrar errores y posibles mejoras en la interfaz los cuales se resumen en datos cualitativos. Además, para resumir la opinión de los participantes se toman medidas cuantitativas.

La prueba de usabilidad recluta a usuarios a través del foro de la Universidad de Chile *UCursos*. A los usuarios se les pide que tengan experiencia en inteligencia artificial, por ejemplo, que hayan desarrollado proyectos en el área o trabajado como analistas de datos. Con esto, se espera que los participantes de la prueba sean capaces de dar un buen *feedback* y encontrar errores relevantes que ayuden a mejorar la herramienta y trazar objetivos a futuro.

Para medir usabilidad se utilizan indicadores usuales. En primer lugares se utilizan dos indicadores objetivos:

- **Eficacia**: se mide si los usuarios cumplieron el objetivo de cada una de las tareas asignadas, independiente del tiempo que les haya tomado.
- **Eficiencia**: para usuarios que completaron el objetivo de las tareas asignadas, se mide el tiempo que les tomó hacerlo.

A esto se agrega un indicador que mide la percepción subjetiva de los usuarios a través del cuestionario *NASA-TLX*.

1.4. Situación actual

DashAI es un proyecto que ya cuenta con ciertas componentes de código, principalmente de *backend*.

1. **API**: implementa variados *endpoint* que permiten una comunicación fluida con la interfaz gráfica.
2. **Tasks y modelos**: se cuenta con un conjunto básico de *tasks* y sus modelos asociados. Los cuales entregan una visión de las principales funcionalidades de *DashAI*. Entre estas *tasks* se encuentran: clasificación numérica, clasificación de texto y traducción automática.
3. **Interfaz gráfica**: Se cuenta con una componente muy básica de interfaz gráfica desarrollada utilizando *Dash*, que permite cargar datos y seleccionar modelos.

1.5. Descripción general de la solución

La interfaz gráfica diseñada e implementada consiste en una aplicación web que será ejecutada de manera local (en el equipo personal de los usuarios) y que ha sido desarrollada utilizando el *framework React.js*. Esta aplicación web se comunica con el *backend* a través de la *API* descrita anteriormente en la sección "Situación actual".

La interfaz cuenta con un diseño vertical, es decir, cada paso se despliega hacia abajo, y se ha diseñado para que siga los pasos usuales de un proceso de experimentación con modelos de *machine learning*, esto es: cargar datos, añadir y configurar modelos, visualizar resultados y finalmente probar los modelos viendo sus respuestas a determinados *input*.

Las cuatro componentes principales de la interfaz, y que corresponden a los "pasos" que se van desplegando hacia abajo, son las siguientes:

- **Carga de datos**: componente donde el usuario puede subir los datos con los que quiere trabajar.
- **Configurar experimento**: en esta componente se realizan todas las configuraciones necesarias para un experimento, es decir, añadir modelos que se quieran entrenar y configurar sus parámetros.
- **Ver resultados**: componente para visualizar los resultados del experimento y comparar los modelos entrenados.
- **Probar modelo**: componente para entregar *input* al modelo y que este devuelva una respuesta, por ejemplo, en una tarea de traducción se le entrega un texto en inglés y el modelo entrega la traducción al español.

1.6. Estructura del documento

Este documento se estructura en 6 capítulos (incluyendo el capítulo en el que se encuentra el lector).

- El capítulo 2, se dedica a dar una definición exhaustiva de todos los conceptos necesarios para entender correctamente el documento, estas definiciones contienen conceptos de inteligencia artificial, descripción de las tecnologías utilizadas, descripción de las soluciones existentes, entre otras.
- El capítulo 3 presenta una descripción detallada del problema, se establecen los requisitos que debe tener la solución y sus criterios de aceptación.
- El capítulo 4 describe la solución, la metodología utilizada, el proceso de diseño y varias capturas de pantalla que ilustran el flujo de la interfaz y sus funcionalidades.
- El capítulo 5 describe la validación de la solución, presenta los resultados de un test de usabilidad realizado usando tablas, gráficos y un diagrama de afinidad.

- Finalmente, el capítulo 6 expone las conclusiones de todo el trabajo realizado, los aprendizajes y el trabajo futuro.

Capítulo 2

Estado del Arte

En este capítulo se describen en detalle las soluciones existentes al problema abordado, junto con todos los conceptos relevantes para el correcto entendimiento de la solución propuesta, esto es: tecnologías consideradas, metodologías de desarrollo y definiciones.

2.1. Soluciones existentes

Como fue mencionado previamente existen otros software con los cuales es posible trabajar con modelos de *machine learning* sin la necesidad de programar. Dos ejemplos son la principal inspiración de *DashAI*:

- *WEKA*[5][3] *Waikato Environment for Knowledge Analysis*, en español entorno para análisis del conocimiento de la Universidad de Waikato. Es un *software* para *machine learning* y minería de datos escrito en el lenguajde de programación *JAVA*, desarrollado por la universidad de *Waikato* en 1993.
- *RapidMiner*[7] Es un *software* desarrollado por la universidad técnica de *Dortmund*. Posee una interfaz gráfica orientada a procesos y principalmente trabaja con datos tabulares.

Sin embargo, hay dos fuertes razones por las que ambos *software* han perdido popularidad con el tiempo, estas son:

1. **Están desarrollados en JAVA**: Dado que las herramientas nombradas fueron desarrolladas entre los años 1990 y 2000, tiene sentido que fuesen desarrolladas en *JAVA*. Sin embargo, en los tiempos modernos el campo de la inteligencia artificial se ha popularizado mayormente en el lenguaje de programación *Python*, en especial las herramientas de *Deep Learning*.
2. **No implementan modelos del estado del arte**: Ambos software están diseñados para resolver tareas principalmente de *clustering*, clasificación y regresión. Sin embargo,

hoy en día los requerimientos en inteligencia artificial van mucho más allá de estas tareas, por ejemplo requiriendo realizar traducción automática, procesamiento de audio o video, generación de texto, etc. Todas las tareas mencionadas anteriormente, no son posibles de realizar en los software nombrados, al menos en su estado base. En *WEKA* es posible realizar tareas más modernas como clasificación de imágenes o texto a través de "plugins". Sin embargo, estos son solo una extensión de las tareas principales por lo que, en realidad, siguen siendo esencialmente clasificación y regresión.

Después de *WEKA* y *RapidMiner* se han desarrollado otras alternativas para software de inteligencia artificial que no requieran conocimientos de programación, algunos de estos son:

- ***Perceptilabs***[1] (2018): Permite construir modelos de machine learning del estado del arte integrándose con librerías como *Tensorflow* y *HuggingFace*. Todo esto usando una interfaz "drag and drop" que lo hace amigable para usuarios sin experiencia en programación.
- ***Akkio***[2] (2019): Personas sin conocimientos de programación pueden utilizar *Akkio* para construir modelos de *machine learning* gracias a su interfaz "wizard-based" (guía al usuario a través de ventanas de diálogo).

Ahora bien, existen también otros *software* que han decidido llevar el objetivo de disminuir las barreras de entrada y así democratizar la inteligencia artificial un paso más allá, en este caso no solo no se necesita saber programar para utilizarlos, sino que tampoco es necesario tener experiencia previa en inteligencia artificial. Estos software son denominados "AutoML" y, aunque su objetivo no está completamente alineado con el de *DashAI*, el estudiar como estos abordan el problema de democratizar la inteligencia artificial es un caso digno de estudio. Algunos de estos software son:

- ***H2O***[3]: Su plataforma *H2O Driverless AI* permite agilizar mucho el proceso completo de un proyecto de ciencia de datos.
- ***KNIME***[4]:Agiliza mucho los proyectos de ciencia de datos, proporcionando una amigable interfaz "drag and drop".

2.2. Interfaces de usuario

En la sección anterior se ha tratado el tema de los *software* para IA sin programar en su totalidad. Sin embargo, no se ha analizado adecuadamente la componente principal para el presente trabajo de título: la interfaz de usuario. Se expondrá entonces, en lo siguiente, algunas interfaces de usuario de los principales *software* para IA sin programar.

[1] www.perceptilabs.com
[2] www.akkio.com
[3] H2O Driverless AI
[4] www.knime.com

2.2.1. WEKA

WEKA es la principal inspiración para *DashAI*, por lo que es natural que se muestre y se tomen algunas ideas de su interfaz de usuario.

Utilizando *WEKA* (en su versión 3.9.6), la siguiente imagen muestra lo primero que aparece al ejecutar el software:

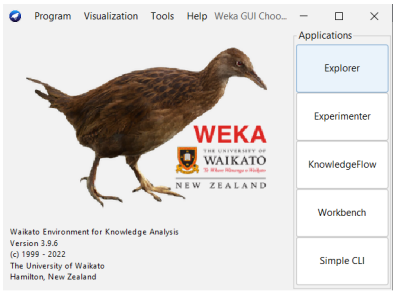

Figura 2.1: Interfaz inicial que se muestra al ejecutar *WEKA*.

Seleccionando "Experimenter", pues es la parte del *software* que se asemeja más a la funcionalidad que se quiere para *DashAI*, se muestra la interfaz para configurar un experimento, esto es: seleccionar *dataset* y modelos para entrenar. En el ejemplo de la imagen siguiente se muestra que se decide entrenar "Naive Bayes" y "Random Forest" sobre "Iris Dataset".

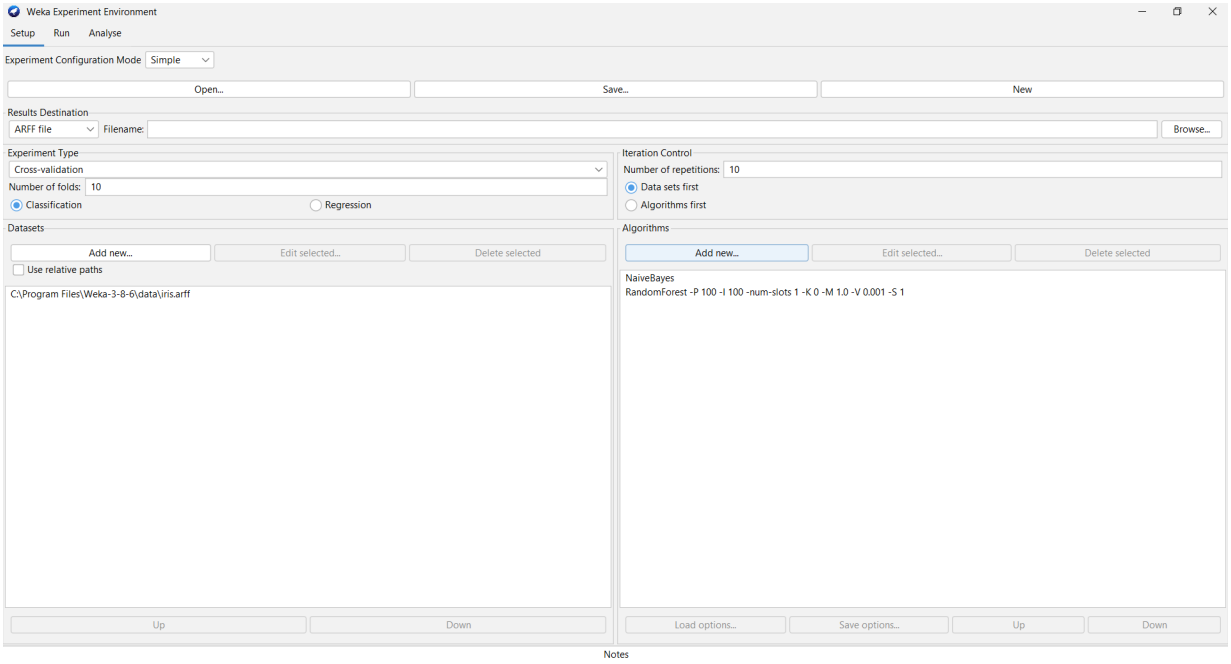

Figura 2.2: Interfaz que se muestra al seleccionar "Experimenter" desde la interfaz inicial, configurado para datos de "Iris Dataset" y modelos "Naive Bayes" y "Random Forest".

Ahora seleccionando la pestaña "Run" y ejecutando el experimento se muestra la siguiente *tab* con información sobre el estado del experimento:

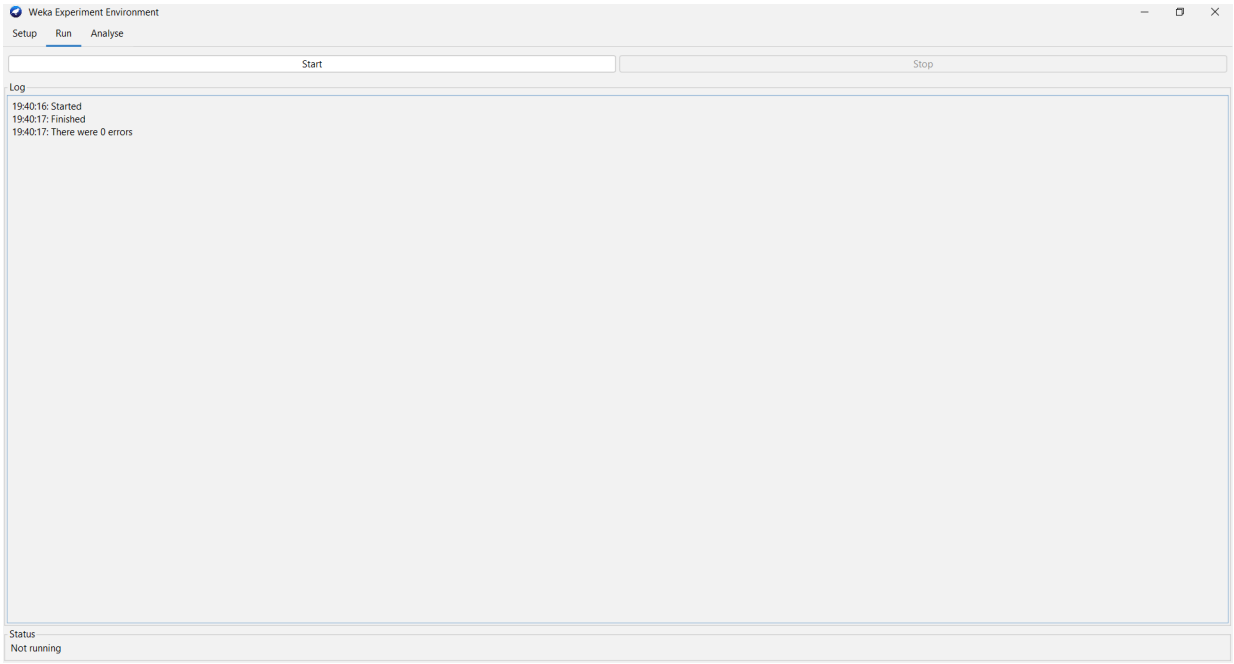

Figura 2.3: Interfaz que se muestra al seleccionar “Run” y presionar el botón “Start”.

Finalmente, seleccionando la pestaña “Analyze” se muestra un resumen con los resultados del experimento:

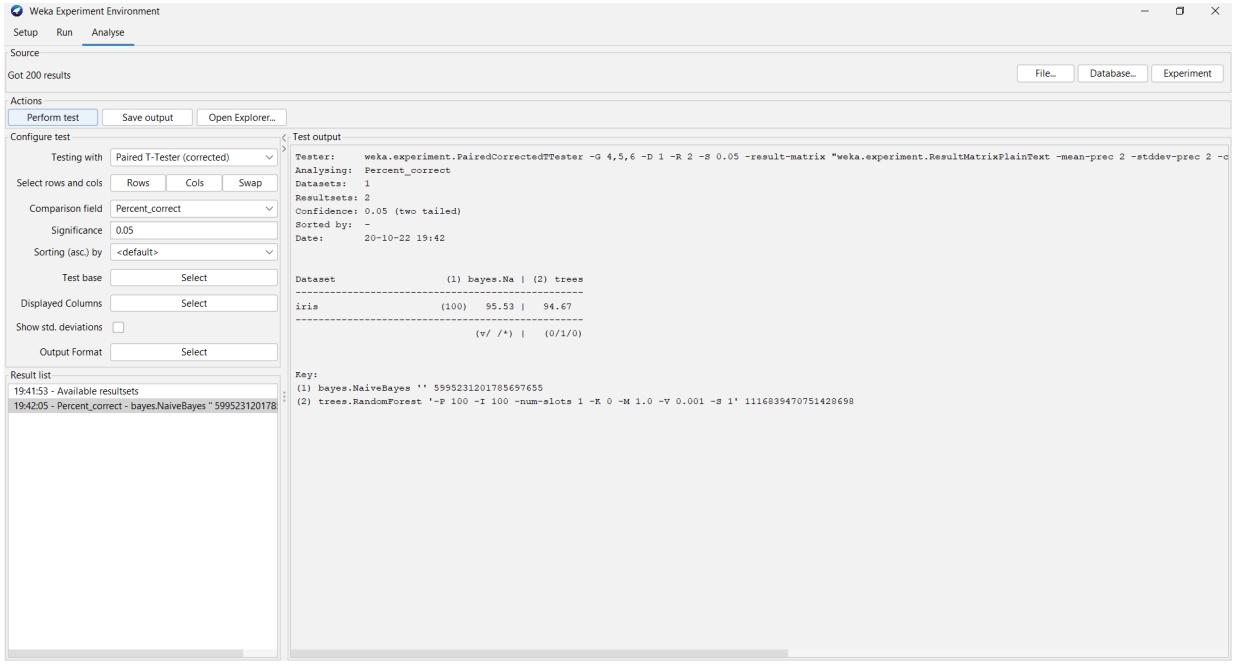

Figura 2.4: Interfaz que se muestra al seleccionar la pestaña “Analyze”, presionar el botón “Experiment” y luego presionar el botón “Perform test”.

Considerando las imágenes anteriores, una de las primeras impresiones que se puede tener al ver la interfaz de *WEKA* es que es compleja, sobre todo si no se tiene demasiada expe-

riencia entrenando modelos de *machine learning*. Una de las principales razones por las que puede ser compleja es debido a la gran cantidad de opciones de configuración disponibles mostrándose todas a la vez, esto se puede apreciar especialmente en la Figura 2.2.

Es una interfaz útil, pero demasiado compleja y que podría incluso representar un problema para personas que no tienen mucho conocimiento de inteligencia artificial y utilizan *WEKA* para intentarlo por primera vez.

2.2.2. Interfaces modernas

Se vio que el principal problema de la interfaz de *WEKA* es que es compleja. Sin embargo, en secciones anteriores también se mencionaron otros software más modernos que cumplen con el objetivo de ser usados por personas sin conocimientos de programación.

Para resolver el problema de dismunuir la complejidad, los software modernos han elegido dos caminos[5]: interfaces "drag and drop" en las que se puede arrastrar componentes de modelos como, por ejemplo, capas de redes neuronales para conectarse entre sí y obtener un resultado. Un ejemplo de esto es el *software Perceptilabs*.

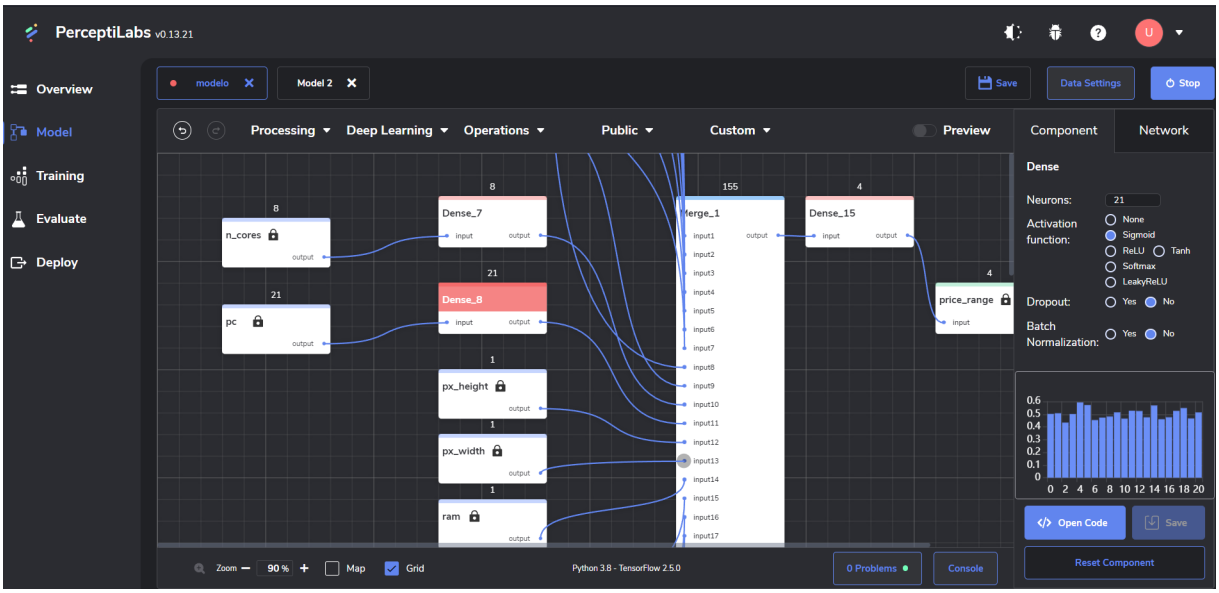

Figura 2.5: Perceptilabs, ejemplo de red neuronal para predecir el precio de smartphones basado en sus características.

Otra forma de resolver el problema de la complejidad corresponde a las interfaces de tipo "wizard-based" que utilizan ventanas de diálogo para guiar al usuario y que este sepa claramente lo que debe hacer en cada momento. Un ejemplo de esto es el *software*: *Akkio*.

[5]https://towardsdatascience.com/no-code-ai-platforms-bring-ai-to-everyone-here-is-how-8f75b2f6ce9d

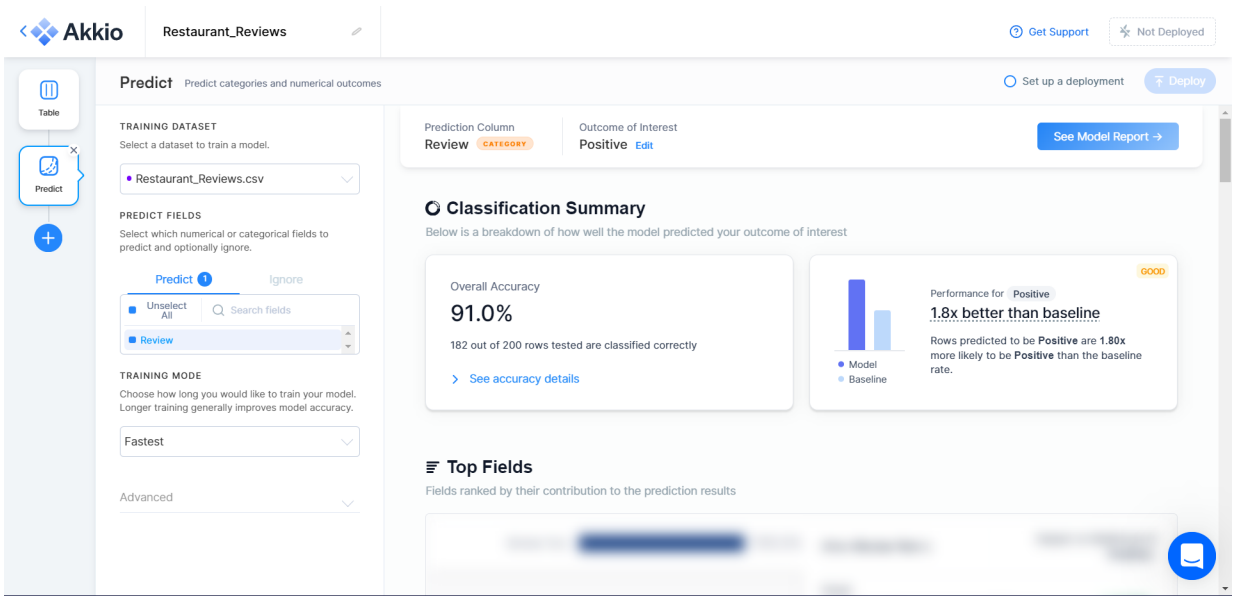

Figura 2.6: Akkio, clasificación de sentimientos en reviews de restaurantes.

Estos tipos de interfaces cumplen con ser menos complejos, lo suficiente para ser abordables por alguien que no tenga demasiada experiencia en inteligencia artificial, con lo cual pueden ser usadas como referencia para el presente trabajo de título. Pero se debe ser cuidadoso para adaptarlas correctamente a los requerimientos de *DashAI*

2.2.3. Interfaces de *AutoML*

Aunque las interfaces de *AutoML* podrían incluirse dentro de la sección "interfaces modernas", se ha decidido realizar una sección aparte para mostrar la interfaz de un software en particular que es bastante interesante.

H2O Flow[6] cuenta con una interfaz de "celdas" (algo parecido a *Jupyter Notebook*[7]) con las cuales se puede armar un flujo de trabajo. Por ejemplo, añadir una celda para subir un set de datos, otra celda para pre-procesarlos, otra celda para ejecutar modelos automáticamente y una última celda para ver los resultados de estos.

[6]https://docs.h2o.ai/h2o/latest-stable/h2o-docs/flow.html
[7]https://jupyter.org

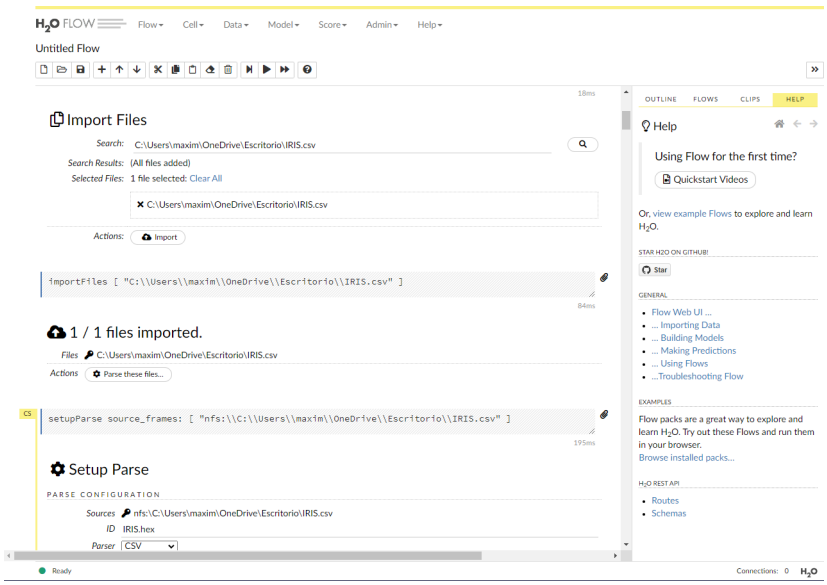

Figura 2.7: Ejemplo de "celdas" en H2O Flow. Se puede observar la celda *Import Files* para entregar la url de un set de datos, *files imported* para importar el set de datos y *Setup Parse* para configurar la forma de interpretar los datos previamente importados.

Como se puede ver en la figura (2.7) estas celdas consisten en pequeños formularios que el usuario debe llenar y cuyos "output" pueden ser usados también por otras celdas. Esta interfaz es interesante, puesto que es simple, ya que permite que el usuario arme su propio flujo de trabajo con celdas, las cuales son fácilmente reemplazables en caso de errores o cambios.

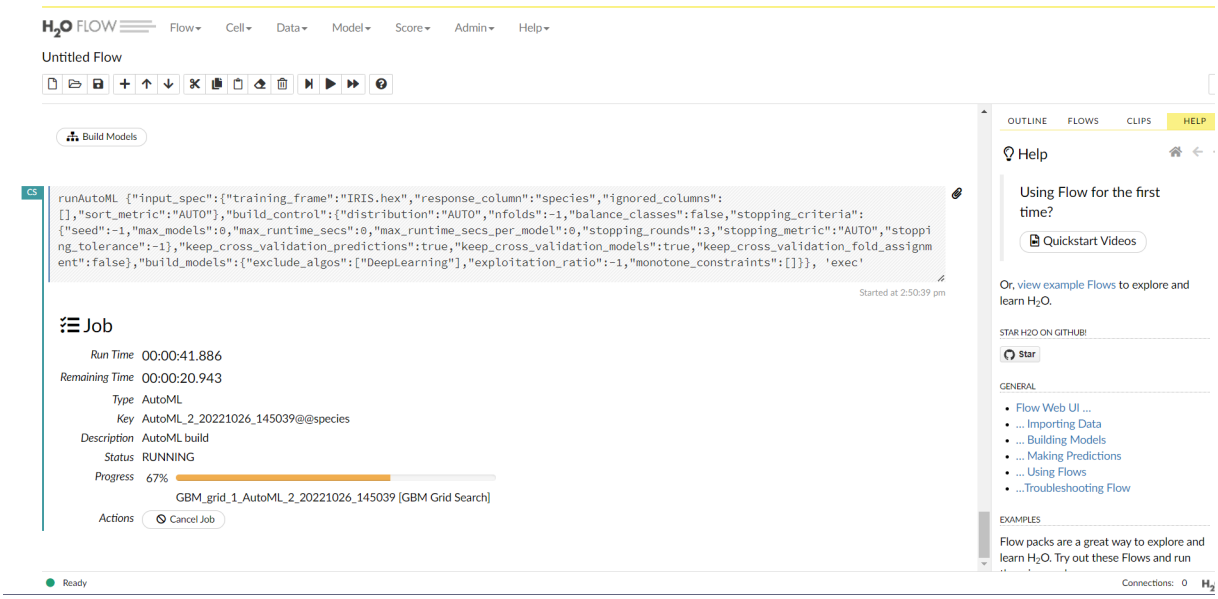

Figura 2.8: Ejemplo de *AutoML* en *H2O Flow*, existe un formulario en el que se pueden configurar parámetros como: columna para usar como *target*, o algoritmos a excluir, pero la mayoría de las decisiones son automáticas y se puede obtener resultados con poca interacción.

Aunque el software *H2O Flow* no está completamente alineado con los objetivos de *DashAI*, este presenta una interfaz interesante de la cual se pueden obtener ideas.

2.3. Metodologías de desarrollo

Uno de los problemas más importantes en lo que respecta al desarrollo de *software* es la forma en la que se organiza el trabajo a realizar, es por esto que la solución a este problema, es decir, las técnicas y métodos organizativos para diseñar una solución de *software* son también muy importantes. Estas se conocen como metodologías de desarrollo de *software*.

Existen una gran variedad de metodologías de desarrollo de software, las cuales pueden adaptarse también a una gran cantidad de equipos o situaciones. Se nombrarán brevemente algunas de las más características:

- **Waterfall (cascada)**: tal como su nombre indica, el desarrollo se divide en fases, las que se implementan de "arriba hacia abajo". Para avanzar a la fase siguiente se debe revisar el producto de la fase actual. Tiene como contra el que es muy "lenta" debido a que para ver resultados es necesario estar en las fases finales del desarrollo.
- **Incremental**: se construye el producto final de manera progresiva, donde en cada etapa se le agrega una nueva funcionalidad, debido a esto es capaz de producir resultados más rápidos, produciendo software utilizable desde sus etapas tempranas.
- **SCRUM**: se encuentra dentro de la clasificación de metodologías ágiles, es muy utilizada en equipos pequeños, divide el trabajo en *sprints* con un periodo de tiempo fijo, contempla también además reuniones diarias (daily meetings) para resumir brevemente el trabajo de cada integrante del equipo.

Habiendo realizado una introducción al concepto de metodologías de desarrollo de software llega el momento de exponer la metodología usada para el desarrollo de la solución presentada, esta corresponde a *Shape up*[19].

Shape up es una metodología que toma los beneficios de las metodologías de desarrollo ágiles (rápidos resultados, flexibilidad) y resuelve sus más grandes problemas los cuales son:

- **Ineficiencias de diseño**: planear todos los requisitos de una iteración es una tarea complicada y que requiere mucho esfuerzo. Por ello, esta tarea es desempeñada normalmente por los desarrolladores con más experiencia los cuales deben invertir gran parte de su tiempo en diseñarlas. Por otra parte, los desarrolladores con menos experiencia deben acatar estas decisiones y seguirlas lo mejor que puedan, sin poder utilizar sus conocimientos para aportar activamente en el diseño.
- **Rapidez de resultados**: se presentó anteriormente como ventaja la rapidez de los resultados en las metodologías ágiles, pero un análisis más detallado arroja que esta rapidez tiene también una debilidad. Una vez se obtiene un resultado no existe el

adecuado tiempo para reflexionar sobre este y si es realmente lo que se necesita o se quiere, pues la siguiente iteración debe comenzar.

Para dar solución a estos problemas se ha ideado *Shape up* que puede resumirse brevemente en sus cuatro ideas clave:

- **Ciclos de seis semanas**: se establecen seis semanas como un periodo lo suficientemente largo como para desarrollar un avance importante, pero a la vez lo suficientemente corto para sentir cierta presión por la *deadline*. En este periodo se le da libertad al equipo para distribuir su tiempo sin *micro-management*, es decir, no hay reuniones diarias ni tampoco se cuentan las horas dedicadas al desarrollo.
- **Diseño del trabajo (Shaping the work)**: también se realiza una etapa para planear la solución hecha por desarrolladores con más experiencia. Sin embargo, en *Shape up* se establecen solo los **objetivos clave** de manera general, dando libertad a los equipos de desarrollo encargados de llevar a cabo estos objetivos para diseñar ellos mismos los detalles de implementación. Para cada objetivo se establece el tiempo que se quiere pasar trabajando en este, al cual se le llama **appetite** (apetito).
- **Hacer a los equipos responsables**: como se ha enfatizado antes, los equipos tienen más libertades, ellos son quienes definen sus propias tareas y los alcances de estas, siguiendo obviamente los objetivos clave establecidos. De esta forma los desarrolladores con más experiencia también tienen más tiempo para dedicar al diseño del proyecto, tiempo que usando otras metodologías se dedica en manejar a los equipos.
- **Considera el riesgo**: una parte del diseño de la solución corresponde a considerar el riesgo de no entregar a tiempo, para tratar de evitar esto se identifican los aspectos que, en un principio no lo aparentan, pero que podrían tomar mucho tiempo de implementar ("rabbit holes"). Esto combinado con el "apetito" de cada tarea y un periodo de "cooldown" (1-2 semanas) entre ciclos para repensar la solución entregada permite bajar los múltiples riesgos del proyecto.

Tomando en cuenta estas cuatro ideas clave el equipo, que está dirigido por los desarrolladores con mayor experiencia, establece los objetivos clave, los riesgos y el tiempo para dedicar a cada objetivo, los cuales se resumen por escrito en un *Pitch*.

Una vez hecho el *Pitch*, el equipo comienza el desarrollo en un ciclo de seis semanas (que también puede cambiarse a un tiempo menor dependiendo del caso) y cuando este termina pasa a un periodo de *cooldown* para que el equipo pueda repensar la solución, hacer ajustes necesarios e idear el *pitch* siguiente.

2.4. Tecnologías consideradas

2.4.1. Desarrollo web

El desarrollo web es un término para referirse a la creación de páginas para la *web*. La forma de interactuar con estas páginas es a través de un navegador web, el cual las interpreta y muestra a los usuarios una interfaz gráfica para que estos puedan interactuar. Un navegador es capaz de acceder a páginas a través de internet o bien puede acceder a páginas locales en la máquina en que se ejecuta.

En este contexto, el desarrollo web se puede dividir en dos grandes áreas: *Frontend* y *Backend*.

Frontend

Se refiere a la interfaz gráfica con la que interactúa un usuario a través de un navegador. Las tecnologías más importantes en esta área son:

- *HTML* (HyperText Markup Language o lenguaje de marcado de hipertexto), como lenguaje interpretable por los navegadores.
- *CSS* (Cascading Style Sheets u hojas de estilo en cascada) para proporcionar colores y diseños llamativos a las interfaces gráficas.
- *Javascript* como lenguaje de programación para proveer de interacción, eventos y procesos.

Estas tres tecnologías son suficientes para desarrollar una interfaz interpretable por navegadores web. Sin embargo, para estructurar y estandarizar el desarrollo normalmente se utilizan *Framework* (marco de trabajo), en este caso, los *framework* relevantes para entender el presente trabajo son los siguientes:

- **Dash**[8]: Es un *framework* para el lenguaje de programación *Python* que permite crear aplicaciones web de forma fácil y rápida, enfocándose principalmente en aplicaciones de ciencia de datos. Es decir, facilita crear páginas web que muestren gráficos y tablas interactivas.
- **React.js**[9]: Es un *framework* que se manifiesta como una biblioteca de código abierto para el lenguaje de programación *Javascript*. Se basa en el uso de componentes, los cuales, en forma de bloques y en conjunto con sus *Hooks*[10] son capaces de construir una aplicación con datos que están en constante cambio. Es uno de los *framework* más utilizados para *frontend* por lo que cuenta con gran cantidad de material en forma de soporte, tutoriales y buenas prácticas.

[8] https://dash.plotly.com
[9] https://reactjs.org/docs/getting-started.html
[10] https://reactjs.org/docs/hooks-intro.html

Backend

Se refiere a la lógica de la aplicación web, su conexión con bases de datos (en caso de haberlas) y el manejo del servidor utilizado. Como en *frontend*, existen también *framework* para estructurar el desarrollo de esta componente.

La tecnología relevante a presentar corresponde a *FastAPI*[11], el cual es un *framework* para el lenguaje de programación *Python* que, como su nombre indica, se utiliza para construir *API* (application programming interface o interfaz de programación de aplicaciones), que es simplemente la forma en que dos o más programas se comunican entre sí. En este caso se comunica la capa de lógica, es decir, backend con la interfaz gráfica, es decir, frontend. Este *framework* es ampliamente utilizado y, en particular, ha sido reconocido como de gran utilidad para aplicaciones de ciencia de datos[12]

2.4.2. Inteligencia artificial

Se dará una breve descripción de las tecnologías necesarias para comprender el presente trabajo, la introducción a los conceptos de inteligencia artificial se deja para una sección posterior.

- **Scikit-learn**[13]: Es una biblioteca de código abierto para *machine learning* en el lenguaje de programación *Python*, incluye algoritmos de clasificación, regresión, clustering (análisis de grupos). Su principal fortaleza corresponde a su fácil integración con otras herramientas en *Python* ya sea para manejo de datos, graficar datos, etc.
- **Pytorch**[14]: Es una biblioteca de código abierto para el lenguaje de programación *Python* enfocada principalmente en *Deep Learning*. Es una de las bibliotecas de *Deep Learning* más utilizadas debido a su flexibilidad y poder de cómputo.
- **Hugging Face**[15]: corresponde a una empresa cuya misión es democratizar la inteligencia artificial, cuenta con un sitio web y una comunidad. Además tiene una biblioteca para el lenguaje de programación *Python* con gran cantidad de modelos, los cuales pueden usarse fácilmente al ser de código abierto.

[11] https://fastapi.tiangolo.com
[12] https://towardsdatascience.com/you-should-start-using-fastapi-now-7efb280fec02
[13] https://scikit-learn.org/stable/index.html
[14] https://pytorch.org
[15] https://huggingface.co

2.5. Conocimientos clave

2.5.1. Diseño UX/UI

El diseño UX/UI es la disciplina que se encarga de crear productos digitales que sean fáciles de usar y agradables a la vista esto incluye:

- Experiencia de usuario (UX, por sus siglas en inglés)
- La interfaz de usuario (UI, también por sus siglas en inglés)

Ambos conceptos son esenciales para cualquier herramienta que pretenda tener interacción con sus usuarios, esto es, para asegurarse de que el producto final sea simple, fácil de usar y atractivo visualmente.

Para ello se establecen ciertos principios que una vez que se conocen parecen de sentido común, sin embargo, son bastante fáciles de ignorar, por ejemplo, utilizar jerarquías visuales entre los elementos de la interfaz, dejar los conceptos más importantes en letras más grandes, etc.[11]

2.5.2. Inteligencia artificial y *machine learning*

La inteligencia artificial[18] es una disciplina que se preocupa no solo por entender, sino también por construir sistemas inteligentes, es decir, que pueden calcular cómo actuar de manera efectiva y segura en una amplia variedad de situaciones nuevas.

Por otra parte, *machine learning*[15][23] o aprendizaje automático, se refiere al proceso mediante el cual un programa de computador adquiere habilidades para realizar tareas específicas, mejorando su rendimiento a medida que se expone a más datos o experiencias. A través del aprendizaje automático, el programa es capaz de mejorar su desempeño en tareas específicas sin ser programado explícitamente para ello. El *machine learning* puede considerarse como una subárea de la inteligencia artificial.

Ambos conceptos pueden relacionarse de la siguiente forma: IA es el concepto de que sistemas puedan realizar ciertas tareas que normalmente requieren de un ser humano, mientras que *machine learning* es un método específico para lograr inteligencia artificial, este método consiste en entrenar sistemas usando datos.

Teniendo en cuenta lo expuesto en los párrafos anteriores, y para evitar ambigüedades, en el presente informe se utiliza el término *inteligencia artificial* (IA) para referirse a la disciplina a nivel general. Mientras que se utiliza *machine learning* (ML) para referirse a modelos que se encuentran diseñados con este paradigma.

En lo siguiente la descripción se enfoca en subáreas de la IA y en modelos concretos de

machine learning, todos ellos serán necesarios para comprender la solución implementada en el presente informe.

Procesamiento de lenguaje natural

El procesamiento de lenguaje natural (NLP, por sus siglas en inglés) como subárea de la IA, corresponde a un campo que se dedica a diseñar métodos y algoritmos que toman como *input* o producen como *output* datos en forma de lenguaje humano. Estas tareas pueden ser por ejemplo, traducción automática, extracción de información, reconocimiento de entidades nombradas (*named-entity recognition*), entre otras.

Algo fundamental sobre el procesamiento de lenguaje natural para entender lo expuesto más adelante en este trabajo es que es posible aplicar modelos que trabajan con números para tareas que involucren texto mediante la "conversión" de la información contenida en el texto a información numérica, una forma de hacer esto es utilizar el método *Bag of words* (bolsa de palabras) que consiste en tomar documentos (secuencias de palabras) y contar las apariciones de cada palabra distinta en este, una vez que se tenga esto se pueden calcular métricas como la cantidad de apariciones de cada palabra (lo cual es información numérica). [8]

Algunos modelos de *machine learning* relevantes

Se presenta a continuación una lista junto con descripciones de los modelos de *machine learning* que son utilizados en la solución implementada y que, por lo tanto, es relevante conocer en el contexto del presente trabajo.

- **KNN**: *K-nearest neighbors* o k-vecinos más cercanos es un método de clasificación de datos numéricos que estima la probabilidad de que un elemento x pertenezca a la clase C utilizando otras observaciones que sean "cercanas" a x ("cercanas" de acuerdo a una métrica la cual puede ser, por ejemplo, la distancia euclidiana).
- **SVM**: *Support Vector Machines* o máquinas de vectores de soporte es un algoritmo de clasificación y regresión que trabaja con datos numéricos. En el caso de la clasificación, este modelo es capaz de buscar una recta (o hiperplano dependiendo de las dimensiones con las que se trabaje) que sea capaz de separar los datos en diferentes clases de manera óptima.
- **Random forest**: es un algoritmo que se utiliza en problemas de clasificación o regresión para números, su funcionamiento consiste en combinar muchos árboles de decisión (otro modelo que clasifica utilizando un conjunto de reglas que son construidas utilizando un conjunto de datos, cada "nodo" del árbol representa una de estas reglas frente a la cual se debe tomar una decisión que influye en la clasificación final del elemento). Luego, para hacer una predicción se toma una "votación" entre todos los árboles de decisión.

- **Modelo "Helsinki-NLP/opus-mt-en-es"**[16]: es un *transformer*[17] (un tipo de red neuronal que aprende el contexto y el significado de datos secuenciales como, por ejemplo, las palabras en una oración) de *Huggingface*. En específico este modelo se encuentra entrenado para trabajar en traducción de texto inglés-español. El que se encuentre entrenado quiere decir que es posible descargar el modelo y utilizarlo para tareas de traducción. Pero además, también es posible realizar entrenamiento adicional para que mejore en tareas más específicas (*fine tuning*).

2.5.3. Interactive machine learning

Se ha mencionado anteriormente la existencia de herramientas de software que juntan el área de IA con el área de *HCI* al producir interfaces gráficas para realizar tareas de IA. Es por esto que no es de extrañar que exista un área de investigación que aborda este problema: *Interactive Machine Learning*[1], en donde se estudian métodos para que los humanos seamos capaces de aplicar la percepción e inteligencia, sobre modelos matemáticos (*machine learning*) a través de ciclos iterativos donde se entrega un input y se analizan los resultados. De esta forma se puede usar el conocimiento generado en *Interactive Machine Learning* con el objetivo de guiar el diseño de interfaces para aplicaciones de *machine learning*, específicamente las que están destinadas a usuarios sin demasiado conocimiento del área.

Otro concepto relevante son los sistemas de *AutoML*. Estos sistemas van un paso más allá con el objetivo de democratizar la IA, pues permiten que incluso usuarios sin conocimientos técnicos sean capaces de crear y entrenar modelos de forma rápida y sencilla, esto se logra automatizando las configuraciones como la elección del modelo o el ajuste de hiperparámetros y , con esto, permiten que los usuarios se centren mayoritariamente en la interpretación y en utilizar los resultados obtenidos.

De acuerdo a lo expuesto anteriormente, las interfaces de herramientas de *AutoML* son un objeto de estudio interesante para obtener formas de guiar a los usuarios durante el desarrollo de proyectos de IA. Por esto mismo, existen estudios sobre todo desde el área de *HCI* que establecen patrones, lineamientos y dan recomendaciones sobre como construir estas interfaces[9].

2.5.4. Test de usabilidad

Los test de usabilidad son entrevistas estructuradas que se enfocan en funcionalidades específicas en un prototipo de interfaz. Se utilizan para revelar información sobre la forma en que la gente usa una interfaz, prototipo o *mockup*.

Consisten en una serie de tareas que son realizadas por los evaluadores de la interfaz (que típicamente pertenecen al público objetivo). A partir de esto, se recolectan datos como los errores, aciertos, malentendidos o críticas. Estos datos se almacenan en forma de grabaciones

[16]https://huggingface.co/Helsinki-NLP/opus-mt-en-es
[17]https://blogs.nvidia.com/blog/2022/03/25/what-is-a-transformer-model/

o notas que posteriormente son evaluados y utilizados como retroalimentación por el equipo de desarrollo.

Para realizar un test de usabilidad se debe ser muy cuidadoso con el perfil de la gente que se evalúa, pues su perfil debe estar alineado con el público objetivo. También se debe ser cuidadoso con la cantidad de personas que participan, pues esta debe suficiente para asegurar una evaluación relevante de la interfaz y asegurar retroalimentación de calidad para el equipo de desarrollo.

Los test de usabilidad se pueden dividir en dos tipos[18]:

- **Cuantitativos**: son un tipo de evaluación sumativa, es decir, que se utiliza para evaluar un producto en su estado final. Por esto, este tipo de test es difícil de realizar, se necesita un reclutamiento sistemático que represente realmente al público objetivo, recolectar diversas métricas del desempeño de los usuarios y utilizar estadísticas para resumir estos resultados.
- **Cualitativos**: son un tipo de evaluación que generalmente es formativa. Esto quiere decir que se utiliza para evaluar el estado actual de un proyecto en desarrollo, detectar las cosas que funcionan y detectar tempranamente sus errores más críticos. Son más simples de realizar y requieren menos recursos, puesto que necesitan un menor número de participantes (normalmente 5). Es importante destacar que este tipo de estudios no buscan generalizar, su principal objetivo es encontrar errores en el *software* que se evalúa. Es por esto, que no usan estadísticas para "predecir" métricas sobre los usuarios (por ejemplo, formar intervalos de confianza sobre el tiempo que le toma al usuario objetivo completar una tarea).

¿Cuántos usuarios se necesitan para un test de usabilidad?

Es una pregunta compleja para la cual existen varias recomendaciones. Si se realiza un estudio de tipo cuantitativo y, por lo tanto, se requiere utilizar estadística, se requiere una cantidad representativa que normalmente se encuentra entre las 20 personas[19].

Para los estudios cualitativos no es tan sencillo, puesto que al ser estudios más simples los beneficios de aumentar la cantidad de participantes es superado por los costos. Para resolver este problema existe la "Suposición de los 5 usuarios" (*Five-user assumption*)[20], la cual postula un modelo matemático que predice que con 5 usuarios se pueden encontrar el 85 % de los problemas de la interfaz que se está evaluando.

La "suposición de los 5 usuarios" ha sido ampliamente criticada desde que fue postulada debido a que se han utilizado modelos matemáticos más sofisticados y estudios estadísticos más complejos. Por ejemplo, un estudio más reciente[2] que realiza un test de usabilidad con 60 personas, para posteriormente dividirlas en varios grupos de 5. Se comprueba que el grupo

[18]https://www.nngroup.com/articles/5-test-users-qual-quant/
[19]https://www.nngroup.com/articles/quantitative-studies-how-many-users/
[20]https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/

de 5 personas con el peor desempeño fue capaz de encontrar el 55 % de los errores. Mientras que de media estos grupos encontraron alrededor del 85 % de los errores.

Entonces, para un estudio cualitativo la respuesta sobre la cantidad de usuarios adecuada es que esta depende y debe adecuarse al contexto de cada proyecto considerando sus alcances, su estado de desarrollo y los recursos de los que se disponga.

Metodología para un test de usabilidad

Se describe, en lo que sigue, un método particular para realizar tests de usabilidad[12], este método será aplicado para evaluar la solución desarrollada en este trabajo.

El método para realizar tests de usabilidad requiere pocos usuarios (5 es el mínimo siguiendo la "suposición de los 5 usuarios"), y está orientado mayoritariamente a recoger información cualitativa (es decir, es un estudio de tipo cualitativo). Se enfoca principalmente en resolver tareas con el usuario y luego tener una entrevista para recibir retroalimentación adicional. Los pasos de este método para test de usabilidad son los siguientes.

- Introducción: se presenta el contexto, es una etapa para "romper el hielo" con el usuario y que este pueda proporcionar respuestas más sinceras.
- Entrevista preliminar: preguntas para caracterizar al usuario.
- Tareas: actividades con la herramienta que se quiere evaluar.
- Entrevista final: preguntas sobre la experiencia con la herramienta que permiten obtener retroalimentación adicional sobre ciertos puntos que interesen al entrevistador.

Se puede analizar las tareas utilizando variables cuantitativas, pero debido a la baja cantidad de usuarios esto debe tomarse como una conclusión solo respecto al grupo entrevistado y no a una generalización.

En resumen, el objetivo de esta metodología es encontrar errores en el *software* para poder abordarlos tempranamente y no generalizar sobre el grupo de usuarios objetivo.

2.5.5. Diagramas de afinidad

Un diagrama de afinidad es una herramienta utilizada en el proceso de análisis y síntesis de datos cualitativos. Se utiliza para organizar y agrupar información de manera visual y estructurada. Esto permite identificar patrones y tendencias en los datos recopilados.

Los diagramas de afinidad se componen de tarjetas o *post it* con anotaciones o ideas, que posteriormente se agrupan en categorías o temas similares, estos grupos reciben el nombre de grupos afines[13].

2.5.6. Cuestionario: NASA-TLX

El cuestionario NASA Task Load Index o NASA-TLX[6] es un método para valorar la carga mental de una experiencia concreta. Este cuestionario se basa en asignar un puntaje entre 1 y 20 para 6 categorías las cuales son:

- Exigencia mental.
- Exigencia física.
- Exigencia temporal.
- Rendimiento.
- Esfuerzo.
- Frustración.

El hecho de que sean 6 categorías permite aliviar la carga mental del usuario al evaluar la experiencia.

Capítulo 3

Problema

Se ha mencionado en secciones anteriores que el objetivo del presente trabajo corresponde al diseño e implementación de una interfaz de usuario que permita realizar tareas de inteligencia artificial sin tener la necesidad de programar. Pues bien, en esta sección se da una descripción más detallada del problema al que da solución dicha interfaz, incluyendo los requisitos que esta debe cumplir y también sus criterios de evaluación y aceptación.

3.1. Contexto: *DashAI*

Se ha mencionado en secciones anteriores que existe una brecha entre el desarrollo tecnológico en el área de inteligencia artificial y los profesionales con las habilidades para aplicar estos avances. Pues bien, ¿qué es entonces lo que impide el desarrollo de profesionales capacitados en IA?. Principalmente son las altas barreras de entrada que tiene esta área, entre estas barreras se encuentran:

1. El gran número y heterogeneidad de tareas (*tasks*) que existen. Algunas de ellas son: clasificación de texto, regresiones, procesamiento de imágenes, traducción de texto, entre otras. Todas ellas son fundamentalmente distintas e incluso un profesional experimentado en una de ellas podría no dominar demasiado las demás.
2. Se mencionó previamente que día a día se realizan nuevos avances en IA. Con un área que avanza tan rápidamente y está en constante cambio es muy difícil adaptarse, sobre todo a los nuevos métodos o modelos para los cuales es fundamental dominar lenguajes de programación como *Python* y no solo eso. Sino que además, se necesita dominar bibliotecas complejas como *Pytorch* o *Tensorflow*.
3. El conocimiento necesario para aplicar correctamente los algoritmos o métodos en IA, lo cual corresponde principalmente a matemáticas (estadística, probabilidades, algoritmos, etc.) y programación (se mencionó en el punto anterior el dominar lenguajes o bibliotecas complejas que, además, se encuentran en constante actualización).

Con las barreras de entrada expuestas anteriormente se manifiesta una respuesta de parte del

mundo del desarrollo de software: una herramienta capaz de disminuir al mínimo las barreras (1) y (2) mientras que facilite superar (3).

Los *software* para realizar tareas de inteligencia artificial sin tener que programar no son algo nuevo, un ejemplo de esto es *WEKA*, un *software* libre desarrollado por la Universidad de Waikato en 1993. Existen además otros *software* similares como *RapidMiner*, otros más actuales como *Perceptilabs* y *Akkio* e incluso existen software que intentan disminuir la barrera del conocimiento automatizando la mayor parte del proceso (AutoML), entre estos *software* se pueden incluir *KNIME* o *H2O*.

Sin embargo, se piensa que todos estos *software* son insuficientes, sobre todo con la gran explosión en el área de IA en que nos encontramos, las razones de esto son las siguientes:

- Algunos de ellos se han quedado atrás tanto en el lenguaje de programación que utilizan, como en la implementación de modelos del estado del arte. Algunos de estos software se diseñaron solo para realizar clasificaciones, regresiones y *clustering*. Sin embargo, hoy en día, los requerimientos son mucho mayores: traducción automática, transcripción de audio, generación de imágenes, etc.
- Otros son de pago, esto quiere decir que es necesario contratar un servicio para acceder a todo el beneficio y potencial que estos software ofrecen.
- Algunos son demasiado especializados, por ejemplo, enfocándose en soportar solo ciertas bibliotecas, lo cual, podría terminar provocando que no sean capaces de adaptarse al estado del arte en los años venideros.
- Otros *software* no abordan el tema de explicabilidad de modelos. Ni por medios propios ni tampoco integrando bibliotecas científicas existentes. De manera que los modelos entrenados son vistos solamente como "cajas negras".

Para intentar resolver estas falencias surge *DashAI*, el cual es un *software* de código abierto (*open source*) en desarrollo que permite que usuarios sin tener la necesidad de programar puedan: configurar, entrenar, comparar y utilizar modelos de *machine learning* tanto tradicionales como del estado del arte. Ahora bien, conociendo la existencia de herramientas actuales que también son capaces de construir modelos del estado del arte sin tener que programar y además, de herramientas que ni siquiera necesitan contar con experiencia previa en inteligencia artificial (*AutoML*), se hace necesario abordar la pregunta: ¿qué hace a *DashAI* diferente a dichas herramientas?. La respuesta a esta pregunta puede resumirse en los siguientes puntos:

1. **Tasks**: en primer lugar, una de las principales fortalezas con las que cuenta *DashAI* es que no está enfocado en modelos, sino que su enfoque principal es en "tasks" (tareas) de inteligencia artificial. Entendiendo por "task" a, por ejemplo, clasificación de texto o clasificación de imágenes. Estas "tasks" estarán unificadas en un patrón de *software* común. Por ejemplo, todas tienen una función "fit" para entrenar y otra "predict" para entregar su *output*). Este enfoque en "tasks" hace que el *software* sea fácilmente extensible puesto que entonces cualquier programador debiese poder diseñar una nueva

"task" e implementar modelos para ella. Este enfoque en "tasks" facilita el no se quedarse atrás con el tiempo, pues es independiente de la biblioteca utilizada y le permite adaptarse a los cambios que surjan en el área de IA.

2. **Configuración**: *DashAI* permite adaptarse al conocimiento que tengan los usuarios en IA. Esto se puede aplicar principalmente a su módulo de configuración de un experimento. Este módulo permite configurar parámetros con toda la flexibilidad que el usuario quiera. La forma en que se aborda esto es a través de la idea de "objetos configurables" los cuales engloban a la categoría de modelos y, a su vez, a parámetros que podrían tener estos modelos. De esta forma es posible proporcionar una configuración muy flexible y que se adapta al conocimiento que tengan los usuarios.

3. **Código abierto**: Otra de las grandes fortalezas de *DashAI* es ser *Open Source* (de código abierto), lo que le proporciona una gran ventaja puesto que el objetivo principal de las aplicaciones de IA sin código es acercar este campo a la mayor cantidad de personas posible, disminuyendo las barreras de entrada. La mayoría de aplicaciones para IA sin programar se hace cargo de la barrera de entrada de los conocimientos en programación. Sin embargo, también existen otras barreras de entrada como el dinero que se debe pagar para utilizar estas herramientas. *DashAI* ofrece disminuir las barreras de entrada al mundo de la inteligencia artificial al mínimo para, de esta forma, incluir a la mayor cantidad de personas. Esto reportaría grandes beneficios como una mejor reproducibilidad de los experimentos realizados, una detección más rápida de errores y poder idear aplicaciones más creativas de estos modelos[20].

4. **Extensibilidad**: Aunque *DashAI* está pensado para evitar la programación, los usuarios que quieran utilizarla podrán beneficiarse de una de las más grandes fortalezas del *software*: su extensibilidad. Entendiendo por "extensibilidad" a que cualquier usuario sea capaz de diseñar e implementar sus propias *task* con sus modelos asociados, así como también implementar sus propios modelos para *tasks* pre-existentes, y utilizar todo esto en conjunto con el estado base del *software*. Además, este proceso es muy simple de realizar, de forma que basta simplemente agregar archivos de código que contengan los modelos/*tasks* implementados, para que estos puedan ser utilizados en su interfaz gráfica.

5. **Explicabilidad**: *DashAI* considera integrar el uso de bibliotecas científicas que permitan facilitar la comprensión de los fundamentos de los modelos, y con esto, permitir a los usuarios tomar decisiones informadas sin tener que aprender a utilizar estas bibliotecas. Logrando de esta forma, que no necesiten programar para aplicar la inteligencia artificial.

3.2. Problema a abordar

Ahora bien, una de las componentes fundamentales para que *DashAI* logre su objetivo de bajar las barreras de entrada para el área de IA es la forma en que el *software* interactúa con los usuarios, es decir, la interfaz gráfica (GUI), y es aquí donde se presenta el problema que se aborda específicamente en el presente trabajo.

Como se ha dicho anteriormente *DashAI* es aún un proyecto en desarrollo, se cuenta con una componente principal por parte del *backend* que involucra un pequeño conjunto de modelos y *tasks*. Además cuenta con una interfaz de usuario básica que permite cargar un set de datos y seleccionar los modelos con los que se quiere trabajar en un experimento.

Entonces, el problema que se quiere resolver es que la interfaz con la que cuenta actualmente *DashAI* está incompleta al punto que esta no representa las principales ventajas con las que cuenta el *software*, las cuales fueron descritas en los párrafos anteriores. Estas ventajas permitirían dar al *software* una mejor presentación que comunique realmente sus fortalezas y, con esto, impulsar su desarrollo a través de agregar a instituciones y personas interesadas en estas ideas al proyecto o poner a prueba estas ideas con potenciales usuarios para así descubrir las debilidades que puedan tener.

3.3. Relevancia

Se piensa que desarrollar *DashAI* y, en específico, su interfaz gráfica es relevante puesto que permitiría generar un impacto en el área de inteligencia artificial gracias a que es de código abierto, lo que permitiría incorporar la retroalimentación de varios potenciales usuarios. Se cuenta además con el apoyo de ingenieros de *CENIA*[1] (Centro Nacional de Inteligencia Artificial) donde el proyecto *DashAI* forma parte de *OpenCENIA*[2] junto con otros proyectos también de código abierto.

Se diferencia de otros *software* presentando ventajas como:

- Permite adaptarse a la rápida expansión del área de la IA gracias a su enfoque en *tasks* y su extensibilidad.
- Permite adaptarse a la experiencia del usuario al configurar un experimento a través de su idea de objetos configurables.

Ahora bien, en la sección 3.1 "Contexto: *DashAI*" se han presentado los objetivos y las ventajas del *software*. Es un proyecto ambicioso, y a largo plazo, por lo que no se espera que el presente trabajo de memoria aborde todos ellos, algunos de estos objetivos serán dejados para el futuro.

Específicamente la interfaz de usuario que se diseñe e implemente abordará las siguientes ventajas del *software*:

- Enfoque en *tasks*.
- Configuración con alta flexibilidad de acuerdo a la experiencia del usuario.

[1] www.cenia.cl
[2] github.com/OpenCENIA

- Código abierto.
- Permitir ser extensible para facilitar que el usuario agregue *tasks* y modelos nuevos.

Se puede notar que esta primera versión de *DashAI* no aborda la explicabilidad. Esto hace que los usuarios deban ver los modelos como "cajas negras" lo cual perjudica mayoritariamente a los que se estén iniciando en el área y deseen expandir sus conocimientos.

Existen varios peligros que pueden ocurrir al no entender los fundamentos de los modelos:

- Se pueden tomar malas decisiones al malinterpretar algunas métricas específicas que pueden dar aparentemente buenos resultados, pero esconder problemas en el entrenamiento de los modelos.
- Los modelos pueden esconder sesgos, los cuales se hacen particularmente peligrosos cuando se está abordando un problema que involucra seres humanos. Por ejemplo, un modelo entrenado sin entender sus fundamentos podría discriminar por raza o género[16]. Tal como pasó al utilizar un predictor de riesgos de reincidencia en tribunales de Estados Unidos, el cual se equivocaba el doble con personas afrodescendientes al predecir que estas eran de alto riesgo cuando realmente no era así[3].

Considerando los riesgos que implica no tener explicabilidad, es entendible que no se quiera dejar este problema sin abordar del todo. Por ello, se presentan algunas recomendaciones para ayudar a los usuarios a ser más críticos con los resultados al utilizar esta primera versión de *DashAI*:

- Crear documentación y tutoriales claros sobre la forma de usar la plataforma. Estos recursos deben dejar en claro las limitaciones en cuanto a la interpretación de los resultados de un experimento.
- Facilitar mediante *backend* que los usuarios puedan acceder a los modelos que entrenaron en formatos tradicionales. De manera que facilite que los usuarios puedan complementar su experimento con bibliotecas científicas de explicabilidad.
- Promover el uso del módulo para entregar *inputs* a los modelos entrenados. Aunque esto no permite descartar que el modelo entrenado tenga sesgos, en algunos casos sí sería posible detectarlos al entregar unos pocos *input* y ver lo que el modelo entrega para estos.
- Prestar atención a canales de comunicación que se tenga con los usuarios, de manera que estos puedan reportar sus preocupaciones sobre los modelos disponibles y que estas preocupaciones puedan ser atendidas rápidamente.

Habiendo expuesto esto, se tiene que *DashAI* puede generar un impacto en el área de inteligencia artificial siendo un *software* que presenta ventajas y aborda problemas que el resto de herramientas competidoras no hace. Mientras que proporcionar una interfaz para una

[3]https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing

versión inicial del proyecto, aún sin implementar explicabilidad pero siguiendo las recomendaciones anteriores, permitiría dar un empuje inicial al proyecto al comunicar la mayoría de sus ventajas a través de una interfaz gráfica. De esta forma, se podría presentar el proyecto a instituciones o expertos en el área que deseen unirse al equipo y colaborar para que *DashAI* pueda cumplir, en próximas versiones, todos los objetivos que se han planteado previamente.

3.4. Requisitos

Una gran ventaja que se tiene para diseñar la interfaz es la existencia de *software* para IA sin programar, todos con una gran variedad de interfaces de las cuales se puede aprender tanto de sus aciertos como de sus errores. Es por esto que para establecer los requisitos de la interfaz de usuario a diseñar e implementar se han analizado las interfaces de otros software junto con revisar literatura sobre el tema.

3.4.1. Tareas

Debido a la variedad de intentos por hacer que los humanos puedan interactuar más fácilmente con modelos de *machine learning* a través de interfaces gráficas, es posible definir un estándar con ciertas tareas que el usuario debe poder realizar para proporcionarle una experiencia adecuada. En particular, este estándar se ha estudiado enfocándose en aplicaciones de *AutoML*[9], las que corresponden a *software* en los que no se necesita ni experiencia en programación, ni tampoco demasiada experiencia trabajando con modelos de *machine learning*.

Aunque los objetivos de *DashAI* y los *software* de *AutoML* sean distintos, es posible encontrar elementos comunes con los que establecer requisitos tomando las siguientes consideraciones:

- El objetivo principal de la interfaz será poder definir experimentos para tasks específicas. Poder configurar los modelos de estos experimentos de manera flexible y entrenar estos modelos para que resuelvan esta task.
- No se consideran tareas de ingeniería de datos (*Data Engineering*) puesto que es un área que está demasiado poco trabajada aún en el proyecto, por esto, se dejarán estas tareas para más adelante.

Se presenta entonces a continuación en la tabla 3.1 con las tareas que la interfaz de *DashAI* debe poder realizar, las cuales están fuertemente basadas en literatura sobre interfaces de usuario [9] junto con algunas tareas adicionales para hacer énfasis en las ventajas comparativas que tiene *DashAI* sobre otros *software*.

	Lista de tareas
1	Un usuario puede determinar el tipo de los modelos a utilizar
2	Un usuario puede usar un experimento creado previamente
3	Un usuario puede determinar los valores para los parámetros de un modelo
4	Un usuario puede incluir múltiples modelos en un mismo experimento
5	Un usuario puede comparar los resultados de distintos modelos
6	Un usuario puede incluir en un experimento modelos del mismo tipo con distinta configuración de parámetros
7	La interfaz muestra información relevante en todo momento
8	Un usuario puede corregir sus errores sin perder su progreso
9	La interfaz recuerda interacciones recientes
10	La interfaz muestra las consecuencias de las acciones del usuario
11	La interfaz provee controles globales
12	Un usuario puede determinar la task con la que se realizará el experimento
13	Un usuario puede determinar el set de datos que usará para un experimento
14	Un usuario puede probar modelos previamente entrenados

Tabla 3.1: Requisitos funcionales para la interfaz gráfica de *DashAI*, obtenidos inspirándose en [9], adaptándolos a las necesidades de *DashAI*

3.4.2. Criterios de aceptación

Un primer criterio de aceptación tiene que ver con las tareas expuestas en la tabla 3.1 y es que la mayoría de estas deben cumplirse. Para ser más exacto, **al menos el 75 % de las tareas deben poder realizarse en la interfaz**

Se ha mencionado también que se quiere realizar una prueba de usabilidad con la interfaz gráfica. Pues bien, los criterios de aceptación se adecúan a una prueba de usabilidad con pocos usuarios y más orientada a lo cualitativo.

Para evaluar usabilidad se utilizan medidas objetivas de eficacia y eficiencia, así como también medidas subjetivas que miden la percepción del usuario. Se presenta a continuación los criterios de aceptabilidad para cada una de estas medidas.

Eficacia

Para evaluar eficacia se utiliza un criterio simple basado en si estos fueron capaces de cumplir el objetivo de cada tarea, independiente del tiempo que les haya tomado hacerlo.

Para aceptar la eficacia de la solución se debe cumplir que: **para cada tarea al menos un 80 %** de los usuarios debe haber tenido éxito.

Eficiencia

Para evaluar eficiencia se considera el tiempo que les toma a los usuarios completar cada una de las tareas asignadas. Para aceptar que la solución es eficaz se debe cumplir que para cada tarea **la mediana del tiempo de todos los usuarios debe ser inferior al 70 % del tiempo límite asignado**. Es decir, de los usuarios que completaron la tarea, la mayoría lo hizo razonablemente rápido.

Percepción subjetiva de los usuarios

Por último, para evaluar la percepción subjetiva de los usuarios se les pide responder el cuestionario *NASA-TLX*, en el cual se pide asignar un puntaje entre 1 y 20 para 6 categorías las cuales son:

- Exigencia mental (-)
- Exigencia física (-)
- Exigencia temporal (-)
- Rendimiento (+)
- Esfuerzo (-)
- Frustración (-)

En la lista anterior los símbolos significan:

- (-) Mientras menor sea el puntaje asignado, mejor evaluada está la interfaz. (Por ejemplo, menor frustración es mejor).
- (+) Mientras mayor sea el puntaje asignado, mejor evaluada está la interfaz (Por ejemplo, mayor rendimiento es mejor).

El criterio de aceptabilidad que se pide es el siguiente: **para las categorías negativas (-) se pide que su puntaje promedio entre todos los usuarios sea inferior a 10. Mientras que para la categoría positiva (+) se pide que este promedio sea superior 10**. Es decir, los criterios negativos deben estar en valores bajos, mientras que los positivos deben estar en valores altos.

Capítulo 4

Solución

En este capítulo se parte dando un contexto del resto de componentes del proyecto *DashAI* que no respectan a la interfaz de usuario, para luego entregar una descripción de la solución a implementar. Se continúa la sección haciendo un reporte de un análisis realizado para varias interfaces de *software* similares, este análisis revisa las formas que estas tienen de abordar cada requisito de los que han sido presentados en secciones anteriores. Finalmente, se presenta la solución que se ha diseñado e implementado junto con las justificaciones de su diseño.

4.1. Componentes actuales de *DashAI*

DashAI tiene todo un equipo de trabajo detrás. Hay alumnos realizando su memoria o tesis en el área de *backend* con proyectos como construir una *API* que se comunique con el *frontend* de la herramienta , implementar modelos del estado del arte para ser agregados al estado base del *software* o encontrar la mejor arquitectura para asegurar la extensiblidad de la herramienta. El estado de la herramienta en lo que concierne al *backend* se detalla a continuación:

4.1.1. Endpoints

La *API* se comunica con la interfaz gráfica utilizando *endpoints* los cuales aún son básicos debido a que los estudiantes encargados se encuentran comenzando su proceso de memoria, en lo siguiente se describen los principales *endpoints*

- **uploadDataset**: este *endpoint* se utiliza para cargar un *dataset*, mediante el cual se identifica la *task* a la que pertenece y con esta, se retorna a la interfaz gráfica una lista que contiene los modelos compatibles con dicha *task*.
- **selectModel**: recibe el nombre de un modelo, lo busca en los archivos locales y, en caso de encontrarlo, retorna un objeto *JSON* con los parámetros correspondientes a dicho modelo.

- **selectedParameters**: Recibe un objeto *JSON* que contiene la configuración de parámetros e instancia un modelo con ellos que estará listo para ser entrenado.
- **runExperiment**: se encarga de entrenar los modelos instanciados usando el dataset cargado.
- **getResults**: Entrega los resultados de entrenamiento y *test* (métrica principal) de los modelos que fueron entrenados en el experimento.
- **play**: Recibe un *input* para uno de los modelos entrenados previamente, realiza una predicción con este *input* y retorna lo que el modelo ha entregado como *output*.

4.1.2. Tasks y Modelos implementados

Se cuenta con un conjunto básico de *tasks* y modelos implementados en el *software* los cuales son descritos a continuación.

Task: Clasificación numérica

Para clasificación numérica se cuenta con tres modelos típicos.

- KNN.
- SVM.
- Random Forest.

Task: Clasificación de texto

Como es usual en tareas que involucran procesamiento de lenguaje natural, para procesar texto se necesita transformar estos datos a números, para englobar todo el proceso se tiene un modelo el cual es un *Wrapper* de texto. El modelo recibe este nombre pues utiliza otros modelos numéricos para clasificar, pero incluye un preprocesamiento en los datos para convertir el lenguaje natural a datos numéricos.

Task: Traducción automática

Para esta *task* se cuenta con un *transformer* obtenido desde *Huggingface*. Este modelo recibe el nombre de "Helsinki-NLP/opus-mt-en-es"[1]. Algo particular de este modelo es que ha sido el último en ser agregado, se ha implementado en paralelo con la construcción de la interfaz gráfica. Por esto, en secciones posteriores se utiliza este modelo para probar la extensibilidad de la herramienta.

[1]https://huggingface.co/Helsinki-NLP/opus-mt-en-es

4.2. Descripción de la solución

Se comienza dando una descripción de la solución la cual corresponde a la interfaz gráfica diseñada e implementada, estableciendo las bases de lo que se priorizó y las razones de esto.

Cuando se quiere experimentar con modelos de *machine lerning* es común realizar los siguientes pasos principales:

1. En primer lugar se definen los datos con los que se trabaja y se realiza un preprocesamiento (en caso de ser necesario o útil).
2. Luego, se define el modelo/algoritmo que se usará y también se definen los parámetros de dicho modelo/algoritmo.
3. Teniendo los datos y el modelo se realiza el entrenamiento del modelo utilizando los datos.
4. Una vez ha terminado el entrenamiento, se visualizan los resultados que ayudan a darse una idea de qué tan buenas predicciones realiza el modelo.
5. Por último, si se quiere obtener un mejor entendimiento del modelo entrenado, se le entregan *inputs* específicos para observar y analizar sus resultados.

Estos pasos funcionan a modo de *pipeline*, es decir, el resultado de un paso se usa como entrada para el paso siguiente, por lo que es razonable separar los pasos entre sí y que cada uno de estos tenga su propia lógica interna para tratar con su entrada y así generar su salida. Una última observación es que es posible juntar el paso dos y tres en un único paso en el que se define el modelo, sus parámetros y se comienza el entrenamiento, con esto se obtienen finalmente cuatro pasos, estos cuatro pasos pueden ser convertidos a cuatro componentes de la interfaz.

Para lograr los requisitos la solución consta de 4 componentes principales donde cada una cumple una función específica:

- **Carga de datos**: Interfaz inicial donde el usuario podrá subir el set de datos con el que desea trabajar, se le presenta la *task* con la que se corresponde dicho set de datos y se le notifica si hubo algún error en la carga.
- **Configuración**: Como su nombre indica, tiene como principal función configurar un experimento. Es decir, añadir uno o más modelos al experimento, opcionalmente configurar dichos modelos, visualizar la configuración del experimento y comenzar la ejecución de este mismo. Es en esta visualización donde se centra la mayor interacción con el usuario. Debido a esto, se deben incluir también formas de guiar al usuario y permitirle conocer las consecuencias de sus acciones.
- **Analizar y comparar**: Esta interfaz permite al usuario analizar los modelos que ha entrenado, revisar los parámetros utilizados y comparar en caso de haber entrenado a más de un modelo.

- **Utilizar modelos**: Esta interfaz tiene como objetivo proporcionar al usuario una forma de "jugar" con los modelos, para observar los resultados que arroja un modelo con *inputs* específicos. Por ejemplo, en el caso de la tarea de traducción automática inglés-español el usuario proporciona al modelo texto en inglés y el modelo le devuelve la traducción de dicho texto al español.

Algo que es necesario notar es que tanto los requisitos como la descripción general de la solución no están considerando ingeniería de datos, aunque esto es un dato clave para una aplicación de IA y está contemplado en el proyecto *DashAI*. Esto se debe a que *DashAI* aún tiene mucho camino por recorrer en el área de ingeniería de datos, debido a que no hay una parte desde el *backend* que soporte estas tareas y que su formato o requerimientos podrían variar mucho en el futuro. Por lo tanto, se ha decidido no diseñar ni implementar funcionalidades en esta área.

4.3. Metodología de trabajo

Se mencionó con anterioridad que el proyecto *DashAI* cuenta con un equipo de desarrollo, el cual está enfocado mayoritariamente en el *backend*. Para organizar al equipo se utiliza la metodología *Shape up*, la cual es similar a la metodología *Scrum* pero las definiciones de tareas son más generales para entregar al desarrollador una mayor libertad para diseñar las soluciones. Se tiene una reunión semanal para coordinar al equipo, presentar resultados parciales y obtener retroalimentación del resto del equipo.
Enfocándose únicamente en la construcción de la interfaz gráfica se han desarrollado tres *Pitches*, cada uno consta de un total de cuatro semanas, con una semana de *cooldown*.

4.3.1. Pitch 1

El primer *pitch* se enfocó completamente en replicar la interfaz gráfica previamente construida en *Dash*, pero esta vez utilizando *React*. Las razones para este cambio de tecnología se debieron principalmente a las siguientes razones:

- Construir interacciones de usuario complejas que se adapten a las necesidades de extensibilidad de la aplicación era más complicado utilizando *Dash* que utilizando *React*.
- Problemas de rendimiento, ya que se estaban notando algunas interacciones que tomaban más tiempo del que deberían. Al utilizar *Dash* estas interacciones son mucho más complicadas de optimizar que si se utiliza *React*.
- En caso de querer dejar disponible el proyecto *DashAI* en un servidor, *Dash* no contaba con el concepto de sesiones, por lo cual, no era factible que varios usuarios interactuaran con la interfaz al mismo tiempo.

Para construir la interfaz en *React* se utilizó el enfoque de componentes funcionales. Esto con el objetivo de poder utilizar los *Hooks* para realizar interacciones dinámicas y eficientes.

Al finalizar este *pitch* se contó con una interfaz con una vista muy básica, la cual contaba con la mayoría de interacciones, pero con un diseño deficiente.

4.3.2. Pitch 2

Este *pitch* se enfocó en mejorar el diseño de la aplicación, para hacerla más amigable a la vista. Para ello se utilizaron principalmente reglas *CSS* junto con la biblioteca *styled-components*. Se pidió además consejo a una diseñadora para la paleta de colores, tipos de fuente y corrección de algunas interacciones de la interfaz.

Al terminar este *pitch* se contó con una interfaz visiblemente amigable y funcional, pero aún con muchos detalles y errores que manejar.

4.3.3. Pitch 3

Este *pitch* estuvo enfocado en mejorar la usabilidad de la interfaz. Por ejemplo, añadiendo guías para que el usuario fuera capaz de entrenar modelos. Además de mejorar el manejo de errores y reestructurar el código. Al terminar este *pitch* se contó con la interfaz gráfica lista para su evaluación.

4.4. Diseño

Se pasa ahora a una primera exposición de la interfaz gráfica diseñada e implementada, principalmente enfocada en la parte del diseño, es decir, colores, tipografías y qué tan agradable a la vista es.

Como se ha mencionado anteriormente, la interfaz está organizada en cuatro componentes principales, las cuales se encargan de cargar datos, configurar experimento, visualizar resultados y utilizar un modelo. Estas cuatro componentes se pueden visualizar en la siguiente imagen con una vista total de la interfaz gráfica. Normalmente, la vista se encuentra centrada en solo un paso a la vez y los pasos se van desplegando a medida que el usuario los necesita.

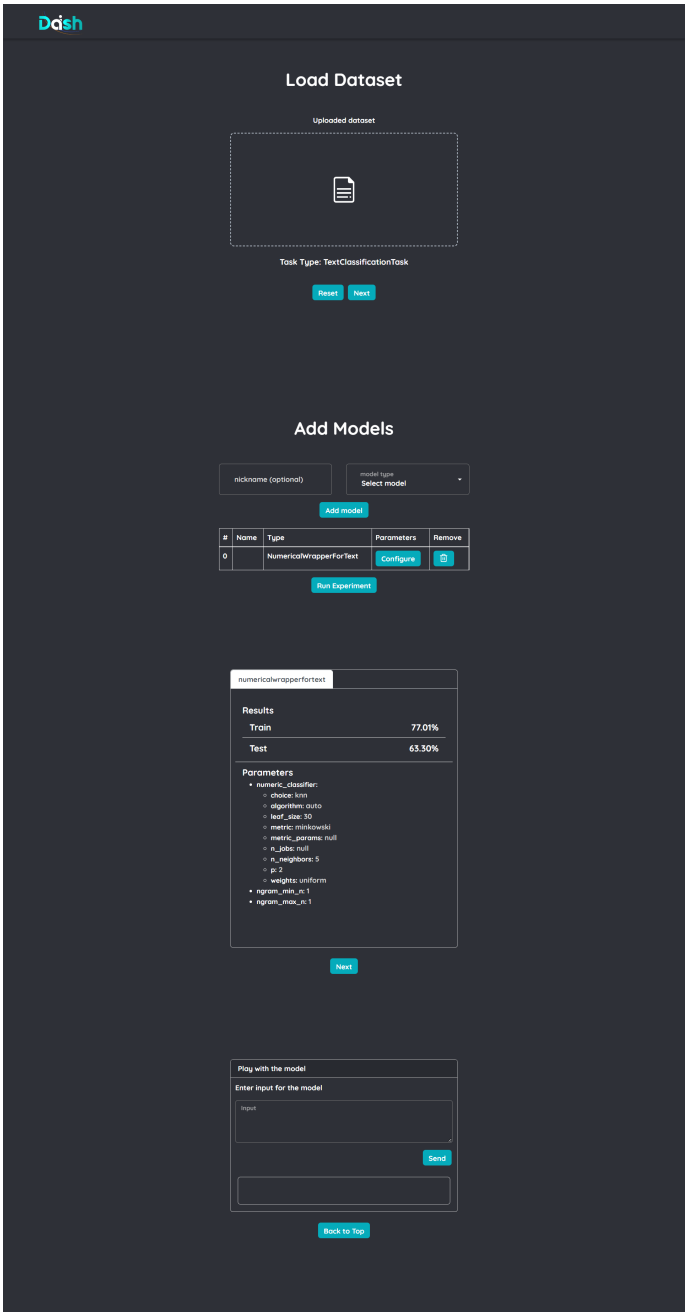

Figura 4.1: Vista completa de la interfaz (realizada con la extensión *Full Page Screenshot* de *chrome web store*). Normalmente, la vista se encuentra enfocada en solo uno de los cuatro pasos a la vez.

Al contar con un diseño vertical en el que la información aparece de acuerdo al progreso del usuario y según se la necesita se tiene que esta parte del diseño aborda una de las tareas expuestas en la tabla 3.1:

7. **La interfaz muestra información relevante en todo momento**

4.4.1. Color

Para la elección de la paleta de colores se tomó como referencia el logo de *DashAI* que ya había sido previamente diseñado por el equipo.

Figura 4.2: Logo de *DashAI*

Los colores inicialmente utilizados fueron ligeramente corregidos al consultar con la diseñadora para verse mejor al ser usados en una interfaz gráfica.

La disposición de los elementos se ha realizado de manera iterativa, realizando prototipos de la funcionalidad directamente en el código, para presentarlos al equipo en la reunión semanal y conservarlos o descartarlos. En caso de conservarlos se les iba añadiendo complejidad a medida que avanzaba el desarrollo.

Para conseguir que la aplicación sea fácilmente adaptable se han definido los colores en un archivo *JSON* aparte del código, desde donde se extraen todos los colores utilizados en la interfaz gracias a la biblioteca que se utilizó para estilos: *Styled components*.

Para finalizar se entrega la paleta de colores utilizada, además de una pequeña justificación de su uso.

- (hex, 2e3037) Se ha utilizado para el fondo, un tono oscuro agradable a la vista.
- (hex, 00bebb) Se ha utilizado para el fondo de los botones debido a su contraste con el color oscuro del fondo.
- (hex, ffffff) Se ha utilizado blanco para texto y marcos, debido a que tiene un alto contraste con el fondo oscuro, pero además, es capaz de contrastar también con el color de fondo de los botones.

Se han utilizado además ligeras variaciones de estos colores para motivos de usabilidad, por ejemplo, al pasar el puntero sobre un botón, este se oscurece ligeramente.

4.4.2. Tipografía

Para la tipografía se ha decidido utilizar: *Quicksand*[2] en su estilo *Bold*.

La razón de uso de esta tipografía es que se siguió la recomendación realizada por la diseñadora a la que se pidió consejo.

4.5. Diseño de interacciones principales

Aunque en general la innovación es algo bueno, cuando se quiere tener un diseño que sea simple para la mayoría de la gente, lo mejor es apegarse a estándares o patrones comunes. Es por esto que, ya teniendo establecidos los requisitos/tareas que debe cumplir la interfaz, es necesario analizar otras interfaces que apuntan a objetivos parecidos. Todo esto con el objetivo de extraer mecanismos comunes que se utilicen para resolver estos requisitos/tareas.

Tomando como base las tareas establecidas en la tabla 3.1, se han analizado las interfaces gráficas de los siguientes *software*:

- WEKA (Experimenter)[3]
- RapidMiner Studio[4]
- Perceptilabs[5]
- Akkio[6]
- H2O Flow[7]
- KNIME[8]

Cada interfaz de los *software* mencionados en la lista anterior ha sido probada y cuidadosamente analizada, poniendo especial atención en las tareas de la tabla 3.1 con el objetivo de identificar el mecanismo con el que se cumple cada tarea mencionada. Por ejemplo, en algunas interfaces para crear y configurar un modelo de *machine learning* es necesario utilizar el puntero para arrastrar y soltar (*drag and drop*) ciertos elementos, con lo cual, el mecanismo

[2]https://fonts.google.com/specimen/Quicksand
[3]https://www.cs.waikato.ac.nz/ml/weka/
[4]https://rapidminer.com/platform/
[5]wwww.perceptilabs.com
[6]www.akkio.com
[7]https://docs.h2o.ai/h2o/latest-stable/h2o-docs/flow.html
[8]www.knime.com

utilizado para completar la tarea se ha registrado como *drag and drop*. Con esto en mente, se construye la tabla que se presenta a continuación. La tabla 4.3 tiene el objetivo de encontrar estándares o patrones comunes que sigan las interfaces y poder replicarlos para conseguir diseñar una interfaz fácil de usar.

Tarea	WEKA (Experimenter)	RapidMiner Studio	Perceptilabs	Akkio	H2O Flow	KNIME
Un usuario puede determinar la clase de los modelos que desea utilizar	Value entry	value entry	value entry	**	Value entry	Value entry
Un usuario puede usar un experimento creado previamente	Value entry	value enttry	value entry		value entry	Value entry
Un usuario puede determinar valores para los parámetros de un modelo	Value entry	value entry	value entry	**	Value entry	Value entry
Un usuario puede incluir múltiples modelos en un mismo experimento	Cuadro resumen de los algoritmos a utilizar	drag and drop	Tabla resumen	**	Notebook cells	Drag and drop
Un usuario puede comparar los resultados de distintos modelos	Value entry	**	value entry	**	**	Drag and drop
Un usuario puede incluir en un experimento modelos de la misma clase con distinta configuración de parámetros	Cuadro comparativo	drag and drop	drag and drop	**	Notebook cells	Drag and drop
La interfaz muestra información relevante en todo momento	Steps in tabs	Steps in tabs	Step in tabs	Step in tabs	Notebook cells	Drag and drop
Un usuario puede corregir sus errores sin perder su progreso	Navegación hacia atrás	Navegación hacia atrás	Navegación hacia atrás	Navegación hacia atrás	Notebook cells	Drag and drop
La interfaz recuerda interacciones recientes	stored files	stored files, database, repository	stored files repository	**	stored files	stored files
La interfaz muestra las consecuencias de las acciones del usuario	Texto	Texto	**	Ventanas de diálogo	Texto	Texto
La interfaz provee controles globales	Value entry	value entry	**	**	**	Value entry
Un usuario puede determinar la task con la que realizará un experimento	**	**	selection	**	**	**
Un usuario puede determinar el set de datos que usará en un experimento	input file?	input	input	input	input	input
Un usuario puede jugar con modelos previamente entrenados	**	value entry	value entry	Value entry	input	Drag and drop

Figura 4.3: Tabla comparativa que muestra el mecanismo con el que cada interfaz analizada resuelve cada requisito, para "**" significa que no cumple el requisito señalado.

Utilizando la tabla anterior se pueden notar ciertos patrones en las interfaces analizadas. Por ejemplo, todas utilizan la entrada de valores (*value entry*) como mecanismo para que el usuario pueda determinar la clase de modelos que desea utilizar.

Teniendo esto en cuenta se establecen los mecanismos de interacción que utilizará *DashAI*, los cuales se resumen en la siguiente tabla y serán expuestos con mayor detalle en la sección que sigue.

Tarea	DashAI
Un usuario puede determinar la clase de los modelos que desea utilizar	Value entry
Un usuario puede usar un experimento creado previamente	Value entry
Un usuario puede determinar valores para los parámetros de un modelo	Value entry
Un usuario puede incluir múltiples modelos en un mismo experimento	Tabla resumen
Un usuario puede comparar los resultados de distintos modelos	Tabla resumen
Un usuario puede incluir en un experimento modelos de la misma clase con distinta configuración de parámetros	Tabla
La interfaz muestra información relevante en todo momento	Steps
Un usuario puede corregir sus errores sin perder su progreso	Navegación hacia atrás
La interfaz recuerda interacciones recientes	local storage
La interfaz muestra las consecuencias de las acciones del usuario	Texto
La interfaz provee controles globales	Value entry
Un usuario puede determinar la task con la que realizará un experimento	input
Un usuario puede determinar el set de datos que usará en un experimento	input
Un usuario puede jugar con modelos previamente entrenados	Value entry

Figura 4.4: Tabla que establece las interacciones diseñadas para *DashAI*.

Algunas observaciones con respecto a la tabla 4.4.

- Se ha optado por no usar drag and drop, puesto que esto tiene más sentido cuando se trabaja con modelos en vez de *task*, además de la excesiva complejidad para tomar en cuenta modelos que pueden ser, a su vez, parámetros de otros modelos, una funcionalidad que se ha mencionado anteriormente como "objetos configurables" y que se explicará en detalle más adelante.
- Se han adaptado algunas funcionalidades a su versión en desarrollo web. Por ejemplo, para recordar interacciones recientes la mayoría de aplicaciones analizadas usa *stored files* (archivos locales, pues la mayoría son aplicaciones de escritorio), mientras que *DashAI* utiliza *local storage*, una herramienta de los navegadores web para almacenar datos.

4.6. Screenshots

En lo siguiente se muestra en detalle la interfaz, su diseño, sus interacciones y lo que es posible llevar a cabo al usarla.

4.6.1. Cargar datos

Se comienza presentando la primera parte de la interfaz gráfica, la visualización que se muestra apenas iniciar la aplicación.

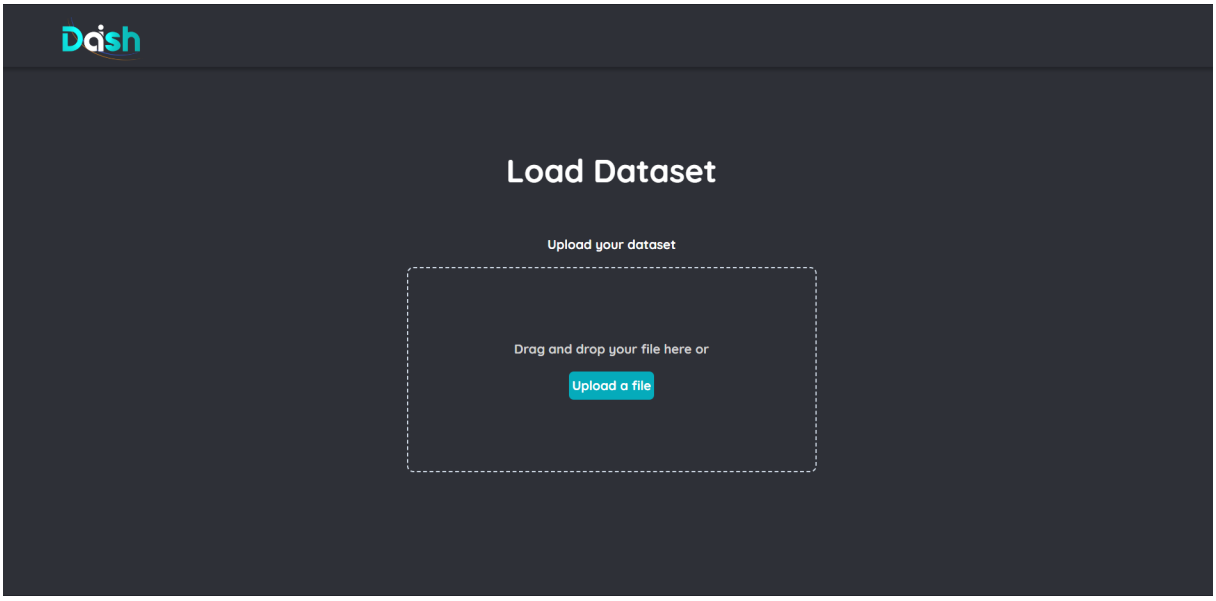

Figura 4.5: Primera vista de la aplicación, muestra un cuadro el que permite subir un set de datos

Una vez se le entrega un set de datos, mientras este se carga, se muestra la siguiente vista.

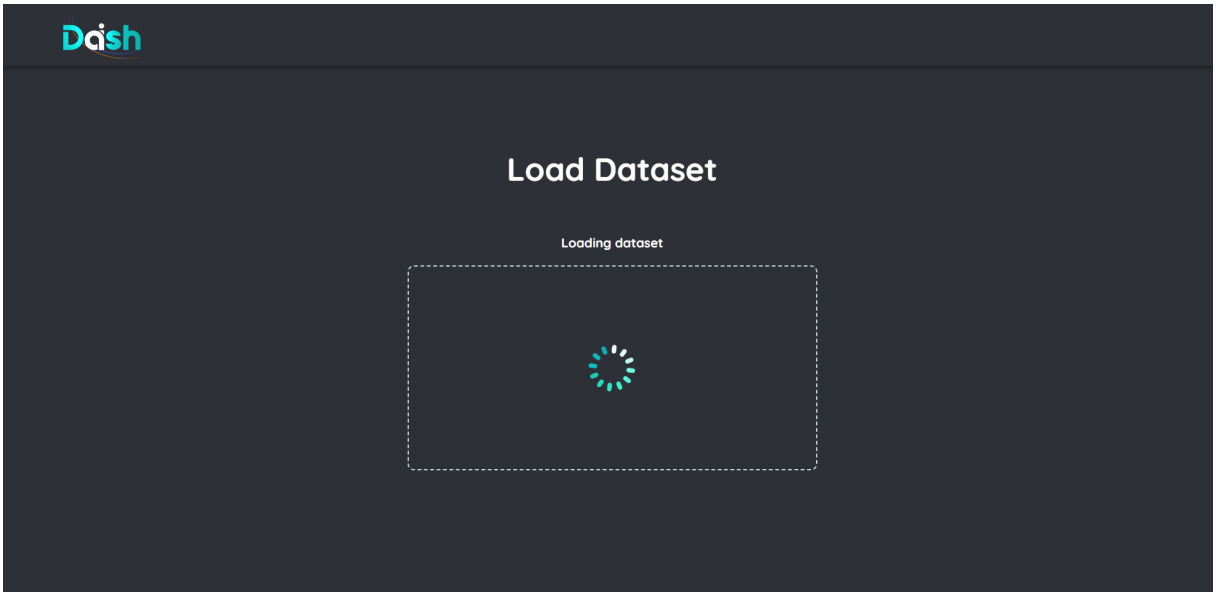

Figura 4.6: Al entregar un archivo, se muestra la siguiente vista, para que el usuario espere mientras su set de datos es cargado en el *backend*.

Cuando el set de datos se carga correctamente, se muestra la vista de la figura 4.7. Esta incluye lo siguiente:

- Muestra un ícono como *feedback* de que el set de datos ha sido cargado correctamente.
- Muestra en texto el tipo de *task* que ha sido identificado a partir del set de datos cargado. En este caso, se muestra *TextClassificationTask*, es decir, una tarea de clasificación de texto.

Luego, se puede presionar el botón *Next*, el cual lleva al usuario al siguiente paso realizando *scroll* automático.

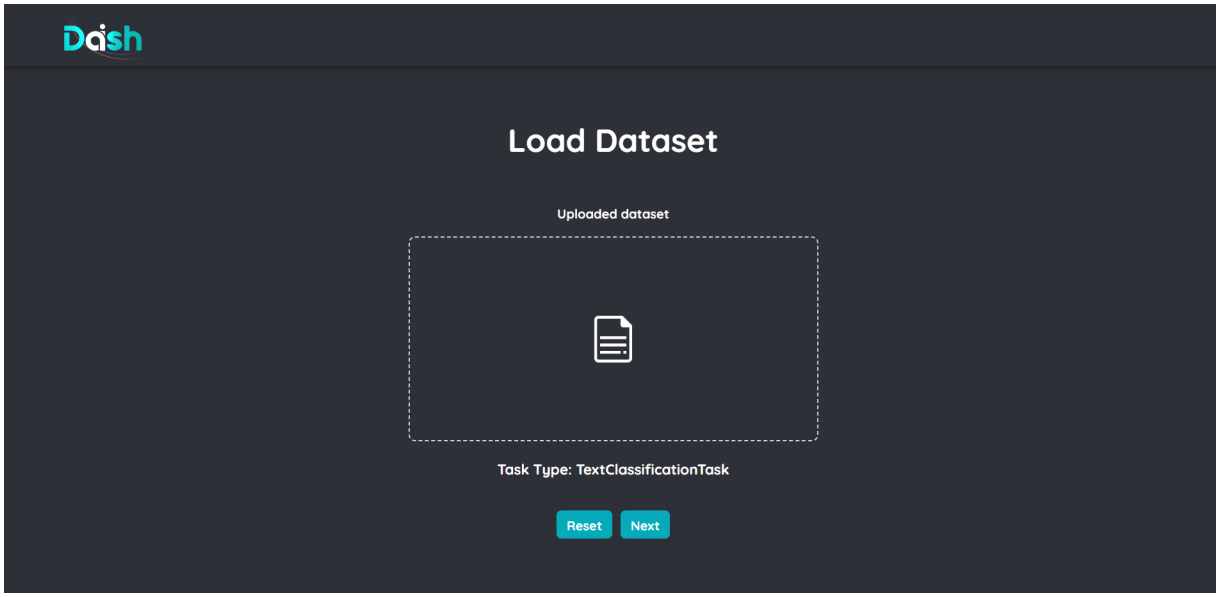

Figura 4.7: Vista que entrega *feedback* al usuario de que su set de datos se ha cargado correctamente, además del tipo de *task* que ha sido reconocido a partir del set de datos.

En caso de cargar un set de datos incorrecto , se muestra la siguiente vista para indicar al usuario que debe reiniciar el paso actual. Para más información sobre el formato del set de datos se puede revisar *Anexo A*.

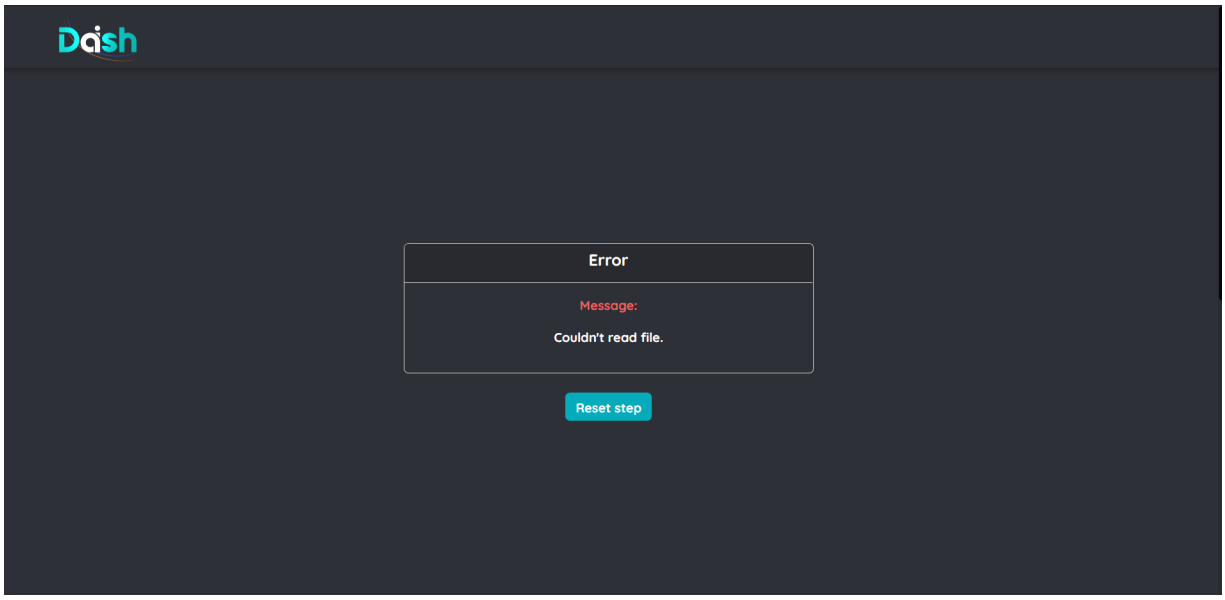

Figura 4.8: Vista de error con un mensaje que indica que el archivo subido no tiene el formato correcto.

Esta componente apunta a cumplir dos tareas de las que fueron listadas en la tabla 3.1.

12. **Un usuario puede determinar la task con la que realizará un experimento**

13. **Un usuario puede determinar el set de datos que usará en un experimento**

4.6.2. Configurar experimento

Habiendo realizado correctamente el paso anterior, es decir, cargar un set de datos válido, se muestra la vista que aparece luego de presionar el botón *Next*.

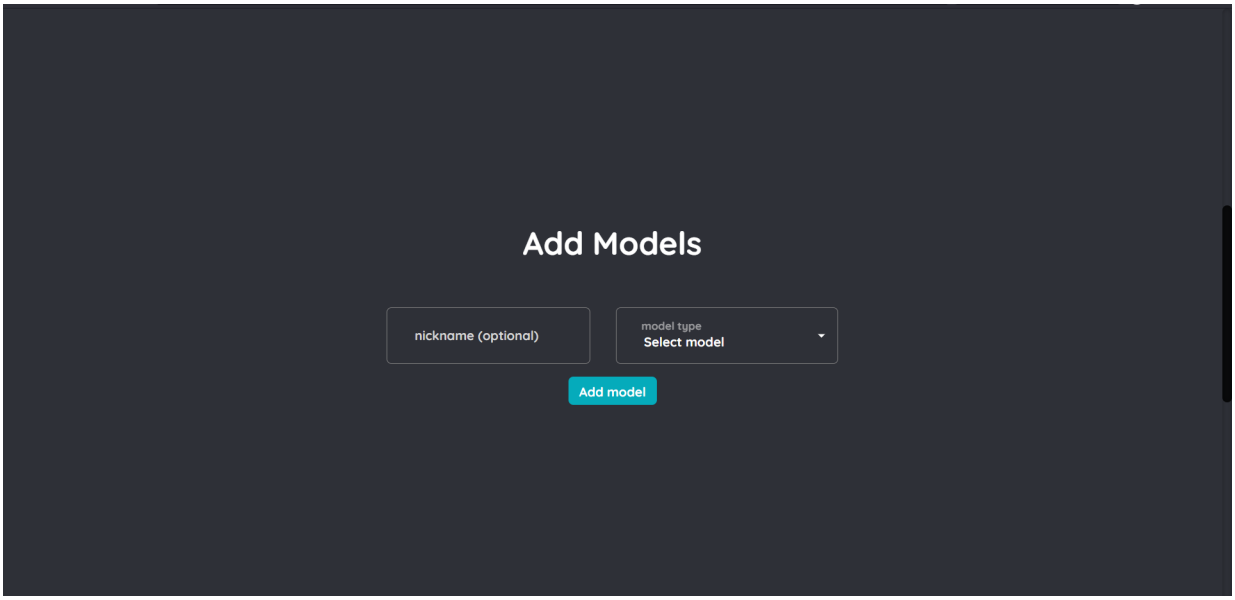

Figura 4.9: Visualización para agregar modelos a un experimento, se muestra que es posible agregar un *nickname* para diferenciar al modelo y seleccionar su tipo.

La vista presenta dos campos de un formulario para agregar modelos al experimento:

- *nickname (optional)* permite al usuario escribir un nombre con el cual identificar su modelo.
- *model type* corresponde a un selector, el cual permite al usuario elegir el tipo de un modelo (de los que son compatibles con la *task* del experimento).

Una vez el formulario ha sido llenado al presionar el botón *Add model* se agrega el modelo correspondiente a una tabla, tal como se muestra en la siguiente vista.

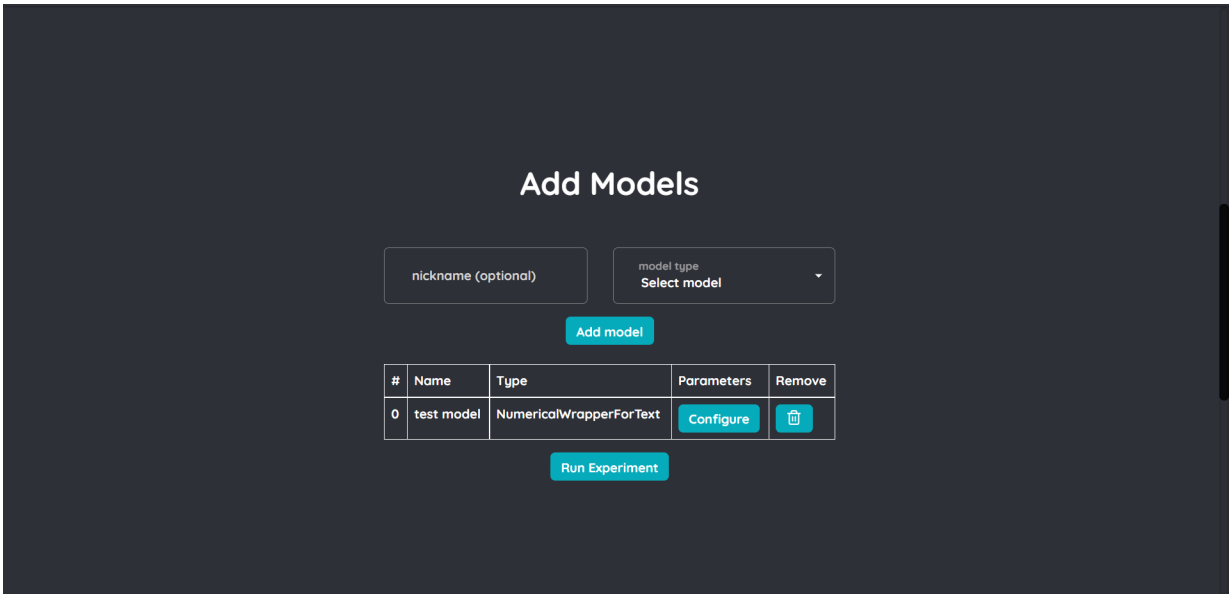

Figura 4.10: Modelo agregado al experimento. Se ha elegido como *nickname*: *test model* y como tipo: *NumericalWrapperForText*, un objeto que utiliza modelos numéricos luego de aplicar un preprocesamiento a un set de palabras.

Una vez agregado un modelo se tienen dos opciones: ejecutar el experimento (presionando el botón *Run Experiment*) utilizando parámetros por defecto, o bien, configurar parámetros.

Debido a que ejecutar el experimento se explorará en la siguiente subsección, se cubre ahora la parte de configurar parámetros del modelo. Para acceder a la vista de configuración de parámetros se debe simplemente presionar el botón *Configure*, una vez presionado este botón se muestra lo siguiente.

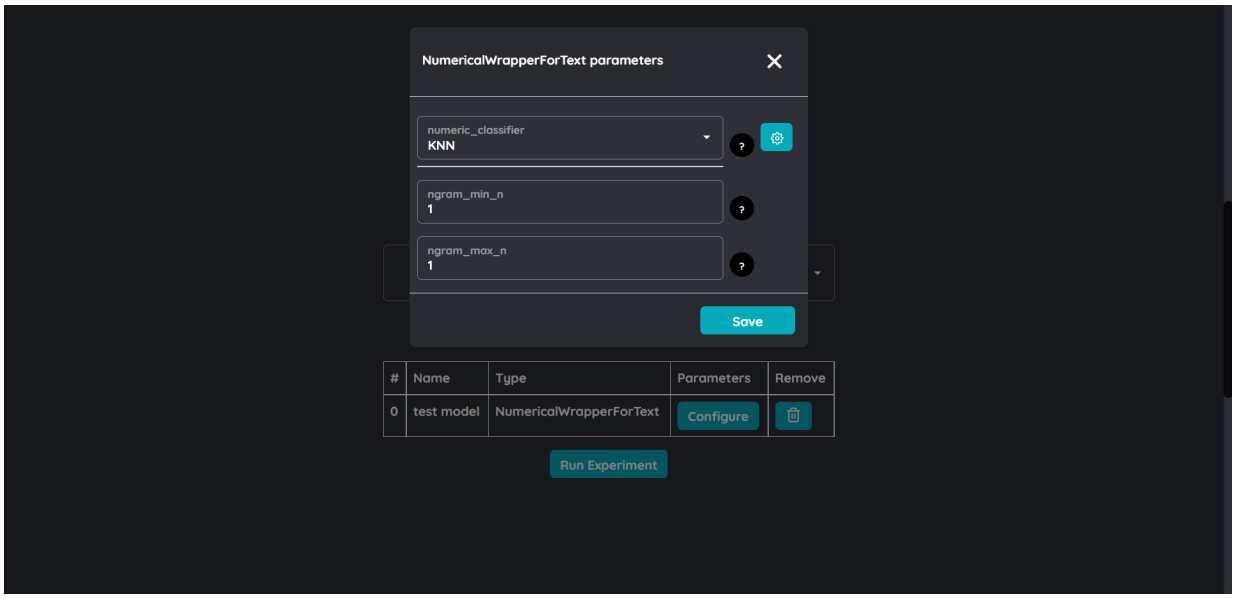

Figura 4.11: Formulario para configurar parámetros de *NumericalWrapperForText*

Como se puede apreciar el formulario consta de tres parámetros, dos numéricos (*ngram_min_n* y ngram_max_n) y uno el cual es de tipo *object*.

Un parámetro de tipo *object* significa que es un parámetro que, a su vez, cuenta con sus propios parámetros configurables es un "objeto configurable". En adelante, se refiere a este tipo de parámetros como "parámetros recursivos" debido a que estos parámetros contienen formularios dentro de otros formularios. Utilizando estos "parámetros recursivos" es posible proporcionar al usuario de flexibilidad en cuanto a configuración que se adapta a sus conocimientos. En caso de que un usuario quiera configurar hasta el último detalle puede ir más profundo en el formulario utilizando los botones con un ícono de engranaje. Mientras que si solo quiere una configuración de parámetros más general puede ignorarlos.

Este formulario, incluyendo los parámetros recursivos, se ha implementado utilizando una combinación de la biblioteca *formik* y los *hooks* de *React*. La biblioteca *formik* se ha utilizado por ser muy conocida para manejar formularios con componentes controladas. Además, esta biblioteca cuenta con sus propios *hooks* los cuales fueron muy útiles para adaptar todos los ajustes necesarios para la implementación de un formulario que puede tener varios otros formularios dentro.

Ahora bien, para acceder a los parámetros correspondientes a un parámetro recursivo se debe presionar el botón con un icono de engranaje que se encuentra al costado de cada parámetro recursivo.

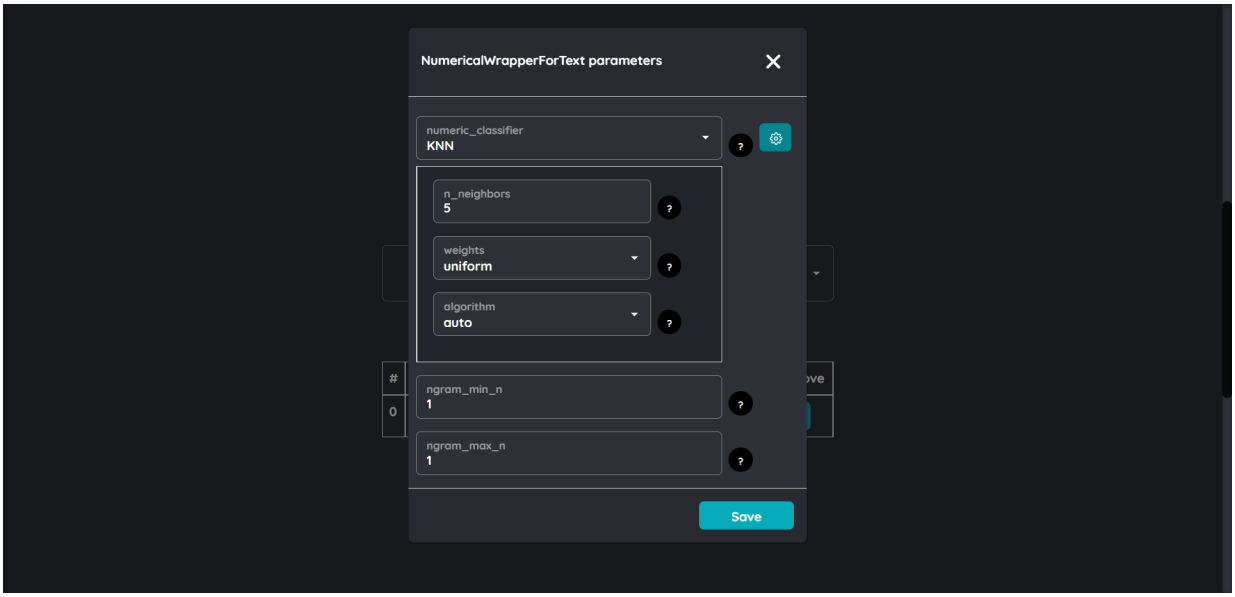

Figura 4.12: Formulario para *NumericalWrapperForText*, en el cual se muestran a su vez, los parámetros de un modelo *KNN*.

En caso de no tener demasiado conocimiento del modelo que se está utilizando se cuenta con una pequeña ayuda. Al posar el cursor sobre el ícono con un signo de interrogación (?) se muestra un *tooltip* con una descripción del parámetro en cuestión tal como se puede observar en la siguiente imagen.

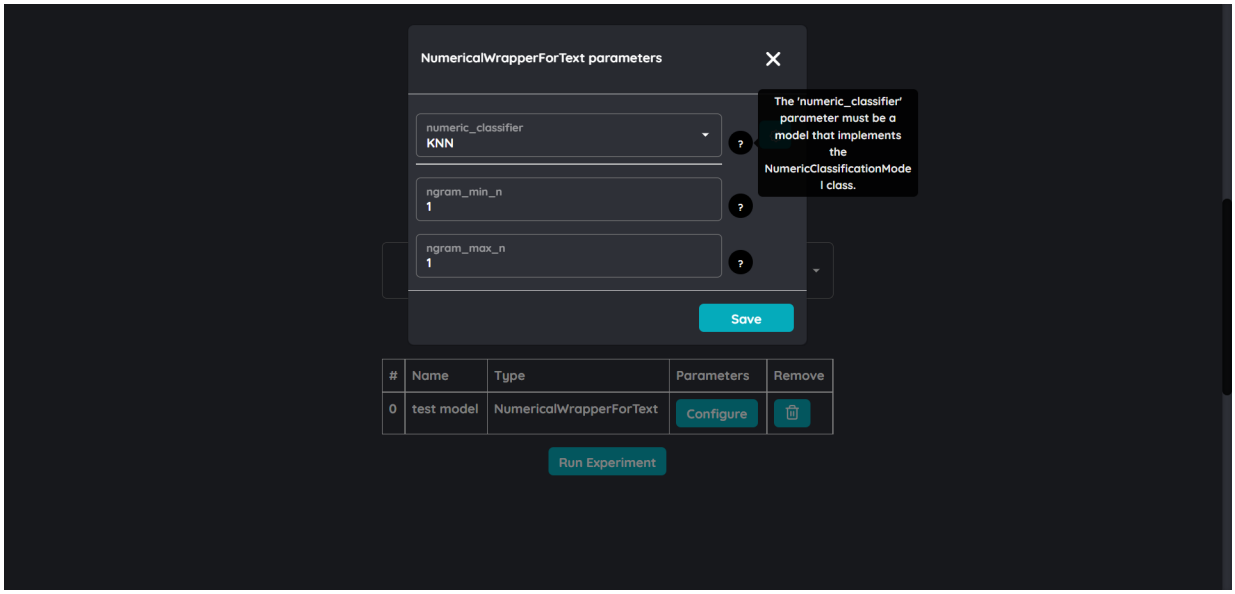

Figura 4.13: Tooltip que despliega una descripción del parámetro *numeric_classifier*.

Para complementar a los *tooltip* en cuanto a formas de guiar al usuario se cuenta también con validación del formulario, la cual se encuentra implementada con la biblioteca *Yup*, debido a que se complementa muy bien con *formik*.

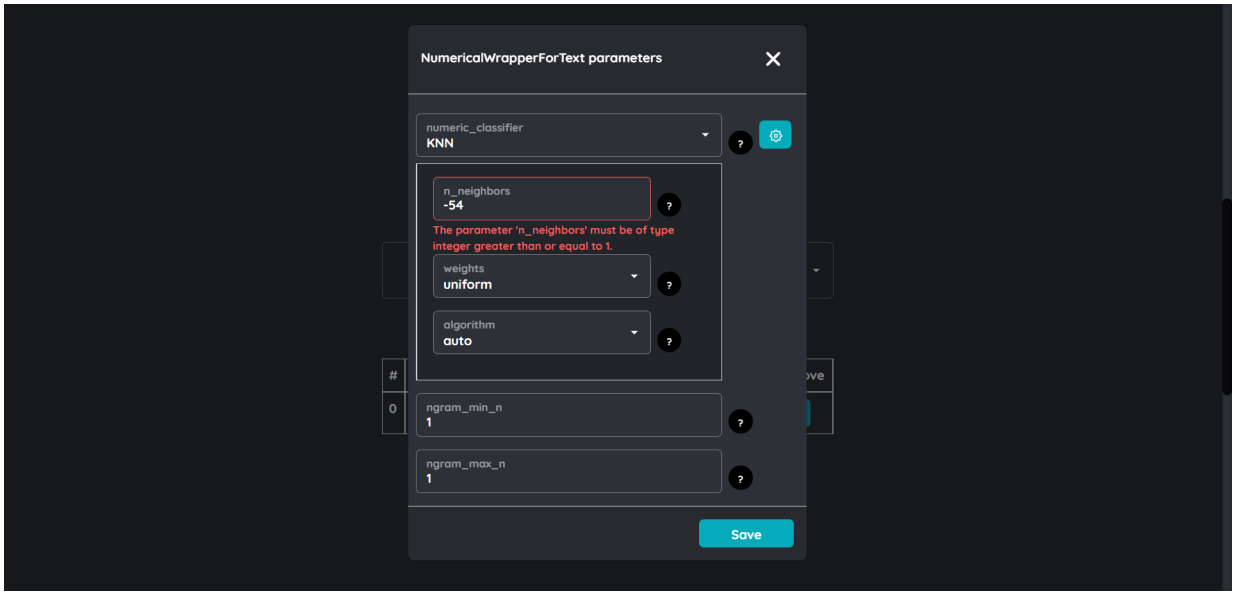

Figura 4.14: Ejemplo de validación del formulario, se muestra que solo se permiten números enteros mayores o iguales a 1.

Volviendo a la tabla de modelos, es posible agregar al experimento distintos modelos, ya sean del mismo tipo pero con distintos parámetros, o bien de tipos distintos.

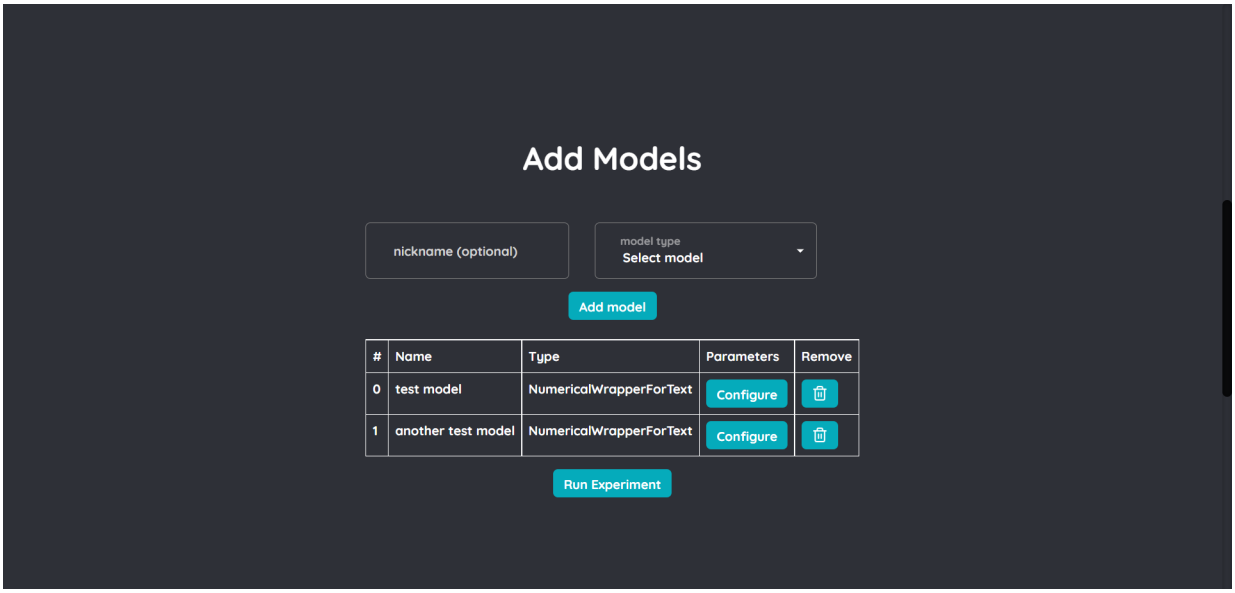

Figura 4.15: Se ha agregado otro modelo, de la misma clase (*NumericalWrapperForText*) a la tabla.

Es necesario destacar que el formulario de parámetros, junto con su validación y guías para el usuario ha sido la parte más compleja de implementar, esto se debe a que debe adaptarse a todos los modelos que contempla la aplicación.

Como se ha explicado en secciones anteriores, la *API* guarda objetos para representar modelos, estos objetos contienen los parámetros del modelo, las restricciones que contienen estos parámetros y una pequeña descripción como guía para utilizarlos. El formulario de parámetros debe ser capaz de tomar cada uno de estos objetos y "traducirlo" a elementos de la interfaz de usuario, ya sea renderizar distintos tipos de entrada (numérica, texto, selector) o bien renderizar otro formulario completo puesto que se encuentra con un parámetro recursivo.

Esta sección es la más capaz de todas para adaptarse a nuevas adiciones en la *API*, incluyendo añadir nuevos modelos a esta, lo que contribuye a facilitar la extensibilidad de la herramienta.

Para finalizar esta subsección se listan los requisitos que aborda esta componente:

1. **Un usuario puede determinar el tipo de los modelos que desea utilizar**
3. **Un usuario puede determinar valores para los parámetros del modelo**
4. **Un usuario puede incluir múltiples modelos en un mismo experimento**
6. **Un usuario puede incluir en un experimento modelos del mismo tipo con distinta configuración de parámetros**
10. **La interfaz muestra las consecuencias de las acciones del usuario**

4.6.3. Resultados del experimento

En el paso descrito en la subsección anterior (Configurar experimento), se debe presionar el botón *Run experiment*. Este botón realiza un *scroll* automático, el cual, lleva a la siguiente vista que muestra que los modelos se están entrenando.

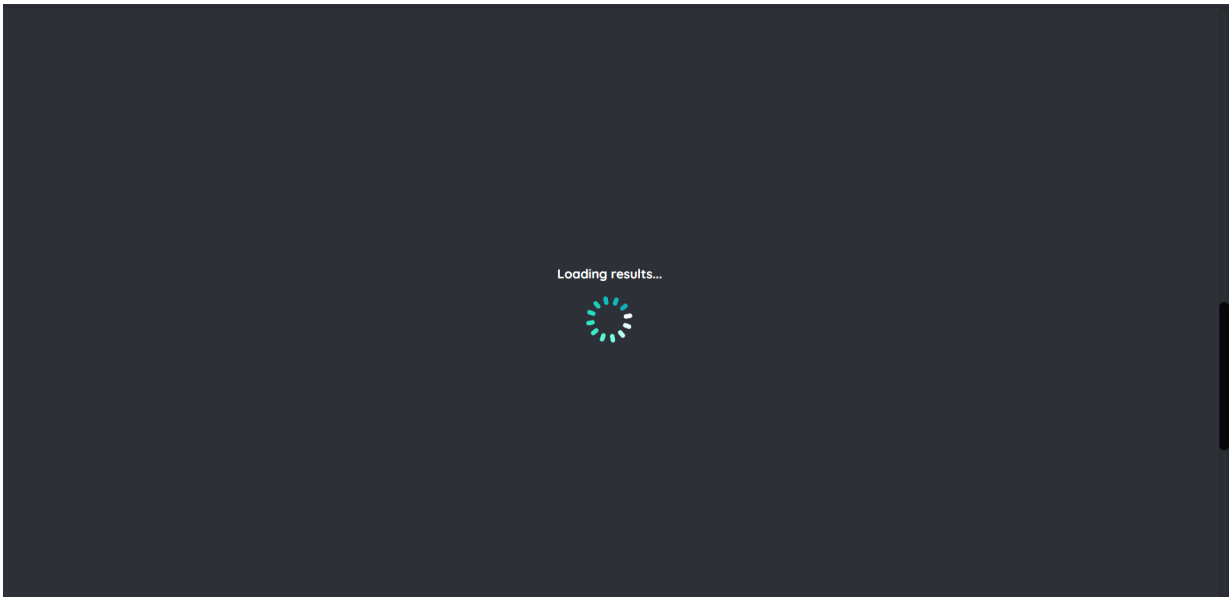

Figura 4.16: Vista para que el usuario espere mientras sus modelos son entrenados.

Una vez el entrenamiento ha terminado, se muestran los resultados, tal como se puede apreciar en la siguiente imagen.

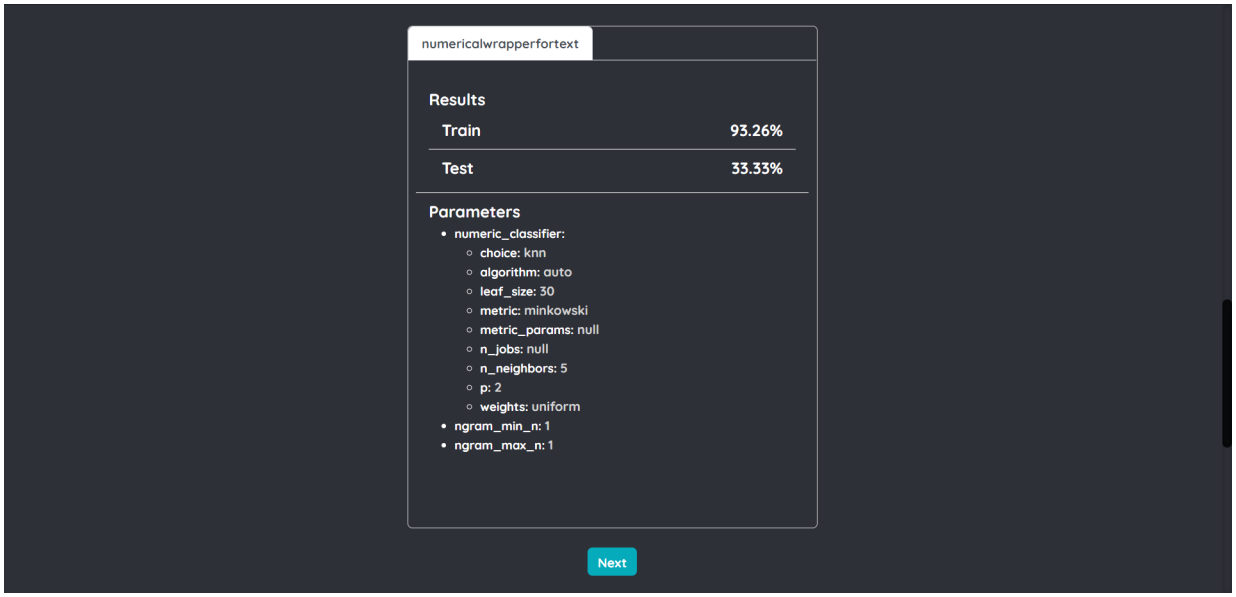

Figura 4.17: Se muestran los resultados de un modelo *NumericalWrapperForText*. En este caso, las métricas corresponden a *accuracy*.

La visualización de resultados contiene dos secciones:

- La sección "Results" que muestra las métricas que tenga el modelo. En este caso, se muestra el resultado del método *score* de *Scikit-learn*, se verá más adelante que es posible mostrar diferentes métricas dependiendo del modelo. Esto muestra que, al igual que el formulario de parámetros, la componente de resultados tiene un grado de adaptabilidad para los modelos que existen actualmente en la interfaz, pero también para los que se implementarán en el futuro.
- La segunda sección corresponde a los parámetros. En esta se muestra una lista anidada (para parámetros recursivos) con los valores de los parámetros utilizados para cada modelo entrenado.

En lo siguiente se muestran algunas variaciones de la visualización de resultados dependiendo del experimento. Para comenzar, se muestra la variante de visualización para cuando se entrena más de un modelo.

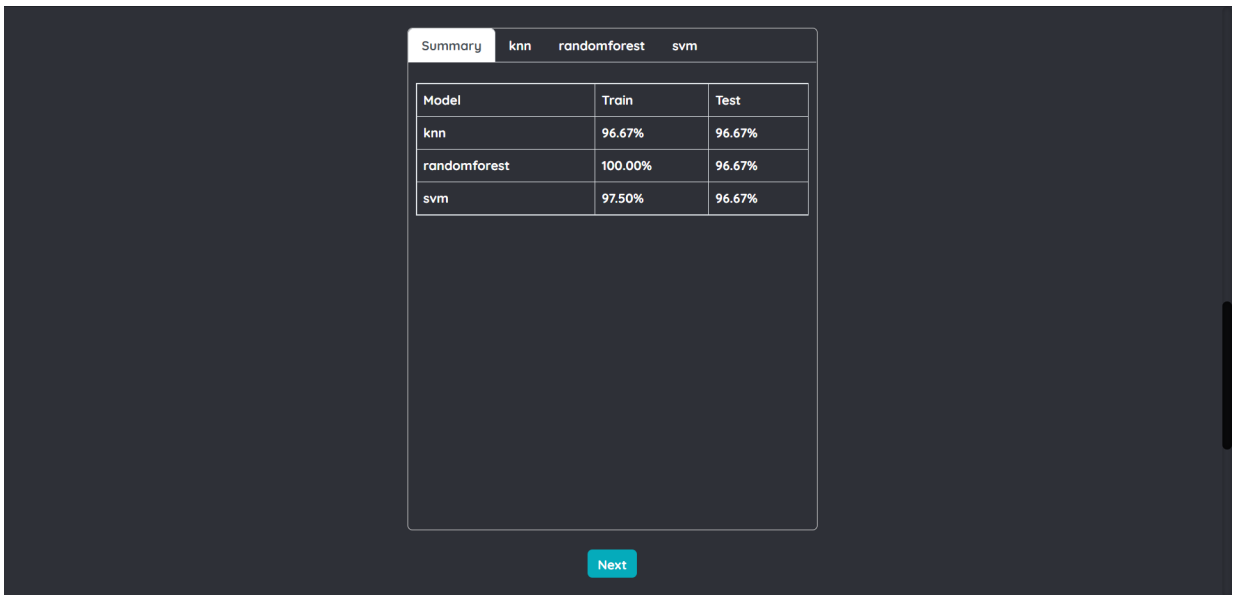

Figura 4.18: Visualización de resultados para varios modelos. Se muestra un resumen (*Summary*), de la principal métrica de los modelos, con la cual pueden ser comparados entre sí.

Para acceder al tipo de visualización que muestra la figura (4.17) se puede seleccionar la *tab* que corresponda al modelo que se quiere ver en detalle.

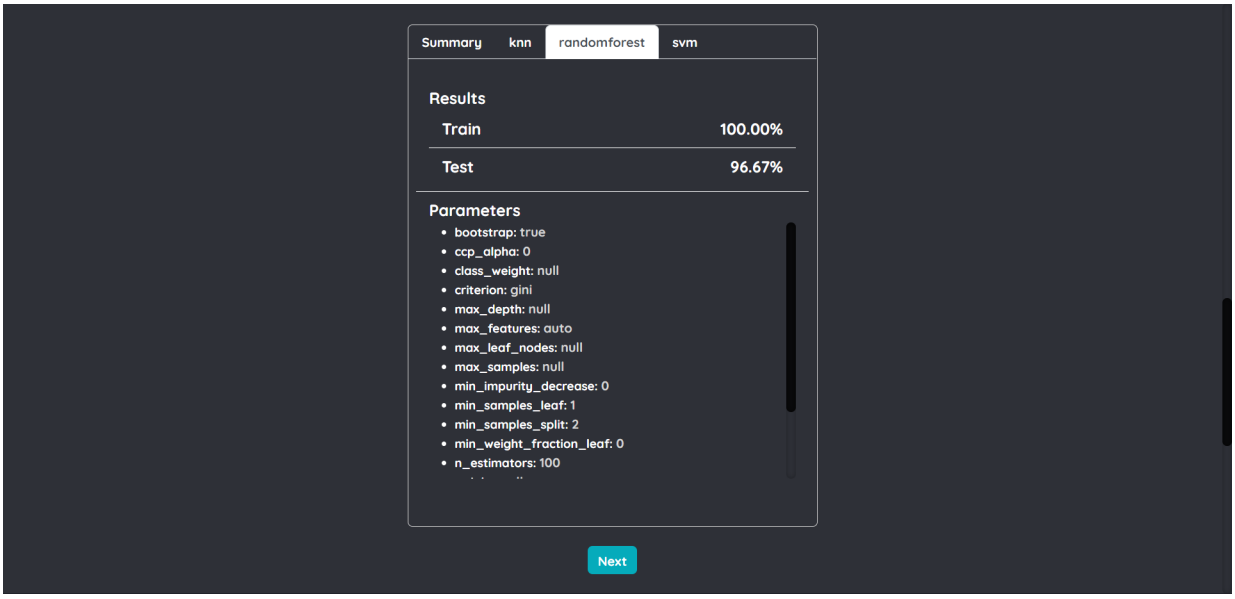

Figura 4.19: Visualización para múltiples modelos al seleccionar uno de los modelos para ver en detalle. En este caso, se ha seleccionado *random forest*.

Por último se muestra la capacidad de la visualización de resultados para adaptarse a diferentes modelos y métricas, en este caso, se muestran las métricas para un modelo de traducción automática (*"Helsinki-NLP/opus-mt-en-es"*).

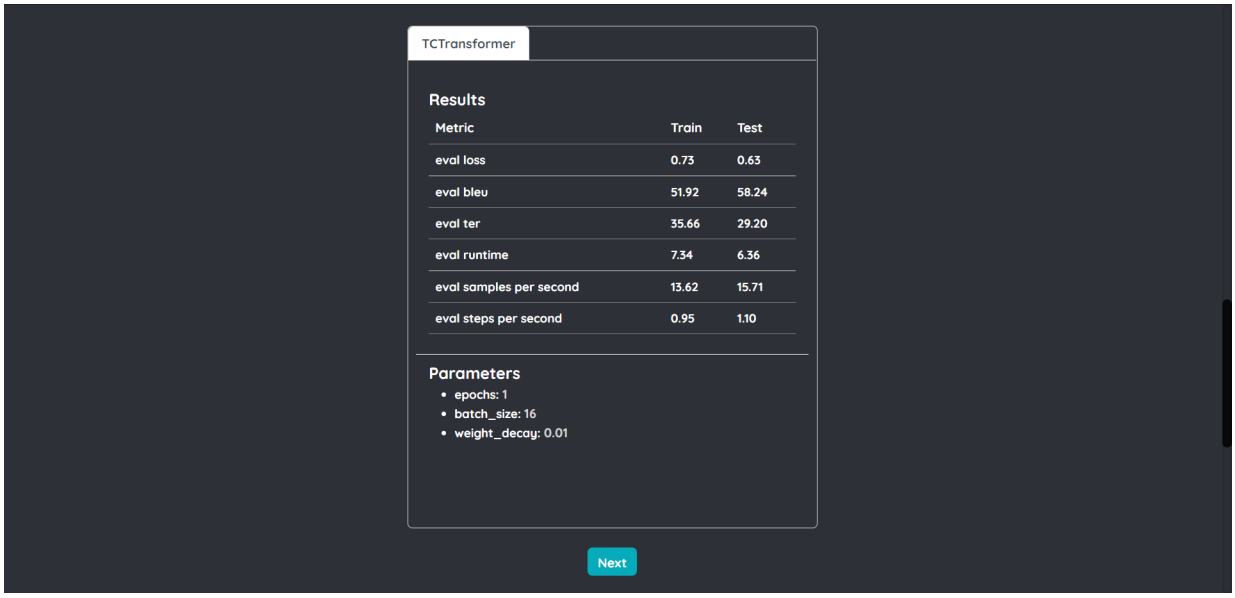

Figura 4.20: Visualización de diferentes métricas para un modelo *"Helsinki-NLP/opus-mt-en-es"*.

La componente de resultados aborda los siguientes requisitos:

5. **Un usuario puede comparar los resultados de distintos modelos.**

4.6.4. Probar el modelo

Para finalizar, se muestra la última de las cuatro componentes, la cual está dedicada a probar el modelo entregándole *inputs* y observando los *outputs* que devuelve. Lo primero que se observa luego de presionar el botón *Next* visible en la figura (4.20) es lo que sigue.

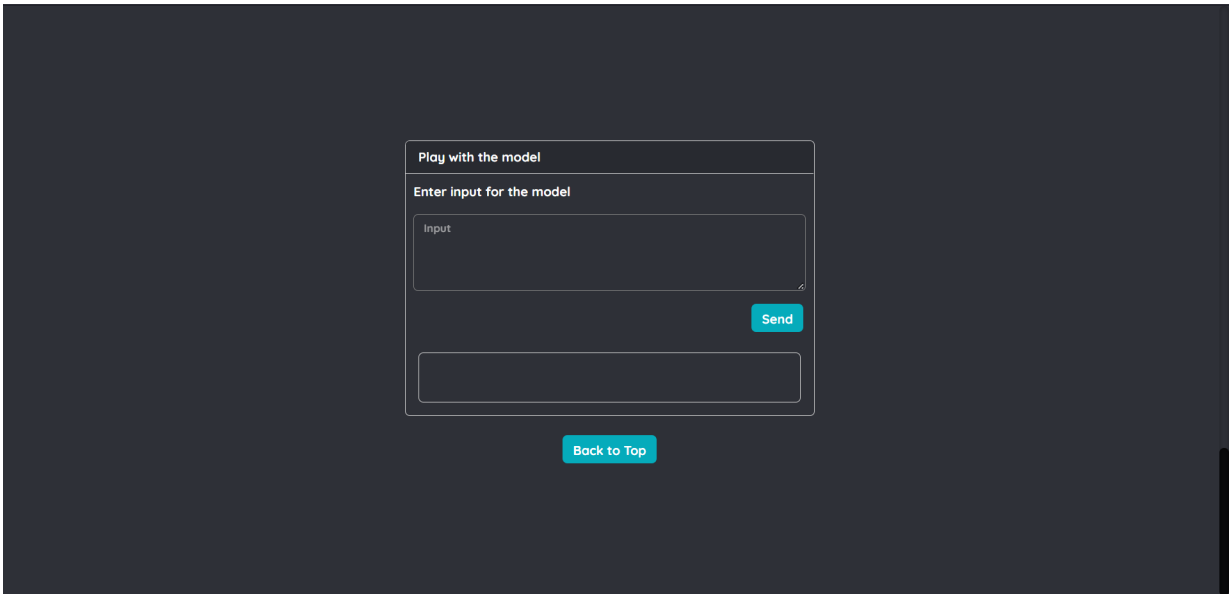

Figura 4.21: Vista para probar el modelo, se muestra un elemento *textarea* para entregar un *tweet* que el modelo clasificará como un sentimiento positivo o negativo según corresponda.

Al escribir un ejemplo de un *tweet* en inglés y presionar el botón *Send* se muestra la respuesta del modelo.

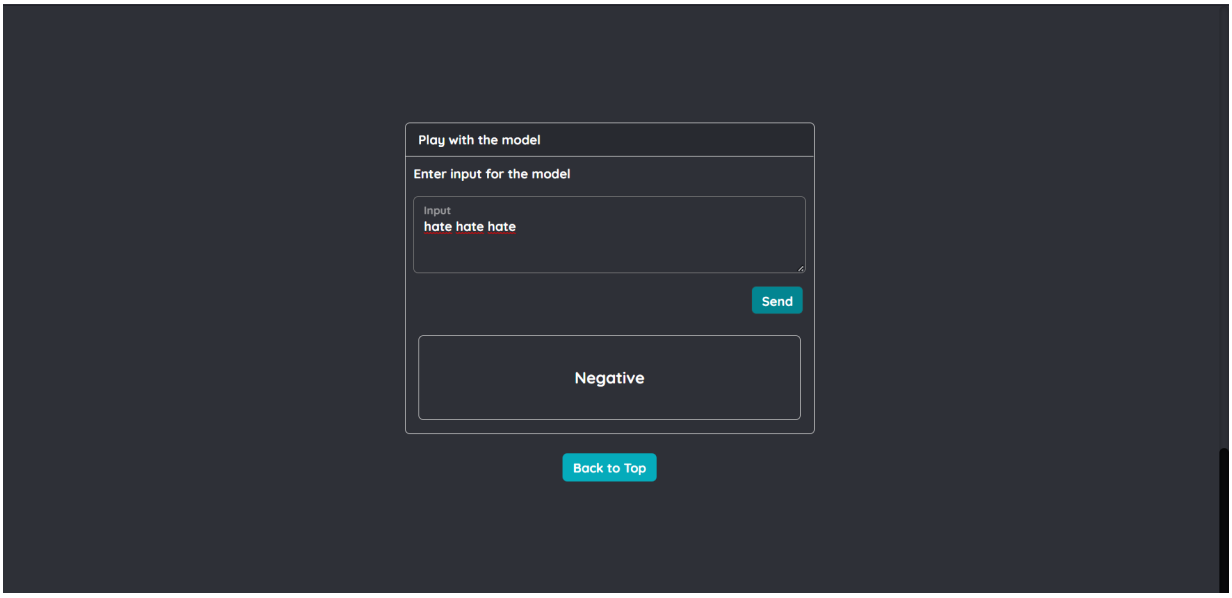

Figura 4.22: Vista para probar el modelo *NumericalWrapperForText* entrenado con datos de *twitter*. En este caso, se ve que clasifica el *input hate hate hate* como *Negative*.

Al realizar otro experimento, con un modelo traductor de inglés a español se muestra la siguiente visualización para probar el modelo.

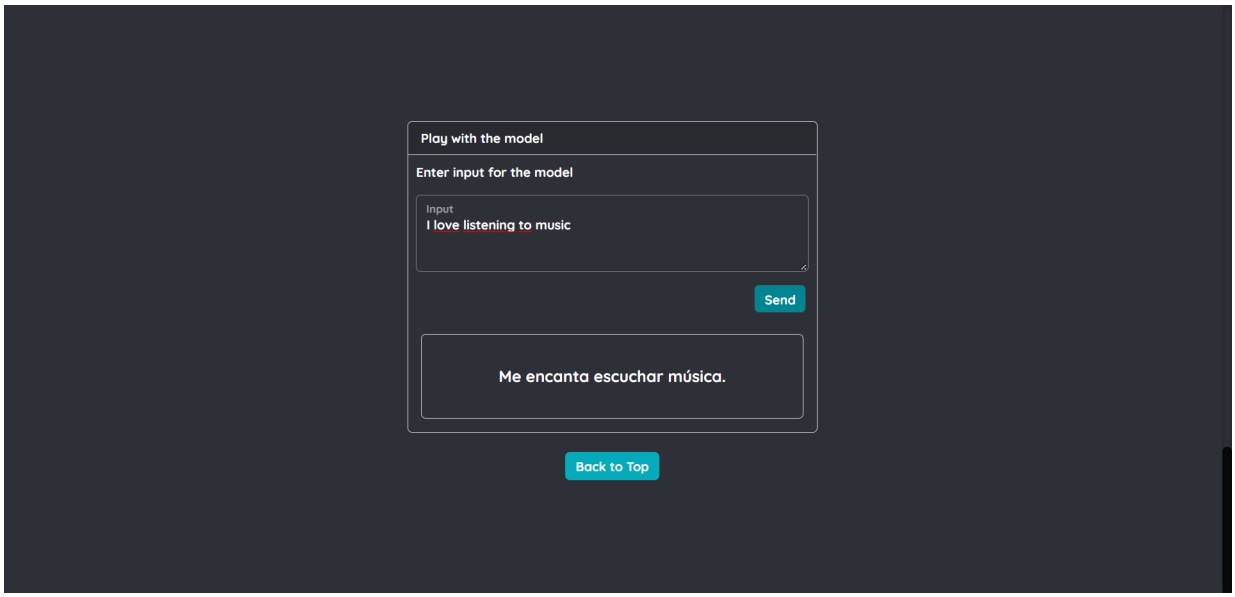

Figura 4.23: Al entrenar un modelo traductor *"Helsinki-NLP/opus-mt-en-es"*. y entregarle un texto en inglés se muestra lo que el modelo devuelve como traducción al español.

Esta componente ha abordado los siguientes requisitos:

14. **Un usuario puede probar modelos previamente entrenados**

Para finalizar, se toma en cuenta una última tarea cumplida que se ha hecho presente en toda la interfaz. Es la que respecta a que el usuario puede corregir sus errores. Esta tarea se ve cumplida al ver, por ejemplo, la figura 4.8 pues para enmendar el error basta simplemente presionar un botón. Otro ejemplo puede ser la figura 4.14 donde se le muestra al usuario el error que ha cometido con los valores de los parámetros y para enmendarlo solo debe introducir un valor correcto. Con esto, se cumple también la tarea:

8. **Un usuario puede corregir sus errores sin perder su progreso.**

4.6.5. Sobre la variedad de modelos en la interfaz gráfica

Antes de finalizar esta sección es necesario realizar una observación. Cuando se establecieron las ventajas de *DashAI* sobre otros *software* que abordan el mismo problema se comentó como una de sus ventajas que es capaz de implementar modelos recientes. Sin embargo, en esta sección se han presentado mayoritariamente modelos tradicionales (*KNN*, *SVM* y *Random Forest*). La razón de esto es que, como fue comentado en la sección 4.1 "Situación actual", el *backend* de *DashAI* se encuentra en un estado muy inicial en cuanto a desarrollo. Esto quiere decir, que por ahora solo cuenta mayoritariamente con modelos tradicionales debido a

su simplicidad de implementación, teniendo en mente agregar modelos más recientes según avance el desarrollo del *backend*.

Aún así, se ha implementado la interfaz gráfica considerando la etapa temprana de desarrollo en la que se encuentra el *backend*, de manera que esta pueda adaptarse según la cantidad de modelos crezca. Un ejemplo que comprueba esto se encuentra en el *transformer* "*Helsinki-NLP/opus-mt-en-es*" el cual corresponde a un modelo reciente que no se encontraba inicialmente en el *software* sino que fue agregado con un desarrollo paralelo pero independiente al de la interfaz gráfica. Se comprobó que para agregar este *transformer* fue necesario realizar casi la totalidad de los cambios del *software* en el *backend*. Mientras que la interfaz, realizando arreglos menores, fue capaz de recibir la información necesaria para construir su formulario de parámetros, desplegar sus métricas y realizar pruebas entregándole *inputs*.

Lo descrito anteriormente apunta hacia el objetivo de *DashAI* de poder integrar tanto modelos tradicionales como modelos más recientes.

Capítulo 5

Evaluación

En este capítulo se presenta la evaluación de la solución que se ha expuesto en detalle en el capítulo anterior. Se comienza con una revisión de las tareas planteadas en el capítulo 3 (Problema), los que fueron cumplidos, los que no y a qué se debió esto.

Para evaluar la solución se ha utilizado una prueba de usabilidad. Se presentan los resultados utilizando diagramas de afinidad, además de tablas y gráficos que resumen toda la información relevante que se ha podido obtener de las pruebas.

5.1. Evaluación de tareas

En la tabla 3.1 se mostraron las tareas que se tenía como meta cumplir, y en el capítulo anterior se mostraron las componentes de la interfaz mientras se las asociaba con las tareas a las cuales estaban orientadas. Pues bien, se presenta a continuación una tabla que muestra las tareas que se cumplieron y las que no.

Tarea
Un usuario puede determinar la clase de los modelos que desea utilizar
Un usuario puede usar un experimento creado previamente
Un usuario puede determinar valores para los parámetros de un modelo
Un usuario puede incluir múltiples modelos en un mismo experimento
Un usuario puede comparar los resultados de distintos modelos
Un usuario puede incluir en un experimento modelos de la misma clase con distinta configuración de parámetros
La interfaz muestra información relevante en todo momento
Un usuario puede corregir sus errores sin perder su progreso
La interfaz recuerda interacciones recientes
La interfaz muestra las consecuencias de las acciones del usuario
La interfaz provee controles globales
Un usuario puede determinar la task con la que realizará un experimento
Un usuario puede determinar el set de datos que usará en un experimento
Un usuario puede jugar con modelos previamente entrenados

Figura 5.1: Se puede observar que se han cumplido un total de 11 de las 14 tareas totales

Se puede observar en la figura 5.1 que se han cumplido un total de 11 de 14 tareas, esto

corresponde a un total de 78 % de tareas cumplidas, el cual es superior a un 75 %, por lo cual, se verifica que se aprueba la evaluación de requisitos.

En cuanto a las justificaciones de las tareas no cumplidas se destaca que son, en general, tareas menos importantes que no se pudo cumplir pues se subestimó la complejidad de algunas otras tareas, presentando complicaciones con los plazos a cumplir.

5.2. Prueba de usabilidad

Se describe en lo siguiente, los usuarios reclutados, la metodología de la prueba y se presentan sus resultados.

5.2.1. Reclutamiento

La prueba de usabilidad se ha realizado gracias a la ayuda de 5 voluntarios los cuales fueron reclutados utilizando el foro de la Universidad de Chile, *U-Cursos*.

Los usuarios reclutados tienen algunos puntos en común:

- Son estudiantes (pregrado y magíster), hombres de entre 20 y 30 años.
- Todos tienen al menos conocimiento básico de inteligencia artificial, junto con experiencia en el área de al menos 1 año.
- Todos dominan el inglés.
- Para todos era su primera vez interactuando con *DashAI*

Justificación cantidad de usuarios

Considerando el contexto en el que se realiza este test de usabilidad:

- **Estado de desarrollo**: Como se ha mencionado anteriormente, *DashAI* cuenta actualmente con solo 5 modelos (3 modelos de clasificación numérica, 1 de clasificación de texto y 1 de traducción de texto). El hecho de ofrecer poca variabilidad de modelos, y con esto, de tareas que el usuario puede realizar, hace que sea beneficioso posponer un test de usabilidad que evalúe a un mayor número de usuarios.
- **Alcances del proyecto**: Considerando que la interfaz gráfica es el producto de un trabajo de título de pregrado y que además es la primera interfaz gráfica desarrollada para *DashAI*. Se decide que realizar una primera prueba con una cantidad baja de usuarios sería beneficioso para detectar tempranamente sus errores y problemas más evidentes.

- **Recursos** Para este punto se toma en cuenta que no se dispone de muchos recursos para reclutamiento (por ejemplo, pagar a los usuarios). A esto se añade que se necesitan usuarios con al menos un conocimiento básico de inteligencia artificial y que ya fue complicado encontrar 5 usuarios que, de forma voluntaria, aceptaran tener una reunión de 40 minutos para realizar la prueba.

Por estas razones y que, además, se tienen estudios que muestran que un test de usabilidad con 5 personas puede encontrar, en promedio, el 85 % de los errores de una interfaz[2] (y también siguiendo la "suposición de los 5 usuarios"[1]). Se decide realizar esta prueba de usabilidad con baja cantidad de usuarios para evaluar el *software* en una etapa temprana y establecer los próximos objetivos del proyecto. Se añade a esto que más adelante se propone como trabajo futuro cuando se obtenga un *backend* más completo, realizar una prueba de usabilidad con mayor cantidad de usuarios.

5.2.2. Estructura

A continuación se da una descripción de la metodología usada para la prueba, si se quiere profundizar en algún punto el lector puede revisar *Anexo B*, donde se exponen puntos más extensos como, por ejemplo, el guión que se ha utilizado para que la prueba fuese lo más similar posible con todos los usuarios.

Introducción

El entrevistador se presenta como estudiante realizando la evaluación de su trabajo de título, entrega una breve descripción de la herramienta y una vista general de en lo que consistirá la prueba de usabilidad.

Entrevista preliminar

Primero se hace una entrevista al usuario con preguntas para caracterizarlo y también, en parte, para que pueda comenzar a tener más confianza con el entrevistador y así poder dar respuestas más sinceras. Preguntas del estilo *¿A qué te dedicas?, ¿hace cuánto te iniciaste en el área de inteligencia artificial?*.

Tareas

Lo siguiente es dar acceso a la interfaz de *DashAI*, para lo cual se utilizó *Google Remote Desktop*[2], de esta forma el usuario era capaz de conectarse y controlar el equipo del entrevistador que estaba ejecutando localmente la herramienta *DashAI*.

[1] https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/
[2] https://remotedesktop.google.com/support/

Una vez hecho esto se pasaba a resolver las tareas, las cuales eran 3:

1. Utilizando el set de datos *Iris* (clasificación numérica) se pide al usuario entrenar tres modelos distintos con sus parámetros por defecto y luego elegir el que, según los datos proporcionados por la interfaz gráfica, daba los mejores resultados.
    - **Tiempo límite**: 480 segundos (8 minutos)
    - **Objetivo**: la tarea se considera exitosa cuando el usuario comunica por voz el modelo que considera el mejor de los 3 que ha entrenado.
2. Utilizando un set de datos de *twitter* (clasificación de sentimientos, clasificación de texto) se pide al usuario entrenar un modelo, esta vez, configurando sus parámetros como este desee para, finalmente, probar el modelo entregando un *input*.
    - **Tiempo límite**: 360 segundos (6 minutos)
    - **Objetivo**: la tarea se considera exitosa cuando el usuario ha entregado al menos un *input* al modelo y ha recibido el resultado de este.
3. Utilizando un set de datos de traducción (inglés-español) se pide al usuario entrenar (*fine tuning*) un traductor (generación de texto) para finalmente, al igual que en la tarea anterior, entregar un *input* al modelo.
    - **Tiempo** límite: 360 segundos (6 minutos)
    - **Objetivo**: la tarea se considera exitosa cuando el usuario ha entregado al menos un *input* al modelo y ha recibido el resultado de este.

Cada uno de los tiempos ha sido elegido midiendo el tiempo que le toma al desarrollador de la interfaz completar cada tarea, para posteriormente multiplicarlo por un ponderador que se basa en la dificultad de la tarea. Finalmente, se redondea para obtener números cerrados. Esta forma de establecer tiempos fue obtenida desde las referencias que se usaron para diseñar el test de usabilidad [12].

- Tarea 1: Se utiliza un ponderador de 4 pues es el primer contacto con la interfaz y se pide entrenar 3 modelos.
- Tarea 2: Se utiliza un ponderador de 3 considerando que el usuario ya tuvo su primer contacto con la interfaz.
- Tarea 3: Se mantiene el ponderador de 3. Pues aunque el usuario ya tiene más experiencia con la herramienta, se considera que se trata de una tarea más compleja y que demora más en entrenar (traducción de texto).

Preguntas finales

Se realiza una ronda de preguntas finales, en las cuales, el entrevistador podía enfocar la atención del usuario en ciertas componentes de la interfaz para obtener *feedback* sobre estas

(e.g Opinión sobre la complejidad del formulario para configurar parámetros), además de obtener información para guiar la dirección del proyecto en su totalidad (e.g opinión sobre si *DashAI* debiese ser local o debiese estar en un servidor).

Cuestionario

Finalmente, se pide al usuario que llene el cuestionario *NASA-TLX* según la experiencia que tuvo al utilizar *DashAI*. Las razones para usar este cuestionario son principalmente el hecho de que su evaluación está muy alineada con los objetivos de la interfaz de *DashAI*. Por ejemplo, el cuestionario mide exigencia mental, exigencia temporal, exigencia física, esfuerzo, frustración y rendimiento. Todas estas categorías son muy importantes para evaluar si una interfaz es usable. La segunda razón es que *NASA-TLX* es un cuestionario corto (se debe asignar un puntaje entre 1 y 20 a 6 categorías) que no es muy agotador de responder.

5.2.3. Resultados

Se presentan ahora los resultados de la prueba de usabilidad. En primer lugar, se da una caracterización de los usuarios que tomaron la prueba, luego se muestran tablas que resumen el desempeño en tareas de los usuarios. Lo siguiente es presentar los resultados para todos los usuarios del cuestionario *NASA-TLX*. Finalmente, se presenta un diagrama de afinidad donde se resume todas las notas obtenidas mediante observación durante la resolución de tareas y el *feedback* de los usuarios durante las preguntas finales.

Caracterización usuarios

Los usuarios tienen entre 1 y 5 años de experiencia en el área de inteligencia artificial, el promedio para todos ellos es de 2.4 años de experiencia. De los 5 participantes, tan solo 2 usaron anteriormente interfaces gráficas para realizar tareas de inteligencia artificial sin tener que programar. Estos datos pueden observarse en la tabla a continuación:

Usuario	**Tiempo experiencia**	**¿Ha usado interfaces gráficas de machine learning?**
U1	2 años	No
U2	5 años	Sí
U3	1 año	No
U4	3 años	Sí
U5	1 año	No

Tabla 5.1: Caracterización de cada uno de los usuarios que participaron en el test de usabilidad.

Tareas

Para comenzar la evaluación se aborda la medición de la eficacia de la interfaz diseñada. Para esto, se presenta la siguiente tabla en la que se muestra los usuarios que lograron el objetivo de cada tarea independiente del tiempo que les haya tomado. Este objetivo ha sido expuesto anteriormente en la sección 5.2.2 "Estructura"

Tarea	**U1**	**U2**	**U3**	**U4**	**U5**	**% éxito**
Entrenar 3 modelos y elegir el mejor (T1)	1	0	1	0	1	60 %
Entrenar un clf de texto y entregar input (T2)	1	1	1	1	1	100 %
Entrenar un traductor y entregar input (T3)	1	1	1	1	1	100 %

Tabla 5.2: Desempeño de los usuarios de acuerdo a las tareas que completaron con éxito, se utiliza para evaluar eficacia. En este caso 1 significa éxito, mientras que 0 significa fracaso. (clf es una abreviación para clasificador).

Se observa que para todas las tareas se obtiene una aprobación mayor al 80 %, excepto para la primera, la cual obtiene una aprobación del 60 %, debido al criterio de aceptación en esta parte la interfaz falla. Esto se puede deber a lo siguiente:

- Existió cierta confusión en la tarea 1, pues esta no incluía el paso de probar el modelo. Sin embargo, la mayoría de los usuarios intentó realizar este paso llevando a varias confusiones que desencadenaron en que dos usuarios fallaran la prueba.
- Para los dos usuarios que fallaron la prueba, coincidió que ambos intentaron entrenar los modelos uno por uno en vez de todos a la vez, lo que los llevó a una confusión.

Con los motivos identificados, aunque se haya fracasado en este apartado, se tiene un indicio de cómo mejorar en este aspecto.

Continuando con la evaluación, se presenta ahora la siguiente tabla que evalúa el desempeño de los usuarios de acuerdo al tiempo que les tomó resolver las tareas, con esto se evalúa la eficiencia de la herramienta.

Es importante destacar que en la tabla no se considera el tiempo de los usuarios que fallaron la tarea. En estos casos se les asigna el valor "*"

Tarea	**U1[s]**	**U2[s]**	**U3[s]**	**U4[s]**	**U5[s]**
Entrenar 3 modelos y elegir el mejor (T1)	234	*	235	*	175
Entrenar un clf de texto y entregar input (T2)	301	215	242	234	190
Entrenar un traductor y entregar input (T3)	182	160	145	186	134

Tabla 5.3: Desempeño de los usuarios de acuerdo al tiempo en segundos ([s]) que les tomó resolver cada tarea, se utiliza para evaluar eficiencia. (clf es una abreviación para clasificador).

Se presenta ahora, una tabla que muestra el criterio de aprobación para la mediana de los tiempos de los usuarios.

Tarea	**Mediana tiempo usuarios** [s]	**tiempo límite** [s]	**70 % de tiempo límite** [s]
T1	234	480	336
T2	234	360	252
T3	160	360	252

Tabla 5.4: Contraste entre la mediana de los tiempos de los usuarios con el 70 % y tiempo límite para completar la tarea.

De la tabla 5.4 se observa que todas las tareas cumplen el criterio. Por lo tanto, se considera la interfaz como aprobada en cuanto a eficiencia.

Cuestionario de aspectos subjetivos

Habiendo presentado los resultados objetivos, se muestran ahora los resultados subjetivos que dan cuenta de la percepción de los usuarios sobre la interfaz evaluada.

La tabla a continuación contiene las respuestas de los usuarios para el cuestionario *NASA-TLX*

	U1	U2	U3	U4	U5	Promedio
Exigencia mental	5	6	10	4	12	7.4
Exigencia física	7	3	5	3	5	4.6
Exigencia temporal	6	10	6	4	7	6.6
Rendimiento	18	15	15	5	18	14.2
Esfuerzo	4	7	10	2	8	6.2
Frustración	1	11	5	2	3	4.4

Tabla 5.5: Tabla que muestra las respuestas de los usuarios para cada categoría del cuestionario *NASA-TLX*.

Para complementar los datos de la tabla se presenta además un gráfico de barras.

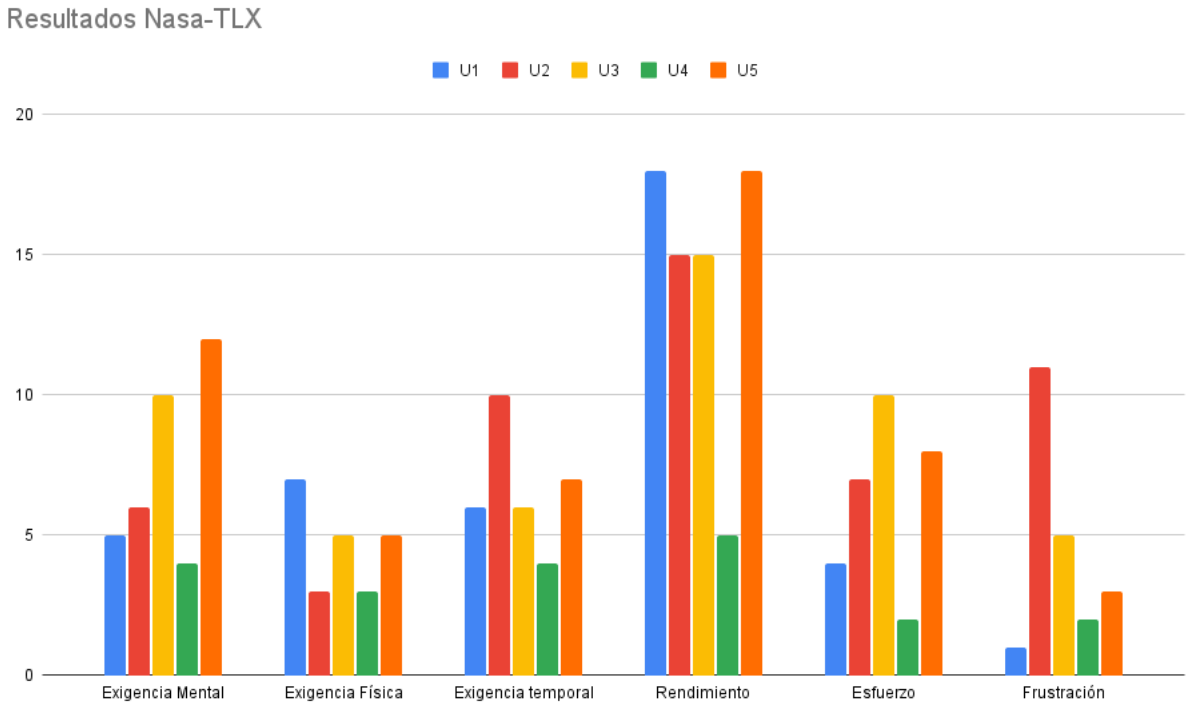

Figura 5.2: Resultados del cuestionario *NASA-TLX* de todos los usuarios

En general se obtuvieron resultados positivos en el cuestionario. Se observa que, para la mayoría de los usuarios las categorías negativas como exigencia mental, exigencia física, exigencia temporal, esfuerzo y frustración se ubican en los valores bajos de la escala, es decir, sus promedios son inferiores a 10. Mientras que la categoría positiva: rendimiento, se encuentra en los valores altos de la escala, promedio superior a 10. Por esto, se considera la solución aprobada en cuanto a sus medidas subjetivas.

Diagrama de afinidad

Se presenta en lo siguiente un diagrama de afinidad construido con las anotaciones del entrevistador durante la prueba de usabilidad. Estas notas fueron tomadas en su mayoría durante la fase de tareas y preguntas finales.
Debido a la gran extensión del diagrama, se muestra por cada grupo afín.

En primer lugar, se presenta el significado de los distintos colores de las notas del diagrama.

Figura 5.3: Colores utilizados para distinguir la categoría de cada nota.

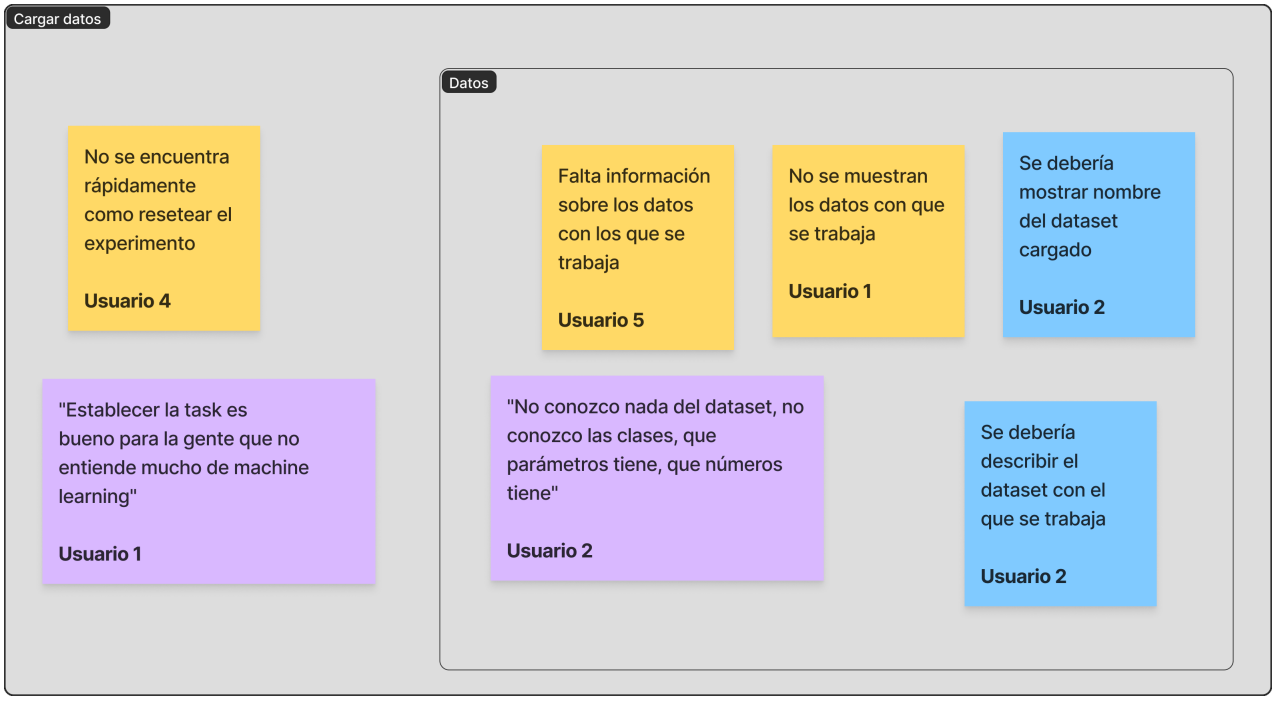

Figura 5.4: Anotaciones tomadas de los usuarios durante la prueba de usabilidad que respectan a la carga de datos.

Se puede observar en la figura 5.4 que existe una falta de información sobre los datos con los que se trabaja en el experimento. Tres de los participantes lo hicieron notar siendo una de las anotaciones más ilustrativas del problema la cita: “No conozco nada del dataset, no conozco las clases, que parámetros tiene, que números tiene”. Este problema es una consecuencia de posponer el desarrollo de un módulo de carga de datos, pero con esto, se confirma que es algo muy necesario e importante para la herramienta.

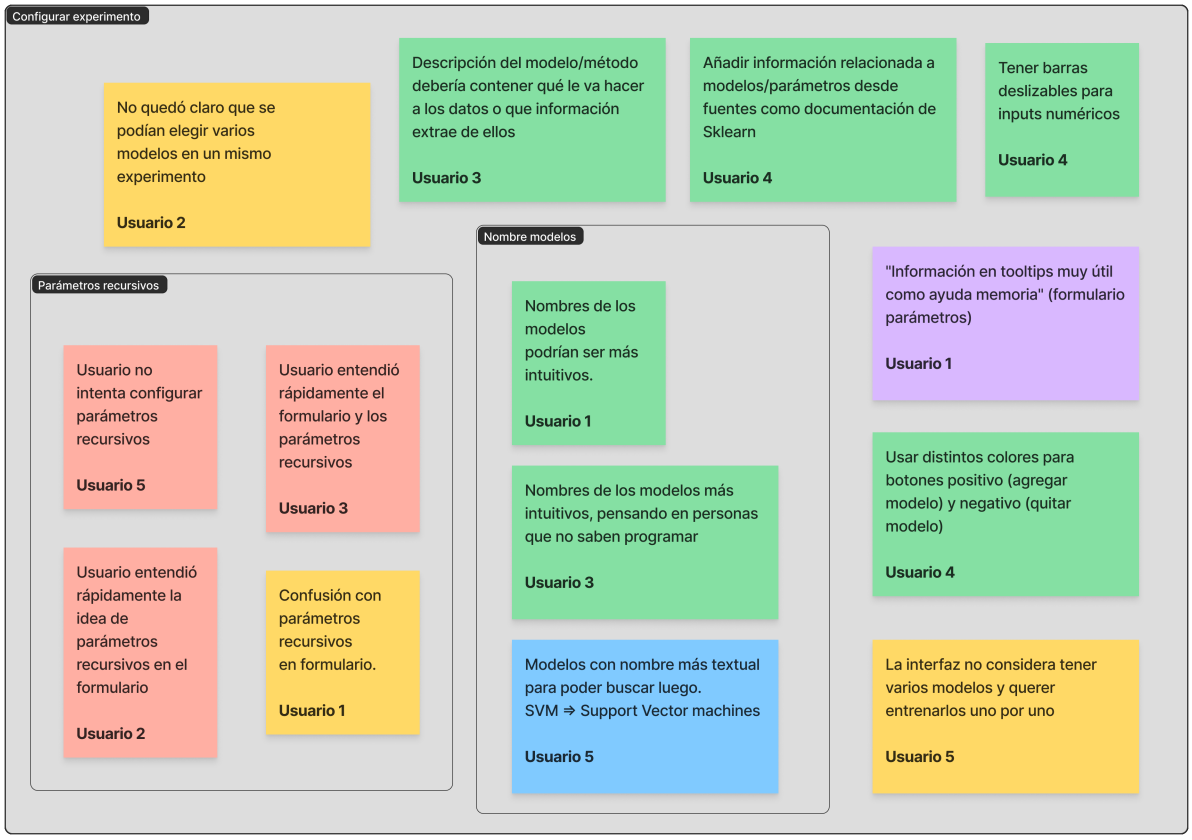

Figura 5.5: Anotaciones tomadas de los usuarios durante la prueba de usabilidad que respectan a la configuración de experimentos.

- **Parámetros recursivos**: Se observa una división en los usuarios, algunos entendieron rápidamente el funcionamiento de parámetros recursivos. Pero otros, no lograron entenderlos y, por ello, no pudieron usarlos.
- **Nombre modelos**: Usuarios manifiestan que los nombres usados para los modelos pueden ser confusos y proponen ideas para mejorar esto como usar nombres más textuales.

En cuanto al resto de anotaciones, se puede observar que los usuarios proponen mejoras con respecto a formularios como por ejemplo: “Tener barras deslizables para inputs numéricos”. Además, manifiestan que la información para guiar al usuario es de ayuda, pero que podría mejorarse agregando referencias desde documentación oficial.

Figura 5.6: Anotaciones tomadas de los usuarios durante la prueba de usabilidad que respectan a los resultados del experimento.

Los usuarios piensan que la interfaz es simple para mostrar los resultados, que tiene lo más importante e incluso se muestran contentos con "summary" para comparar los modelos entre sí. Sin embargo, una de sus principales carencias es la falta de información gráfica como evidencia la siguiente cita: "Se deberían añadir gráficos, al menos algo básico".

Por otra parte, se puede apreciar que falta *feedback* visual con respecto al entrenamiento de los modelos. En efecto, los usuarios observan una animación de carga, pero esta no les indica nada sobre el estado del entrenamiento. Esto se evidencia en la nota que cita al usuario 5 luego de esperar un rato con la animación de carga: "Supongo que se está ejecutando...".

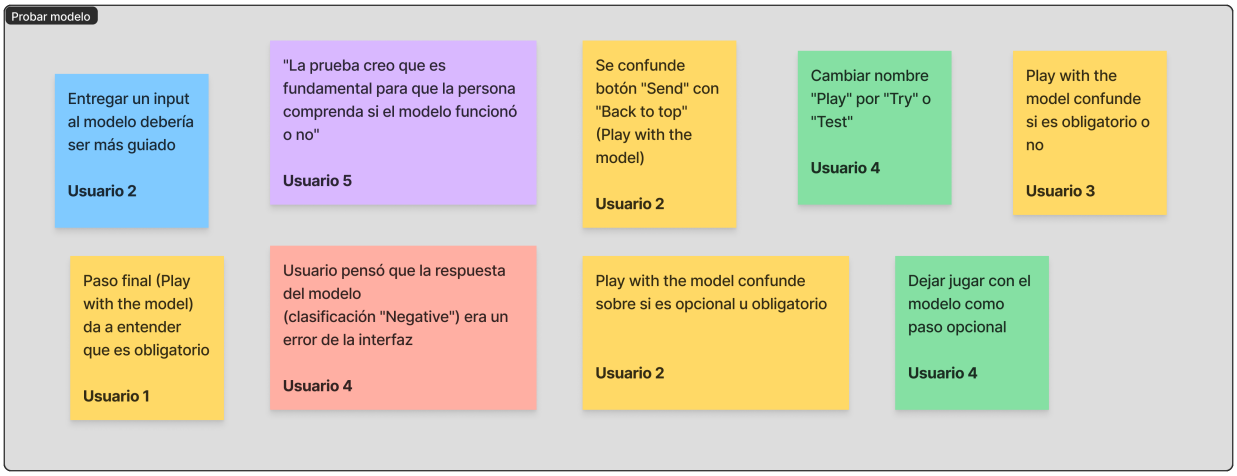

Figura 5.7: Anotaciones tomadas de los usuarios durante la prueba de usabilidad que respectan a probar el modelo.

Se puede observar que la funcionalidad para probar el modelo es confusa debido a algunas razones:

- No se guía lo suficiente a los usuarios.
- Los usuarios piensan que deben probar los modelos luego del entrenamiento. Esto, de hecho, fue lo que causó que dos usuarios fallaran una de las tareas tal como fue expuesto anteriormente.
- Las disposiciones de los elementos pueden ser confusas. Por ejemplo: el nombre del módulo, el botón para enviar y para volver al inicio del “pipeline”, etc.

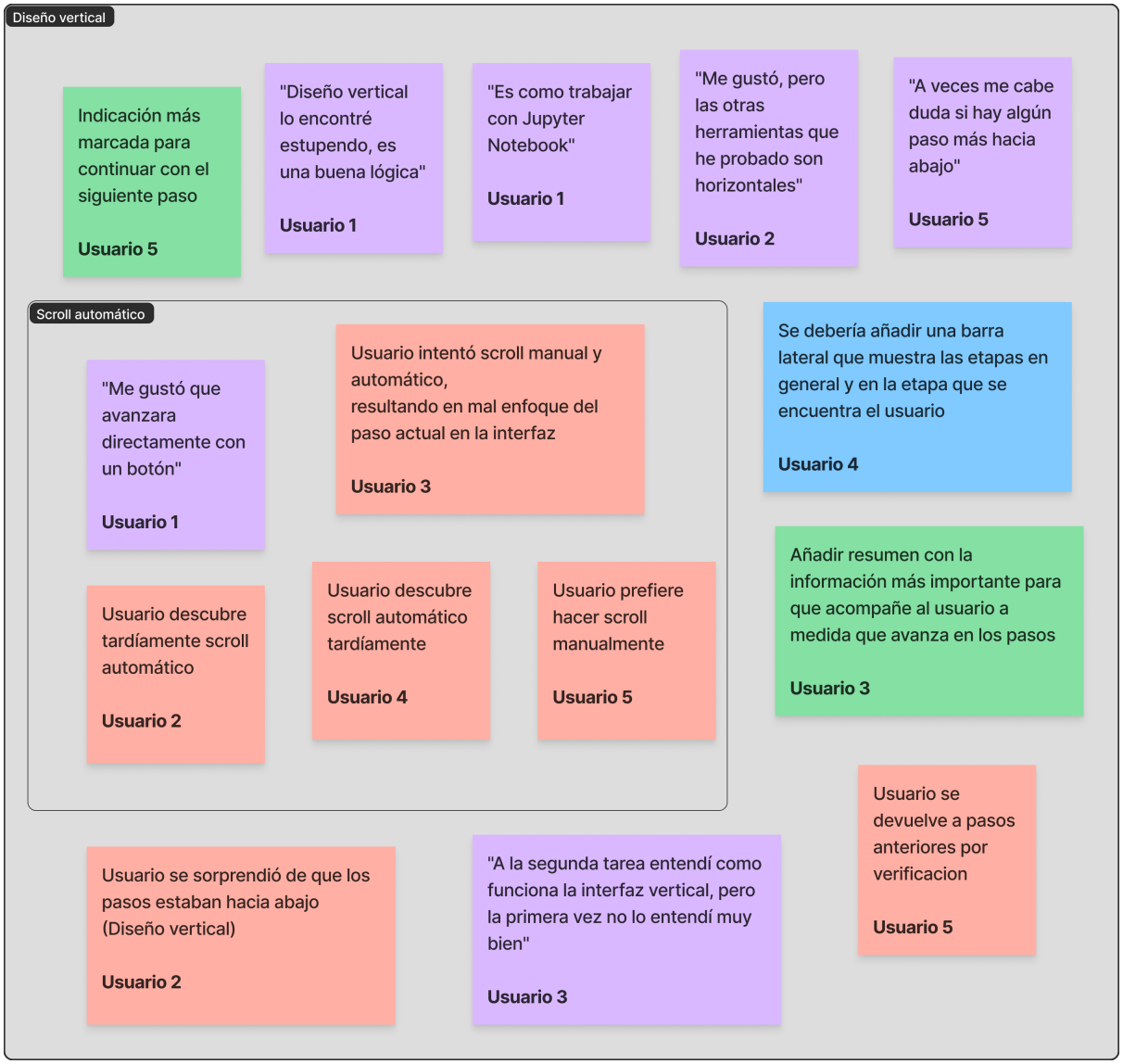

Figura 5.8: Anotaciones tomadas de los usuarios durante la prueba de usabilidad que respectan al diseño vertical de la interfaz gráfica.

En cuanto al diseño vertical de la interfaz (“avanzar hacia abajo”) se encuentran opiniones mixtas: a un grupo de usuarios les gusta el diseño vertical, creen que es una buena lógica y lo comparan con *Jupyter Notebook*. Mientras que otro grupo, piensa que es insuficiente ya que se desvía demasiado de otras herramientas, o también porque necesita de más guías sobre las etapas en general y en la etapa en la que se encuentra el usuario.

En cuanto al *scroll* automático, también se notan opiniones mixtas, aunque más cargadas hacia las críticas. Algunos usuarios descubrían tardíamente el *scroll* automático pues preferían realizarlo de forma manual. En cambio, a otros usuarios les gustó el *scroll* automático, pues para ellos era más cómodo.

Figura 5.9: Anotaciones tomadas de los usuarios durante la prueba de usabilidad que respectan a temas generales

Sobre los subgrupos:

- **Enseñar machine learning / IA**: En general los usuarios piensan que es una buena herramienta para enseñar la importancia de los parámetros en un experimento. Sin embargo, también piensan que es demasiado simple y que se quedaría corto si se quisiera

ir más profundo. Esta es una de las limitaciones presentadas anteriormente, sobre que no es posible tener un entendimiento más profundo si se siguen viendo los modelos como "cajas negras".

- **Local o servidor**: Una pregunta sobre el proyecto en general. Los usuarios manifiestan opiniones mixtas: algunos prefieren un servidor, otros se muestran contentos con que la plataforma sea local. Pero también existen usuarios que piensan que ambos acercamientos serían la mejor opción.

En otras notas los usuarios se muestran contentos con la simpleza de la plataforma, la cual permite realizar experimentos rápidos. O también entregan recomendaciones como añadir una versión en español, agregar tutoriales o poder extraer los modelos entrenados de la plataforma para poder complementar con otras herramientas.

Capítulo 6

Conclusiones

Con la meta de disminuir las complejas barreras de entrada que presenta iniciarse en el mundo de la inteligencia artificial se crea el proyecto *DashAI*, el cual, mediante una interfaz gráfica y una ordenada, extensible y general lógica de *backend*, pretende disponibilizar la mayor cantidad de tareas de inteligencia artificial para analistas de datos o científicos que necesiten agilizar su flujo de trabajo, o bien que tengan interés por el área de IA y quieran potenciar su aprendizaje.

Uno de los puntos fundamentales para lograr los objetivos de *DashAI* es su interfaz gráfica, la componente que se dedica a interactuar con los usuarios y guiarlos a través de la compleja lógica del *backend*. El desarrollo de esta interfaz fue el problema que se abordó como trabajo de título. Para tener una interfaz que refleje las fortalezas del proyecto se fijó como objetivo al diseñar e implementar la interfaz que esta fuese usable. Además de otros objetivos como, por ejemplo, una serie de tareas que un usuario debería ser capaz de realizar y que fueron establecidas de acuerdo a una exhaustiva investigación.

Como resultado se obtuvo una interfaz con diseño vertical, dividida en cuatro componentes principales encargados de cargar datos, configurar el experimento, visualizar resultados y probar un modelo.

En cuanto a las tareas establecidas se han implementado correctamente 11 de las 14 funcionalidades fijadas en un inicio. Las 3 tareas restantes se dejaron para último momento debido a su poca urgencia, pues la herramienta podía funcionar sin ellas. Esto junto con subestimar algunas tareas como la vista de configuración del experimento la cual, como se dijo anteriormente, fue la parte más compleja de la implementación. Esto pues debía poder adaptarse no solo a los modelos que estaban ya implementados, sino que también a los que se implementarían en el futuro.

La interfaz además fue evaluada en una prueba de usabilidad con 5 usuarios, presentando sus conclusiones a través de indicadores objetivos como eficacia y eficiencia. Además de incluir también un indicador subjetivos a través del cuestionario *NASA-TLX*. Estos resultados se resumen a continuación:

- **Eficacia**: Con la excepción de la primera tarea, las dos restantes pudieron ser completadas por la mayor parte de los participantes de la prueba.
- **Eficiencia**: La mayor parte de los usuarios fue capaz de completar cada tarea en un tiempo razonable.
- **Cuestionario**: La mayoría de los usuarios ha dado puntajes bajos a las categorías negativas mientras que dieron puntajes altos a las categorías positivas.

En los resultados de esta prueba pueden influir varias cosas:

- Los resultados evidencian que se eligió la tarea más difícil (tarea 1) para comenzar, lo cual no fue una buena idea y terminó por provocar que algunos de ellos no pudieran completarla.
- A esto se sumó que, quizás el planteamiento de la primera tarea no fue el mejor. Puesto que se pedía a los usuarios detenerse en la vista de resultados, justo antes del paso final, cuando había en la interfaz un botón "diciéndoles" que fueran al paso siguiente.
- Como se puede comprobar, las dos tareas restantes obtuvieron los mejores resultados. Aunque es un buen resultado esto puede estar influenciado porque los usuarios ya no se enfrentaban a la interfaz por primera vez y que ambas tareas eran bastante similares entre sí.

Los resultados finales entregan que se ha aprobado en eficiencia y percepción subjetiva. Mientras que en eficacia no se ha conseguido aprobar. Habiendo aprobado dos de los tres indicadores de usabilidad, mientras que para el tercero se han identificado los motivos de su reprobación. Se puede entender como un balance positivo que entrega una buena proyección para la interfaz diseñada e implementada. Se debe considerar además que la mayoría de aplicaciones que proporcionan interfaces para realizar tareas de IA sin programar cuentan con tutoriales ya sean en videos o plasmados en la misma aplicación para enseñar al usuario cuando este la ejecuta por primera vez. Por lo que si *DashAI* aprovecha estos medios la interfaz podría incluso mejorar la facilidad su facilidad de uso.

El presente trabajo representa una de las primeras versiones de la herramienta *DashAI* y es alentador que, en general, se hayan obtenido buenos resultados para el potencial que presenta esta herramienta al abordar el problema de disminuir las barreras para entrar al mundo de la inteligencia artificial y que, en un futuro, *DashAI* pueda ser utilizado para introducir a principiantes a la IA, enseñarles la importancia de la configuración de parámetros, en qué casos es mejor aplicar un modelo o que sean capaces de entender mejor sus modelos entregándoles *inputs* y observando sus *outputs*. También se espera que personas que trabajan en la industria sean capaces de utilizar *DashAI* para agilizar su flujo de trabajo. O quizás que gracias a su categorización como código abierto se genere una comunidad de usuarios que puedan ser capaces de crear sus propios modelos y *tasks*. Esta es la principal razón de que la herramienta esté en inglés, ser capaces de difundirla para que usuarios puedan desarrollar y aprovechar su propio potencial.

En cuanto a lecciones aprendidas se tienen el correcto gestionamiento del tiempo, entender que a veces hay funcionalidades que son bastante más difíciles de lo que parecen y que pueden terminar consumiendo mucho más tiempo del previsto. Otro de los grandes aprendizajes es el de haber realizado por primera vez una prueba de usabilidad, el que los usuarios puedan utilizar y dar su opinión sobre el *software* desarrollado por uno mismo es una experiencia enriquecedora que permite ser más consciente de las falencias que tiene, pero también descubrir posibles funcionalidades que no habían sido imaginadas durante el diseño. Toda esta experiencia puede potenciar muchísimo el desarrollo de un proyecto y permitir que vaya bien encaminado.

Finalmente, y en concordancia con el párrafo anterior, se exponen algunos puntos para el trabajo futuro que fueron obtenidos gracias a las observaciones de usuarios durante la prueba de usabilidad y que se encuentran resumidos en el diagrama de afinidad presentado anteriormente.

- En primer lugar, los usuarios mostraron descontento con que se haya ignorado el área de ingeniería de datos, varios presentaron dificultades en esto. Se reconoce que sería un beneficio enorme para el proyecto contar con visualizaciones que permitan a los usuarios entender de mejor forma los datos con los que trabajan. Algunos ejemplos de esto: tabla con algunas filas como forma de muestra del dataset que se suba, permitir realizar algún preprocesamiento como seleccionar los conjuntos de *train* y *test*, selección de *features*, incluso algo más simple como mostrar el nombre del set de datos cargado.
- Contar con información más gráfica en la vista de resultados, alguna matriz de confusión, gráficos de la función de *loss* en redes neuronales. Es una tarea desafiante puesto que cada tarea puede tener visualizaciones distintas.
- Mejorar la visualización para utilizar el modelo, de forma que esta pueda adaptarse de mejor forma a los datos con los que se trabaja. Guiar más al usuario sobre el formato que este *input* debe tener.
- Obviamente también un trabajo futuro es implementar las tareas que faltaron, funcionalidades como guardar experimentos pasados y ser capaz de utilizarlos o modificarlos es algo que aunque no fue urgente añadir, pero que le aportarían un valor inmenso a la herramienta.

En cuanto a trabajo futuro de más largo plazo se tiene:

- Realizar un estudio más completo sobre la interfaz gráfica, con el cual se pueda generalizar al público objetivo y realizar predicciones sobre estos usando estadística.
- Agregar un módulo de explicabilidad que utilice bibliotecas científicas para proporcionar a los usuarios un entendimiento de los fundamentos de los modelos entrenados.
- Realizar una versión para utilizar en un servidor y que, de esta forma, la herramienta no sea solo de uso local.

Bibliografía

[1] John J Dudley and Per Ola Kristensson. A review of user interface design for interactive machine learning. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 8(2):1–37, 2018.

[2] Laura Faulkner. Beyond the five-user assumption: Benefits of increased sample sizes in usability testing. *Behavior Research Methods, Instruments, & Computers*, 35(3):379–383, 2003.

[3] Eibe Frank, Mark Hall, Geoffrey Holmes, Richard Kirkby, Bernhard Pfahringer, Ian H. Witten, and Len Trigg. *Weka-A Machine Learning Workbench for Data Mining*, pages 1269–1277. Springer US, Boston, MA, 2010.

[4] Rebecca A Grier, Aaron Bangor, Philip Kortum, and S Camille Peres. The system usability scale: Beyond standard usability testing. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 57, pages 187–191. SAGE Publications Sage CA: Los Angeles, CA, 2013.

[5] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. 11(1):10–18, nov 2009.

[6] Sandra G Hart and Lowell E Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In *Advances in psychology*, volume 52, pages 139–183. Elsevier, 1988.

[7] Felix Jungermann. Information extraction with rapidminer. In *Proceedings of the GSCL Symposium'Sprachtechnologie und eHumanities*, pages 50–61. Citeseer, 2009.

[8] Dan Jurasfky and Martin James H. *Speech and Language Processing, (2nd Edition)*. Pearson, 2014.

[9] Thanh Tung Khuat, David Jacob Kedziora, and Bogdan Gabrys. The roles and modes of human interactions with automated machine learning systems, 2022.

[10] Jurek Kirakowski and Mary Corbett. Sumi: the software usability measurement inventory. *British Journal of Educational Technology*, 24:210 – 212, 10 2006.

[11] Steve Krug. *Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability*. New Riders Publishing, USA, 3rd edition, 2014.

[12] Mike Kuniavsky. *Observing the user experience: a practitioner's guide to user research*. Elsevier, 2003.

[13] Andrés Lucero. Using affinity diagrams to evaluate interactive prototypes. In *IFIP conference on human-computer interaction*, pages 231–248. Springer, 2015.

[14] Arnold M Lund. Measuring usability with the use questionnaire12. *Usability interface*, 8(2):3–6, 2001.

[15] Kevin P Murphy. *Probabilistic machine learning: an introduction*. MIT press, 2022.

[16] Safiya Umoja Noble. Algorithms of oppression. In *Algorithms of Oppression*. New York University Press, 2018.

[17] Verweij G. Cameron E. Rao, A. Sizing the prize: What is the real value of ai for your business and how can you capitalise?, 2017. `https://www.pwc.com/gx/en/issues/analytics/assets/pwc-ai-analysis-sizing-the-prize-report.pdf` (visited 2023-01-24).

[18] Stuart Russell and Peter Norvig. Artificial intelligence: a modern approach, global edition 4th. *Foundations*, 19:23, 2021.

[19] Ryan Singer. *Shape Up: Stop Running in Circles and Ship Work that Matters*. Basecamp, 2019. `https://basecamp.com/shapeup/shape-up.pdf`(visited 2022-06-22).

[20] Sören Sonnenburg, Mikio Braun, Cheng Soon Ong, Samy Bengio, Léon Bottou, Geoffrey Holmes, Yann Lecun, Klaus-Robert Müller, Fernando Pereira, Carl Rasmussen, Gunnar Rätsch, Bernhard Schölkopf, Alexander Smola, Pascal Vincent, Jason Weston, and Robert Williamson. The need for open source software in machine learning. *Journal of Machine Learning Research*, 8:2443–2466, 10 2007.

[21] Ministerio de Ciencia Tecnología, Conocimiento e Innovación. Encuesta nacional de innovación 2017 - 2018, 2020. `https://www.minciencia.gob.cl/legacy-files/1_presentacion_resultados_eni_2017-2018.pdf` (visited 2023-01-24).

[22] Ministerio de Ciencia Tecnología, Conocimiento e Innovación. Política nacional de inteligencia artificial, 2021. `https://minciencia.gob.cl/uploads/filer_public/bc/38/bc389daf-4514-4306-867c-760ae7686e2c/documento_politica_ia_digital_.pdf` (visited 2023-01-24).

[23] Ian H Witten, Eibe Frank, Mark A Hall, Christopher J Pal, and MINING DATA. Practical machine learning tools and techniques. In *Data Mining*, volume 2, 2005.

ANEXOS

Anexo A

API

A.1. Formato del set de datos

Para trabajar con la herramienta es necesario, por ahora, utilizar los set de datos que han sido creados explícitamente por el equipo de desarrollo. Se presentan, en lo que sigue, dos ejemplos del formato correcto de set de datos, el primer ejemplo muestra un dataset de clasificación numérica, mientras que el segundo muestra un ejemplo de clasificación de texto.

```
{
    "task_info":{
        "task_type":"NumericClassificationTask"
    },
    "train":{
        "x":[
            [1,2,3],
            [2,3,4],
            [3,2,1],
            [5,2,0],
            [1,2,3],
            [2,3,4],
            [3,2,1],
            [5,2,0]
        ],
        "y":[
            "1",
            "0",
            "1",
```

```
            "1",
            "1",
            "0",
            "1",
            "1"
        ]
    },
    "test":{
        "x":[
            [1,2,3],
            [2,3,4],
            [3,2,1],
            [5,2,0],
            [1,2,3],
            [2,3,4],
            [3,2,1],
            [5,2,0]
        ],
        "y":[
            "1",
            "0",
            "1",
            "1",
            "1",
            "0",
            "1",
            "1"
        ]
    }
}
```

Listing A.1: Ejemplo de dataset de clasificación numérica, se puede apreciar como principales atributos su tipo de task, su conjunto de *test* y su conjunto de *train*

```
{
    "task_info":{
        "task_type":"TextClassificationTask"
    },
    "train":{
        "x":[
            "hola",
            "como",
            "estas",
            "tu"
        ],
        "y":[
            "1",
            "0",
            "1",
            "1"
        ]
    },
    "test":{
        "x":[
            "hola",
            "como",
            "estas",
            "tu"
        ],
        "y":[
            "1",
            "0",
            "1",
            "1"
        ]
    }
}
```

Listing A.2: Ejemplo de dataset de clasificación de texto,

A.2. Modelos

Los modelos son otro tipo de objeto, el cual tiene definidos los nombres de sus parámetros, los valores permitidos para estos y sus parámetros por defecto.

Figura A.1: Ejemplo de un objeto que representa un modelo y sus parámetros, en este caso, se presenta para el modelo KNN

Anexo B

Prueba de usabilidad

B.1. Guión

En lo siguiente se presenta el guión que se siguió para estandarizar la prueba de usabilidad y que esta fuera lo más similar posible para todos los participantes.

B.1.1. Introducción

Hola, muchas gracias por venir. ¿Cómo estás? ¿Te molesta si grabo la sesión, esta grabación será utilizada solamente para un análisis que será presentado en un informe de manera escrita?

Me presento, soy Maximiliano Aguilar, estoy haciendo mi memoria que consiste en diseñar e implementar la interfaz gráfica de DashAI, DashAI es una herramienta de software de código abierto que está siendo desarrollada en conjunto con el centro nacional de inteligencia artificial (CENIA), como parte de la iniciativa OpenCENIA. La principal función de DashAI es permitir entrenar modelos de machine learning sin tener que programar.

En este semestre sacamos un primer prototipo con el equipo y queremos probar qué tan bien funciona la herramienta con su público objetivo, ver que piensan, en qué se puede mejorar y detectar errores.

Esta prueba debería durar máximo 40 min y está enfocada en que resuelvas algunas tareas con la herramienta.

Como dije antes este es un primer prototipo de la herramienta, a parte de implementación también estamos evaluando diseño , queremos ver qué cosas funcionan y qué cosas hay que cambiar, así que pueden haber ciertas features que no funcionen correctamente.

La prueba va a funcionar así: te voy a hacer unas pocas preguntas al principio, luego te voy a mandar un link con el que deberías poder acceder a la interfaz y con eso vamos a hacer tres tareas. Una vez terminadas te voy a mandar un cuestionario simple y te voy a hacer algunas preguntas finales, con eso terminaríamos. ¿Alguna pregunta?

Algunos puntos importantes sobre la prueba:

- Los resultados de la prueba y tus respuestas se usarán solo en el estudio y existirá una reserva total de tu identidad y la información que entregues.
- Podemos detener la prueba en cualquier momento.
- Te puedes ir en cualquier momento sin que haya ninguna consecuencia.

B.1.2. Entrevista preliminar

Ahora voy a hacer algunas preguntas iniciales:

- ¿A qué te dedicas?
- ¿Hace cuanto tiempo te iniciaste en el área de machine learning?
- ¿Puedes contarme algo sobre tu experiencia en machine learning/ inteligencia artificial?
- ¿Has usado aplicaciones de machine learning que no requieran programación? algo como WEKA, Rapidminer, KNIME, H2O, etc.

B.1.3. Instrucciones de evaluación

Antes de pasar a las tareas, unas instrucciones generales: Tú estás probando la herramienta, no al revés. Si hay algo que no funciona, se ve confuso o raro no es tu culpa, de hecho, sería una gran ayuda que pudieras decirlo. También me interesa saber qué cosas te gustan de la herramienta pueden ser colores, la forma en que está ordenada, etc.

Tampoco te contengas con las palabras, si algo está muy mal hecho esta es la instancia para descubrirlo. Si hay algo que se ve horrible o raro o no tiene sentido sería una gran ayuda poder saberlo.

¿Alguna pregunta antes de empezar?

B.1.4. Tareas(20 min)

1. Te encuentras investigando herramientas de machine learning sin código para un nuevo proyecto, decides probar DashAI, en tu primer intento usas el dataset Iris, entrenando

solo un modelo sin realizar configuraciones adicionales, visualizas los resultados del entrenamiento y por último pruebas el modelo con un input para ver cómo lo clasifica. [**8 min**]

2. Siguiendo con la prueba, decides ahora intentar un dataset de clasificación de texto (twitter dataset), el cual contiene datos de tweets realizados por usuarios en Twitter, a los cuales se les ha asociado con un sentimiento (Positivo, Negativo).La tarea consiste en entrenar **un** modelo, esta vez, **configurando sus parámetros** en la forma que el usuario desee, para luego visualizar los resultados entregados y, finalmente, probar el modelo entregando un input específico.[**6 min**]

3. Por último, decides intentar con un dataset que represente un problema más moderno, para ello entrenas (fine tuning) un modelo de generación de texto (traducción), visualizas sus resultados y pruebas entregarle un input para ver lo que el modelo responde. [**6 min**]

B.1.5. Preguntas finales

- ¿Qué te parece el diseño vertical de la interfaz?
- ¿Qué opinas de la elección de colores? ¿Te gustaría que hubiera un tema claro?
- ¿Qué opinas sobre que se muestre la task luego de subir el dataset? ¿Crees que es fácil de pasar por alto?
- Con respecto al formulario de configuración de parámetros ¿es complejo, simple? ¿añadirías o quitarías algo?
- Siguiendo con el formulario ¿Qué opinas de las formas de guiar al usuario? ¿Falta algo, son suficientes?
- ¿Qué opinas de los nombres utilizados para los modelos? (numericalWrapperForText)
- ¿Qué opinas sobre la visualización de resultados, estarías interesado en algo más que se muestre ahí ?
- En la sección de resultados ¿Qué te parece que se muestren los parámetros con los que fue entrenado el modelo? ¿Crees que deberían tener más importancia, menos importancia?
- ¿Crees que tiene sentido que probar el modelo sea el paso final? ¿lo moverías a otro apartado?
- ¿Qué opinas de utilizar la aplicación localmente? ¿Crees que sería más conveniente que estuviese alojado en un servidor?
- ¿Usarías la aplicación para probar modelos o para enseñar machine learning/IA a alguien?

B.2. Cuestionario: Nasa-TLX

Se muestra a continuación un *screenshot* del cuestionario que se pidió responder a los participantes de la prueba de usabilidad.

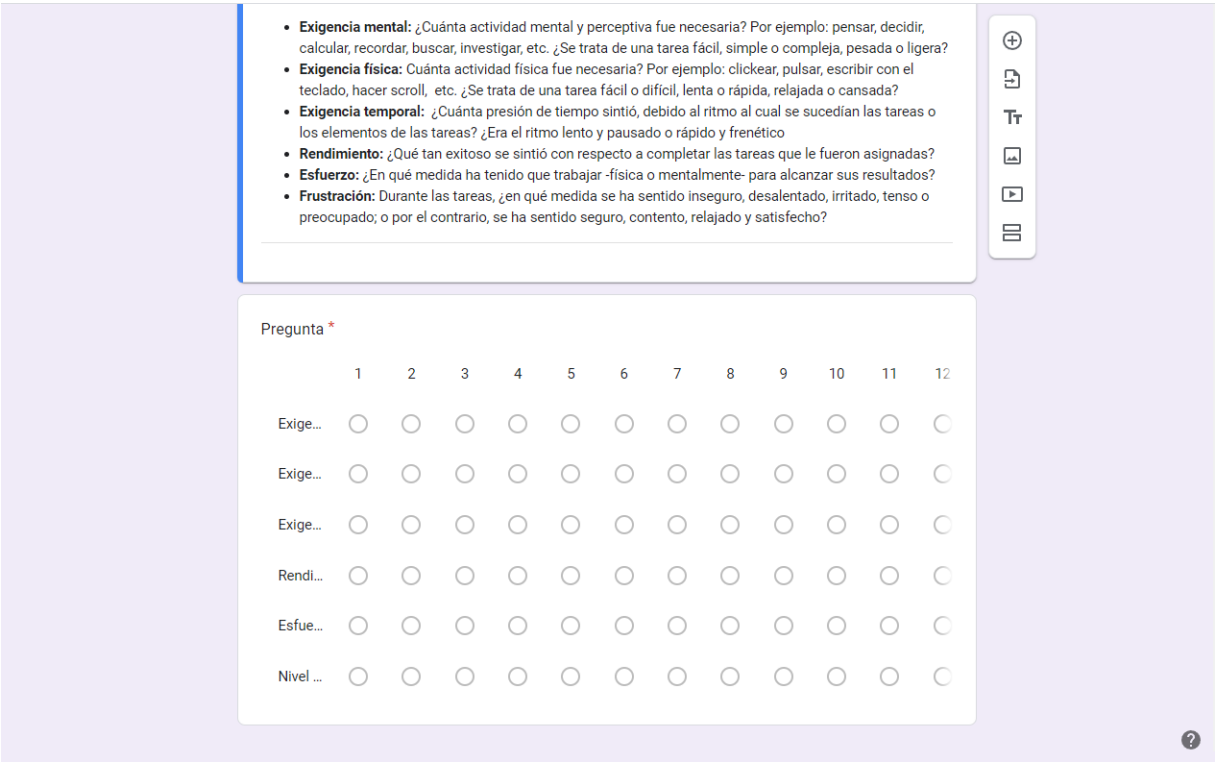

Figura B.1: Cuestionario en *Google Forms* que se pidió a los usuarios que respondieran.