Universidad Internacional de La Rioja (UNIR)

ESIT

Máster Universitario en Inteligencia Artificial

[Sistema de predicción de resultados de análisis clínicos para pruebas rápidas]

Trabajo Fin de Máster

Presentado por: Vegas Diez, David

Director/a: Giuffra Palomino, Cecilia Estela

Ciudad: Bilbao
Fecha:02/02/2021

Resumen

En la actualidad se usan diferentes métodos para la detección de ciertas enfermedades. Dentro de todos estos métodos existen un grupo de pruebas que se denominan pruebas rápidas, las cuales tiene como fin la detección de algunas enfermedades o patologías en un corto periodo de tiempo, habitualmente no superior a 30 minutos. Existen varios métodos para la evaluación del resultado de las pruebas rápidas, siendo el análisis visual el más usado. El objetivo de la investigación realizada es la búsqueda de un nuevo método de predicción de resultados obtenidos mediante sistemas de pruebas rápidas, con el fin de mejorar la fiabilidad de los resultados, así como de agilizar su resolución. La metodología a seguir consta de la realización de 15000 test, los cuales aportarán los datos de las mediciones. Junto con el resultado de cada prueba se obtiene la evaluación por parte de especialistas de laboratorio, con experiencia en análisis de pruebas rápidas. Una vez obtenidos estos datos se procede al estudio y filtrado de los datos creando dos grupos. El primer grupo de entrenamiento sirve para entrenar el modelo de predicción y el segundo grupo de test se utiliza para la comprobación de la precisión del modelo. Con los datos de test se consigue un resultado con una precisión del 94.78%.

Palabras Clave: Prueba Rápida, Longitud de onda, Caracterización, Red de neuronas.

Abstract

Different methods are currently used to detect certain diseases. Within all these methods there is a group of tests called rapid tests, which are intended to detect some diseases or pathologies in a short period of time, usually not more than 30 minutes. There are several methods for evaluating the results of rapid tests, visual analysis being the most used. The objective of the research carried out is the search for a new method of predicting the results obtained through rapid test systems, in order to improve the reliability of the results, as well as speed up their resolution. The methodology to be followed consists of performing 15,000 tests, which will provide the measurement data. Together with the result of each test, the evaluation is obtained by laboratory specialists with experience in rapid test analysis. Once these data are obtained, the data is studied and filtered, creating two groups. The first training group is used to train the prediction model and the second test group is used to verify the precision of the model. With the test data, a result is achieved with a precision of 94.78%.

Keywords: Lateral Flow, Wavelength, Characterization, Neuron network.

Índice de contenidos

Índice de tablas

Índice de figuras

1 Introducción

La detección de ciertas enfermedades o patologías en sus primeras fases aporta grandes ventajas a la hora de poder aplicar tratamientos adecuados antes de posibles complicaciones por la enfermedad, mejorando de esta forma la esperanza de la obtención de un buen resultado. Con el fin de poder realizar una detección prematura se desarrollaron los test denominados pruebas rápidas. Estas pruebas rápidas suelen constar de una tira de papel en la cual se deposita un reactivo el cual reacciona al contacto con un fluido, habitualmente sangre o saliva del que se quiere hacer la prueba, dando su resultado mediante el cambio de la coloración de la tira de papel.

En la actualidad existen diferentes dispositivos capaces de realizar el análisis de las tiras de pruebas rápidas. El problema que tienen dichos dispositivos es, por un lado, su alto coste debido a las tecnologías necesarias de adquisición habitualmente basadas en cámaras tipo C.C.D y por otro lado su complejidad de manejo, por lo que este tipo de instrumentación se encuentra solo en los laboratorios especializados no cumpliendo con uno de los objetivos principales para los que se diseñaron las pruebas rápidas, que es dar un servicio a pie de consulta el cual agilice los resultados de ciertos diagnósticos.

Para solventar este problema se desarrolló un dispositivo de bajo coste y fácil manejo, capaz de adquirir los datos de las pruebas rápidas de una forma simple y fiable. Con la finalidad de reducir el coste y mejorar el acceso por parte del personal sanitario a dicho dispositivo, al cual no se le doto de inteligencia reduciendo su funcionalidad única y exclusivamente a la de hacer lecturas de las pruebas rápidas obteniendo datos de lectura los cuales se suben a un repositorio en la nube. Una vez que los datos se encuentran en la nube, se ejecutan los algoritmos necesarios para la predicción de los resultados.

Con el nuevo enfoque de disponer de un dispositivo de lectura de bajo coste el cual solo se dedique a realizar lecturas de pruebas rápidas se podrían conseguir dos hitos importantes. Primero la reducción de coste de cada dispositivo de lectura de las tiras rápidas, al poder hacer uso de tecnologías de lectura de bajo coste para el diseño de estos equipos de lectura sin inteligencia y, segundo, la posibilidad de tener una inteligencia centralizada que se puede aplicar a todos los datos independientemente de que dispositivo de lectura los lea mediante el volcado de datos en la nube de cada uno de los equipos de lectura, lo cual facilita mucho el mantenimiento del servicio y potencia su mejora continua.

1.1 Motivación

La motivación para la realización del proyecto vino dada por el problema planteado por parte del sector médico de atención primaria. El sector médico de atención primaria está compuesto por profesionales de la salud que practican la medicina general, siendo este sector el primero en ser visitado habitualmente cuando se necesita atención médica. Por tratarse de un sector de medicina general se plantea el uso de tiras de papel de pruebas rápidas, para detectar posibles enfermedades o patologías de una forma rápida y segura, pudiendo de esta forma derivar al paciente al especialista correspondiente. El problema planteado se da en la interpretación de los resultados de algunos tipos de pruebas rápidas. El proceso de uso de una prueba rápida se basa en la utilización de una tira de papel impregnada con uno o varios reactivos, en la cual se deposita una gota de fluido del paciente para analizar. A partir de eso, la tira va cambiando su color en función del resultado de la prueba.

El problema surge en la interpretación de los resultados de algunas de estas pruebas rápidas, ya que por un lado los reactivos no actúan igual debido a la diferencia de valores de los distintos componentes de la sangre de cada individuo, como pueden ser distintos niveles de triglicéridos o glóbulos rojos, dando así diferentes tonalidades del color del reactivo para individuos que tengan la misma patología. Por otro lado, el resultado tiene que ser interpretado en un corto espacio de tiempo, debido a que, una vez hecha la prueba el reactivo pierde sus propiedades pasados pocos minutos. Este tiempo depende del fabricante de la tira rápida, siendo los tiempos más comunes comprendidos entre 10 y 20 minutos.

Este último problema junto con el objetivo principal de la prueba rápida, que es el de brindar un resultado en periodo de tiempo corto, obliga que el resultado de la prueba tenga que ser evaluada por el ojo humano, introduciendo de esta forma la incertidumbre de la interpretación del color o tonalidad por cada individuo, haciendo que la evaluación del resultado sea variable en función de quien lo esté observando.

Por la imprecisión que este tipo de método genera, se vio la necesidad de realizar la lectura de resultados usando otros métodos que fuesen más fiables y de fácil acceso.

Este problema es relevante, ya que limita el uso de pruebas rápidas, las cuales permiten el diagnóstico de enfermedades de fácil detección de una forma rápida y precisa.

1.2 Planteamiento del trabajo

Para solucionar este problema se plantea la creación de un sistema que realice la interpretación de la lectura de los resultados de pruebas rápidas.

Para el desarrollo de este sistema se dispone de un dispositivo ya existente capaz de hacer lecturas de los resultados de pruebas rápidas, este dispositivo carece de inteligencia de interpretación de los datos leídos. El objetivo de este sistema es el de analizar e interpretar los datos leídos por el dispositivo de lectura, aportando la predicción del resultado.

En este trabajo se plantea el desarrollo de dicho sistema. Para ello primero se estudian las necesidades que tiene que cumplir la aplicación para satisfacer los objetivos de realizar predicciones de resultados de los datos obtenidos, de una forma rápida y fiable. A continuación, se analiza las necesidades que se requieren en el tratamiento y la realización de predicciones, para los datos provenientes de la lectura de las tiras de papel de pruebas rápidas. Una vez conocidas las necesidades para la realización del sistema, se plantea una solución con el fin de resolver el problema planteado, para posteriormente implementarla y ponerla en funcionamiento.

Una vez estudiado el problema se decide realizar una aplicación que esté enfocada al reconocimiento y clasificación de patrones ya que los resultados se pueden comparar con unos patrones estándar creados por los fabricantes de las tiras de papel, se decide usar para ello algoritmos de aprendizaje automático o deep learning. Se decide usar este tipo de algoritmos ya que los datos obtenidos pueden diferir ligeramente unos de otros siendo el resultado el mismo, por lo que el sistema tiene que ser capaz de identificar estas diferencias, aprenderlas e ir entrenándose para mejorar sus predicciones. Otra de las razones por la que se decide usar este tipo de algoritmos es por sus buenos resultados en el reconocimiento de patrones.

1.3 Estructura de la memoria

A continuación, se detalla la estructura del trabajo, a partir del capítulo 2.

En el capítulo del contexto y estado del arte, se expone como se realizan las lecturas de pruebas rápidas, y se hace un estudio del estado del arte con el fin de ver qué tecnologías o métodos se utilizan para resolver problemas similares.

En el apartado de objetivos y metodología se plantean los objetivos del sistema y se explican cuáles son los pasos a seguir, para realizar el desarrollo de la aplicación.

En el punto cuarto, identificación de requisitos, se estudia el problema más a fondo y se determinan cuáles son los requisitos necesarios para poder realizar el proyecto de desarrollo de la aplicación.

El punto quinto trata de la descripción del desarrollo de la aplicación realizada, tratando los puntos clave de este y haciendo hincapié en los algoritmos más destacados usados.

En el sexto punto, se evalúan los resultados obtenidos por el software desarrollado, comprobando si se cumple el objetivo marcado en los requisitos.

En el séptimo punto, se describen las conclusiones obtenidas de la evaluación de resultados y se plantean los trabajos futuros en la línea de investigación.

En el octavo punto se muestra la bibliografía consultada, seguida de los anexos del proyecto.

2 Contexto y estado del arte

El apartado del contexto y estado del arte se enfoca en estudiar los métodos usados para realizar un sistema capaz de realizar una predicción de los valores leídos por los dispositivos de lectura ópticos.

Primero se realiza una búsqueda con el fin de detectar cuales son los pasos que se deben seguir para el desarrollo de un proyecto con las características del que se quiere desarrollar.

Entre la bibliografía consultada se encuentran diferentes métodos como son los expuestos por Alonso y Calonge (2001), donde se propone realizar una primera etapa de adquisición de datos (realizada por el lector de tiras de papel para pruebas rápidas, fuera del alcance del estudio), una segunda etapa de extracción de características de los datos y una tercera etapa para el análisis de patrones para su evaluación y predicción.

2.1 Adquisición de datos

La adquisición de datos se basa en diferentes técnicas utilizadas para obtener datos de un entorno del cual se quiere conocer su estado, que viene representado por los datos adquiridos. La adquisición de los datos se realiza mediante un captador el cual es el encargado de realizar una transducción del dato medido y trasladarlo a un tipo de dato comprensible por el sistema que trabaja con los datos que se adquieren, como expone J.M Diaz (2008).
El amplio espectro de adquisición de datos recorre desde un método de toma de datos tan simple como puede ser tomar las respuestas de una encuesta de satisfacción con un producto, hasta la medida del ángulo de posicionamiento de un eje mediante un encoder incremental como muestran Rairán, D., & Fonseca, J. (2015).

Por tratarse de un campo de estudio tan amplio y diverso, se hace hincapié en las técnicas usadas para la medición de propiedades de la luz visible. Dentro de las propiedades de la luz se pueden destacar entre otras la refracción, la reflexión, la intensidad o la longitud de onda, siendo esta última la de mayor interés de estudio para el proyecto. Entre los diferentes métodos de medición de longitudes de onda de la luz, se usa una amplia gama de sensores, de los que se encuentran, entre otros, los fotodiodos tal como es explicado por Fontal B. (2005).

Los fotodiodos son semiconductores con unión tipo PN, los cuales generan una circulación de corriente cuando son excitados por la luz. Mediante la medición de esta corriente se puede realizar una lectura espectral obteniendo de esta forma valores de señal para diferentes longitudes de onda.

Las longitudes de onda de la luz pueden estar comprendidas en diferentes rangos del espectro, como la ultravioleta comprendida entre 320 y 400 nanómetros (nm), la cual no es visible para el ojo humano o las longitudes de ondas comprendidas entre 410 y 670 que representan los colores visibles por el ojo humano como muestra Fontal B. (2005).

2.1.1 Lectura de datos en tiras de papel de pruebas rápidas

Dentro de los métodos de adquisición de datos de tiras de papel de pruebas rápidas, se utilizan diferentes técnicas entre las que se encuentra la adquisición de imagen de la tira por medio de máquinas que usan sensores llamados charge-coupled-device (CCD), dispositivo el cual se encuentra habitualmente localizado como elemento captador de las cámaras fotográficas o microscopios de laboratorio como es la cámara de la marca Leica modelo DMC4500.

Su funcionamiento se basa en el uso de un número determinado de elementos captadores acoplados entre sí denominados pixeles, los cuales transforman la incidencia de los fotones sobre ellos en un potencial eléctrico determinado.

El número de captadores del que dispone un dispositivo CCD determina la cantidad de muestras de la imagen captada. Cada uno de los elementos captadores transfiere su carga eléctrica a un circuito electrónico encargado de transformar el valor de potencial eléctrico analógico en un valor digital para poder ser tratado a posteriori por un sistema de nivel superior como puede ser un microprocesador con un sistema operativo capaz de analizar los datos adquiridos.

Este tipo de dispositivos son de alta calidad y con un amplio rango de funcionalidades. Un ejemplo de este tipo de dispositivo puede ser el lector de tiras de orina BW-300 de la marca BIOWAY BIOLOGICAL TECHNOLOGY Co. Ltd, el cual cuenta según sus especificaciones, entre otras funcionalidades, con un sistema operativo embebido, trazabilidad de los resultados, archivos de datos fotografiados e inteligencia artificial de última generación con visión artificial. Debido a la tecnología que se usa y la complejidad de estos dispositivos, el precio de los equipos de estos métodos de lectura suele ser elevado.

Otro método usado para la lectura de las tiras de papel es el uso de fotodiodos enfocados a zonas específicas de la tira de papel de la prueba rápida con el fin de detectar las longitudes de onda de dicha zona y adquirir los valores correspondientes a las longitudes de ondas de la zona específica.

El principio de funcionamiento de este método es el mismo que el usado en los dispositivos CCD, con la diferencia que los datos adquiridos solo representan los valores de las zonas en las cuales están enfocados los fotodiodos.

Existen varios tipos de tiras de papel las cuales se pueden distinguir por las distintas zonas de la que se componen. Cada una de estas zonas de la tira de papel se encuentra dopada con un reactivo diferente, con el fin de obtener un resultado concreto para cada una a las patologías o enfermedades que se quieren detectar. El número de zonas de la tira define el número de fotodiodos necesarios para la lectura de cada una de las zonas de la tira de papel de la prueba rápida.

Como norma general la última de las zonas de las tiras de papel de las pruebas rápidas está dopada con un reactivo genérico el cual reacciona al contacto con la muestra de fluido del paciente al que se le está realizando la prueba rápida. El objetivo de esta última zona no es detectar ningún tipo de enfermedad o patología, su función está enfocada en comprobar el correcto funcionamiento de la prueba rápida, por lo que la reacción se debe de dar independientemente de si existe la enfermedad o patología a testear o no.

Esta zona se usa como método de control e indica que la muestra ha sido depositada correctamente y que las reacciones han sido correctas por lo que los datos se pueden tomar como buenos. El resto de las zonas están destinadas a reactivos los cuales son específicos de la prueba que se quiere realizar.

Los dispositivos encargados de adquirir datos usando el método de fotodiodos enfocados a zonas, hacen la lectura especifica de las diferentes zonas de la tira de papel de la prueba rápida, para ser tratados a posteriori a niveles superiores, como pueden ser microprocesadores, encargados de transformar los datos eléctricos provenientes de los fotodiodos en variables con valores que representan la lectura realizada.

De los diferentes métodos para la adquisición de datos el más usado para el análisis de las tiras de papel de pruebas rápidas es la evaluación por medio del ojo humano, haciendo uso para ello de la comparación de los valores observados en forma de colores con patrones definidos por los propios fabricantes de las tiras de papel, siendo en este caso el ojo humano el elemento encargado de adquirir el dato de la tira de papel, y el cerebro el sistema superior el cual trata el dato lo analiza y da una evaluación del resultado.

Se observa que independientemente del método utilizado para la adquisición de datos de la tira de papel de pruebas rápidas, ya sea mediante sistemas basados en CCD en fotodiodos o bien mediante el propio ojo humano, existe siempre un nivel superior encargado de dar una interpretación a los datos adquiridos.

2.2 Tratamiento de datos

El tratamiento de datos es la fase del proceso en la cual se utilizan técnicas para poder filtrar, organizar y normalizar los datos disponibles con el objetivo de conseguir un nuevo conjunto de datos los cuales se encuentren dentro de un marco de referencia conocida.

2.2.1 Filtrado de datos

Las técnicas de filtrado de datos se utilizan para la eliminación de datos erróneos debido a ruidos o distorsiones propias del entorno o de los sensores de captación. Dentro de las técnicas de filtrado de datos existen varios métodos que se pueden aplicar para filtrar los datos en función de la aplicación. En el campo de filtrado de imágenes uno de los métodos más usados es el filtrado de mediana de señales digitales (Baxes, 1994, Richards, 1995).

El método de filtrado de mediana se basa en ir recorriendo los datos adquiridos y remplazar cada dato con la mediana de los datos adyacentes, consiguiendo de esta forma la eliminación de ruidos como pueden ser el ruido sal y pimienta que cubre de forma dispersa toda una imagen con pixeles blancos y negros, como se ve expuesto por Peña-Peñate, Silva & Alcolea (2016).

Otro tipo de filtro aplicado al tratamiento de datos es la comparación con atributos de referencia. El método de comparación con atributos de referencia se basa en la comparación de los datos con unos valores de referencia como pueden ser limites máximo y mínimos o tipo de dato deseado.

2.2.2 Organización de los datos

La organización de datos es una técnica utilizada para la creación de estructuras lógicas que permitan la realización de operaciones como pueden ser la edición, guardado o actualización de los datos. En la actualidad los dos grandes grupos usados para la organización de datos se dividen en formatos estructurados o no estructurados tal como es expuesto por Camargo-Vega, Camargo-Ortega & Joyanes-Aguilar (2015).

Los formatos no estructurados están enfocados en el almacenamiento de gran variedad de información con diferentes tipos de temas o significados de los datos. Los formatos estructurados se enfocan en el almacenamiento de datos de información definida con tipos de datos estructurados y significados similares o relacionados.

2.2.3 Normalización de datos

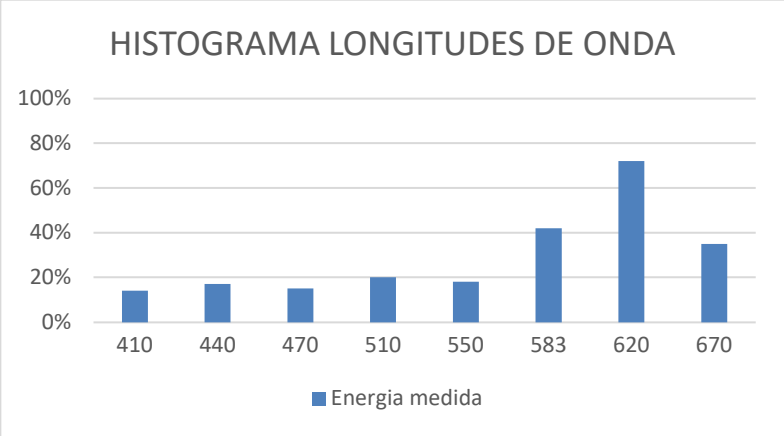
La normalización de datos es una técnica usada para dar formato a un conjunto de datos los cuales están contenidos bajo una norma dada. La norma de los datos es dependiente de los propios datos y del origen de los mismos, así bien una dirección de correo postal se puede normalizar como "País+Ciudad+Calle+Portal+Piso+Destinatario", un dato numérico se puede normalizar con un valor proporcional a su máximo y mínimo, esto indica que la normalización de datos es un campo extenso donde cada conjunto de datos debe ser estudiado para realizar una correcta normalización de los datos adquiridos (L. Al Shalabi and Z. Shaaban 2006).

2.3 Extracción de características

En la fase de extracción de características de datos, se tiene como objetivo el buscar y extraer características las cuales puedan servir para definir o identificar los datos adquiridos por una o varias cualidades propias de los datos. La extracción de características se realiza después del tratamiento de datos, para poder trabajar con los datos ya preparados.

En una primera fase, en la extracción de características se debe hacer un estudio de los datos, conocer su origen y entender su significado, para facilitar la comprensión de los datos de cada registro se suelen utilizar diferentes técnicas de representación de los datos como puede ser el uso de histograma tal como propone González (2005). Haciendo uso de esta herramienta se pueden entender mejor los datos, tal como se muestra en la figura 1, la cual representa los datos tomados por un fotodiodo como una señal discreta en frecuencias, representando la energía medida en cada de longitud de onda correspondiente.

Figura 1. Histograma longitudes de onda.



(Fuente: Datos de muestra del autor).

El vector de datos a evaluar en forma de histograma aporta una información directa por sí mismo de forma visual, pero para completar la información que estos datos aportan se aplican técnicas de estadística descriptiva con las cuales se crea una extracción de características de la señal formando de este modo un nuevo vector que contenga tanto los valores leídos como sus características descriptivas principales como se ve en Sabadías (1995).

Otro método utilizado para conseguir más características de los datos es el uso de la aproximación por mínimos cuadráticos como vemos en el artículo de Ferré (2004). Este método aporta una información adicional de los componentes de la señal, este método aporta buenos resultados cuando es utilizado como indicador de similitud entre patrones.

2.4 Reconocimiento de patrones

El análisis o reconocimiento de patrones, es la ciencia que estudia las características de elementos tanto físicos como abstractos, con el fin de establecer relaciones entre los elementos de un grupo.

En el entorno de la inteligencia artificial existen diversas técnicas o metodologías para realizar el reconocimiento de patrones. Entre las técnicas más utilizadas se encuentran la correlación cruzada, las máquinas de soporte vectorial, los algoritmos genéticos y las redes de neuronas, tal como se ve en el artículo de Gonzalez Gawron & Lage (2014).

El método de correlación cruzada se basa en el cálculo de una medida de similitud entre dos elementos, normalmente este método es usado para la búsqueda de subconjuntos de series de datos dentro de conjuntos de series más grandes. El grado de similitud viene dado por el análisis directo de la muestra de datos tomada y no por sus características. El método de correlación cruzada puede ser usado para el reconocimiento de patrones como se muestra en el artículo de Luna et al *(2013),* donde se muestra el uso de la correlación cruzada junto con el uso de un perceptrón multicapa para realizar reconocimiento del habla.

Las máquinas de soporte vectorial (SVM) son un conjunto de algoritmos enfocados en la clasificación y regresión, por sus características las SVM son utilizados en el reconocimiento de patrones, tal como se ve en el artículo de Ibargüren, Muñoz & Marín (2006). Este conjunto de algoritmos fue desarrollado por Vladimir Vapnik junto con su equipo. Su funcionamiento se basa en la creación de un hiperplano o conjunto de ellos dentro de un espacio de alta dimensionalidad, pudiendo ser hasta una dimensionalidad infinita. La SVM trata de separar de una forma óptima los puntos de un grupo o subgrupo pertenecientes a una clase en un hiperplano correspondiente a esa clase. Los modelos de SVM, se relacionan habitualmente con las redes de neuronas, ya que los modelos basados en SVM pueden ser usados como una función kernel, siendo este un método

utilizado para la clasificación de elementos polinomiales, como en el caso de reconocimiento de patrones, tomando un nuevo vector de datos y asignarle al hiperplano del patrón que le corresponde (S. V. M. Vishwanathan y M. Narasimha Murty 2002).

Los algoritmos genéticos (AG), se enmarcan dentro de los denominados algoritmos evolutivos. Son algoritmos inspirados en el mecanismo de la selección natural y la combinación genética en la reproducción de las especies. Siguiendo el método de supervivencia de las especies, los algoritmos genéticos usan estructuras de datos nucleares llamadas cromosomas, las cuales representan una o varias posibles soluciones a un problema concreto. Los cromosomas van evolucionando a través de iteraciones a las que se denominan generaciones. Para cada una de las generaciones, el algoritmo genético evalúa los cromosomas mediante el uso de medidas de aptitud, usan los diferentes operadores y combinaciones para preservar la información importante y transferírsela a la siguiente generación como explican Kuri-Morales, A. y Galaviz-Casas, J. (2002).

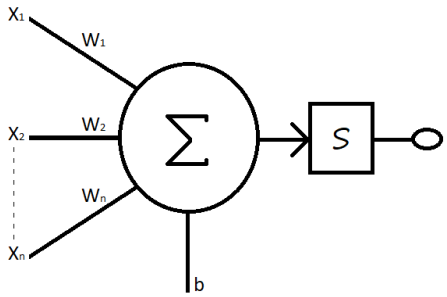
Dentro de los posibles campos de aplicación de los algoritmos genéticos, se encuentra el uso de estos para el reconocimiento de patrones como se ve en el artículo de Rodríguez, García & Puerta (2003), en el cual se plantea el uso de algoritmos genéticos para el reconocimiento de patrones en el habla.

Las redes de neuronas son modelos computacionales los cuales están basados en la observación del comportamiento de las neuronas biológicas de los seres vivos. Las redes de neuronas están compuestas por un conjunto de unidades denominadas neuronas artificiales, las cuales se encuentran conexionadas entre sí, con la capacidad de transmitirse señales. (R. López and J. Fernández, 2008).

Cada una de las neuronas artificiales de la red de neuronas es una unidad de cálculo, consistente en realizar un sumatorio ponderado de las entradas de la neurona artificial junto con un offset como valor umbral. Al resultado de dicho sumatorio se le aplica una función no lineal denominada función de activación (R. López and J. Fernández, 2008).

Una neurona artificial se puede definir de una forma gráfica tal como se muestra en la figura 2, donde los valores de entrada a la neurona artificial están representados por la letra X. Los pesos de ponderación se representan con la letra W, y el valor de offset está definido por la letra b. La función de activación aplicada en la neurona artificial se encuentra localizada en un paso posterior al sumatorio y es representada por la letra S.

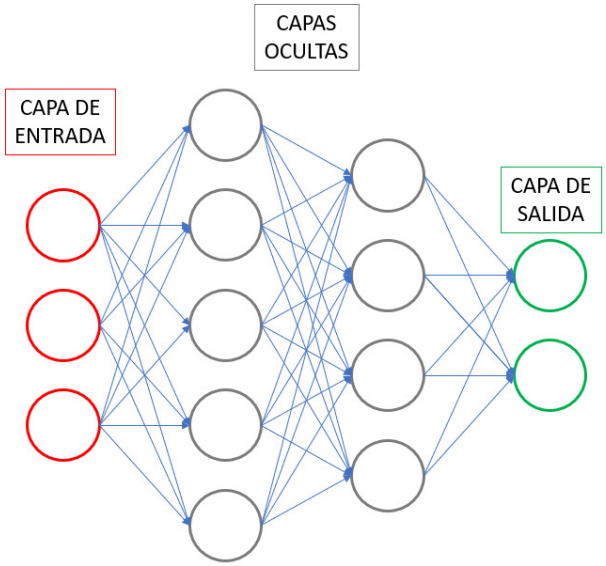
Figura 2. Neurona artificial.



(Fuente: Elaborado por el autor adaptado de Izurieta, F., Saavedra, C. (2006)).

Las redes de neuronas están compuestas por neuronas conectadas entre sí distribuidas en diferentes capas. La distribución de las capas sigue un orden típico donde existe una primera capa de entrada, compuesta por un numero de neuronas igual al número de datos del vector de entrada, a continuación se encuentran la capa o capas denominadas ocultas, siendo tanto el número de capas como el número de neuronas diferente en función del diseño de la propia red de neuronas, y por último se encuentra la capa de salida, con un numero de neuronas definido por el tipo de salida deseada para la función que este diseñada la red de neuronas (R. López and J. Fernández, 2008). En la figura 3 se muestra un ejemplo de una red de neuronas compuesta por una capa de entrada con 3 neuronas, dos capas ocultas con 5 y 4 neuronas respectivamente, y una última capa de salida compuesta por 2 neuronas.

Figura 3. Red de Neuronas Artificial.



(Fuente: Elaborado por el autor adaptado de Izurieta, F., Saavedra, C. (2006)).

Las redes de neuronas aportan una gran versatilidad de soluciones para la resolución de diferentes tipos de problemas tal como se ve en el artículo de Aldabas (2012). Debido a los buenos resultados que se están consiguiendo con las redes de neuronas estas son uno de los métodos más usados para la función de reconocimiento de patrones. Por sus buenos resultados y su potencia de resolución de problemas las redes de neuronas se encuentran en una línea de investigación bastante activa en los últimos años.

2.5 Conclusiones del estado del arte

Del estudio del estado del arte visto en este capítulo, se extrae como conclusión que no existe un único método para realizar una aplicación la cual sea capaz de predecir los resultados de la lectura de una tira de papel de prueba rápida.

Aunque en el estado del arte no se encuentre un método para el desarrollo de un sistema de lectura de tiras de papel de pruebas rápidas, sí que nos aporta una guía de los pasos que se pueden seguir para su diseño.

Tomando como punto de partida los conceptos vistos en los puntos 2.2, 2.3 y 2.4 de este capítulo, se pueden seguir unas guías de diseño para el desarrollo de un sistema para la predicción de tiras de papel de pruebas rápidas.

Como primer paso para la realización de un proyecto de reconocimiento de tiras de papel de pruebas rápidas, se puede aplicar un proceso de tratamiento de los datos tal como lo visto en el punto 2.2. Este primer paso debiera estar compuesto por una etapa de filtrado, una segunda etapa de organización de los datos y una última etapa de normalización.

Siguiendo lo visto en el estado del arte, se puede realizar un segundo paso de extracción de características, donde se pueden aplicar tanto métodos estadísticos como aproximación por mínimos cuadráticos para la extracción de características, creando de esta forma un vector patrón de características.

En el último paso se puede hacer uso de una red de neuronas tal como se ve en el punto 2.4 diseñada para realizar funciones de reconocimiento de patrones, aportando como salida la predicción de pertenencia de los datos de entrada a un grupo de categorías dados.

3 Objetivos y metodología de trabajo

En este capítulo se definen los objetivos que se buscan con este trabajo y que metodología se seguirá para el desarrollo del sistema.

3.1 Objetivo general

Desarrollar un sistema de predicción de las pruebas médicas rápidas.

3.2 Objetivos específicos

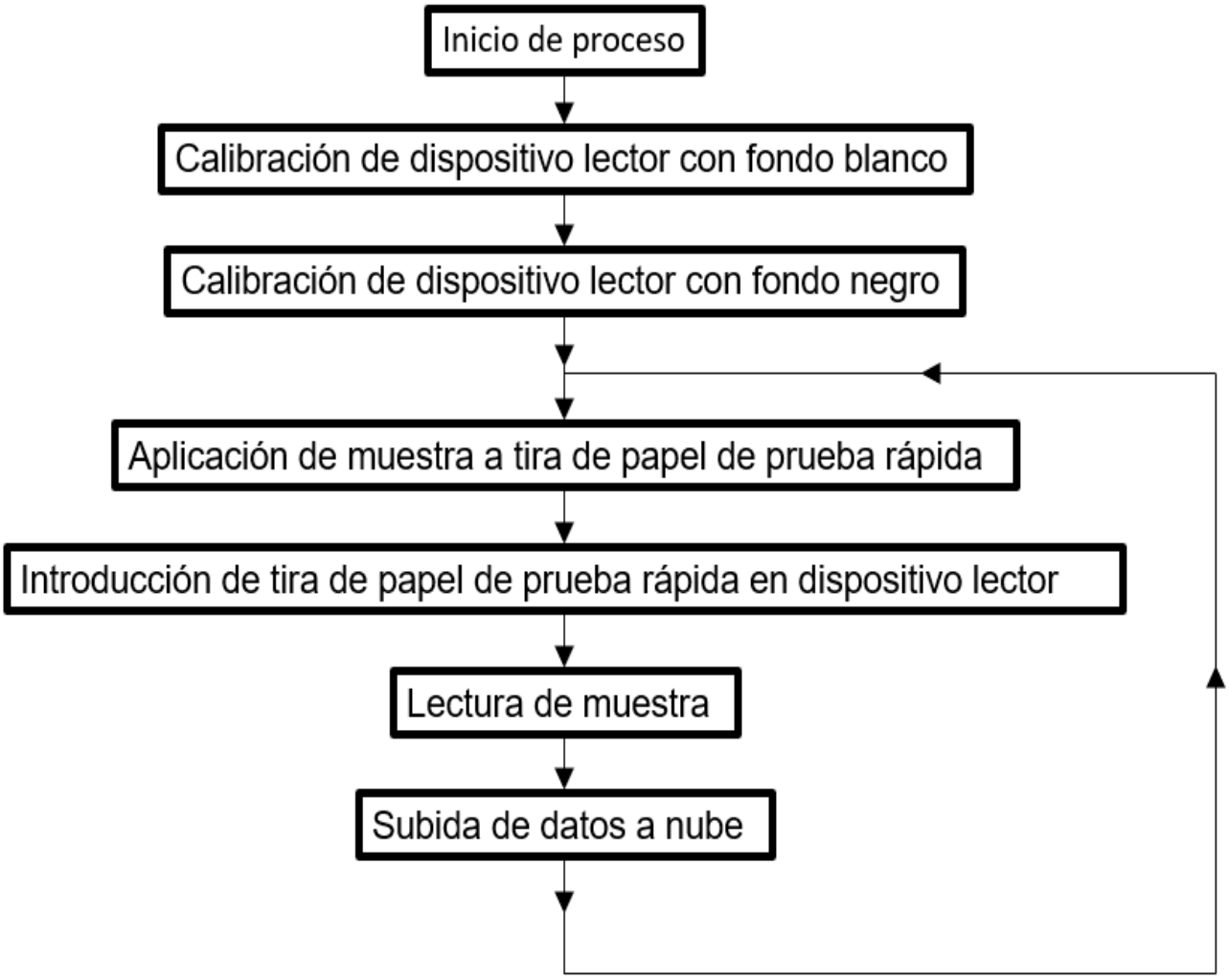
Como objetivos específicos para el desarrollo de la aplicación tenemos los siguientes puntos:

- Analizar que valores descriptivos caracterizan mejor los datos obtenidos al analizar pruebas rápidas.
- Determinar cuál de los algoritmos usados en redes de neuronas es más adecuado para la aplicación propuesta.
- Diseñar e implementar el algoritmo de predicción de resultados seleccionado.
- Evaluar el algoritmo de predicción con 3000 datos obtenidos de muestras conocidas.

3.3 Metodología del trabajo

La metodología a seguir parte de la obtención de muestras de pruebas rápidas realizadas en laboratorios con concentraciones de proteínas del virus SARS-Cov-2 conocidas y cuantificadas, las cuales son leídas por el dispositivo de lectura siguiendo el proceso que se muestra en la figura 4.

Figura 4. Flujograma proceso toma de muestra.



(Fuente: Elaborada por el autor).

Los datos adquiridos por los distintos dispositivos de lectura son subidos a un repositorio en la nube, al cual se puede acceder para leerlos. Estos datos subidos por los dispositivos vienen dados en un formato denominado en crudo (raw). El formato raw es un formato sin normalización ni filtrado, por lo que los datos en este formato deben ser tratados y filtrados para obtener un resultado fiable.

El primer paso que se debe dar es la realización de un tratamiento de los datos.

El tratamiento de datos consiste en realizar un filtrado y extracción de características, para conseguir una caracterización de la muestra adquirida. Con las características obtenidas de los datos se genera un vector patrón que represente un conjunto de propiedades características de la muestra leída, como se ve expuesto por Betancourt, Suárez & Franco (2004).

Los dispositivos encargados de generan el vector correspondiente a las lecturas de las tiras de papel de las pruebas rápidas, están basados en el uso de tecnologías de lectura mediante el uso de fotodiodos, lo cuales crean un vector de valores de las mediciones en las longitudes de onda de 410, 440, 470, 510, 550, 583 ,620, 670 nanómetros respectivamente, lo cual discretiza el espectro visible en ocho valores de longitudes de onda.

Estos datos se dan normalizados frente a unos valores de calibración y representan el porcentaje de los valores de la señal en cada una de las longitudes de ondas medidas. Esto facilita el filtrado descartando aquellos vectores que contengan algún valor fuera de la norma, ya que se considera una lectura no válida y no puede ser evaluada. Se hace uso de la normalización para realizar un primer filtrado descartando valores que no se encuentren dentro de los valores de normalización, ya que no debieran de existir.

Cada una de las muestras está ligada a un código identificativo único e invariable, el cual se sube a un servidor en la nube junto con los datos leídos para poder ser identificada de forma inequívoca.

Con el fin de obtener un conjunto de muestras controladas con las cuales poder trabajar con un método de alta fiabilidad, para la creación de cada una de las muestras se utiliza un valor conocido de proteína del elemento activador del reactivo de la tira de papel, estando los valores de proteínas de las muestras generadas dentro de un grupo categórico especifico de la muestra.

Se crean cinco categorías con un valor especifico de proteínas para cada categoría. Con las muestras creadas se procede a la lectura de cada una de ellas, para adquirir los datos de las longitudes de onda correspondientes a cada muestra.

Una vez adquiridos todos los datos de la muestra, se procede a hacer un análisis descriptivo en el que se determina que significa el valor de cada uno de los datos de la muestra.

Conociendo el significado de los datos de la muestra se procede a extraer las características propias de la muestra, con el fin de clasificar mediante un patrón el tipo de muestra generada.

Para la realización de esta caracterización se utilizan, por un lado, momentos estadísticos, que son formulaciones matemáticas capaces de caracterizar una muestra y, por otro lado, se calculan los parámetros de aproximación a dos patrones dados por el método de mínimos cuadrados.

Entre los momentos estadísticos utilizados para caracterización de datos, se utilizan los momentos estadísticos mostrados en la tabla 1, como punto de partida.

Tabla 1. Momentos estadísticos utilizados.

Momento estadístico	Ecuación
Media($\bar{X}$)	$\frac{\sum_{i=1}^{n} X_i}{n}$
Mediana(M_e)	$\frac{\left(\left({X_n}/{2}\right)+\left({X_{n+1}}/{2}\right)\right)}{2}$
Desviación Estándar(σ)	$\frac{\sqrt{\sum_{i=1}^{n}(\bar{X}-X_i)^2}}{n}$
Varianza(σ^2)	$\frac{\sum_{i=1}^{n}(\bar{X}-X_i)^2}{n}$
Asimetría (S_k)	$\frac{\frac{\sum_{i=1}^{n}(\bar{X}-X_i)^3}{n}}{\left[\frac{\sum_{i=1}^{n}(\bar{X}-X_i)^2}{n-1}\right]^{2/3}}$
Curtosis (K_t)	$\frac{\frac{\sum_{i=1}^{n}(\bar{X}-X_i)^4}{n}}{\left[\frac{\sum_{i=1}^{n}(\bar{X}-X_i)^2}{n-1}\right]^{2}}-3$

(Fuente: Elaborada por el autor).

El otro método de caracterización de los datos de entrada será el uso de los parámetros de aproximación por mínimos cuadrados y el error de los mismos, siendo estos parámetros calculados tal como se muestra en las siguientes formulas:

$$(1) \qquad \left(\vec{X}\right) = A^T * \vec{b} * [A^T * A]^{-1}$$

$$(2) \qquad (\varepsilon) = \frac{\sum_{i=0}^{n}\left(b_i-\left((W_i*X_0)+(C_i*X_1)\right)\right)^2}{n+1}$$

Donde:

- $\vec{b}$: Vector de valores de muestra leída
- X_0 : Primer componente de aproximación cuadrática
- X_1: Segundo componente de aproximación cuadrática
- $\overrightarrow{W}$: Vector de valores de calibración blanco.
- $\vec{C}$: Vector de valores leídos de zona de control.
- n : Numero de dato del vector de muestra leída.

Siendo utilizada la formula (1) para el cálculo del valor de los parámetros de aproximación por mínimos cuadrados, y la formula (2) como cálculo del error correspondiente a la aproximación realizada con los parámetros de aproximación calculada para los mínimos cuadrados.

Los datos característicos se calculan a partir de los datos de las lecturas de las tiras de papel de pruebas rápidas, los cuales vienen dados en un vector de nueve datos, siendo los tres primeros datos de identificación y categoría perteneciente de la muestra, y los seis últimos datos del vector los valores propios de lectura de la muestra correspondientes a las longitudes de onda propios de la muestra entregadas por el dispositivo de lectura de la tira de papel de la prueba rápida, tal como se ve en la tabla 2.

Tabla 2. Vector de datos de entrada.

ID	Sensor	Categoría Muestra	410 nm	440 nm	470 nm	510 nm	550 nm	583 nm	620 nm	670 nm

(Fuente: Elaborada por el autor).

Al vector de valores de entrada, se añaden los parámetros calculados para la caracterización de la muestra, creando de esta forma un vector intermedio con un formato como el que se muestra en la tabla 3.

Tabla 3. Vector de intermedios.

ID	Sensor	Categoría Muestra	410 nm	440 nm	470 nm	510 nm	550 nm	583 nm	620 nm	670 nm	$\bar{X}$	M_e	σ	σ^2	S_k	K_t	$\overrightarrow{X_0}$	$\overrightarrow{X_1}$	ε

(Fuente: Elaborada por el autor).

A partir del vector de datos intermedios creados, se diseña una red de neuronas para la predicción de resultados de los datos.

Esta red de neuronas tiene como entrada una parte del vector de datos intermedio visto en la tabla 3. Tomando como vector de entrada a la red de neuronas, un vector de datos con un formato como el mostrado en la tabla 4, el cual cuenta con los valores de longitud de onda medidos en la lectura de las tiras de papel de las pruebas rápidas y los parámetros de caracterización de estos valores propios de medida, lo cual define una capa de entrada de la red compuesta de diecisiete neuronas, este dato es importante ya que la red de neuronas debe tener una capa de entrada con un numero de neuronas igual al número de datos de los que está compuesto el vector de entrada.

Tabla 4. Vector entrada red de neuronas.

410 nm	440 nm	470 nm	510 nm	550 nm	583 nm	620 nm	670 nm	$\bar{X}$	M_e	σ	σ^2	S_k	K_t	$\overrightarrow{X_0}$	$\overrightarrow{X_1}$	ε

(Fuente: Elaborada por el autor).

Como capa de salida, la red de neuronas debe contar con una capa compuesta por cinco neuronas, una para cada una de las categorías que se han definido en la creación de las diferentes muestras con los distintos valores de proteínas de activador utilizadas.

En la tabla 5 se muestran las cinco categorías creadas y normalizadas frente al valor de proteínas usado para la generación de cada muestra.

Tabla 5. Categorías.

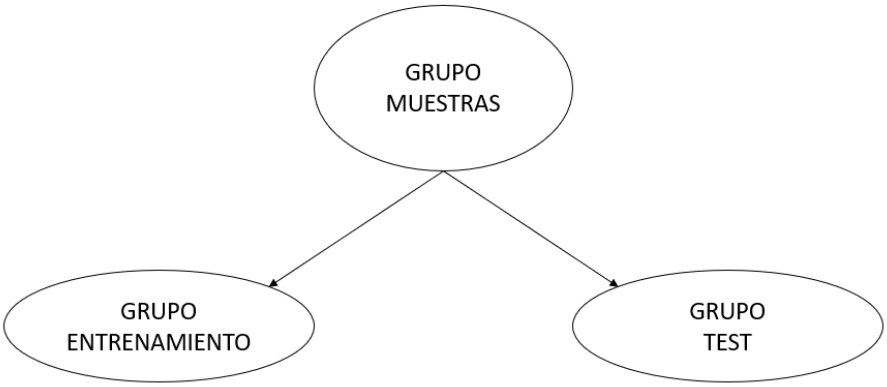
CATEGORIAS	Valor
Categoría 1	80% < C <= 100%
Categoría 2	60% < C <= 80%
Categoría 3	40% < C <= 60%
Categoría 4	20% < C <= 40%
Categoría 5	0% < C <= 20%

(Fuente: Elaborada por el autor).

Para la parametrización y comprobación de la red de neuronas se toman todas las muestras generadas y caracterizadas en el laboratorio, siendo divididas en dos subgrupos.

Un primer subgrupo llamado grupo de entrenamiento y un segundo subgrupo llamado grupo de test, tal como se muestra en la figura 5.

Figura 5. División de grupos.



(Fuente: Elaborado por el autor).

La división del número de muestras se realiza siguiendo el siguiente criterio:

- Grupo de entrenamiento → 80% de las muestras
- Grupo de test → 20% de las muestras

Siendo la selección de las muestras elegidas de forma aleatoria para cada grupo dentro de cada una de las categorías, asegurando de este modo una distribución uniforme de las muestras.

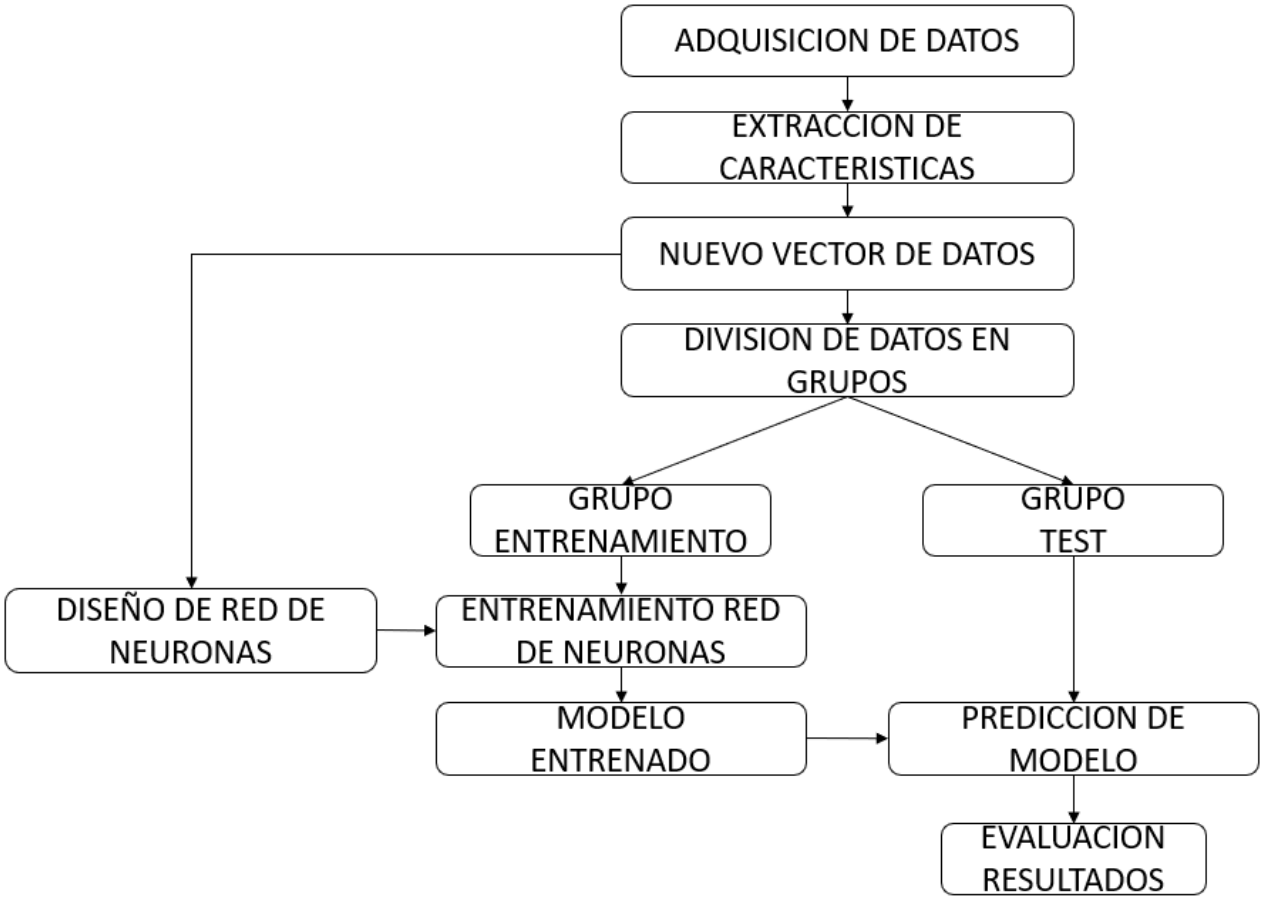
Se utiliza el primer grupo de entrenamiento para entrenar el modelo de la red de neuronas y hacer los ajustes de los parámetros de la red de neuronas.

El segundo grupo de test, se utiliza para hacer la predicción de los resultados obtenidos por el modelo de la red de neuronas entrenado.

La evaluación de los resultados se realiza comprobando el porcentaje de acierto en la predicción de los resultados de categorización de los datos del grupo de test aportados por el modelo de la red de neuronas.

La metodología descrita para realizar el proyecto queda resumida tal como se ve en el flujograma mostrado en la figura 6.

Figura 6. Flujograma de metodología.



(Fuente: Elaborado por el autor).

4 Identificación de requisitos

A continuación, se detallan los requisitos necesarios que se han especificado para el desarrollo del software de la aplicación para la clasificación y predicción de los resultados de las lecturas de las tiras de papel de pruebas rápidas.

4.1 Desarrollo

El algoritmo debe estar realizado en código abierto y correctamente comentado, con el fin de poder ser entendido y modificado en el futuro, para ello se deberá comentar de una forma clara y concisa. El lenguaje elegido para la creación del código es Python.4.2. Entorno de desarrollo

El entorno de desarrollo utilizado para el código de la aplicación es Jupyter/NoteBook.

4.2 Entorno de ejecución

La ejecución del código se realiza en una máquina virtual alojada en servidores externos pertenecientes a Amazon Web Services (AWS).

4.3 Acceso a datos

Los datos con la información de los test realizados se encuentran localizados en un servicio prestado por Amazon llamado Amazon Simple Services (Amazon S3) el cual presta un servicio de almacenamiento en la nube ofreciendo una API para la administración y acceso a los datos. Los datos se presentan en un archivo con formato CSV.

4.4 Variables de entrada

Los datos adquiridos se guardan en un vector de variables con un formato como el mostrado en la figura 7.

Figura 7. Formato de vector de lectura

ID	Sensor	Categoría Muestra	F1_XX	F2_XX	F3_XX	F4_XX	F5_XX	F6_XX	F7_XX	F8_XX

(Fuente: Elaborado por el autor).

Se pasa a detallar el significado y tipo de cada una de las variables del vector:

ID: Identificador de lectura, se trata de una variable de tipo alfanumérica. Su finalidad es la identificación única e inequívoca de la lectura de la prueba realizada. Cada lectura consta de un registro único, por lo que el identificador ID no se puede repetir y viene dado en el propio nombre del archivo.

Sensor: Identifica el sensor que ha realizado la lectura. Variable de tipo numérico, su valor está comprendido entre 1 y 3.

Categoría Muestra: Indica la categoría a la que pertenece la muestra. Variable de tipo numérico, con valores comprendidos entre 1 y 5.

F1_XX: Indica valor medido por el sensor para la longitud de onda de 410 nanómetros. Variable de tipo numérico, su valor está comprendido entre 0 y 65535.

F2_XX: Indica valor medido por el sensor para la longitud de onda de 440 nanómetros. Variable de tipo float32, su valor está comprendido entre 0 y 1.

F3_XX: Indica valor medido por el sensor para la longitud de onda de 470 nanómetros. Variable de tipo numérico, su valor está comprendido entre 0 y 65535.

F4_XX: Indica valor medido por el sensor para la longitud de onda de 510 nanómetros. Variable de tipo numérico, su valor está comprendido entre 0 y 65535.

F5_XX: Indica valor medido por el sensor para la longitud de onda de 550 nanómetros. Variable de tipo numérico, su valor está comprendido entre 0 y 65535.

F6_XX: Indica valor medido por el sensor para la longitud de onda de 583 nanómetros. Variable de tipo numérico, su valor está comprendido entre 0 y 65535.

F7_XX: Indica valor medido por el sensor para la longitud de onda de 620 nanómetros. Variable de tipo numérico, su valor está comprendido entre 0 y 65535.

F8_XX: Indica valor medido por el sensor para la longitud de onda de 670 nanómetros. Variable de tipo numérico, su valor está comprendido entre 0 y 65535.

4.5 Criterios de aceptación de datos

Para un correcto funcionamiento del sistema los datos tienen que cumplir con unos criterios de aceptación ya que de lo contrario puede tratarse de datos erróneos que conlleven a fallos en las predicciones. Para ello se deben de tomar como referencia los datos aportados por los valores de calibración de los fotodiodos de lectura, realizando este ajuste de calibración de los fotodiodos mediante el uso de dos muestras de calibración denominadas White y Drak, que cuentan con unos valores específicos en las diferentes longitudes de onda de lectura de los fotodiodos. El dato de resultado Result, viene dado por la normalización del dato de lectura de la tira de papel de la prueba rápida frente a los datos de calibración de los fotodiodos. El conjunto de datos de calibración y result tienen que estar comprendido entre los valores dados mostrados en la tabla 6.

Tabla 6. Criterios de aceptación de datos.

3000<	F1_White	<4000
32000<	F2_White	<40000
40000<	F3_White	<48000
20000<	F4_White	<24000
32000<	F5_White	<38000
37000<	F6_White	<41000
32000<	F7_White	<37000
22000<	F8_White	<27000

400<	F1_Dark	<1000
3000<	F2_Dark	<5000
6000<	F3_Dark	<10000
2500<	F4_Dark	<4500
4300<	F5_Dark	<7000
4500<	F6_Dark	<7500
4500<	F7_Dark	<8000
5000<	F8_Dark	<8000

0<	F1_Result	<120
0<	F2_Result	<120
0<	F3_Result	<120
0<	F4_Result	<120
0<	F5_Result	<120
0<	F6_Result	<120
0<	F7_Result	<120
0<	F8_Result	<120

(Fuente: Elaborado por el autor).

Debido a una mala calibración o un fallo en la lectura por parte del fotodiodo, se pueden generar datos de lectura que contenga fallos o erróneos, por lo que si los valores se encuentran fuera de estos rangos se deberá devolver un mensaje de error indicando que los datos no son validos.

4.6 Variables de salida

Como salida se entrega una matriz con el formato mostrado en la tabla 7:

Tabla 7. Matriz de salida.

CAT	TEST_1	TEST_2	TEST_3
CAT_1			
CAT_2			
CAT_3			
CAT_4			
CAT_5			

(Fuente: Elaborado por el autor).

La matriz se debe rellenar con el valor de porcentaje de cada una de las categorías predichas.

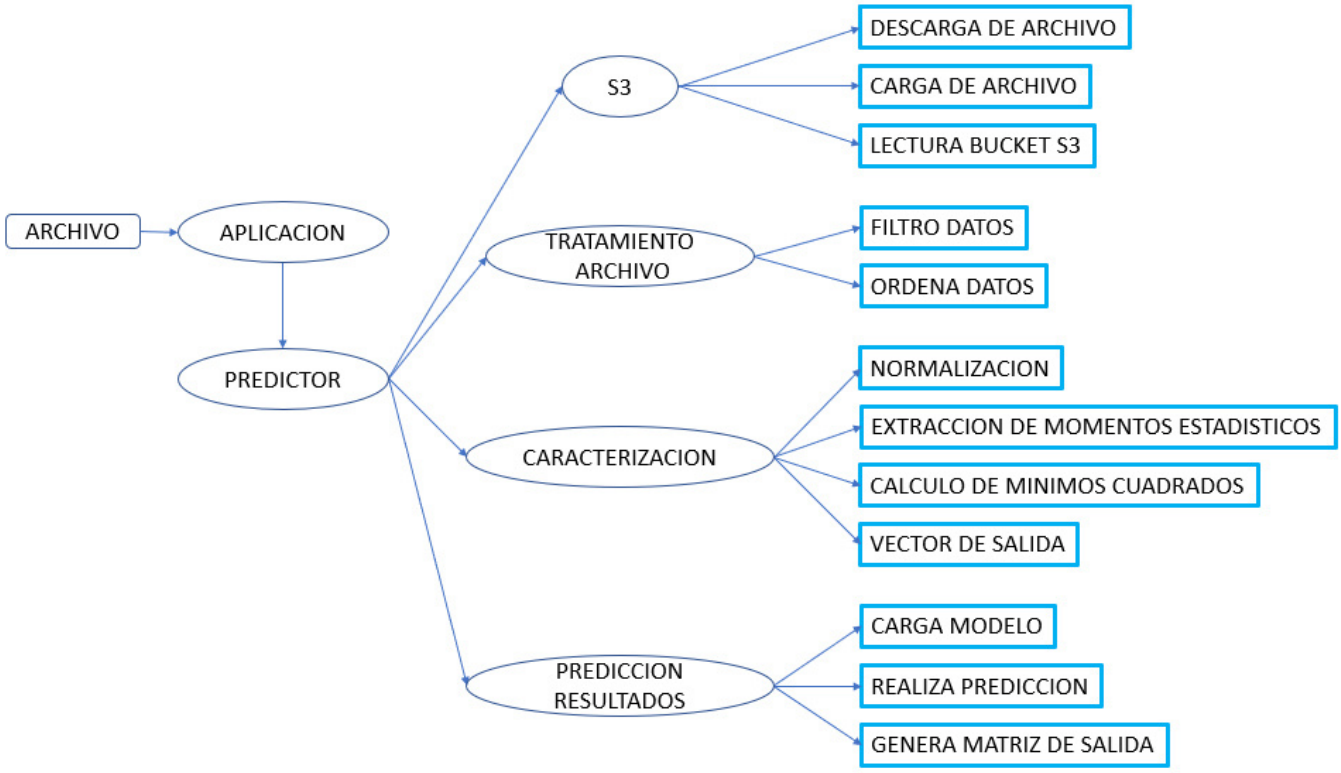
5 Descripción de la herramienta software desarrollada

El software desarrollado cumple con la función de realizar una predicción del resultado de una prueba rápida clasificándola en una de las cinco categorías posibles.

Para facilitar el desarrollo de la aplicación se ha dividido en seis partes, cada una con una funcionalidad y objetivo diferente, haciendo de esta forma más sencilla tanto su comprensión como posibles mantenimientos o evoluciones.

Con la finalidad de desarrollar un sistema lo más modular y estructurado posible, se crea un grupo de clases y funciones en Python las cuales se diseñan para realizar tareas concretas y repetitivas. La estructura seguida para la creación de las clases y funciones se muestra en la figura 8

Figura 8. Estructura de clases y funciones.



(Fuente: Elaborado por el autor).

5.1 Adquisición de datos

Para la adquisición de datos se dispone de dispositivos de lectura los cuales usan fotodiodos para la realización de las lecturas de las tiras de papel de las pruebas rápidas.

Los datos son leídos por cada uno de los distintos dispositivos de lectura y se envían a un repositorio denominado *bucket* (Directorio en S3) creado para la aplicación.

El motor de la base de datos de S3 se encarga de gestionar los nuevos archivos y su estado. El formato de los datos en el repositorio es en archivos tipo csv tal como se muestra en la figura 9.

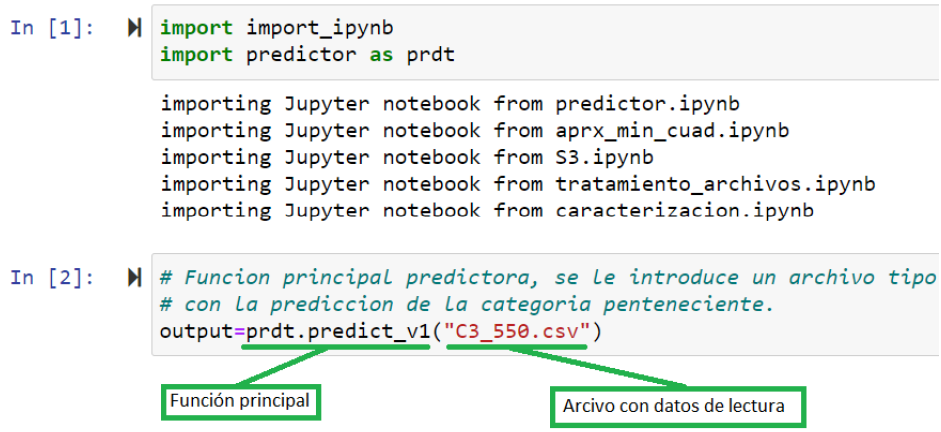
Figura 9. Archivo principal de muestra.

	A	B	C	D	E
1	Channel	Sensor1	Sensor2	Sensor3	
2	F1_WHITE	3500	3500	3500	
3	F2_WHITE	35000	35000	35000	
4	F3_WHITE	45000	45000	45000	
5	F4_WHITE	22000	22000	22000	
6	CLEAR_WHITE	27777	27777	27777	
7	NIR_WHITE	6678	6678	6678	
8	F5_WHITE	35000	35000	35000	
9	F6_WHITE	40000	40000	40000	
10	F7_WHITE	35000	35000	35000	
11	F8_WHITE	25000	25000	25000	
12	F1_DARK	600	786	750	
13	F2_DARK	4000	4061	4449	
14	F3_DARK	7000	7963	7415	
15	F4_DARK	3000	3600	3542	
16	CLEAR_DARK	27777	31458	30448	
17	NIR_DARK	3122	3740	3670	
18	F5_DARK	4600	5752	5650	
19	F6_DARK	5000	6105	5925	
20	F7_DARK	5000	6366	6079	
21	F8_DARK	6000	6786	6810	
22	F1_SAMPLE	2365	3199	3166	
23	F2_SAMPLE	23198	30000	33755	
24	F3_SAMPLE	28798	37769	34814	
25	F4_SAMPLE	14998	20257	21705	
26	CLEAR_SAMPLE	65535	65535	65535	
27	NIR_SAMPLE	6454	7182	7028	
28	F5_SAMPLE	24348	33711	35087	
29	F6_SAMPLE	30596	36022	36569	
30	F7_SAMPLE	28525	33877	33988	
31	F8_SAMPLE	20255	21974	22178	
32	F1_RESULT	88.5866417	89.8346687	87.1356721	
33	F2_RESULT	89.1020797	91.8462243	92.066974	
34	F3_RESULT	90.6088799	87.8904106	88.0903877	
35	F4_RESULT	90.7459288	93.2165939	92.3825811	
36	F5_RESULT	95.3548905	94.0608757	94.1399376	
37	F6_RESULT	93.312416	93.5402706	95.174183	
38	F7_RESULT	100.123003	98.017819	96.5244185	
39	F8_RESULT	94.2350271	97.8889498	97.6040644	
40	CLEAR_RESULT	100	100	100	
41	NIR_RESULT	93.700783	112.778511	113.985062	
42					

(Fuente: Elaborado por el autor).

Una vez que se tiene un archivo en el repositorio de S3, este realiza una llamada a la aplicación. En esta llamada, a la aplicación se le pasa como entrada de los datos, el archivo el cual se quiere calcular la predicción, por lo que la aplicación estará esperando a su entrada un archivo de tipo csv tal como se ve en la figura 10, donde se muestra la función principal de la aplicación con el archivo de datos del cual se quiere realizar la predicción de las categorías.

Figura 10. Método función principal.



(Fuente: Elaborado por el autor).

La función principal hace uso de la librería predictor, diseñada para la aplicación.

Esta librería contiene un método denominado predict_v1 mostrado en la figura 11, el cual es el encargado de gestionar toda la ejecución de la aplicación.

Figura 11. Método predict_v1.

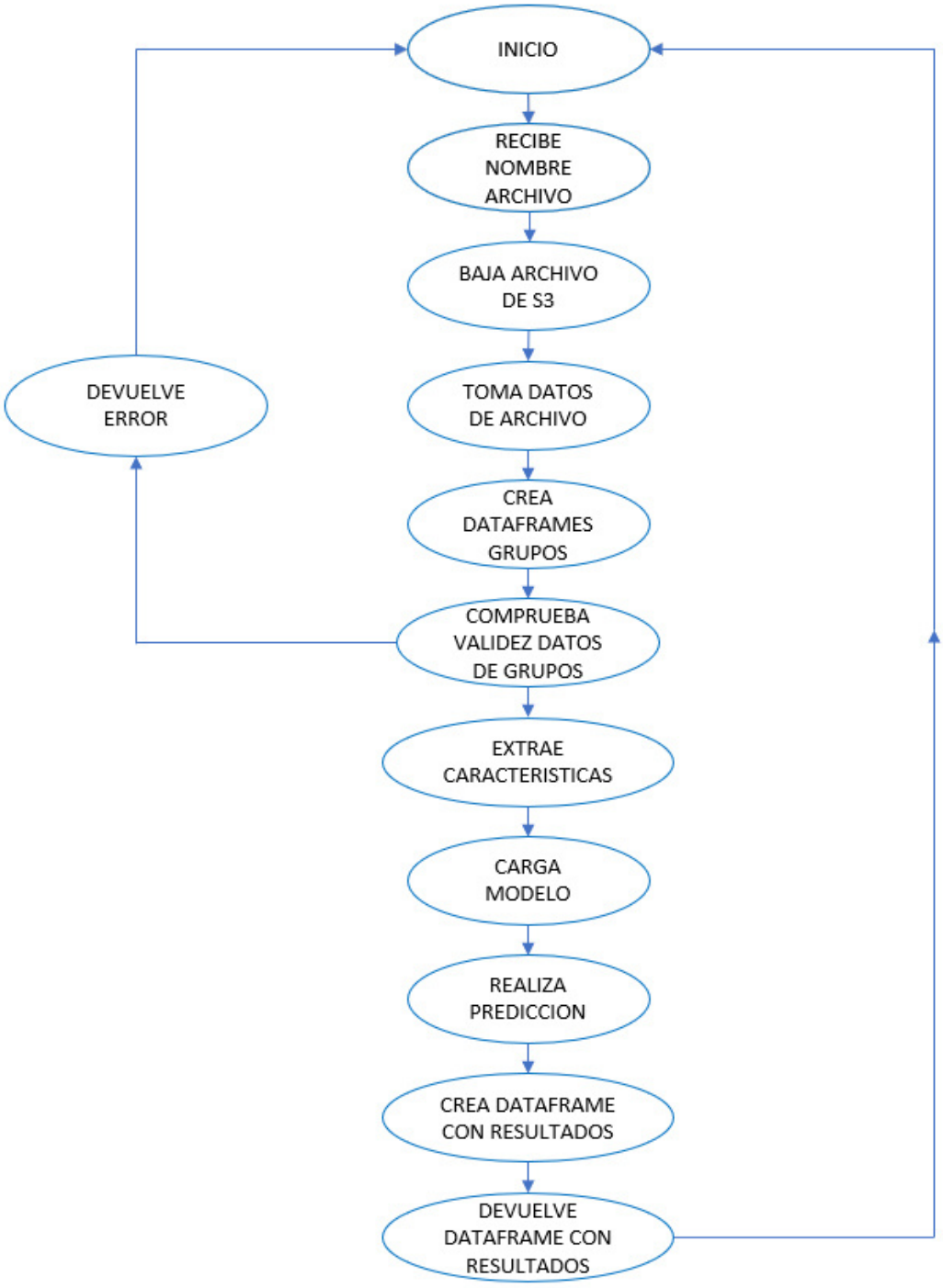
```
def predict_v1(file=None):
    if file==None: return "ERROR NO FILE"
    else:
        x_test=[]
        csv_objet=s3.S3(object_name=file)#Creo un objeto de S3 que puede bajar arcivos
        file_filter=ta.select_data(csv_objet.download_file(object_name=file)) #Creo objeto para tratar el archivo
        W,D,R=file_filter.trata_file() #Tomo los datos que nos interesan
        if file_filter.filtra(): # Compruebo que los datos sean validos
            out_vect=ca.caracteriza(R) # Se crea el vector con los datos de caracterizacion
            new_model = tf.keras.models.load_model('model_v1') # Se carga el modelo creado y entrenado
            x_test.append(out_vect["Sensor1"].values) # Se añaden los valores representantes del valor del sensor 1
            x_test.append(out_vect["Sensor2"].values) # Se añaden los valores representantes del valor del sensor 2
            x_test.append(out_vect["Sensor3"].values) # Se añaden los valores representantes del valor del sensor 3
            x_test=np.array(x_test) # Se convierten los valores en un array de numpy
            predict = new_model.predict(x_test) # Se hace la prediccion del modelo
            predict_T=np.transpose(predict) # Se trasponen los resultados
            output=pd.DataFrame(predict_T,index=categorias,columns=titulos) # Se crea el dataframe de salida
            return output # Retornamos el dataframe de salida
        else: return "ERROR" # Retornamos mensaje de error
```

(Fuente: Elaborado por el autor).

Este método se comporta como un sistema operativo de bucle infinito, el cual ira llamando a diferentes funciones o clases con el fin de devolver un resultado del archivo de datos de lectura de

las tiras de papel de pruebas rápidas con el que se está trabajando, siguiendo el flujograma mostrado en la figura 12.

Figura 12. Flujograma método predict_v1.



(Fuente: Elaborado por el autor).

5.2 Tratamiento de datos

Una vez adquirido el archivo con los datos de lectura de las tiras de papel de las pruebas rápidas correspondiente a uno de los dispositivos de lectura, se comienza a trabajar con dichos datos para darles el formato adecuado y facilitar su manejo.

Para realizar esta tarea, el método predict_v1() hace uso de una clase denomina select_data(), mostrada en la figura 13.

Figura 13. Función select_data.

```
class select_data:
    def __init__(self,file):
        self.file=file
        self.w=[]
        self.d=[]
        self.r=[]
```

(Fuente: Elaborado por el autor).

El primer paso para trabajar con esta clase es inicializarla, para ello es necesario crear un objeto de la clase pasando como argumento de entrada el archivo de datos con el que se quiere trabajar.

Una vez inicializado el objeto con el archivo de datos correspondiente, se comienza a hacer uso de dos los dos métodos heredados, necesarios para realizar el tratamiento de los datos del archivo leído. El primero de los métodos heredados por el objeto tipo select_data se llama trata_file() mostrada en figura 14.

Figura 14. Metodo trata_file.

```
def trata_file(self,file=None):
    if file == None: file=self.file
    dt_file=pd.read_csv(file) # Leo archivo de lista
    filas=dt_file['Channel'].values
    for x,y in enumerate(filas):
        filas[x]=y.upper()
    dt_file=dt_file.drop('Channel',axis=1) # le quito el canal ya que es el nombre de columnas
    columnas=dt_file.columns
    output_file=pd.DataFrame(dt_file.values,index=filas,columns=columnas)
    output_file=output_file.drop(lista_quitar,axis=0)
    self.w=output_file.loc[lista_white,:]
    self.d=output_file.loc[lista_dark,:]
    self.r=output_file.loc[lista_result,:]
    return self.w,self.d,self.r,
```

(Fuente: Elaborado por el autor).

A este método se le pasa el nombre del archivo que se quiere tratar y el método comienza a realizar los pasos necesarios para realizar el tratamiento del archivo.

El primer paso que ejecuta el método trata_file() es la creación de un DataFrame de pandas, librería usada para facilitar el trabajo con datos en formatos matriciales, manteniendo en una primera instancia la misma disposición y arquitectura de los datos originales que se encuentran en el archivo de lectura tal como se muestra en la figura 15.

Figura 15. Dataframe con datos originales.

	Sensor1	Sensor2	Sensor3
F1_WHITE	626.000000	806.000000	756.0
F2_WHITE	4245.000000	4997.000000	5441.0
F3_WHITE	7371.000000	9023.000000	8286.0
F4_WHITE	3571.000000	4247.000000	4189.0
CLEAR_WHITE	32256.000000	34253.000000	32593.0
NIR_WHITE	3078.000000	3386.000000	3224.0
F5_WHITE	5595.000000	6902.000000	6739.0
F6_WHITE	6367.000000	7612.000000	7335.0
F7_WHITE	6094.000000	7438.000000	7056.0
F8_WHITE	6441.000000	7227.000000	7185.0
F1_DARK	618.000000	820.000000	767.0
F2_DARK	4218.000000	4990.000000	5507.0
F3_DARK	7221.000000	9132.000000	8443.0
F4_DARK	3521.000000	4226.000000	4217.0
CLEAR_DARK	32087.000000	34724.000000	32862.0
NIR_DARK	3050.000000	3448.000000	3248.0
F5_DARK	5599.000000	6958.000000	6880.0
F6_DARK	6329.000000	7602.000000	7402.0
F7_DARK	6057.000000	7512.000000	7210.0
F8_DARK	6363.000000	7226.000000	7213.0
F1_SAMPLE	626.000000	821.000000	768.0
F2_SAMPLE	4264.000000	4992.000000	5515.0
F3_SAMPLE	7284.000000	9137.000000	8447.0
F4_SAMPLE	3575.000000	4226.000000	4218.0
CLEAR_SAMPLE	32569.000000	34737.000000	32882.0
NIR_SAMPLE	3114.000000	3450.000000	3248.0
F5_SAMPLE	5601.000000	6961.000000	6884.0
F6_SAMPLE	6331.000000	7606.000000	7408.0
F7_SAMPLE	6057.000000	7518.000000	7214.0
F8_SAMPLE	6368.000000	7227.000000	7213.0
F1_RESULT	100.000000	100.000000	100.0
F2_RESULT	170.370370	28.571429	800.0
F3_RESULT	42.000000	500.000000	400.0
F4_RESULT	108.000000	0.000000	100.0
CLEAR_RESULT	285.207101	1300.000000	2000.0
NIR_RESULT	228.571429	200.000000	0.0
F5_RESULT	200.000000	300.000000	400.0
F6_RESULT	5.263158	40.000000	600.0
F7_RESULT	0.000000	600.000000	400.0
F8_RESULT	6.410256	100.000000	0.0

(Fuente: Elaborado por el autor).

Se realiza esta operación de conversión de formato con el fin de facilitar el trabajo y manipulación de los datos, pudiendo hacer uso, de esta forma, de la potencia de tratamiento de datos que aporta la librería Pandas. En este primer paso, los datos no son manipulados ni tratados, simplemente son trasladados del archivo csv al formato del dataframe de pandas.

Se toma como nombre de las columnas los valores correspondientes a la primera fila del archivo de datos, y como nombre para las filas se toma el valor contenido en la primera columna del archivo de datos. Una vez creado el dataframe se comienza a trabajar con él, para ir dando forma al vector de datos con el que trabaja el resto de las partes de la aplicación.

Para la selección de los datos que componen el vector de trabajo, primero se eliminan valores del dataframe que han sido trasladados desde el archivo original y que no aportan información relacionada para nuestra aplicación, como pueden ser datos adquiridos para longitudes de onda de valor infrarrojo, o aportan una información redundante, como son los valores leídos sin normalizar.

Tanto los registros que comienzan con la palabra “CLEAR”, como los que empiezan por la palabra “NIR”, son registros provenientes de longitudes de onda fuera del espectro de la luz visible, por lo que la información que aportan para la lectura de las tiras de papel de las pruebas rápidas no es válida. Los registros que finalizan la palabra “SAMPLE”, son registros redundantes. Se tratan de registros que contienen el valor de lectura en crudo y no se encuentran normalizados contra el fondo de escala de la calibración White y Dark.

Su valor es normalizado por el propio dispositivo lector, devolviendo como resultado el registro denominado “RESULT”. Para realizar esta normalización el dispositivo lector aplica la fórmula de normalización tal como se muestra en la figura 16.

Figura 16. Normalización de valor de lectura.

$$\text{Result} = \frac{Sample - Dark}{White - Dark} \times 100$$

(Fuente: Elaborado por el autor).

Donde:

- Sample: Es el valor leído por el sensor para la longitud de onda correspondiente
- Dark: Es el valor leído por el sensor para la calibración dark para la longitud de onda correspondiente

- White: Es el valor leído por el sensor para la calibración white para la longitud de onda correspondiente

Con el fin de eliminar los registros no relevantes o innecesarios para nuestra aplicación, se crea una lista de eliminación de registros, tal como se muestra en la figura 17.

Figura 17. Lista de registros a eliminar.

```
lista_quitar=['CLEAR_WHITE','NIR_WHITE','CLEAR_RESULT','NIR_RESULT',
              'NIR_DARK','CLEAR_DARK','NIR_SAMPLE','CLEAR_SAMPLE',
              'F1_SAMPLE','F2_SAMPLE','F3_SAMPLE','F4_SAMPLE',
              'F5_SAMPLE','F6_SAMPLE','F7_SAMPLE','F8_SAMPLE']
```

(Fuente: Elaborado por el autor).

El método trata_file() hace uso de esta lista y la aplica al dataframe original, eliminado los registros que se encuentran en la lista, creando de esta forma un nuevo dataframe, como el que se muestra en la figura 18, el cual solo contiene los registros con los que trabaja la aplicación.

Figura 18. Vector de datos filtrados.

	Sensor1	Sensor2	Sensor3
F1_WHITE	3500.0	3500.0	3500.0
F2_WHITE	35000.0	35000.0	35000.0
F3_WHITE	45000.0	45000.0	45000.0
F4_WHITE	22000.0	22000.0	22000.0
F5_WHITE	35000.0	35000.0	35000.0
F6_WHITE	40000.0	40000.0	40000.0
F7_WHITE	35000.0	35000.0	35000.0
F8_WHITE	25000.0	25000.0	25000.0
F1_DARK	600.0	786.0	750.0
F2_DARK	4000.0	4061.0	4449.0
F3_DARK	7000.0	7963.0	7415.0
F4_DARK	3000.0	3600.0	3542.0
F5_DARK	4600.0	5752.0	5650.0
F6_DARK	5000.0	6105.0	5925.0
F7_DARK	5000.0	6366.0	6079.0
F8_DARK	6000.0	6786.0	6810.0
F1_RESULT	97.0	97.0	97.0
F2_RESULT	97.0	97.0	97.0
F3_RESULT	96.0	96.0	96.0
F4_RESULT	97.0	97.0	97.0
F5_RESULT	96.0	96.0	96.0
F6_RESULT	97.0	97.0	97.0
F7_RESULT	96.0	96.0	96.0
F8_RESULT	97.0	97.0	97.0

(Fuente: Elaborado por el autor).

Una vez eliminados del dataframe los datos sin información relevante o redundante, se dividen los datos resultantes en tres grupos. Cada uno de estos grupos contienen los valores correspondientes a la lectura realizada. Para ello se nombran los grupos según su origen:

- Grupo *White*, correspondiente de F1_White a F8_White.
- Grupo *Dark*, correspondiente de F1_Dark a F8_Dark.
- Grupo *Result*, correspondiente de F1_Result a F8_Result.

En cada grupo se toman los valores correspondientes a los tres sensores. El método trata_file() crea un *dataframe* para cada uno de los grupos como se muestra en la figura 19:

Figura 19. Grupos de dataframes.

	Sensor1	Sensor2	Sensor3
F1_WHITE	3500.0	3500.0	3500.0
F2_WHITE	35000.0	35000.0	35000.0
F3_WHITE	45000.0	45000.0	45000.0
F4_WHITE	22000.0	22000.0	22000.0
F5_WHITE	35000.0	35000.0	35000.0
F6_WHITE	40000.0	40000.0	40000.0
F7_WHITE	35000.0	35000.0	35000.0
F8_WHITE	25000.0	25000.0	25000.0

	Sensor1	Sensor2	Sensor3
F1_DARK	600.0	786.0	750.0
F2_DARK	4000.0	4061.0	4449.0
F3_DARK	7000.0	7963.0	7415.0
F4_DARK	3000.0	3600.0	3542.0
F5_DARK	4600.0	5752.0	5650.0
F6_DARK	5000.0	6105.0	5925.0
F7_DARK	5000.0	6366.0	6079.0
F8_DARK	6000.0	6786.0	6810.0

	Sensor1	Sensor2	Sensor3
F1_RESULT	97.0	97.0	97.0
F2_RESULT	97.0	97.0	97.0
F3_RESULT	96.0	96.0	96.0
F4_RESULT	97.0	97.0	97.0
F5_RESULT	96.0	96.0	96.0
F6_RESULT	97.0	97.0	97.0
F7_RESULT	96.0	96.0	96.0
F8_RESULT	97.0	97.0	97.0

(Fuente: Elaborado por el autor).

Los dataframe creados para cada uno de los tres diferentes grupos, son guardados como variables internas propias de la clase select_data(), por lo que los métodos o funciones pertenecientes a esta clase pueden hacer uso de ellos después de ser generados.

El segundo método que ha heredado el objeto creado con la clase select_data(), se llama filtra().

El método filtra() hace uso de las variables internas creadas por el método trata_file() a partir de los dataframes de los tres grupos.

Figura 20. Valores máximos y mínimos.

```
lista_acep_w={'F1_WHITE':[3000,4000],
              'F2_WHITE':[32000,40000],
              'F3_WHITE':[40000,48000],
              'F4_WHITE':[20000,24000],
              'F5_WHITE':[32000,38000],
              'F6_WHITE':[37000,41000],
              'F7_WHITE':[32000,37000],
              'F8_WHITE':[22000,27000]}

lista_acep_d={'F1_DARK':[400,1000],
              'F2_DARK':[3000,5000],
              'F3_DARK':[6000,10000],
              'F4_DARK':[2500,4500],
              'F5_DARK':[4300,7000],
              'F6_DARK':[4500,7500],
              'F7_DARK':[4500,8000],
              'F8_DARK':[5000,8000]}

lista_acep_r={'F1_RESULT':[0,120],
              'F2_RESULT':[0,120],
              'F3_RESULT':[0,120],
              'F4_RESULT':[0,120],
              'F5_RESULT':[0,120],
              'F6_RESULT':[0,120],
              'F7_RESULT':[0,120],
              'F8_RESULT':[0,120]}
```

(Fuente: Elaborado por el autor).

El propósito del método filtra() es aplicar los criterios de aceptación de datos definidos en el punto 4.5 del apartado cuarto. Para realizar esta operación de filtrado, el método filtra() aplica un filtro de validación de los datos, haciendo uso de una lista creada en la clase para cada uno de los grupos, como se muestra en la figura 20.

Con la lista de datos de aceptación definida, se ejecuta el método filtra (), el cual contiene diferentes bucles de aceptación, uno para cada dataframe creado, tal como se muestra en la figura 21. Este método devuelve un True si los datos cumplen con los criterios de aceptación, o un False en caso contrario.

Figura 21. Método filtrado de datos leídos.

```
def filtra(self):
    estatus=True

    for x in lista_acep_w: # Bucle aceptacion de datos white
        for y in self.w.columns:
            if self.w.loc[x,y]<=lista_acep_w[x][0] or self.w.loc[x,y]>=lista_acep_w[x][1]:
                estatus=False
                break
        if estatus == False:break

    for x in lista_acep_d: # Bucle aceptacion de datos dark
        for y in self.d.columns:
            if self.d.loc[x,y]<=lista_acep_d[x][0] or self.d.loc[x,y]>=lista_acep_d[x][1]:
                estatus=False
                break
        if estatus == False:break

    for x in lista_acep_r: # Bucle aceptacion de datos result
        for y in self.r.columns:
            if self.r.loc[x,y]<=lista_acep_r[x][0] or self.r.loc[x,y]>=lista_acep_r[x][1]:
                estatus=False
                break
        if estatus == False:break

    return estatus # Retorno del resultado
```

(Fuente: Elaborado por el autor).

Si el resultado devuelto por esta función es satisfactorio TRUE, la aplicación continuara realizando los siguientes correspondientes pasos, en caso contrario, se devuelve un mensaje de error y se finaliza el hilo de ejecución, volviendo al estado inicial esperando un nuevo archivo de entrada.

5.3 Extracción de características.

Después de realizar el tratamiento de datos de entrada y comprobar su validación, el método predict_v1(), continua con su flujo de ejecución, haciendo uso de una función denominada caracteriza(), la cual contiene el código de ejecución interno para la realización de la extracción de características definidas para la aplicación, como se muestra en la figura 22.

Figura 22. Función caracteriza.

```
def caracteriza(dataframe):
    lista=["X0","X1","ERROR"]
    dic={}
    dataframe=dataframe/100
    media=dataframe.mean()
    mediana=dataframe.median()
    des_std=dataframe.std()
    varianza=dataframe.var()
    asimetria=0.5+(dataframe.skew()/10)
    kurtosis=0.5+(dataframe.kurtosis()/10)
    for i in dataframe.columns:
        dic[i]=amc.param(dataframe[i].values)
        dic[i]=np.append(dic[i],(amc.error(dataframe[i].values,dic[i]))/100)
    dataframe.loc["MEDIA"]=media
    dataframe.loc["MEDIANA"]=mediana
    dataframe.loc["DESVIACION_ESTANDAR"]=des_std
    dataframe.loc["VARIANZA"]=varianza
    dataframe.loc["ASIMETRIA"]=asimetria
    dataframe.loc["KURTOSIS"]=kurtosis
    for j,k in enumerate(dataframe.columns):
        dataframe.loc[lista[j]]=dic[k]+0.5
    return dataframe
```

CALCULO MOMENTOS ESTADISTICOS

CALCULO MINIMOS CUADRADOS

CREACION VECTOR PREDECIR

(Fuente: Elaborado por el autor).

Para inicializar la función de caracterización, a esta se le debe de pasar como parámetro de entrada el dataframe validado del grupo RESULT, ya que es el dataframe que contiene los datos de las lecturas de las tiras de papel de la prueba rápida.

La función caracteriza() se ejecuta tomando los datos de cada uno de los sensores y sacando los siguientes momentos estadísticos para cada uno de ellos:

- Media.
- Mediana.
- Desviación estándar.
- Varianza.
- Asimetría.
- Curtosis.

Para el cálculo de estos momentos estadísticos se utilizan métodos dados por la librería panda, los cuales devuelven el valor correspondiente a la métrica de cada columna identificada con cada sensor.

En la segunda parte de la función caracteriza(), se hace uso de la clase llamada min_cuadra() mostrada en la figura 23, la cual extrae características de aproximación por medio de mínimos cuadrados.

Figura 23. Clase min_cuadra.

```
import numpy as np

class min_cuadra:
    def __init__(self,v_read,v_fondo=[100,100,100,100,100,100,100,100],v_control=[75,78,77,71,70,85,85,93]):
        self.fondo=v_fondo
        self.control=v_control
        self.read=np.transpose(v_read)
        self.vector_negro=[1,1,1,1,1,1,1,1]
        self.vector_blanco=[100,100,100,100,100,100,100,100]
        self.patron_rojo=[75,78,77,71,70,85,85,93]
        self.patron_azul=[38,40,38,33,34,32,37,35]

    def param(self,vector1,vector2,vector_read):# white,control,test
        M_A_T=np.array([vector1,vector2])
        M_A=np.transpose(M_A_T)
        MAT_MA=M_A_T.dot(M_A)
        MAT_RD=M_A_T.dot(vector_read)
        INV_MAT_MA=np.linalg.inv(MAT_MA)
        result=MAT_RD.dot(INV_MAT_MA)
        result1=INV_MAT_MA.dot(MAT_RD)
        return result

    def error(self,vector1,vector2,vector_read,pesos):
        error=0
        i=0
        for k, a in enumerate(vector_read):
            error=((vector_read[k]-((vector1[k]*pesos[0])+(vector2[k]*pesos[1])))**2)+error
            i=k
        error=error/(i+1)
        return error
```

(Fuente: Elaborado por el autor).

Para la realización de este cálculo se diseñan los métodos param() y error().

El método param() es el encargado de calcular los valores correspondientes a los coeficientes de aproximación X0, X1.

El método error() calcula el error en la aproximación del vector de valores de entrada frente a dos vectores de valores fijos creados en la clase min_cuadra(), usando para el cálculo del error los coeficientes X0 y X1 generados por el método param().

El último paso que realiza la función caracteriza() es añadir los valores de las características extraídas al *dataframe* de los valores *Result*, dando como resultado un nuevo dataframe el cual contiene los datos de los valores leídos por cada sensor del dispositivo y sus correspondientes valores de caracterización, tal como se muestra en la figura 24.

Figura 24. Dataframe vectores caracterizados.

	Sensor1	Sensor2	Sensor3
F1_RESULT	0.970000	0.970000	0.970000
F2_RESULT	0.970000	0.970000	0.970000
F3_RESULT	0.960000	0.960000	0.960000
F4_RESULT	0.970000	0.970000	0.970000
F5_RESULT	0.960000	0.960000	0.960000
F6_RESULT	0.970000	0.970000	0.970000
F7_RESULT	0.960000	0.960000	0.960000
F8_RESULT	0.970000	0.970000	0.970000
MEDIA	0.966250	0.966250	0.966250
MEDIANA	0.970000	0.970000	0.970000
DESVIACION_ESTANDAR	0.005175	0.005175	0.005175
VARIANZA	0.000027	0.000027	0.000027
ASIMETRIA	0.435594	0.435594	0.435594
KURTOSIS	0.276000	0.276000	0.276000
X0	0.509557	0.500133	0.500000
X1	0.509557	0.500133	0.500000
ERROR	0.509557	0.500133	0.500000

(Fuente: Elaborado por el autor).

La aplicación utiliza este dataframe como valor de entrada para la red de neuronas que se ha desarrollado.

Con el fin de poder entrenar y validar el modelo de la red de neuronas diseñada para la aplicación, se desarrolla código del cual no hace uso el método predict_v1(), pero que se ha utilizado como herramienta para la creación de los grupos de entrenamiento y validación del modelo.

Con el fin de poder realizar el entrenamiento y validación del sistema, se añade al dataframe de los vectores de caracterización el valor CAT_Result correspondiente a la categoría de la muestra para cada uno de los sensores. Este valor de categoría se obtiene del vector de datos de entrada para las muestras de laboratorio visto en la tabla 2 del punto 3.3, obteniendo como resultado el dataframe que se muestra en la figura 25.

Figura 25. Dataframe con categorias.

	Sensor1	Sensor2	Sensor3
F1_RESULT	0.970000	0.970000	0.970000
F2_RESULT	0.970000	0.970000	0.970000
F3_RESULT	0.960000	0.960000	0.960000
F4_RESULT	0.970000	0.970000	0.970000
F5_RESULT	0.960000	0.960000	0.960000
F6_RESULT	0.970000	0.970000	0.970000
F7_RESULT	0.960000	0.960000	0.960000
F8_RESULT	0.970000	0.970000	0.970000
MEDIA	0.966250	0.966250	0.966250
MEDIANA	0.970000	0.970000	0.970000
DESVIACION_ESTANDAR	0.005175	0.005175	0.005175
VARIANZA	0.000027	0.000027	0.000027
ASIMETRIA	0.435594	0.435594	0.435594
KURTOSIS	0.276000	0.276000	0.276000
X0	0.509557	0.500133	0.500000
X1	0.509557	0.500133	0.500000
ERROR	0.509557	0.500133	0.500000
CAT_RESULT	5.000000	4.000000	5.000000

(Fuente: Elaborado por el autor).

El proceso visto hasta este punto se repite para cada una de las muestras generadas en el laboratorio.

A partir de cada una de las muestras, se crea un dataset denominado dataset_muestras, con los dataframes categóricos, creados para cada una de las muestras de laboratorio leídas.

Este dataset estará compuesto por 15000 vectores correspondientes a los valores de lectura de las longitudes de onda de cada sensor, sus valores de caracterización y la categoría perteneciente.

5.4 Red de neuronas

Para el desarrollo de la aplicación se diseña una red de neuronas tipo *Fully Connected Network*, tal como se muestra en la figura 26.

Figura 26. Red de neuronas.

```
model = keras.Sequential()#Creamos el modelo
model.add(keras.layers.Flatten(input_shape=(17,))) #Añadimos la primera capa flatten
model.add(keras.layers.BatchNormalization(momentum=0.99))#Añado la capa batchnormalization
model.add(keras.layers.Dense(N1, activation=tipos_activacion[act1]))# Añado una capa dense con tipo de activacion relu
model.add(keras.layers.Dense(N2, activation=tipos_activacion[act2]))# Añado una capa dense con tipo de activacion relu
model.add(keras.layers.Dense(N3, activation=tipos_activacion[act3]))# Añado una capa dense con tipo de activacion relu
model.add(Dropout(DP, input_shape=(32,)))#Hago dropout para quitar overfiting
model.add(keras.layers.Dense(5, activation='softmax')) #Añado capa final de salida

# Se compila el modelo
model.compile(optimizer=optimizadores[opti],loss='sparse_categorical_crossentropy',metrics=['accuracy'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 17)                0
_________________________________________________________________
batch_normalization (BatchNo (None, 17)                68
_________________________________________________________________
dense (Dense)                (None, 500)               9000
_________________________________________________________________
dense_1 (Dense)              (None, 1000)              501000
_________________________________________________________________
dense_2 (Dense)              (None, 500)               500500
_________________________________________________________________
dropout (Dropout)            (None, 500)               0
_________________________________________________________________
dense_3 (Dense)              (None, 5)                 2505
=================================================================
Total params: 1,013,073
Trainable params: 1,013,039
Non-trainable params: 34
_________________________________________________________________
```

(Fuente: Elaborado por el autor).

La red de neuronas se ha construido haciendo uso de 7 capas. Se define una primera capa de entrada con 17 neuronas correspondiente al número de datos del vector de entrada de la muestra. Como segunda capa se utiliza el método BatchNormaliztion, con el propósito de normalizar los valores de salida de las neuronas.

Las siguientes tres capas son capas tipo dense y están diseñadas para poder hacer pruebas con diferentes funciones de activación haciendo uso de la selección del tipo de activación mediante las variables actx, así como cambiar su número de neuronas con los parámetros Nx.

Como penúltima capa añade una capa con el método Dropout, con el fin de evitar que las neuronas memoricen partes de la entrada, eliminando de este modo el sobreajuste. Para la realización de

diferentes pruebas de la red con distintos valores de dropout, se define una variable llamada DP que representa el valor del dropout aplicado.

En la última capa se coloca como salida una capa tipo dense con 5 neuronas, pertenecientes a cada una de las categorías que se pueden clasificar los datos de entrada. Como función de activación de esta capa se utiliza una función del tipo softmax.

En la figura 27 se muestran las variables utilizadas para la realización de pruebas en la configuración de la red de neuronas.

Figura 27. Variables configuración red de neuronas.

```
ver_pro=1 #Variable para ocultar datos de entrenamiento de los modelos, si se pone a 1 se ven los resultado a 0 se ocultan
valid_prp=0.2 #Variable para definir el % del grupo de validacion
tipos_activacion=["sigmoid","relu","tanh","elu"]#Se definen las funciones de activacion que se usaran para los modelos
inicializacion=["zero","random_uniform"]
ini=1 #Se selecciona la inicializacion tipo random_uniform

opti=3 # Selector de optimizador 0=sgd param modificados 1=adam param modificados 2=sgd 3=adam
call=0 # Selector de callback, 0=val_loos 1=val_acurracy

N1=500 # Selector numero de neuronas de capa 1
N2=1000 # Selector numero de neuronas de capa 2
N3=500 # Selector numero de neuronas de capa 2

act1=1 # Selector numero de funcion de activacion de capa 1
act2=1 # Selector numero de funcion de activacion de capa 2
act3=1 # Selector numero de funcion de activacion de capa 3

DP=0.5 # Selector valor de dropout


#Creamos diferentes modos de callback para hacer pruebas
v_loss = keras.callbacks.EarlyStopping(monitor='val_loss', mode='min', verbose=ver_pro, patience=4)
v_acur = keras.callbacks.EarlyStopping(monitor='val_accuracy', mode='max', verbose=ver_pro, patience=4)
callback=[v_loss,v_acur] #Creamos variable con los diferentes callbacks

#Creamos diferentes configuraciones de optimizadores para hacer pruebas
adam = optimizers.Adam(learning_rate=0.001,beta_1=0.9,beta_2=0.999,epsilon=1e-07,amsgrad=True)
sgd = optimizers.SGD(lr=0.001, decay=1e-6, momentum=0.9, nesterov=True)
optimizadores=[sgd,adam,'sgd','adam'] #Creamos variable con los diferentes optimizadores
```

(Fuente: Elaborado por el autor).

Haciendo uso de estas variables se hacen diferentes configuraciones de la red de neuronas para su correspondiente evaluación.

5.5 Entrenamiento

Para el entrenamiento de la red de neuronas, se dispone de un grupo de 5000 muestras creadas en laboratorio con una distribución uniforme de las categorías, como resultado tendremos 1000

muestras de cada una de las cinco categorías que se han definido en la tabla 5 del punto 3.3 por cada uno de los sensores, dando un total de 15000 muestras.

Del grupo de muestras se selecciona de forma aleatoria un 80% de cada categoría, correspondiente al valor de proteína con la que se ha creado la muestra. Con este 80% se crea el grupo de entrenamiento.

El 20% restantes de las muestras se destinan a crear el grupo de test con el que se realiza la evaluación de la red de neuronas. Quedando de esta forma el dataset principal con 15000 muestras de los tres sensores, dividido en dos grupos, el primer grupo de entrenamiento toma 4000 muestras por cada sensor, con una distribución uniforme por las 5 categorías lo que da un grupo de entrenamiento total de 12000 vectores, quedando el grupo de test compuesto por el conjunto de 3000 muestras.

Para la creación de los grupos de entrenamiento y test se desarrolla la función mostrada en la figura 28, la cual se encarga de tomar los datos del dataset de forma aleatoria e ir creando los grupos de test y entrenamiento con un valor de distribución uniforme de las categorías.

Figura 28. Creador de grupos entrenamiento y validación.

```
y_train=[]
x_train=[]
y_test=[]
x_test=[]
for x in range(5):
    for y in range(1000):
        if y <=799:
            dt_file=pd.read_csv(lista_pathf[x]+lista_arq[x][y])
            x_train.append(dt_file["Sensor1"])
            x_train.append(dt_file["Sensor2"])
            x_train.append(dt_file["Sensor3"])
            y_train.append(x)
            y_train.append(x)
            y_train.append(x)
        else:
            dt_file=pd.read_csv(lista_pathf[x]+lista_arq[x][y])
            x_test.append(dt_file["Sensor1"])
            x_test.append(dt_file["Sensor2"])
            x_test.append(dt_file["Sensor3"])
            y_test.append(x)
            y_test.append(x)
            y_test.append(x)
y_train=np.array(y_train)
x_train=np.array(x_train)
y_test=np.array(y_test)
x_test=np.array(x_test)
```

(Fuente: Elaborado por el autor).

Con los grupos de entrenamiento y evaluación creados se toma el grupo de entrenamiento para entrenar el modelo de la red de neuronas usando para ello diferentes métricas de parametrización.

Para seguir un orden y detectar cambios en el entrenamiento de la red de neuronas, se definen unos parámetros de partida como los mostrados en la figura 29, para los cuales se ha realizado cada uno de los entrenamientos de la red de neuronas.

Figura 29. Configuracion pruebas entrenamiento.

	PRUEBA 1	PRUEBA 2	PRUEBA 3	PRUEBA 4	PRUEBA 5	PRUEBA 6	PRUEBA 7	PRUEBA 8
OPTIMIZADOR	sgd	sgd	sgd	sgd	sgd	sgd	adam	xxx
N1	100	100	500	100	100	500	100	xxx
N2	100	500	1000	100	500	1000	100	xxx
N3	100	100	500	100	100	500	100	xxx
act1	sigmoid	sigmoid	sigmoid	ReLu	ReLu	ReLu	ReLu	xxx
act2	sigmoid	sigmoid	sigmoid	ReLu	ReLu	ReLu	ReLu	xxx
act3	sigmoid	sigmoid	sigmoid	ReLu	ReLu	ReLu	ReLu	xxx

(Fuente: Elaborado por el autor).

Se realiza un entrenamiento para cada una de las diferentes configuraciones de las pruebas, para determinar que modelo es el más preciso y eficiente para el sistema propuesto.

Como criterio para la realización de las pruebas, se toma como punto de partida el optimizador de descenso del vector gradiente (sgd). Se inicializan las capas ocultas con 100 neuronas cada una y una función de activación tipo sigmoid.

Con el punto de partida descrito, la progresión de las pruebas sigue el criterio de aumentar el número de neuronas, en una primera fase (prueba 2), se configuran como N1=100, N2=500, N3=100, observando los resultados, para volver hacer un incremento (prueba 3) del número de neuronas con la configuración de N1=500, N2=1000, N3=500, y se procede a observar el resultado.

Una vez aplicado el proceso de incremento del número de neuronas, este valor se vuelve a configurar como en origen N1=100, N2=100, N3=100, y se modifica la función de activación usando para este caso una función de activación tipo ReLu tal como se ve en la prueba 4 (figura 29), repitiendo a partir de este punto el mismo criterio de aumento del número de neuronas aplicado en las pruebas 2 y 3 para las pruebas 5 y 6.

En la prueba 7 se vuelve a inicializar el número de neuronas con la configuración de N1=100, N2=100, N3=100, se mantiene la función de activación tipo ReLu y se modifica el optimizador usando para esta prueba el optimizador tipo Adam.

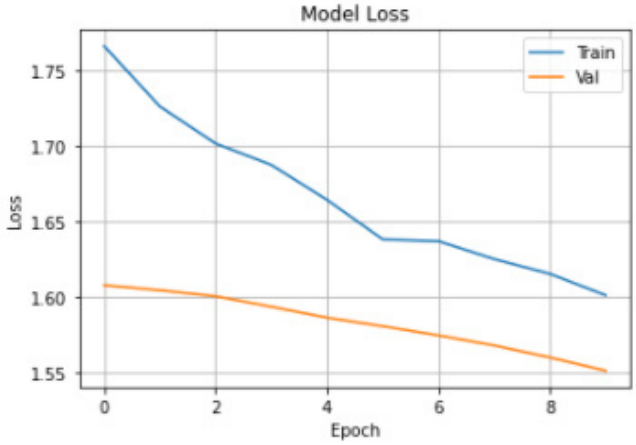
A partir de las 7 primeras pruebas y la observación de los resultados, se trata de buscar una relación entre parámetros que nos permita determinar los parámetros óptimos para la realización de la prueba 8, siendo estos últimos los parámetros para utilizar en nuestra aplicación.

5.5.1 Prueba 1 entrenamiento

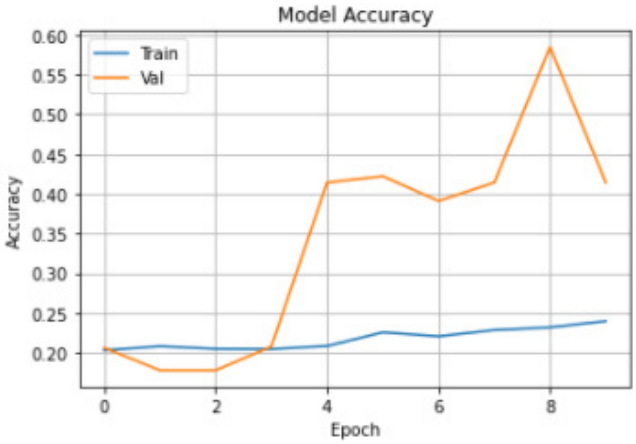
Los gráficos de resultados de los datos del entrenamiento de la prueba 1 se muestran la figura 30.

Figura 30. Graficos prueba 1 entrenamiento.

```
# Representamos la información de loss
plot_loss(history, title="Model Loss")
```




```
# Representamos la información de precisión
plot_acc(history, title="Model Accuracy")
```



(Fuente: Elaborado por el autor).

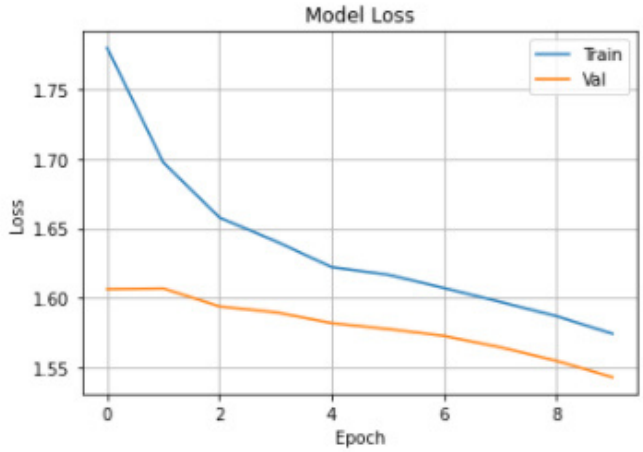
Se puede observar de los datos aportados por los gráficos de la prueba 1 vistos en la figura 30, que si bien el error de pérdida es descendiente, la precisión del modelo es muy baja por lo que no se puede dar por válida esta configuración para el entrenamiento del modelo.

5.5.2 Prueba 2 entrenamiento

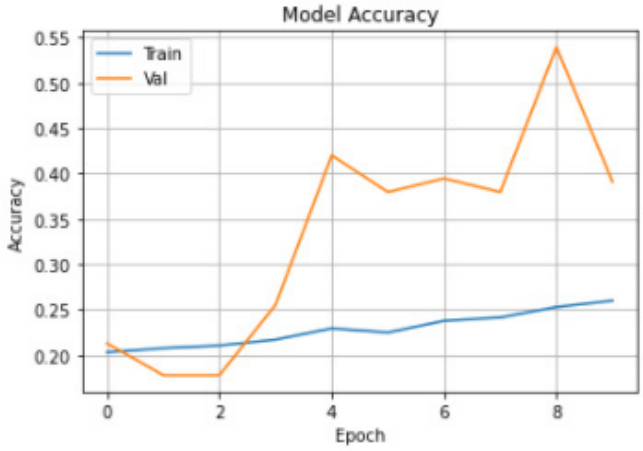
Los gráficos de resultados de los datos del entrenamiento de la prueba 2 se muestran la figura 31.

Figura 31. Graficos prueba 2 entrenamiento.

```
# Representamos la información de loss
plot_loss(history, title="Model Loss")
```




```
# Representamos la información de precisión
plot_acc(history, title="Model Accuracy")
```



(Fuente: Elaborado por el autor).

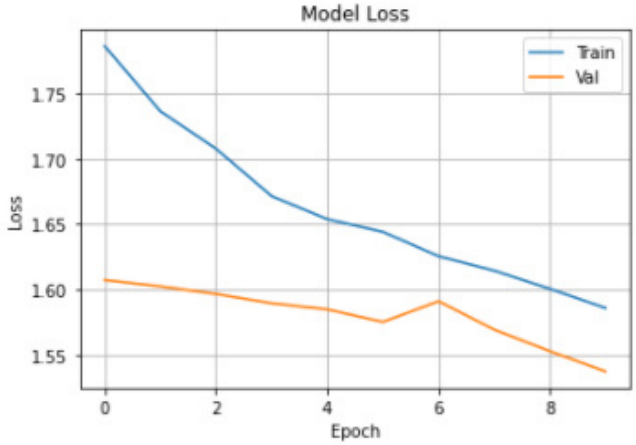
Al igual que los resultados obtenidos para la prueba 1, en los gráficos de la prueba 2 se observa un valor decreciente en el error de pérdida, pero un valor de precisión bastante bajo, aunque se nota una pequeña mejoría frente a la prueba 1, lo cual indica que el aumento del número de neuronas en las capas ocultas es favorable para esta configuración, pero no suficiente.

5.5.3 Prueba 3 entrenamiento

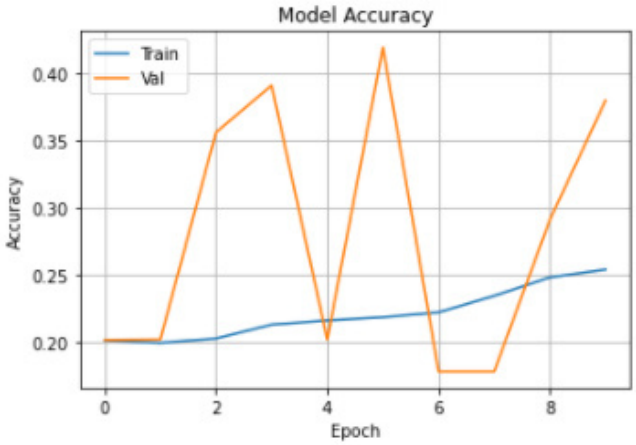
Los gráficos de resultados de los datos del entrenamiento de la prueba 3 se muestran la figura 32.

Figura 32. Graficos prueba 3 entrenamiento.

```
# Representamos la información de loss
plot_loss(history, title="Model Loss")
```




```
# Representamos la información de precisión
plot_acc(history, title="Model Accuracy")
```



(Fuente: Elaborado por el autor).

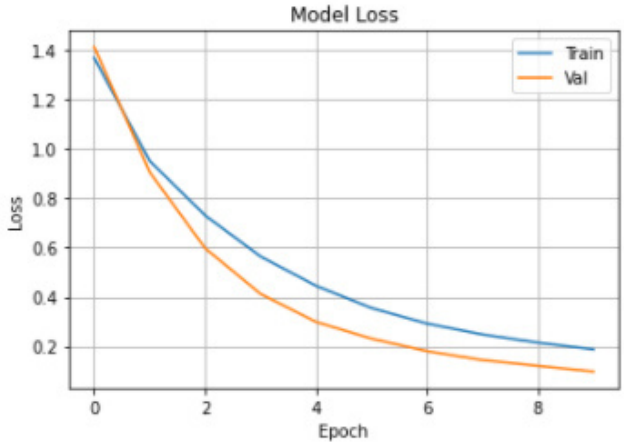
La prueba 3 aporta resultados muy similares a los observados en la prueba 2, tanto en el valor de error de pérdida como el de precisión. Por otro lado, en los valores de validación de la precisión, se observan grandes saltos, esto es debido a la memorización que está haciendo el modelo al aumentar el número de neuronas y el ajuste que hace el dropout para tratar de eliminar esta memorización. Esto nos indica que para el modelo con el que se trabaja no es necesario aumentar tanto el número de neuronas ya que el modelo no mejora, pero en cambio puede sufrir overfitting, que es un fenómeno que se da cuando la red de neuronas aprende detalles propios de los datos en lugar de aprender las características generales de estos.

5.5.4 Prueba 4 entrenamiento

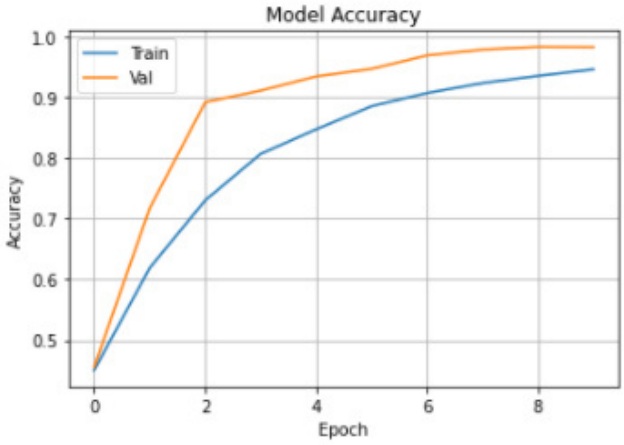
Los gráficos de resultados de los datos del entrenamiento de la prueba 4 se muestran en la figura 33.

Figura 33. Graficos prueba 4 entrenamiento.

```
# Representamos la información de loss
plot_loss(history, title="Model Loss")
```




```
# Representamos la información de precisión
plot_acc(history, title="Model Accuracy")
```



(Fuente: Elaborado por el autor).

De los gráficos obtenidos de los valores de entrenamiento para la prueba 4, se observa una gran mejora, tanto en el valor de pérdida del error, como en la precisión del modelo. Estos valores se pueden comparar con los vistos para la prueba 1, ya que la configuración en el número de neuronas es la misma y la única diferencia se encuentra en la función de activación.

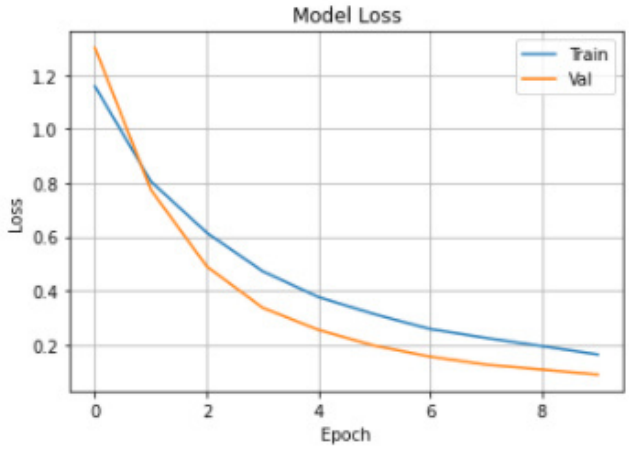
Si se realiza dicha comparación se observa que el uso de una función de activación u otra tiene resultados totalmente diferentes, mejorando bastante la evaluación del sistema cuando se usa la función de activación tipo ReLu.

5.5.5 Prueba 5 entrenamiento

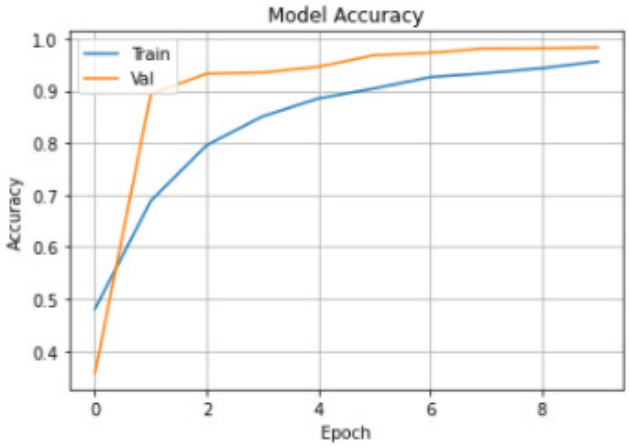
Los gráficos de resultados de los datos del entrenamiento de la prueba 5 se muestran la figura 34.

Figura 34. Graficos prueba 5 entrenamiento.

```
# Representamos la información de loss
plot_loss(history, title="Model Loss")
```




```
# Representamos la información de precisión
plot_acc(history, title="Model Accuracy")
```



(Fuente: Elaborado por el autor).

En los gráficos de los resultados obtenidos en la prueba 5 si bien se ve una leve mejora, se puede observar que los valores son muy similares a los obtenidos en la prueba 4, donde el número de neuronas era menor.

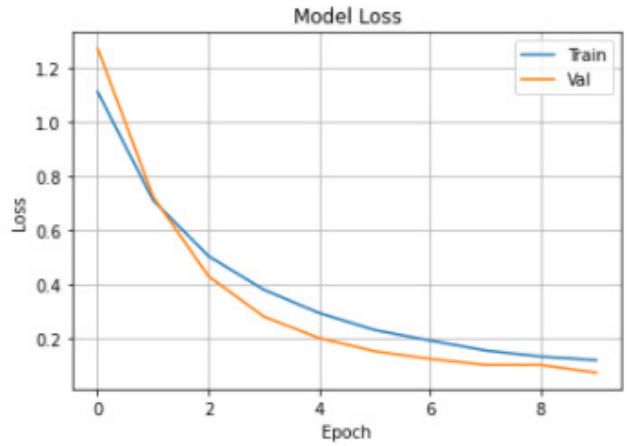
Esta tendencia se ve reforzada o rechazada en la prueba 6 donde el valor del número de neuronas de las capas ocultas incrementa más.

5.5.6 Prueba 6 entrenamiento

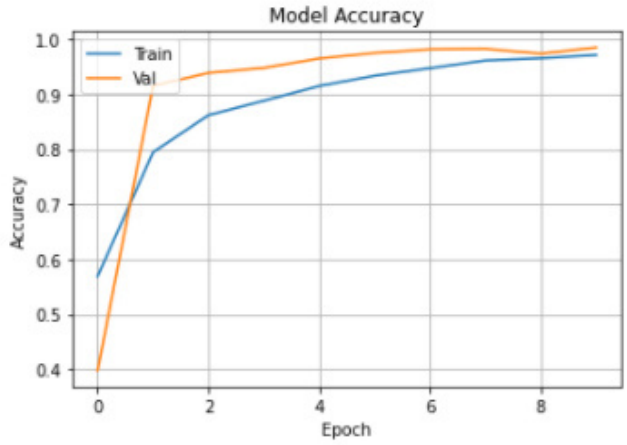
Los gráficos de resultados de los datos del entrenamiento de la prueba 6 se muestran la figura 35.

Figura 35. Graficos prueba 6 entrenamiento.

```
# Representamos la información de loss
plot_loss(history, title="Model Loss")
```




```
# Representamos la información de precisión
plot_acc(history, title="Model Accuracy")
```



(Fuente: Elaborado por el autor).

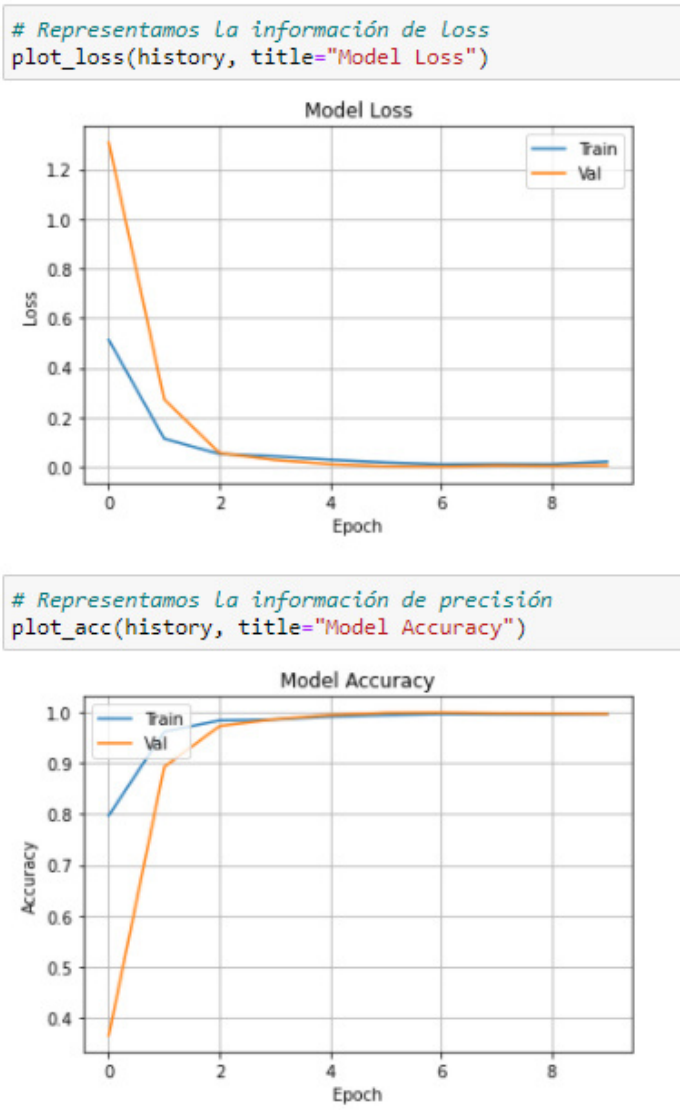
En los gráficos de los resultados de la prueba 6 se observa al igual que en la prueba 4 y la prueba 5 una mejora considerable tanto en error de pérdida como en precisión del sistema, frente a las pruebas 1, 2 y 3. También se observa que entre las pruebas 4, 5 y 6 si bien es cierto que la precisión mejora un poco según se van subiendo el número de neuronas, esta mejoría es tan baja

que no es suficiente para justificar el aumento de neuronas realizado, ya que esto perjudica la eficiencia del modelo, haciendo que sea más lento a la hora de realizar predicciones.

5.5.7 Prueba 7 entrenamiento

Los gráficos de resultados de los datos del entrenamiento de la prueba 7 se muestran la figura 36.

Figura 36. Graficos prueba 7 entrenamiento.



(Fuente: Elaborado por el autor).

En la prueba 7 se observan unos resultados en sus gráficos los cuales mejoran los obtenidos en las pruebas 4, 5 y 6, llegando a ser el valor de pérdida del error prácticamente 0 y consiguiendo unos valores de precisión prácticamente del 100%. Este cambio es debido al uso del optimizador adam, en lugar del sgd.

5.5.8 Prueba 8 entrenamiento

Para realizar la prueba número 8 se tienen en cuenta los resultados observados en las anteriores pruebas y las tendencias vistas de los ajustes realizados. Donde se ha observado que la mayor mejora en los resultados de entrenamiento se ha dado cuando se han cambiado las funciones de activación de tipo sigmoid a tipo ReLu.

También se ha observado que el cambio del optimizador ha aportado mejoría, obteniendo mejores resultados para nuestro modelo, usando un optimizador del tipo adam frente a uno tipo sgd.

Por último, se establece una relación entre el aumento del número de neuronas y la precisión del modelo, aunque se observa que este aumento no siempre es favorable debido al incremento en la carga de parámetros, generando un modelo menos eficiente en su ejecución, y por otro lado provoca el posible overfitting, lo cual es algo no deseado, ya que puede llevar a crear un modelo que genere buenas predicciones para los valores de entrenamiento, pero malos para la evaluación del mismo.

Debido a la observación del efecto en el incremento de neuronas de las capas ocultas de la red y los resultados obtenidos para la prueba 7, para la prueba 8 se decide optar por mantener tanto la función de activación tipo ReLu como el optimizador tipo adam, y hacer una reducción en el número de neuronas de las capas intermedias, comprobando de esta forma si es posible evitar el overfitting sin necesidad de aumentar el dropout y conseguir un modelo más eficiente en su ejecución, manteniendo una pérdida en el error baja y una precisión alta.

Siguiendo los criterios comentados anteriormente, se ajusta la red de neuronas para realizar la última prueba una configuración como la mostrada en la figura 37:

Figura 37. Parámetros prueba 8.

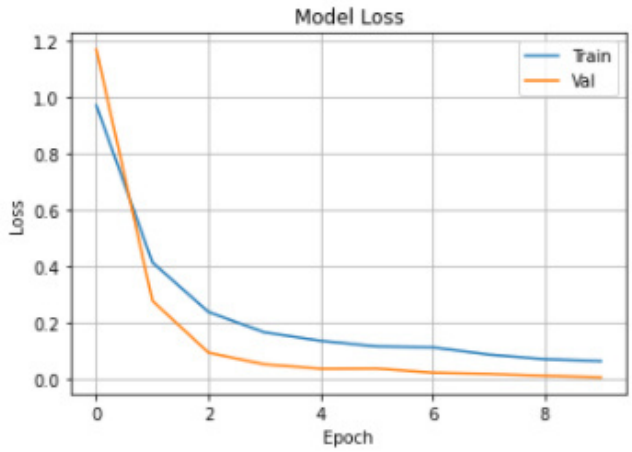
	PRUEBA 8
OPTIMIZADOR	adam
N1	25
N2	50
N3	25
act1	ReLu
act2	ReLu
act3	ReLu

(Fuente: Elaborado por el autor).

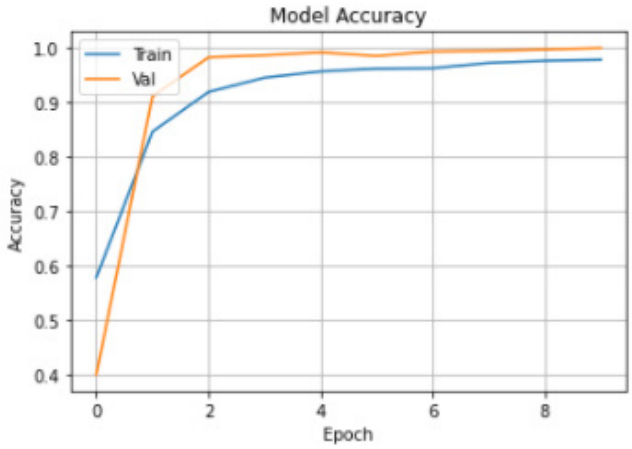
Los gráficos de resultados del entrenamiento de la prueba 8 se muestran la figura 38.

Figura 38. Graficos prueba 8 entrenamiento.

```
# Representamos la información de loss
plot_loss(history, title="Model Loss")
```




```
# Representamos la información de precisión
plot_acc(history, title="Model Accuracy")
```



(Fuente: Elaborado por el autor).

Los gráficos de los resultados de la prueba 8 confirman lo observado hasta ahora, devolviendo como resultado una pérdida en el error bastante baja y una precisión del modelo cercano al 100%.

De los gráficos también se puede observar que su aproximación al 100% de precisión se hace de una forma mucho menos abrupta que para el caso de los datos de la prueba. Esto se debe a la reducción en el número de neuronas de las capas intermedias realizado para la configuración de la red en la prueba 8, reduciendo con este ajuste la posibilidad de overfitting y mejorando la eficiencia de ejecución del modelo.

5.6 Configuración de la red de neuronas

Una vez realizadas las pruebas de entrenamiento se toma la configuración realizada para la prueba 8, quedando la configuración de la red de neuronas del siguiente modo:

- Capa de entrada tipo Flatten con 17 neuronas.
- Capa de normalización tipo BatchNormalization con momentum = 0.99
- 1º Capa oculta tipo Dense con 25 neuronas y función de activación ReLu.
- 2º Capa oculta tipo Dense con 50 neuronas y función de activación ReLu.
- 3º Capa oculta tipo Dense con 25 neuronas y función de activación ReLu.
- Función Dropout = 0.5
- Capa de salida con 5 neuronas y función de activación softmax.
- Optimizador = adam

5.7 Variables de salida

Como variables de salida el sistema devuelve una matriz como la indicada en el punto 4.7 del apartado 4 especificaciones del software. Junto con la matriz de resultados el sistema devuelve el ID de la muestra para confirmar su procedencia, y la versión de la aplicación con la que se ha hecho la predicción.

Como dato de salida para la visualización de los resultados, se genera un dataframe de salida como el mostrado en la figura 39, el cual contiene la predicción de pertenencia de la muestra a cada categoría.

Figura 39. Dataframe de salida.

```
# Se muestra la prediccion de categoria perteneciente
output
```

	TEST_1	TEST_2	CONTROL
CAT_1	0.048238	2.925770e-07	9.026758e-19
CAT_2	0.916531	9.863954e-01	9.700803e-16
CAT_3	0.025528	9.986082e-03	1.000000e+00
CAT_4	0.007104	3.618047e-03	1.176693e-15
CAT_5	0.002600	2.116366e-07	2.179602e-19

(Fuente: Elaborado por el autor).

6 Evaluación

Para la evaluación del sistema desarrollado, se dispone del grupo de muestras de test creado a partir del dataset principal de muestras.

El grupo de muestras de test está compuesto por 3000 muestras con una distribución normal de las categorías, asegurando de este modo la contención de muestras de todas las categorías para la evaluación del sistema.

Para realizar una primera evaluación del sistema, se hace uso del método evaluate(), ofrecido por la librería Keras para la evaluación de modelos, al cual se le pasan como parámetros de entrada el grupo de muestras de test y sus correspondientes categorías tal como se muestra en la figura 40.

Figura 40. Evaluación del modelo.

```
print("La accuracy del modelo con los datos de test es: ",model.evaluate(x_test, y_test))
```

```
94/94 [==============================] - 0s 989us/step - loss: 0.1787 - accuracy: 0.9473
La accuracy del modelo con los datos de test es:  [0.1786544770002365, 0.9473333358764648]
```

(Fuente: Elaborado por el autor).

Como se puede observar del resultado aportado por el método evaluate() de keras, la precisión que nos entrega el modelo entrenado es cercana al 95% y la pérdida del error es muy baja cercana al 0.18, lo cual nos indica que el modelo entrenado aporta buenos resultados para la función que está diseñado.

Con el fin de realizar una evaluación del sistema más detallada, se desarrolla una función capaz de generar una matriz de confusión de los datos predichos por el modelo a partir de los datos de test, la clasificación conocida de dichos datos y el modelo entrenado, tal como se muestra en la figura 41.

Esta función toma como entrada los valores del grupo de test y el valor de la categoría de cada uno de estos valores, así como el modelo entrenado, recorriendo cada uno de los vectores de datos del grupo de test y haciendo la predicción de la categoría mediante el uso del modelo entrenado.

Figura 41. Función matriz de confusión.

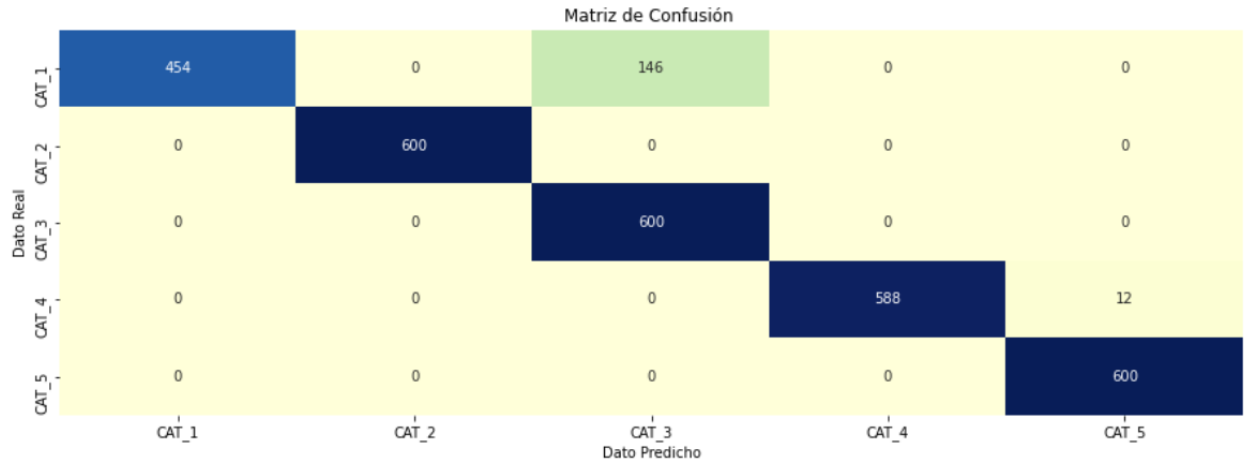
```
def matriz_confusion(x_test,y_test, modelo):
    pred_total=[]
    real_total=[]
    errores=[]
    fig, ax=plt.subplots()
    fig.set_size_inches(15,5)
    for k,j in enumerate(x_test):
        m=[]
        m.append(j.tolist())
        lista=modelo.predict(m).tolist()
        pos_max=lista[0].index(max(lista[0]))
        pred_total.append(pos_max)
        real_total.append(y_test[k])
    rn_cm = confusion_matrix(real_total, pred_total) # Hacemos la matriz de confusion
    tipos_clases=["CAT_1","CAT_2","CAT_3","CAT_4","CAT_5"]
# Creamos un dataframe con la matriz de confusion
    rn_cmp = pd.DataFrame(rn_cm,index = tipos_clases,columns = tipos_clases)
# Configuramos el tamaño y caracteristicas de de la figura
    sns.heatmap(rn_cmp, annot=True, cmap="YlGnBu", fmt="d",cbar=False)
    ax.set_title('Matriz de Confusión ') #Damos nombre a la figura
    ax.set_ylabel('Dato Real') #Damos nombre al eje Y
    ax.set_xlabel('Dato Predicho') #Damos nombre al eje X
    plt.show() # Hacemos que solo se mueste la figura
    return real_total, pred_total, rn_cmp
```

(Fuente: Elaborado por el autor).

Cada una de las predicciones generadas se añade a una lista de predicciones, creando de esta forma un array de valores de predicciones de las categorías correspondiente a cada una de las muestras del conjunto de test.

Figura 42. Matriz de confusión.

```
real_total, pred_total, rn_cmp=matriz_confusion(x_test,y_test, model)
```



(Fuente: Elaborado por el autor).

Una vez que se crean todas las predicciones correspondientes a cada vector de valores de las muestras del grupo de test, se crea la matriz de confusión a partir del array de predicciones creados por el modelo y los datos de las categorías reales de cada una de las muestras, dando como resultado de salida la matriz de confusión tal como se muestra en la figura 42. La matriz de confusión nos muestra de una forma gráfica el resultado de las predicciones realizadas por el modelo.

En las filas se sitúan los valores reales de la categoría perteneciente a cada una de las muestras. Haciendo el sumatorio de todos los valores de cada una de las filas, el valor de todas ellas se comprueba que es de 600 muestras por categoría siendo este igual para todas, por lo que el método utilizado para la generación del grupo de test el cual daba una distribución homogénea de todas las categorías en el grupo está bien implementado.

Las columnas de la matriz de confusión indican la predicción realizada para cada muestra y la categoría asignada.

Como se puede ver en la figura 41, el modelo ha predicho perfectamente las muestras de las categorías 2, 3 y 5, haciendo la peor predicción para la categoría 1 con 146 muestras mal asignadas a la categoría 3 y equivocándose en 12 muestras de la categoría 4 que han sido asignadas como categoría 5.

Con la finalidad de tener un numero cuantitativo para la evaluación del sistema, se desarrolla la función report() mostrada en la figura 43 la cual nos da un reporte de los resultados obtenidos por el modelo.

Figura 43. Función report.

```
def report(real_total, pred_total, rn_cmp):
    print('Datos de matriz de Confusión ')
    print(classification_report(real_total, pred_total,
                                target_names=rn_cmp.columns))
    return None
```

(Fuente: Elaborado por el autor).

En la figura 44 se muestran los resultados obtenidos para la evaluación del modelo entrenado.

Figura 44. Resultados de función report.

```
report(real_total, pred_total, rn_cmp)
```

```
Datos de matriz de Confusión
              precision    recall  f1-score   support

       CAT_1       1.00      0.76      0.86       600
       CAT_2       1.00      1.00      1.00       600
       CAT_3       0.80      1.00      0.89       600
       CAT_4       1.00      0.98      0.99       600
       CAT_5       0.98      1.00      0.99       600

    accuracy                           0.95      3000
   macro avg       0.96      0.95      0.95      3000
weighted avg       0.96      0.95      0.95      3000
```

(Fuente: Elaborado por el autor).

A continuación, se analizan los resultados obtenidos por el conjunto de test mostrados en las figuras 42 de la matriz de confusión y la figura 44 con los resultados de la función report.

La columna denominada precisión mostrada en la figura 44, nos indica la relación entre el número de acierto de las predicciones y el número de muestras predichas para una categoría. Un valor alto en el parámetro de precisión, nos indica que el modelo genera pocos falsos positivos.

En el caso de la categoría 1 se han predicho 454 muestras como categoría 1 de las cuales son correctas las 454, como se muestra en la figura 42. Como todas las muestras que se han predicho como categoría 1 son correctas cuando se hace la operación (muestras acertadas =454 / muestras predichas para la categoría =454) el resultado de precisión es un 1.0, puesto que no se ha asignado a esta categoría ninguna muestra que no pertenezca a ella.

En el caso de la categoría 3 el valor de precisión es de 0.8. Para la categoría 3 se han predicho 746 muestras siendo su valor real de muestras correctas de 600, tal como se ve en la figura 42. Al haber asignado a la categoría 3 muestras no correspondientes a esta categoría el valor de la precisión se reduce ya que si se hace la operación (muestras acertadas =600 / muestras predichas para la categoría =764) el valor de precisión que nos da es de 0.8 para la categoría 3.

La misma situación se da para las categorías 4 y 5 ya que existe una predicción errónea para 12 muestras de la categoría 4 asignadas a la categoría 5.

La columna denominada recall mostrada en la figura 44, nos indica la relación entre el número de aciertos en la asignación de una muestra a una categoría y el número total de muestras de la categoría. Un valor de recall alto, indica que el modelo genera pocos falsos negativos.

Siguiendo el análisis visto para el parámetro de precisión, para la categoría 1 se observa un valor de recall de 0.76 debido a que, del total de las muestras pertenecientes a esta categoría, solo se le han asignado a la categoría 1 un total de 454 muestras, por lo que si se realiza la operación (muestras asignadas correctamente =454 / muestras pertenecientes a la categoría =600) el valor de recall para la categoría 1 es de 0.76.

En el caso de la categoría 3 el valor de recall es 1.0 ya que, de las 600 muestras existentes para esta categoría, se han asignado las 600 correctamente, por lo que si se hace la operación (muestras asignadas correctamente =600 / muestras pertenecientes a la categoría =600), el valor que da el parámetro de recall para esta categoría es de 1.0

Al igual que en el caso del parámetro de precisión, esta situación también se da entre las categorías 4 y 5, por los fallos en la predicción de las muestras entre las dos categorías.

La columna denominada f1-score mostrada en la figura 44, da la media armónica entre la precisión y el recall. Este parámetro da un resumen de los valores de precisión y recall en un único dato. Reduce el impacto de valores altos y aumenta el de valores bajos, por lo que, si existe una gran diferencia ente los valores de precisión y recall, el resultado se aproximaría al valor más bajo.

Como se observa en la figura 44, todos los valores de f1-score son bastante elevados, siendo superiores en todos los casos al 0.85, lo que indica que el modelo tiene pocos fallos en las predicciones tanto en falsos positivos como en falsos negativos.

Por último, la columna denominada support mostrada en la figura 44, indica el número de muestras de cada una de las categorías, así como el número de muestras totales del conjunto.

7 Conclusiones y trabajo futuro

A continuación, se presentan las conclusiones obtenidas del trabajo realizado y las posibles líneas de trabajo futuro.

7.1 Conclusiones

Se concluye que el sistema aporta unos resultados bastante buenos para la función que fue desarrollado, haciendo uso de la caracterización de los datos y la red de neuronas para la predicción de resultados, aportando unos valores de precisión en la categorización de las muestras de laboratorio cercanos al 95%.

Se determina que los valores descriptivos obtenidos caracterizan correctamente los datos y son representativos de estos, debido a los buenos resultados obtenidos con los valores descriptivos seleccionados para la categorización de las muestras evaluadas.

Dentro de los posibles algoritmos que se pueden aplicar, se ha seleccionado un modelo compuesto por una red de neuronas secuencial simple por la configuración del tipo de dato de entrada, siendo estos datos de entrada un vector lineal con los datos leídos y sus valores característicos.

Una vez que la red de neuronas ha sido definida se han realizado diferentes pruebas con distintos tipos de funciones de activación y numero de neuronas por capa, llegando a la conclusión que el modelo que mejores resultados ha aportado ha sido la configuración dada en la prueba 8, la cual consta de una primera capa de entrada con 17 neuronas, seguida de la técnica de normalización BatchNormalization con un momentun de 0.99. Como primera capa oculta se han utilizado 25 neuronas con una función de activación tipo ReLU. En la segunda capa oculta se ha hecho uso de 50 neuronas con función de activación ReLU. La tercera capa oculta se ha configurado con 25 neuronas y función de activación tipo ReLU. Para evitar el efecto de overfitting se ha usado la técnica de dropout, con un valor de 0.5. Por último, como capa de salida se han puesto 5 neuronas una por cada categoría con una función de activación del tipo softmax. Como optimizador se ha usado el modelo Adam.

Los datos obtenidos han sido generados mediante la realización de las medidas con las 3000 pruebas de laboratorio, que se planteó en los objetivos específicos. Destacar que todas las pruebas realizadas hasta la fecha se han hecho con muestras controladas, con dispositivos de lectura conocidos y perfectamente calibrados y con tiras de papel de pruebas rápidas que contenían reactivos de coloración rojiza, lo que limita a longitudes de onda elevadas dentro del espectro de la visión humana.

7.2 Líneas de trabajo futuro

Como líneas de trabajo futuro se plantean el posible uso de diferentes tecnologías para la adquisición de datos, dando esto lugar al uso de otros algoritmos para la predicción de los resultados como puede ser el uso de redes de neuronas convolucionales en el caso de usar tecnologías de adquisición de los datos tipo escáner.

Otra línea de trabajo futura puede ser el uso de tiras de papel con reactivos de diferentes coloraciones, para lo que se tendría que realizar una ampliación del estudio en la extracción de características y la posible modificación del modelo de predicción.

8 Bibliografía

Aldabas E. (2012), Introducción al reconocimiento de patrones mediante redes neuronales. UPC-Campus Terrassa. Barcelona

Alonso, L. y Calonge, T. (2001). Redes neuronales y reconocimiento de patrones. En L. Alonso, "Reconocimiento de patrones con redes neuronales". Salamanca: Imprenta Cardenal

Baxes, G. A. 1994. Digital Image Processing - Principles and Applications. U.S.A. John Wiley & Sons.

Betancourt, O. Gustavo, A. Giraldo Suárez, E. Franco, B. John, F. Reconocimiento de patrones de movimiento a partir de señales electromiográficas Scientia Et Technica, vol. X, núm. 26, diciembre, 2004, pp. 53-58 Universidad Tecnológica de Pereira, Pereira, Colombia

Camargo-Vega, Juan José; Camargo-Ortega, Jonathan Felipe; Joyanes-Aguilar, Luis Conociendo Big Data Facultad de Ingeniería, vol. 24, núm. 38, enero-junio, 2015, pp. 63-77 Universidad Pedagógica y Tecnológica de Colombia Tunja, Colombia

Diaz Cabrera, J.M 2008, Análisis de los sistemas de adquisición de datos en los sistemas automáticos del currículo de tecnología industrial II. ISSN 1988-6047 DEP.LEGAL: GR 2922/2007 Nº5 - Abril de 2008

Ferré Baldrich, J. (2004), Calibración multivariable en análisis cuantitativo, el modelo directo. Técnicas de laboratorio, 297, 986-989

F.J, Muñoz, P.A, Marín, J.I. (2006), Reconocimiento de comandos de voz usando la transformada wavelet y máquinas de vectores de soporte. Scientia et Technica, 2, 35-40

Fontal, B. (2005), El espectro magnético y sus aplicaciones. Venezuela: Escuela venezolana para la enseñanza de la química.

González Henríquez, J.J. 2005, El histograma con la TI-92: optimización de clases. Revista de didáctica de las matemáticas,61,67-72

Gonzalez, Nahuel & Gawron, Olaf & Lage, Fernando. (2014). Analisis de metodos para el reconocimiento de patrones en ECG.

Izurieta, F., Saavedra, C.: Redes Neuronales Artificiales. U. d. C. Departamento de Física, Ed., Concepción (2006).

Kuri-Morales, A. and Galaviz-Casas, J. (2002). Algoritmos Genéticos, Fondo de Cultura Económica/UNAM/IPN.

L. Al Shalabi and Z. Shaaban, "Normalization as a Preprocessing Engine for Data Mining and the Approach of Preference Matrix," 2006 International Conference on Dependability of Computer Systems, Szklarska Poreba, 2006, pp. 207-214, doi: 10.1109/DEPCOS-RELCOMEX.2006.38.

Luna-Ortega, Carlos & Mora-González, Miguel & Martíunez Romo, Julio César & Luna-Rosas, Francisco & Muñoz-Maciel, Jesús. (2013). Reconocimiento del habla mediante el uso de la correlación cruzada y una perceptrón multicapa. Nova scientia. 6. 108-124. 10.21640/ns.v6i12.26.

Peña-Peñate, Adrián, Silva Rojas, Luis Guillermo, & Alcolea Núñez, Rubén. (2016). Módulo de filtrado y segmentación de imágenes médicas digitales para el proyecto Vismedic. Revista Cubana de Ciencias Informáticas, 10(1), 13-27. Recuperado en 08 de diciembre de 2020, de http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S2227-18992016000100002&lng=es&tlng=es.

Rairán, D., & Fonseca, J. (2015). Medición del sentido de giro, velocidad y posición angular de un eje mediante encoders. Respuestas, 15(1), 33-42. https://doi.org/10.22463/0122820X.177

RICHARDS, J. A. 1995. Remote Sensing Digital Image Analysis - An Introduction. Berlín. Springer-Verlag

R. López and J. Fernández, Las Redes Neuronales Artificiales, Netbiblo, 2008.

Rodríguez-Ruiz, Luis & García-Varea, Ismael & Puerta, Jose. (2003). Tecnica de reconocimiento de patrones del habla basada en algoritmos geneticos.

Sabadías, A.V. Estadística descriptiva e inferencial. Recuperado de https://books.google.es/books?id=RbaC-wPWqjsC.

S. V. M. Vishwanathan and M. Narasimha Murty, "SSVM: a simple SVM algorithm," Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290), Honolulu, HI, USA, 2002, pp. 2393-2398 vol.3, doi: 10.1109/IJCNN.2002.1007516.

Vélez, Nini Juliana, & Erazo, Jorge Humberto, & Loaiza, Humberto (2009). Sistema de clasificación de imágenes basado en técnicas de reconocimiento de patrones aplicado en termografía y robótica. El Hombre y la Máquina, 33,45-52

9 Anexos

9.1 Anexo. Artículo de investigación

Sistema de predicción de resultados de análisis clínicos para pruebas rápidas

David Vegas Diez

Universidad Internacional de la Rioja, Logroño (España)

Fecha 02/02/2021

Resumen

El presente documento trata de comprobar la utilidad de la aplicación de técnicas de inteligencia artificial para la predicción de resultados de análisis de tiras de papel de pruebas rápidas. El objetivo es desarrollar un sistema que sea capaz de realizar predicciones de los resultados adquiridos en la lectura de las tiras de papel de las pruebas rápidas.

Para la realización de este estudio se han hecho 15000 test de tiras de papel de prueba rápidas en un laboratorio. Se han realizado la lectura de los resultados y se han hecho las predicciones para comprobar la fiabilidad y precisión del sistema desarrollado.

El resultado aportado por el sistema ha sido de un 95% de acierto en sus predicciones, por lo que se concluye que es una opción válida para la predicción de resultado de las tiras de papen de las pruebas rápidas.

Palabras Clave

Caracterización, Longitud de onda, Prueba rápida, Red de neuronas.

I. Introducción

En la actualidad uno de los distintos tipos de pruebas médicas que se realizan son las denominadas pruebas rápidas. Su base principal de funcionamiento se basa en el uso de reactivos impregnados en tiras de papel sobre las cuales se depositan los fluidos que se quieren analizar. Los resultados de este tipo de pruebas se dan mediante el cambio de la coloración de las zonas en las cuales se encuentra el reactivo al contacto con él fluido analizado. La evaluación de los resultados aportados por este tipo de pruebas es habitualmente realizada por el ojo humano, determinando si el cambio producido en la coloración del reactivo es suficiente o no como para determinar si existe enfermedad o patología en el fluido analizado. Este tipo de evaluación genera una gran incertidumbre ya que el resultado depende de lo entrenado que este cada observador. Para solventar este problema, se diseñaron sistemas electrónicos para la lectura de los cambios de coloración de los reactivos. Siguiendo esa línea de trabajo este estudio trata de crear un sistema capaz de dotar a dichos dispositivos de lectura con la inteligencia suficiente para poder realizar las predicciones de los resultados de los cambios de coloración observados en los reactivos.

Con el fin de dotar de inteligencia a los dispositivos de lectura, se plantea un sistema capaz de admitir datos provenientes de estos dispositivos de lectura de tiras de papel de pruebas rápidas. El sistema planteado trata los datos filtrándolos, organizándolos y normalizándolos, para crear a posteriori una caracterización de la lectura realizada de la prueba rápida. Teniendo la caracterización de los datos el sistema procede a predecir el resultado de la categoría perteneciente de la muestra, mediante el uso de una red de neuronas que se ha entrenado con muestras controladas de laboratorio.

Para conseguir un modelo de red de neuronas que aporte resultados fiables de forma rápida, se realizan diferentes configuraciones con el fin de conseguir un modelo optimo para realizar esta función.

Después de entrenar al sistema con 12000 muestras controladas se comprueba sus resultados con 3000 muestras de test.

Del análisis de los resultados obtenidos se concluye que el sistema desarrollado cumple con la funcionalidad para la cual fue diseñado, aportando unas predicciones con alta exactitud en sus resultados y una baja métrica de error de perdida.

II. Estado del Arte

En el estado del arte se hace un estudio de los métodos utilizados para la realización de sistemas con redes de neuronas enfocadas al reconocimiento de patrones y predicción de su clasificación. Para ello se siguen los pasos propuestos por Alonso y Calonge [1], donde se proponen cuatro etapas para realizar el desarrollo de un sistema de estas características. Dentro de cada una de estas etapas se realizan diferentes pasos los cuales se deben estudiar y conocer para poder llevar a cabo una buena implementación del sistema.

Siguiendo la línea de trabajo expuesta por Alonso y Calonge [1] se puede dividir el estudio del estado del arte en cuatro etapas.

En la primera etapa se tratan los métodos existentes para la adquisición de datos. Se estudian cuáles son las diferentes técnicas usadas para este fin, tal como propone Diaz Cabrera, J.M [4], donde se hace hincapié en la necesidad del conocimiento de los elementos captadores de la señal, por lo que se realiza un estudio básico del funcionamiento de los fotodiodos [7], que son los sensores captadores usados por los dispositivos de lectura con los que se están trabajando para el proyecto. Continuando con la línea de investigación de los sensores captadores se ve la

necesidad de comprender y entender el significado y propiedades del entorno que se está monitorizando, realizando un estudio de las propiedades de la luz [7].

Una vez comprendido como se adquieren los datos en los dispositivos de lectura, la segunda fase propuesta por Alonso y Calonge [1], se centra en el tratamiento de estos datos. Dentro de las diferentes técnicas para el tratamiento de datos se ha hecho foco en el estudio de sistemas de filtrado, como exponen Baxes [2] y Richards [12]. Otro de los métodos de estudio en el marco del tratamiento de los datos ha sido la organización y los tipos de estructuras de almacenamiento [3]. Por último, en esta etapa se ha estudiado también la normalización de los datos [10] para dejarlos preparados para la siguiente fase.

El tercer paso que se propone en el estudio del arte para la creación de un sistema de reconocimiento de patrones con redes de neuronas es la extracción de características de los datos obtenidos, empezando por una comprensión visual de los datos mediante el uso de histogramas [8]. Otra técnica usada para la extracción de características es el uso de la estadística descriptiva [14] y el uso de la aproximación por medio de mínimos cuadráticos [5], con el fin de crear un vector de características propias de los datos.

El cuarto y último paso estudiado es el reconocimiento de patrones, para lo que se hace un estudio de las diferentes técnicas utilizadas para este fin como son la correlación cruzada [11], las máquinas de vector soporte [6], los algoritmos genéticos [9] y las redes de neuronas [13].

III. Objetivos y Metodología

El objetivo general es desarrollar un sistema de predicción para las pruebas rápidas de tiras de papel.

Se especifican los objetivos de:

- Analizar que valores describen mejor los datos adquiridos.
- Determinar que tipología de las usadas para el diseño de redes de neuronas se adapta mejor a la aplicación propuesta.
- Diseñar e implementar la red de neuronas para la realización de predicciones de lectura de pruebas rápidas de tiras de papel.
- Evaluar el sistema mediante la predicción de 3000 datos obtenidos con muestras conocidas.

La metodología de trabajo seguida para la realización del sistema de predicción de resultados de pruebas rápidas de tiras de papel, parte de la obtención de 15000 datos de lecturas de muestras realizadas en laboratorio de las cuales se conoce el resultado.

Con las muestras obtenidas se crean dos grupos de datos, un primer grupo de 12000 muestras que se utilizan para realizar el entrenamiento de la red de neuronas, y un segundo grupo compuesto por las 3000 muestras restantes usado para el test de funcionamiento de la red de neuronas.

Para realizar los entrenamientos y test con las muestras obtenidas se diseñan ocho configuraciones diferentes de redes de neuronas donde se van combinando diferente número de las neuronas de las distintas capas como los tipos de funciones de activación y los optimizadores del sistema.

Se realizan los entrenamientos y test para las ocho diferentes configuraciones planteadas. Se comprueban y comparan los resultados obtenidos para cada una de las configuraciones, dando por valida la configuración que aporta el mejor resultado.

IV. Contribución

El sistema desarrollado aporta como contribución un método capaz de realizar una predicción de forma fiable, económica y segura del resultado de pruebas rápidas de tiras de papel, dando de esta forma una mejor y mayor utilidad a este tipo de pruebas las cuales pueden diagnosticar enfermedades o patologías en sus primeras fases, mejorando de esta manera las posibles actuaciones o diagnósticos, consiguiendo un mejor resultado en el tratamiento para las patologías o enfermedades analizadas.

V. Evaluación y Resultados

A continuación, se presentan los resultados obtenidos por las diferentes configuraciones de las redes de neuronas diseñadas para la evaluación del sistema.

Cada una de las configuraciones ha sido entrenada y testeada con los mismos datos, tanto para los datos del grupo de entrenamiento como los datos del grupo de test, para poder hacer una evaluación con los mismos datos de partida.

Evaluación 1

La primera prueba se ha realizado usando una red de neuronas con la siguiente configuración:

Capa de entrada: 17 Neuronas

Capa de normalización: BachNormalization momentum= 0.99

1ª Capa: 100 Neuronas → Función de activación Sigmoid.

2ª Capa: 100 Neuronas → Función de activación Sigmoid.

3ª Capa: 100 Neuronas → Función de activación Sigmoid.

Función Dropout: 0.5

Capa salida: 5 Neuronas → Función de activación Softmax.

Optimizador: SGD

En la Figura 1 se ve el resultado obtenido para el valor de la

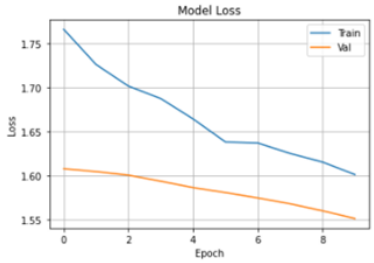


Fig. 1. Resultado de la métrica de pérdida del error

métrica de error correspondiente a la primera configuración de la red de neuronas para el conjunto de datos de test y una realización de 10 epoch.

En la Figura 2 se ve el resultado obtenido para el valor de la exactitud correspondiente a la primera configuración de la red de neuronas para el conjunto de datos de test y una realización de 10

epoch.

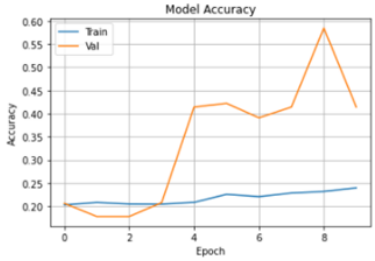


Fig. 2. Resultado de la métrica de exactitud.

Evaluación 2

La segunda prueba se ha realizado usando una red de neuronas con la siguiente configuración:

Capa de entrada: 17 Neuronas

Capa de normalización: BachNormalization momentum= 0.99

1ª Capa: 100 Neuronas → Función de activación Sigmoid.

2ª Capa: 500 Neuronas → Función de activación Sigmoid.

3ª Capa: 100 Neuronas → Función de activación Sigmoid.

Función Dropout: 0.5

Capa salida: 5 Neuronas → Función de activación Softmax.

Optimizador: SGD

En la Figura 3 se ve el resultado obtenido para el valor de la

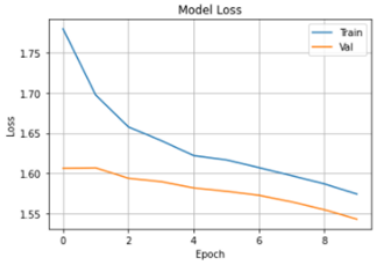


Fig. 3. Resultado de la métrica de pérdida del error

métrica de error correspondiente a la primera configuración de la red de neuronas para el conjunto de datos de test y una realización de 10 epoch.

En la Figura 4 se ve el resultado obtenido para el valor de la exactitud correspondiente a la primera configuración de la red de neuronas para el conjunto de datos de test y una realización de 10 epoch.

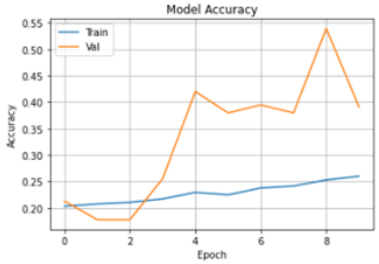


Fig. 4. Resultado de la métrica de exactitud.

Evaluación 3

La tercera prueba se ha realizado usando una red de neuronas con la siguiente configuración:

Capa de entrada: 17 Neuronas

Capa de normalización: BachNormalization momentum= 0.99

1ª Capa: 500 Neuronas → Función de activación Sigmoid.

2ª Capa: 1000 Neuronas → Función de activación Sigmoid.

3ª Capa: 500 Neuronas → Función de activación Sigmoid.

Función Dropout: 0.5

Capa salida: 5 Neuronas → Función de activación Softmax.

Optimizador: SGD

En la Figura 5 se ve el resultado obtenido para el valor de la

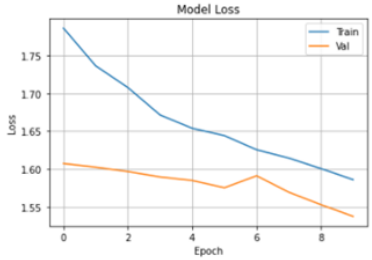


Fig. 5. Resultado de la métrica de pérdida del error

métrica de error correspondiente a la primera configuración de la red de neuronas para el conjunto de datos de test y una realización de 10 epoch.

En la Figura 6 se ve el resultado obtenido para el valor de la

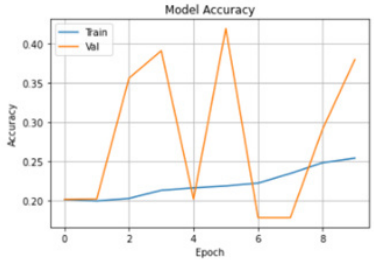


Fig. 6. Resultado de la métrica de exactitud.

exactitud correspondiente a la primera configuración de la red de neuronas para el conjunto de datos de test y una realización de 10 epoch.

Evaluación 4

La cuarta prueba se ha realizado usando una red de neuronas con la siguiente configuración:

Capa de entrada: 17 Neuronas

Capa de normalización: BachNormalization momentum= 0.99

1ª Capa: 100 Neuronas → Función de activación ReLU.

2ª Capa: 100 Neuronas → Función de activación ReLU.

3ª Capa: 100 Neuronas → Función de activación ReLU.

Función Dropout: 0.5

Capa salida: 5 Neuronas → Función de activación Softmax.

Optimizador: SGD

En la Figura 7 se ve el resultado obtenido para el valor de la

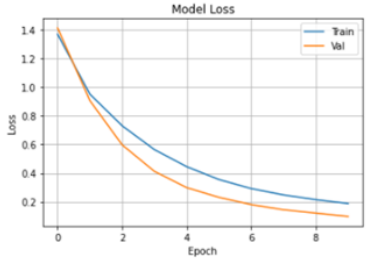


Fig. 7. Resultado de la métrica de pérdida del error

métrica de error correspondiente a la primera configuración de la red de neuronas para el conjunto de datos de test y una realización de 10 epoch.

En la Figura 8 se ve el resultado obtenido para el valor de la exactitud correspondiente a la primera configuración de la red de neuronas para el conjunto de datos de test y una realización de 10 epoch.

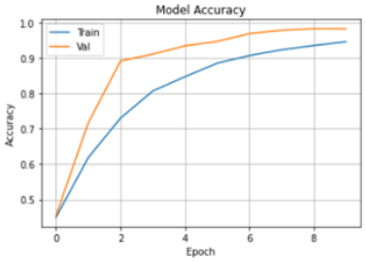


Fig. 8. Resultado de la métrica de exactitud.

Evaluación 5

La quinta prueba se ha realizado usando una red de neuronas con la siguiente configuración:

Capa de entrada: 17 Neuronas

Capa de normalización: BachNormalization momentum= 0.99

1ª Capa: 100 Neuronas → Función de activación ReLU.

2ª Capa: 500 Neuronas → Función de activación ReLU.

3ª Capa: 100 Neuronas → Función de activación ReLU.

Función Dropout: 0.5

Capa salida: 5 Neuronas → Función de activación Softmax.

Optimizador: SGD

En la Figura 9 se ve el resultado obtenido para el valor de la

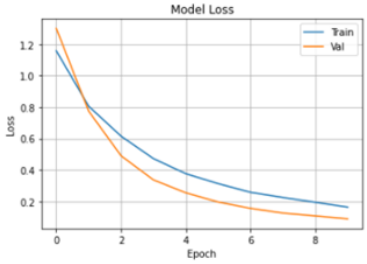


Fig. 9. Resultado de la métrica de pérdida del error

métrica de error correspondiente a la primera configuración de la red de neuronas para el conjunto de datos de test y una realización de 10 epoch.

En la Figura 10 se ve el resultado obtenido para el valor de la

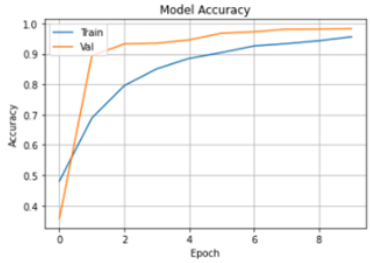


Fig. 10. Resultado de la métrica de exactitud.

exactitud correspondiente a la primera configuración de la red de neuronas para el conjunto de datos de test y una realización de 10 epoch.

Evaluación 6

La sexta prueba se ha realizado usando una red de neuronas con la siguiente configuración:

Capa de entrada: 17 Neuronas

Capa de normalización: BachNormalization momentum= 0.99

1ª Capa: 500 Neuronas → Función de activación ReLU.

2ª Capa: 1000 Neuronas → Función de activación ReLU.

3ª Capa: 500 Neuronas → Función de activación ReLU.

Función Dropout: 0.5

Capa salida: 5 Neuronas → Función de activación Softmax.

Optimizador: SGD

En la Figura 11 se ve el resultado obtenido para el valor de la

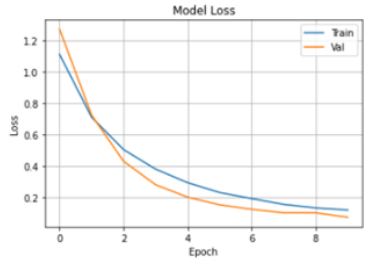


Fig. 11. Resultado de la métrica de pérdida del error

métrica de error correspondiente a la primera configuración de la red de neuronas para el conjunto de datos de test y una realización de 10 epoch.

En la Figura 12 se ve el resultado obtenido para el valor de la exactitud correspondiente a la primera configuración de la red de neuronas para el conjunto de datos de test y una realización de 10 epoch.

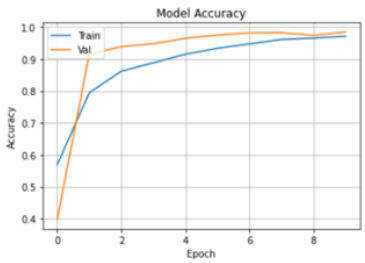


Fig. 12. Resultado de la métrica de exactitud.

Evaluación 7

La septima prueba se ha realizado usando una red de neuronas con la siguiente configuración:

Capa de entrada: 17 Neuronas

Capa de normalización: BachNormalization momentum= 0.99

1ª Capa: 100 Neuronas → Función de activación ReLU.

2ª Capa: 100 Neuronas → Función de activación ReLU.

3ª Capa: 100 Neuronas → Función de activación ReLU.

Función Dropout: 0.5

Capa salida: 5 Neuronas → Función de activación Softmax.

Optimizador: Adam

En la Figura 13 se ve el resultado obtenido para el valor de la

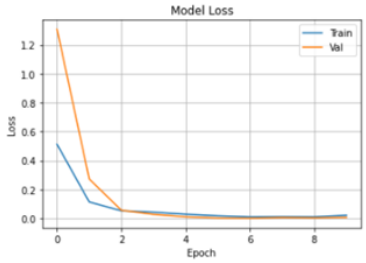


Fig. 13. Resultado de la métrica de pérdida del error

métrica de error correspondiente a la primera configuración de la red de neuronas para el conjunto de datos de test y una realización de 10 epoch.

En la Figura 14 se ve el resultado obtenido para el valor de la

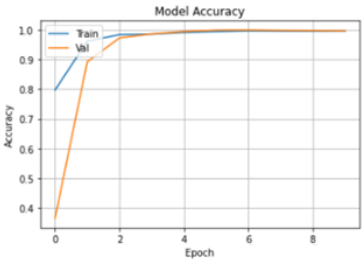


Fig. 14. Resultado de la métrica de exactitud.

exactitud correspondiente a la primera configuración de la red de neuronas para el conjunto de datos de test y una realización de 10 epoch.

Evaluación 8

La octava prueba se ha realizado usando una red de neuronas con la siguiente configuración:

Capa de entrada: 17 Neuronas

Capa de normalización: BachNormalization momentum= 0.99

1ª Capa: 25 Neuronas → Función de activación ReLU.

2ª Capa: 50 Neuronas → Función de activación ReLU.

3ª Capa: 25 Neuronas → Función de activación ReLU.

Función Dropout: 0.5

Capa salida: 5 Neuronas → Función de activación Softmax.

Optimizador: Adam

En la Figura 14 se ve el resultado obtenido para el valor de la

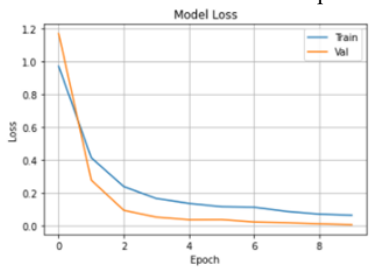


Fig. 14. Resultado de la métrica de pérdida del error

métrica de error correspondiente a la primera configuración de la red de neuronas para el conjunto de datos de test y una realización de 10 epoch.

En la Figura 15 se ve el resultado obtenido para el valor de la

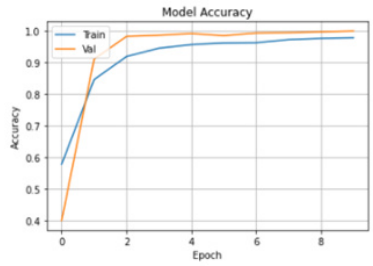


Fig. 15. Resultado de la métrica de exactitud.

exactitud correspondiente a la primera configuración de la red de neuronas para el conjunto de datos de test y una realización de 10 epoch.

VI. Discusión

De los resultados obtenidos de las ocho evaluaciones realizadas, se observa que existe una mejora considerable cuando se cambia la función de activación de tipo Sigmoid a tipo ReLU. Esto lo podemos ver tanto por el valor de la métrica del error como el de la precisión, donde el valor máximo conseguido con la función Sigmoid ha sido del 26%, mientras que con la función ReLU se han conseguido valores cercanos al 100% para todas las evaluaciones realizadas.

Dentro del grupo de evaluaciones realizadas con la función ReLU (evaluaciones 4, 5, 6, 7, 8) se observa que el incremento del número de neuronas no aportaba un cambio significativo en los resultados como se puede ver en las evaluaciones 4, 5 y 6 donde se han ido incrementando el número de neuronas sin obtener grandes cambios o mejoras. Lo que se ha observado es que el cambio del tipo de optimizador sí que ha tenido un efecto sobre los resultados, mejorando los resultados cuando se ha aplicado el optimizado Adam llegando a un valor cercano al 100% de una forma rápida, reduciendo el número de epoch necesarios para llegar a estos valores de exactitud.

Por último, se comprueba que una reducción del número de neuronas en las capas intermedias con la función de activación ReLU y el optimizador Adam como se ha realizado en la evaluación 8, aporta unos resultados muy parecidos a la evaluación 7 que contaba con un número mayor de neuronas. Por lo que se opta por tomar como la configuración optima la realizada en la evaluación 8 ya que nos da valores de error y exactitud aceptables reduciendo el coste computacional en comparación con la evaluación 7.

VII. Conclusiones

Después de las pruebas realizadas y los resultados obtenidos se concluye que el sistema aporta un grado de exactitud cercano al 95% por lo que se puede tomar como un sistema valido y fiable para la generación de predicciones de pruebas rápidas de tiras de papel.

De los resultados obtenidos se concluye que los valores descriptivos que se han tomado para los datos de lectura de las pruebas rápidas consiguen una buena caracterización de estos por lo que se pueden realizar buenas predicciones con ellos. De estos mismos resultados se puede concluir que el algoritmo usado se adapta perfectamente a la funcionalidad que se estaba buscando, aportando una exactitud bastante aceptable, con un error bajo y un coste computacional bajo.

Se tiene que destacar que todas las mediciones se realizaron con pruebas de laboratorio controladas con tiras de papel creadas con reactivos de coloración rojiza, y con dispositivos captadores dotados con fotodiodos. Por lo que se plantea para posibles trabajos futuros la realización de mediciones con reactivos de diferentes coloraciones, así como la realización de las mediciones con distintos elementos captadores, lo que daría pie a usar diferentes topologías de las redes de neuronas y configuraciones de estas.

Referencias

[1] Alonso, L. y Calonge, T. (2001). Redes neuronales y reconocimiento de patrones. En L. Alonso, "Reconocimiento de patrones con redes neuronales". Salamanca: Imprenta Cardenal.

[2] Baxes, G. A. 1994. Digital Image Processing - Principles and Applications. U.S.A. John Wiley & Sons.

[3] Camargo-Vega, Juan José; Camargo-Ortega, Jonathan Felipe; Joyanes-Aguilar, Luis Conociendo Big Data Facultad de Ingeniería, vol. 24, núm. 38, enero-junio, 2015, pp. 63-77 Universidad Pedagógica y Tecnológica de Colombia Tunja, Colombia.

[4] Diaz Cabrera, J.M 2008, Análisis de los sistemas de adquisición de datos en los sistemas automáticos del currículo de tecnología industrial II. ISSN 1988-6047 DEP.LEGAL: GR 2922/2007 Nº5 - Abril de 2008.

[5] Ferré Baldrich, J. (2004), Calibración multivariable en análisis cuantitativo, el modelo directo. Técnicas de laboratorio, 297, 986-989.

[6] F.J, Muñoz, P.A, Marín, J.I. (2006), Reconocimiento de comandos de voz usando la transformada wavelet y máquinas de vectores de soporte. Scientia et Technica, 2, 35-40.

[7] Fontal, B. (2005), El espectro magnético y sus aplicaciones. Venezuela: Escuela venezolana para la enseñanza de la química.

[8] González Henríquez, J.J. 2005, El histograma con la TI-92: optimización de clases. Revista de didáctica de las matemáticas,61,67-72.

[9] Kuri-Morales, A. and Galaviz-Casas, J. (2002). Algoritmos Genéticos, Fondo de Cultura Económica/UNAM/IPN.

[10] L. Al Shalabi and Z. Shaaban, "Normalization as a Preprocessing Engine for Data Mining and the Approach of Preference Matrix," 2006 International Conference on Dependability of Computer Systems, Szklarska Poreba, 2006, pp. 207-214, doi: 10.1109/DEPCOS-RELCOMEX.2006.38.

[11] Luna-Ortega, Carlos & Mora-González, Miguel & Martíunez Romo, Julio César & Luna-Rosas, Francisco & Muñoz-Maciel, Jesús. (2013). Reconocimiento del habla mediante el uso de la correlación cruzada y una perceptrón multicapa. Nova scientia. 6. 108-124. 10.21640/ns.v6i12.26.

[12] Richards, J. A. 1995. Remote Sensing Digital Image Analysis - An Introduction. Berlín. Springer-Verlag.

[13] R. López and J. Fernández, Las Redes Neuronales Artificiales, Netbiblo, 2008.

[14] Sabadías, A.V. Estadística descriptiva e inferencial. Recuperado de https://books.google.es/books?id=RbaC-wPWqjsC.