Proyecto 1

1st Barrios, Alonso
Universidad de Tecnología e Ingeniería
Ciencia de la Computación
Lima, Perú
alonso.barrios@utec.edu.pe

2nd Rodriguez, Mauricio
Universidad de Tecnología e Ingeniería
Ciencia de la Computación
Lima, Perú
mauricio.rodriguez@utec.edu.pe

3rd Tenazoa, Renzo
Universidad de Tecnología e Ingeniería
Ciencia de la Computación
Lima, Perú
renzo.tenazoa@utec.edu.pe

***Abstract*—Este proyecto está desarrollado para poner a prueba lo aprendido acerca del modelo de regresión multivariada. El objetivo del proyecto es predecir un atributo haciendo uso del modelo mencionado y analizar las distintas gráficas obtenidas por los errores de entrenamiento y validación.**

***Index Terms*—regresión, predicción, error, underfiting, overfiting**

I. Introducción

Con el objetivo de poner a prueba el modelo de regresión multivariada se utilizó el *dataset* *Forest Fire*, en cual consta de 517 intancias con 12 atributos cada una que representan distintas características del suelo. Se buscó predecir el campo denominado como *area*, el cual representa el área del campo que podría ser afectado de producirse un incendio. Posteriormente, en el proceso de entrenamiento del modelo se obtendrán distintas gráficas las cuales representan al error de validación y de entrenamiento del modelo. Se usaron estas gráficas para visualizar si se obtuvo *underfiting* u *overfiting* en el modelo de regresión.

II. Explicación

Para la realización del modelo de regresión multivariada usaremos las siguientes funciones:

- *passdata()*: Se encarga de recolectar los datos del *dataset* para almacenarlos en listas.
- *normalize()*: Se encarga de normalizar los datos ya que la diferencia entre estos llega a ser muy grande. Cabe resaltar que se normalizó tanto el x e y, además en x no se normalizaron las columnas *month* y *day*.
- *hypothesis()*: Es el resultado que se obtiene de la predicción cada vez volviéndose más preciso.
- *derivate()*: Se encarga de derivar las funciones necesarias.
- *error()*: Se encarga de calcular el error de cada fase de entrenamiento.
- *update()*: Se encarga de actualizar los valores para que cada fase de entrenamiento sea más precisa.
- *train()*: Es la función principal en la que se empieza a entrenar el modelo de regresión.

Para mejorar el modelo de regresión se dividió el *dataset* de manera aleatoria en 3 partes:

- *train*: Es el 70% de los datos con los que se entrenó al modelo.
- *validation*: Es el 20% de los datos con los que se valida el modelo.
- *test*: Es el 10% de los datos con los que se pone a prueba el modelo.

III. Resultados

La primera gráfica muestra cómo en el *epoch* 0, es decir, en la primera iteración, el error empieza siendo bastante alto cuando el modelo es aplicado en los 3 segmentos de datos. Asimismo, se aprecia que en cada *epoch* o iteración el error disminuye en los 3 segmentos de datos. Una observación de esta primera imagen de resultados es que los 3 errores son distintos en cada *epoch*, pero muy cercanos entre ellos.

```
epoch: 0
train: 14.74555080397353
validation: 14.699389307903008
test: 15.14257768173282

epoch: 1
train: 14.515091924839325
validation: 14.46856131902421
test: 14.903401784582734

epoch: 2
train: 14.288372369914418
validation: 14.24148098939732
test: 14.668128142869579

epoch: 3
train: 14.065331393763659
validation: 14.018087424604524
test: 14.436693192032594

epoch: 4
train: 13.845909237831572
validation: 13.79832071963286
test: 14.20903440157452
```

Fig. 1. Impresión de primeros errores

En la segunda gráfica se observan las últimas 5 iteraciones que se realizan para corregir el error del modelo. Se aprecia cómo los errores se mantienen en un mínimo óptimo y disminuye en proporciones muy pequeñas. Asimismo, la diferencia entre los errores se mantuvo mínima por lo que se podría considerar que no hubo muestras de *overfitting* ni *underfitting*. Por el contrario, el resultado parece indicar que el modelo

funciona correctamente y se mantiene con resultados estables en los 3 segmentos de datos.

```
epoch: 0
train: 14.74555080397353
validation: 14.699389307903008
test: 15.14257768173282

epoch: 1
train: 14.515091924839325
validation: 14.46856131902421
test: 14.903401784582734

epoch: 2
train: 14.288372369914418
validation: 14.24148098939732
test: 14.668128142869579

epoch: 3
train: 14.065331393763659
validation: 14.018087424604524
test: 14.436693192032594

epoch: 4
train: 13.845909237831572
validation: 13.79832071963286
test: 14.20903440157452
```

Fig. 2. Impresión de últimos errores

En la gráfica se puede visualizar y confirmar el cómo los errores del modelo empiezan siendo altos y van disminuyendo, de forma que se genera una curva similar a una esperada. También se visualiza que la diferencia entre los errores del modelo aplicado a los 3 segmentos es mínima en todo momento, siendo casi imperceptible la diferencia entre las curvas. Esto señala la ausencia de *overfitting* y *underfitting*, indicándonos que el modelo se desarrolló de manera óptima.

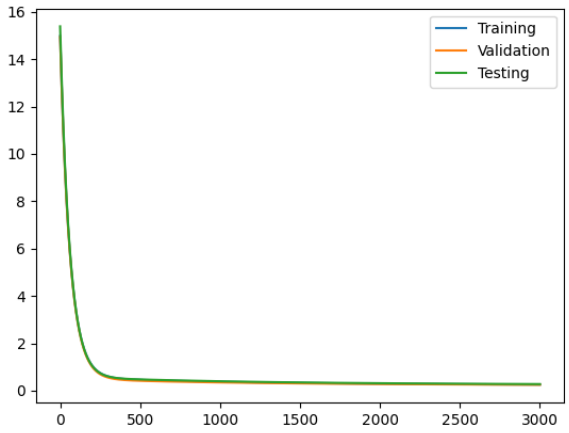

Fig. 3. Gráfico de errores

Para llegar a estos resultados óptimos, se hicieron una serie de pruebas:

1) **Normalización:** La primera prueba que se realizó fue la de no normalizar ningún dato, esto nos dio como resultado errores muy altos, incluso llegando al infinito.

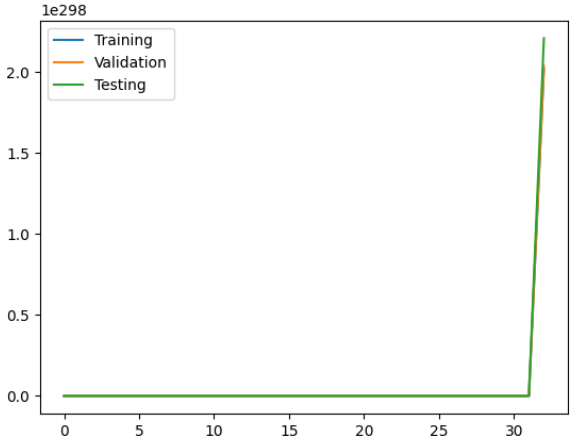

Fig. 4. Gráfico de errores con datos no normalizados

Dada la gráfica pudimos concluir que los datos no normalizados nos retornan un *overfitting*.

2) ***Alpha*:** La siguiente prueba fue comprobar el correcto funcionamiento del algoritmo con diferentes *alphas*.

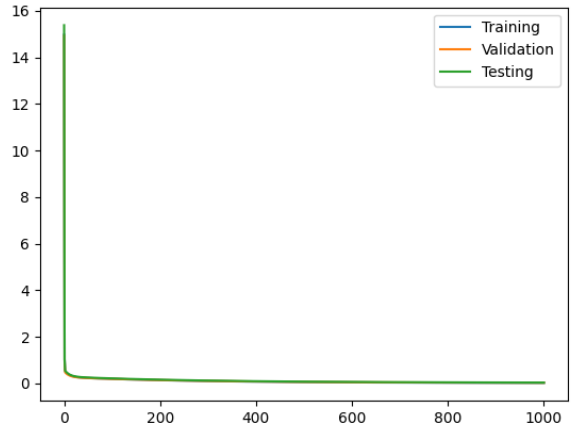

Fig. 5. Gráfico de errores con *alpha* = 0.01

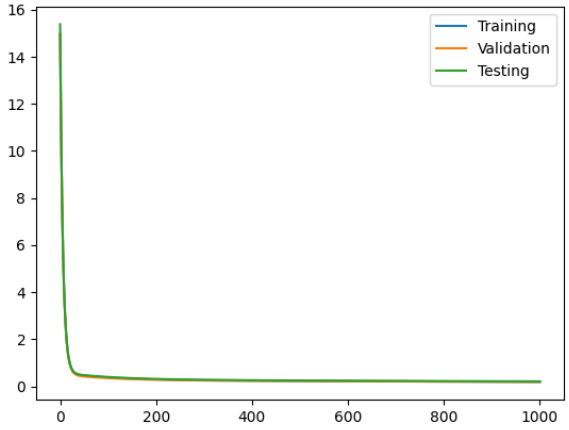

Fig. 6. Gráfico de errores con *alpha* = 0.001

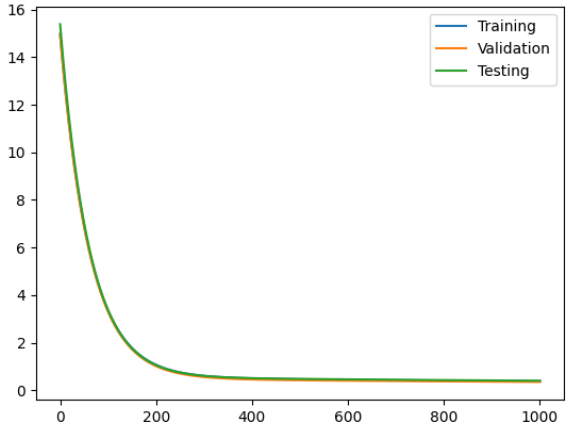

Fig. 7. Gráfico de errores con *alpha* = 0.0001

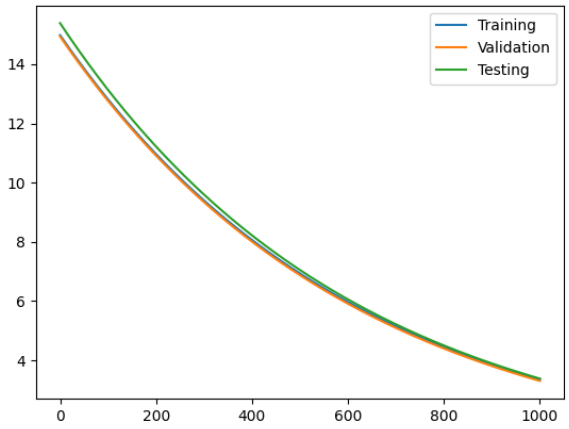

Fig. 8. Gráfico de errores con *alpha* = 0.00001

Dada las gráficas podemos concluir que con todas se llega a un mínimo óptimo, la diferencia es el tiempo que tomará en llegar a ser óptimo. Esto se debe a que al disminuir el *alpha* este debe de tardarse más, ya que tendrá que dar más saltos para llegar al mínimo óptimo.

Conclusiones

- Se aplicaron los conocimientos de regresión multivariada aprendidos en clase de forma satisfactoria.
- En los resultados se aprecia cómo el error disminuye a un mínimo aceptable.
- Tanto de forma visual como numérica se observa que la diferencia entre los errores de *training*, *validation* y *texting* es mínima, por lo que no habría *overfitting* ni *underfitting*. Esto significaría que el modelo es óptimo.
- Al no normalizar los datos caeremos en un *overfitting*, ya que el error se dispara hasta el infinito.
- Al disminuir el *alpha* el algoritmo necesitará mayor cantidad de *epoch* para poder hallar un error mínimo óptimo.

References

[1] Montero, R. (2016). *Modelo de regresión lineal múltiple*. Universidad de Granada. España.
[2] Bishop, C. (2006). *Pattern Recognition and Machine Learning*. Cambridge CB3 0FB, U.K.