REDES DE NEURONAS ARTIFICIALES

PRACTICA 2-1

Universidad Carlos III Madrid, Ingeniería Informática, 2021

IVAN AGUADO PERULERO – 100405871
JORGE SERRANO PÉREZ – 100405987

Contenido

1.- INTRODUCCIÓN

El objetivo de esta primera parte de la práctica es abordar un problema de clasificación clásico con el Perceptrón Multicapa (PM). El problema en cuestión es el de clasificar vehículos por su silueta ("*Vehicle Silhouettes*"). Para ello se nos ha proporcionado un conjunto de datos con 846 instancias, 18 atributos enteros y 4 clases que corresponden a los distintos tipos de vehículos (OPEL, SAAB, BUS, VAN). Tendremos que realizar una previa preparación de los datos antes de comenzar el aprendizaje de las redes. Además, se nos ha proporcionado un script en *Python* que deberemos modificar para experimentar correctamente.

El presente documento se divide en un capítulo que explica cómo se ha realizado el preproceso de los datos, otro capítulo para la experimentación realizada, los resultados obtenidos y un análisis de los resultados, y finalmente un capítulo con las conclusiones obtenidas.

2.- PREPARACIÓN DE DATOS Y MEJORAS DEL SCRIPT

Para la preparación de los datos hemos utilizado el programa Weka, que nos permite aplicar distintos filtros a nuestro fichero de datos. Para la correcta lectura del fichero por parte del programa tuvimos que hacer cambios sobre el original, eliminando la cabecera.

A continuación, una vez abierto el fichero desde Weka, aplicamos el filtro *Normalize*, el cual es un filtro de la categoría atributo de no supervisado. Este aplicará la transformación lineal necesaria para acotar las variables dentro del rango [0, 1]. Tras esto, para aleatorizar los datos utilizamos *Randomize*, filtro en la sección del tipo no supervisado. Esto nos permite desordenar los datos para que el entrenamiento de la rede se realice correctamente.

El último paso fue dividir el fichero en dos diferentes uno para el entrenamiento y test. El primero de ellos contendría ⅔ de los datos y el otro los restantes. Más tarde el código proporcionado dividirá el fichero de test en 80% entrenamiento y 20% para validación.

Cabe destacar que el script proporcionado realiza la codificación (one-hot encoding) de la salida deseada, de esta manera se podrá llevar a cabo un problema de clasificación, en este caso con 4 clases.

En cuanto a las mejoras realizadas sobre el script básico, una la hemos realizado una en el apartado de guardar los datos de entrenamiento en ficheros. Al ejecutar el programa nos dimos cuenta de que no se devolvía el número de ciclos óptimo, es decir, en el que se consigue la mínima *loss* de validación. Debido a esto añadimos una nueva funcionalidad que extraía el número del ciclo donde la *loss* mencionada era mínima, una vez teníamos ese número lo utilizamos como índice, para extraer el resto de datos *loss* y *accuracy* en ese ciclo, de cara a poder obtener información más precisa durante el entrenamiento y poder plasmarlo en la memoria. Por otro lado, tras evaluar el modelo final hemos sacado las salidas de este y las esperadas para posteriormente disponerlas en un fichero de texto como se solicita.

De cara a realizar experimentos con más de una capa oculta en el apartado “definir el modelo” simplemente habría que poner tantas veces como capas se requiera la línea de código que las crea.

3.- PERCEPTRÓN MULTICAPA

Para realizar la experimentación y decidir cuál es el mejor, se ha utilizado como medida del error el *MSE* (*Medium Squared Error*).

El error de validación determina el número de ciclos óptimo, es decir, a partir del cual dicho error sube. Por lo tanto, los valores de los errores mostrados más abajo son del ciclo con el error de validación mínimo.

En la columna ciclos de la siguiente tabla se detalla el número de ciclos con el que hemos experimentado y entre paréntesis el ciclo óptimo en el que recogemos los errores y del que sacamos el modelo, en caso de que dicho experimento sea el que posee mejores resultados. Este será aquel que posea el valor *loss* más bajo en el conjunto de validación.

En la columna del número de neuronas entre paréntesis tiene el número de capas ocultas del Perceptrón Multicapa.

Experimentación realizada

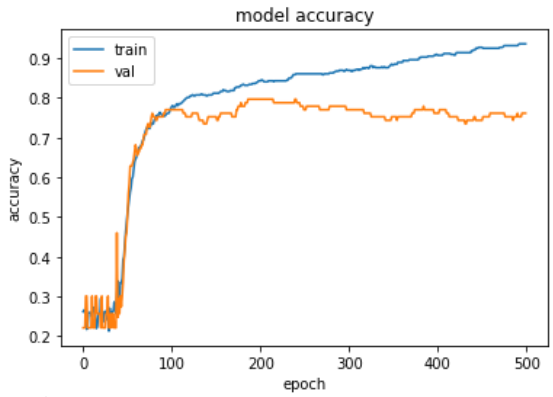
En la siguiente tabla se muestran los resultados obtenidos en el orden de experimentación que hemos seguido. Primero hemos modificado el factor de aprendizaje y sus respectivos ciclos, luego el número de neuronas, y más adelante el número de capas. Más abajo se explicará más detalladamente.

Learning Rate	N. neuronas(capas)	Ciclos	Accuracy (Entrenamiento)	Loss (Entrenamiento)	Accuracy (Validación)	Loss (Validación)
0.5	350(1)	100(100)	0.767184	0.079335	0.769911	0.085107
0.5	350(1)	500(192)	0.835920	0.059360	0.796460	0.074812
0.2	350(1)	100(100)	0.769401	0.083829	0.743362	0.089559
0.2	350(1)	500(436)	0.855875	0.052037	0.805309	0.070518
0.1	350(1)	1000(1000)	0.895787	0.040339	0.823008	0.071146
0.1	350(1)	2000(1687)	0.929046	0.031720	0.831858	**0.059067**
0.01	350(1)	1000(1000)	0.924611	0.805309	0.033606	0.067426
0.01	350(1)	5000 (5000)	0.957871	0.023950	0.823008	0.064317
0.1	175(1)	2000(19)	0.940133	0.028229	0.867256	0.062691
0.1	30(1)	1700(1636)	0.953436	0.020499	0.831858	0.068397
0.1	500(1)	3000(43)	0.217294	0.389456	0.300884	0.349345
0.1	150(2)	3000(904)	0.909090	0.037108	0.778761	0.062081
0.1	350(2)	2500(1056)	0.904656	0.037388	0.769911	0.073298
0.01	350(2)	3000(3000)	0.827050	0.058979	0.752212	0.083454
0.01	150(2)	5000(5000)	0.884700	0.043998	0.787610	0.070350
0.1	150(3)	1500(450)	0.800443	0.066896	0.761061	0.076284
0.1	350(3)	1500(1485)	0.871396	0.047813	0.743362	0.077371

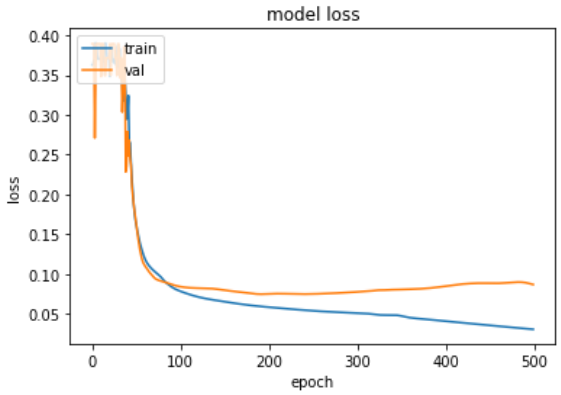
Tabla 1.

Resultados obtenidos

Para comenzar la experimentación probamos un factor de aprendizaje de 0.5 y con 350 neuronas en una única capa oculta. Una vez obtuvimos los resultados bajamos el factor de aprendizaje, pues este era muy alto y realizaba un aprendizaje rápido, pero quizás no de la manera óptima. Aquí podemos observar la *loss* y la *accuracy* para este experimento con 500 ciclos, y se demuestra que a partir del ciclo mostrado en la tabla el valor en la gráfica asciende de nuevo.

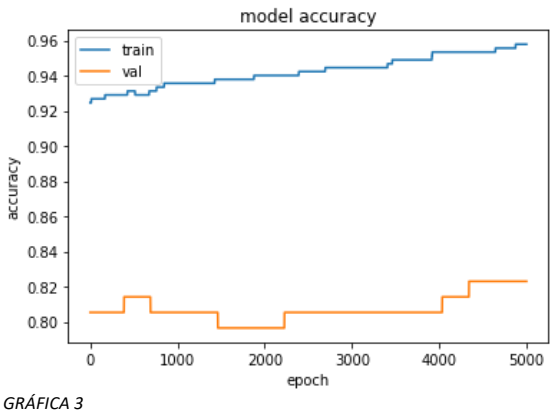


GRÁFICA 1

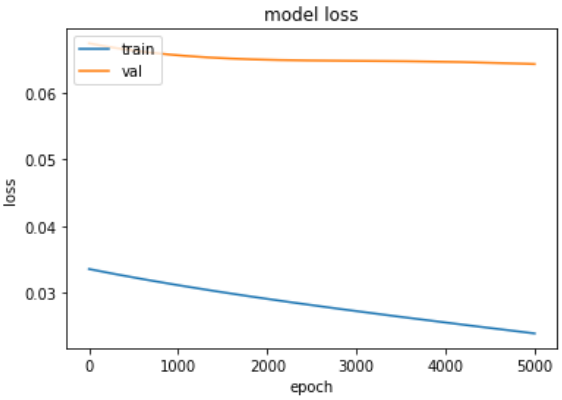


GRÁFICA 2

A continuación, fuimos bajando el factor de aprendizaje hasta 0.01, todavía con 350 neuronas en una sola capa oculta. Aquí se puede ver un ejemplo de experimento realizado con 5000 ciclos, pero como dijimos anteriormente en la tabla, no llegaba a aprender del todo porque alcanzaba el número de ciclos máximo y el valor de *loss* seguía bajando en el conjunto de validación. Por ello, debido a que tardaba tanto el aprendizaje, decidimos cambiar de estrategia.

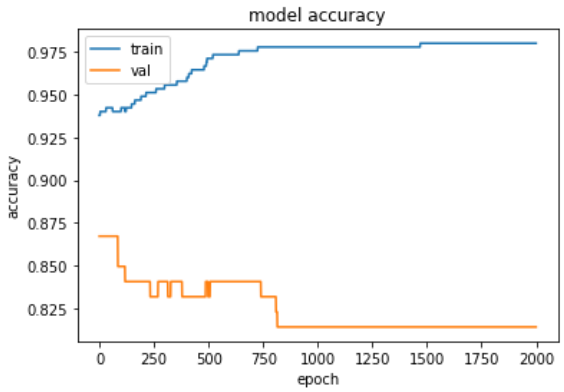


GRÁFICA 3

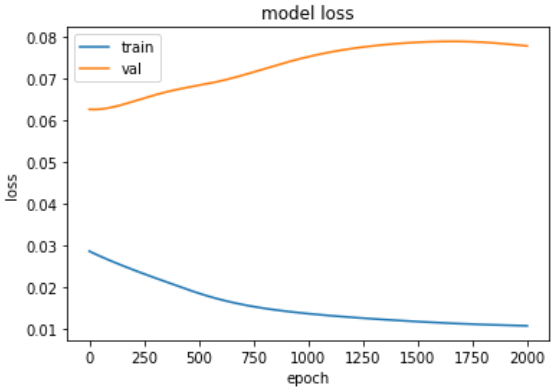


GRÁFICA 4

A continuación, pasamos a modificar y variar el número de neuronas. Utilizamos un factor de aprendizaje de 0.1 y primero bajamos a 175 neuronas. Utilizando entonces 2000 ciclos vemos que dan unos resultados bastante buenos y a pesar de haber aprendido muy rápidamente, llegando en el ciclo 19 al valor de 0.062691. Se aprecia en la segunda gráfica como el error de validación empieza a subir casi al principio y, por lo tanto, se empieza a realizar sobreaprendizaje.

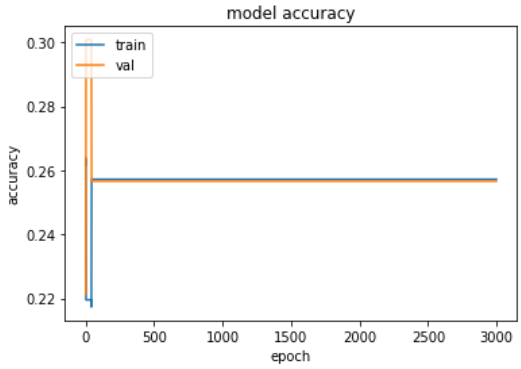


GRÁFICA 5

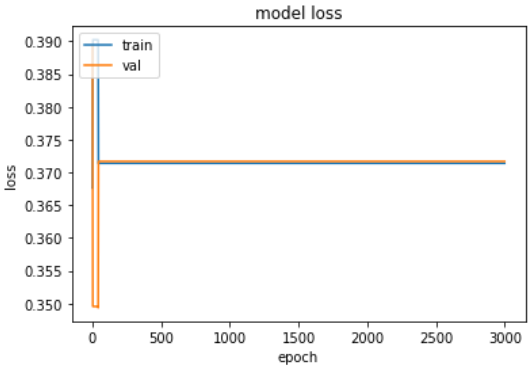


GRÁFICA 6

Bajamos más el número de neuronas, pero no mejoraban los resultados. Por lo tanto, decidimos subirlo a ver si de esta manera sí lo hacía. En las siguientes gráficas observamos la evolución de la *accuracy* y la *loss* para un experimento con 500 neuronas y 0.1 de factor de aprendizaje. Como se puede apreciar, los resultados son pésimos, ya que, al tener un número tan alto de neuronas, aprende muy rápido (alcanza el mínimo valor en el ciclo 43 de 3000), pero de manera incorrecta, dando un valor de *loss* de 0.349345.

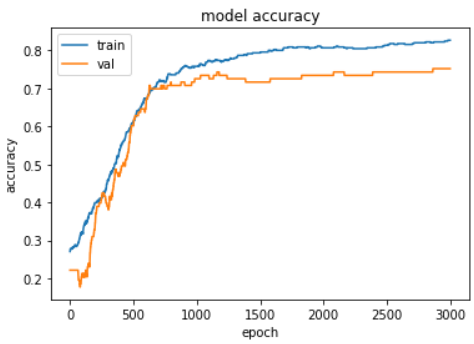


GRÁFICA 7

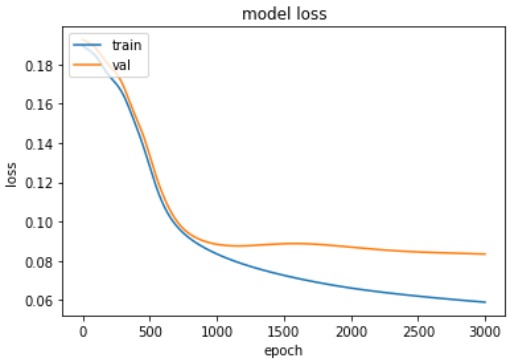


GRÁFICA 8

El siguiente paso que dimos fue cambiar ahora el número de capas, pasando a dos. Probamos primero con un factor de aprendizaje de 0.1, que es el que mejores resultados nos había dado hasta el momento, y luego primero 150 neuronas en cada una de las dos capas y a continuación 350.



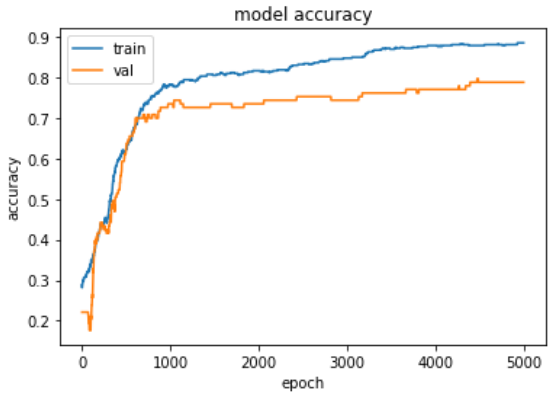
GRÁFICA 9



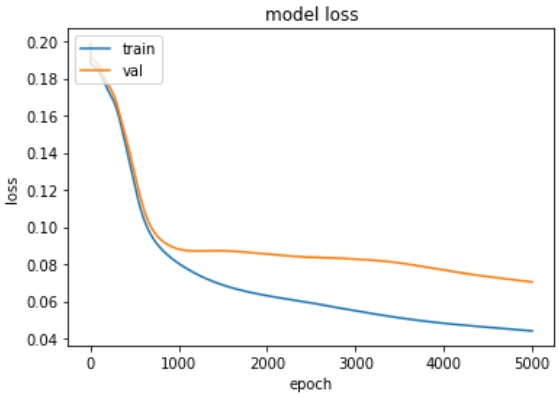
GRÁFICA 10

En las gráficas mostradas más arriba se puede observar la evolución de este último experimento, que nos dio peor resultado que el anterior mencionado con un valor de *loss* en validación de aproximadamente 0.07 frente a 0.06. Esto nos lleva a pensar que quizá si hubiéramos bajado más el número de neuronas, habría llegado a ser el mejor experimento.

El siguiente factor a probar fue 0.01, que parecía mejor que otros más altos, también con dos capas de neuronas. Sin embargo, los resultados no mejoraban en ningún caso.



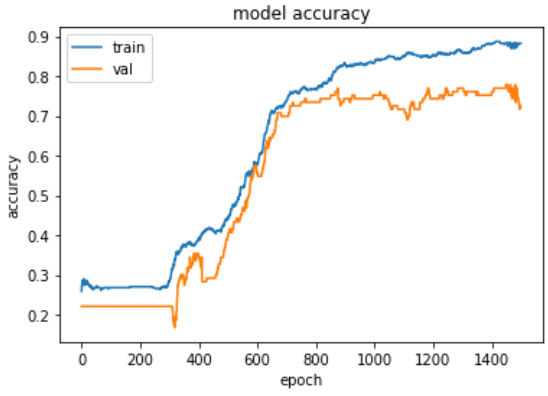
GRÁFICA 12



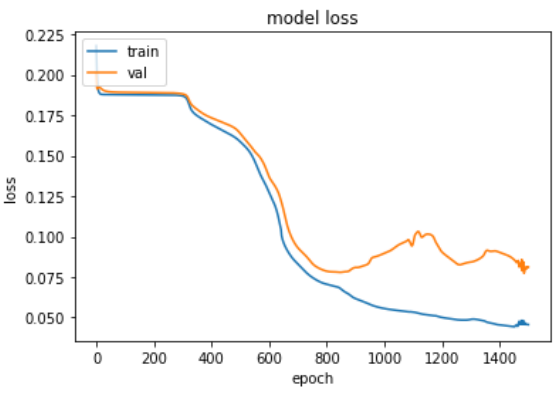
GRÁFICA 13

En las gráficas se puede observar que necesitaba un número de ciclos demasiado alto para aprender correctamente (de hecho, no alcanzábamos el valor óptimo), por lo que decidimos no seguir experimentando con estos valores.

Finalmente, pasamos a probar con 3 capas de neuronas y un factor de aprendizaje de 0.1. Probamos con un número bajo (150) y un número alto (350) de neuronas, pero en ningún caso mejoraban los resultados anteriores. A continuación, se pueden ver las gráficas de la *accuracy* y la *loss* para este último experimento.



GRÁFICA 14

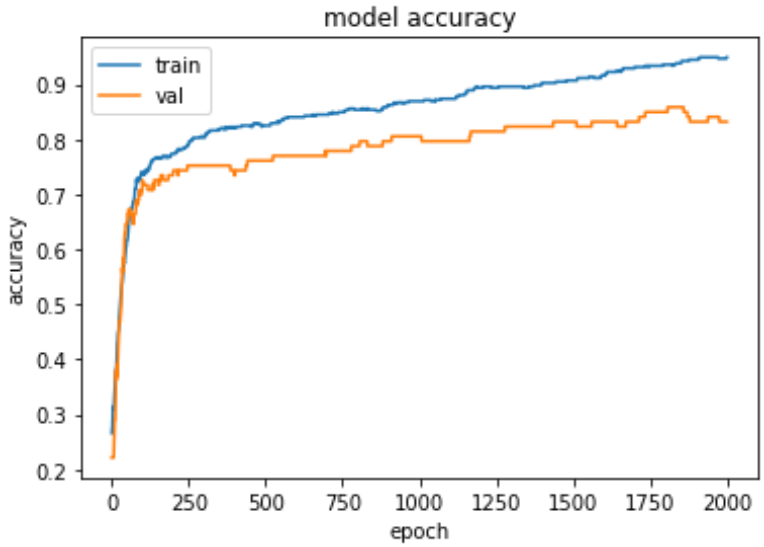


GRÁFICA 15

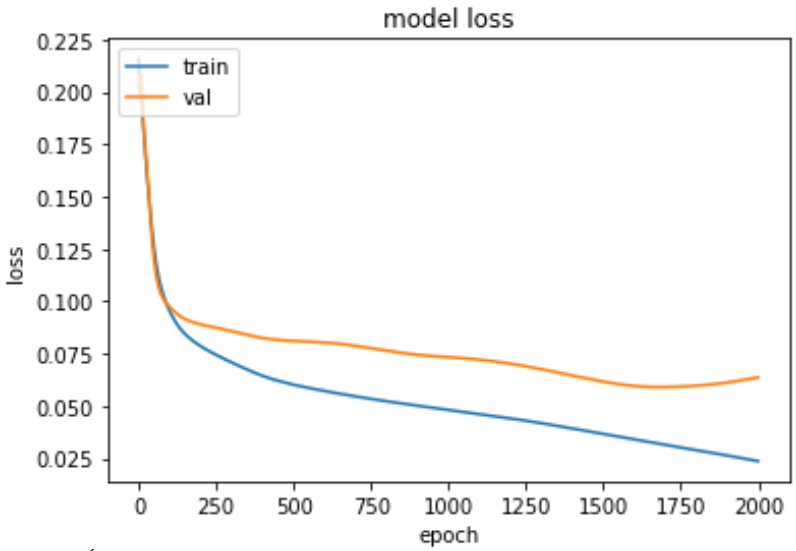
Vemos que el error en el conjunto de validación empieza a ascender en el ciclo 1485, lo que significa que se empieza a realizar sobreentrenamiento.

Análisis de los resultados del modelo final

Finalmente, el mejor experimento que hemos obtenido ha sido con un factor de aprendizaje de 0.1, una sola capa oculta que contaba con 350 neuronas y un número de ciclos óptimo de 1687. Las gráficas de la evolución de la *accuracy* y la *loss* se muestran a continuación.



GRÁFICA 16



GRÁFICA 17

Como se puede observar, la *accuracy* va ascendiendo en cada ciclo, lo que tiene sentido porque significa que se está realizando el entrenamiento correctamente y está aumentando cada vez más la precisión. Por otro lado, en la gráfica de la *loss*, se aprecia como en el ciclo óptimo mencionado anteriormente (el 1687 de 2000) empieza a subir de nuevo el error en el conjunto de validación. Esto quiere decir que a partir de aquí se empieza a realizar sobreentrenamiento.

Con los hiperparámetros mencionados construimos y entrenamos el modelo final, esta vez sin separar datos de entrenamiento y validación, utilizando todos ellos para lo primero. Una vez entrenado se pasan los datos de test, obteniendo la siguiente matriz de confusión:

```
[74  0  1  1]
[ 0 30 37  1]
[ 2  5 63  2]
[ 2  1  1 62]
```

Matriz de confusión 1

La primera columna y fila se corresponden con la clase bus, las segundas con opel, las terceras con saab y las cuartas con van. Dicho esto, podemos observar que en general se realiza una buena clasificación, pues la diagonal principal contiene valores altos y el resto de celdas valores pequeños. Sin embargo, esto no pasa con la clase saab la cual se clasifica mal confundiéndose con la de opel en la mayoría de las ocasiones. Esto puede ser debido a un desbalance de los datos con esa clase, o alguna otra característica de dichas instancias.

Aun así, se consigue una precisión general de 0.812056, un valor alto que nos informa de que la mayoría de los datos se clasifica de forma correcta.

4.- CONCLUSIONES

Durante la realización de la práctica hemos podido observar que para este problema en concreto lo mejor es un factor de aprendizaje medio, ni muy alto ni muy bajo. Si este es bajo, el entrenamiento se realiza de una forma muy lenta y puede conllevar varias horas para, ni aun así, conseguir unos resultados destacables. Por el contrario, si el factor de aprendizaje es alto el entrenamiento se realiza de forma rápida, pero no óptima, pues los resultados en cuanto al *loss* de validación no son los mejores.

En cuanto al número de neuronas en la capa oculta vemos que por regla general los mejores resultados se consiguen con valores medios y bajos, pues la *loss* de validación aumenta cuando aumentamos este parámetro.

Por último, en cuanto al número de capas ocultas a utilizar, añadir más de una no nos ayuda a conseguir un mejor modelo, de hecho, el seleccionado solo cuenta con una de estas.