Proyecto 1 - Inteligencia Artificial

*Link del código: https://github.com/Andres20-Utec/Proyecto-1-IA

1to Luis Chahua
Ciencia de la computación
luis.chahua@utec.edu.pe

2to Andres Riveros
Ciencia de la computación
andres.riveros@utec.edu.pe

3to Massimo Imparato
Ciencia de la computación
massimo.imparato@utec.edu.pe

I. Introducción

En el presente documento se mostrará el desarrollo del proyecto 1 del curso de Inteligencia Artificial (CS2601) perteneciente a la carrera Ciencia de la Computación. El proyecto consiste en predecir el área de incendio aplicando 2 métodos de regresión : Regresión Lineal Múltiple y Regresión No-Lineal. Luego, realizar la comparación de ambos modelos, aplicando regularización y sin regularización, y variando los hiperparámetros.

II. Regresión Lineal

Este modelo es usado para predecir el valor de una variable en base a una o más variables. La variable que se busca predecir es la variable dependiente y las variables usadas para predecir el valor de la otra son variables independientes.

Esta forma de análisis busca estimar los coeficientes de la ecuación lineal, incluyendo uno o un grupo de variables independientes que mejor predigan la variable independiente. Estos coeficientes formaran la recta que más se ajusta a la nube de puntos, y que minimice los errores entre el valor real y el valor que se encuentra en la recta.

En el **modelo de regresión lineal simple** se supone que la función dependiente con una variable independiente es lineal, es decir:

$$Y = \Theta * x + b$$

En el **modelo de regresión lineal múltiple** se supone que la función dependiente con las variables independientes es lineal, es decir:

$$Y = \Theta_0 + \Theta_1 x_1 + \Theta_2 x_2 + \ldots + \Theta_p x_p$$

El desarrollo del trabajo se enfocará en el modelo de regresión lineal múltiple y regresión no-lineal.

A. Notation

- n: El número de características o variables
- m: El número de ejemplos de entrenamiento
- Θ_0: Es el bias
- $x^{(i)}$: El input de la matriz para el i^{esimo} ejemplo de entrenamiento
- $x_j^{(i)}$: El input de la matriz para el i^{esimo} ejemplo de entrenamiento
- $y^{(i)}$: El valor real para el i^{th} ejemplo de entrenamiento
- Θ: Los pesos o valores de los parámetros escogidos $(\Theta_0, \Theta_1, \Theta_2)$
- $h_\Theta(x^{(i)})$: El algoritmo de predicción para el i^{esimo} ejemplo de entramineto usando los parámetros Θ

B. Hypothesis

La hípotesis para la regresión lineal univariada es:

$$h_\Theta(x) = \Theta_0 + \Theta_1 x$$

Extendiendo la función anterior a múltiples características, la hipótesis de la regresión lineal multivariable viene dada por,

$$h_\Theta(x) = \Theta_0 + \Theta_1 x_1 + \Theta_2 x_2 + \ldots + \Theta_p x_p, x_0 = 1$$

$$h_\Theta(x) = \Theta^T x$$

C. Cost Functions

Para una variable:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

Fig. 1. Cost function

Para más de una variable:

$$= \frac{1}{2m} \sum_{i=1}^{m} \left(\left(\sum_{j=0}^{n} \theta_j x_j^{(i)} \right) - y^{(i)} \right)^2$$

Fig. 2. Cost function

MSE es una de las métricas más utilizadas, y si el modelo hace una predicción muy mala, la parte cuadrada de la función

$$MSE = \frac{1}{n}\Sigma\left(y - \widehat{y}\right)^2$$

Fig. 3. Mean Square Error

magnifica el error para poder acercar más los coeficientes a los valores óptimos.

$$\mathbf{MAE} = \frac{1}{n}\sum_{i=1}^{n}|\boldsymbol{x}_i - \boldsymbol{x}|$$

Fig. 4. Mean Absolute Error

EL MAE, sin embargo, no es muy sensible a valores atípicos en comparación con MSE ya que no penaliza tan fuertemente grandes errores de predicción.

D. Gradient Descent

El descenso de gradiente es un algoritmo de optimización iterativo de primer orden para encontrar el mínimo de una función. El cual realiza dos pasos de manera iterativa.
1. Calcular la pendiente (gradiente) que es la derivada de primer orden de la función en el punto actual.
2. Mover en dirección opuesta al incremento de la pendiente desde el punto actual en la cantidad calculada.

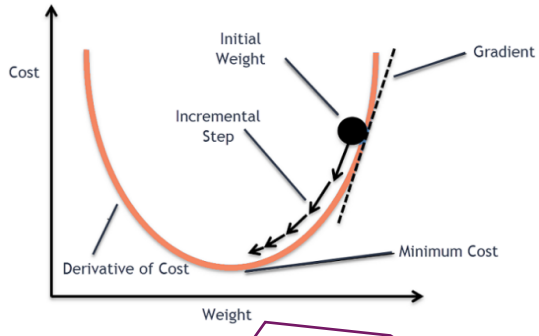

Fig. 5. Gradient Descent

La siguiente ecuación describe Gradient Descent: donde b son los coeficientes que serán actualizados, a son los coeficientes actuales, la resta hace referencia a la minimización del descenso de la gradiente. La variable alpha es un factor de aprendizaje y se multiplica con la derivada que es la dirección del descenso más empinado.

$$b = a - \alpha\frac{\partial J(\Theta)}{\partial(\Theta)}$$

Lo que nos dice está formula es el siguiente paso de hacia donde ir, cuál es la dirección de descenso más rapido. Y se repetirá hasta que converga.

La derivada parcial se representa de la siguiente manera:

$$\frac{\partial J(\Theta)}{\partial(\Theta)} = \frac{1}{m}\sum_{i=1}^{m}(h_\Theta(x^i) - y^{(i)})x_j^{(i)}$$

E. Learning Rate

Es un hiperparámetro que definirá el tamaño de los pasos que toma el gradient descent en la dirección del mínimo local, dependiendo del valor se moverá más rápido o más lento hacia los coeficientes o pesos óptimos.
Para que el gradient descent alcance el mínimo local, se debe establecer al learning rate un valor adecuado, que no sea demasiado bajo ni demasiado alto. Esto importante porque si los pasos que da son demasiado grandes, es posible que no alcance el mínimo local porque rebota de un lado a otro entre la función convexa del descenso de gradiente. Si usa un valor muy pequeño, el gradient descent alcanzará el mínimo local, pero podría tomar demasiado tiempo.

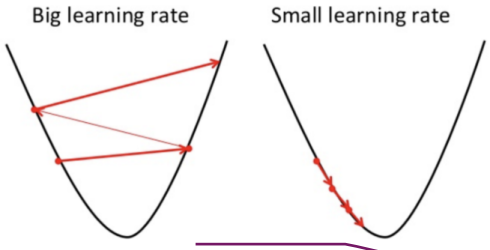

Fig. 6. High and low learning rate

Por este motivo, el learning rate nunca puede ser muy grande o muy pequeño. Una de las maneras para entender si el learning rate escogido es bueno es graficarlo.

F. Types of Gradient Descent

- **Batch Gradient Descent:**
  Batch gradient descent también llamado vanilla gradient descent, calcula el error por cada ejemplo dentro de los datos de entrenamiento, pero solo después de que se hayan evaluado todos los ejemplos de entrenamiento, el modelo se actualiza. Y todo el proceso se realiza dentro de un epoch.
  Las ventajas que trae son el error estable y convergencia estable. Las desventajas son; la gradiente de error estable a veces puede dar como resultado un estado de convergencia que no es el más óptimo que se podría lograr, y requiere que todo el conjunto de datos de entrenamiento esté cargado en memoria y disponible para el entrenamiento.

- **Stochastic Gradient Descent:**
  El SGD actualiza los parámetros para cada iteración de

entrenamiento uno por uno. Dependiendo del problema, esto puede hacer que SGD sea más rapido que el BGD. Al actualizar constantemente nos permite tener una tasa de mejora bastante detallada, pero puede generar gradientes ruidosos, lo que puede hacer que la tasa de error salte en lugar de disminuir lentamente.

- **Mini Batch Gradient Descent:**
  Es una combinación de los dos conceptos vistos anteriormente en el cual dividimos los datos de entrenamiento en pequeños lotes o "batches" con respecto al tamaño original, y realizamos una actualización de los parámetros por cada uno de estos mini lotes. Los tamaños usualmente escogidos son de potencia de 2, como 64 o 256, no existe una regla claramente definida porque varía según las diferentes aplicaciones o modelos.

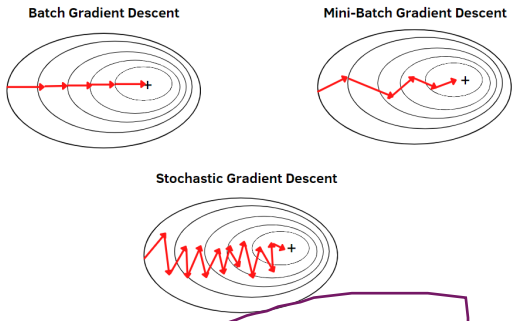

Fig. 7. Types of Gradient Descent

III. Regresión No Lineal

El segundo metodo de análisis que utilizamos para el proyecto es Regresión No Lineal. Este modelo de regresión consiste en modelar la data a observar en una función que sea no lineal en naturaleza (polinomial). La principal diferencia entre estos dos tipos de análisis es cómo podemos relacionar la data contra los resultados: mientras que una regresión lineal intenta hacerlo mediante una recta y ajustarla lo más posible a la distribución de resultados, la regresión no lineal, ya que es polinomial, genera una relación a través de una curva. Esto es importante ya que, dependiendo de las variables que tomemos como data a observar y los resultados que esperamos predecir y su comportamiento, es posible que con un análisis de regresión lineal no podamos ajustarnos bien a la distribución de los resultados reales mediante la recta y no nos sea suficiente para predecir de una forma precisa, caso donde debemos optar por un modelo más complejo como la regresión no lineal (Como se muestra en la Figura 8).

Como se observará a continuación mediante las ecuaciones, la diferencia entre estos 2 métodos de regresión se encuentra sólamente en cómo se modela la relación entre las variables de observación y los datos reales. En otras palabras, poco cambia fuera del modelado de nuestra hipótesis, ya que la estructura de entrenamiento y ajuste de coeficientes es la misma.

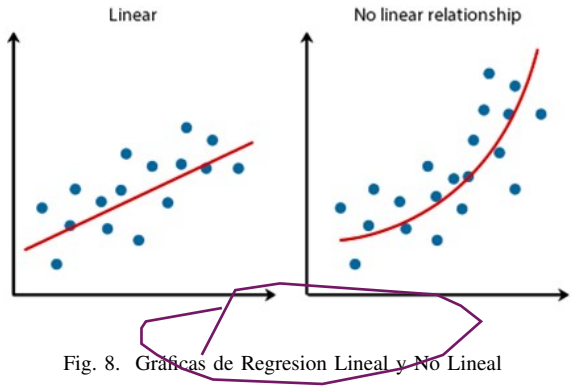

Fig. 8. Gráficas de Regresion Lineal y No Lineal

A. Hypothesis

La hipótesis para la regresión no lineal es la siguiente, donde se puede notar que las características de X se elevan a la potencia "i" para obtener la curva:

$$h_(x_i) = b + x_i w_1 + x_i^2 w_2 + x_i^3 w_3 + ... + x_i^p w_p$$

Lo cual se puede resumir en la siguiente ecuación:

$$h_(x_i) = b + \sum_{j=1}^{p} x_i^j w_j$$

Además, para trabajar esta función más limpiamente se puede denominar a ese bias como un w0, y agregarle a nuestro dataset "X" una característica al inicio conteniendo un 1 (x_i^0) con el fin de poder realizar los cálculos mediante operaciones matriciales e incrementar a la vez la eficiencia.

$$h_(x) = X * W.T$$

B. Loss Function

Las funciones de Loss para este análisis son las mismas que en la regresión lineal:

MSE:

$$L = \frac{\sum_{i=0}^{n}(y_i - h(x_i))^2}{2n}$$

MAE:

$$L = \frac{\sum_{i=0}^{n}|(y_i - h(x_i))|}{n}$$

C. Gradient

Finalmente, esta es la ecuación derivada que dictará la dirección de ajuste de los coeficientes w_j:

$$\frac{\partial L}{\partial W_j} = \frac{\sum_{i=0}^{n}(y_i - h(x_i))(-x_i^j)}{n}$$

IV. Objetivos

A. General

- Evaluar la eficacia de una regresión lineal y una no lineal para la predicción del área después de un entrenamiento que integra datos observacionales provenientes de la base de datos forestfire.

B. *Específicos*

- Construir una regresión lineal y una no lineal capaz de predecir el área
- Realizar la predicción en base a diferentes tipos de técnicas de optimización del descenso gradiente, así como también en diferentes tipos de funcion de pérdida.
- Realizar variaciones en los hiperparámetros tales como en el learning rate, coeficiente de regularización y tamaño de batches.
- Evaluar el grado de precisión de los modelos utilizando datos de testeo que han sido separados adecuadamente del entrenamiento del modelo.
- Realizar comparaciones con otros modelos que tienen distintos hiperparámetros y también modelos que han sido implementados en librerías oficiales de python como sklearn.

V. Metodología

A. *Toma de datos*

En este proyecto, se trabajó con un total de 12 atributos o "características" obtenidas de la base de datos forestfire, de los cuales 10 atributos son numéricos y 2 categóricos. Sin embargo, se ha trabajado solo las variables wind,temperature, relative humidity y rain. Esto se debe a que los otros atributos se pueden calcular en base a estos cuatro mencionados.
FFMC (Fine Fuel Moisture Code), DMC (Duff Moisture Code) y DC (Drought Code) tienen que ver con códigos de combustible, mientras que ISI (valor que representa la velocidad de propagación del fuego) y BUI (cantidad de combustible disponible) son índices de comportamiento del fuego. Finalmente FWI (Fire Weather Index) combina los últimos componentes mencionados de comportamiento del fuego.
La manera en que todos estos códigos e índices se relacionan se puede observar en la siguente gráfica:

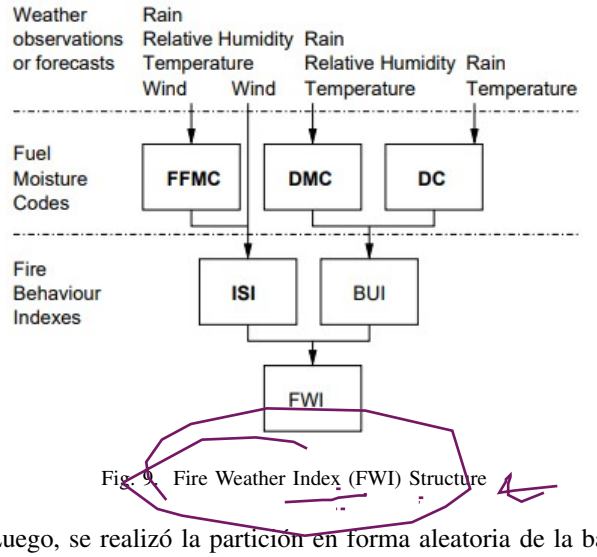

Fig. 9. Fire Weather Index (FWI) Structure

Luego, se realizó la partición en forma aleatoria de la base de datos en 3 grupos, el primero tendrá el 70% de los datos para realizar el entrenamiento, el segundo 15%, utilizados para validación y el último 15% para hacer el testeo.

Después de verificar la correspondencia de los datos con los atributos y la partición, debido a la variación de los valores que podrían tomar los atributos, se procedió a normalizar de forma separada cada uno de los 3 grupos formados(entrenamiento, validación, testeo) con el método de escalado min-max.

Además, se añadió una columna adicional con todos los valores iguales a 1. Este valor se agrega para poder multiplicarse con el bias que previamente denominamos como "w_0" sin afectarlo.

B. *Implementación del modelo*

Para la construcción del modelo, se implementaron las funciones de pérdida, el cálculo de la gradiente y la función de entrenamiento utilizando las librerías de numpy y pandas, precisamente para el cálculo matricial de las operaciones y el cálculo del coeficiente de determinación de la predicción para poder tener una métrica de comparación con el modelo implementado usando la librería sklearn y de cierta manera verificar nuestros resultados. Además, estas funciones recibirán como parámetros a los hiperparámetros descritos anteriormente.

VI. Evaluación experimental

En esta experimentación se procedió a correr el código usando diferentes valores para los hiperparámetros, tales como el número de épocas del entrenamiento, el learning rate, el coeficiente de regularización lambda (solo para la regresión no lineal). Todas las iteraciones con las funciones de pérdida Mean Square Error y Mean Absolute Error. Primero se evaluará el caso cuando la regresión es lineal. Todos los resultados que se obtengan para este caso se compararán con el resultado que se obtiene mediante la librería sklearn el cual es 0.0032. A continuación se mostrarán las tablas con los valores del coeficiente de determinación de la predicción tanto con la función de pérdida MSE y MAE:

	0.50000	0.10000	0.01000	0.00100	0.00010	0.00001
50	-0.933	-0.117	-15.324	-30.811	-25.261	-1.031
100	-1.214	-0.124	-2.560	-0.721	-9.204	-5.955
300	-20.379	-0.814	-1.636	-11.910	-1.045	-19.135
500	-24.368	-0.795	-0.954	-4.993	-6.352	-13.388
1000	-31.181	-0.885	-0.240	-2.998	-8.877	-22.747
2000	-30.466	-0.728	-0.388	-3.710	-1.981	-4.229
4000	-36.989	-1.020	-0.608	-1.202	-3.115	-10.394

Fig. 10. Cálculo del coeficiente de determinación de la predicción en MAE

.

	0.50000	0.10000	0.01000	0.00100	0.00010	0.00001
50	-0.288	-2.915	-17.707	-25.085	-26.778	-8.639
100	-1.940	-0.337	-12.188	-3.051	-21.199	-16.470
300	-23.335	-2.829	-0.835	-0.106	-10.881	-11.832
500	-37.349	-0.853	-0.023	-3.015	-2.351	-7.727
1000	-38.306	-15.941	-2.547	-4.379	-5.988	-19.913
2000	-31.602	-30.305	-0.093	-12.759	-5.649	-9.283
4000	-32.561	-36.043	-1.278	-2.759	-10.469	-9.330

Fig. 11. Cálculo del coeficiente de determinación de la predicción en MSE

. A continuación, se mostrarán las gráficas correspondientes a la variación del loss training y validation training a lo largo de la cantidad de épocas utilizando ambas funciones de pérdida. Importante resaltar que en estos entrenamientos se han agregado minibatches con un tamaño de 64.

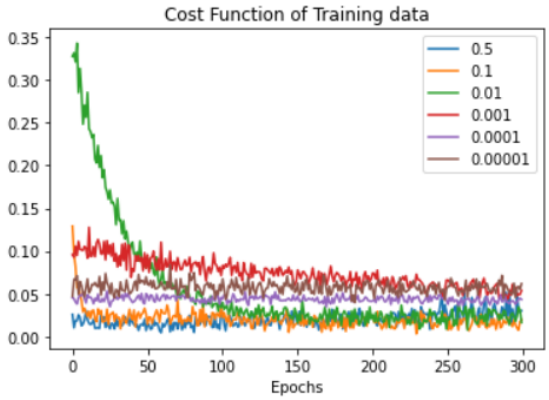

Fig. 12. Variación del loss training a lo largo de la cantidad de épocas utiliando MSE

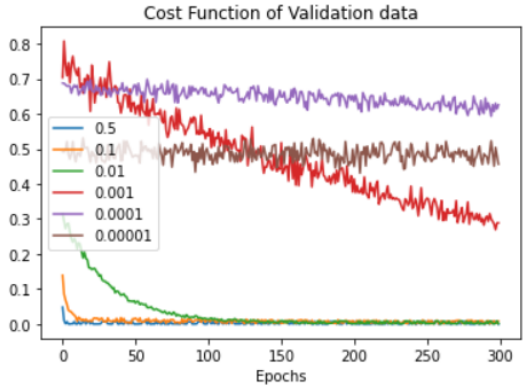

Fig. 13. Variación del loss validation a lo largo de la cantidad de épocas utilizando MSE

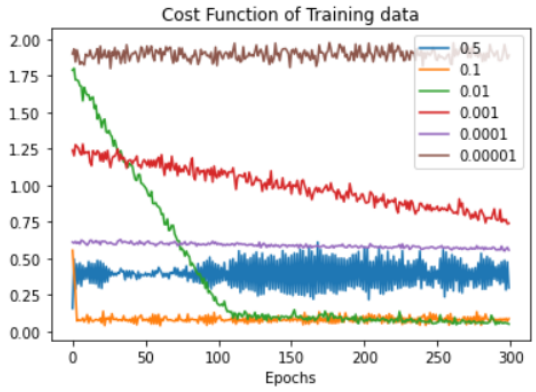

Fig. 14. Variación del loss training a lo largo de la cantidad de épocas utilizando MAE

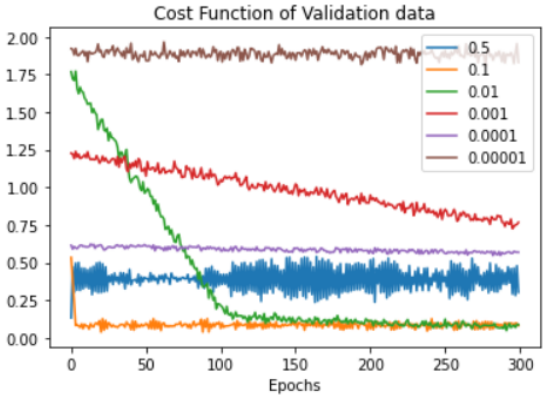

Fig. 15. Variación del loss validation a lo largo de la cantidad de épocas utilizando MAE

.

Ahora, respecto al modelo de regresión no lineal, obtuvimos la siguiente tabla de coeficiente de determinación, esto se comparará con el resultado obtenido en sklearn, el cual es ....

	0.01000	0.00100	0.00010	0.00001
100	-804.218	-2066718.487	-3141572.649	-3767059.650
200	-870.039	-717989.661	-4813464.000	-4266544.834
400	-1068.389	-130546.416	-3926373.680	-5210170.398
800	-995.228	-377.733	-2338992.005	-2767030.034
1500	-962.401	-360.277	-525407.509	-3242202.237

Fig. 16. Cálculo del coeficiente de determinación de la predicción en MSE, con parámetro de regularización lambda=1(para el caso de ridge)

.

	0.01000	0.00100	0.00010	0.00001
100	-804.218	-2066718.487	-3141572.649	-3767059.650
200	-870.039	-717989.661	-4813464.000	-4266544.834
400	-1068.389	-130546.416	-3926373.680	-5210170.398
800	-995.228	-377.733	-2338992.005	-2767030.034
1500	-962.401	-360.277	-525407.509	-3242202.237

Fig. 17. Cálculo del coeficiente de determinación de la predicción en MAE, con parámetro de regularización lambda=1(para el caso de ridge)

.

Las siguientes gráficas corresponden a la variación del loss training y validation training a lo largo de las épocas, utilizando ambas funciones de pérdida.

Para el caso de la regularización lasso, se tiene:

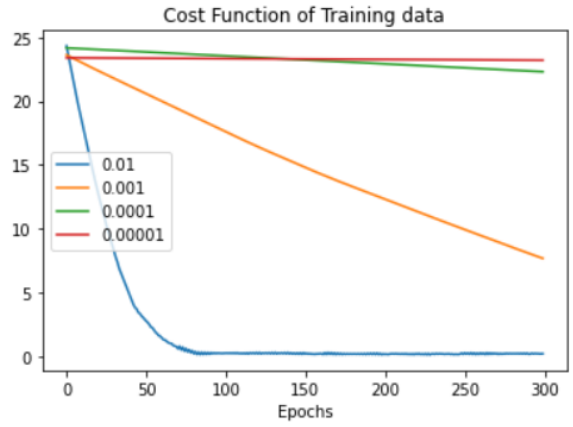

Fig. 18. Variación del loss training a lo largo de la cantidad de épocas utilizando MAE, con coeficiente de regularización igual a 1

.

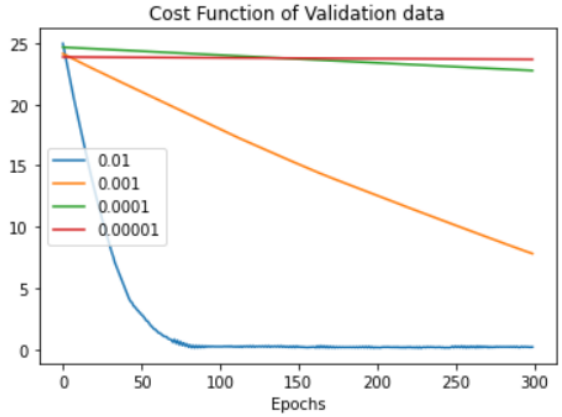

Fig. 19. Variación del loss validation a lo largo de la cantidad de épocas utilizando MAE, con coeficiente de regularización igual a 1

. Para el caso de la regularización ridge se tiene:

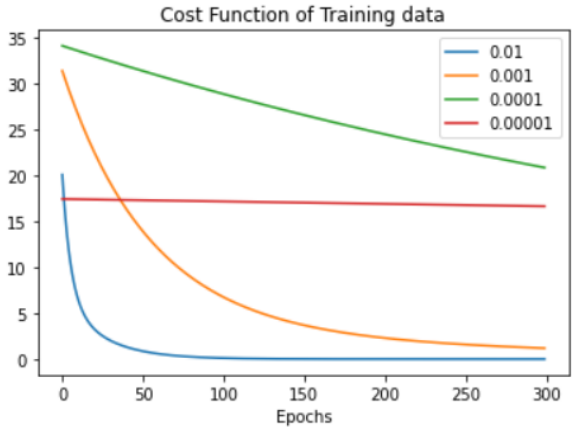

Fig. 20. Variación del loss training a lo largo de la cantidad de épocas utilizando MSE, con coeficiente de regularización igual a 1

.

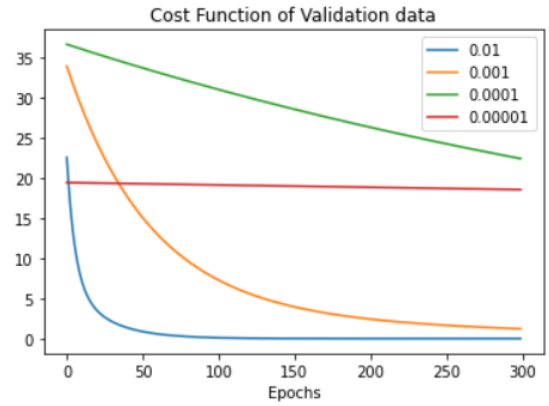

Fig. 21. Variación del loss validation a lo largo de la cantidad de épocas utilizando MSE, con coeficiente de regularización igual a 1

.

VII. Discusión de resultados

A. Primer modelo

Respecto a la regresión lineal, después de la experimentación, se puede observar que en el entrenamiento del modelo, el valor de las funciones de pérdida disminuyen a valores muy cercanos a 0, y muestran un comportamiento muy similar al esperado tanto en el training loss como en el validation loss. Sin embargo, al momento de calcular el coeficiente de determinación de la predicción utilizando los valores del test de datos, se obtiene un resultado que muestra no ser preciso. Sin embargo, el resultado del coeficiente de determinación utilizando la librería SKLearn también es bastante lejano a 1, lo cual quiere decir que es muy probable que se necesite una gráfica más compleja para tener una mejor predicción de los datos.

B. Segundo modelo

Con respecto a la regresión no lineal, se puede observar de forma similar a la primera regresión, que el entrenamiento funciona de manera eficaz, esto se puede observar en las gráficas. Sin embargo, al momento del testeo, se observa una notoria diferencia en el cálculo del coeficiente de determinación a comparación con el de la librería SKLearn. Respecto a los coeficientes de regularización, se notó una mejora al asignarle al coeficiente el valor de 1. Sin embargo, si se colocase el coeficiente con un valor mucho mas grande, el loss training y validation subiran de manera muy drástica, lo cual no sería eficiente en este caso.

VIII. Conclusiones y propuestas de mejora

Gracias a la experimentación y a la comparación con el modelo y resultados obtenidos de SKLearn, se puede concluir que para predecir este resultado, se necesita una mayor cantidad de datos y un modelo con una complejidad mayor, puesto que, en los casos de regresión lineal y no lineal, se obtienen resultados no precisos. En este proyecto se han aplicado las técnicas sumamente importantes de regularización y normalización al momento de realizar el entrenamiento. También, se realizaron bastantes experimentos con diversos hiperparámetros para poder realizar comparaciones y con ello poder lograr obtener los resultados más precisos posibles.

References

[1] Shams S. Research assistant at Purdue University, "Multivariate Linear Regression" *Machine Learning Medium*, 23 de agosto de 2022. Available: https://machinelearningmedium.com/2017/08/23/multivariate-linear-regression/

[2] Sushant Patrikar, "Batch, Mini Batch & Stochastic Gradient Descent", *Towards Data Science*, 30 de Septiembre de 2019 . Available: https://towardsdatascience.com/batch-mini-batch-stochastic-gradient-descent-7a62ecba642a

[3] Paulo Cortez, Anibal Morais, "A Data Mining Approach to Predict Forest Fires using Meteorological Data", *Department of Information Systems/ R&D Algoritmi Centre, University of Minho, 4800-058 Guimaraes, Portugal*, 2007 . Available: http://www3.dsi.uminho.pt/pcortez/fires.pdf

[4] Will Kenton, Somer Anderson, "Defining Nonlinear Regression", *Investopedia*, 29 de Mayo de 2022 . Available: https://www.investopedia.com/nonlinear-regression