Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
Mención ingeniería del Software

INTERFAZ GRÁFICA PARA UNA BATERÍA DE EXPERIMENTOS PARA IDENTIFICACIÓN DE USUARIOS CON DATOS DE RELOJES INTELIGENTES

Alumno:
Méndez Calvo, Manuel

Tutor:
Moro Sancho, Quiliano Isaac

Agradecimientos y Dedicatorias

A mi familia, cuyo apoyo ha hecho posible que esta etapa finalice con éxito.

A mi pareja, Nora, quien me ha animado y acompañado hasta el final.

A Arancha, por su perseverancia y su sabiduría.

A Quiliano, por su paciencia y guiado durante este proyecto.

Índice

Índice de figuras

Índice de tablas

1. Introducción

1.1. Motivaciones

Se parte del trabajo [1] realizado por un grupo de investigadores del Departamento de Informática de la UVa. En él se usan datos de dos tipos relojes inteligentes para identificar al usuario que lo lleva puesto. El elemento que realiza la identificación es una red neuronal artificial.

El diseño de los experimentos realizados condiciona la estructura de los datos con los que se van a contar en este TFG, por lo que en este trabajo asumimos que dichos datos ya están disponibles y han sido preprocesados de manera adecuada (la que se indica en el [1]).

Dado que, dentro del concepto de "red neuronal artificial", podemos encontrar un infinito número de posibilidades, todas ellas posibles candidatos al sistema identificador del usuario, se necesita un entorno de trabajo sencillo e intuitivo que permita la definición de un conjunto de modelos (en nuestro caso, redes neuronales artificiales), que los entrene, pruebe, y genere ficheros y gráficas de resultados que hagan posible su comparación y la elección del mejor modelo. Todo esto, orientado a un usuario que NO tiene conocimientos del lenguaje de programación usado para la construcción/entrenamiento/prueba del modelo (red neuronal).

1.2. Objetivos

El objetivo principal es desarrollar una Interfaz Gráfica funcional que pueda servir como un IDE desde el que se pueda crear, configurar y modificar diferentes clasificadores mediante el uso de Python.

Como objetivos auxiliares se quiere destacar el visionado de las ejecuciones de forma gráfica en función del error medio, y de igual manera, intentar aproximar el mundo de las redes neuronales y clasificadores mediante una interfaz simple con la que cualquier nuevo usuario podría empezar. No será necesario tener conocimientos previos de Python, Tensorflow o Keras. Tampoco de como diseñar baterías de experimentos, o de como tratarlas. Por lo tanto, se le intenta dar un enfoque simple e intuitivo, insistiendo en que se trate de una aplicación "User Friendly".

Todo esto implica que la interfaz debe permitir a cualquier usuario:

- **Seleccionar los datos** (el formato ya está prefijado, pero se podrán ajustar algunos parámetros si esto fuese necesario.)
- **Seleccionar dónde se almacenarán los ficheros con las distintas informaciones sobre cada experimento.**
- **Configurar el conjunto de clasificadores que se van a utilizar en los experimentos**
- **Configurar el Compile.** ((el proceso que construye el objeto que forma la red neuronal, así como los parámetros de entrenamiento, validación, medida de errores...)
- **Definir la batería de experimentos con los dos puntos anteriores.**

- **Ejecutar los experimentos (lo que incluye el entrenamiento y la prueba), y recoger los resultados obtenidos**
- **Visualizar los resultados de forma gráfica, si así se desea.**

1.3. Alcance

La aplicación ahora mismo estaría restringida a la UVa y el departamento de investigación de mi tutor, Quiliano Isaac Moro Sancho, pero el objetivo es que se pueda usar como prototipo para desarrollar interfaces más complejas con los objetivos que se han comentado anteriormente.

1.4. Estado del arte

Actualmente hay aplicaciones que ya se encargan de hacer este tipo de trabajo en mayor o menos medida. A continuación se van a listar los más conocidos y explicar brevemente su utilidad y aproximación a la funcionalidad de nuestro proyecto.

- **KNIME:** Se trata de un software de análisis de datos de código abierto que nos permite el diseño visual de pipelines de procesamiento de datos, así como la construcción de modelo de aprendizaje automático. En el siguiente enlace podremos acceder a la web oficial(`https://www.knime.com/`).

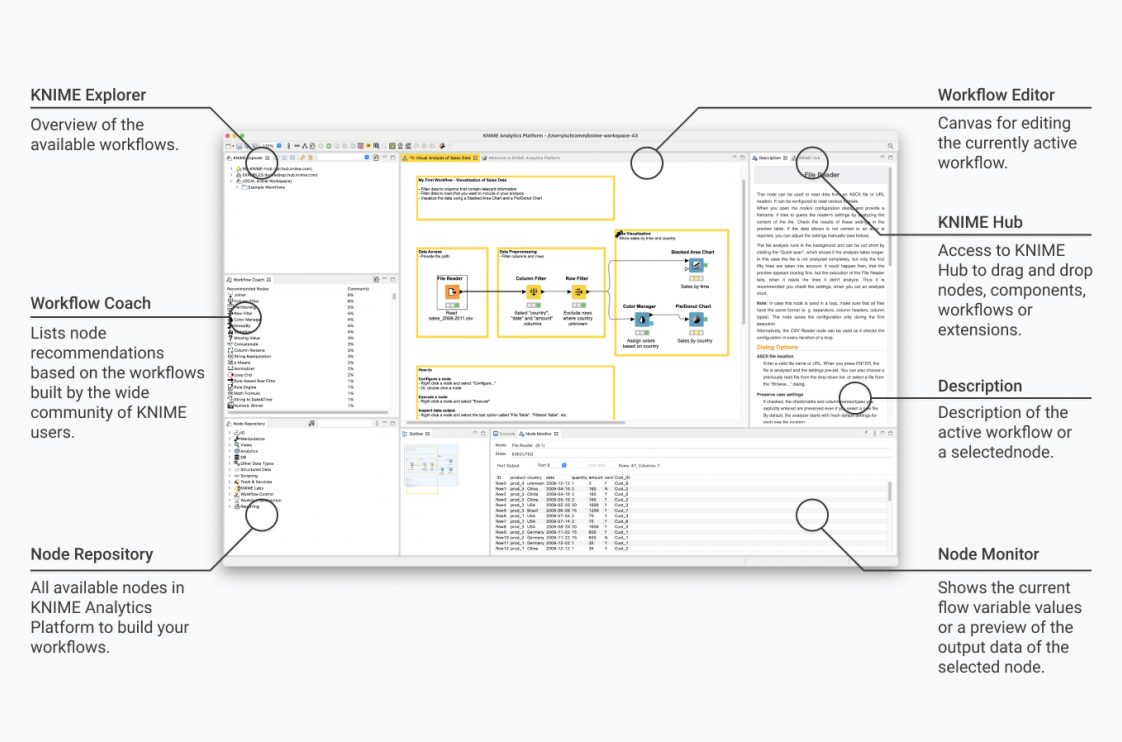

Figura 1: KNIME

- **Weka:** Es una suite de software de aprendizaje automática, también de código abierto, que permite la exploración de datos, su preparación y la construcción de modelos de aprendizaje automático. En el siguiente enlace podremos acceder a la web oficial (`https://www.cs.waikato.ac.nz/ml/weka/`).

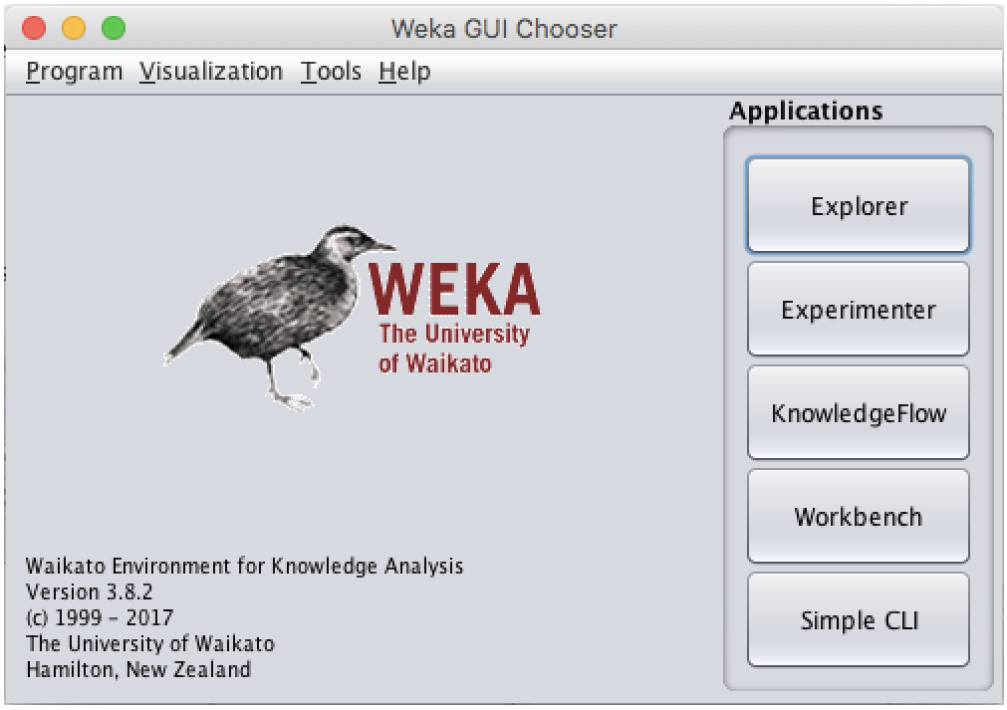

Figura 2: Weka

- **Orange:** En este caso se trata de una plataforma de análisis de datos de código abierto, la cual nos permite una construcción visual de diferentes flujos de trabajo de análisis de datos y la correspondiente creación de modelos de aprendizaje automático. En el siguiente enlace podremos acceder a la web oficial(`https://orangedatamining.com/`).

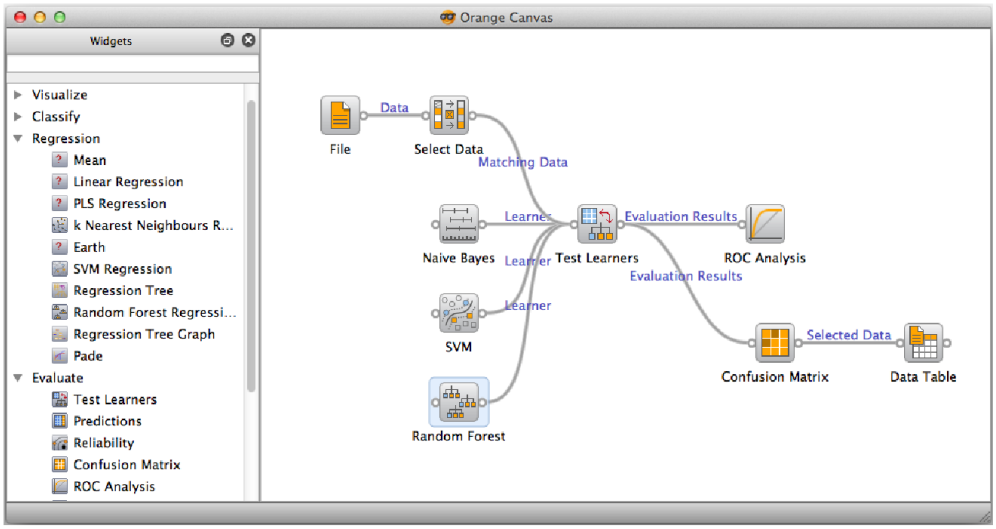

Figura 3: Orange

- **RapidMiner:** Cumple con la misma funcionalidad que la anterior mencionada **Orange**. De nuevo, en el siguiente enlace tenemos la web oficial (`https://rapidminer.com/`).

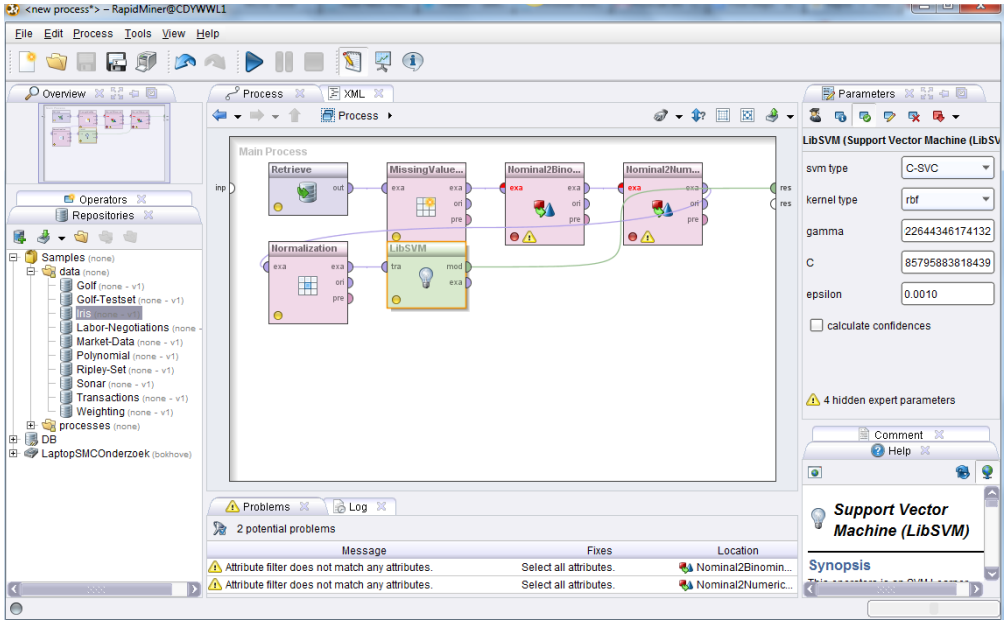

Figura 4: RapidMiner

En este caso debemos hacer una herramienta final que será de uso especifico y limitado al tipo de datos proporcionado por el anterior trabajo de investigación ***[1]***

2. Planificación

En esta sección se introducirá la organización, distribución de las fases del trabajo con su correspondiente estimación del tiempo de desarrollo y la metodología utilizada.

2.1. Metodología de trabajo

Para este proyecto se ha decidido seguir una **metodología Agile**, donde se van proponiendo una serie de **Sprints** que comprenden tiempo desde 4 días hasta 20 aproximadamente para poder ir mejorando el proyecto, y abarcando poco a poco todas las dificultades que puedan surgir. A continuación se listan los diferentes sprints que se han propuesto, con su correspondiente fecha de inicio y fin y los temas a tratar.

Índice	Tema Sprint	Duración en días	Fecha Inicio	Fecha Fin
1	Análisis de requisitos	3	1/02/2023	4/02/2023
2	Búsqueda de la tecnología	7	6/02/2023	13/02/2023
3	Formación	10	14/02/2023	23/02/2023
4	Creación de un prototipo de la ventana Configuración de clasificador	10	27/02/2023	5/03/2023
5	Perfeccionamiento de la ventana y nuevas funcionalidades	20	7/03/2023	27/03/2023
6	Prototipo de la ventana principal	10	28/03/2023	7/04/2023
7	Perfeccionamiento de la ventana principal y funcionalidad	10	10/04/2023	20/04/2023
8	Parámetros del compilador y adaptación de la funcionalidad	20	20/04/2023	10/05/2023
9	Logger informativo para el usuario	10	10/05/2023	20/05/2023
10	Formación para hacer las gráficas	15	20/05/2023	4/06/2023
11	Implementación de la parte de gráficas	10	4/06/2023	14/06/2023
12	Perfeccionamiento de los últimos retoques	8	14/06/2023	22/06/2023

Tabla 1: Sprints

También se quiere destacar que mediante esta metodología será posible hacer reuniones de seguimiento periódicas cómodas tanto para el tutor como para el alumno, dando así lugar a una comunicación y monitorización adecuada del proyecto. Por último comentar que en cada sprint se trabajará en la **Memoría de TFG** con sus correspondientes correcciones.

La repartición de roles que se ha llevado a cabo son:

- **Tutor** $\rightarrow$
    1. **Scrum master**
    2. **Product Owner**

- **Alumno →**
    1. **Product Manager**
    2. **Equipo de desarrollo (todo el conjunto)**

2.2. Gestión de riesgos y oportunidades

En esta sección se plasmará una tabla con los posibles riesgos del proyecto y otra tabla con las posibles oportunidades. Los riesgos estarán compuestas de los siguientes campos;

- **Identificador de riesgo.**
- **Nombre.**
- **Breve descripción.**
- **Categoría.**
- **A qué afecta la vulnerabilidad**
- **Probabilidad de que ocurra (“Media, alta o baja”).**
- **Su impacto (“Medio, alto o bajo”).**
- **La exposición al riesgo (“Media, alta o baja”).**
- **Estado de riesgo.**
- **Acciones de mitigación.**
- **Acciones correctivas.**

Para las oportunidades, los campos que componen el esquema de la tabla son:

- **Identificador de oportunidad.**
- **Descripción de la oportunidad.**
- **Nombre de la oportunidad.**
- **Probabilidad de que ocurra (“Media, alta o baja”).**
- **Cuán positivo es su alcance (“Medio, alto o bajo”).**
- **Nivel de oportunidad (“Media, alta o baja”).**
- **Estado de la oportunidad.**

Para poder gestionar tanto los riesgos como las oportunidades, se ha utilizado [2]

2.2.1. Riesgos

Como se puede ver en la figura **5**, tenemos la distribución anteriormente mencionada. A destacar, la probabilidad alta-media-baja hace referencia a cuán probable se ha previsto que pueda suceder. El impacto se refiere al impacto inmediato del mismo en el proyecto, y finalmente la exposición al riesgo es un cálculo entre ambas partes.

ID	Nombre	Descripción	Categoría	Vulnerabilidad	Amenaza	Probabilidad	Impacto	Riesgo Total, exposición o nivel de riesgo	Estado del Riesgo	Acciones de mitigación	Acciones correctivas
RSK001	Irrupción del trabajo	Debido a una causa de fuerza mayor, puede que me vea incapacitado para continuar mi trabajo durante un periodo de tiempo	Personal	Causa de fuerza mayor	Necesidad de atrasar la entrega o modificar los plazos	ALTA	BAJO	MEDIO	ACEPTADO	1. Preveer los tiempos teniendo en cuenta que este riesgo es imposible de evitar	1. Reorganizar el proyecto 2. Modificar los plazos de entrega
RSK002	Tecnología nueva	El proyecto requiere del uso de una tecnología desconocida para mi hasta ahora	Complejidad tecnológica	Tecnología nueva para el usuario	Necesidad de aprendizaje previo antes de empezar el desarrollo del proyecto	ALTA	MEDIO	ALTO	ABIERTO	1. Anticiparme los plazos y determinar un periodo exclusivo para la formación 2. Elegir un proyecto con una tecnología en la que ya esté especializado 3. Obtener una formación especializada	1. Replanificar el proyecto 2. Extender los plazos de entrega
RSK003	Programas insoportables	El equipo no puede soportar el software necesario para desarrollar el proyecto	Complejidad tecnológica	El equipo está demasiado obsoleto como para poder correr el software necesario	Podría suceder que el equipo sea demasiado obsoleto y haya que buscar otros medios para poder llevar a cabo el proyecto	BAJA	MEDIO	BAJO	ABIERTO	1. Desarrollo del proyecto en una máquina virtual remota que cumpla los requisitos necesarios para utilizar el software	1. Reorganizar el proyecto 2. Obtener otro equipo si así fuese posible 3. Utilizar un equipo alternativo para desarrollar las partes del proyecto que no puede cubrir nuestro ordenador
RSK004	Trabajo indisciplinado	Debido a no seguir una metodología especifica para trabajar (como scrum), el trabajo podría verse retrasado debido a la falta de presión, o concrección de plazos	Personal	Falta de disciplina debido a la no estipulación de plazos con tecnologías Aguiles	Problemas a la hora de presentar y defender el propio tfg debido a la falta de disciplina	BAJA	ALTO	MEDIO	MITIGANDOSE	1. Implementar desde el principio un horario de trabajo y una tecnología Aguile para cumplicar con plazos y evolutivos de forma organizada	1. Reorganizar el proyecto
RSK005	Desinteres	A lo largo del proyecto puede uno darse cuenta de que no es lo que estaba esperando y perder la motivación progresivo	Personal	Desinteres	Perdida completa de la motivación y por tanto procrastinación total del trabajo o incluso abandono	MEDIA	ALTO	ALTO	ABIERTO	1. Determinar tiempo al proyecto, pero cuando sea necesario tomarse un tiempo de pausa 2. Cada vez que se cumpla un objetivo, obtener una pequeña recompensa	1. Reorganizar el proyecto 2. Encontrar uno nuevo

Figura 5: Riesgos

2.2.2. Oportunidades

De igual manera que en la sección de riesgos, aquí vemos una tabla con un par de oportunidades y la estructura mencionada en **2.2**

ID	Nombre	Descripción	Categoría	Oportunidad	Probabilidad	Impacto positivo	Nivel de oportunidad	Estado de oportunidad
O001	Aprendizaje nuevo	Especializarse en una nueva tecnología que nos abrirá nuevas puertas en el mundo laboral	Complejidad tecnológica	Especialización en una tecnología concreta	MEDIA	ALTO	ALTO	Abierta
O002	Oportunidad laboral	Posibilidad de hacer el tfg en una empresa con posibilidad de contratación posterior	Personal	Salida al mundo laboral	MEDIA	BAJO	BAJO	Aprovechando

Figura 6: Oportunidades

2.3. Coste

A continuación se citarán los costes estimados que ha tenido el proyecto. Para ello se ha utilizado la bibliografía indicada en [2].

2.3.1. Coste personal

Sabiendo que la carga lectiva del TFG es de 300 horas, de acuerdo a la tarifa oficial, el coste estimado es de 3000€ brutos.

2.3.2. Coste de software

El coste de software también ha sido gratuito, ya que se ha utilizado la versión "community" de **PyCharm**, la versión gratuita de **Overleaf** y la versión gratuita de **QT Designer**

2.3.3. Coste de formación

Para la formación se ha utilizado un par de cursos de **Udemy** [3] , así como diferentes libros y tutoriales. El coste del curso ha sido de 15€, mientras que los otros tipos de formación han sido gratuitos.

2.3.4. Coste de Hardware

Estrictamente hablando, el hardware tampoco ha tenido coste, puesto que el alumno poseía todo lo necesario para su desarrollo. Sin embargo, en caso de tener que alquilar el material o comprarlo, se ha hecho la siguiente estimación.

Objeto	**Coste (€)**
Ordenador Portátil	900
Monitor	150

Tabla 2: Coste hardware

2.3.5. Coste Total

En consecuencia el coste total inicial estimado es de 31065€, como podemos ver en la tabla 2.3.5

Necesidad	**Coste (€)**
300 horas de proyecto	10€/h
PyCharm	0
Overleaf	0
QTDesigner	0
Udemy	15
Portatil	900
Monitor	150
Total	4065

Tabla 3: Coste Total

3. Análisis del sistema

A lo largo de este apartado se tratará de mostrar la estructura y comportamiento que debería tener nuestro sistema. Para ello se han examinado las interacciones entre los componentes del sistema. Se mostrará una licitación de actores, requisitos, y demás información que pueda resultarnos de interés.

También hablaremos brevemente de los Requisitos de Hardware, y como es el equipo que ha sido utilizado para realizar este proyecto.

3.1. Requisitos de Hardware

Esta aplicación no necesita requisitos de hardware concretos, pero hay que tener en cuenta que hace una enorme cantidad de cálculos, y que además utilizamos paralelización a través de hilos en nuestro programa, logrando así hacer las 5 repeticiones que tenemos puesto para entrenar cada modelo.. Esto implica que cuantos más núcleos tenga nuestro procesador, y más memoria RAM disponible, antes terminará las ejecuciones. Sin embargo, a continuación cito los componentes más importantes con los que se han llevado a cabo las pruebas.

- **Procesador:** Intel Core i5-11600k 3,9GHz con 6 cores
- **Memoria ram:** 2x8GB DDR4
- **Tarjeta Gráfica:** Asus Phoenix GeForce RTX 3060 12GB GDDR6

Para dar una visión aproximada explicaré 2 ejecuciones diferentes en la sección *7* que he hecho que nos permitirán entender cuanto se tarda aproximadamente.

3.2. Funcionalidades de la aplicación

La aplicación se distribuye en una ventana principal y varias ventanas auxiliares. Desde la ventana principal se irán invocando las diferentes ventanas para hacer posible el cumplimiento de los objetivos anteriormente mencionados. A continuación, se explicarán las ventanas que hay sin entrar en detalle sobre su funcionamiento, ya que esto ocupará una sección entera más adelante:

- **Ventana principal**. Se trata del hilo conductor donde se llevará a cabo nuestra ejecución
- **Configurar Clasificador**. Esta Ventana nos permitirá crear la configuración del clasificador de forma simple e intuitiva.
- **Configuración del Compilador**. Aquí podremos definir una configuración especifica que queremos que tenga el modelo a la hora de hacer la ejecución.
- **Generar Gráficas**. La función de esta ventana será la visualización de los resultados de los diferentes experimentos que hagamos de forma gráfica

3.3. Actores del sistema

En principio, el software está destinado a estar integrado en el trabajo de investigación [1], como se ha indicado con anterioridad, y la idea de la interfaz es que el sistema interactué con un único actor (*Usuario*)

3.4. Requisitos

En esta subsección se desarrollarán los requisitos, tanto funcionales como no funcionales.

3.4.1. Requisitos funcionales

Los requisitos funcionales son aquellos que reflejan lo que el sistema debe permitir hacer, es decir, la funcionalidad del sistema. En la tabla 4 se puede ver la funcionalidad que debe abarcar nuestro sistema

ID	Descripción
RF-01	El sistema permitirá introducir una ruta con el fichero de datos a usar
RF-02	El sistema permitirá introducir una ruta para los ficheros de salida
RF-03	El sistema permitirá configurar un clasificador adecuado para el input dado
RF-04	El sistema permitirá añadir nuevas capas en el clasificador
RF-05	El sistema permitirá editar parámetros concretos del clasificador
RF-06	El sistema permitirá borrar capas del clasificador
RF-07	El sistema permitirá intercambiar el lugar de las capas del clasificador
RF-08	El sistema permitirá configurar los parámetros de la función de compilación del modelo/red
RF-09	El sistema permitirá ejecutar el entrenamiento y la prueba de la modelo/red.
RF-10	El sistema permitirá mostrar una ventana para visualizar los resultados
RF-11	El sistema permitirá seleccionar los campos que queremos utilizar para hacer las gráficas
RF-12	El sistema permitirá visualizar los resultados de forma gráfico de Barras
RF-13	El sistema permitirá visualizar los resultados de forma gráfico de Dispersión
RF-14	El sistema permitirá visualizar los resultados de forma gráfico de Lineas
RF-15	El sistema permitirá visualizar los resultados de forma gráfico de Pastel
RF-16	El sistema permitirá visualizar los resultados de forma gráfico de Caja y bigotes

Tabla 4: Requisitos Funcionales

3.4.2. Requisitos no funcionales

Los requisitos no funcionales están destinados a indicar cuales son las limitaciones con las que deberá lidiar nuestro sistema.

ID	Descripción
RNF-01	El sistema solo funcionará en *Python* 3.9.12 o superior
RNF-02	El sistema solo funcionará con la versión del paquete *Keras* 2.12 o superior
RNF-03	El sistema solo funcionará con la versión del paquete *MatPlotlib* 3.7.1 o superior
RNF-04	El sistema solo funcionará con la versión del paquete *NumPy* 1.23.5 o superior
RNF-05	El sistema solo funcionará con la versión del paquete *Pandas* 2.0.2 o superior
RNF-06	El sistema solo funcionará con la versión del paquete *TensorFlow* 2.12.0 o superior
RNF-07	El sistema solo funcionará con la versión del paquete *xmltodict* 0.13.0 o superior
RNF-08	El sistema solo funcionará PyQT6, pero podrá funcionar con PyQT5 en su mayoría
RNF-09	El sistema permitirá adaptar la ventana a diferentes tamaños según prefiera el usuario
RNF-10	El sistema notificará al usuario en caso de que alguna información que haya introducido no sea correcta
RNF-11	El sistema habilitará los botones de forma adecuada para que el usuario pueda seguir el flujo del programa
RNF-12	El sistema debe poder procesar los datos de forma adecuada y almacenar los ficheros con los resultados en un sistema de subdirectorios donde indique el usuario
RNF-13	El sistema debe mostrar al usuario la traza de la ejecución para que pueda ver que está sucediendo en cada momento
RNF-14	El sistema debe ser simple e intuitivo para reducir al mínimo el margen de error del usuario

Tabla 5: Tabla de requisitos no funcionales

3.4.3. Casos de uso

A continuación se muestra el diagrama principal simplificado de ***Casos de Uso*** y sus descripciones.

Como indicación especial del diagrama **7** destacar que todas las opciones se encuentran en la pantalla principal, desde la que se ve van mostrando las otras ventanas.

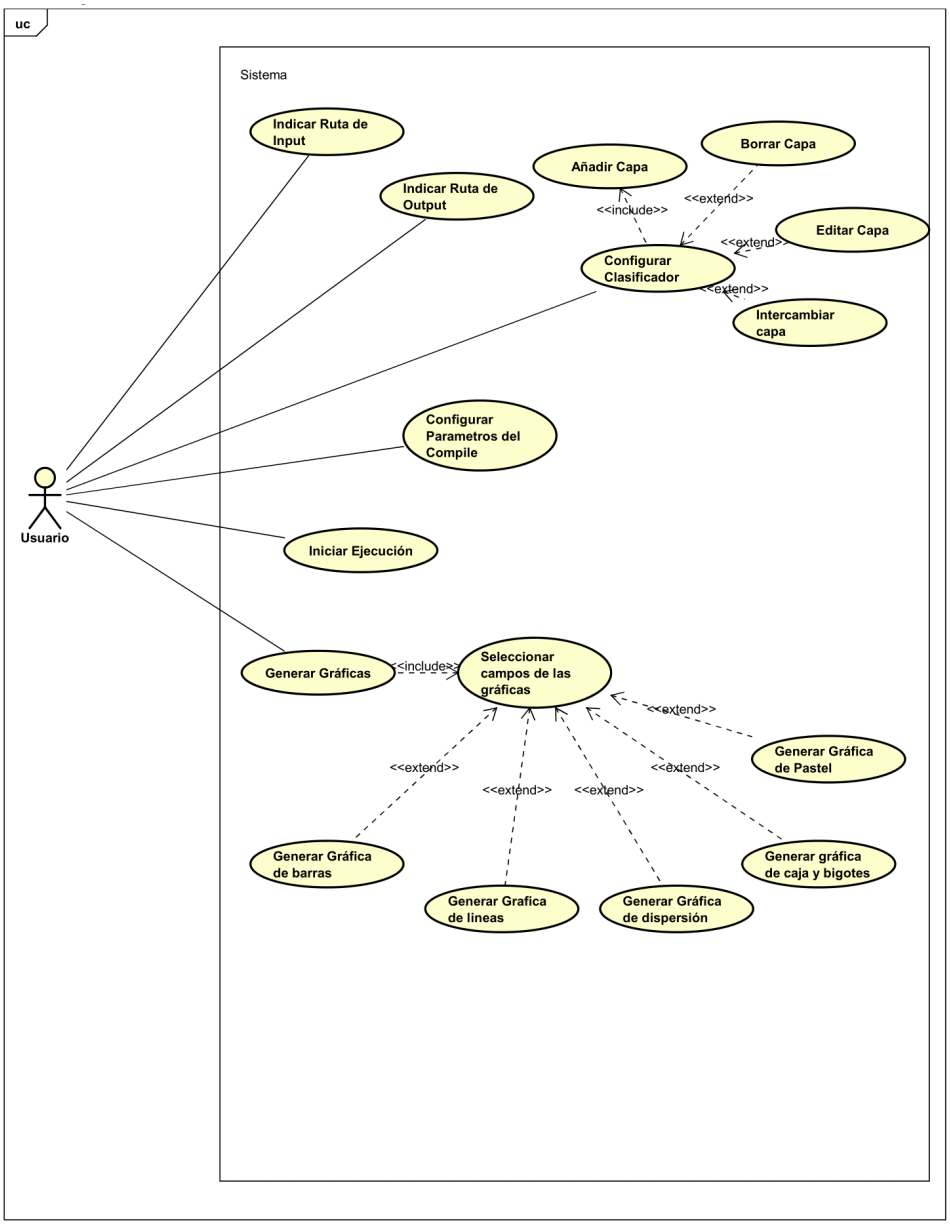

Figura 7: Diagrama Casos de Uso

A continuación se irán describiendo las diferentes especificaciones de los casos de uso mostrados en el diagrama, indicando a qué requisito funcional hacen referencia

Identificador	*CU-01 Introducir Ruta entrada*
Descripción	Este caso de uso describe el proceso de introducir la ruta de entrada para el sistema.
RF	*RF-01*
Actor	Usuario
Precondiciones	1.- El sistema está en un estado válido para recibir la ruta de Input.
Flujo Principal	1.- El usuario selecciona el botón “Ruta Input” 2.- El sistema muestra un pop-up para introducir la ruta de input 3.- El usuario introduce la ruta de los ficheros de input 4.- El sistema procesa la ruta de entrada
Flujo Alternativo	3a.- El usuario cierra el cuadro de texto. El caso de uso vuelve al paso 1
Postcondiciones	- La ruta de Input queda guardada. - Se habilitan los botones de “Configurar Clasificador” y “Parametros del Compile”.

Tabla 6: CU-01 Introducir Ruta Entrada

Identificador	*CU-02 Introducir Ruta Salida*
Descripción	Este caso de uso describe el proceso de introducir la ruta de salida para el sistema.
RF	*RF-02*
Actor	Usuario
Precondiciones	1.- El sistema está en un estado válido para recibir la ruta de Salida.
Flujo Principal	1.- El usuario selecciona el botón“Ruta Output”. 2.- El sistema muestra un pop-up para introducir la ruta de salida. 3.- El usuario introduce la ruta de los ficheros de salida. 4.- El sistema procesa la ruta de salida.
Flujo Alternativo	3a. El usuario cierra el cuadro de texto. 1.-El caso de uso vuelve al paso 1.
Postcondiciones	- La ruta de Output queda guardada. - Se habilita el botón de ”Gráficas”.

Tabla 7: CU-02 Introducir Ruta Salida

Identificador	*CU-03 Configurar Clasificador*
Descripción	Este caso de uso describe el proceso de configuración del clasificador en el sistema.
RF	*RF-03*
Actor	Usuario
Precondiciones	1.- El sistema contiene una ruta de input ***Ver 6***.
Flujo Principal	1.- El usuario selecciona el botón "Configurar Clasificador". 2.- El sistema muestra la interfaz de configuración del clasificador.
Flujo Alternativo	
Postcondiciones	Se muestra la ventana correspondiente a la configuración del clasificador.

Tabla 8: CU-03 Configurar Clasificador

Identificador	*CU-04 Añadir Capa*
Descripción	Este caso de uso describe el proceso de añadir una capa al clasificador en el sistema.
RF	*RF-04*
Actor	Usuario
Precondiciones	1.- La interfaz de Configurar Clasificador está abierta. Ver ***8***
Flujo Principal	1.- El usuario selecciona el botón "Añadir". 2.- El sistema muestra un pop-up para introducir el nombre de la capa. 3.- El usuario introduce el nombre de la capa. 4.- El sistema muestra un pop-up para seleccionar el tipo de capa. 5.- El usuario introduce el tipo de capa. 6.- El sistema muestra un pop-up para introducir el valor de Nep. 7.- El usuario introduce el valor de Nep. 8.- El sistema muestra un pop-up para seleccionar el tipo de activación. 9.- El usuario introduce el tipo de función de activación. 10.- El sistema comprueba que los datos introducidos sean válidos. 11.- El sistema añade la capa al clasificador.
Flujo Alternativo	10a.- El sistema detecta que los datos introducidos no son válidos. 1.- El sistema muestra un mensaje de error indicando el campo inválido. 2.- El caso de uso vuelve al paso 2 del flujo principal.
Postcondiciones	- Se añade una capa al clasificador en el sistema y se guarda la configuración

Tabla 9: CU-04 Añadir Capa

Identificador	*CU-05 Modificar Campo de Capa*
Descripción	Se describe el proceso de modificar un campo de una capa del clasificador.
RF	*RF-05*
Actor	Usuario
Precondiciones	1.- La interfaz de Configurar Clasificador está abierta.***Ver 8*** 2.- Existe al menos una capa creada en el clasificador. ***Ver 9***
Flujo Principal	1.- El usuario selecciona el campo de la capa a modificar. 2.- El sistema muestra un pop-up para introducir los nuevos datos. 3.- El usuario modifica el valor de un campo específico. 4.- El sistema valida el nuevo valor del campo. 5.- El sistema actualiza el campo modificado en la capa.
Flujo Alternativo	4a.- El sistema detecta que el nuevo valor del campo no es válido. 1.- El sistema muestra un mensaje de error indicando el valor inválido. 2.- El caso de uso vuelve al paso 1 del flujo principal.
Postcondiciones	- Se actualiza el campo del clasificador y se guarda la configuración actual

Tabla 10: CU-05 Modificar Campo de Capa

Identificador	*CU-06 Borrar Capa*
Descripción	Este caso de uso describe el proceso de borrar una capa del clasificador en el sistema.
RF	*RF-06*
Actor	Usuario
Precondiciones	1.- La interfaz de Configurar Clasificador está abierta.***Ver 8*** 2.- Existe al menos una capa creada en el clasificador. ***Ver 9***
Flujo Principal	1.- El usuario selecciona la capa que desea borrar. 2.- El usuario selecciona el botón “Borrar”. 3.- El sistema muestra un pop-up de confirmación para borrar la capa. 4.- El usuario selecciona “Sí”para confirmar el borrado. 5.- El sistema borra la capa seleccionada y actualiza el clasificador.
Flujo Alternativo	4a.- El usuario selecciona “No” en el pop-up de confirmación. 1.- El sistema cancela la operación de borrado. 2.- El caso de uso finaliza sin realizar cambios en el clasificador.
Postcondiciones	La capa seleccionada se ha borrado del clasificador.

Tabla 11: CU-06 Borrar Capa

Identificador	*CU-07 Intercambiar Capas*
Descripción	Se describe el proceso de intercambiar capas en el clasificador del sistema.
RF	*RF-07*
Actor	Usuario
Precondiciones	1.- La interfaz de Configurar Clasificador está abierta.***Ver 8*** 2.- Existe al menos dos capas creadas en el clasificador. ***Ver 9***
Flujo Principal	1.- El usuario selecciona la capa que desea mover. 2.- El usuario selecciona la flecha hacia arriba o hacia abajo para mover la capa. 3.- El sistema intercambia la posición de la capa seleccionada con la capa adyacente.
Flujo Alternativo	2a.- La capa seleccionada es la primera y se selecciona la flecha hacia arriba. 1.- El sistema notifica el error y muestra de nuevo el clasificador. 2b.- La capa seleccionada es la última y se selecciona la flecha hacia abajo. 1.- El sistema notifica el error y muestra de nuevo el clasificador.
Postcondiciones	Las capas seleccionadas han intercambiado su posición en el clasificador.

Tabla 12: CU-07 Intercambiar capa

Identificador	*CU-08 Configurar Parámetros Compile*
Descripción	Se describe el proceso de configurar los parámetros de compilación en el sistema.
RF	*RF-08*
Actor	Usuario
Precondiciones	1.- El sistema contiene una ruta de input ***Ver 6***.
Flujo Principal	1.- El usuario selecciona la opción “Parámetros del Compile”. 2.- El sistema muestra la interfaz de configuración de los parámetros de compilación. 3.- El usuario elige la “función loss” deseada. 4.- El usuario selecciona el optimizer deseado. 5.- El usuario introduce el número de épocas deseado. 6.- El sistema valida los parámetros introducidos. 7.- El sistema guarda la configuración de los parámetros de compilación.
Flujo Alternativo	7a.- El usuario introduce valores incorrectos para los parámetros. 1.- El sistema muestra un mensaje de error indicando valores inválidos. 2.- El caso de uso vuelve al paso 2.
Postcondiciones	Los parámetros de compilación han sido configurados y guardados en el sistema.

Tabla 13: CU-08 Configurar Parámetros del Compile

Identificador	*CU-09 Iniciar Ejecución*
Descripción	Se describe el proceso de iniciar la ejecución del sistema una vez que se han configurado todos los elementos necesarios
RF	*RF-09*
Actor	Usuario
Precondiciones	- Se ha indicado una ruta de entrada válida. ***Ver 6*** - Se ha indicado una ruta de salida válida. ***Ver 7*** - El clasificador ha sido configurado correctamente.***Ver 9*** - Los parámetros del compile han sido indicados correctamente.***Ver 13***
Flujo Principal	1. El usuario inicia selecciona el botón “Iniciar Ejecución”. 2. El sistema verifica que se cumplen las precondiciones. 3. El sistema procesa los datos de entrada utilizando el clasificador y los parámetros del compile. 4. El sistema genera los resultados de la ejecución. 5. El sistema almacena los resultados en la ruta de output indicada.
Flujo Alternativo	3a. Hay algún error en la ejecución. 1. El sistema muestra un mensaje de error indicando el motivo. 2. El caso de uso finaliza.
Postcondiciones	- Los resultados de la ejecución han sido almacenados en la ruta de output. - El sistema vuelve a mostrar la interfaz principal.

Tabla 14: CU-09 Iniciar Ejecución

Identificador	*CU-10 Obtener Gráficas*
Descripción	Se describe el proceso de muestra de la ventana de visualización
RF	*RF-10*
Actor	Usuario
Precondiciones	- Se ha indicado una ruta de output válida.***Ver 7*** - La ruta de salida contiene la información necesaria para generar las gráficas.
Flujo Principal	1. El usuario selecciona la opción “Gráficas”. 2. El sistema verifica que se cumplan las precondiciones. 3. El sistema muestra la ventana de visualización al usuario
Flujo Alternativo	2a. El sistema detecta que alguna de las precondiciones no se cumple. 1. El sistema muestra un mensaje de error indicando el motivo. 2. El caso de uso finaliza.
Postcondiciones	- Las gráficas han sido mostradas al usuario. - El sistema vuelve al estado inicial.

Tabla 15: CU-10 Gráficas

Identificador	*CU-11 Seleccionar Campos*
Descripción	Se describe el proceso de selección de campos para generar gráficas
RF	*RF-11*
Actor	Usuario
Precondiciones	- Se ha indicado una ruta de output válida. ***Ver 7*** - La ruta de salida contiene datos válidos. - La ventana de visualización de gráficas está mostrada. ***Ver 15***
Flujo Principal	1. El usuario selecciona el botón “Seleccionar Campos”. 2. El sistema muestra una checklist con todos los campos posibles para generar las gráficas 3. El usuario selecciona los campos que quiere utilizar para generar las gráficas.
Flujo Alternativo	3a. El usuario cierra la checklist. 1. El sistema vuelve a mostrar la ventana de visualización de gráficas.
Postcondiciones	- Se habilitan los botones de generar gráficas. - Se guardan los campos seleccionados por el usuario.

Tabla 16: CU-11 Seleccionar Campos

Identificador	*CU-12 Generar Gráfico de Barras*
Descripción	Se describe el proceso de generación de gráficos de barras
RF	*RF-12*
Actor	Usuario
Precondiciones	- Se ha indicado una ruta de output válida. ***Ver 7*** - La ruta de salida contiene datos válidos. - La ventana de visualización de gráficas está mostrada. - Se han seleccionado campos para generar las gráficas.
Flujo Principal	1. El usuario selecciona el botón ”Gráfico de Barras”. 2. El sistema genera una gráfica de barras para cada atributo seleccionado. 3. El sistema genera una gráfica de barras general con todos los atributos juntos.
Flujo Alternativo	
Postcondiciones	El sistema muestra al usuario las gráficas y le permite interactuar con ellas

Tabla 17: CU-12 Generar Gráfico de Barras

Identificador	*CU-14 Generar Gráfico de Dispersión*
Descripción	Se describe el proceso de generación de gráficos de dispersión
RF	*RF-12*
Actor	Usuario
Precondiciones	- Se ha indicado una ruta de output válida. ***Ver 7*** - La ruta de salida contiene datos válidos. - La ventana de visualización de gráficas está mostrada. - Se han seleccionado campos para generar las gráficas.
Flujo Principal	1. El usuario selecciona el botón ”Gráfico de Dispersión”. 2. El sistema genera una gráfica de dispersión para cada atributo seleccionado. 3. El sistema genera una gráfica de dispersión general con todos los atributos juntos.
Flujo Alternativo	
Postcondiciones	El sistema muestra al usuario las gráficas y le permite interactuar con ellas

Tabla 18: CU-13 Generar Gráfico de Dispersión

Identificador	*CU-14 Generar Gráfico de líneas*
Descripción	Se describe el proceso de generación de gráficos de líneas
RF	*RF-14*
Actor	Usuario
Precondiciones	- Se ha indicado una ruta de output válida. ***Ver 7*** - La ruta de salida contiene datos válidos. - La ventana de visualización de gráficas está mostrada. - Se han seleccionado campos para generar las gráficas.
Flujo Principal	1. El usuario selecciona el botón ”Gráfico de Líneas”. 2. El sistema genera una gráfica de líneas para cada atributo seleccionado. 3. El sistema genera una gráfica de líneas general con todos los atributos juntos.
Flujo Alternativo	
Postcondiciones	El sistema muestra al usuario las gráficas y le permite interactuar con ellas

Tabla 19: CU-14 Generar Gráfico de Líneas

Identificador	*CU-15 Generar Gráfico de Pastel*
Descripción	Se describe el proceso de generación de gráficos de pastel
RF	*RF-14*
Actor	Usuario
Precondiciones	- Se ha indicado una ruta de output válida. ***Ver 7*** - La ruta de salida contiene datos válidos. - La ventana de visualización de gráficas está mostrada. - Se han seleccionado campos para generar las gráficas.
Flujo Principal	1. El usuario selecciona el botón ”Gráfico de Pastel”. 2. El sistema genera una gráfica de pastel para cada atributo seleccionado. 3. El sistema genera una gráfica de pastel general con todos los atributos juntos.
Flujo Alternativo	
Postcondiciones	El sistema muestra al usuario las gráficas y le permite interactuar con ellas

Tabla 20: CU-15 Generar Gráfico de Pastel

Identificador	*CU-16 Generar Gráfico de Caja y Bigotes*
Descripción	Se describe el proceso de generación de gráficos de Caja y Bigotes
RF	*RF-14*
Actor	Usuario
Precondiciones	- Se ha indicado una ruta de output válida. ***Ver 7*** - La ruta de salida contiene datos válidos. - La ventana de visualización de gráficas está mostrada. - Se han seleccionado campos para generar las gráficas.
Flujo Principal	1. El usuario selecciona el botón ”Gráfico de Caja y Bigotes”. 2. El sistema genera una gráfica de caja y bigotes para cada atributo seleccionado. 3. El sistema genera una gráfica de caja y bigotes general con todos los atributos.
Flujo Alternativo	
Postcondiciones	El sistema muestra al usuario las gráficas y le permite interactuar con ellas

Tabla 21: CU-15 Generar Gráfico de Caja y Bigotes

4. Arquitectura del Software

En este apartado nos centraremos en cuál es la idea inicial de cómo construir las ventanas y su comportamiento, explicando el motivo de que tengan la forma elegida.

4.1. Ventana Principal

Esta es la ventana más importante y será el hilo conductor de nuestra aplicación. Su misión será ir mostrando una traza de la ejecución a medida que invoca a las diferentes acciones según se necesiten. La idea es hacerla simple e intuitiva, y permitir una ejecución guiada al usuario. Además se plantea la idea de que sea reescalable para que el log pueda verse mejor si así lo desea el usuario. Para ello, el layout se debería mantener constante indiferentemente de cómo se cambie el tamaño de la ventana

El croquis inicial de la Ventana Principal lo podemos apreciar en ***8***, donde aún no ha habido ninguna interacción. La ventana se divide en 2 partes, una donde están los botones, y otra donde está el log que se utilizará para que el usuario obtenga información de lo que va pasando con la aplicación.

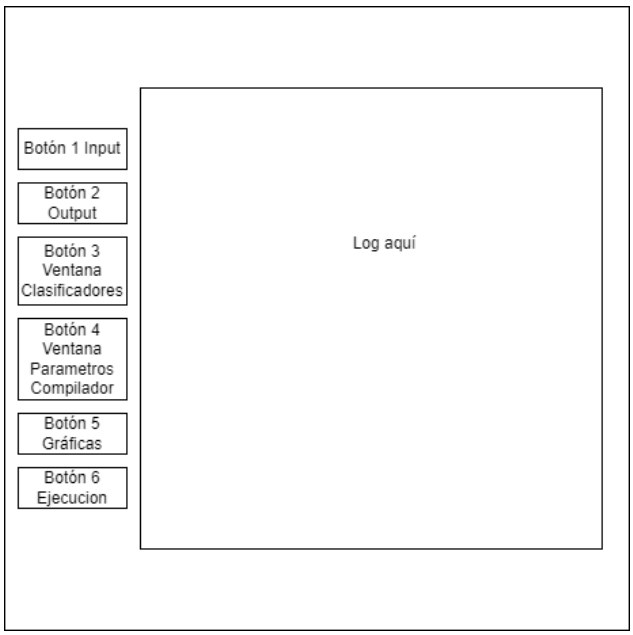

Figura 8: Croquis de la Ventana Principal

Los pasos a seguir serían: introducir la ruta dónde estará el fichero con los datos al dar al "botón 1" mediante un pop-up o un cuadro de dialogo. El segundo botón funcionará de igual manera pero se indicará la ruta donde se quiere dejar los resultados. El tercer botón nos abrirá una instancia de una ventana que nos permita configurar nuestros propios clasificadores mientras que el cuarto botón nos abrirá una instancia de otra ventana para introducir los parámetros del compilador. Por

último, habrá un botón para iniciar la ejecución y otro para ver las diferentes gráficas con los resultados obtenidos.

4.2. Configurar Clasificador

La ventana de configuración del clasificador nos permitirá generar nuestros propios clasificadores para utilizarlos finalmente en nuestra ejecución. La idea de esta ventana es que también sea una ventana simple e intuitiva. De primeras, nos encontraríamos ante una tabla vacía donde se introducirían los parámetros del clasificador. La ventana tendrá 3 botones, uno para añadir capas nuevas, otro para borrarlas, y por último uno para editar algún elemento concreto de una capa. El croquis del aspecto inicial sería el que podemos apreciar en ***9***.

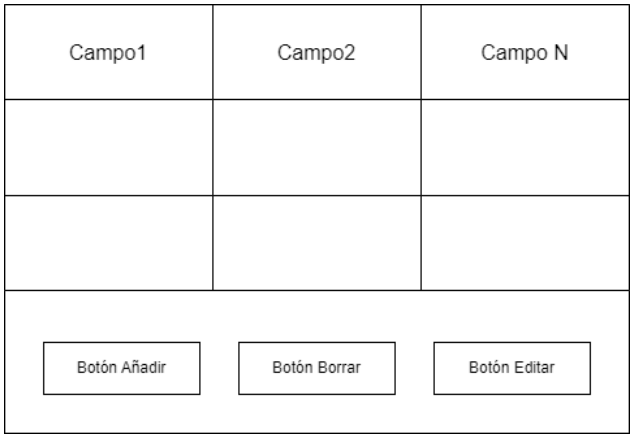

Figura 9: Croquis de Configurar Clasificador

4.3. Parámetros del Compile

Esta ventana sería la encargada de mostrar al usuario las posibles opciones que tendrá para elegir los parámetros para la compilación de la red. En principio, solo habrá que concretar la "Función Loss", el "Algoritmo de optimización" y por último las "Épocas". El croquis inicial se puede ver en ***10***

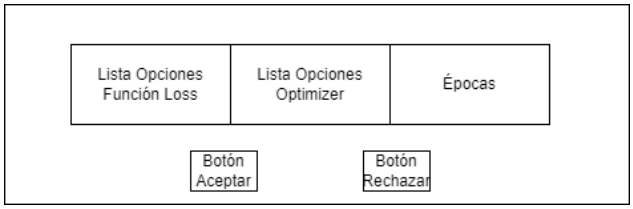

Figura 10: Croquis de Parámetros del Compilador

4.4. Ejecución

Al dar al botón de “Ejecución”, la idea es que el log de la ***“Ventana Principal”*** vaya escribiendo una traza para que así el usuario pueda tener un feedback constante y pueda seguir la ejecución. De esta forma no sería necesario crear otra ventana para esta parte, haciendo que la navegabilidad del usuario entre las ventanas sea más simple

4.5. Gráficas

La idea de esta ventana es que el usuario pueda hacer gráficas para poder visualizar los resultados de la ejecución. Como croquis inicial tenemos la imagen ***11***, donde podemos apreciar que el usuario podrá elegir entre “n” opciones diferentes de gráfica que visualizará en la parte derecha de la propia ventana.

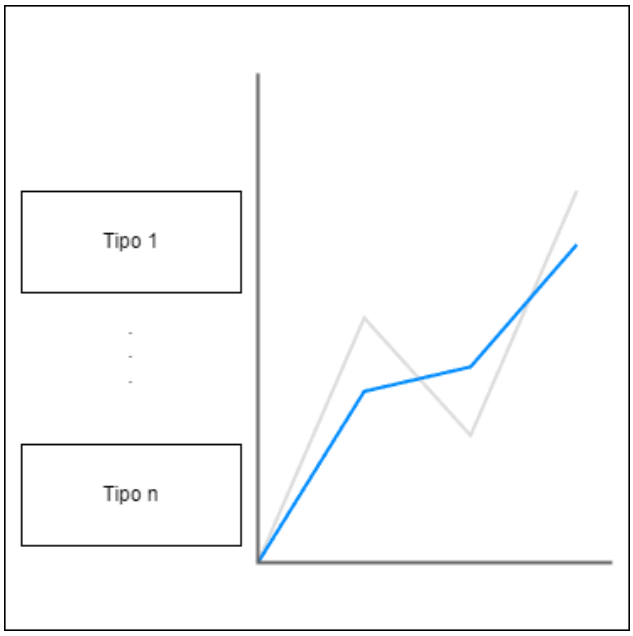

Figura 11: Croquis de Gráficas

5. Implementación final

En esta sección se mostrará cuál es el resultado final de las ventanas de las que se ha hablado en la sección ***4***. Además se intentará explicar en detalle como es el flujo de interacción de las diferentes ventanas, y cuál es su comportamiento final, haciendo de esta forma un manual para el usuario.

5.1. Ventana Principal

El diseño final de la ***Ventana Principal*** podemos apreciarlo en ***12***. Como se puede ver, es bastante similar al croquis dado, solo que se ha añadido un scroll a la parte del log y se muestra un mensaje con instrucciones para poder realizar la ejecución de forma guiada. Además se puede apreciar que solo están activados los botones de *"Ruta Input"* y *"Ruta Output"* para que así la ejecución sea todavía más intuitiva.

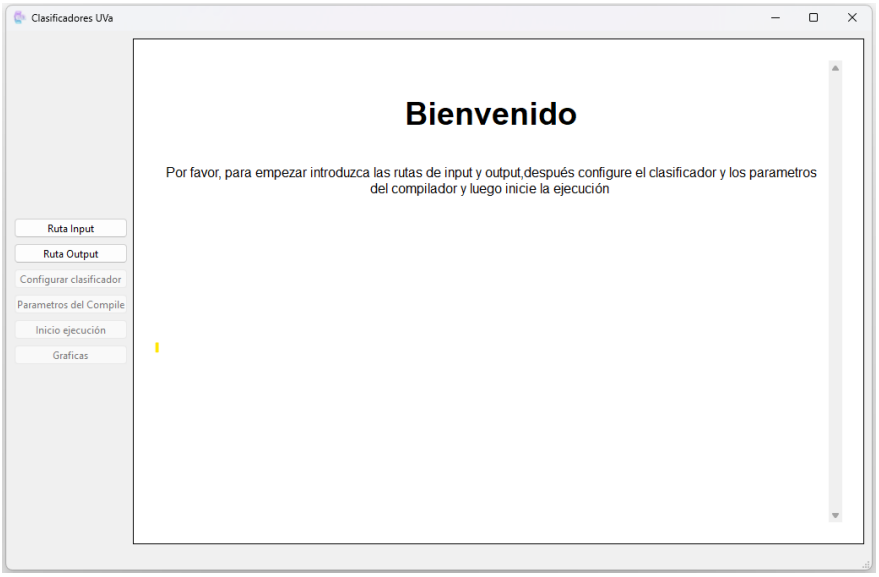

Figura 12: Ventana Principal

Una vez se introducen las rutas, se hacen las verificaciones oportunas, y si alguna no se cumple, se muestra un mensaje de error y se pide de nuevo esa ruta. Sin embargo, hay una validación muy especifica. Esta aplicación está pensada para leer un archivo de datos de tipo csv llamado *Todo* (Todo.csv), de modo que al introducir la ruta de input se valida que en ese directorio exista el csv con ese nombre, ya que sino no sería posible realizar la ejecución.

Una vez introducidas las rutas correctamente la ventana actualizar el log y nos muestra un mensaje indicándonos las rutas introducidas y habilitando 3 botones, que se corresponden a 3 nuevas ventanas como puede verse en 13.

Por otra parte, las ventanas habilitadas son:

- ***Configurar Clasificador***, de la cual hablaremos en el apartado ***5.2***.
- ***Parámetros del Compile***, la cual también desarrollaremos en el apartado 5.3.
- ***Gráficas***, la cual se desarrollará en profundidad en el apatado 5.5

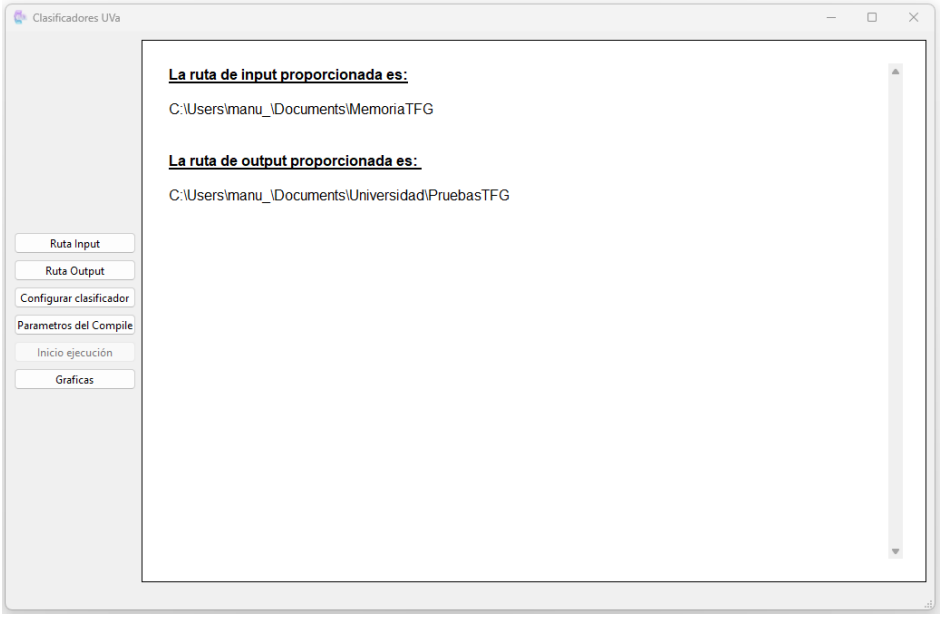

Figura 13: Ventana Principal Con rutas

5.2. Configurar Clasificador

El aspecto final de la ventana ***"Configurar Clasificador"*** dista un poco de la idea inicial. Al principio únicamente tenía 3 botones, mientras que ahora tiene 5. A parte de los botones mencionados en la sección***4***, se ha añadido dos botones que nos permitirán mover las capas por si queremos cambiarlas de orden. Además se ha restringido la tabla a 4 columnas, debido a que se ha decidido que los clasificadores sean únicamente para redes de tipo MLP.

A lo largo de las diferentes subsecciones de esta ventana se irá mostrando la funcionalidad final de cada uno de estos botones, dando lugar a la creación final del clasificador. Por último también se hablará de las validaciones que tiene cada celda de forma general.

5.2.1. Botón añadir

Al principio solo podemos dar al botón de "Añadir". Este botón nos mostrará un cuadro de dialogo para introducir cada uno de los elementos que tendrá nuestra capa y los añadirá a la tabla, generando así un nueva capa del clasificador. En la imagen ***14*** podemos ver el aspecto que tendrían estos pop-ups y como se han ido añadiendo las capas al clasificador. Además podemos ver que se nos han habilitado otros botones, cuya funcionalidad se explica en los 3 apartados siguientes.

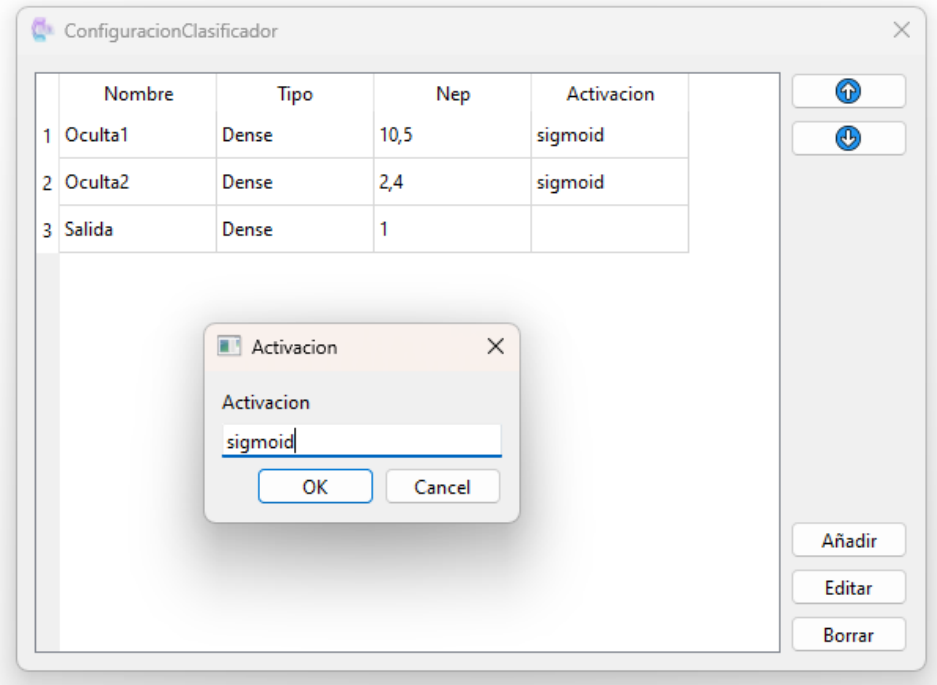

Figura 14: Configurar Clasificador Con 3 capas

5.2.2. Botón Editar

El botón de editar nos permitirá seleccionar una celda en concreto de una capa del clasificador y mediante un pop-up se pedirá el nuevo dato al usuario. Si el usuario introduce un valor que cumpla las validaciones, el valor anterior será reemplazado en el clasificador. El pop-up que se muestra se vería como en la figura ***15***, mostrándonos el valor de la celda que vamos a editar y marcando la celda de la tabla en gris para que el usuario pueda reconocerla

5.2.3. Botón Borrar

El botón de borrar nos permitirá eliminar una capa completa del clasificador. Para ello mostrará un pop-up de seguridad preguntando si de verdad desea borrarla y si es así, se eliminará la fila de la tabla y y por tanto la capa del clasificador.

5.2.4. Botones de desplazar

Los botones de desplazar son dos flechas que nos permitirá intercambiar las filas (capas) de lugar.

5.2.5. Validaciones

Para hablar de las validaciones hay que dejar claro que esta app es escalable, y que ahora mismo solo está funcionando para MLP, sin embargo, en la sección ***9*** se proponen una serie de mejoras para que pueda abarcar más tipos de redes neuronales. Las limitaciones de las que se va a hablar a

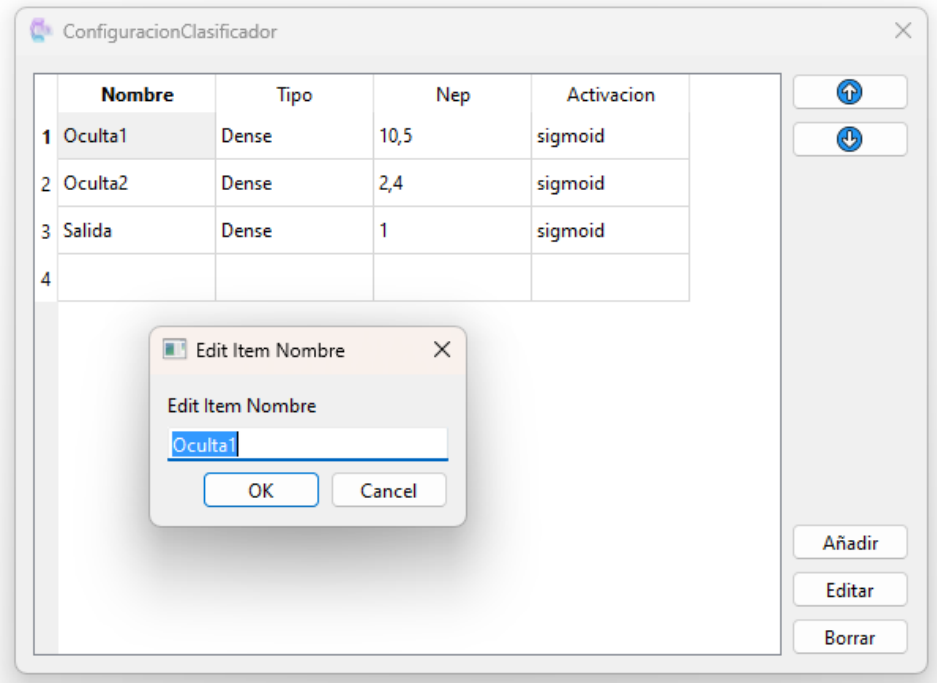

Figura 15: Configurar Clasificador Botón Editar

continuación están añadidas tanto a la función de "añadir capa" como a la de "editar los items de la capa".

- ***La celda "Tipo":*** está limitada al parámetro "Dense"
- ***La celda "Nep":*** debe ser una lista de enteros separada por comas
- ***La función de activación:*** está limitada a *"sigmnoid", "relu", "tanh", "softmax"*.

Para los botones de desplazar, la limitación es que si no hay celda con la que intercambiarlo se indicará al usuario y no se hará el cambio.

5.2.6. Funciones de activación

En este apartado se va a hablar brevemente de lo que es una función de activación y que como funcionan las elegidas para nuestro programa. La función de activación introduce no linealidad en la salida de una neura, lo que permite a la red neuronal aprender y modelar relaciones más complejas entre entradas y salidas. Si no utilizásemos función de activación, la capacidad de la red neuronal sería muy limitada puesto que solo computaría una combinación lineal de entradas. A continuación se profundizará en la explicación de las funciones utilizadas en nuestro programa

- ***sigmoid*** $\rightarrow$ Mapea los valores de entrada a un rango entre 0 y 1, lo cual hace que sea especialmente útil en problemas de clasificación binaria.
- ***relu*** $\rightarrow$ Tiene un formato no lineal que retorna el valor de entrada si es mayor que cero, o cero en caso contrario. Esta función es altamente eficiente a nivel computacional, y tiene

un extra añadido debido a que nos ayuda a resolver el problema del desvanecimiento del gradiente

- ***tanh*** → La *Tangente Hiperbólica* es similar a la función "sigmoid", pero en lugar de mapear la entrada entre 0 y 1, lo mapea entre -1 y 1
- ***softmax*** → Se utiliza en la capa de salida de una red neurnal para resolver problemas de clasificación multiclase. Para ello mapea las salidad a una distribución de probabilidad, donde la suma de todas las salidas debe ser 1.

5.2.7. Traza en la ventana de ejecución

Cuando cerramos la ventana del clasificador, el log de la ventana principal nos mostrará como ha quedado nuestro clasificador en un formato de tabla ***HTML*** como podemos ver en la imagen ***16***.

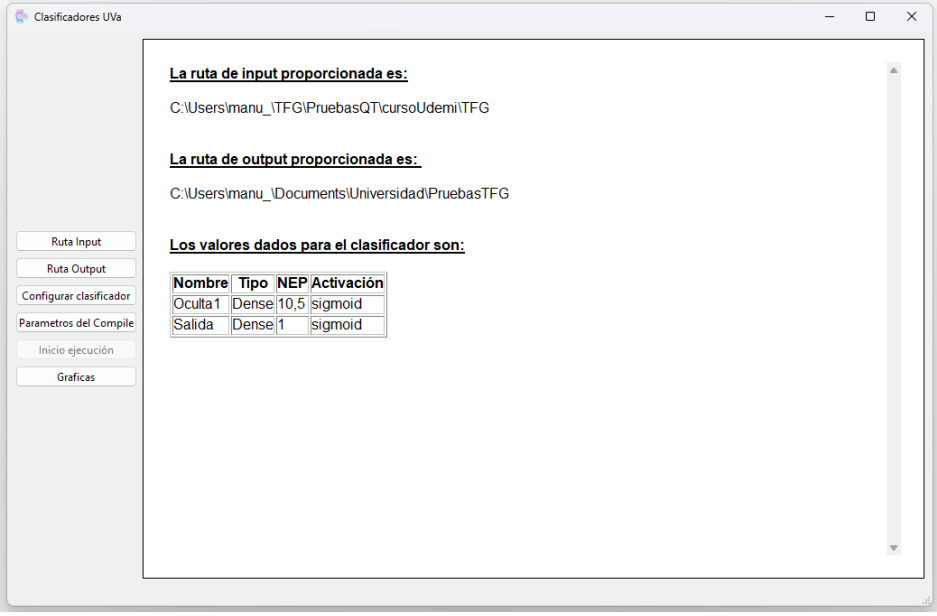

Figura 16: Ventana Principal Tabla Clasificador

5.3. Parámetros del Compile

La ventana de ***"Parámetros del Compilador"*** podemos verla en ***17***. Ha mantenido la esencia del croquis previamente diseñando, haciendo que la lista de los posibles parámetros, tanto de la "Función Loss" como del "Optimizer" sean una lista, mientras que las "Epocas" son un entero estrictamente positivo. Estos parámetros son los que se utilizarán tanto para entrenar el modelo con la función *fit* como para usar la función *compile*. La ventana tiene un diseño muy simple y minimalista, logrando de esta forma que su interacción sea simple e intuitiva, y evitando errores imprevistos.

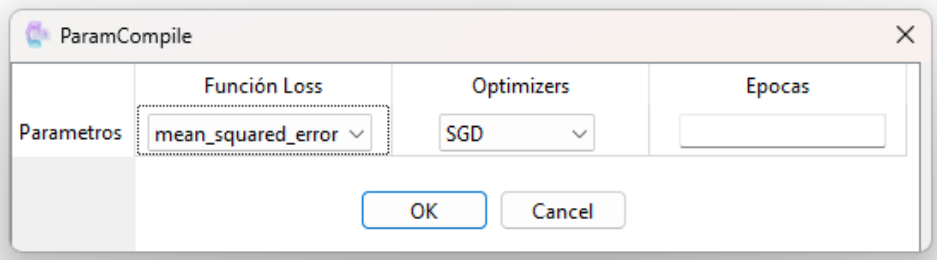

Figura 17: Parámetros del Compile

A continuación se explicará en detalle cuales son las opciones que tendrá el usuario para elegir y como funciona cada una de ellas

- ***Función loss:*** esta función se utiliza para calcular una medida numérica que nos permitirá calcular la discrepancia entre las predicciones del modelo con los valores reales. Las posibles opciones que facilita nuestra ventana son:

  - ***mean_squared_error*** → Calcula el error cuadrático medio. Para ello calculará los promedios de los errores al cuadrado y lo divide entre el numero de muestras. Su formula sería $mean\_squared\_error = (1/n) * sum((y_{pred} - y_{true})^2)$, donde y_{pred} representa la predicción del modelo, y_{true} representa el valor real y n es el numero total de ejemplos del conjunto de datos
  - ***categorical_crossentropy*** → Calcula la entropía cruzada categórica, para ello compara la distribución de probabilidades predicha por el modelo con la distribución real de las etiquetas. Su formula sería $categorical\_crossentropy = -sum(y_{true} * log(y_{pred}))$ donde y_{pred} representa la predicción del modelo, y_{true} representa el valor real
  - ***binary_crossentropy*** → Calcula la entropía cruzada binaria, solo sirve cuando hay 2 clases por lo que realmente ahora no sería útil para nuestro proyecto, pero está pensado por si se quiere hacer un estudio más concreto de los datos con únicamente 2 clases. Su formula es $binary\_crossentropy = -(y_{true} * log(y_{pred}) + (1 - y_{true}) * log(1 - y_{pred}))$ donde y_{pred} representa la predicción del modelo, y_{true} representa el valor real
  - ***mean_absolute_error*** → Calcula el error absoluto medio. Se utiliza para medir la discrepancia promedio absoluta entre los valores que han sido predichos por el modelo y los valores reales que hay en el conjunto de datos. Su formula es $mean\_absolute\_error = (1/n) * \sum |y_{true} - y_{pred}|$ donde y_{pred} representa la predicción del modelo, y_{true} representa el valor real y n es el numero total de ejemplos del conjunto de datos
  - ***mean_squared_logarithmic_error*** → Calcula el error cuadrático logarítmico medio de manera muy similar al *mean_squared_error*, pero en lugar de calcular las diferencias cuadradas entre las predicciones y los valores reales, primero toma el logaritmo de ambos y luego calcula la diferencia cuadrada. Su formula es $mean\_squared\_logarithmic\_error = (1/n) * \sum(log(1 + y_{true}) - log(1 + y_{pred}))^2$ donde y_{pred} representa la predicción del modelo, y_{true} representa el valor real y n es el numero total de ejemplos del conjunto de datos

- ***kullback_leibler_divergence*** $\rightarrow$ Calcula divergencia de Kull-Leibber que trata de medir la diferencia dos distribuciones de probabilidad. Es una medida de la diferencia entre una distribución de probabilidad estimada por el modelo y una distribución de probabilidad de referencia. Su formula es $kullback\_leibler\_divergence(P||Q) = \sum P(x) * log(P(x)/Q(x))$ donde P(x) y Q(x) son las probabilidades de los eventos x en las distribuciones P y Q, respectivamente.

- ***Optimizer:*** Se trata de un parámetro referido al algoritmo de optimización que se utiliza para ajustar los pesos del modelo durante el proceso de entrenamiento. Además es el responsable de actualizar los parámetros del modelo para minimizar la función de pérdida durante el entrenamiento. El objetivo final del optimizador es encontrar los valores óptimos de los parámetros del modelo que minimicen la función de pérdida y mejoren el rendimiento del modelo en los datos de entrenamiento. Las posibles opciones que se ofrecen al usuario son:

  - ***SGD*** $\rightarrow$ Se trata del **descenso estocástico del gradiente**, su objetivo es minimizar la función de perdida actualizando los pesos del modelo en función del gradiente de la función de perdida calculado en un subconjunto de muestras aleatorias. La formula que lo describiría es $\theta = \theta - \alpha * \nabla J(\theta, x)$ donde θ representa los pesos del modelo, α representa el factor de aprendizaje que determina el tamaño del paso que se da en la dirección opuesta al gradiente, y por último $\nabla J(\theta, x)$ es el gradiente de la función de perdida (J) con con respecto a los pesos, calculado en base a un subconjunto de muestras x. Como se ha explicado, esto es un algoritmo estocástico, lo que significa que los pesos se actualizan de forma estocástica en cada iteración, permitiendo una convergencia muy rápida y una gran eficiencia computacional.
  - ***RMSprop*** $\rightarrow$ El *Root Mean Square Propagation* es una variable de SGD, que busca mejorar aún más la convergencia y el rendimiento. Para ello, adapta el factor de aprendizaje de forma individual para cada parámetro del modelo. La formula de RMSprop se divide en dos partes, la primera sería calcular el peso, haciéndose de igual manera que en el SGD pero diviendolo entre la raíz de la suma de la acumulación de los cuadrados del gradiente (ϵ) y un valor muy pequeño que impide hacer divisones por 0 (σ). La formula del calculo de pesos entonces sería: $\theta = \theta - \alpha * \frac{(\nabla J(\theta,x)}{\sqrt{(\sigma+\epsilon)}}$. Por otra parte tendríamos la obtención de la acumulación de los cuadrados del gradiente, cuya formula es $\sigma = \beta * \sigma + (1-\beta) * (\nabla J(\theta, x)^2)$. De esta fórmula, a mayores de lo anteriormente explicado habría que entender que β representa el factor de decaimiento que controla la contribución relativa de las actualizaciones anteriores, siendo su valor cercano a 0,9
  - ***Adam*** $\rightarrow$ *Adapttive Moment Estimation* es un algoritmo que combina el RMSprop y el algoritmo momentum. La idea principal detrás de Adam es ma ntener una estimación adaptativa de los momentos de primer y segundo orden del gradiente. Estos momentos se calculan utilizando promedios móviles ponderados de los gradientes y sus cuadrados, respectivamente. Durante el entrenamiento, Adam actualiza los parámetros del modelo tomando en cuenta tanto el gradiente actual como la estimación de los momentos anteriores.

- ***Épocas:*** Por último tenemos las épocas que están referidas a la cantidad de veces que el algoritmo de entrenamiento recorre todo el conjunto de datos de entrenamiento, a mayor

número de épocas, se puede permitir aprender patrones más complejos de datos, pero no puede ser demasiado grande debido a que sino se producirá sobreajuste

Finalmente, en la imagen ***18*** podemos ver como quedaría la ventana principal después de aceptar los parámetros introducidos. También se puede apreciar como se ha activado el botón de inicio de ejecución debido a que ya tenemos toda la configuración necesarios para la misma.

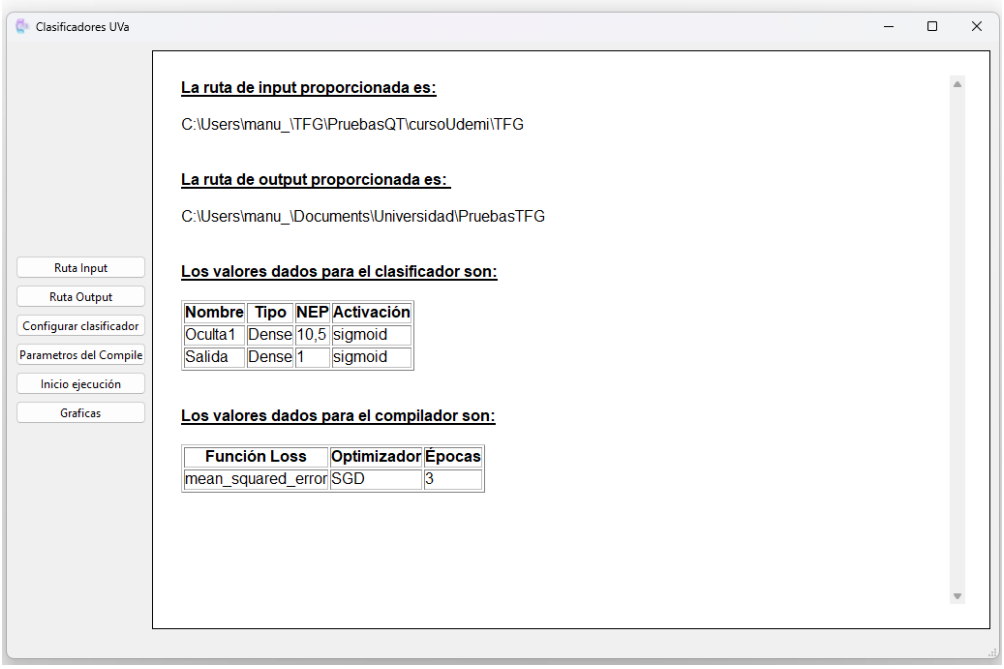

Figura 18: Ventana principal con Parámetros del Compile

5.4. Iniciar Ejecución

Se ha mantenido la idea de prevalecer la información de la ejecución en la ventana principial, actualizarndo el log para obtener la traza. También se ha decidido que la salida se almacene en la ruta de output, generando una jerarquía de ficheros donde podremos ver un *.mod* con el modelo y un *.xml* que inicialmente solo tendrá la descripción de la arquitectura del modelo, pero que una vez entrenado se añadirán los resultados.

La traza del programa completa podría verse en ***19 y 20***, donde se le ofrece información valiosa al usuario, desde el número de capas del metamodelo después de expandirlo, hasta los propios modelos. La información de después nos indica los pasos que está siguiendo el programa, esto se hace para que el usuario no tenga la sensación de estar en vacío esperando, sino que pueda obtener un feedback inmediato y hacerse una idea de por donde va progresando el programa. Hay que destacar que el programa consume gran cantidad de memoria RAM, de modo que si no se tiene demasiada, quizá el feedback vaya un poco desfasado.

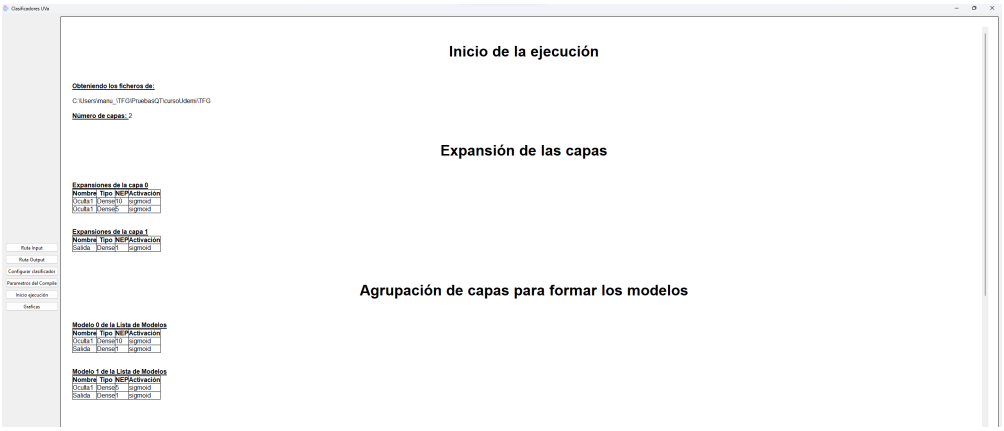

Figura 19: Primer parte de la ejecución

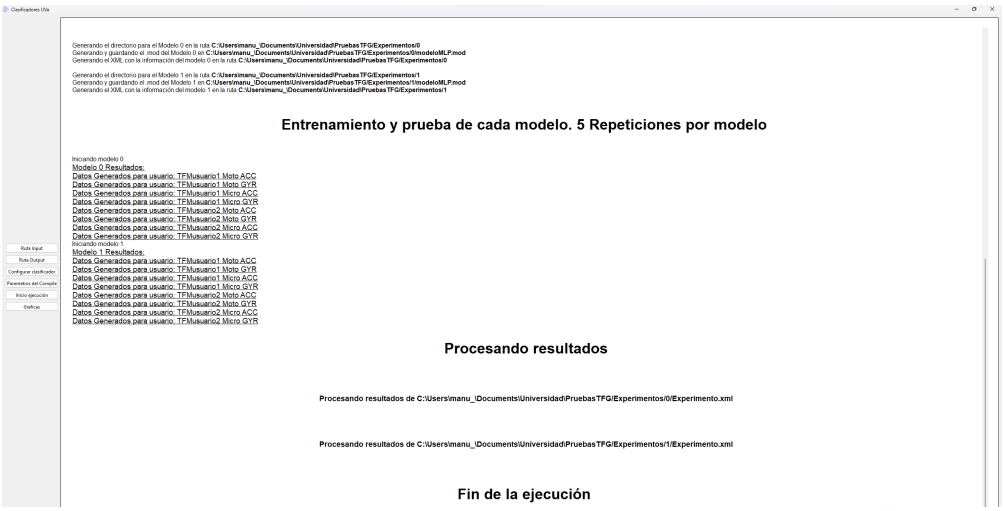

Figura 20: Segunda parte de la ejecución

De la imagen ***19***, además se quiere destacar la parte que se indica como “Expansión de las capas”. Esto es una de las características principales de este programa, y gracias a ello podemos crear y entrenar todos los modelos que queramos en una sola ejecución. La expansión de capas no es más que el producto cartesiano entre nos “Nep” de cada capa.

Por ejemplo, supongamos que tenemos 3 capas: La capa *Oculta1* con “Nep” 10 y 5, la capa *Oculta2* con “Nep” 2,4, y la capa de Salida con Nep 1, el resultado nos crearía 4 modelos con la combinación de los Nep de las capas $\rightarrow$ (10, 2, 1), (10, 4, 1), (5, 2, 1), (5, 4, 1)

5.5. Gráficas

Finalmente llegamos a la ventana de ***“Gráficas”***. Una ventana que ha cambiado bastante con respecto a la idea inicial que se motraba en el croquis. Lo primero de todo es que antes de abrirse, validará que exista una jerarquía de carpetas acorde a una ejecución realizada con esta aplicación. Al dar la botón de “Gráficas” se nos mostrará una nueva ventana con 6 botones, y solo 1 de ellos

habilitados al principio como podemos ver en ***21***.

Figura 21: Ventana Generar Gráfica

Cuando este botón es seleccionado, la aplicación mostrará una *"CheckList"* al usuario para que elija los atributos con los que el usuario podrá hacer las gráficas. Una vez el usuario ha seleccionado los campos sobre los que va a querer hacer las gráficas, se activarán el resto de botones como podemos comprobar en la imagen ***22***

Figura 22: Ventana Generar Gráfica con Campos Seleccionados

Una vez llegados a este punto, podemos elegir cualquier tipo de gráfica de las que vienen en la ventana. Cada una tendrá sus ventajas y sus desventajas, siendo la ventana que más información da, pero a la vez la más compleja, la de caja y bigotes. Una vez demos a uno de estos botones, se nos generará una gráfica independiente por cada tipo de atributo seleccionado, y luego una gráfica

general donde se unen las gráficas individuales.

En la imágenes ***23 y 24*** podemos ver como serían las gráficas individuales, mientras que en la gráfica ***25*** podemos ver como quedaría en una única imagen la únión de las gráficas individuales

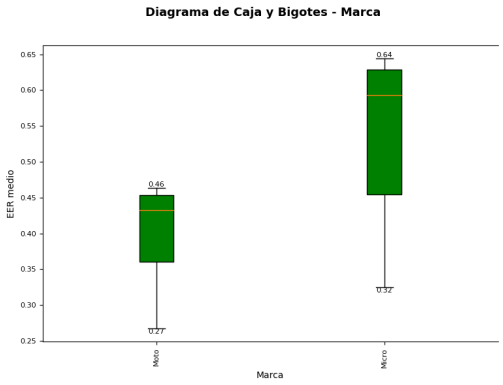

Figura 23: Diagrama Caja y Bigotes Marca

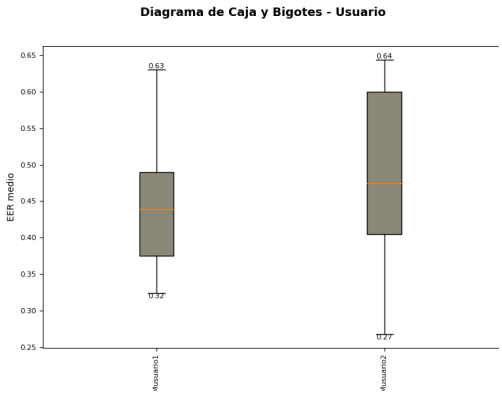

Figura 24: Diagrama Caja y Bigotes Usuario

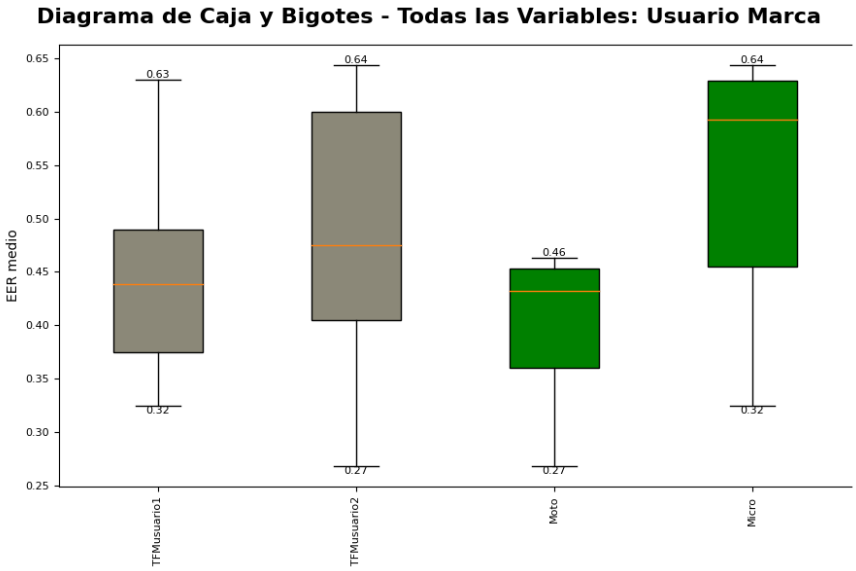

Figura 25: Diagrama Caja y bigotes Grupal

Además las gráficas no se generan en la propia ventana, sino que se están generando con ***matplotlib.pylot***, lo cual nos facilita múltiples opciones para interactuar con la gráfica, desde hacer zoom hasta cambiar parámetros concretos a nuestro antojo, e incluso, guárdala con formato png directamente

6. Construcción del Software

Antes de entrar en aspectos específicos de la **Construcción del Software**, voy a hablar de los requisitos que debe tener nuestra aplicación tanto para software. Una vez ya se hayan explicado, se expondrá el software utilizado y unos diagramas de secuencia simplificados para que se entienda todavía mejor como funciona la interacción entre el usuario y las distintas ventanas apoyandonos en los sprints descritos en la sección ***2.1***

6.1. Requisitos de Software

1. Tener instalado *Python*, la versión 3.9.12 o superior
2. Tener instalado *Keras*, la versión 2.12.0 o superior
3. Tener instalado *xmltodict* la versión 0.13.0 o superior
4. Tener instalado *Pandas* la versión 2.0.2 o superior
5. Tener instalado *Tenserflow* la versión 2.12.0 o superior
6. Tener instalado *MatPlotlib* la versión 2.12.0 o superior

6.2. Software utilizado

En esta parte hablaremos del software utilizado tanto para realizar el programa como para realizar la memoria.

- ***PyCharm community edition:*** Para desarrollar el programa, el IDE utilizado ha sido PyCharm, el cual está especificamente pensado para trabajar con Python y cualquiera de sus variantes
- ***Python y PythonQT6:*** A la hora de desarrollar esta aplicación, se ha utilizado Python y su variante PythonQT, la versión 6. Esto se debe a que gran parte de la funcionalidad ha sido integrada en python, mientras que la parte gráfica y de interfaces ha sido desarrollada en pythonQT
- ***HTML:*** Es un lenguaje de marcado, el cual se ha utilizado para dotar de un aspecto más profesional al log que se va mostrando sobre el flujo de la ejecución del programa.
- ***XML:*** Para la recogida de datos de las ejecuciones y su correspondiente procesado se ha utilizado un formato XML. Esto se debe a que facilitaba la laborar a la hora de hacer la representación gráfica y nos da un formato universal que cualquier persona podría comprender.
- ***QTDesigner:*** Es una herramienta de diseño visual, la cual nos permitía crear un esqueleto del aspecto que tendrían nuestras ventanas. También es posible dotarlas de funcionalidad en la propia herramienta, pero esto resulta mucho más engorroso con hacerlo desde el IDE. QTDesigner nos generar un archivo de tipo *.ui*, que nosotros podemos fácilmente convertir en un archivo python con la sentencia

pyuic6 nombreInterfaz.ui -o nombreDeseadoCodigo.py

- ***GitHub y GitLab:*** se ha utilizado tanto GitHub como GitLab para poder tener varias versiones del software almacenadas en la nube. Se han utilizado estos dos controles de versiones debido a que uno es personal del que hace el proyecto y el otro es el que ofrece de forma gratuita la universidad.
- ***Astah y Visual Paradigm:*** Son herramientas de modelado UML, cuya utilidad ha sido modelar los diagramas que se han utilizado para llevar a cabo el proyecto y redactar la memoria.
- ***Overleaf:*** Es un editor de documentos de tipo LaTeXcolaborativo cuyo almacenamiento se produce en la nube. La ventaja de utilizar Overleaf sobre cualquier otro editor es que nos permite generar el documento en tiempo real, esto implica que no necesitamos ejecutar sentencias para compilar nuestro resultado y ver los posibles cambios que se han hecho, sino que directamente la interfaz de este editor nos dota de dos ventanas, una donde vamos editando el texto y otra donde vemos el resultado del mismo cuando deseamos compilarlo. La utilidad de Overleaf en este proyecto ha sido la de redactar la memoria

6.3. Diagramas de secuencia

En esta sección se irán mostrando los diagramas de secuencia para las diferentes ventanas del programa, de esta manera podemos comprender mejor como funciona. Son diagramas de secuencia donde podemos contemplar, a un nivel simplificado, todas las opciones que puede hacer el usuario con las diferentes ventanas. Para ello nos apoyaremos en los sprints de la sección***2.1*** como se ha dicho anteriormente. Con todo esto lo que se quiere es intentar que, junto con la información recogida en la sección ***4***, se pueda comprender cuáles son todas las opciones que ofrece el programa y cómo acceder a ellas.

Para empezar, en la imagen ***26*** se puede contemplar un diagrama de secuencia sobre como funcionaría el flujo principal del programa. A lo largo de las siguientes subsecciones, haremos diagramas de secuencia de las ventanas de nuestro programa, a excepción de la ventana principal, ya que con el ***26***, tenemos una aproximación de como sería su diagrama de secuencia

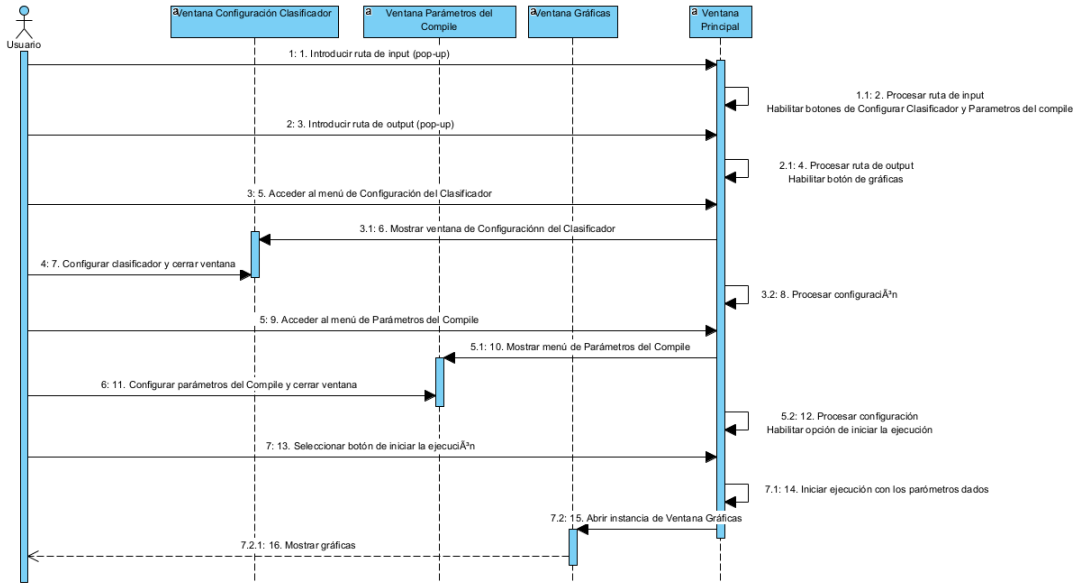

Figura 26: Diagrama de secuencia general simplificado

6.3.1. Diagrama de secuencia de la ventana Configuración de Clasificador

En la imagen ***27*** se puede ver como es el flujo de interacción del usuario con la ventana. Además de este diagrama, tendríamos que tener en cuenta que el usuario puede cerrar las ventanas en cualquier momento, lo cual haría que volviese a la ventana principal.

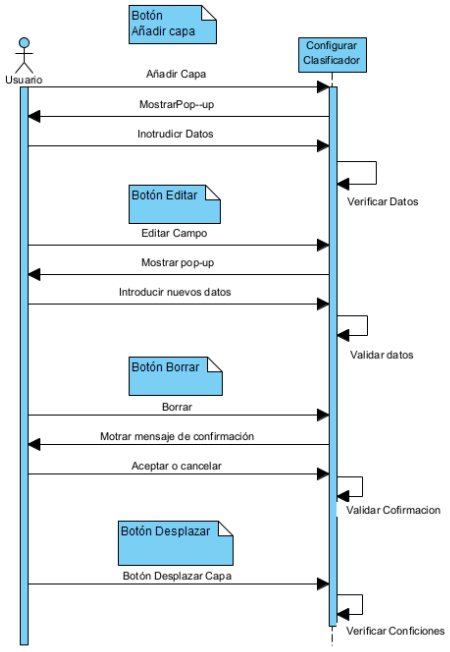

Figura 27: Diagrama de secuencia Configurar Clasificador

6.3.2. Diagrama de secuencia de la ventana de Parámetros del Compilador

En la imagen ***28*** podemos ver el diagrama de secuencia para los parámetros del compilador. Como puede verse es muy simple debido a que esta ventana no tiene botones ni ningún elemento que requiera validaciones más allá de las épocas,

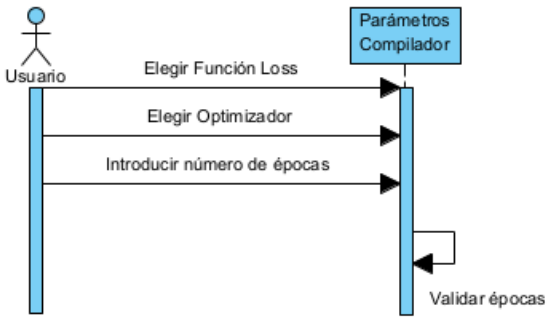

Figura 28: Diagrama de secuencia Parámetros del Compilador

6.3.3. Diagrama de secuencia de la ventana de Generar Gráficas

Por último podemos ver en ***29*** una imagen muy breve de como funcionaría la generación de gráficas en el diagrama de secuencia. Se ha hecho para una sola gráfica, debido a que indiferentemente de la gráfica que selecciones, todas funcionan de la misma manera

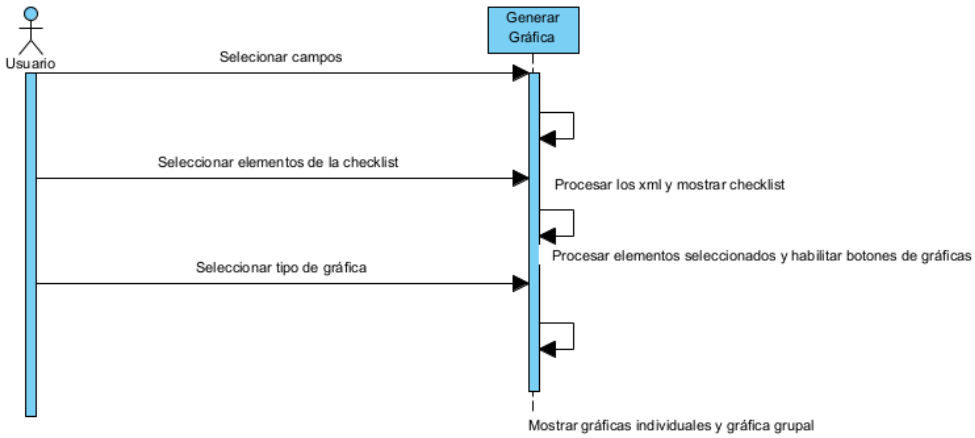

Figura 29: Diagrama de secuencia Gráficas

7. Pruebas

En esta sección se explicarán las diferentes pruebas que se han realizado para verificar que el programa funcione correctamente y como es su rendimiento.

7.1. Pruebas de ejecución

7.1.1. Ejecución pequeña

Al hablar de una *"ejecución pequeña"*, nos refieros a una ejecución con un clasificador que contenga únicamente 2 capas, (una oculta y la de salida) y 2 elementos diferentes como valor *NEP*. Además, de todo el conjunto de datos, no se iterará por todo el conjunto de usuarios (son 24) sino que se elegirán elegíamos 2 usuarios (los dos primeros) para llevar a cabo esta prueba. Respecto al *número de épocas*, la *función loss* y demás parámetros de configuración no entraré en detalle, ya que en ambas ejecuciones se usó la misma configuración.

La duración de esta ejecución oscila entre los 10 y los 12 minutos.

7.1.2. Ejecución grande

Sin embargo, cuando hablamos de la ejecución grande estamos hablando de una ejecución con 3 capas (2 ocultas y una de salida). Cada capa oculta contiene dos posibles valores para el parámetro *NEP*, y en este caso, no se restringirán los usuarios, sino que se iterará sobre los 24 que vienen en los datos de entrada.

La duración aproximada de esta ejecución fue de unas 34h.

7.2. Pruebas de funcionamiento

En este apartado se hablará de las distintas pruebas que se han utilizado para verificar el correcto funcionamiento del código. Estas han sido principalmente pruebas de caja blanca.

Cuando hablamos de pruebas de caja blanca, hablamos de pruebas que hacemos conociendo el código y verificando que el resultado es el adecuado. Para este programa no se ha hecho testing, pero se ha hecho un análisis de valores limites y de pruebas de control de flujo.

7.2.1. Pruebas de la ventana Configurar Clasificador

En la tabla ***22*** se pueden ver las diferentes pruebas que se han llevado a cabo para la ventana de configuración del clasificador y su resultado obtenido.

Indice	**Descripción**	**Resultado esperado**	**Resultado obtenido**
1	Desplazar una capa hacia arriba cuando es la primera	Aviso de error	Aviso de error
2	Desplazar una capa hacia abajo cuando es la última	Aviso de error	Aviso de error
3	Desplazar una capa hacia abajo cuando no hay capas	Aviso de error	Aviso de error
4	Añadir una capa con campo “Tipo” diferente a “Dense”	Aviso de error y petición de nuevo del parámetro	Aviso de error y petición de nuevo del parámetro
5	Añadir una capa con campo “Nep” un valor de tipo numero-letra-numero (ej 11aa11)	Aviso de error y petición de nuevo del parámetro	Aviso de error y petición de nuevo del parámetro
6	Añadir una capa con campo “Nep” un valor de tipo letra-numero-letra (ej aa11aa)	Aviso de error y petición de nuevo del parámetro	Aviso de error y petición de nuevo del parámetro
7	Añadir una capa con campo “Nep” un valor numérico como primer elemento y luego una letra (ej 11,aa)	Aviso de error y petición de nuevo del parámetro	Aviso de error y petición de nuevo del parámetro
8	Añadir en la función de activación un valor diferente a “sigmoid”, “relu” , “tanh”, “softmax”	Aviso de error y petición de nuevo del parámetro	Aviso de error y petición de nuevo del parámetro
9	Intentar editar un campo con las pruebas anteriores	Aviso de error y petición de nuevo del parámetro	Aviso de error

Tabla 22: Pruebas Configurar Clasificador

7.2.2. Pruebas de la ventana principal

En la tabla ***23*** podemos ver las pruebas de la ventana principal. No se han necesitado demasiadas, y esto se debe a que la mayoría de las validaciones se hacen en las propias ventanas.

Indice	**Descripción**	**Resultado esperado**	**Resultado obtenido**
1	Dar a "OK" sin introducir nada en las rutas	Log ruta no valida	Log ruta no valida
2	Ruta input donde no existe el fichero de datos	Log ruta sin fichero	Log ruta sin fichero
3	Botón de gráficas sin datos de los experimentos	Aviso de error	Aviso de error

Tabla 23: Pruebas Ventana Principal

7.2.3. Pruebas de Parámetros del Compile

En la tabla ***24*** podemos ver que en la lista hace referencia principalmente a la parte de las épocas. Esto se debe a que para prevenir errores en la "Función loss" y en el "Optimizer" se han añadido valores por defecto para intentar blindar al máximo esas partes y evitar errores imprevistos

Indice	**Descripción**	**Resultado esperado**	**Resultado obtenido**
1	Dar a "OK" sin introducir nada en las nada en las épocas	Error de número de épocas invalidas	Error de número de épocas invalidas
2	Dar a "OK" introduciendo épocas pero sin cambiar el resto de campos	Valores por defecto y épocas	Valores por defecto y épocas
3	Configurar la ventana y luego volver a abrirla y configurarla de nuevo	Se guarda la última versión	Se guarda la última versión
4	Introducir un "0" en las épocas	Error de número de épocas invalidas	Error de número de épocas invalidas
5	Introducir un número negativo en las épocas	Error de número de épocas invalidas	Error de número de épocas invalidas

Tabla 24: Pruebas de Parámetros del compile

7.2.4. Pruebas de Generar Gráficas

En la tabla ***25*** se puede ver las pruebas que se han hecho en para la ventana de "Generar Gráficas". Además de hacer pruebas con esa ventana, también se ha intentado experimentar con la ventana que muestra las propias gráficas, modificando los ejes, haciendo zoom, y viendo si había algún fallo en la forma .

Indice	**Descripción**	**Resultado esperado**	**Resultado obtenido**
1	Seleccionar 1 campo	Se activan los botones de gráficas	Se activan los botones de gráficas
2	Cuando ya hay un campo marcado, volver a abrir la checklist de seleccionar campos	El campo anteriormente marcado sigue marcado	El campo anteriormente marcado sigue marcado
3	Cuando ya hay un campo seleccionado, volver a abrir la ventana y desmarcar el campo	Se desactivan los botones	Se desactivan los botones
4	Seleccionar 3 campo y seleccionar cualquier tipo de gráfica	Se generan las gráficas individuales y la grupal	Se generan las gráficas individuales y la grupal
5	Guardar la gráfica generada	Se guarda correctamente como png	Se guarda correctamente como png
6	Cambiar los parámetros de de una gráfica generada	Se actualizan a los nuevos parámetros introducidos	Se actualizan a los nuevos parámetros introducidos
7	Utilizar el zoom en una gráfica generada y dar a volver a centrar	Se vuelve al punto medio de las gráficas	Se vuelve al punto medio de las gráficas

Tabla 25: Pruebas Generar Gráficas

8. Conclusiones

A nivel de técnico, se han cumplido todos los objetivos. La parte con la que más contento estoy es la parte de visualización de gráficas, ya que fue una parte de investigación bastante extensa, probando diferentes formas de representar los gráficos e intentar que fueran lo más legibles posible, hasta que finalmente conseguí hacerlo como está ahora. Además me parece que se ha conseguido que toda la interfaz sea muy simple e intuitiva, haciendo que cualquier persona que utilice la aplicación pueda saber manejarla sin ningún problema.

A nivel personal, me ha servido para profundizar en mi aprendizaje, tanto de python, como de aprendizaje automático. Del mismo modo he aprendido nuevas tecnologías como PythonQT, y he podido aprender y poner en práctica nuevas formas de resolver proyectos, como la **Metodología Agile**. Por otro parte, ha sido un proyecto bonito y satisfactorio de entender y completar. Al ser principalmente un apartado gráfico, puedes ver como lo que construyes va cobrando forma poco a poco, y eso me ha resultado altamente gratificante.

Por último concluir que ha sido una experiencia altamente enriquecedora. Embarcarte en un proyecto relativamente grande siempre es una aventura y no sabes que puede suceder, pero considero que se ha podido sortear todos los obstaculos que han ido apareciendo, lo cual también me hace estar orgulloso de mi trabajo.

9. Posibles mejoras

En este apartado se tratará de explicar las posibles mejoras que se podrían aplicar al programa de cara al futuro y que no han sido posibles de incluir en esta versión por falta de tiempo.

9.1. Redes Neuronales

Cómo se ha ido explicando a lo largo de la memoria, este programa se puede escalar para que sea más generalizado en cuanto a clasificadores y no sea únicamente para MLP. Para ello se van a listar las posibles mejoras y dónde podrías hacerse.

- ***Ventana Configurar Clasificador*** $\rightarrow$ En esta ventana es donde se debería cambiar la forma de crear el clasificador para hacerlo más general y que no sea únicamente para *MLP*. Para ello habría que tener en cuenta los siguientes cambios y validaciones

    1. ***Campo "Tipo"*** $\rightarrow$ Podríamos añadir 2 tipos nuevos con los que tendríamos que hacer validación en tiempo de ejecución (igual que se hace ahora para comprobar que el valor que se pone sea "Dense"). Estos tipos serían, entre otros, *"MaxPooling1D"* y *"Conv1D"* (ambas arquitectura de redes convolucionales). A continuación se explica para que se usa cada tipo y que implicaciones tendría.
        - ***Conv1D*** $\rightarrow$ Utilizada principalmente para extraer características relevantes en datos unidimensionales. La ventana debería habilitar 3 campos nuevos: **nº de filtros, los núcleos y el Padding**, y desactivaríamos el campo **Nep**.
        - ***MaxPooling1D*** $\rightarrow$ Se aplica para extraer características relevantes y reducir la longitud de la secuencia. En el caso de utilizar este tipo de capa, el clasificador debería desactivar el resto de campos para esa capa, ya que, por lo general, no se necesita concretar ninguno de los otros parámetros

La dificultad de esta parte es cambiar el primer formulario para que, no solo fuese dinámico en la creación de las capas, sino en la parte de los campos. Para ello sugiero crear una capa "base", de la que puedan partir todas las demás y que tenga todas las columnas. Al dar al botón de añadir, se validará el tipo de red que se está poniendo y en función de ello se creará o no el resto de la capa activada. Para ello también habría que añadir nuevas validaciones a los botones de "Añadir Capa" y "Editar item".

También se podría hacer la expansión del metamodelo de alguna otra forma. Al final el programa lo está realizando con un producto cartesiano, pero se podría haber pensado cualquier otro tipo de combinación que nos permitiera expandir el clasificador.

9.2. Parámetros de compilación de la red

De igual manera que se podía modificar la parte de tipos de redes neuronales, también se podrían incluir nuevos parámetros para la compilación de la misma.

Anexo

La descarga del proyecto puede hacerse en:`https://gitlab.inf.uva.es/manmend/TFG`

Además verificar que todas las URL de la bibliografía se han confirmado disponibles a día 10/07/2023

Referencias

[1] I. Salvador-Ortega, C. Vivaracho-Pascual, and A. Simon-Hurtado, "A successful study and effective system proposal for biometric user authentication by means of commercial wearables," *IEEE TRANSACTION ON MOBILE COMPUTING*.

[2] B. Hughes, *Software project management*. London: McGraw-Hill, 2009.

[3] P. Forogh, "Learn how to build graphical user interface (gui) applications with python, pyqt6 and qt designer." [Online]. Available: https://indra.udemy.com/course/python-gui-development-with-pyqt6/

[4] Codemy.com, "How to open a second window - pyqt5 gui thursdays 24," 2021. [Online]. Available: https://www.youtube.com/watch?v=R5N8TA0KFxc&ab_channel=Codemy.com

[5] Codemy.com, "Pass data between windows - pyqt5 gui thursdays 25," 2021. [Online]. Available: https://www.youtube.com/watch?v=R5N8TA0KFxc&ab_channel=Codemy.com

[6] keras.io, "Losses," 2023. [Online]. Available: https://keras.io/api/losses/

[7] keras.io, "Optimizers," 2023. [Online]. Available: https://keras.io/api/optimizers/

[8] keras.io, "Activacion," 2023. [Online]. Available: https://keras.io/api/layers/activations/

[9] keras.io, "Conv1d layer," 2023. [Online]. Available: https://keras.io/api/layers/convolution_layers/convolution1d/

[10] Runebook.dev, "tf.keras.layers.conv1d," 2023. [Online]. Available: https://runebook.dev/es/docs/tensorflow/keras/layers/conv1d

[11] sitiobigdata.com, "Redes neuronales de aprendizaje profundo," 2019. [Online]. Available: https://sitiobigdata.com/2019/07/08/redes-neuronales-aprendizaje-profundo/#

[12] keras.io, "Maxpooling1d layer," 2023. [Online]. Available: https://keras.io/api/layers/pooling_layers/max_pooling1d/

[13] Runebook.dev, "tf.keras.layers.maxpool1d," 2023. [Online]. Available: https://runebook.dev/es/docs/tensorflow/keras/layers/maxpool1d

[14] M. Sotaquirá, "Padding, strides, max-pooling y stacking en las redes convolucionales," 2019. [Online]. Available: https://www.codificandobits.com/blog/padding-strides-maxpooling-stacking-redes-convolucionales/