UNCUYO
UNIVERSIDAD
NACIONAL DE CUYO

FACULTAD
DE INGENIERÍA

TP N°3:
Machine Learning

Alumnos:

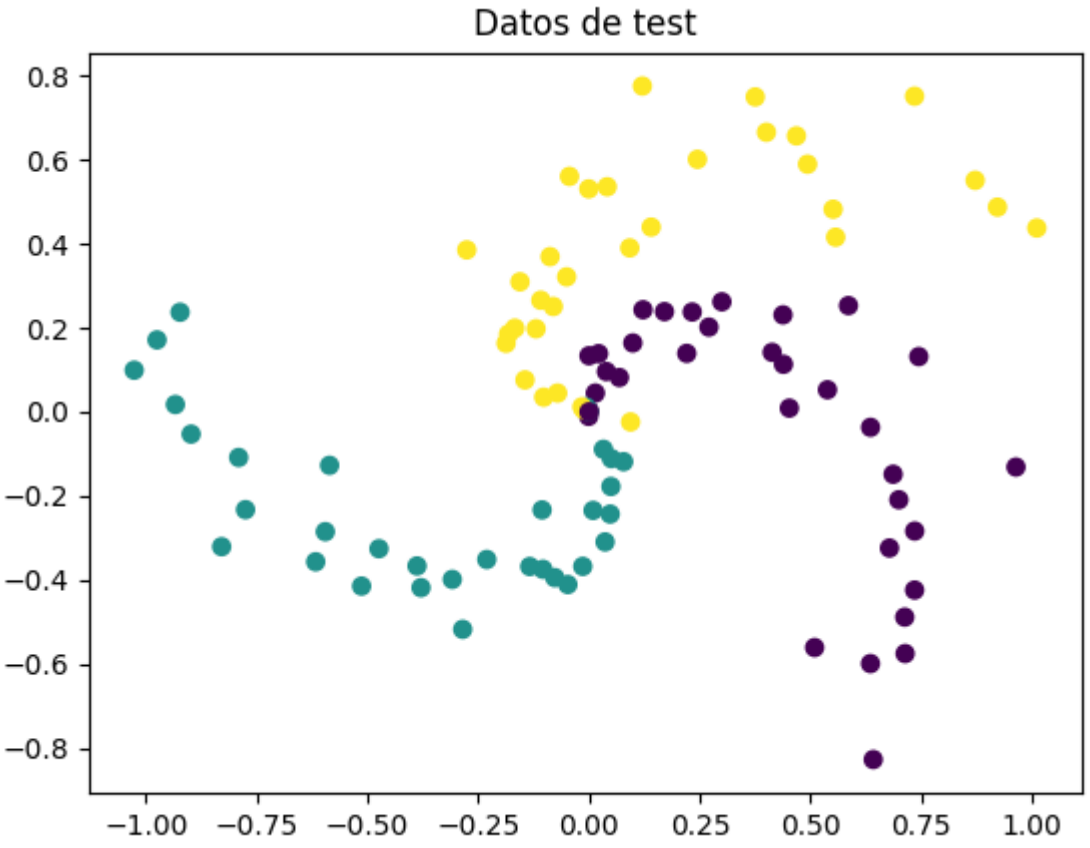
- Tomás Suárez
- Brenda Gudiño
- Esteban Vila

Para el desarrollo de este trabajo práctico partimos del código brindado por el profesor y a partir de allí realizamos modificaciones. Con el script 'generar_datos_clasificacion.py' creamos sets de datos que luego utilizamos en las distintas modificaciones.

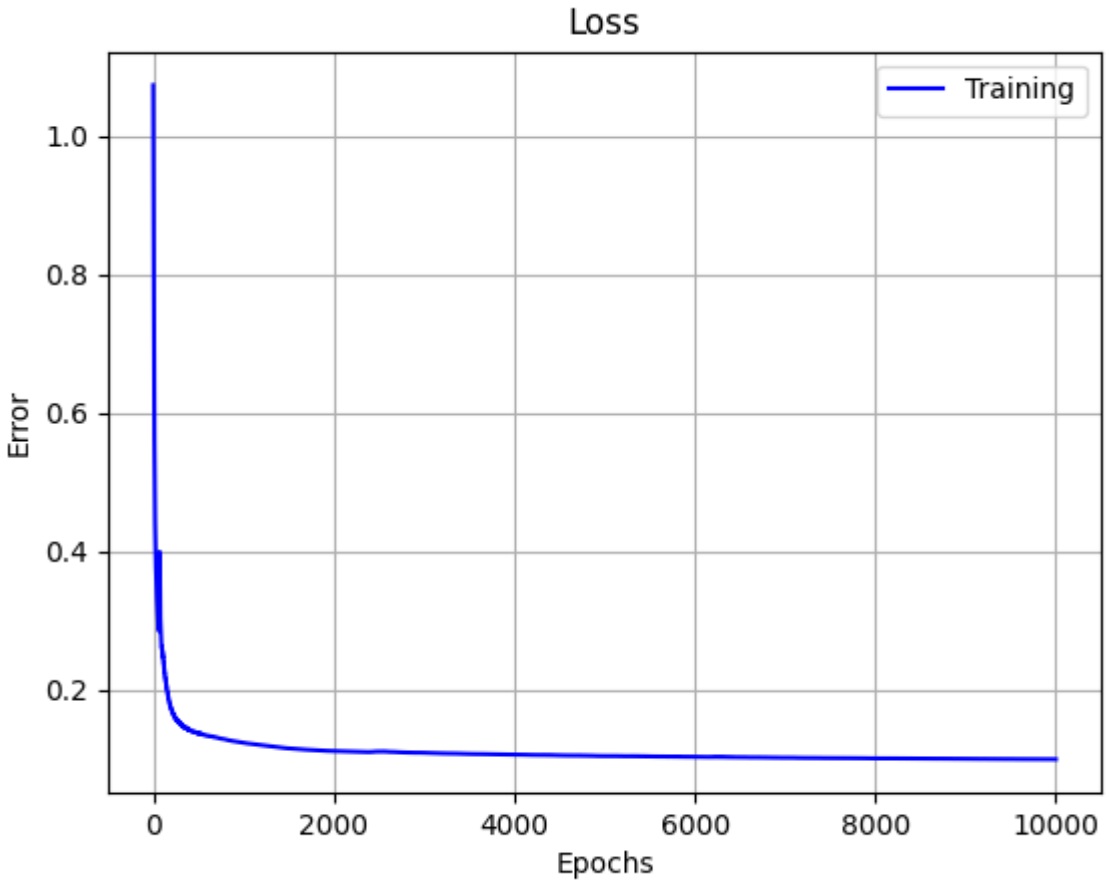
En esta primera parte se trabajó con la función de pérdida Softmax usada para resolver problemas de clasificación.

Primero se agregó la medición de la precisión además del valor de pérdida. Esto se realiza a partir de la cantidad de ejemplos clasificados correctamente sobre la cantidad total, para ello se utilizan un set de datos destinados únicamente al testeo, por lo que se deben crear más ejemplos, así se utilizan dos set de datos distintos para las etapas de train y testeo.

```
La precisión porcentual calculada es del 95.0 %.
```



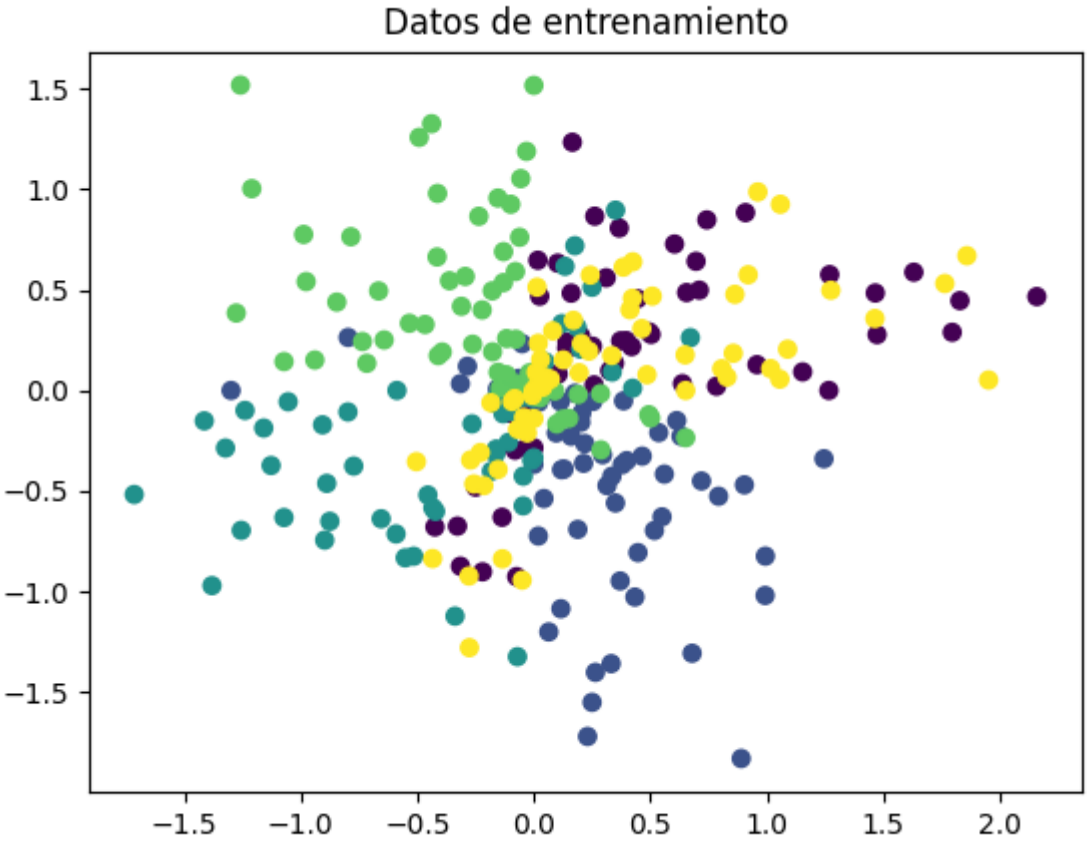
Luego se realiza la parada temprana donde se utiliza la validación, allí también se requieren mayor cantidad de ejemplos y un set de datos distintos al de train y test. La idea de esto es verificar el valor de la función de pérdida cada cierta cantidad de epochs, y se evalúa la diferencia entre la función de pérdida del entrenamiento y de validación, si esta diferencia es menor que cierta tolerancia, la cual es negativa, entonces se detiene el entrenamiento para evitar el overfitting.

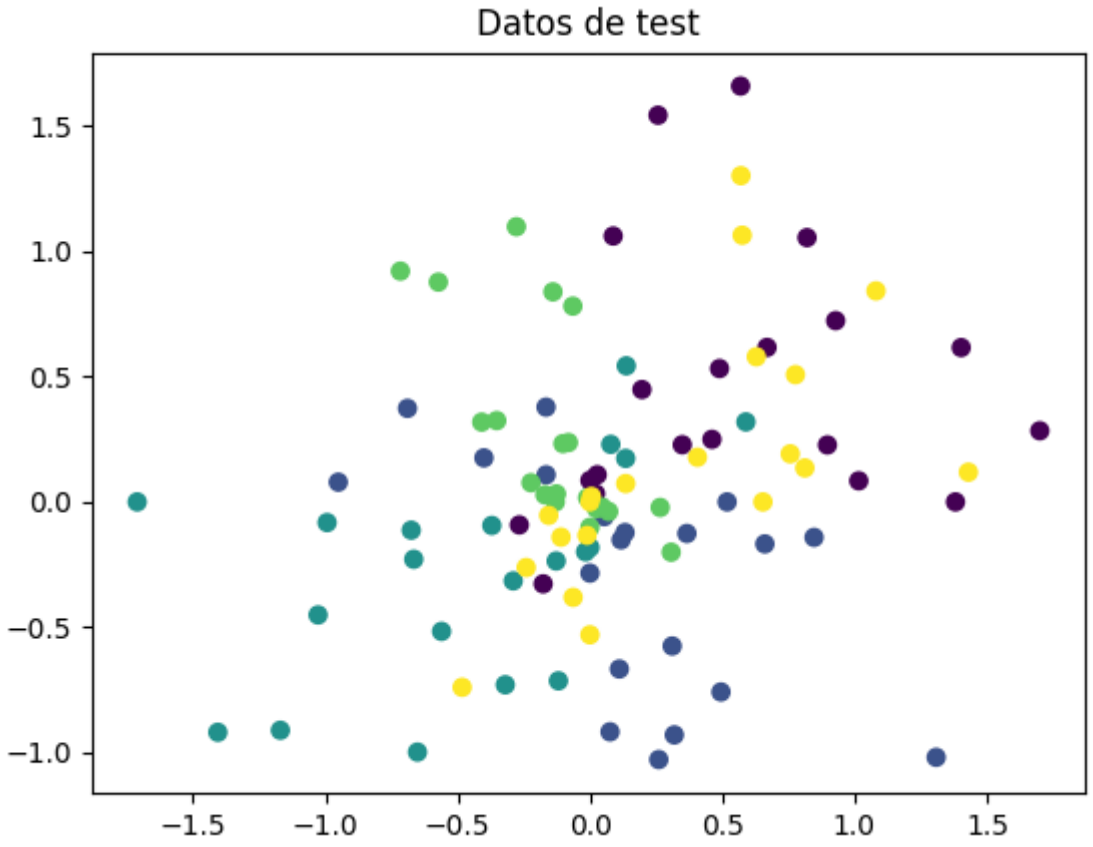


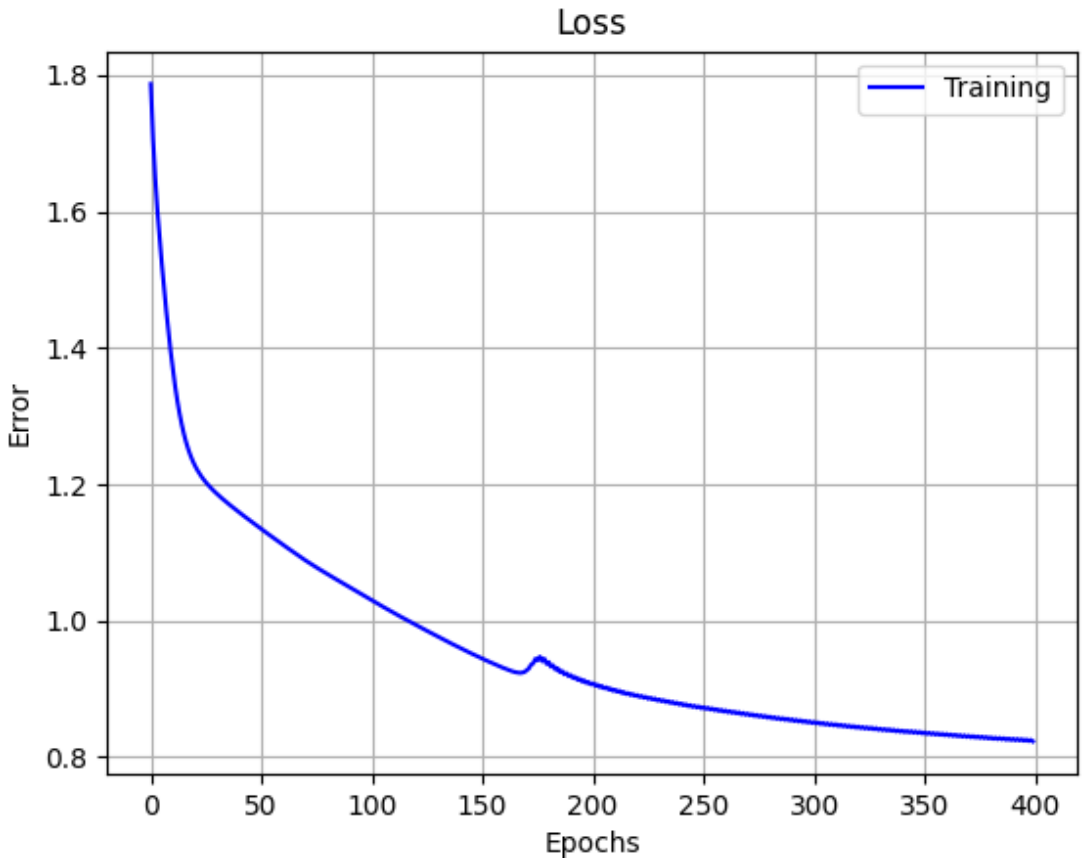
La validación utiliza la misma función de pérdida que la etapa de training pero con sets de datos específicamente tomados para la etapa de validación, acá solo se usa la probabilidad de la clase correcta que se toma del array de target.

Ahora se propone generar nuevos conjuntos de datos más complejos con clases más solapadas o con más clases, variando los parámetros del generador de datos proporcionado en el código del comienzo.

Para una cantidad de clase igual a 6, un factor de ángulo de 0.5 y una amplitud de aleatoriedad de 0.6 obtenemos los siguientes grupos de datos.


Datos de entrenamiento
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


Datos de test
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


Loss
Training
Error
1.8
1.6
1.4
1.2
1.0
0.8
0
50
100
150
200
250
300
350
400
Epochs

```
-0.14239368115669715
Se detiene el entrenamiento, comienza a aumentar el valor de loss.

La precisión porcentual calculada es del 56.0 %.
```

En la última imagen se puede observar como disminuye el valor de la precisión debido a que el diseño de esta red neuronal no es apta para datos que se solapan como se muestra, esta precisión se puede mejorar aumentando la cantidad de neuronas en la capa oculta y/o aumentando la cantidad de capas ocultas.

Problema de regresión

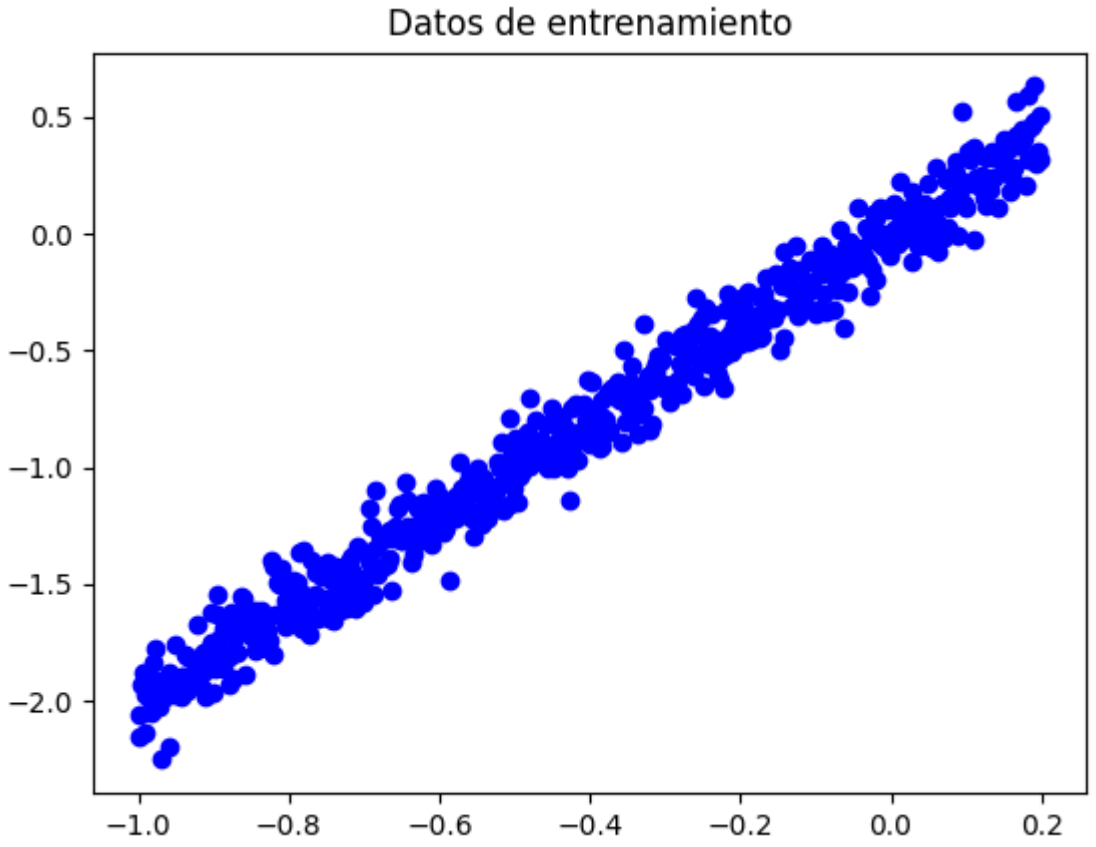
Para resolver el problema de regresión utilizamos la función de pérdida MSE e hicimos las modificaciones correspondientes en el código en cuanto a sus derivadas.
Luego generamos 3 sets de datos nuevos, uno que se corresponde con una función lineal, otro que se corresponde con una función senoidal y el último con una función 3D.
La red neuronal toma la cantidad de entradas según el set de datos con el que se elija trabajar, esas pueden ser una o dos.

En cuanto al funcionamiento general lo que se hace es un barrido de hiperparametros del tipo “Grid Layout” para el learning rate, la cantidad de neuronas de la capa oculta y el tipo de función de activación. Generamos un vector con valores de learning reates y neuronas de capa oculta cada un cierto paso y le damos la posibilidad de trabajar con dos funciones de activación, “ReLu” y “Sigmoide”, luego generamos todas las combinaciones posibles de esos parámetros y tomamos un porcentaje para que con cada combinación calcule el valor rmse, con ésto obtenemos la configuración que genera menor valor rmse y eso son los parámetros que luego introduciremos para generar la regresión.

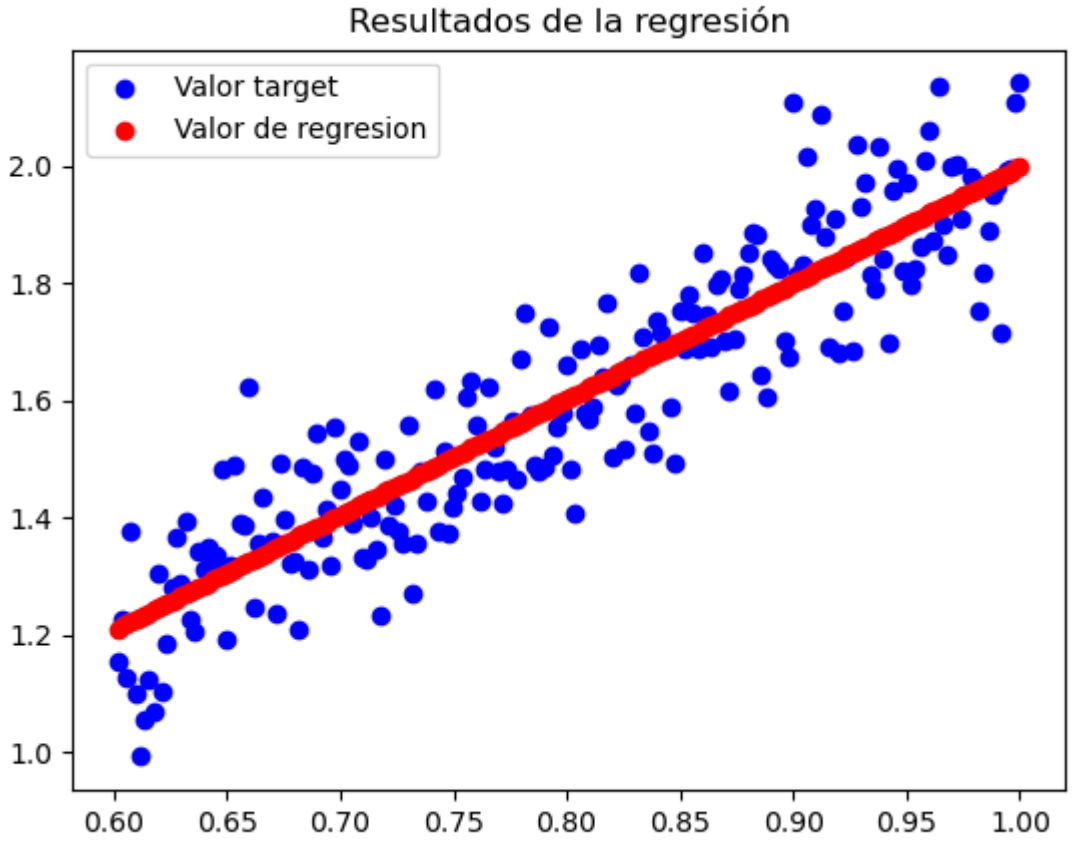
Del barrido de hiperparametros obtenemos lo que sigue:

```
------------------------------------------------------------
------------------------------------------------------------
Learning Rate: 0.75 || Capa oculta: 200 || Activacion: ReLU
Se detiene el entrenamiento, comienza a aumentar el valor de loss.

RMSE: 2.8
------------------------------------------------------------
------------------------------------------------------------
Learning Rate: 1.0 || Capa oculta: 60 || Activacion: Sigmoide
Se detiene el entrenamiento, comienza a aumentar el valor de loss.

RMSE: 198.22
------------------------------------------------------------
------------------------------------------------------------
Learning Rate: 0.5 || Capa oculta: 200 || Activacion: ReLU
RMSE: 0.58
------------------------------------------------------------
------------------------------------------------------------
Learning Rate: 0.02 || Capa oculta: 30 || Activacion: Sigmoide
Se detiene el entrenamiento, comienza a aumentar el valor de loss.

RMSE: 2.38
------------------------------------------------------------
------------------------------------------------------------
Learning Rate: 0.02 || Capa oculta: 30 || Activacion: ReLU
Se detiene el entrenamiento, comienza a aumentar el valor de loss.

RMSE: 1.92
```

De acá se busca la combinación que genere menor valor rmse ya que nos indicará la diferencia entre el target y la salida.
Entonces tomando un learning rate de 0.5, 200 neuronas en la capa oculta y una función de activación ReLu tenemos:

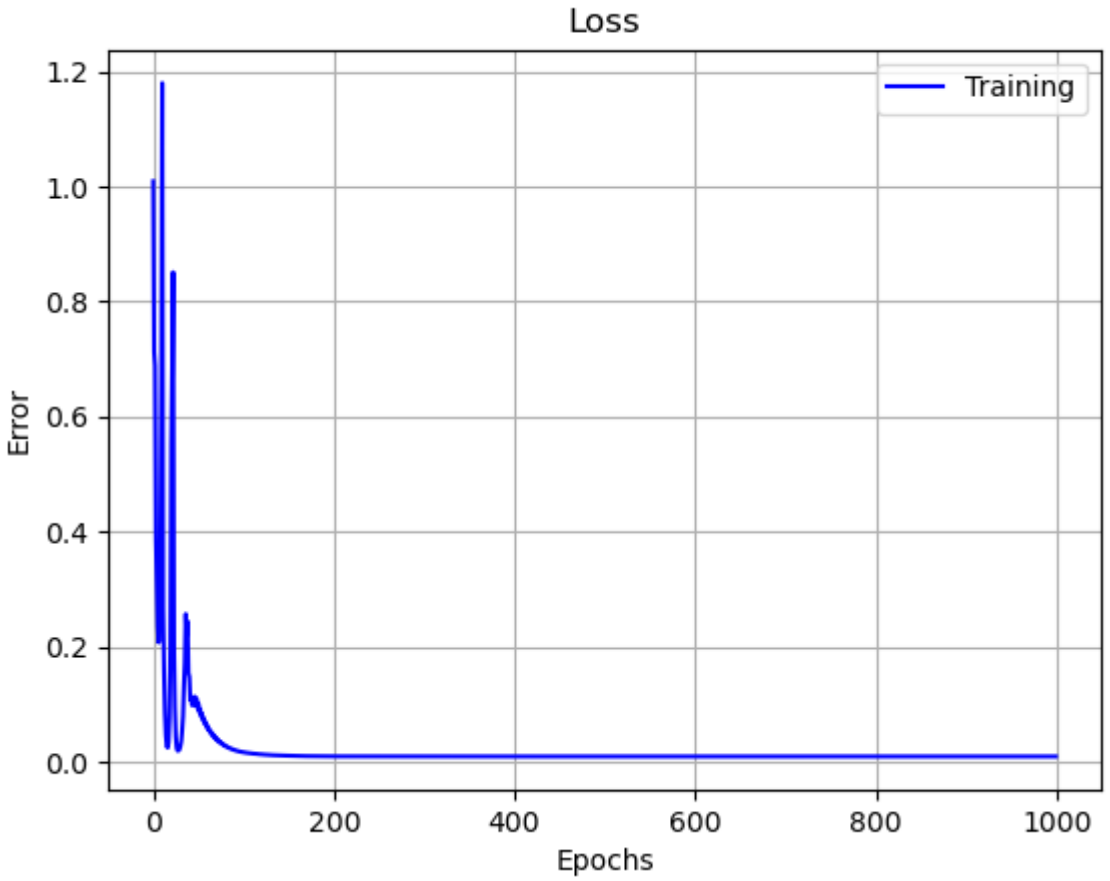


En esta primera imagen tenemos los datos de entrenamiento. Para tener estos datos tomamos un set de datos generado y de él dividimos el 60% para los datos de entrenamiento, el 20% para los de test y el 20% restante para la validación.



En esta imagen se puede observar lo que se obtiene con los parámetros anteriores aunque tiene cierta aleatoriedad por lo que cada vez que se corra el código no se mostrará la misma imagen.

Y por último en la siguiente imagen tenemos como va variando el error y allí se puede observar como el mismo tiende a 0 a medida que se llega al valor de epochs elegido que es 1000.



Para cuantificar el éxito de la regresión lo que usamos es valor rmse entre el target del test y la regresión de los datos del test, éste nos indicará que tanto error tiene nuestro modelo.