Universidad Nacional de Rosario
Facultad de Ciencias Exactas,
Ingeniería y Agrimensura
Departamento de Ciencias de la
Computación

Introducción a la Inteligencia Artificial

Trabajo Práctico 4

Cerruti Lautaro
Poma Lucas

Junio de 2023

1. Ejercicio 1

En este ejercicio trabajamos sobre un dataset de información crediticia y el objetivo es categorizar a las personas como 'good' o 'bad' en base a que tan buenos candidatos son para otorgarles un préstamo. El conjunto cuenta de 1000 instancias donde el 70 % de las mismas están categorizadas como 'good'.

1.1. Preparación de los datos

Para la carga del dataset utilizamos el paquete pandas. Para normalizar los datos a tipo numérico utilizamos OrdinalEncoder.

1.2. Entrenamiento de un modelo

1.2.1. a

Entrenando un árbol con los parámetros por defecto obtenemos un accuracy del 69 % sobre el conjunto de prueba pero del 100 % sobre el conjunto de entrenamiento. Esta disparidad nos muestra un claro overfitting.

1.2.2. b

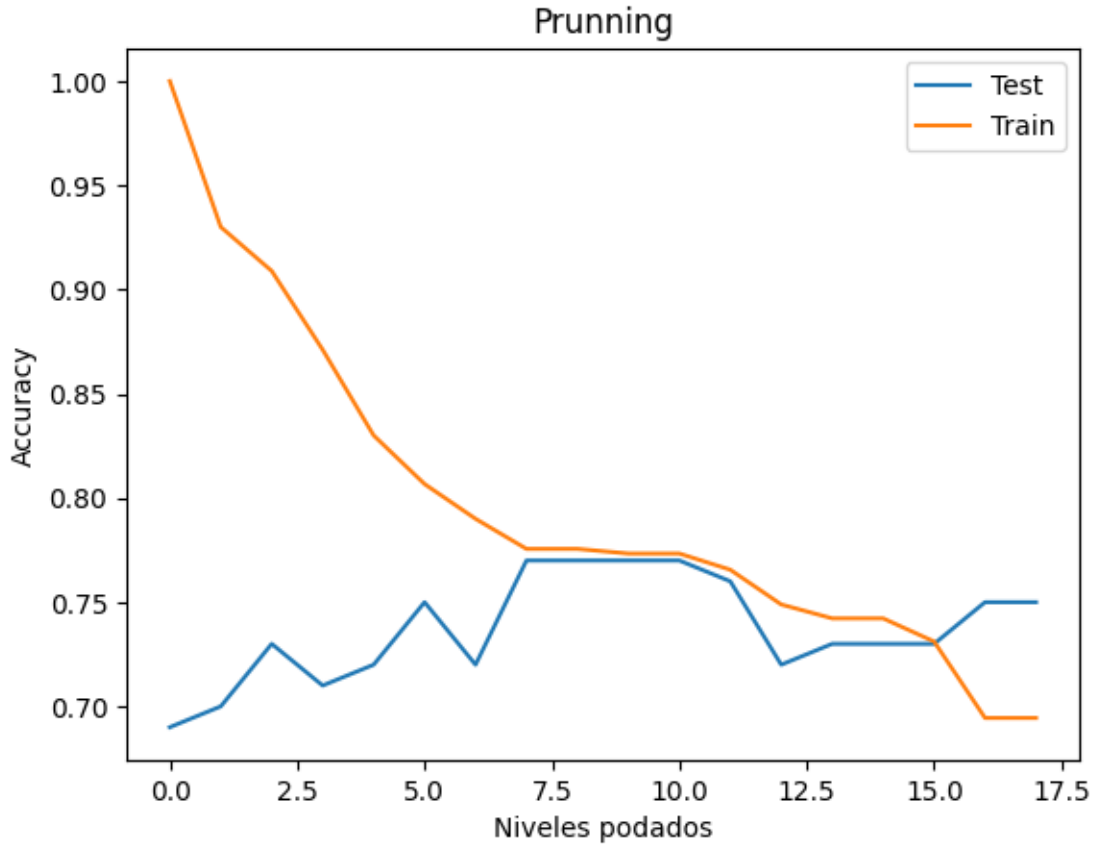
Intentamos buscar una combinación de parámetros óptimos con GridSearchCV, jugando con el rango de `max_depth`, `min_samples_leaf`, `min_samples_split` y `criterion`. El árbol resultante es

```
DecisionTreeClassifier(max_depth=5, min_samples_leaf=32, min_samples_split=2,
criterion='gini');
```

Con este árbol se obtiene un accuracy de 76 % sobre el conjunto de prueba y un 75,88 % sobre el conjunto de entrenamiento. Siendo este un modelo mucho mas general.

1.2.3. c

En este apartado realizamos poda por niveles desde el árbol con parámetros por defecto.



Podemos ver que al podar entre 7 y 10 niveles del árbol se obtiene la mayor accuracy sobre el conjunto de prueba (77 %) a la vez que la diferencia con el conjunto de entrenamiento es muy chica. El punto óptimo seria una poda de 9 o 10 niveles, donde el accuracy sobre el conjunto de prueba es 77 % y en el conjunto de entrenamiento 77,33 %.

1.3. Evaluación de modelo

1.3.1. a

En el modelo que estamos evaluando nos interesaría principalmente no tener Falsos Positivos ya que esto implicaría darle prestamos a personas cuya capacidad financiera es dudosa. Las mejores medidas que podríamos usar son precision, fallout y specificity.

Cuando la precision tiende a 1 tenemos que la cantidad de false positives que es nuestra mayor preocupación, sera mucho menor que la cantidad de true positives.
Por otro lado cuando el fallout tiende a 0 nos indica que sobre el conjunto de negativos, la cantidad de falsos positivos es muy baja.
Por ultimo, utilizando specificity cuando esta tiende a 1, nos dice que la cantidad de negativos que fueron categorizados correctamente es mucho mayor que la cantidad de falsos positivos. Si tuviésemos que elegir una en particular creemos que esta seria la mas indicada, pero consideramos que no se puede utilizar sola, ya que si obtenemos un modelo que clasifique todo como negativo, esta medida da como resultado 1.

Al momento de optimizar los parámetros la única que ya viene por defecto por parte de Scikit learn es precision. Por lo que decidimos trabajar con esta.

1.3.2. b

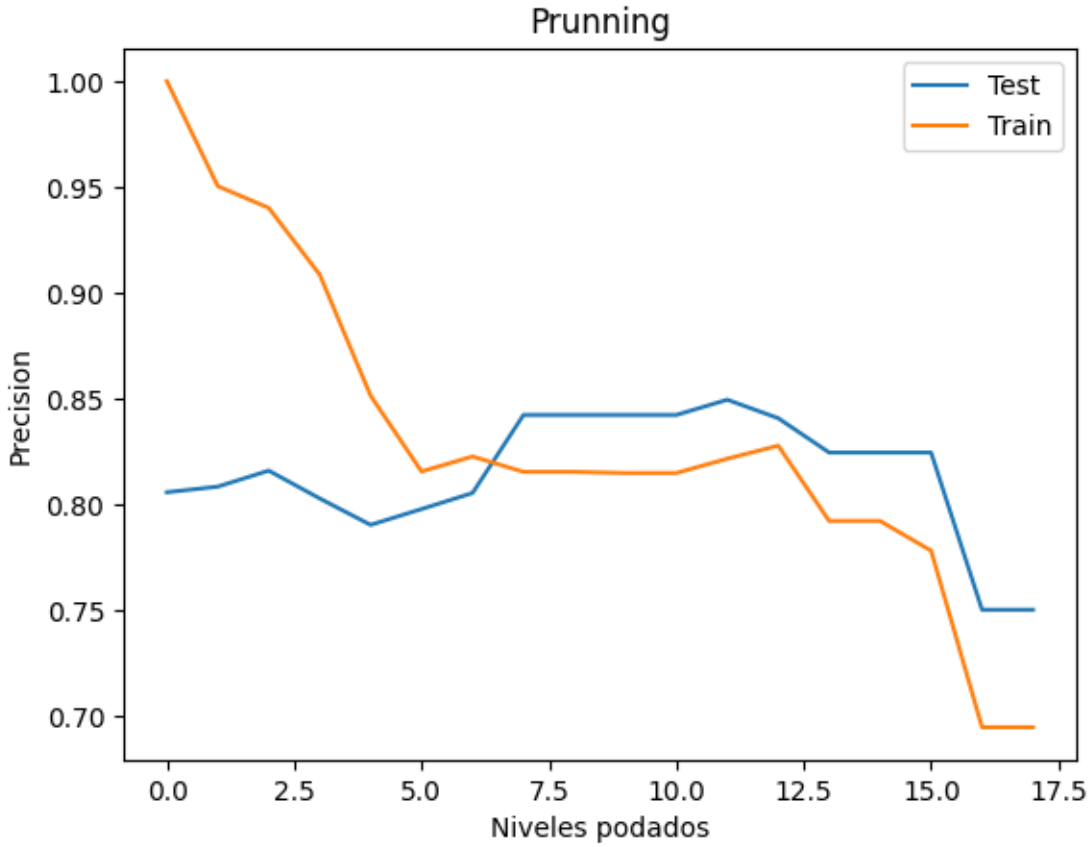
Evaluando los 2 modelos anteriores con la precision obtenemos en el caso del generado con GridSearchCV un 84,93 %, mientras que, el que obtuvimos luego de realizar la poda un 84,21 %.

Realizando la busqueda de parametros con GridSearchCV pero utilizando precision como medida obtenemos el siguiente árbol como mejor:

```
DecisionTreeClassifier(max_depth=8, min_samples_leaf=4, min_samples_split=15,
criterion='entropy');
```

Donde la precision fue de 82,05 %. Los parámetros son distintos aquí tenemos un árbol con mayor profundidad y distinto criterio de generación.

Realizando la poda sobre el original pero calculando la precision se obtiene el siguiente gráfico:

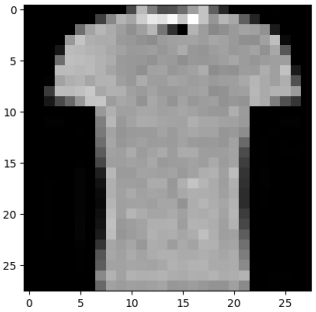


Donde el mejor resultado fue con una poda de 11 niveles, obteniendo una precisión del 84,93 %. Podemos ver que esto es un nivel poda mas que cuando utilizamos como medida la accuracy.

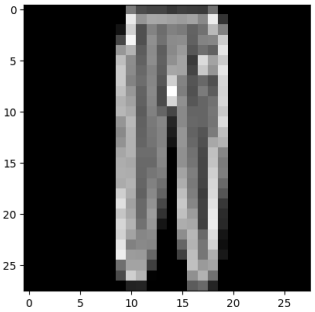
2. Ejercicio 2

2.1. Análisis de los datos

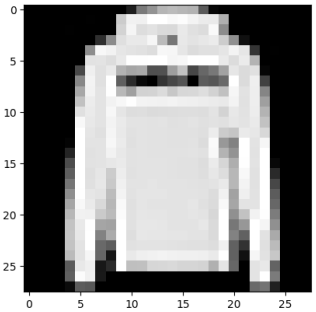
Analizando el conjunto de datos vemos que tenemos 10 clases en total, identificadas con números del 0 al 9, en el conjunto de entrenamiento tenemos 60000 elementos distribuidos equitativamente en cada clase, es decir 6000 de cada una. En el conjunto de prueba se tienen 10000 elementos, 1000 de cada clase.



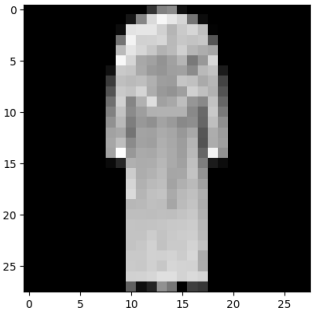
Remera



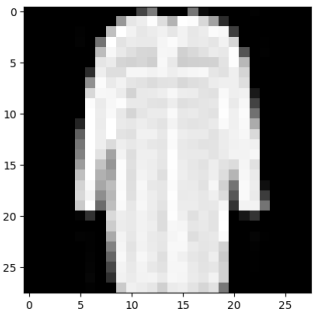
Pantalon Largo



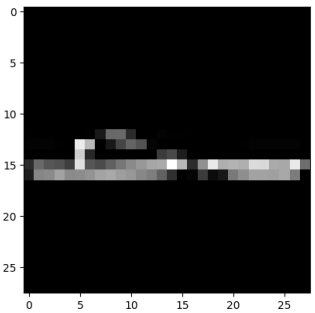
Pullover



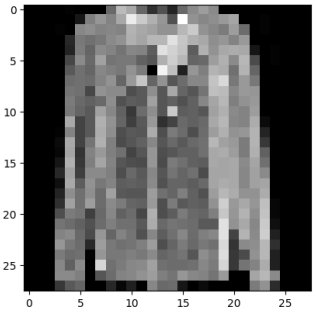
Vestido



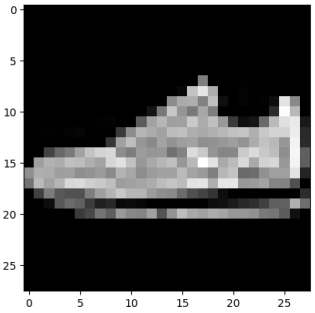
Abrigo



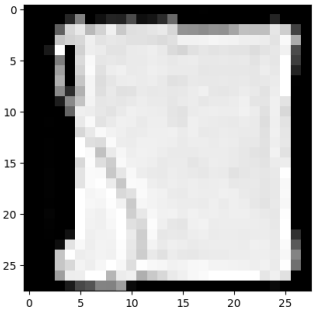
Sandalia



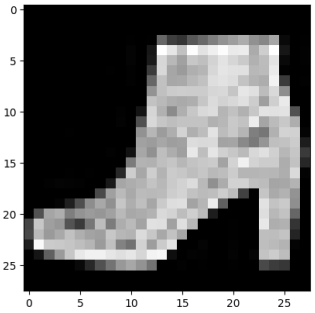
Camisa



Zapatilla



Mochila

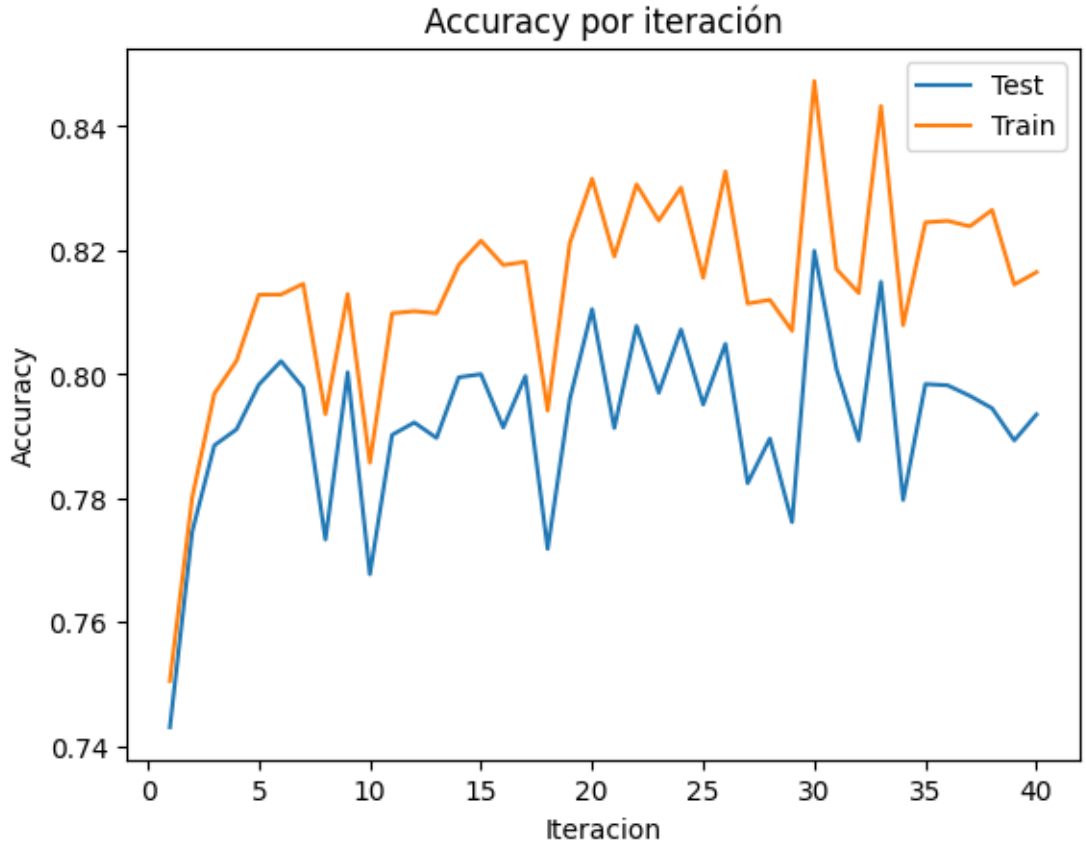


Zapato

2.2. Redes Neuronales con distintas cantidades de capas

2.2.1. Red sin capas ocultas

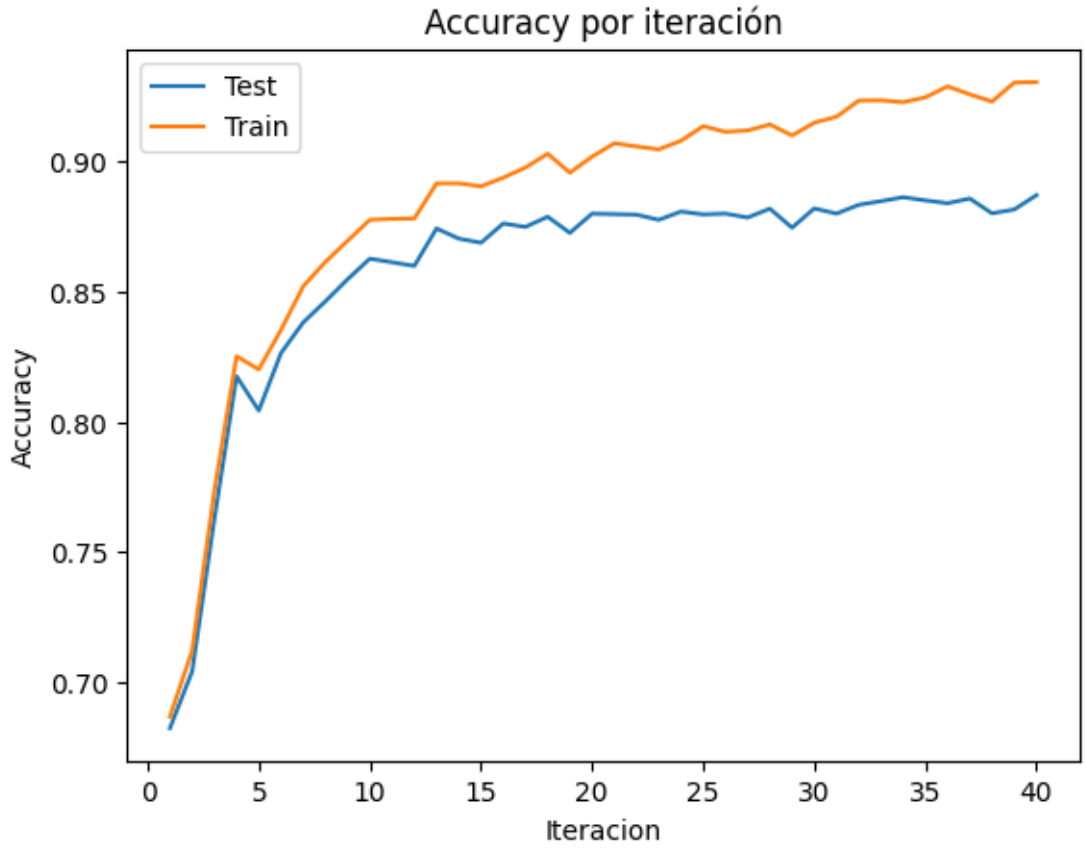
La primera red neuronal que entrenamos no tiene capas ocultas, los acuraccy sobre las distintas iteraciones del entrenamiento fueron los siguientes:



Obteniendo en la iteración numero 40 un accuracy de 79,35 % sobre el conjunto de prueba y de 81,64 % sobre el de entrenamiento. Sobre esta red podemos ver que al no tener capas ocultas no se estabilizó el aprendizaje y esto se refleja en los saltos abruptos que se aprecian en la gráfica.

2.2.2. Red con 2 capas ocultas

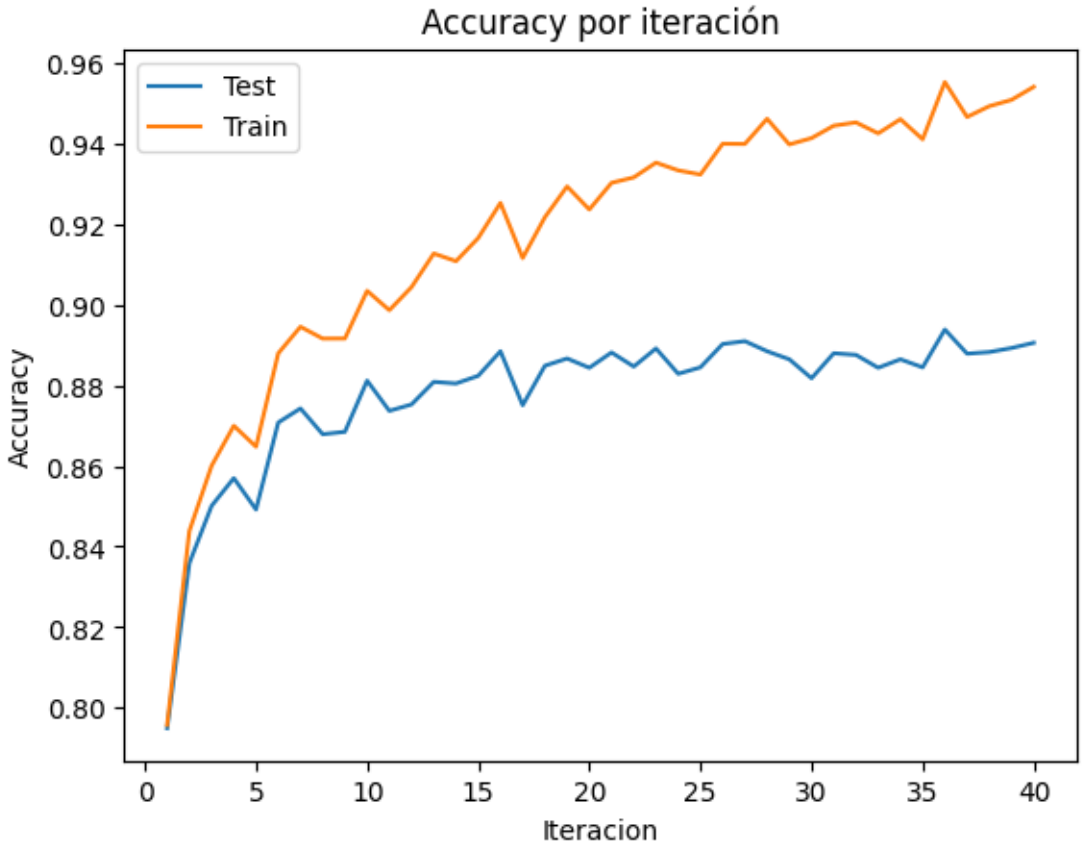
En la segunda red que entrenamos, que contaba de 2 capas ocultas, una de 100 neuronas y otra de 50 neuronas, se obtiene el siguiente gráfico:



Obteniendo en la iteración numero 40 un accuracy de 88,71 % sobre el conjunto de prueba y de 93,06 % sobre el de entrenamiento. Esta red tiene mucho mejor accuracy que la anterior pero a mas épocas, mayor es el riesgo de overfitting, el cual consideramos como una diferencia del 5 % entre los accuracys de los conjuntos.

2.2.3. Red con 6 capas ocultas

En la tercera red que entrenamos, que contaba de 6 capas ocultas, tres de 100 neuronas y otras tres de 50 neuronas, se obtiene el siguiente gráfico:



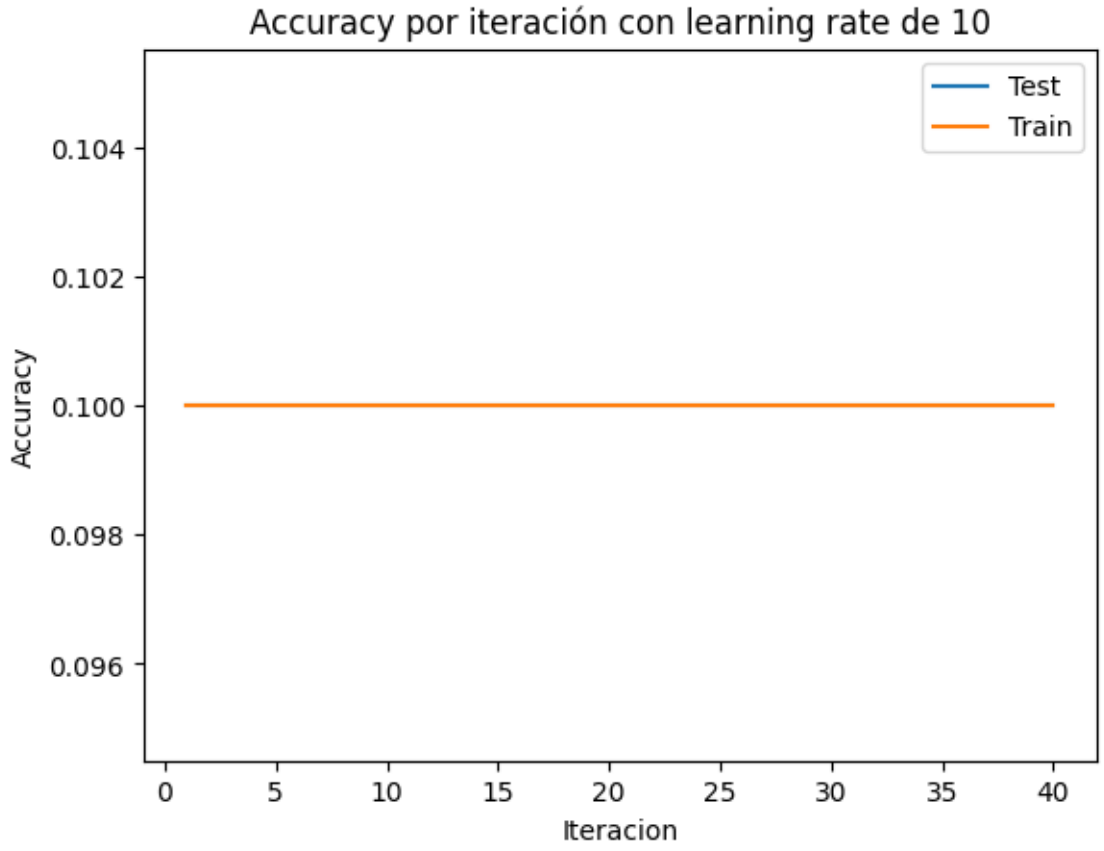
Obteniendo en la iteración numero 40 un accuracy de 89,06 % sobre el conjunto de prueba y de 95,42 % sobre el de entrenamiento. En esta se obtuvo una leve mejora del accuracy sobre el conjunto de prueba pero podemos ver que el overfitting es mucho mas presente a mayor épocas.

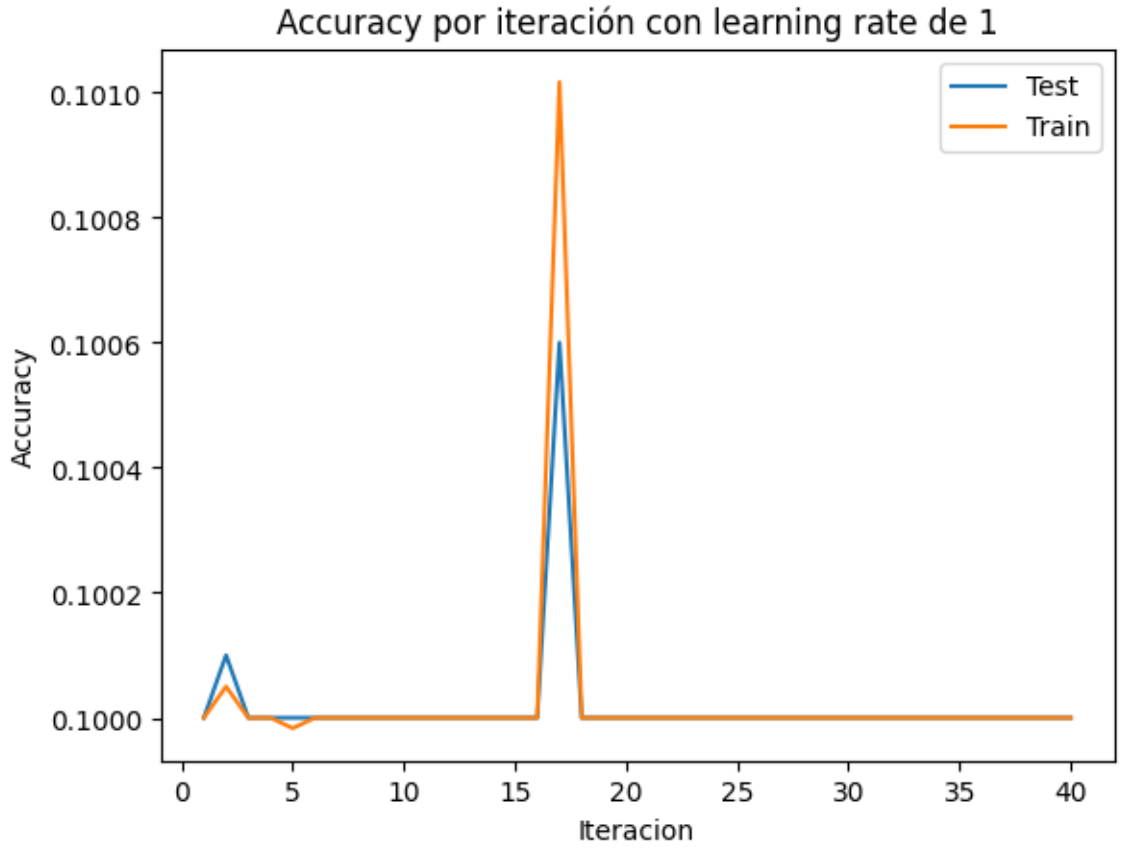
2.3. Redes Neuronales con distinto learning rate

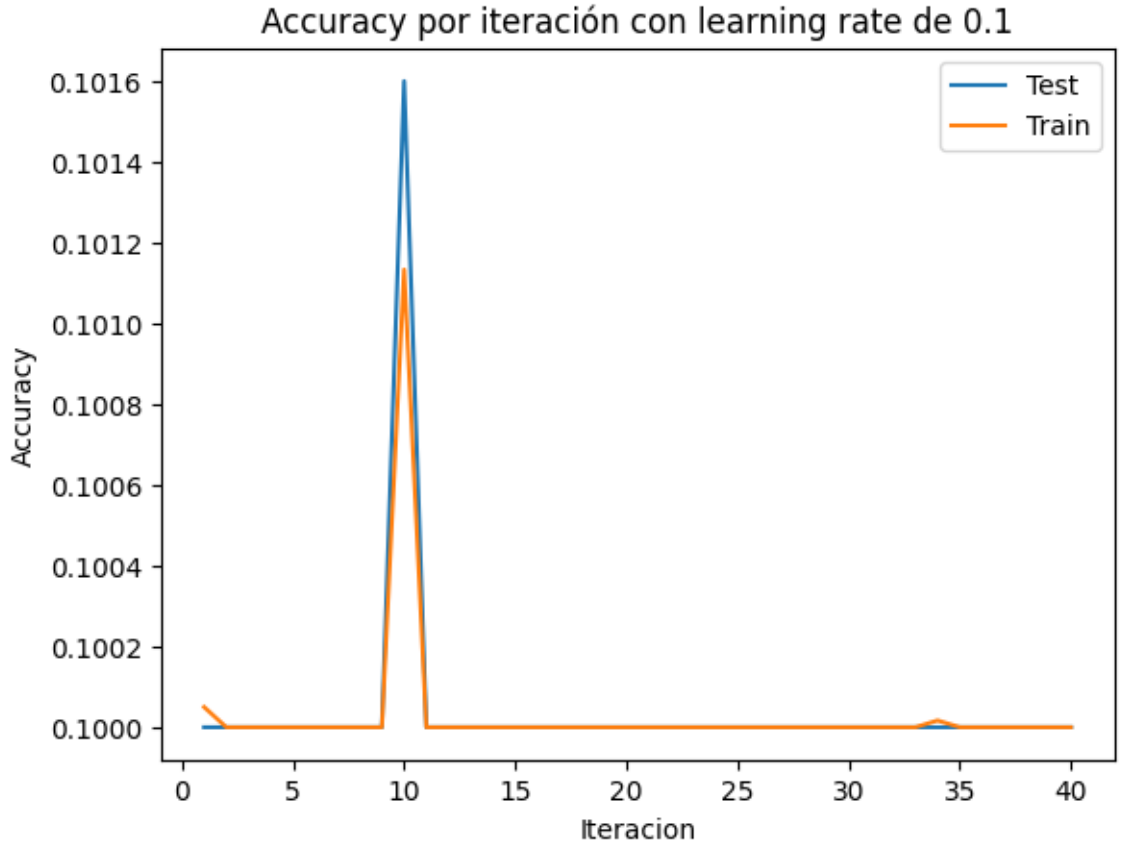
2.3.1. Learning Rate de 10, 1, 0.1, 0.01

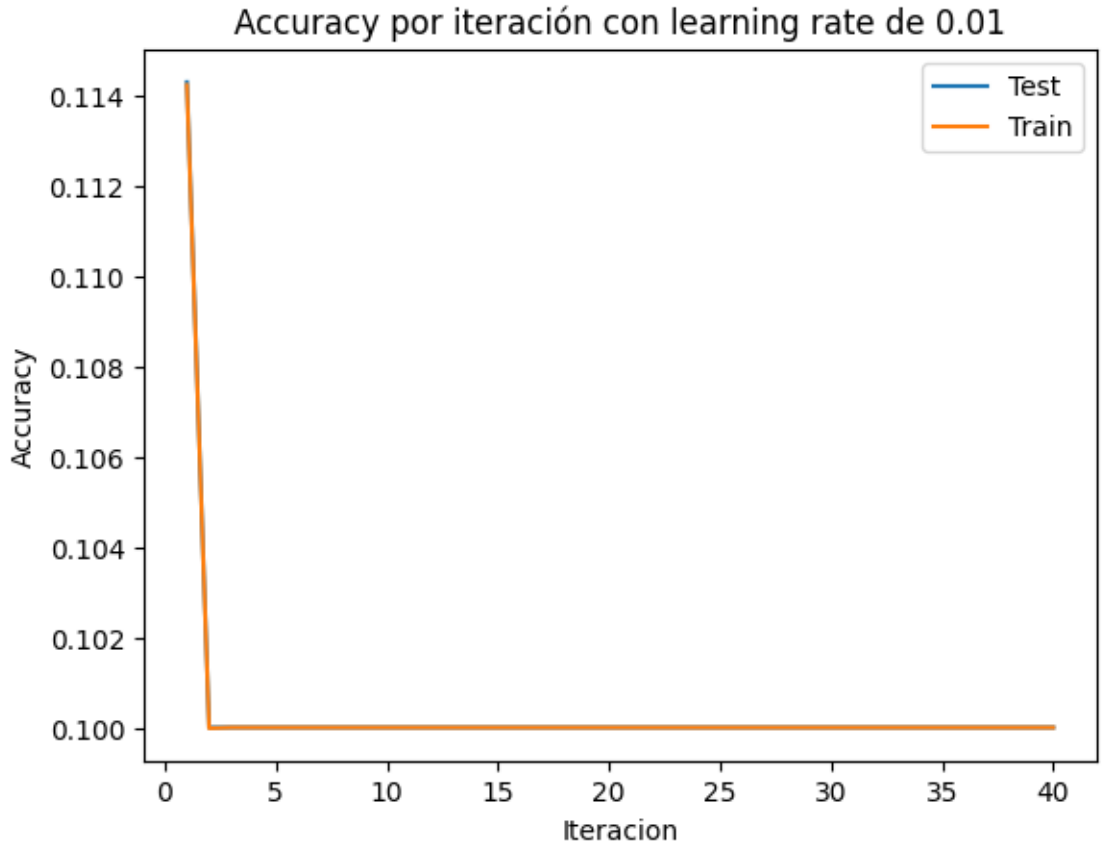
Utilizando los estos learning rates podemos ver que se obtiene casi de manera constante un 10 % de accuracy sobre todos los conjuntos, analizando las predicciones que realizaron estos modelos, notamos que lo que hicieron fue a todos los elementos del conjunto clasificarlos como de una única clase. Siendo que la cantidad de elementos de los conjuntos de prueba y entrenamiento están divididos equitativamente entre cada una de las 10 clases, tiene sentido que el 10 % de cada conjunto sean correctamente clasificados.

Este comportamiento en el entrenamiento del modelo se debe a que los learnings rates son demasiado altos, haciendo que los saltos que se realizan sean muy grandes, provocando que el aprendizaje nunca converja.



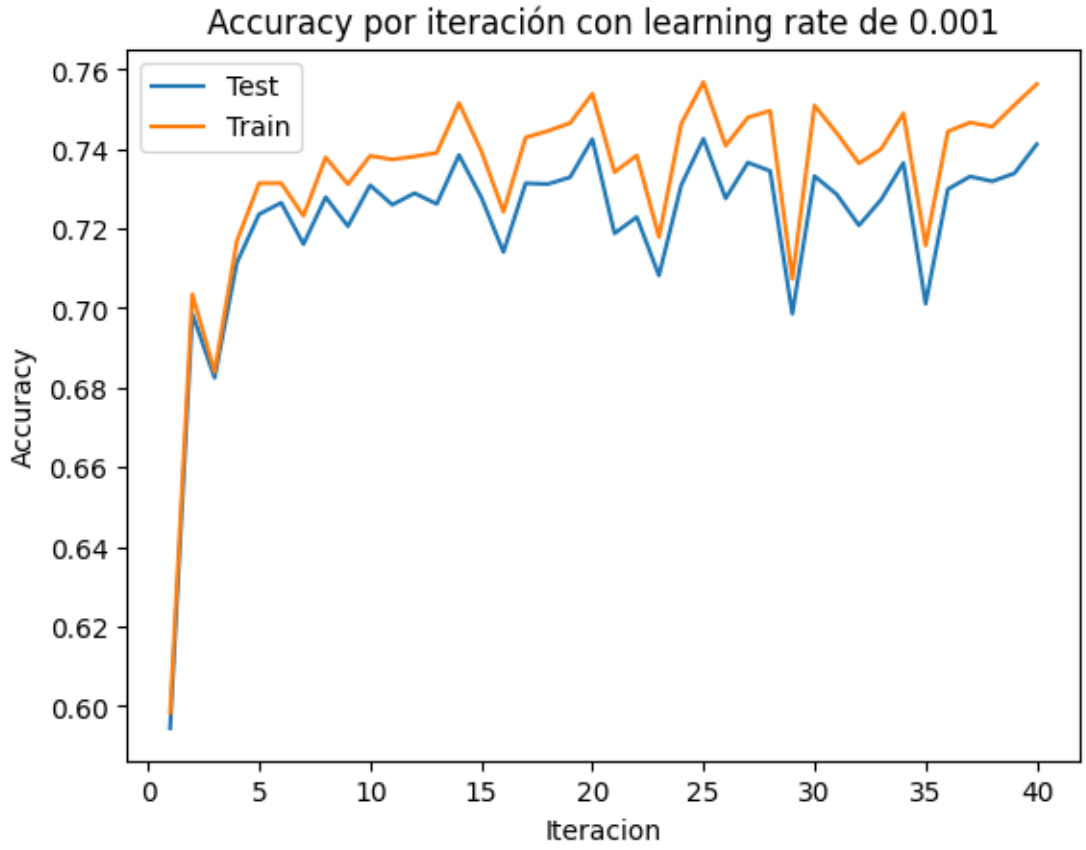

Accuracy por iteración con learning rate de 1
Test
Train
Accuracy
0.1010
0.1008
0.1006
0.1004
0.1002
0.1000
0
5
10
15
20
25
30
35
40
Iteracion


Accuracy por iteración con learning rate de 0.1
Test
Train
Accuracy
0.1016
0.1014
0.1012
0.1010
0.1008
0.1006
0.1004
0.1002
0.1000
0
5
10
15
20
25
30
35
40
Iteracion


Accuracy por iteración con learning rate de 0.01
Test
Train
Accuracy
0.114
0.112
0.110
0.108
0.106
0.104
0.102
0.100
0
5
10
15
20
25
30
35
40
Iteracion

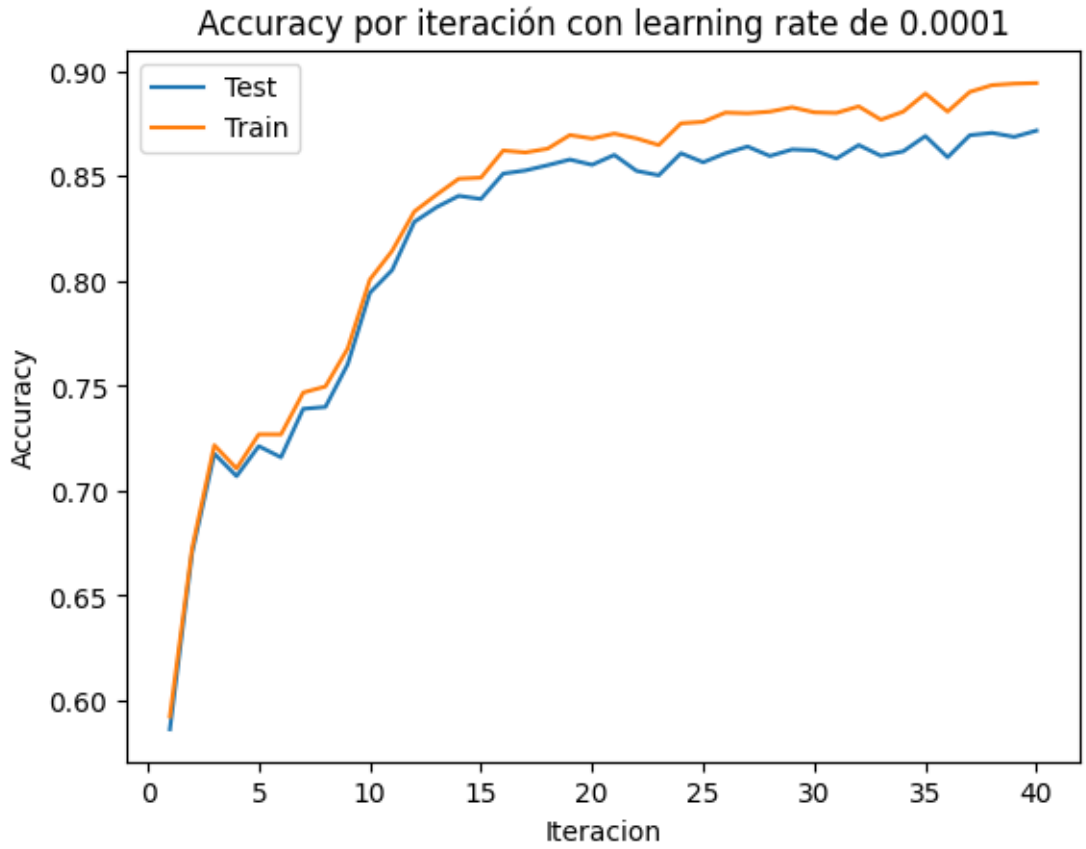
2.3.2. Learning Rate de 0.001

Sobre este learning rate podemos ver que el modelo comienza a aprender obteniendo en la época 40 un accuracy sobre el conjunto de prueba de 74,12 % y sobre el conjunto de entrenamiento de 75,63 %, siendo este modelo muy superior a los anteriores. A pesar de esto podemos apreciar en la gráfica que no es muy suave el aprendizaje de época a época, esto es un indicativo de que el learning rate sigue siendo un poco alto.



2.3.3. Learning Rate de 0.0001

Con este learning rate podemos apreciar un aprendizaje mas suave, sin tantos saltos y un modelo mas satisfactorio en la época 40, teniendo un accuracy sobre el conjunto de prueba de 87,16 % y sobre el conjunto de entrenamiento de 89,43 %. Consideramos este como el learning rate mas adecuado considerando la cantidad de epocas probadas.



2.3.4. Learning Rate de 0.00001

Con este learning rate se pueden apreciar saltos mas pequeños en el accuracy del modelo a partir de la época 10, pero un avance de accuracy mucho menor de época a época, esto se debe a que el aprendizaje es mucho mas lento. Sobre el conjunto de prueba se obtuvo un accuracy final de 74,83 % y sobre el conjunto de entrenamiento de 76,42 %. Con respecto al modelo anterior podemos ver que tiene mucho potencial a seguir mejorando a más épocas, pero un learning rate tan bajo corre el riesgo de que la función de error se quede estancada en un mínimo local y nunca se llegue al mínimo global.

Lo que no terminamos de comprender es el comportamiento errático en las primeras 10 épocas, donde se puede ver que estaba mejorando pero de repente el accuracy baja por casi 15 % para luego volver a subir. Creemos que esto se debe a que el modelo arranca sin la noción de un camino correcto en el aprendizaje, y si toma el camino equivocado cuesta mucho mas volver al indicado. En los dos modelos anteriores podemos observar un comportamiento parecido pero como el learning rate es más alto, tarda menos épocas en corregir al camino indicado hacia el correcto aprendizaje del modelo.