Universidad Autónoma de Baja California
Instituto de Ingeniería
Maestría y Doctorado en Ciencias e Ingeniería

Plataforma de Ejecución y Validación de Modelos de Aprendizaje de Máquina

Tesis que para obtener el grado de:

MAESTRO EN CIENCIAS

Presenta

Henry Alonso Ruiz Guzman

Director de Tesis:

Dr. Félix Fernando González Navarro

Co-Director de Tesis:

Dra. Brenda Leticia Flores Rios

Mexicali, B. C. Junio del 2018

Dedicatorias

Quiero dedicarle este título a Dios por haberme dado la sabiduría y fortaleza necesaria para asumir este reto. a mi familia por su paciencia e incondicional apoyo desde la distancia, a mis maestros por su soporte técnico y el conocimiento compartido en las aulas. a Conacyt y la universidad por proveerme los recursos necesarios para concluir satisfactoriamente y llevar a cabo mi investigación. A todos los demás que no alcanzan a ser mencionados, mis más sinceros agradecimientos por la contribución que hicieron en esta importante meta.

Reconocimientos

Al Consejo Nacional de Ciencia y Tecnología (CONACYT) y la universidad de Baja California(UABC) por haberme dado la oportunidad de llevar a cabo mis estudios de maestría en este grandioso país (México).

Al Instituto de Ingeniería por su recibimiento y acogida durante todo este tiempo.

A mi director Dr. Félix Fernando Navarro, mi codirectora Dra. Brenda Flores y al resto de integrantes del equipo evaluador por su guía y tiempo dedicado

RESUMEN de la Tesis de Henry Alonso Ruiz Guzman, presentado como requisito parcial para la obtención de grado de MAESTRO EN CIENCIAS, Mexicali, Baja California, Junio de 2018.

PLATAFORMA DE EJECUCIÓN Y VALIDACIÓN DE MODELOS DE APRENDIZAJE DE MÁQUINA

Aprobado por:

Dr. Félix Fernando González Navarro
Director de Tesis

Dra. Brenda Flores
Codirectora de Tesis

La herramienta de software que se presenta a continuación, permite al usuario a través de una interfaz web, amigable e intuitiva, descubrir patrones en sus datos, realizar visualizaciones y construir, entrenar y validar modelos de aprendizaje de máquina. La aplicación estará alojada en los servidores de la UABC, será de libre acceso y estará orientada a la comunidad académica. A este tipo de aplicaciones se les conoce como plataformas de machine learning como servicio y entre las mas destacadas se encuentran: machine learning studio de microsoft, watson de IBM, Google Cloud machine learning y Amazon machine learning.

Por otro lado como resultado de la investigación, se propone un modelo de arquitectura escalable, horizontal y verticalmente para la ejecución de algoritmos de aprendizaje de máquina de forma distribuida.

Palabras Clave: Machine Learning como servicio, Machine learning, Aprendizaje de máquina

ABSTRACT of the thesis, presented by Henry Alonso Ruiz Guzman, as a partial requirement to obtain the MASTER IN SCIENCE AND ENGINEERING, Mexicali, Baja California, June,2018.

PLATFORM OF EXECUTION AND VALIDATION OF MACHINE LEARNING MODELS

Approved by:

Dr. Félix Fernando González Navarro
Director de Tesis

Dra. Brenda Flores
Codirectora de Tesis

The software tool presented below, allows the user through a web interface, friendly and intuitive, to discover patterns in their data, make visualizations and build, train and validate machine learning models. The application will be hosted on the servers of the UABC, will be freely accessible and will be aimed at the academic community.

This type of applications are known as machine learning platforms as a service and among the most outstanding are: machine learning studio from Microsoft, IBM watson, Google Cloud machine learning and Amazon machine learning.

On the other hand we propose a scalable architecture model, horizontally and vertically for the execution of machine learning algorithms in a distributed way.

Keywords: Machine Learning as a service, Machine learning

Índice

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

1.1. Plataformas de machine learning

El aprendizaje de máquina, en ingles machine Learning, es un término tan amplio que en internet o en la literatura, según el autor y contexto, se pueden encontrar muchas definiciones. No obstante, en la mayoría de los casos, se usa para referenciar o describir técnicas como: reconocimiento facial, reconocimiento de patrones, reconocimiento de voz, procesamiento de lenguaje natural, análisis predictivo, redes neuronales, visión por computadora, Aprendizaje profundo etc. Son tantas las aplicaciones, que hoy día compañías como Google, Microsoft, IBM y algunas no tan reconocidas, proveen a través de servicios web o librerías (para Python, C++, R etc.) una interfaz para que los desarrolladores puedan añadir a sus soluciones de software dichas funcionalidades. Solo con unas cuantas lineas de código es posible aprovisionar una aplicación con reconocimiento facial ó autenticación biométrica usando los servicios cognitivos de Microsoft. Lo anterior representa un gran aporte, dado que los algoritmos disponibles en el Api (Interfaz de programación de aplicaciones) han sido rigurosamente optimizados.

Algunas de las soluciones más sofisticadas en el mercado, que están enfocadas en el análisis predictivo, incluso ponen a disposición herramientas para que el usuario pueda construir desde una interfaz web sus propios modelos de aprendizaje de máquina y consumirlos desde código a través de servicios web.

A este tipo de herramienta se les conoce como plataformas de aprendizaje de máquina como

servicio (en ingles Machine Learning as a service / MLaaS) y ponen a disposición del usuario un conjunto de herramientas que se pueden adaptar a su negocio y les permiten explorar sus datos, identificar patrones, construir modelos predictivos etc. Sin embargo, los costos de licenciamiento de la mayoría de estas herramientas son muy altos y además la interfaz que proveen no es muy amigable con el usuario.

1.2. Planteamiento del problema

Soluciones como Microsoft Azure Machine Learning Studio, Amazon Cloud, Google Cloud Machine Learning, proveen las herramientas y los recursos suficientes para desarrollar, almacenar y desplegar aplicaciones como sistemas de predicción, detección de Objetos, seguridad, Big data etc. Con su modelo de negocio de infraestructura como servicio el usuario paga solo por lo que consume. Esto ha llevado a que las empresas dejen de invertir en capacidad física y se empiecen a mover hacia la nube. No obstante, los costos hasta hoy siguen siendo muy altos y al final el usuario termina pagando por servicios que no requiere. En la tabla 1.1 se puede observar la diferencia en términos de licenciamiento entre las plataformas de MLaaS mas reconocidas, en ésta se puede evidenciar que el valor a pagar varía de acuerdo a los servicios requeridos, el número de predicciones o el almacenamiento.

Por otro lado en términos de usabilidad, muchas de estas plataformas no son muy amigables con el usuario, y se requiere de tener conocimientos avanzados en aprendizaje de máquina y modelamiento.

Otro factor importante en un experimento es el tiempo, En un análisis predictivo, la selección del modelo, por lo general, se inicia realizando ejecuciones sucesivas de diferentes algoritmos(de clasificación o regresión, según el tipo de problema) y luego basado en métricas como la precisión o la tasa de reconocimiento se termina escogiendo el mejor, este proceso continua cíclicamente hasta que se obtiene el resultado deseado, y en cada iteración lo que se hace, es alterar los parámetros de ajuste del modelo. lo anterior resulta muy dispendioso, puesto que casi siempre se hace uso de algún lenguaje de programación como Python, R o Matlab y la curva de aprendizaje representa una limitante.

La propuesta consiste en una plataforma web de MLaaS, de libre acceso y amigable con el usuario, que le permita, visualizar, interactuar, probar diferentes algoritmos de aprendizaje de máquina, sin tener conocimientos sobre algún lenguaje de programación.

Product	Pricing	Extras
Amazon Machine Learning	▪ Por los servicios de análisis de datos y construcción de modelos: $0.42 por hora. ▪ Tarifas de predicción: $0.10 por mil predicciones en batch, $0.0001 por cada predicción en tiempo real,mas $0.001 por hora, por cada 10 mb de memoria provisionada.	Los costos de almacenamiento en Amazon web services, serán cobrados por separado
Google Cloud Machine Learning	▪ Por el servicio de entrenamiento de modelos: $0.49 por hora, por cada unidad de entrenamiento(unidad de medida definida por Google para referirse a los recursos de computo habilitados) en US, $0.54 en Europa y Asia. ▪ Tarifas de predicción: $0.10 por mil predicciones,mas $0.40 por cada unidad de computo desplegada en US, entre $0.11 y $0.44 en Europa y Asia. El precio varía significativamente según el número de llamados que se hagan a los modelos pre-entrenados.	Se requiere una cuenta de Google Cloud Platform
IBM Watson Machine Learning	▪ $10 por instancia ejecutándose(cada máquina puede correr hasta 20 modelos). ▪ $10 por los servicios de análisis y construcción de modelos: $0.45 por hora de computo. ▪ Tarifas de predicción: $0.50 por 1000 predicciones en batch o en tiempo real. ▪ Es posible crear una instancia de forma gratuita, con hasta dos modelos, 5000 predicciones por mes y máximo 5 horas de computo por día.	Se requiere una cuenta de Bluemix
Microsoft Azure Machine Learning	▪ $9.99 por usuario, por mes por Azure Machine Learning Studio, mas $1.00 por hora de experimentación (unidad de medida definida por Microsoft para referirse a los recursos de computo habilitados) ▪ Los modelos construidos se pueden exponer como servicios web, y el precio por el número de llamadas oscila entre $100,$1000 y hasta $10.000 por mes.	Se requiere una cuenta de Microsoft azure

Tabla 1.1: Tabla de comparación de Soluciones de Machine Learning como servicio

1.3. Objetivos

1.3.1. Objetivo general

Desarrollar e implementar una plataforma web que sirva como Front-End para la ejecución y evaluación de algoritmos de Machine Learning.

1.3.2. Objetivos Específicos

1. Diseñar un modelo de datos flexible que permita llevar el registro de cada experimento sin importar su estructura o diseño.
2. Definir una arquitectura, escalable, desacoplada y robusta, que soporte el modelo de negocio de la solución planteada.
3. Construir un *api rest*, donde se expongan la funcionalidad *core* del sistema, que a futuro facilite el desarrollo e inclusión de nuevos clientes a la arquitectura.
4. Implementar en C++ los algoritmos considerados en la plataforma.
5. Incluir un middleware de encolamiento, para la gestión de tareas de alto procesamiento en la arquitectura

1.4. Metodología

Para dar cumplimiento a los objetivos específicos, se definieron las siguientes actividades:

1. En la primera fase se evaluarán funcionalmente algunas de las plataformas de aprendizaje de máquina como servicio que hay en el mercado.
2. En la fase 2, se definirá el alcance del software con la lista de algoritmos a implementar. Cuales de clasificación, de regresión y agrupamiento.
3. Se seleccionará la metodología de desarrollo que servirá como marco de trabajo y guía para la construcción el desarrollo del software.
4. En la fase 3 se continuará con el análisis y diseño, empezando con el levantamiento de requerimientos,la elaboración de las historias de usuario, la construcción del modelo de datos y la generación del resto de artefactos.

5. En la fase 5, se definirá la arquitectura del sistema.

6. Seguido a la construcción del modelo de datos y la arquitectura, se definirá la pila de herramientas que se usará para implementar y desplegar la plataforma.

7. Para concluir se realizará un experimento de prueba, que sirva como modelo para la validación de la herramienta.

1.5. Estructura del documento

El documento se encuentra organizado de la siguiente forma:
En el capítulo 2 se desarrolla el marco teórico. En éste se abordan los conceptos fundamentales sobre los cuales se soporta esta investigación, las plataformas de aprendizaje de máquina como servicio, El aprendizaje de máquina, sus aplicaciones y enfoques, los sistemas distribuidos son algunos de los tópicos que allí se tratan.

El capitulo 2, finaliza con el estado del arte, se realiza el análisis de algunas herramientas similares a la desarrollada en esta tesis y se destacan sus diferencias.

en el capítulo 3, se describe como fue el desarrollo de la plataforma, los módulos del sistema y su arquitectura, además, se adjuntan los artefactos que articulan el diseño de la solución.

Finalmente, en el capítulo 5, se presentan las conclusiones y trabajo futuro.

Capítulo 2

Plataformas de aprendizaje de máquina como servicio

En los últimos años la inteligencia artificial ha tomado mucha fuerza, en áreas como la medicina, el comercio electrónico, la robótica, la arquitectura etc. Principalmente aplicaciones como el aprendizaje de máquina, la visión por computadora y el aprendizaje profundo. Es por esto que grandes compañías como Microsoft, Google, IBM entre otras, han enfocado sus esfuerzos en construir y desarrollar soluciones tanto de hardware como de software, que permitan a otras empresas u organizaciones realizar investigación en torno a estos campos de acción (Forbes, 2018).

A esta gran revolución los expertos le han llamado la era de la democratización de la inteligencia artificial, debido a que la IA paso de ser una tendencia en los más innovadores, a ser un área de investigación hacia lo que todas las organizaciones están redirigiendo sus esfuerzos (Center, 2016a).

Una de las limitantes más grande a la hora de desarrollar e implementar soluciones de Machine Learning, análisis de datos, visión por computadora etc. esta en los recursos de hardware. Cuando se está trabajando con grandes volúmenes de información, el entrenamiento de los mo-

delos, el pre-procesamiento etc. son tareas muy exigentes en términos de capacidad de computo y la mayoría de las empresas no tienen los recursos necesario para ocuparse de ello. Hoy gracias a las plataformas de Cloud Computing y "Machine Learning as a Service", la infraestructura dejo de representar un problema, debido a que los procesos anteriormente mencionadas se ejecutan en la nube, en máquinas debidamente aprovisionadas (Center, 2016b).

Este cambio de paradigma representa una gran ventaja para las organizaciones en cuanto a costos, ya que resulta más rentable pagar por unas cuantas horas de computo en una máquina virtual robusta, çorriendo en la nube.º por el acceso a algún plataforma de aprendizaje de máquina como servicio, que invertir en hardware para aprovisionar una estación de trabajo local que sirva como plataforma para entrenar, evaluar o implementar los modelos (Forbes, 2012).

2.1. Aprendizaje de máquina

Desde una perspectiva general puede definirse como una disciplina enfocada en como construir sistemas que automáticamente mejoren de acuerdo a su experiencia sin ser programados para ello y descubrir cuales son las leyes fundamentales de la estadística-matemática-computación-información que gobiernan todos los sistemas de aprendizaje, incluyendo computadoras, seres humanos y organizaciones (Brownlee, 2015).

Por otra parte, desde una perspectiva más especifica, consiste en entrenar un modelo matemático o estadístico usando información histórica, con el fin de que pueda inferir o predecir el valor de una variable, la cuál puede ser discreta o continua, dependiendo el tipo de problema, clasificación o regresión respectivamente, en miras a comprender o explicar un fenómeno dado, realizar predicciones a futuro o hacer inferencias, en general a identificar patrones (Brownlee, 2015).

2.1.1. Conceptos

En esta sección se definen los conceptos fundamentales que hacen parte de la nomenclatura (términos estándar) usada cuando se esta describiendo un conjunto de datos o abordando un problema dado en términos de aprendizaje de máquina (Shalev-Shwartz & Ben-David, 2014):

1. **Matriz de Datos:** Cuando se piensa en datos, dicho termino es comúnmente asociado con bases de datos, tablas o quizás una hoja de calculo en excel. Una matriz es una

estructura de datos estándar, que tradicionalmente es usada para almacenar información en filas y columnas y se conserva para representar las entradas de un modelo o algoritmo de aprendizaje de máquina. Los componentes de una matriz de datos son:

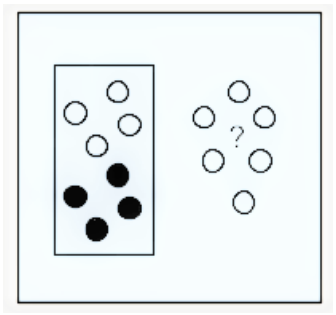
- **Instancia:** Una instancia corresponde a una observación realizada, un registro, o fila en una matriz de datos representada de la forma tradicional.
- **Característica:** En una matriz de datos las columnas representan las características o atributos de una instancia, también se les identifica como variables y las hay de 2 tipos de salida y de entrada, las primeras serán las que conformarán la matriz del entrenamiento del modelo y las ultimas las que se espera el modelo pueda predecir luego de ser evaluado.
- **Tipo de dato:** cada característica, es decir cada columna de la matriz de datos, tienen un tipo de dato definido, este puede ser real, entero o categórico (lista de valores), e incluso almacenar cadenas, fechas o tipos de datos mas complejos, sin embargo por lo general los algoritmos/modelos tradicionales reducen el conjunto de variables a tipo real o categórico.
- **Matriz de entrenamiento:** subconjunto de instancias de la matriz de datos que se usa para entrenar el modelo.
- **Matriz de prueba:** subconjunto de instancias de la matriz de datos que se usa para validar el desempeño o la tasa de reconocimiento de un modelo.

2. **Aprendizaje:** Según el contexto, el tipo de variables etc., existen diferentes modos a través de los cuales un modelo puede aprender de los datos, en esta sección se describirán algunos conceptos importantes con respeto al aprendizaje de los algoritmos (Brownlee, 2015).

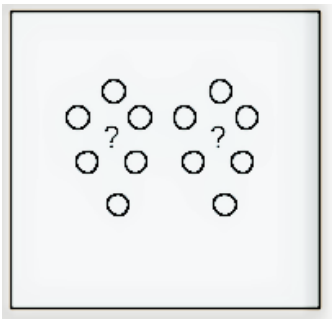
   - **Aprendizaje supervisado:** Como se muestra en la figura 2.1, Las instancias de la matriz de datos están clasificadas, es decir cada observación ha sido previamente categorizada, ejemplo: el conjunto de mensajes de correo electrónico, leídos, no leídos, correo no deseado. El modo de evaluar este tipo de modelos consiste en, primero, separar la matriz de datos en dos grupos, entrenamiento y pruebas respectivamente, luego de acuerdo al problema, esperar a que el modelo trate de forma autónoma predecir a que clase pertenece cada instancia si se trata de un problema de clasificación o inferir el valor de la variable de salida si el problema de regresión, finalmente

basado en el número de errores cometidos medir su exactitud. El proceso de entrenamiento se repite n veces hasta que el modelo logra un nivel de precisión deseado. Algunos algoritmos son: Regresión logística, Redes neuronales etc. (Shalev-Shwartz & Ben-David, 2014).

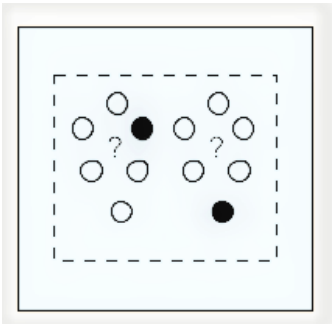
- **Aprendizaje no supervisado:** Tal como se muestra en la figura 2.1, este tipo de aprendizaje se aplica cuando las instancias del conjunto de datos no están categorizadas o clasificadas. Por lo general este tipo de modelos son usados para identificar o deducir estructuras comunes entre si que permitan agrupar las instancias (donde el identificador del grupo termina siendo la clase a la que la instancia pertenece). Esto puede ser extrayendo reglas generales a través de algún método matemático que permita identificar patrones similares(por ejemplo la distancia espacial), o simplemente agrupándolos teniendo en cuenta alguna medida de similitud (Shalev-Shwartz & Ben-David, 2014).
- **Aprendizaje semi-supervisado:** Como se puede visualizar en la figura 2.1, existen algunas instancias del conjunto de datos que están categorizadas, los cuales son usadas para inferir/predecir a que categoría pertenece cada una de las restantes o su respectivo valor en caso de que sea un problema de regresión. Esto es posible solo si la cantidad de instancias clasificadas es mayor al de no clasificadas, en caso tal de que dicha condición no se cumpla existen otras formas de abordar este tipo de problemas, como por ejemplo iniciar haciendo un análisis de clustering (usando modelos de aprendizaje no supervisado)(Shalev-Shwartz & Ben-David, 2014).



Aprendizaje supervisado



Aprendizaje no supervisado



Aprendizaje semi-supervisado

Figura 2.1: Tipos de aprendizaje

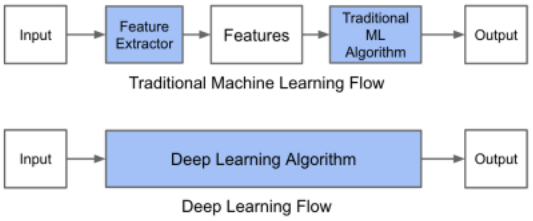


Figura 2.2: Enfoques del aprendizaje de máquina (Moujahid, 2016)

2.1.2. Enfoques

Existen dos enfoques diferentes de abordar un problema usando aprendizaje de máquina, según el contexto y los requerimientos, tal como se muestra en la figura 2.2. La diferencia radica en que mientras con **los modelos tradicionales** de aprendizaje de máquina como: SVM (máquina de vector de soporte), QDL (Discriminante lineal cuadrático), Knn (vecino más cercano) etc. Al momento del entrenamiento el conjunto de características debe estar definido, como se muestra en la figura 2.3, Por otro lado usando **Aprendizaje Profundo**, no es necesario,

puesto que el algoritmo realiza dicha tarea automáticamente extrayendo características de forma autónoma durante el entrenamiento (Bengio, 2016), ver figura 2.4.Lo anterior representa una ventaja significativa en términos de tiempo, puesto que en el enfoque tradicional el conjunto de variables seleccionadas debe ser cíclicamente refinado hasta lograr un buen resultado, lo cual solo es posible realizando ejecuciones metódicas con diferentes características y evaluar la exactitud del modelo. Existen diferentes técnicas y métodos para realizar la selección de características, entre ellas se encuentra el algoritmo *Greedy forward selection*, los algoritmos genéticos etc.(Bishop, 2006).

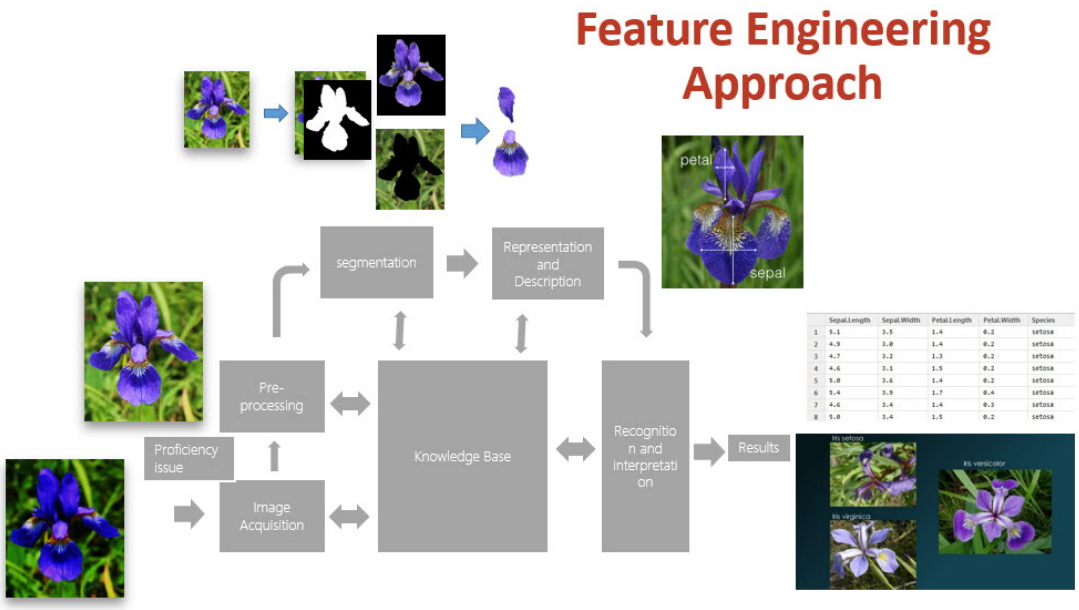


Figura 2.3: Enfoque tradicional

Otra de las diferencias representativa entre los dos enfoques esta en que mientras en el enfoque tradicional se habla de modelos refiriéndonos a los diferentes algoritmos que existen, en el aprendizaje profundo nos referimos a redes neuronales con arquitecturas de red diferentes(número de capas, nodos, salidas, entradas etc.) (Bengio, 2016).

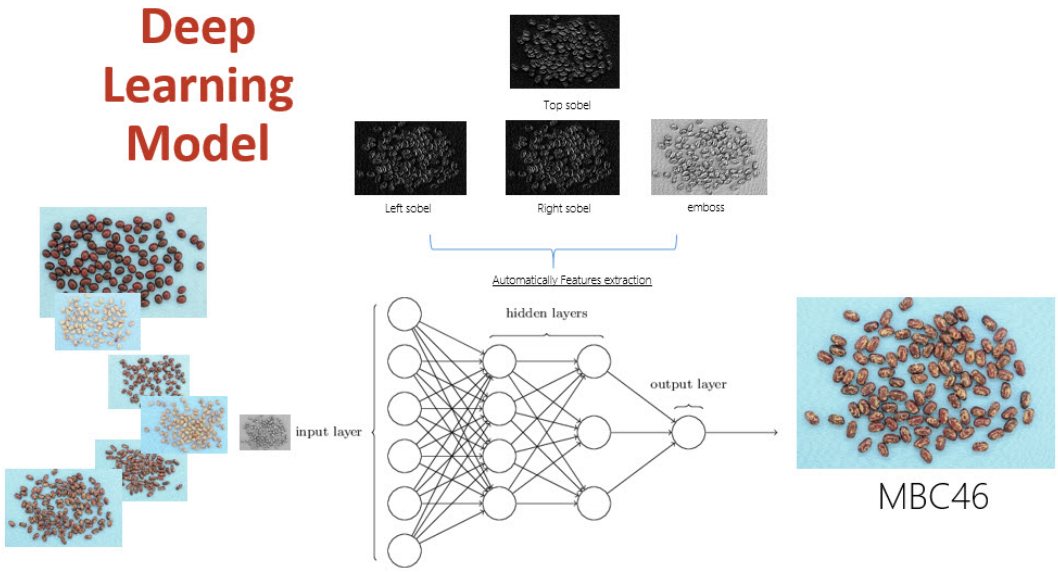


Figura 2.4: Aprendizaje profundo

1. **Aprendizaje Profundo**

   En la figura 2.5 se puede observar el corte transversal del hippocampus tomada del cerebro de un ratón, la cual permite evidenciar que dicho sistema está compuesto morfológicamente de diferentes capas, las cuales comparten información en forma de impulsos nerviosos a través de las neuronas. Basado en este concepto biológico, que ocurre además en el cerebro de los humanos, las redes neuronales son algoritmos bio-inspirados que de acuerdo a la complejidad de su arquitectura pueden tener n cantidad de capas (unidades de procesamiento) (Bengio, 2016).

2. **¿Por qué "Aprendizaje profundo" en lugar de "redes neuronales"?**

   hace unos 40 años atrás (aproximadamente), solo se hablaba de arquitecturas de red de solo 2 capas,ocasionalmente 3 o 4, puesto que en ese momento no se contaba con los recursos de hardware, ni la capacidad de computo que se tiene hoy día. Gracias a la computación en la nube y el computo en paralelo, en la actualidad es muy común encontrar modelos de redes neuronales incluso con mas de 100 capas, motivo por el cual se empezó a acuñar el concepto de aprendizaje profundo (Bengio, 2016). **Algunas Aplicaciones:**

   a) Visión por computadora: Vehículos autónomos, detección de enfermedades

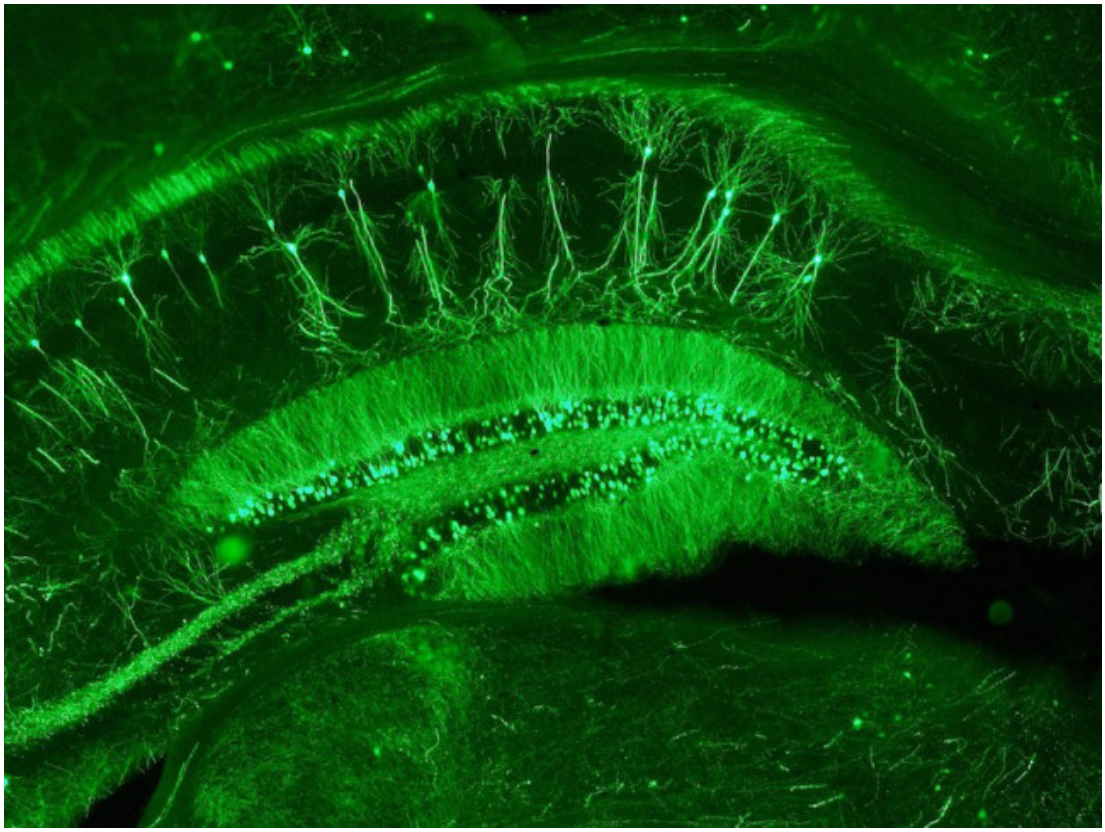


Figura 2.5: Hippocampus de un ratón (news for students, 2014).

b) Análisis de texto: Traducción automática, comprensión de documentos

c) Reconocimiento de voz: Traducción en tiempo real

2.2. Aplicaciones

Visión por computadora

La visión es uno de los sistemas más complejos del nuestro cuerpo, este nos permite obtener información del mundo real, fundamental para interactuar con nuestro entorno y los objetos que nos rodean (reconocer otras personas). Este sistema, que utiliza 3/4 de la capacidad de nuestro cerebro es tan fascinante, que existe un área de investigación conocida como visión por computadora enfocada específicamente al desarrollo de algoritmos que le permitan a las máquinas hasta cierto grado simular este proceso biológico (Gonzalez, 1977).

la también llamada visión artificial, es una disciplina científica que en los últimos años ha tomado mucha fuerza por sus aplicaciones y que forma parte de uno de los grandes campos de la inteligencia artificial, en esta se estudia cómo reconstruir, interpretar y entender una escena 3d a partir de imágenes 2d en términos de sus propiedades o estructuras presentes en la escena (Gonzalez, 1977).

En conclusión el objetivo principal de la visión por computadora, consiste en tratar de simular en una máquina (robot o cualquier otro dispositivo electrónico) el proceso que de forma natural realiza nuestro cerebro para analizar, entender e interpretar las imágenes capturadas por nuestros sistema visual(sensor, cámaras), apoyándose en áreas como el procesamiento digital de imágenes(para obtener información de interés de las imágenes tomadas por los sensores), machine learning (para a partir de dicha información interpretarlas), entre otras. Lo que le permitirá a la máquina reconocer o clasificar los objetos del entorno y tomar sus propias decisiones, tal como lo hacemos nosotros (Gonzalez, 1977).

La visión por computadora es una tecnología emergente que podemos encontrar integrada en muchos de los dispositivos y servicios que utilizamos en nuestras actividades cotidianas, un ejemplo de esto son las cámaras fotográficas, algunas de estas son capaces de reconocer rostros e incluso detectar si las personas están sonriendo en tiempo real (Szeliski, 2010).

Aplicaciones: Detección facial, reconocimiento de caracteres, autenticación biométrica, reconocimiento de objetos, efectos especiales, reconstrucción 3D, análisis de imágenes médicas, autos autónomos,realidad aumentada (Szeliski, 2010).

OpenCV

OpenCV es una biblioteca libre de visión artificial originalmente desarrollada por Intel. Desde que apareció su primera versión alfa en el mes de enero de 1999, se ha utilizado en infinidad de aplicaciones. Desde sistemas de seguridad con detección de movimiento, hasta aplicativos de control de procesos donde se requiere reconocimiento de objetos. Esto se debe a que su publicación se da bajo licencia BSD, que permite que sea usada libremente para propósitos comerciales y de investigación con las condiciones en ella expresadas (Oscar Denis Suarez, 2014). Open CV es multiplataforma, existiendo versiones para GNU/Linux, Mac OS X y Windows. Contiene más de 500 funciones que abarcan una gran gama de áreas en el proceso de visión, como reconocimiento de objetos (reconocimiento facial), calibración de cámaras, visión estereoscópica y robótica (Beyeler, 2017)

2.3. Sistemas Distribuidos

2.3.1. Arquitectura cliente servidor

Hoy casi todas las aplicaciones que se ejecutan en los dispositivos móviles (Smartphone/ Tablet) o equipos de trabajo, son aplicaciones distribuidas; es decir sus componentes, están distribuidos, valga la redundancia y operan de forma independiente en ambientes distintos (base de datos, servidor de aplicaciones, cliente) conectados a través de una red (cliente-middleware-servidor), la arquitectura cliente servidor básicamente nos permite modelar o representar este tipo de sistemas o aplicaciones.

En un sistema cliente-servidor las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores y los demandantes, llamados clientes. Un cliente realiza una petición y es el servidor quien atiende y da respuesta a esta. Este modelo aplica también cuando todos los recursos o componentes del sistema operan sobre una misma máquina (entorno) (Mohr, 1999). La gráfica 2.6 representa cómo opera una aplicación, sistema o software-cliente servidor y se visualizan cada uno de sus componentes.

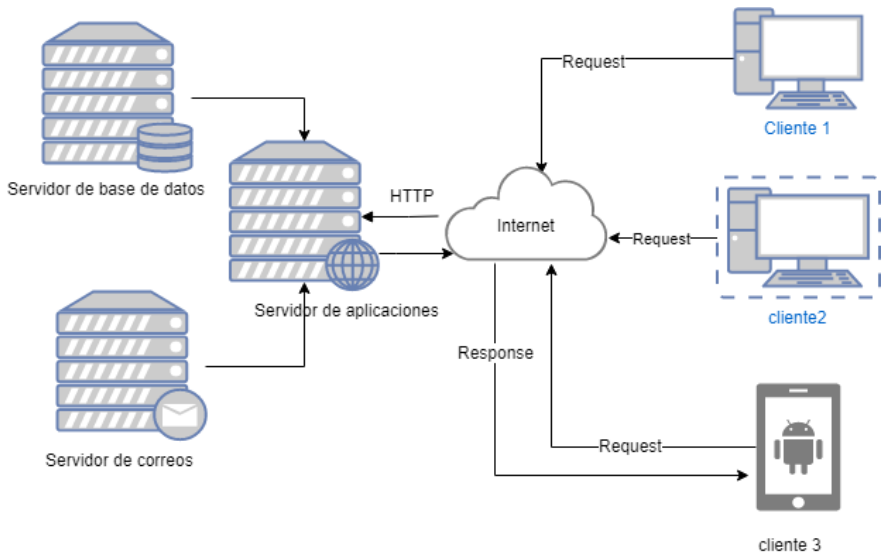


Figura 2.6: Arquitectura cliente servidor

Componentes

- **Browser:**Programa capaz de visualizar en tu equipo los archivos encontrados en la web como respuesta a una solitud que el usuario final realiza.
- **HTTP (Hypertext Transfer Protocol):**Protocolo (reglas) usado en la comunicación entre el cliente y el servidor (Berson, 1992).
- **Server:**computador, equipo o máquina que aloja las aplicaciones (Servidor de aplicaciones) o bases de datos (servidores de bases de datos) (Berson, 1992).
- **Internet:**canal de comunicación entre el cliente y el servidor (Berson, 1992)
- **Servidor de aplicaciones:**Un servidor web de aplicaciones o web server, es el encargado de atender y responder las solicitudes o peticiones (request) entrantes (síncronas y asíncronas) de los clientes o usuarios de nuestras aplicaciones y de servir el contenido (web pages - response) al cual estos quieren acceder. Usando el modelo cliente servidor y el protocolo Http que traduce Hypertext Transfer Protocol definido por la world Wide web (Berson, 1992). Además sirve como plataforma de ejecución de aplicaciones y permiten que el contenido de las páginas sea dinámico, ya que usando lenguajes de programación como java, Python, php para su desarrollo (código que será interpretado y compilado por el servidor de web) es posible generar contenido HTML dinámicamente(Por ejemplo a partir de la información cargada de una base de datos). El servidor de aplicaciones a usar depende de la tecnología elegida para desarrollar la aplicación. PhP/Python – apache, java-tomcat/glassfish, .net-IIS.

2.3.2. Frontend vs. Backend

EL **backend**, es la capa de aplicación responsable de procesar la información capturada y enviada desde los clientes a través de los data entry o formularios de entrada que constituyen el **frontend**, por otro lado el frontend es la capa más superficial de un sistema y con la cual interactúan de forma directa los usuarios finales (UI – User Interface) (Wikipedia, 2013). Para la construcción del backend de las aplicaciones se pude usar cualquier lenguaje de programación al lado del servidor como por ejemplo Python, Java, Ruby etc. Por otro lado para el Frontend, si hablamos de una aplicación móvil usamos el SDK que el fabricante del SO instalado en el móvil nos provea, por ejemplo para Android usamos java, para Windows phone 8 Vb o

para iOS Objective-c, si es una aplicación web, los formularios se construyen usando html5, css3 y JavaScript tecnologías que son ejecutadas al lado del cliente es decir directamente en el navegador (Wikipedia, 2013).

2.3.3. Computación en la nube

Según el NIST (National Institute of Standards and Technology) Cloud Computing es un modelo para habilitar el acceso por demanda, vía estándares WEB, a recursos de cómputo compartidos y configurables que pueden ser rápidamente provisionados y liberados con un mínimo esfuerzo de administración o de interacción por parte del proveedor de los servicios (Erl, 2013).

Citando un ejemplo, Google Apps Engine ofrece una plataforma en la nube para la creación y ejecución de aplicaciones. Actualmente soporta aplicaciones desarrolladas en Python, Java, PHP y Go (Google, 2018a).

GAE nos permite usar el API de java en su totalidad, "sin restricción alguna" y además nos provee su propia API para interactuar con los servicios que nos provee (almacenamiento no estructurado, correo). GAE Usa Jetty Servlet como como contenedor de aplicaciones (hosting) y soporta Java Servlet API in versión 2.4, provee dos modelos de almacenamiento estructurado y no estructurado, acceso vía Java Data Objects (JDO) y Java Persistence API (JPA), nos provee además varios servicios, por ejemplo el de BlobStore para la carga y almacenamiento de archivos hasta de 2gb, vía Http y el manejador de versiones (aplicaciones) (Google, 2018a).

2.4. Revisión de plataformas similares

Las plataformas de *machine learning as a service*, son soluciones de software en la nube que proveen a sus usuarios la infraestructura y herramientas necesarias para que estos puedan aprovisionar sus aplicaciones con funcionalidades como: el reconocimiento facial, análisis predictivo, visión por computadora, sentimental análisis, reconocimiento de objetos etc. Por lo general a través de APIs para diferentes lenguajes de programación, vía servicios REST o desde la interfaz web.A continuación se citan algunas de las mas populares.

Amazon Machine Learning: Provee un marco de ejecución guiado dentro del cual podremos crear, depurar y evaluar nuestros modelos, cuenta además con herramientas de visualización sofisticadas, y nos permite conectarnos a fuentes de datos externas como el *storage de amazon,*

Redshift, or RDS para obtener los datos del modelo (Amazon, 2018).

Algorithmia: Provee toda la infraestructura necesaria para que los desarrolladores puedan construir, validar, implementar y poner en marcha sus algoritmos; Estos quedan organizados en diferentes categorías, por ejemplo: ML, Reconocimiento facial, procesamiento de imágenes etc. y una vez validados, es decir que ya estén públicos, otros desarrolladores pueden desde sus aplicaciones llamaros vía HTTP/REST, sin tener que preocuparse por la infraestructura que requieren para su ejecución. La plataforma cuenta con diferentes modos de licenciamiento: *Free Forever, Pay as you go, Enterprise* (Algorithmia, 2018).

BigML: Plataforma en la nube que a través de una interfaz gráfica permite al usuario construir e implementar modelos de ML sobre servidores en la nube. Requiere de conocimientos previos puesto que la interfaz no es muy intuitiva, vía REST podemos desde cualquier cliente acceder a dichos modelos (BigMl, 2018).

Machine Learning Azure Studio: Basado en programación en bloques(drag and drop), permite a través de una interfaz similar a la de scratch construir modelos de ML y publicarlos para que puedan ser invocados vía REST. Incluye algoritmos para *aprendizaje supervisado (predictive models), aprendizaje no supervisado (understand behavior), anomaly detection (used in fraud detection), Visualización de datos etc.* Otra de sus bondades esta en que través de scripts en R o python nos permite agregar código propio a los modelos. Suele ser muy complejo al inicio (Microsoft, 2018).

Google Prediction APIs: Es un conjunto de mas de 60 servicios web que incluyen reconocimiento facial, procesamiento de audio y vídeo, sentimental análisis y detección de objetos en imágenes que pueden ser invocados en desarrollos propios vía REST o usando las librerías para python, .Net, javascript etc. De acuerdo a la cantidad de peticiones y la ubicación varía el precio. Hasta 100 millones por mes, el valor en EE.UU es de 0.10/mil USD y en Asia o Europa es de 0.11/mil USD. Para solicitudes de más de 100 millones por mes el valor es 0.05/mil USD dentro de EE.UU y 0.05/mil USD para Europa (Google, 2018b).

IBM's Watson Analytics ofrece análisis predictivo y visualización de datos, y una interfaz muy útil si queremos encontrar patrones y relaciones en nuestros datos. Se ofrece una versión

gratuita con limitaciones en los volúmenes de datos (IBM, 2018).

De la tabla 2.1 donde se resaltan las diferencias mas relevantes entre algunas de las soluciones previamente citadas, se puede concluir que la mayoría de dichas herramientas tienen costo y además no cuentan con una interfaz amigable con el usuario final.Lo anterior,puesto que por lo general el publico objetivo hacía el que están dirigidas, son a desarrolladores de software con conocimientos en lenguajes de programación o expertos en machine learning.

Product	**Free**	**Api**	**User Models**	**Coding Skills**	**Mobile App**	**Interface**
Machine Learning Studio Microsoft Azure	NO	YES	YES	YES	NO	1. Azure Machine Learning Studio drag-and-drop enviroment 2. Paquetes para R y python 3. Vía API REST
Cognitive Services Microsoft	NO	YES	NO	YES	NO	1. Vía API REST 2. Paquetes para .Net,python y node.js
Google Prediction APIs	NO	YES	NO	YES	NO	1. Vía linea de comandos 2. Vía REST 3. Paquetes para python y node.js
Watson, IBM	NO	YES	NO	YES	NO	1. Vía API REST 2. Paquetes para python y node.js 3. SPSS software de IBM
BigML	YES	YES	YES	YES	NO	1. Vía API REST 2. Paquetes para python y node.js 3. Vía web

Tabla 2.1: Tabla de comparación de Soluciones de Machine Learning como servicio, Resumen

Capítulo 3

Desarrollo de la plataforma MlTool UABC

3.1. Descripción general del software

El requerimiento con la plataforma en contexto, es ofrecer al usuario final una herramienta que le permita implementar, adaptar, ejecutar y validar diferentes modelos de aprendizaje de máquina a través de un interfaz sencilla y amigable usando sus datos. Los modelos considerados en esta versión son:

1. Preprocessing
    a) Standardization
    b) Normalization
    c) Outliers Removal
    d) Data imputation
    e) Data discretization
2. Classification models
    a) Naive Bayes
    b) Logistic regression
    c) Knn (k-nearest neighbors)
    d) Svm(Support vector machine)
    e) QDC
    f) Neuronal Networks (MLP)
3. Clustering & Dimensionality reduction
    a) K-means
    b) PCA Visualization
4. Validation
    a) Cross validation (nxk)

3.2. Metodología de desarrollo

Para la construcción del software se seleccionó como metodología de desarrollo la programación extrema (X.P, por sus siglas en ingles). Formulada por Kent Beck en el año 1999 en su libro Extreme Programming Explained: Embrace Change, esta metodología ágil se caracteriza entre otras cosas, por permitir que los requerimientos funcionales del sistema, puedan ser ajustados en cualquier momento durante el desarrollo del producto, lo que la hace muy flexible. Algunos de los características que se pueden resaltar son:

1. **Integración continua:** consiste en implementar progresivamente nuevas características al software, el cual puede estar en producción. En lugar de lanzar versiones en función de una planificación previamente realizada, los programadores se reúnen frecuentemente y reconstruyen el proyecto varias veces al día si hace falta.

2. **Refactorización:** mediante la constante eliminación de código duplicado y/o ineficiente los equipos de programación mejoran el diseño del sistema. El código se evalúa continuamente para ofrecer la mayor calidad posible.

3. **Entregas pequeñas:** el producto es evaluado en un ambiente real mediante la colocación de un sistema sencillo en producción el cual se actualizará rápidamente, es decir, cada 2 semanas (3 como máximo) el software será puesto en producción.

En la siguiente sección, se describen las fases de desarrollo de la plataforma web MlTool UABC.

3.2.1. Fase 1: Planificación del proyecto

Historias de usuario: El primer paso de cualquier proyecto que siga la metodología X.P es definir las historias de usuario con el cliente. Las historias de usuario tienen la misma finalidad que los casos de uso, con algunas diferencias: Constan de 3 ó 4 líneas escritas por el cliente en un lenguaje no técnico sin hacer mucho hincapié en los detalles; no se debe hablar ni de posibles algoritmos, base de datos etc. Son usadas para estimar tiempos de desarrollo de la parte de la aplicación en la que se esta trabajando. El tiempo de desarrollo ideal para una historia de usuario es entre 1 y 3 semanas.

Como evidencia de esta fase, a continuación se encuentran las historias de usuario para los casos de uso mas relevantes.

Módulo de Usuarios

Historia #:	001	Titulo:	Registrar Usuario		
Programador:	Henry A. Ruiz	Role:	Non-Administrador	Sprint:	1
Descripción	COMO	Non-Administrador			
	QUIERO	poder desde la pagina de login acceder al formulario de registro para crear una nueva cuenta. ingresando mi Nombre Completo, correo electrónico, login, contraseña, ocupación y biografía			
	PARA	Poder acceder a la plataforma			
Criterios de validación:	El campo Nombre completo será obligatorio, de máximo 40 caracteres. El campo correo electrónico será obligatorio, de máximo 200 caracteres El campo biografía será opcional, de máximo 500 caracteres El campo contraseña será obligatorio, de máximo 25 caracteres El campo login será obligatorio y único, máximo 15 caracteres			Valor negocio:	120
				Prioridad negocio	M
				Puntos estimados	30

Tabla 3.1: Historia 001

Historia #:	002	Titulo:	Iniciar sesión		
Programador:	Henry A. Ruiz	Role:	Non-Administrador	Sprint:	1
Descripción	COMO	Non-Administrador			
	QUIERO	poder ingresar al sistema, ingresando el usuario y la contraseña en el formulario de login 3.1			

	PARA	Iniciar sesión		
Criterios de validación:	El campo password será obligatorio, de máximo 25 caracteres El campo login será obligatorio y unico, máximo 15 caracteres		Valor negocio:	120
			Prioridad negocio	A
			Puntos estimados	30

Tabla 3.2: Historia 002

Historia #:	003	Titulo:	Recuperar contraseña		
Programador:	Henry A. Ruiz	Role:	Non-Administrador	Sprint:	1
Descripción	COMO	Non-Administrador			
	QUIERO	poder recuperar mi contraseña ingresando el correo electrónico			
	PARA	Recuperar contraseña			
Criterios de validación:	El sistema deberá validar si el usuario existe usando el correo			Valor negocio:	120
				Prioridad negocio	A

		Puntos estimados	30

Tabla 3.3: Historia 003

Historia #:	004	Titulo:	Cambiar rol		
Programador:	Henry A. Ruiz	Role:	Administrador	Sprint:	1
Descripción	COMO	Administrador			
	QUIERO	poder cambiar el rol de un usuario a administrador, interfaz 3.2.			
	PARA	Habilitar funciones de administración			
Criterios de validación:				Valor negocio:	120
				Prioridad negocio	B
				Puntos estimados	30

Tabla 3.4: Historia 004

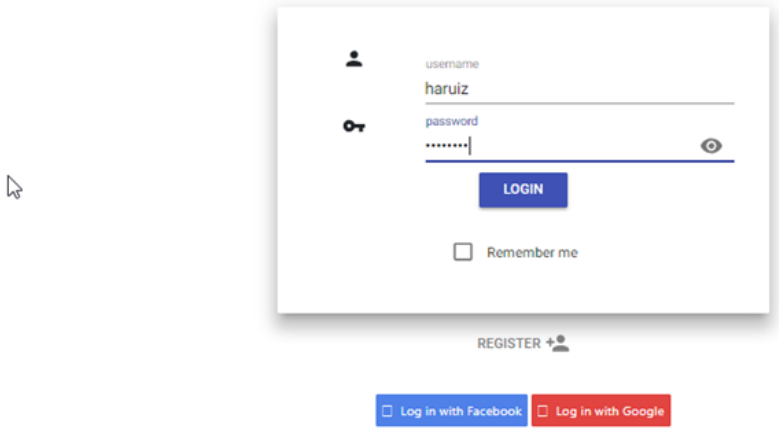


Figura 3.1: Formulario de login

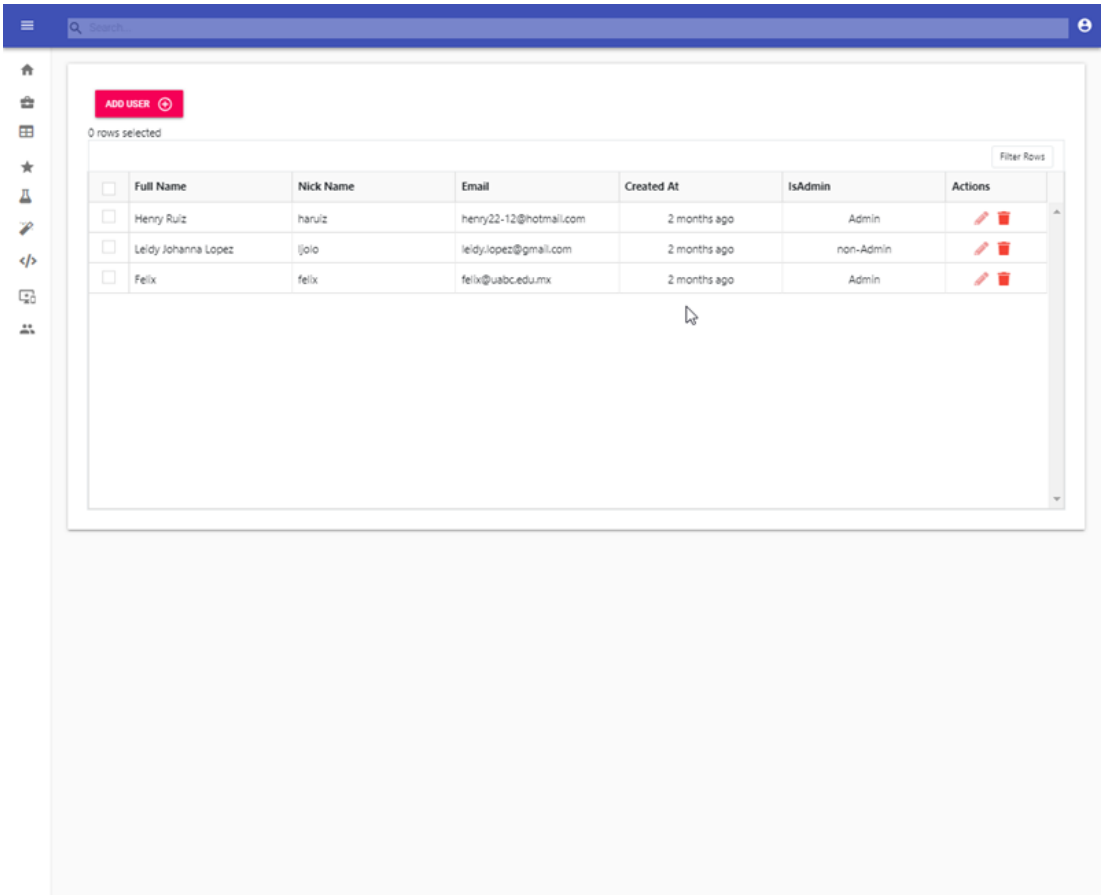


Figura 3.2: Formulario de usuarios

Módulo Workspaces

<table>
<tr><td>Historia #:</td><td>005</td><td>Titulo:</td><td colspan="3">Crear espacio de trabajo</td></tr>
<tr><td>Programador:</td><td>Henry A. Ruiz</td><td>Role:</td><td>Non-Administrador</td><td>Sprint:</td><td>1</td></tr>
<tr><td rowspan="3">Descripción</td><td>COMO</td><td colspan="4">Non-Administrador</td></tr>
<tr><td>QUIERO</td><td colspan="4">Desde el menú workspaces del menú de navegación,
poder acceder a la pagina de gestión de espacios de trabajo y
desde allí crear uno nuevo. el usuario deberá ingresar:
El nombre del espació de trabajo (eje: Mis proyectos) y una
descripción (eje: proyectos de curso)</td></tr>
<tr><td>PARA</td><td colspan="4">Crear un nuevo espacio de trabajo</td></tr>
<tr><td rowspan="3">Criterios
de validación:</td><td colspan="3" rowspan="3">El campo nombre del espació e trabajo es
requerido y debe de tener máximo 30 caracteres
El campo descripción será opcional y máximo puede
tener 200 caracteres</td><td>Valor negocio:</td><td>200</td></tr>
<tr><td>Prioridad
negocio</td><td>A</td></tr>
<tr><td>Puntos
estimados</td><td>30</td></tr>
</table>

Tabla 3.5: Historia 005

Historia #:	006	Titulo:	Editar espacio de trabajo		
Programador:	Henry A. Ruiz	Role:	Non-Administrador	Sprint:	1
Descripción	COMO	Non-Administrador			
	QUIERO	Desde el menú workspaces del menú de navegación, poder acceder a la pagina de gestión de espacios de trabajo y desde allí editar un espació de trabajo existente, figura 3.3,. el usuario podrá actualizar el nombre y la descripción			
	PARA	Editar un espacio de trabajo			
Criterios de validación:	El campo nombre del espació e trabajo es requerido y debe de tener máximo 30 caracteres El campo descripción será opcional y máximo puede tener 200 caracteres			Valor negocio:	200
				Prioridad negocio	A
				Puntos estimados	30

Tabla 3.6: Historia 006

Historia #:	007	Titulo:	Eliminar espacio de trabajo		
Programador:	Henry A. Ruiz	Role:	Non-Administrador	Sprint:	1
Descripción	COMO	Non-Administrador			
	QUIERO	Desde el menú workspaces del menú de navegación, poder acceder a la pagina de gestión de espacios de trabajo y desde allí eliminar un espació de trabajo existente. Al eliminar un espacio de trabajo todos los archivos de datos dentro de este se eliminaran			
	PARA	eliminar espacio de trabajo			

Criterios de validación:		Valor negocio:	200
		Prioridad negocio	M
		Puntos estimados	30

Tabla 3.7: Historia 007

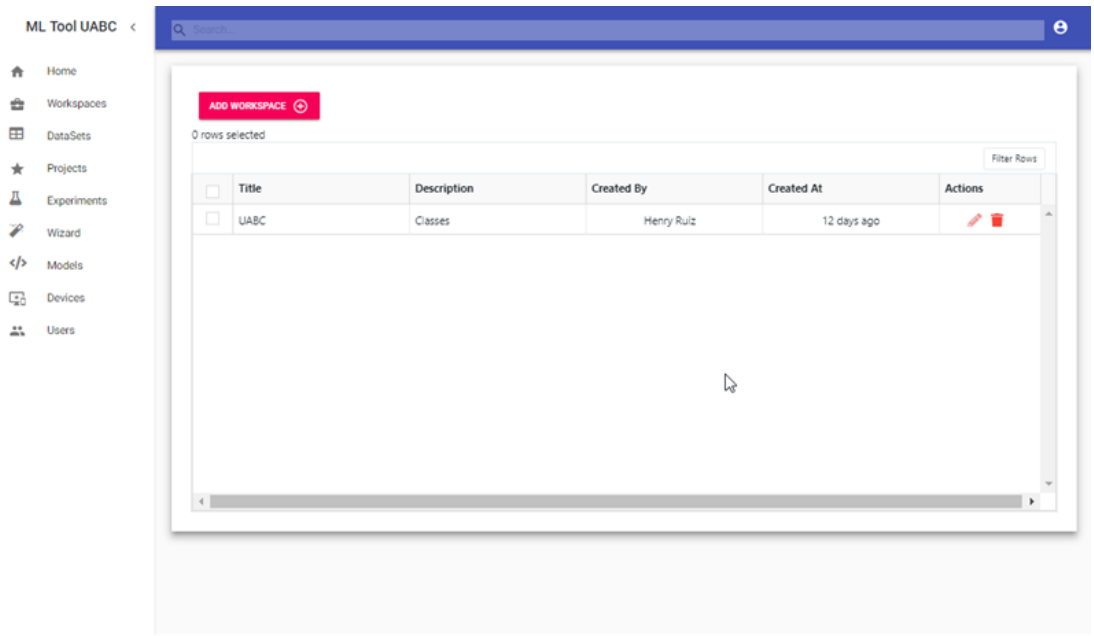


Figura 3.3: Página Espacios de trabajo

Módulo de conjunto de datos

<table>
<tr><td>Historia #:</td><td>008</td><td>Titulo:</td><td colspan="3">Cargar archivo de datos</td></tr>
<tr><td>Programador:</td><td>Henry A. Ruiz</td><td>Role:</td><td>Non-Administrador</td><td>Sprint:</td><td>1</td></tr>
<tr><td rowspan="3">Descripción</td><td>COMO</td><td colspan="4">Non-Administrador</td></tr>
<tr><td>QUIERO</td><td colspan="4">Desde el menú datasets del menú de navegación, poder acceder a la pagina de archivos de datos y desde allí cargar uno nuevo. los campos a ingresar serán : titulo, descripción, workspace, archivo .csv o .xls, figura 3.4</td></tr>
<tr><td>PARA</td><td colspan="4">Cargar un archivo de datos</td></tr>
<tr><td rowspan="3">Criterios de validación:</td><td colspan="3" rowspan="2">El campo titulo será requerido, máximo de 30 caracteres
El campo descripción será opcional, máximo de 50 caracteres
El campo workspace, será una lista desplegable donde aparecerán los workspaces creados por el usuario</td><td>Valor negocio:</td><td>200</td></tr>
<tr><td>Prioridad negocio</td><td>M</td></tr>
<tr><td colspan="3">EL campo archivo será un File Input, requerido y que permitirá solo archivos .csv o .xls.</td><td>Puntos estimados</td><td>30</td></tr>
</table>

Tabla 3.8: Historia 008

Historia #:	009	Titulo:	Editar un archivo de datos		
Programador:	Henry A. Ruiz	Role:	Non-Administrador	Sprint:	1
Descripción	COMO	Non-Administrador			
	QUIERO	Desde el menú datasets en el menú de navegación, poder acceder a la pagina de archivos de datos y desde allí editar un dataset. Los campos modificables serán:el titulo, la descripción y el workspace			
	PARA	Editar un archivo de datos			
Criterios de validación:	El campo titulo será requerido, máximo de 30 caracteres El campo descripción será opcional, máximo de 50 caracteres El campo workspace, será una lista desplegable donde aparecerán los workspaces creados por el usuario			Valor negocio:	200
				Prioridad negocio	M
				Puntos estimados	30

Tabla 3.9: Historia 009

Historia #:	010	Titulo:	Eliminar un archivo de datos		
Programador:	Henry A. Ruiz	Role:	Non-Administrador	Sprint:	1
Descripción	COMO	Non-Administrador			
	QUIERO	Desde el menú datasets en el menú de navegación, poder acceder a la pagina de archivos de datos y desde allí eliminar un dataset.			
	PARA	Eliminar un archivo de datos			

Criterios de validación:	El sistema solo permitirá que el dataset sea eliminado si no tiene experimentos asociados	Valor negocio:	200
		Prioridad negocio	M
		Puntos estimados	30

Tabla 3.10: Historia 010

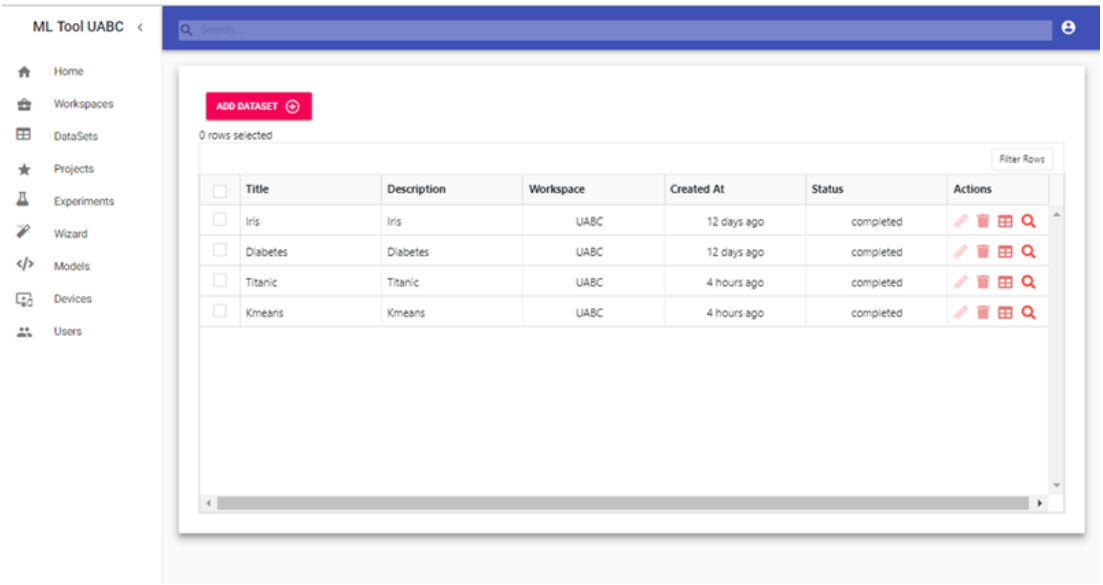


Figura 3.4: Página de conjunto de datos

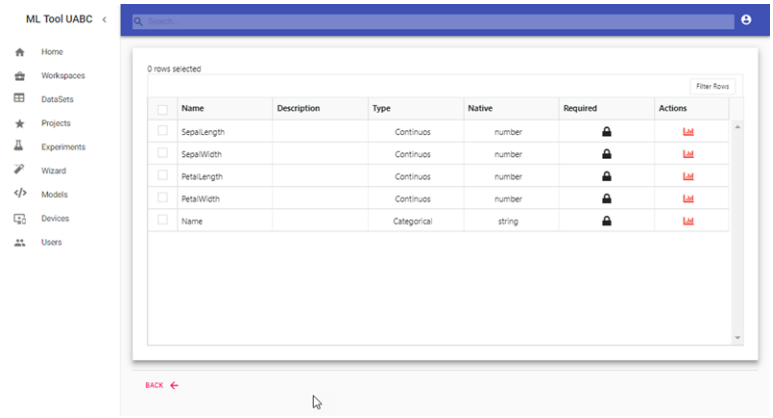


Figura 3.5: Pagina de visualización de las variables del conjunto de datos

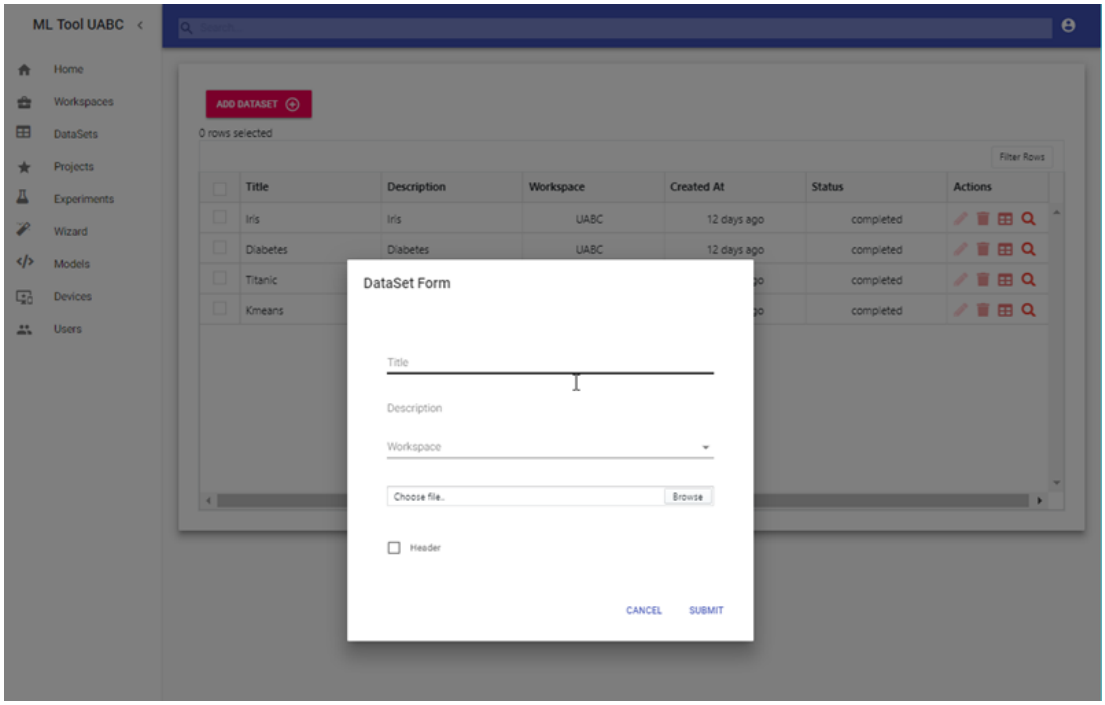


Figura 3.6: Formulario de carga de datos

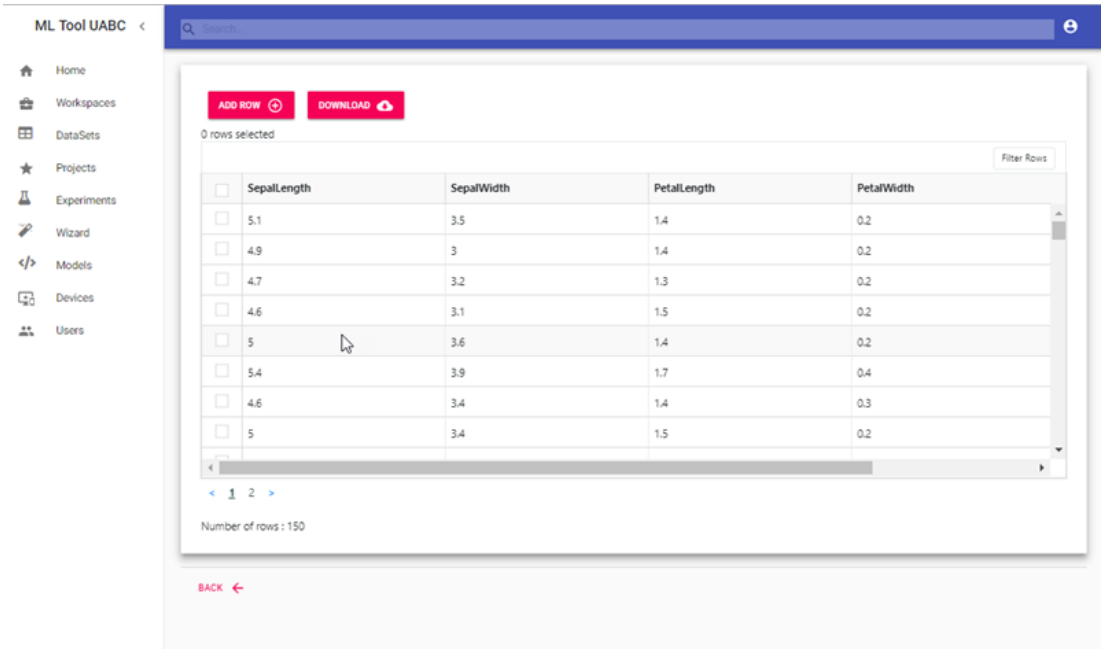


Figura 3.7: Formulario de visualización de los datos

Figura 3.8: Gráfico de distribución de variable seleccionada

Módulo de experimentos

<table>
<tr><td>Historia #:</td><td>011</td><td>Titulo:</td><td colspan="3">Crear un experimento</td></tr>
<tr><td>Programador:</td><td>Henry A. Ruiz</td><td>Role:</td><td>Non-Administrador</td><td>Sprint:</td><td>1</td></tr>
<tr><td rowspan="3">Descripción</td><td>COMO</td><td colspan="4">Non-Administrador</td></tr>
<tr><td>QUIERO</td><td colspan="4">Desde el menú experiments en el menú de navegación, poder acceder a la pagina de experimentos creados y desde allí crear un experimento. los campos serán : titulo, descripción, hipótesis, palabras claves, variables de entrada, variable de salida (opcional), parámetros y modelo 3.10.</td></tr>
<tr><td>PARA</td><td colspan="4">Crear un experimento</td></tr>
<tr><td rowspan="3">Criterios de validación:</td><td colspan="3" rowspan="3">El campo titulo será requerido, máximo 25 caracteres.
El campo descripción será opcional, máximo de 500 caracteres
El campo hipótesis será opcional, max 500 caracteres
El campo palabras claves será opcional, max 5 palabras claves
Las variables de entrada, el usuario deberá seleccionarlas al momento de crear el experimento
La variable de salida, deberá ser seleccionada por el usuario al momento de crear el experimento(opcional)</td><td>Valor negocio:</td><td>200</td></tr>
<tr><td>Prioridad negocio</td><td>M</td></tr>
<tr><td>Puntos estimados</td><td>30</td></tr>
</table>

Tabla 3.11: Historia 011

<table>
<tr><td>Historia #:</td><td>012</td><td>Titulo:</td><td colspan="3">Editar un experimento</td></tr>
<tr><td>Programador:</td><td>Henry A. Ruiz</td><td>Role:</td><td>Non-Administrador</td><td>Sprint:</td><td>1</td></tr>
<tr><td rowspan="3">Descripción</td><td>COMO</td><td colspan="4">Non-Administrador</td></tr>
<tr><td>QUIERO</td><td colspan="4">Desde el menú experiments en el menú de navegación,
poder acceder a la pagina de experimentos creados y
desde allí editar un experimento en curso.
los campos modificables serán :
titulo, descripción, hipótesis, palabras claves,
variables de entrada, variable de salida (opcional),
parámetros y modelo, interfaz 3.16.</td></tr>
<tr><td>PARA</td><td colspan="4">editar un experimento</td></tr>
<tr><td rowspan="3">Criterios
de validación:</td><td colspan="3" rowspan="3">El campo titulo será requerido, máximo 25
caracteres.
El campo descripción será opcional, máximo de
500 caracteres
El campo hipótesis será opcional, max 500 caracteres
El campo palabras claves será opcional, max 5 palabras
claves</td><td>Valor negocio:</td><td>200</td></tr>
<tr><td colspan="3" rowspan="2">Las variables de entrada, el usuario deberá seleccionarlas
al momento de editar el experimento
La variable de salida, deberá ser seleccionada por el
usuario al momento de editar el experimento(opcional,
dependerá del modelo)</td><td>Prioridad
negocio</td><td>M</td></tr>
<tr><td>Puntos
estimados</td><td>30</td></tr>
</table>

Tabla 3.12: Historia 012

<table>
<tr><td>Historia #:</td><td>013</td><td>Titulo:</td><td colspan="3">Eliminar un experimento</td></tr>
<tr><td>Programador:</td><td>Henry A. Ruiz</td><td>Role:</td><td>Non-Administrador</td><td>Sprint:</td><td>1</td></tr>
<tr><td rowspan="3">Descripción</td><td>COMO</td><td colspan="4">Non-Administrador</td></tr>
<tr><td>QUIERO</td><td colspan="4">Desde el menú experiments en el menú de navegación,
poder acceder a la pagina de experimentos creados y
desde allí eliminar un experimento.</td></tr>
<tr><td>PARA</td><td colspan="4">Eliminar un experimento</td></tr>
</table>

Criterios de validación:	El sistema deberá validar que el experimento no este en ejecución, de lo contrario no podrá ser eliminado	Valor negocio:	200
		Prioridad negocio	M
		Puntos estimados	30

Tabla 3.13: Historia 013

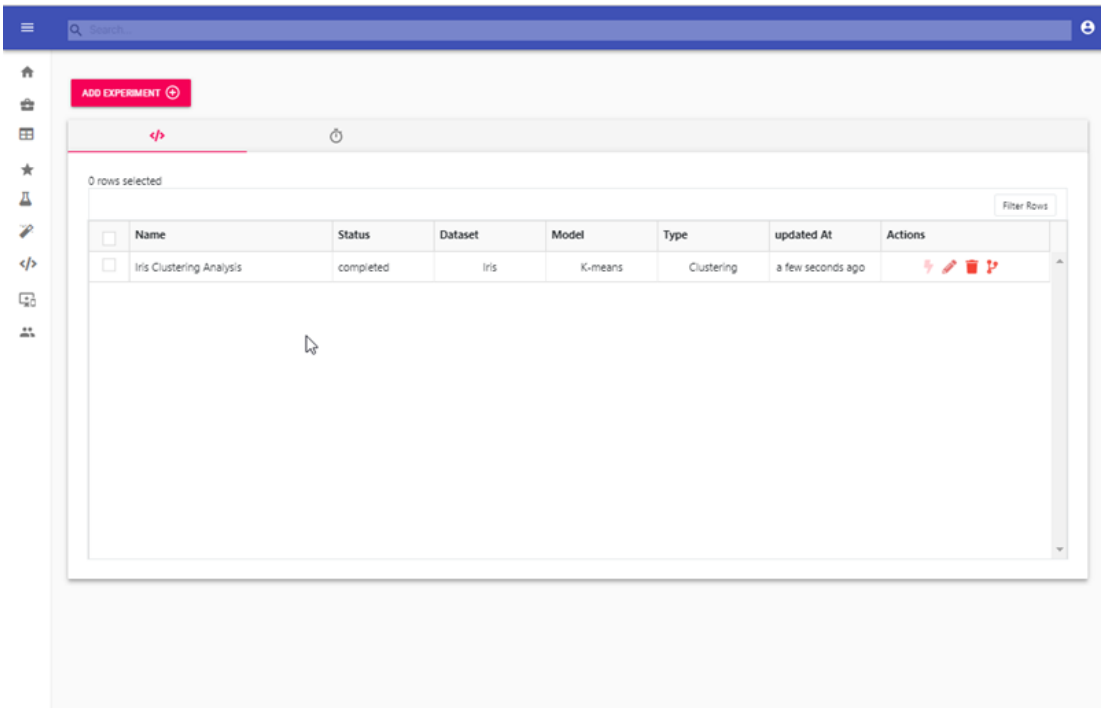


Figura 3.9: Formulario de experimentos

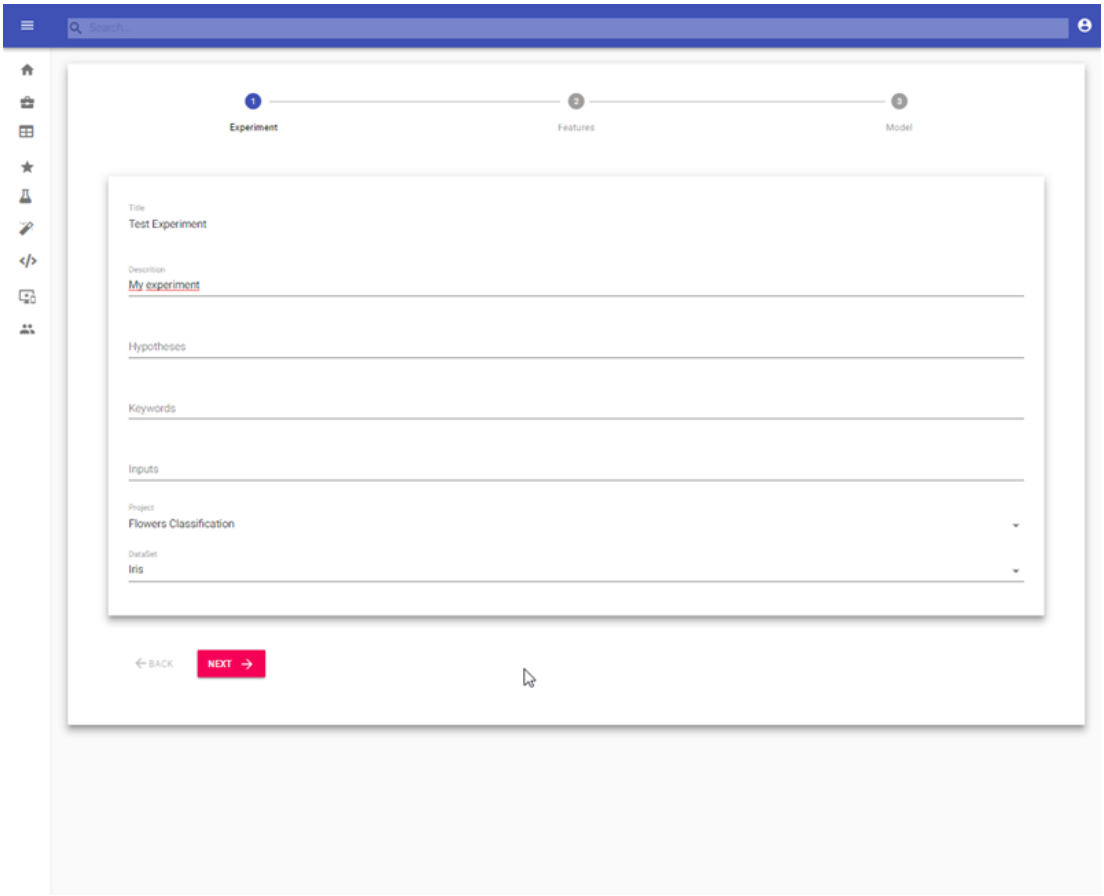


Figura 3.10: Asistente de creación de experimentos, Página 1

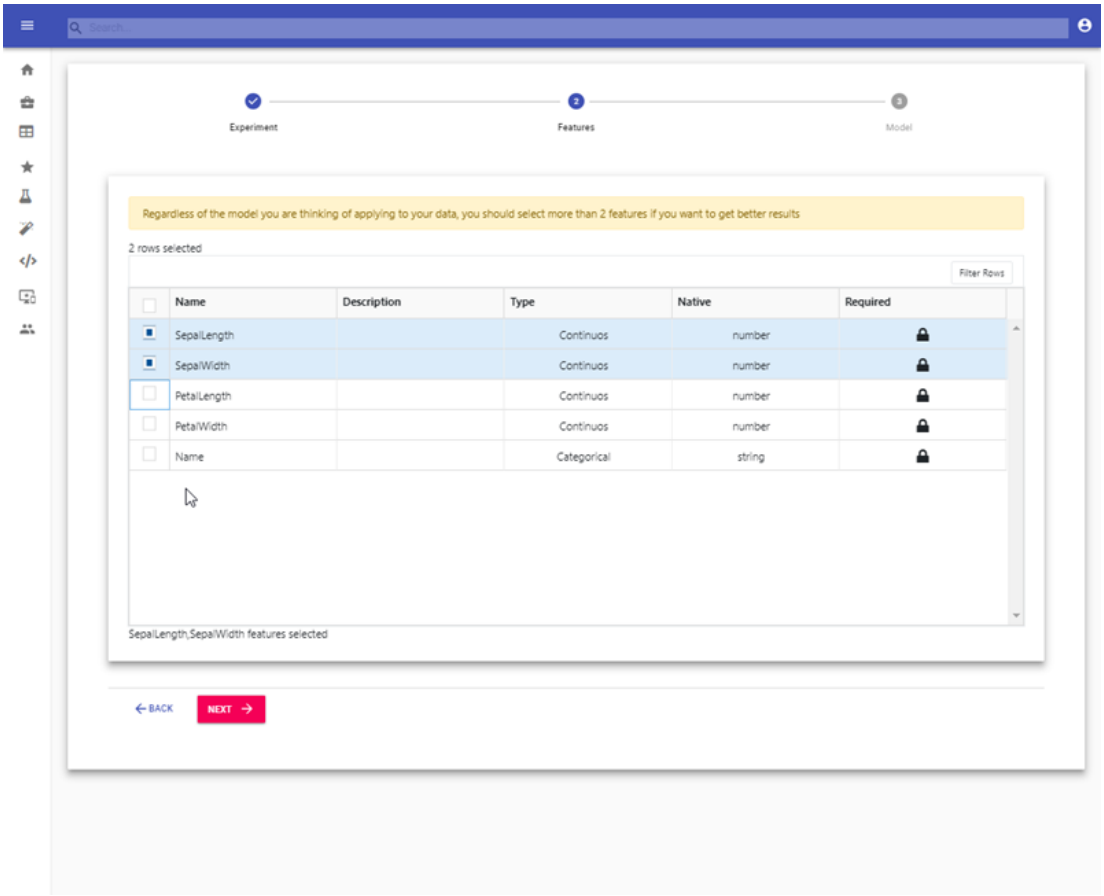


Figura 3.11: Asistente de creación de experimentos, Página 2

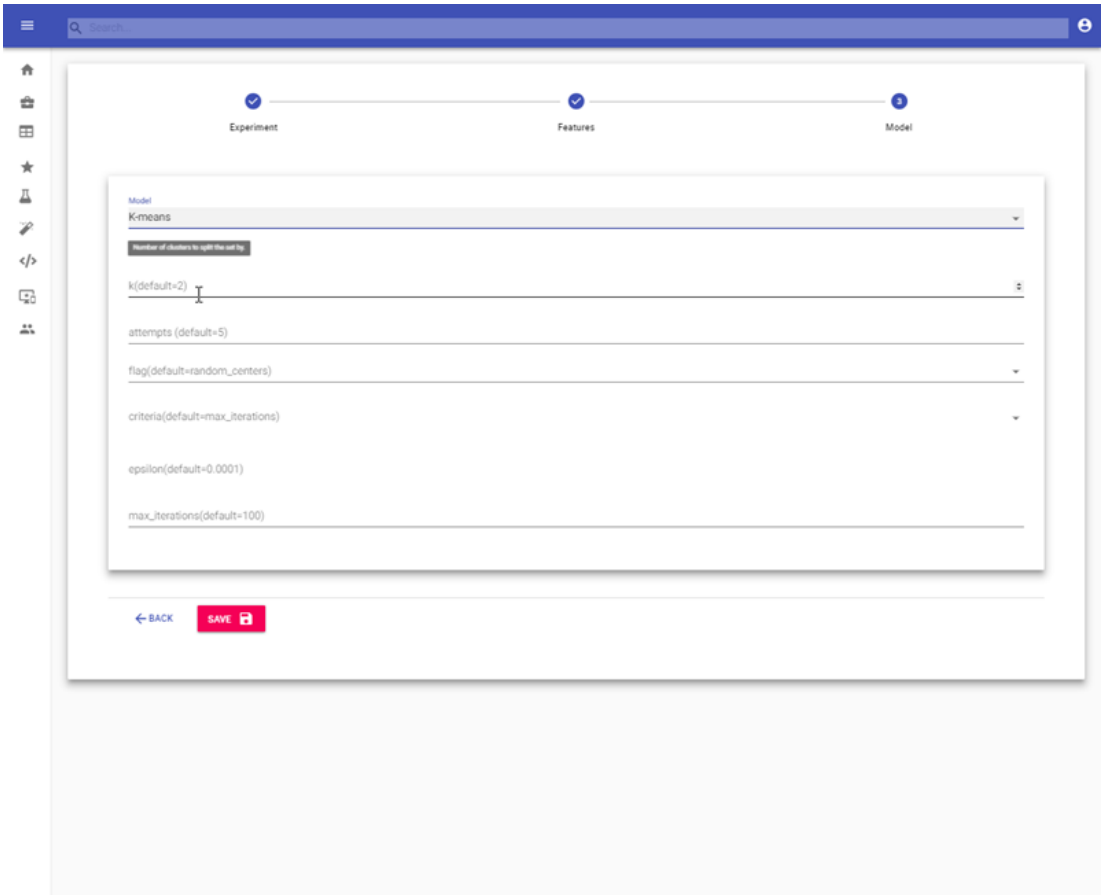


Figura 3.12: Asistente de creación de experimentos, Página 3

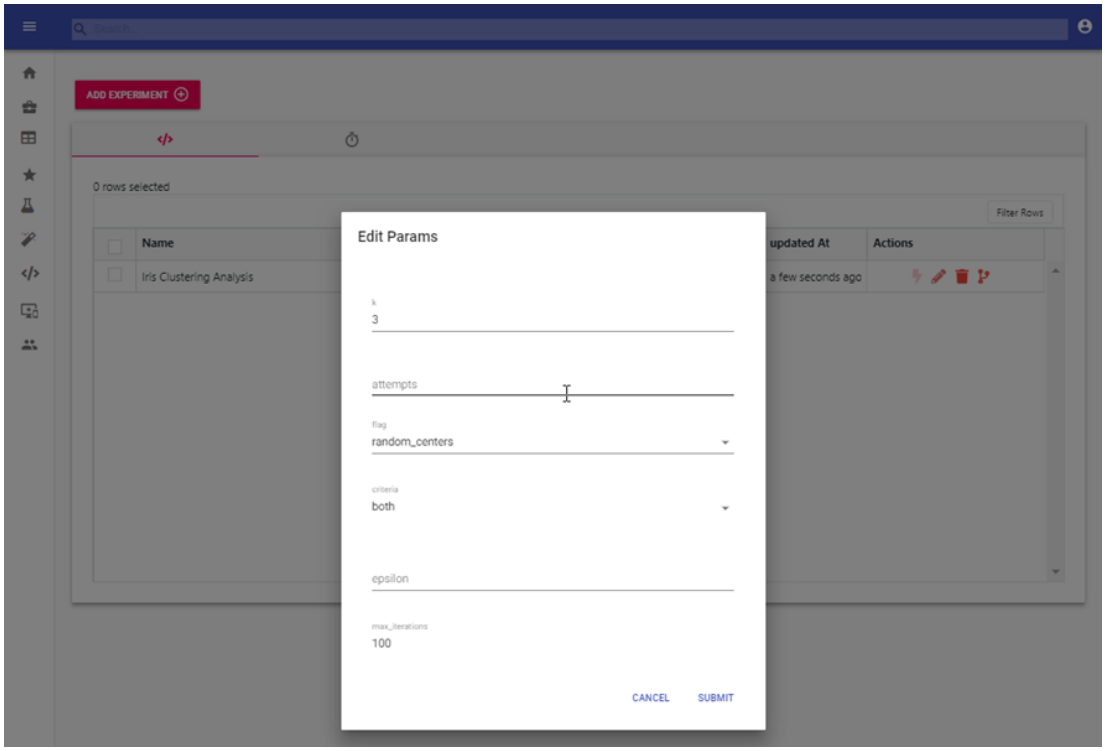


Figura 3.13: Formulario de edición de parámetros

Otra de las interfaces importantes son las de creación de creación de proyectos y la de creación de modelos. Los proyectos pueden contener uno o mas experimentos. respecto a la página de modelos, a través de ésta se pueden agregar mas algoritmos a la plataforma.

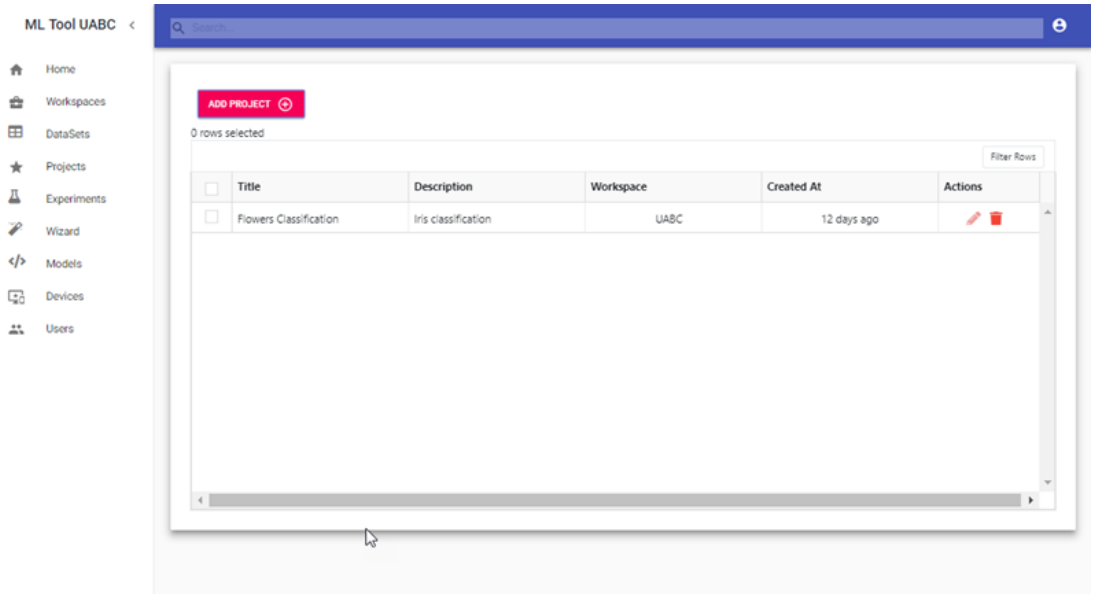


Figura 3.14: Página de proyectos

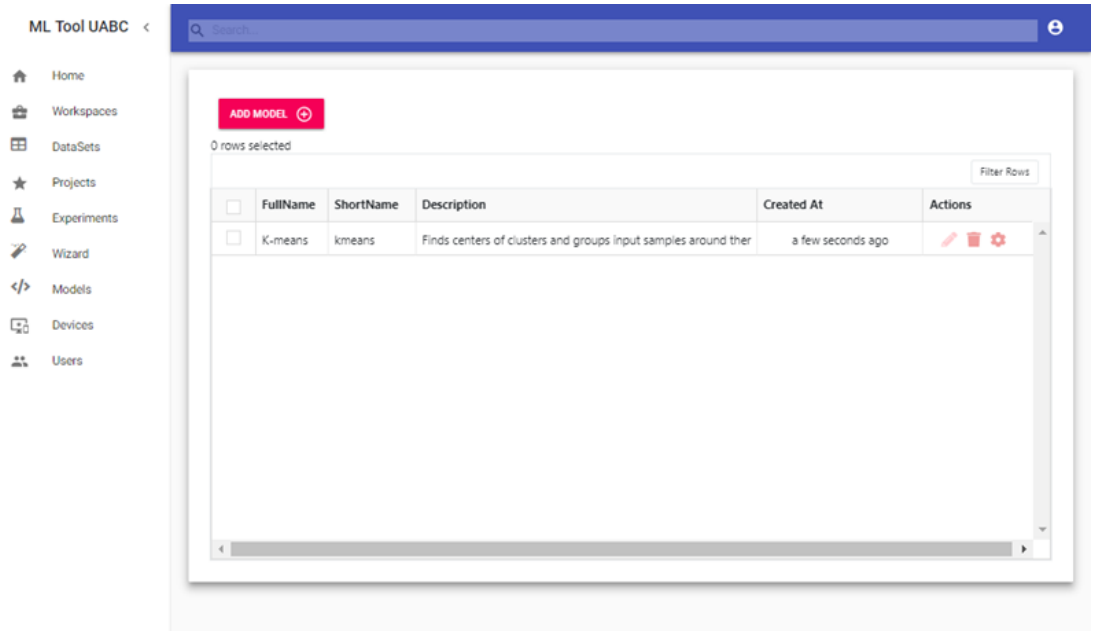


Figura 3.15: Página de modelos implementados

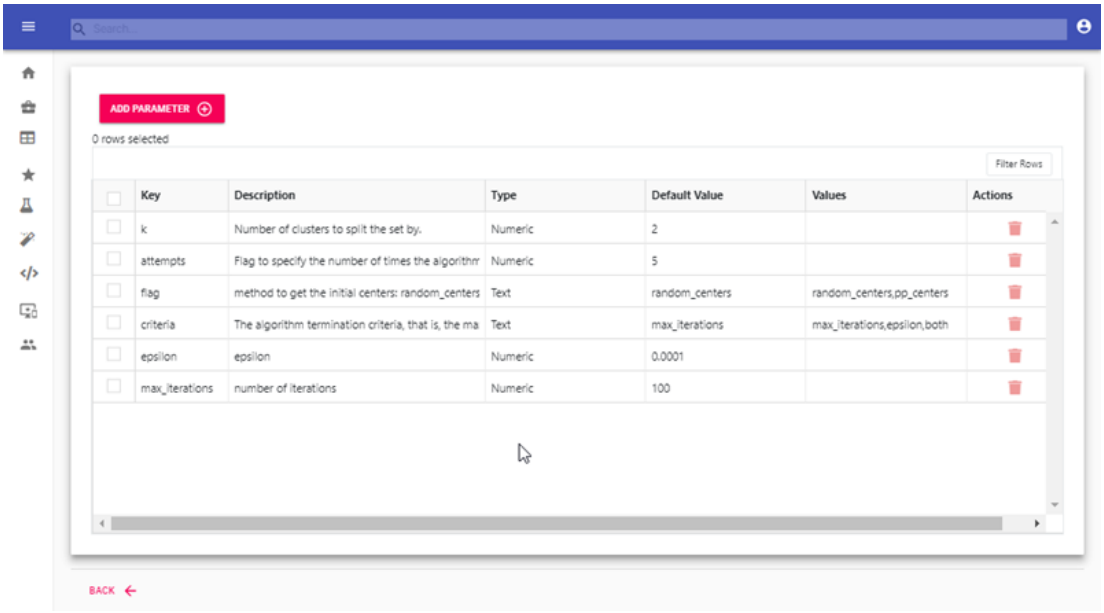


Figura 3.16: Formulario de edición de parámetros de un modelo

3.2.2. Fase 2: Diseño

Diseños simples: La metodología X.P sugiere usar diagramas simples y sencillos para representar el software. Hay que procurar hacerlo todo lo menos complicado posible para conseguir un diseño fácilmente entendible e implementable que a la larga costará menos tiempo y esfuerzo desarrollar. Los diagramas que se presentarán a continuación describe como esta organizado el software y su funcionamiento.

Diagrama Despliegue de componentes de la plataforma

En la figura 3.17 se muestra el despliegue de cada uno de los componentes del sistema y cómo es el intercambio de mensajes e interacción entre estos. Tal como se puede apreciar en dicho gráfico lo que hace novedoso al modelo de arquitectura planteado es que permite que la solución en el tiempo pueda ser escalada tanto vertical como horizontalmente, usando CUDA y con el aprovisionamiento de un sistema de encolamiento basado en workers respectivamente.

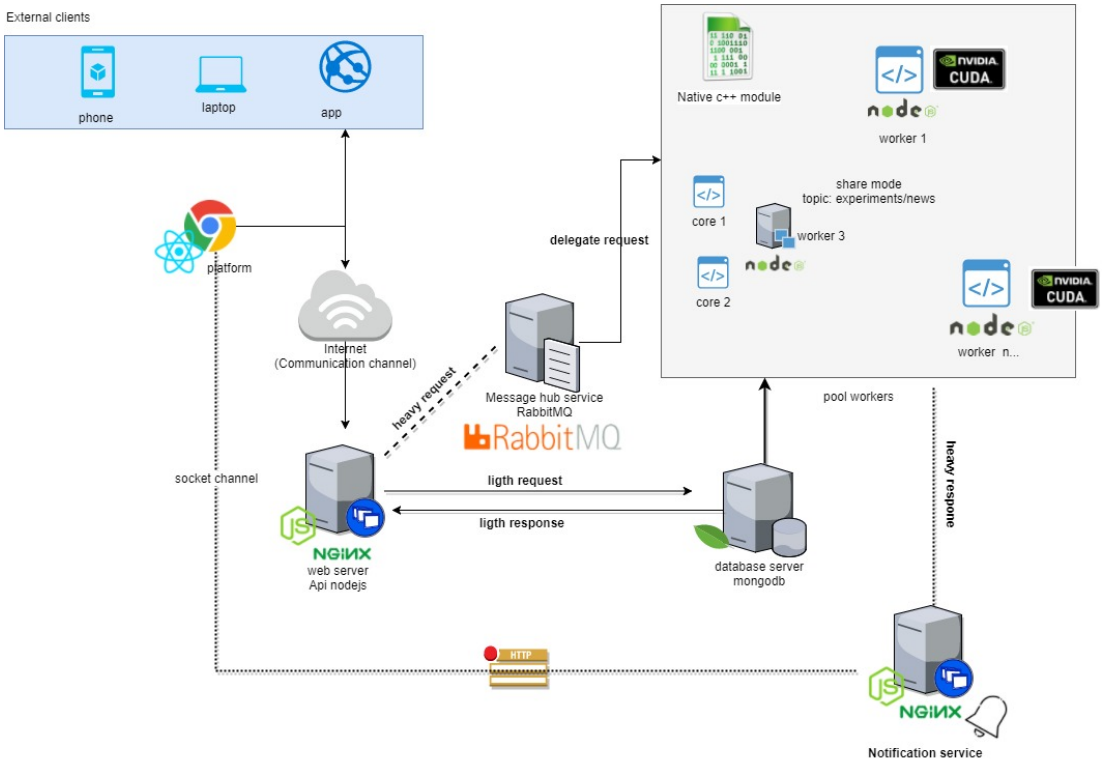


Figura 3.17: Despliegue de componentes

1. **Escalamiento Horizontal**

Sistema de encolamiento de tareas (RabbitMQ)

RabbitMQ es un software de encolamiento de peticiones/mensajes/tareas, de código abierto escrito en Erlang(Lenguaje de programación utilizado para construir sistemas en tiempo real, escalables y de alta disponibilidad) (Erlang, 2018),que implementa el protocolo de comunicación AMQP (Advances Message Queuing Protocol) y utiliza el framework Open Telecom Platform(OTP) para soportar su modelo de ejecución de forma distribuida. Se utiliza a menudo en sistema de alta recurrencia como clientes de mensajería, aplicaciones de internet of things y sistemas que requieren de mucho procesamiento en donde la respuesta al cliente no es inmediata.

La forma en como interactúan la aplicación y RabbitMQ, se describe a continuación:

a) Al iniciar la aplicación, se crea la cola de mensajes dentro de RabbitMQ en donde

irán quedando almacenados de forma cronológica todos los mensajes(peticiones del usuario al sistema) que vayan llegando.

b) Mientras tanto en background, los workers (los cuales pueden ser n procesos independiente, corriendo de forma distribuida) cada cierto tiempo validan si existe un nuevo mensaje en espera en la cola, lo recupera, obtiene los datos del mensaje y ejecutan la tarea que le corresponde.

c) Finalmente el worker actualiza el estado del mensaje y genera la respuesta. El tiempo de respuesta al usuario dependerá de cuanto se demore el worker en atender la solicitud

Un mensaje puede incluir cualquier tipo de información, por ejemplo el id del cliente que genero la petición, el id del modelo que el usuario quiere ejecutar/entrenar, la fecha y hora en que se hizo la petición etc.

2. **Escalamiento vertical**

Procesamiento en paralelo usando CUDA

Los algoritmos de Machine Learning implementados en la plataforma están codificado en C++, algunos usando OpenCV. Dicha librería escrita en c y c++, trae implementados alrededor de 500 algoritmos de visión artificial, ML, reconocimiento de patrones, *clustering*, robótica etc., lo que facilita enormemente el desarrollo de aplicaciones en tiempo real, tales como sistemas de seguridad, detección facial, reconocimiento e inspección de objetos, navegación de robots etc. Originalmente desarrollada por Intel, actualmente OpenCV cuenta con una comunidad de casi 47000 personas alrededor del mundo y más de 7 millones de descargas, puede ser usada en Python, java, c, c++, os, Android e incluso Matlab. Una de sus grandes bondades, que sin duda vale la pena resaltar, es su integración con CUDA, dicha característica resulta realmente fascinante puesto que se puede tomar provecho del cómputo en paralelo para optimizar las implementaciones.

El módulo de GPU, contiene un conjunto de clases, funciones, primitivas de bajo nivel y algoritmos que hacen que no se requiera tener experiencia en CUDA para usar la tarjeta gráfica de la máquina como unidad extra de procesamiento. En el ejemplo de código a continuación se muestra a grandes rasgos como se usa el API.

```
#include <iostream>
#include "opencv2/opencv.hpp"
#include "opencv2/gpu/gpu.hpp"
int main (int argc, char* argv[])
{
    try
    {
        cv::Mat src_host;
        src_host= cv::imread("file.png", CV_LOAD_IMAGE_GRAYSCALE);
        cv::gpu::GpuMat dst, src;
        //cargar imagen en la GPU
        src.upload(src_host);
        //procesar imagen en la GPU
        cv::gpu::threshold(src, dst, 128.0, 255.0, CV_THRESH_BINARY);
        cv::Mat result_host;
        //descargar imagen procesada a memoria
        dst.download(result_host);
        //visualizar resultado
        cv::imshow("Result", result_host);
        cv::waitKey();
    }
    catch(const cv::Exception& ex)
    {
        std::cout << "Error: " << ex.what() << std::endl;
    }
    return 0;
}
```

Listing 3.1: Ejemplo C++

Diagrama de componentes

Un diagrama de componentes representa cómo un sistema de software está dividido y sus dependencias. Los componentes pueden ser archivos, cabeceras, bibliotecas compartidas, módulos, ejecutables, o paquetes. A pesar de que este tipo de artefactos prevalecen en el campo de la arquitectura de software, pueden ser usados para modelar y documentar cualquier arquitectura de sistema. En la figura 3.18, se pueden observar cada uno de los componentes del sistema en contexto, la manera en como están distribuidos y sus relaciones.

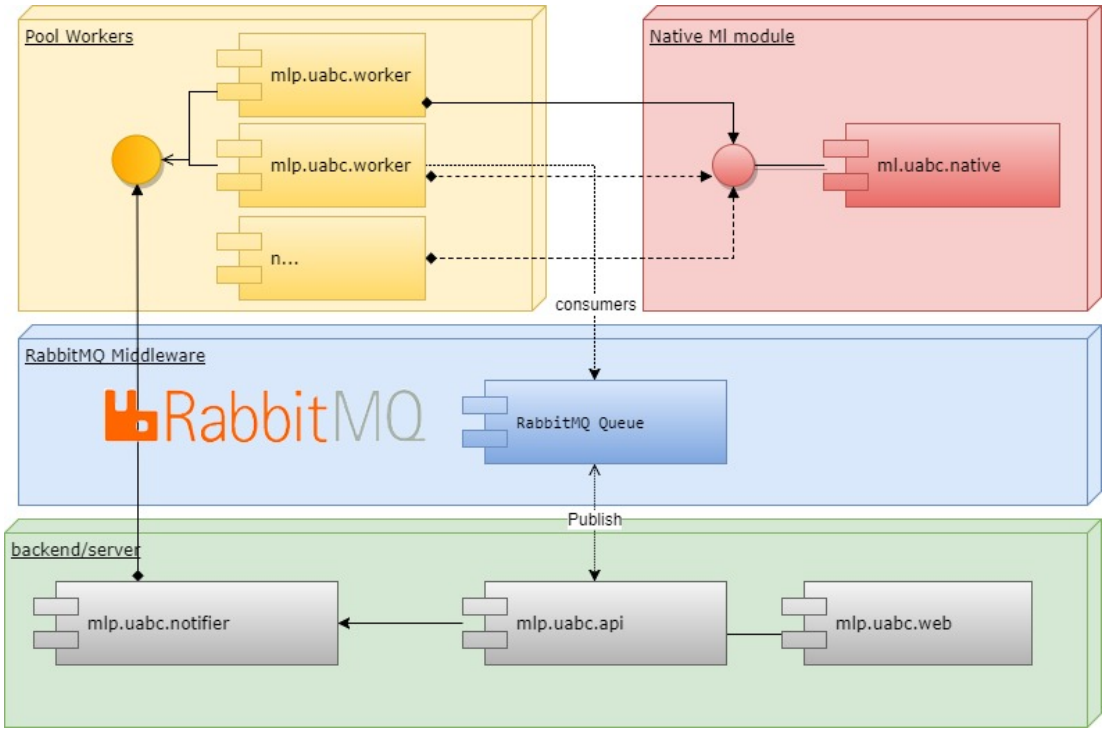


Figura 3.18: Diagrama de componentes del software

Diagrama Arquitectónico

Para el desarrollo de este software se evaluaron diferentes tecnologías y plataformas de desarrollo, al inicio se hicieron pruebas de concepto usando lenguajes de programación de alto nivel como java, Python ,C#, por las bondades que estos presentan al codificar, sin embargo por temas de rendimiento y compatibilidad con las librerías nativas de OpenCV y Cuda se optó por usar como lenguaje de programación C Y C++ para la implementación de los algoritmos

de aprendizaje de máquina, Nodejs en los demás módulos(Api, Workers etc.) y Napi para crear el puente entre ambos lenguajes (C/C++ y JavaScript). A continuación se destacan algunas de las características que se tuvieron en cuenta para elegir C++ como mejor opción:

1. **La rapidez:** Al no ser un lenguaje interpretado, todos los módulos de la aplicación construidos en C++ deben ser compilados para que funcionen en cada máquina en donde se planee ejecutar,dicho proceso en ocasiones es algo tedioso, pero esto representa una ventaja puesto que el programa estará totalmente adaptado a las peculiaridades del sistema donde se encuentre, lo que quiere decir que quedará optimizado. C++ y C son lenguajes de programación multi-plataforma.

2. **Eficiencia:** Característica unida a la anterior.

3. **Librerías y otras:** C y C++ son lenguajes de programación tan robustos y versátiles, que actualmente son los preferidos en la creación de software, como sistemas operativos (windows, GNU/Linux, Mac OSX, android), utilidades (Bibliotecas, servicios, herramientas de mantenimiento), compiladores, depuradores, entornos de desarrollo, sistemas de computación gráfica, visión por computadora etc. esto dado que la interacción con el hardware es casi directa(GPU) y a través de frameworks como CUDA se puede aplicar computo paralelo. El acceso a memoria puede ser administrado y controlado como se desee (depende del desarrollador) y es muy eficiente en tareas de procesamiento complejas, existe mucha documentación en internet y en otros medios sobre algoritmos de IA (inteligencia artificial), procesamiento de imágenes, redes neuronales etc. , implementados en este lenguaje. Es orientado a objetos. librerías como Opencv, Cuda, dlib, TensorFlow etc. son 100 % compatibles con c y c++ y casi todos los ejemplos y documentación vienen codificados en este lenguaje.

   Es claro que todos los computadores ejecutan cualquier programa, a través de una o más unidades centrales de procesamiento (CPU), las cuales solo son capaces de interpretar lo que se conoce como lenguaje o código máquina, el cual consiste en una serie de instrucciones u operaciones elementales de muy bajo nivel, tales como escribir en memoria, sumar un par de números, o leer bytes de diferentes fuentes, por lo tanto, todos los lenguajes de programación en algún momento deben de ser “traducidos” a “lenguaje máquina”, para que los programas puedan ser ejecutados. A este proceso se le suele denominar compilación, y tanto el C como C++, siguen este esquema de ser “compilados” al “lenguaje

máquina" según el procesador sobre el cual se esté ejecutando la aplicación, no obstante este proceso ocurre casi de forma directa, ya que no requiere de que antes las instrucciones sean traducidas a un lenguaje intermedio como es el caso de java o .net. el lenguaje C por ejemplo tiene estructura de datos tan simples que en ocasiones el desarrollador está muy cerca de escribir un lenguaje parecido al que la CPU comprende. En muchos casos, esta simplicidad consigue que el programa tenga un excelente rendimiento dada la simplicidad del lenguaje máquina producida.

4. **El rendimiento:** Al igual que C/C++, java puede ser ejecutado en cualquier clase de dispositivo o CPU independientemente del sistema operativo o arquitectura, no obstante, una de las grandes diferencias con respecto a C, está en que al ser compilado, el código no es traducido directamente a lenguaje máquina, sino a un "pseudo lenguaje máquina"denominado "byte code". Este Java compilado en "byte code"puede ser transportado a cualquier computador sobre el que se esté ejecutando la máquina virtual de java(JVM) que funciona como "Java Runtime Environmet". Es por esto que el rendimiento de una aplicación C no será el mismo si se desarrollase en Java u otro lenguaje de alto nivel.

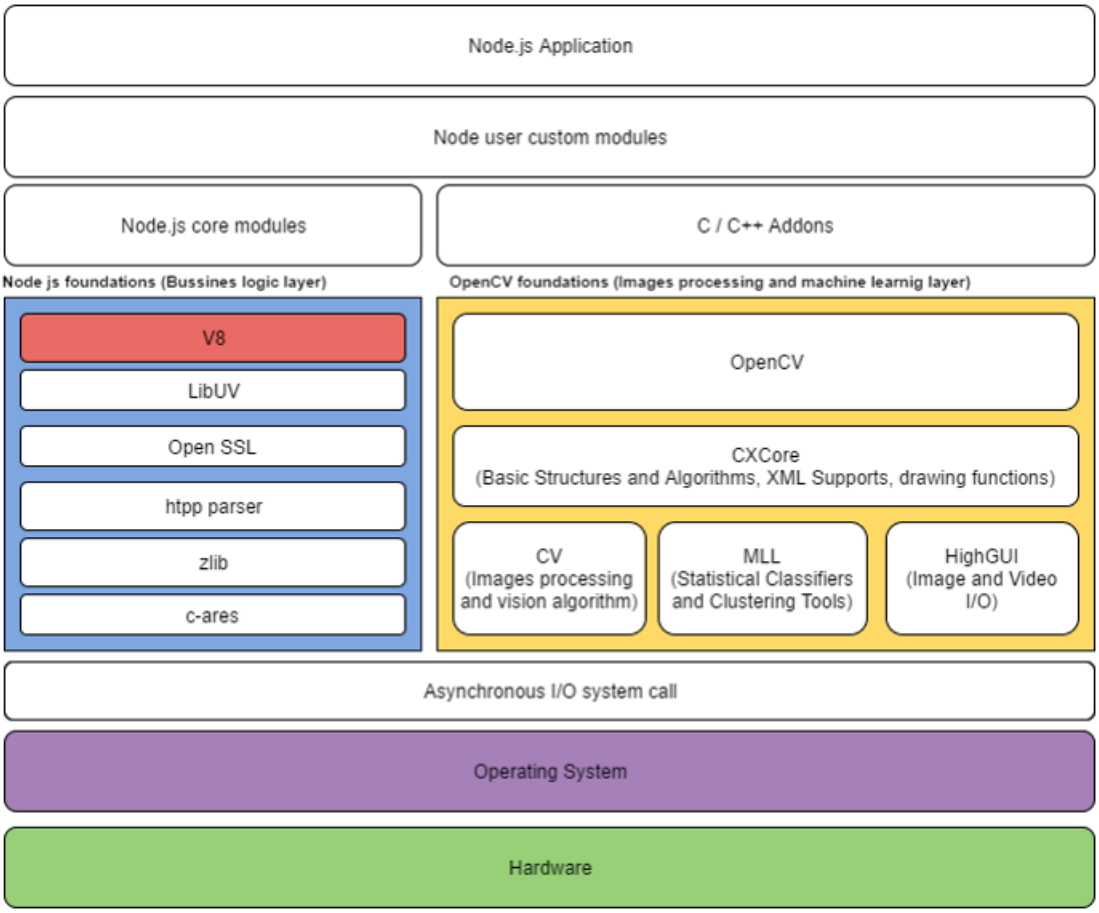


Figura 3.19: Diagrama arquitectónico

3.2.3. Fase 3: Codificación

En la metodología X.P., el cliente es un integrante más del equipo de desarrollo; su presencia es indispensable en las distintas fases. A la hora de codificar una historia de usuario su presencia es aún más necesaria, puesto que son los clientes quienes crean las historias de usuario y negocian los tiempos en los que serán implementadas. Antes del desarrollo de cada historia de usuario el cliente debe especificar detalladamente lo que ésta hará y también tendrá que estar presente cuando se realicen los test que verifiquen que la historia está debidamente implementada. La codificación debe hacerse ateniendo a estándares de codificación ya creados. Programar bajo estándares mantiene el código consistente y facilita su comprensión y escalabilidad. Todos los módulos que componen el software se describen a continuación.

mlp.uabc.web

Aplicación Web desarrollada en React.js, que sirve como interfaz para que el usuario pueda interactuar con el sistema. Cualquier petición realizada desde el front-end será procesada en el servido por el API, otro componente fundamental que se describirá luego en esta misma sección. Este módulo es uno de los más importantes, puesto que es el único medio que el usuario final tiene para poder acceder a la plataforma, cargar sus datos al sistema, ejecutar los modelos, visualizar la información etc.

mlp.uabc.api

API REST implementada en Node.js, que se encarga de atender y procesar todas las solicitudes entrantes al sistema, enviadas desde el front-end. funciones como el control de acceso a la plataforma, la gestión de los modelos, creación de espacios de trabajo etc. son algunas de las funciones que se encuentras implementadas. Para ilustrar lo anterior a través de un ejemplo y entender un poco más como es el ciclo de vida de una petición, supongamos que un usuario se dispone a ingresar a la plataforma. Este entonces ingresa sus credenciales en el formulario de login y hace click sobre el botón iniciar sesión, los datos viajan vía http al servidor y allí el API valida sobre la base de datos, usando el usuario y la contraseña, si dicho usuario existe, finalmente el sistema, habilita o deniega el acceso a la aplicación, lo mismo pasa con cualquier función que el usuario invoque desde a interfaz, para cada una existe un método en el API que se encarga de atender dicha solicitud.

```
@config({
    auth: false,
    cors: true,
    tags: ['api',"users"],
    notes: 'Generate a new Token',
    description: "use this method to authenticate with the platform",
    validate:{
        payload:{
          userName : joi.string().max(30).required().description("Username"),
          password : joi.string().max(40).required().description("the password")
      }
```

```
12      }
13    })
14    @post("/auth")
15    autenticateHandler(request: hapi.Request, reply: hapi.IReply) {
16    #DB Model: User
17        User.findOne(request.payload)
18            .then((usr: IUser) => {
19                return reply(getToken(usr));
20            }).catch((error: any) => {
21                return reply( Boom.unauthorized());
22          });
23    }
```

Listing 3.2: Ejemplo del Método Autenticar

mlp.uabc.notifier

El tiempo de ejecución que el API ocupa para atender y procesar una solicitud en el servidor depende de la acción que el usuario ejecute, este tiempo es directamente proporcional al tiempo de respuesta al usuario. Por ejemplo si un usuario trata de ingresar al sistema, la validación de si está registrado, es procesada por el API en cuestión de segundos, por ende el usuario sabrá rápidamente si puede o no acceder a la plataforma, Pero que pasa cuando un usuario está tratando de entrenar un modelo o de aplicar algún algoritmo a uno de sus conjuntos de datos cargados de 500000 registros. En estos casos lo que hace el API, es encolar la petición en MQRabbit(servicio de encolamiento de tareas), luego uno de los workers se encarga de procesar la solicitud y posteriormente informarle al usuario una vez termine. Para esto es importante el servicio de notificaciones, puesto que es a través de dicho servicio que el worker le hace llegar al usuario el mensaje de que su tarea ha sido procesada satisfactoriamente.

```
1
2  .....
3   case "kmeans":
4      #obtiene el vector de features
```

```
    var features = math.matrix(data).subset(...)
    var k = parameters.k;
    #Ejecuta el algoritmo
    var { centroids, Y,X } = await kmeans(features, k);
    var end = new Date();
    var elapsedTime = evalElapsedTime(start, end);
    console.log(elapsedTime);
    #Actualiza el experimento
    var experiment = await Experiment.findByIdAndUpdate(_id, {
      $set: {
        status: "completed",
        output: {
          type: "matrix",
          duration: elapsedTime.elapsedTimeAsMs,
          result: {X, Y, centroids }
        }
      }
    }, { new: true });

    #invoka al servicio de notificaciones para informar al usuario
    await axios.post(`${config.notifier.uri}/to`,{
     clientId: clientId,
     message: {
        body: `${experiment.title} completed`,
        info: elapsedTime.elapsedTimeAsStr
     }
    });

break;
.........
```

Listing 3.3: Ejemplo del algoritmo K-means implementado en el Worker

```
1
2  const app = require('express')();
3  const http = require('http').Server(app);
4  const io = require('socket.io')(http);
5  const moment = require("moment");
6  const bodyParser = require('body-parser');
7  var multiparty = require('multiparty');
8  const PORT=9000;
9  #create application/json parser
10 var jsonParser = bodyParser.json()
11 #create application/x-www-form-urlencoded parser
12 var urlencodedParser = bodyParser.urlencoded({ extended: false })
13
14
15 app.get('/', function(req, res){  res.send("the service is running");
16
17 app.post('/to', urlencodedParser, jsonParser, (req, res)=>{
18   var client = io.sockets.connected[req.body.clientId];
19   client.emit('client:notify:me', req.body.message);
20   console.log('Result broadcast to client ', JSON.stringify(req.body));
21   res.type("text/plain").sendStatus(200);
22 });
23
24 #mensaje para un usuario en particular
25 app.post('/all', urlencodedParser, jsonParser, (req, res)=>{
26   var clients = io.sockets.emit('client:notify:all', req.body.message);
27   console.log('Result broadcast to all clients ');
28   res.type("text/plain").sendStatus(200);
29 });
30
31 #mensaje para todos
```

```
32 io.on('connection',(socket)=>{
33   console.log("a user connected");
34   console.log(`${moment().format("dddd, MMMM Do YYYY, h:mm:ss a")}`);
35   socket.emit("server:init", socket.id);
36 });
37
38 http.listen(PORT, ()=>{  console.log('listening on *:'+PORT); });#running server
39
40 ........
```

Listing 3.4: Servicio de notificaciones

RabbitMQ Middleware

RabbitMQ es un servicio de encolamiento y distribución de mensajes, que funciona como middleware (Johansson, 2015). Permite que las peticiones que requieren de un tiempo de computo considerable para su procesamiento vayan quedando encoladas mientras el servidor tiene disponibilidad para atenderlas, de esta forma se evita que se pierdan en el proceso. Los workers pueden en todo momento acceder a la pila de mensajes y obtener toda la información de cada uno, como por ejemplo los datos del cliente que genero la solicitud, argumentos importantes para la ejecución de la tarea etc. Para ir atendiendo las peticiones pendientes por procesar y ir liberando espacio en la cola.

mlp.uabc.worker

Un Worker es un servicio que se ejecuta en background, encargado de procesar las peticiones que vayan quedando encoladas en MQRabbit. Puede haber n cantidad de workers en ejecución distribuidos en diferentes máquinas, o sobre una sola, como procesos independientes. Para entender un poco mejor como funciona, supongamos que un usuario hace clic sobre la opción entrenar modelo, la orden es enviada al servidor y recibida por el API REST, luego dicha petición es encolada y queda a la espera de que sea procesada, Posterior a esto los workers entra en acción, si alguno está libre, el servicio recupera la información del mensaje en cola y

ejecuta la acción correspondiente, una vez termine de realizar la tarea, a través del servicio de notificaciones este le envía al usuario la confirmación de que su modelo ha sido entrenado.

```
.........
case "knn":

      var features = math.matrix(data).subset(.....);
      var Y = responseIdx ? math.squeeze(math.matrix(data).subset(....);
      var kfolds = parseInt(parameters.kfolds);
      var k = parseInt(parameters.k);
      console.log(k,kfolds);
      var matrix = math.concat(features,math.transpose([Y])).valueOf();
      var result = await validateKnn(matrix,kfolds,k, true);
      console.log(result.accuracy);
      var end = new Date();
      var elapsedTime = evalElapsedTime(start, end);
        var experiment = await Experiment.findByIdAndUpdate(_id, {
            $set: {
              status: "completed",
              output: {
                type: "matrix",
                duration: elapsedTime.elapsedTimeAsMs,
                result: result
              }
            }
          }, { new: true });


        #Notificacio'n por email
        await sendEmailNotification(....);
        #Notificacio'n por Sockets (Servicio de Notificacio'n)
        await axios.post(`${config.notifier.uri}/to`,..);
```

```
30      break;
31  ....
```

Listing 3.5: Algoritmo knn (worker)

ml.uabc.native

Una de las bondades de desarrollar aplicaciones con node.js, está en que se pueden construir aplicaciones híbridas, es decir codificar parte de la aplicación en C/C++ y parte en Javascript (Gobbo, 2017) NAPI es un API estándar que permite codificar módulos nativos para node y compilarlos para que funcionen sobre múltiples plataformas. Esto quiere decir que sin importar la versión de Nodejs, el sistema operativo, ni sobre que motor de JavaScript se esté ejecutando (v8 motor de JavaScript de Crhome o Chakra motor de JavaScript de Microsoft Edge), no es necesario volver a compilar el módulo para que este funcione. En esta aplicación todos los algoritmos de machine learning fueron escritos en C/C++ con el fin de facilitar en el futuro la integración con CUDA y mejorar el rendimiento (opencv, 2015). A continuación a través de un ejemplo se muestra cómo es posible lograr esta interoperabilidad 3.20.

napi.png

Figura 3.20: Módulos nativos, Node.js

```
#include <nan.h>
using namespace v8;

#definimos nuestra funcio'n (suma=>nombre)
NAN_METHOD(suma){
  #validamos los parametros de entrada que le enviaremos desde js
  if(!info[0]->IsNumber() && !info[2]->IsNumber() ){
```

```
8     #si los argumentos son invalidos arrojamos una exception
9     Nan::ThrowError("argumentos invalidos");
10    return;
11  }
12  #obtenemos los valores
13  double a = info[0]->NumberValue();
14  double b = info[1]->NumberValue();
15
16  #C++ part
17  Local<Number> result = Nan::New(a + b);#calculamos la suma
18  info.GetReturnValue().Set(result);#finalmente retornamos el resultado
19 }
20
21 NAN_MODULE_INIT(Init) {
22  #exportamos la funcio'n suma para que pueda ser
23  #llamada desde Node en javascript
24  Nan::Set(target, Nan::New("suma").ToLocalChecked(),
25           Nan::GetFunction(Nan::New<FunctionTemplate>(suma)).ToLocalChecked());
26 }
27
28 NODE_MODULE(mymodule, Init)
29
30 ......
```

Listing 3.6: Ejemplo modulo nativo para node escrito en C++

A continuación el código Javascript donde se invoca el módulo nativo. Se puede hacer de dos formas al hacer el import, usando el paquete bindings o especificando la ruta completa de donde quedo el archivo .node generado después de la compilación.

```
1
2 #forma 1
3 const mymodule = require("bindings")("mymodule");
```

```
#forma 2
#const mymodule = require("./build/Release/mymodule");

const sum = mymodule.suma(2,2);

console.log("El resultado de la suma es :", sum);//mostramos el resultado
....
```

Listing 3.7: Archivo Javascript(Node.js) que usa el módulo nativo

Capítulo 4

Conclusiones

4.1. Conclusiones

De la presente investigación, en base a los objetivos específicos se puede concluir:

Que en escenarios en donde la estructura de los datos que se quieren almacenar no es homogénea, como por ejemplo un experimento, el cual varía según el modelo, los parámetros y los datos, la mejor opción es usar sistemas de almacenamiento dinámico y flexible, como los no sql, con el desarrollo de este software se pudo corroborar que en este tipo de escenarios mongodb se adapta perfectamente.

Otro descubrimiento significativo, en términos de arquitectura, a la luz de los resultados obtenidos durante el desarrollo de la plataforma mltool UABC, ésta en que los sistemas de encolamiento de tareas como RabbitMQ o Kafka, debidamente configurados, facilitan el escalamiento horizontal de cualquier producto de software, este tipo de sistemas te da la versatilidad de poder ejecutar de forma paralela, nodos autónomos de procesamiento,bien sea en máquinass independientes o diferentes procesos, lo que representa un incremento significativo en el rendimiento de la aplicación, puesto que tener varios nodos que ejecuten la misma tarea mantienen un balance en el uso de recurso de hardware.

Durante la fase de análisis y definición de tecnologías se evaluaron diferentes lenguajes de programación tales como: python con django, Ruby con rails, javascript con node.js etc. No obstante fue este ultimo el que finalmente fue seleccionado y una de las principales razones esta en la versatilidad que tiene de permitir que se puedan construir módulos en c++ para ser invocados desde javascript, lo que representa una ventaja significativa respecto a los demás puesto que la integración es casi nativa y se pudieron implementar todos los algoritmos de aprendizaje de máquina en C++.

Otro punto importante que vale la pena destacar, esta en el modulo de visualización de la aplicación, en la construcción del software se identificó que mientras el número de registros del conjunto de datos no exceda los 1000 puntos, no habrán problemas con la visualización en el navegador, no obstante, cuando la cantidad de instancias es mayor, el navegador se puede quedar pegado tratando de dibujar los puntos. En estos casos, la solución consistió en aplicar PCA para reducir la dimensión del conjunto de datos de n características a 2 o 3 (cantidad máxima de componentes que se pueden visualizar), luego sobre el conjunto de datos resultante se aplica el algoritmo de convexhull y a través de este se obtiene el conjunto de puntos que conforman la frontera del cluster de datos, finalmente este conjunto de coordenadas es la que se muestra en gráfica.

4.2. Trabajo Futuro

Como trabajo futuro se plantea:

1. Extender las lista de algoritmos implementados en la plataforma.
2. Desarrollar un modulo móvil que sirva como herramienta de captura de datos y permita interactuar con los experimentos desde el dispositivo
3. Incluir y mejorar el modulo de visualización, agregando mas gráficas.
4. Construir un modulo de mensajerías mas sofisticado, que permita que se pueda llevar el registro de las notificaciones que un usuario ha recibido.
5. Agregar la funcionalidad de permitir que varios usuarios puedan trabajar sobre el mismo experimento en tiempo real.

Anexos A

Código

A continuación se presenta la implementación de los algoritmos y clases más importantes en la plataforma

A.1. Mongodb Models

```
1
2 import * as mongoose from "mongoose";
3
4 export interface IUser extends mongoose.Document {
5     userName: string;
6     password: string;
7     fullName: string;
8     email: string;
9     job?: string;
10    bio?: string;
11    isAdmin: boolean;
12 }
13
14 export interface IWorkSpace extends mongoose.Document {
15    title: string;
16    description?: string;
17    owner: IUser;
18 }
```

```
19
20 export interface IComponent extends mongoose.Document {
21     name: string,
22     description: string,
23     enabled?: boolean,
24     statistics?: any,
25     categories?: any,
26     required?: boolean,
27     dataType: { name: string,primitive: string}
28 }
29
30 export interface IDataSet extends mongoose.Document {
31   title: string;
32   description?: string;
33   hasHeader: boolean,
34   dimension?:number,
35   data?: any[];
36   components?: Array<IComponent>
37   workspace: IWorkSpace;
38 };
39
40 export interface IModelParameter extends mongoose.Document{
41     key : { type: String, unique: true, required: true},
42     defaultValue: mongoose.Schema.Types.Mixed,
43     description: String,
44     values: [mongoose.Schema.Types.Mixed]
45 }
46
47
48 export interface IModel extends mongoose.Document {
49     fullName: string,
50     shortName: string,
51     category: string,
52     subCategory: string,
53     description: string,
54     parameters: Array<IModelParameter>
55 };
```

```
56
57  export interface IExperiment extends mongoose.Document {
58      title: string;
59      description?: string;
60      hypotheses?: string;
61      keywords?: string;
62      inputs?: string;
63      status: string;
64      error?:string;
65      datasetId: IDataSet,
66      featuresIdx: number[],
67      responseIdx: number,
68      modelId: IModel,
69      parameters?: any,
70      output: { type: string,duration: Date,result: any };
71  };
72
73  export const UserSchema = new mongoose.Schema({
74      userName: { type: String, unique: true },
75      password: { type: String, require: true, minlength: 7 },
76      fullName: String,
77      email: { type: String, trim: true, lowercase: true, unique: true, required: true },
78      job: { type: String, required: true },
79      bio: { type: String, trim: true, lowercase: true },
80      isAdmin: { type: Boolean, default: false }
81  }, { timestamps: { createdAt: 'created_at', updateAt: "update_at" } });
82
83  export const WorkSpaceSchema = new mongoose.Schema({
84      title: { type: String, maxlength: 30, required: true },
85      description: { type: String, maxlength: 200, required: true },
86      owner: { type: mongoose.Schema.Types.ObjectId, ref: "User", required: true }
87  }, { timestamps: { createdAt: 'created_at', updateAt: "update_at" } })
88
89  export const ComponentScheme = new mongoose.Schema({
90      name: String,
91      enabled: { type: Boolean, default: true },
92      description: {type: String,default: ""},
```

```
93      required: { type: Boolean, default: false},
94      categories: mongoose.Schema.Types.Mixed,
95      dataType: mongoose.Schema.Types.Mixed
96  })
97
98  export const DataSetSchema = new mongoose.Schema({
99      title: { type: String, maxlength: 30, required: true, unique: true },
100     description: { type: String, required: true },
101     data:{ type: [], required: true},
102     hasHeader: { type: Boolean, default: false },
103     components: [ComponentScheme],
104     workspace: { type: mongoose.Schema.Types.ObjectId, ref: "WorkSpace",
105     required: true },
106 }, { timestamps: { createdAt: 'created_at', updateAt: "update_at" } });
107
108 export const ModelParameterScheme = new mongoose.Schema({
109     key : { type: String, required: true},
110     defaultValue: mongoose.Schema.Types.Mixed,
111     description: String,
112     values: [mongoose.Schema.Types.Mixed]
113 })
114
115 export const ModelScheme = new mongoose.Schema({
116     fullName: { type: String, required: true},
117     shortName: { type: String, required: true, trim: true},
118     description: String,
119     parameters: [ModelParameterScheme],
120     category: { type: String, required: true, enum: [
121     "Supervised Learning",
122     "Unsupervised Learning"
123     ]},
124     subCategory: { type: String, required: true, enum: [
125     "Clustering",
126     "Dimensionality reduction", "Regression",
127     "Classification" ]}
128 })
129
```

```
export const ExperimentScheme = new mongoose.Schema({
    title: { type: String, maxlength: 30, required: true },
    description: { type: String, maxlength: 500, required: true },
    hypotheses: { type: String, maxlength: 500},
    //design: { type: String, maxlength: 200},
    keywords: { type: String, maxlength: 500 },
    inputs: { type: String, maxlength: 500 },
    status: { type: String, enum: [
    "created",
    "running",
    "error",
    "completed"], default: "created" },
    error: { type: String},
    parameters: mongoose.Schema.Types.Mixed,
    output: {
    type: { type: String },
    duration: { type: Number },
    result: mongoose.Schema.Types.Mixed },
    featuresIdx: [Number],
    responseIdx: Number,
    datasetId: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'DataSet',
    required: true },
    modelId: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'Model',
    required: true }
}, { timestamps: { createdAt: 'created_at', updateAt: "update_at" } });



export const User = mongoose.model<IUser>('User', UserSchema);
export const WorkSpace = mongoose.model<IWorkSpace>('WorkSpace', WorkSpaceSchema);
export const DataSet = mongoose.model<IDataSet>('DataSet', DataSetSchema);
export const Experiment = mongoose.model<IExperiment>('Experiment', ExperimentScheme);
```

```
166 export const Model = mongoose.model<IModel>('Model', ModelScheme);
```

Listing A.1: models.ts

A.2. User Controller

```
1
2  import * as hapi from 'hapi'
3  import * as Boom from "boom";
4  import * as joi from "joi";
5  import { IUser, User } from "./../../models";
6  import getToken from "./Token";
7  import {
8  get,
9  controller,
10 post,
11 put,
12 route,
13 config,
14 Controller } from 'hapi-decorators'
15
16
17 @controller("/api/v1/users")
18 export default class UserController implements Controller {
19
20  baseUrl: string; routes: () => hapi.IRouteConfiguration[];
21
22   @config({
23     auth: false,
24     cors: true,
25     tags: ["api","users" ],
26     notes: "return the list of the users",
27     description: "use this method to get the users list",
28   })
29   @get("/")
30   getHandler(request: hapi.Request, reply: hapi.IReply) {
31       User.find()
32           .then((data)=>{
```

```
33              return reply(data)
34            })
35            .catch((err: Error)=>{
36              console.log(`${err.message}`)
37              return reply(Boom.badImplementation(err.message))
38            })
39   }
40  @config({
41     auth: false,
42     cors: true,
43     tags: ["api","users" ],
44     notes: "return the user found",
45     description: "use this method to find users by email",
46   })
47   @get("/findByEmail/{email}")
48  findUserByEamilHandler(request: hapi.Request, reply: hapi.IReply) {
49      User.findOne({ email: request.params.email })
50          .then((usr: IUser) => {
51                return reply(usr ? false: true)
52            })
53            .catch((err: Error)=>{
54              return reply(Boom.badImplementation(err.message))
55            })
56
57   }
58   @config({
59     auth: false,
60     cors: true,
61     tags: ['api',"users"],
62     notes: 'Returns the user that has been created',
63     description: "use this method to create a new user into the system",
64     plugins: {
65         'hapi-swagger': {
66             payloadType: 'form'
67         }
68     },
69     validate:{
```

```
70      payload:{
71        userName : joi.string().max(30).required().description("Username"),
72        password : joi.string().max(40).required().description("Password"),
73        fullName : joi.string().max(40).required().description("Name user"),
74        email :    joi.string().email().required().description("email address"),
75        job:       joi.string().max(40).allow(null).description("ocupation"),
76        bio:       joi.string().max(500).allow(null).description("bio"),
77        isAdmin:   joi.boolean().default(false).allow(null).description("admin")
78      }
79    }
80  })
81  @post("/")
82  posthandler(request: hapi.Request, reply: hapi.IReply) {
83      new User(request.payload).save()
84          .then((usr: IUser) => {
85        return reply(usr);
86      })
87      .catch((err) => {
88        return reply(Boom.badData(err.message))
89      })
90  }
91  @config({
92    auth: false,
93    cors: true,
94    tags: ['api',"users"],
95    notes: 'Generate a new Token',
96    description: "use this method to authenticate with the platform",
97    plugins: {
98        'hapi-swagger': {
99            payloadType: 'form'
100       }
101   },
102   validate:{
103       payload:{
104         userName : joi.string().max(30).required().description("Username"),
105       password : joi.string().max(40).required().description("Password"),
106     }
```

```
107     }
108   })
109   @post("/auth")
110   autenticateHandler(request: hapi.Request, reply: hapi.IReply) {
111       User.findOne(request.payload)
112           .then((usr: IUser) => {
113               return reply(getToken(usr));
114           }).catch((error: any) => {
115               return reply( Boom.unauthorized());
116        });
117   }
118   @config({
119     auth: 'jwt',
120     cors: true,
121     tags: ["api","users" ],
122     notes: "return the the token of the user",
123     description: "use this method to validate if the token is valid"
124   })
125   @get("/token")
126   getToken(request: hapi.Request, reply: hapi.IReply) {
127       reply({
128       text: 'You used a Token! ' +
129       JSON.stringify(request.auth.credentials) })
130       .header('Authorization', request.headers.authorization);
131   }
132
133
134 }
```

Listing A.2: UserController.ts

A.3. kfold

```
1 #include <vector>
2 #include <iostream>
3 #include <opencv.hpp>
4
5 using namespace std;
```

```
using namespace cv;
using namespace ml;

#pragma once
class Kfold{
public:
 Kfold(int _k, Mat& _src) :
  src(_src), k(_k) {
  if (k <= 0 || !k)
   throw runtime_error("The supplied value of K is invalid " + _k);
  //create the vector of integers
  int foldNo = 0;
  for (int i = 0; i != src.rows; i++) {
   this->whichFoldToGo.push_back(++foldNo);
   if (foldNo == this->k)
    foldNo = 0;
  }
  //Like waving a bag of ballots
  random_shuffle(whichFoldToGo.begin(), whichFoldToGo.end());


 }
 void getFold(int fold, Mat& training, Mat& testing) {
  int k = 0;
  for (int i = 0; i != src.rows; i++) {
   if (whichFoldToGo[k++] == fold) {
    testing.push_back(src.row(i).clone());
   }
   else
    training.push_back(src.row(i).clone());
  }
 }
private:
 Mat src;
 int k;
 vector<int> whichFoldToGo;
```

```
};
```

Listing A.3: Kfold.cpp

A.4. K-means código javascript

```
switch (shortName) {
          case "kmeans":
            var features = math.matrix(data)
            .subset(math.index(math.range(startIndex, data.length), featuresIdx))
            .valueOf();
            var k = parameters.k;
            var { centroids, Y,X } = await kmeans(features, k);
            var end = new Date();
            var elapsedTime = evalElapsedTime(start, end);
            console.log(elapsedTime);

            var experiment = await Experiment.findByIdAndUpdate(_id, {
              $set: {
                status: "completed",
                output: {
                  type: "matrix",
                  duration: elapsedTime.elapsedTimeAsMs,
                  result: {X, Y, centroids }
                }
              }
            }, { new: true });

            await axios.post(`${config.notifier.uri}/to`, {
            clientId: clientId,
            message: {
            body: `${experiment.title} completed`,
            info: elapsedTime.elapsedTimeAsStr
            }});
            await sendEmailNotification(dataset.workspace.owner.email,
            "experiment completed",
            `your experiment ${experiment.title} has been completed,
            elapsedTime : ${elapsedTime.elapsedTimeAsStr} seconds`);
```

```
33
34              break;
```

Listing A.4: K-means

A.5. SVM código javascript

```
1  case "svm":
2
3                var features = math.matrix(data)
4              .subset(math.index(math.range(startIndex, data.length), featuresIdx))
5              .valueOf();
6                var Y = responseIdx ?
7                math.squeeze(math.matrix(data)
8                .subset(math.index(math.range(startIndex, data.length),
9                responseIdx)))
10               .valueOf() : responseIdx;
11               var kfolds = parseInt(parameters.kfolds);
12               var gamma = parseFloat(parameters.gamma);
13               var coef = parseInt(parameters.coef);
14               var nu = parseFloat(parameters.nu);
15               var epsilon = parseFloat(parameters.epsilon);
16               var iters = parseInt(parameters.iters);
17
18               var matrix = math.concat(features,math.transpose([Y])).valueOf();
19               var result = await validateSvm(
20               matrix,
21               kfolds,
22               true,
23               gamma,
24               coef,
25               nu,
26               epsilon,
27               iters);
28               console.log(result.accuracy);
29               var end = new Date();
30               var elapsedTime = evalElapsedTime(start, end);
31                 var experiment = await Experiment.findByIdAndUpdate(_id, {
```

```
                    $set: {
                      status: "completed",
                      output: {
                        type: "matrix",
                        duration: elapsedTime.elapsedTimeAsMs,
                        result: result
                      }
                    }
                  }, { new: true });
                await axios.post(`${config.notifier.uri}/to`, {
            clientId: clientId,
            message: {
            body: `${experiment.title} completed`,
            info: elapsedTime.elapsedTimeAsStr
            }});
            await sendEmailNotification(dataset.workspace.owner.email,
            "experiment completed",
            `your experiment ${experiment.title} has been completed,
            elapsedTime : ${elapsedTime.elapsedTimeAsStr} seconds`);

            break;
```

Listing A.5: SVM

A.6. Perceptron Multicapa Código Javascript

```
...
  case "mlp":
             var features = math.matrix(data)
            .subset(math.index(math.range(startIndex, data.length), featuresIdx))
            .valueOf();
              var Y = responseIdx ?
              math.squeeze(math.matrix(data)
              .subset(math.index(math.range(startIndex, data.length),
              responseIdx)))
              .valueOf() : responseIdx;
             var kfolds = parseInt(parameters.kfolds);
             var activationfn= parameters.activationfn;
```

```
13              var trainingfn = parameters.trainingfn;
14              var hiddenLayers = parameters.hiddenLayers.split(",")
15              .map(e=>parseInt(e));
16              var epsilon = parseFloat(parameters.epsilon);
17              var iters = parseInt(parameters.iters);
18
19              var matrix = math.concat(features,math.transpose([Y])).valueOf();
20               var result = await validateMlp(
21               matrix,
22               kfolds,
23               true,
24               activationfn,
25               trainingfn,
26               hiddenLayers,
27               epsilon,
28               iters);
29               console.log(result.accuracy);
30               var end = new Date();
31               var elapsedTime = evalElapsedTime(start, end);
32                 var experiment = await Experiment.findByIdAndUpdate(_id, {
33                     $set: {
34                       status: "completed",
35                       output: {
36                         type: "matrix",
37                         duration: elapsedTime.elapsedTimeAsMs,
38                         result: result
39                       }
40                     }
41                   }, { new: true });
42
43                  await axios.post(`${config.notifier.uri}/to`, {
44             clientId: clientId,
45             message: {
46             body: `${experiment.title} completed`,
47             info: elapsedTime.elapsedTimeAsStr
48             }});
49             await sendEmailNotification(dataset.workspace.owner.email,
```

```
50              "experiment completed",
51              `your experiment ${experiment.title} has been completed,
52              elapsedTime : ${elapsedTime.elapsedTimeAsStr} seconds`);
53              break;
54 ....
```

Listing A.6: MLP

Referencias

Algorithmia (2018). Algorithmia developer center. 18

Amazon (2018). Amazon machine learning. 18

Bengio, Y. (2016). *Deep Learning*. The MIT Press, Printed in Cambridge, Massachusetts. 11, 12

Berson, A. (1992). *Client/Server Architecture*. J. Ranade Series. 16

Beyeler, M. (2017). *Machine Learning for OpenCV*. Packt, University of washington. 14

BigMl (2018). Bigml. [Online; accessed April 09, 2018]. 18

Bishop, C. (2006). *Pattern Recognition and Machine Learning*. Springer, Printed in Cambridge. 11

Brownlee, D. (2015). Basic concepts in machine learning. 7, 8

Center, M.N. (2016a). Democratizing ai. for every person and every organization. 6

Center, M.N. (2016b). Microsoft expands artificial intelligence (ai) efforts with creation of new microsoft ai and research group. 7

Erl, T. (2013). *Cloud Computing: Concepts, Technology Architecture*. Prentice Hall. 17

Erlang (2018). What is erlang? 44

Forbes, R. (2012). Is cloud computing really cheaper? [Online; accessed April, 2018]. 7

Forbes, R. (2017). The next phase of the cloud computing revolution is here.

Forbes, R. (2018). How ai is transforming the future of healthcare. [Online; accessed April, 2018]. 6

Gobbo, N.D. (2017). How i ported bcrypt to new n-api. 57

Gonzalez, R.C. (1977). *Digital Image Processing*. Prentice Hall, Printed in New Jersey. 13, 14

Google (2018a). App engine. 17

Google (2018b). Google cloud prediction api. 18

Guojun Gan, C.M. & Wu, J. (2007). *Data Clustering: Theory, Algorithms, and Applications*. American Statistical Association and the Society for Insudtrial and Applied mathemathics.

IBM (2018). Watson. 19

Ihrig, C.J. (2015). *Pro Nodejs fro developers*. Apress.

Johansson, L. (2015). Getting started with rabbitmq and node.js. 55

Microsoft (2018). Microsoft azure machine learning studio. 18

Mohr, S. (1999). *Designing Distributed Applications*. Wrox Press, Birmingham. 15

Moujahid, A. (2016). Machine learning approachs. [Online; accessed April 04, 2018]. vii, 10

news for students, S. (2014). The hippocampus, shown here in a mouse. [Online; accessed April 04, 2018]. vii, 13

opencv (2015). Cuda and opencv. 57

Oscar Denis Suarez, I.S.G.y.J.S.t., Noealia Vállez Enano (2014). *OpenCV Essentials*. Packt, Birmingham. 14

Rogers, S. & Girolami, M. (2012). *A First course of machine learning*. CRC Press, Cambridge.

Schildt, H. (2000). *C The Complete Reference*. McGraw-Hill, San Francisco.

Serrano, A.G. (2012). *DesigningInteligencia Artificial (Fundamentos, práctica y aplicaciones)*. Alfaomega, Birmingham.

Shalev-Shwartz, S. & Ben-David, S. (2014). *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, Cambridge. 7, 9

Szeliski, R. (2010). *Computer Vision: Algorithms and Applications*. Springer. 14

Wikipedia (2013). Front-end y back-end. 16, 17