Inteligencia artificial avanzada para la ciencia de datos I

Módulo 2 Análisis y Reporte sobre el desempeño del modelo. (Portafolio Análisis)

Docente

Ivan Mauricio Amaya Contreras

Alumno

Elmer Osiel Avila Vargas
A00826359

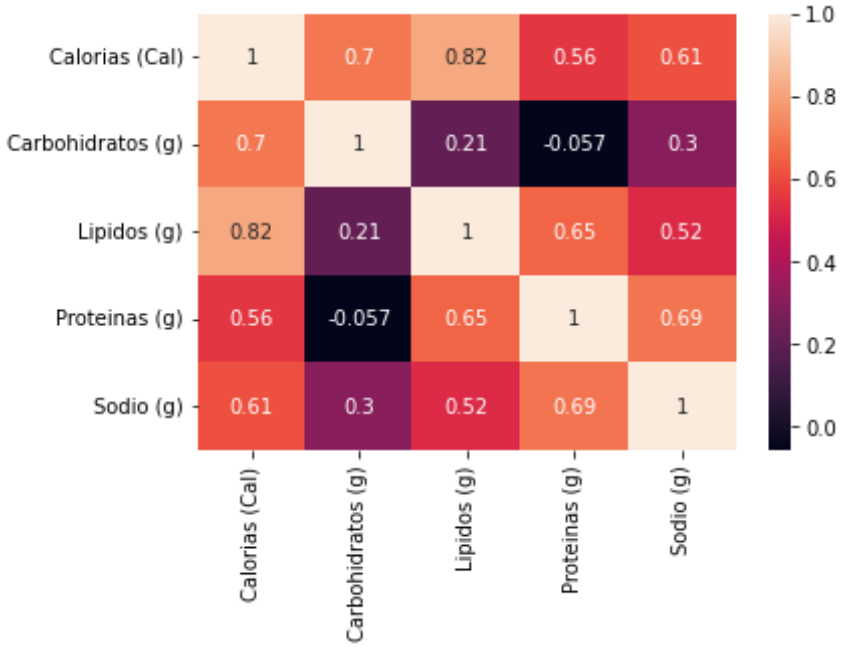
Monterrey, Nuevo León a 11 de septiembre del 2022

Modelo: Regresión Lineal / Optimización: Gradiente Descendente

El modelo desarrollado busca predecir la cantidad de calorías contenida en un alimento en base a los datos nutricionales que aportan cada uno.

Separación del conjunto de Datos (Entrenamiento, validación, prueba)

Inicialmente se realizó una selección de datos para crear un primer modelo, para ello se creó una matriz de correlación de nuestros datos donde podemos observar cuales son las variables que más aportan información a la variable dependiente. De este modo se definió que en primera instancia se utilizarían los datos de Carbohidratos y de Sodio (con mayor correlación respecto a la variable dependiente).



Una vez que se decidieron las variables independientes se procedió a realizar la separación de conjuntos de datos para entrenamiento, validación y prueba. Los porcentajes del contenido total de datos para cada uno de los conjuntos se distribuyó de la siguiente manera: **entrenamiento (60%), validación (20%) y prueba (20%)**. En el siguiente código se aprecia el uso de la librería sklearn para realizar dicha división.

```
# Realizamos una seleccion de datos para entrenamiento, validacion y prueba: entrenamiento (60%), validacion (20%) y prueba (20%)
columns = ['Calorias (Cal)', 'Carbohidratos (g)', 'Lipidos (g)', 'Proteinas (g)', 'Sodio (g)']

# Separacion variables independientes y dependiente
x = data[columns[1:]]
y = data[columns[0]]

# Separación de los datos de entrenamiento.
x_train, x_aux, y_train, y_aux = train_test_split(x, y, train_size = 0.6, random_state = 1)

# Separación de los datos de validacion y prueba.
x_val, x_test, y_val, y_test = train_test_split(x_aux, y_aux, train_size = 0.5, random_state = 1)
```

A continuación, se muestra el conteo de datos para cada uno de los conjuntos donde podemos observar que se realizó la separación de la manera esperada.

```
print('Total de datos:', len(data))
print('No. de datos de entreamiento:', len(x_train))
print('No. de datos para validacion:', len(x_val))
print('No. de datos para prediccion:', len(x_test))
```

```
Total de datos: 276
No. de datos de entreamiento: 165
No. de datos para validacion: 55
No. de datos para prediccion: 56
```

Modelo Obtenido

Se obtuvo el modelo de manera manual, es decir, sin el uso de librerías. A continuación se muestra un pedazo de código representativo.

```
  # Calculo de los nuevos thetas (coeficientes del modelo)
  theta[0] = theta[0] - alpha / n_train * sum_j0
  theta[1] = theta[1] - alpha / n_train * sum_j1
  theta[2] = theta[2] - alpha / n_train * sum_j2

# Impresion de los thetas obtenidos finalmente
print('Thetas calculados:', theta)
```

```
Thetas calculados: [1.002820081419432, 7.242105172328549, 8.174577172328553]
```

De este modo se obtuvo el siguiente modelo:

$$Y = 1.002 + 7.242 * x1 + 8.1745 * x2$$

Análisis

Tras calcular el error con la función de costo para el modelo de regresión lineal con la técnica de gradiente descendente se obtuvieron los siguientes errores:

- Error de entrenamiento: 4444.8059
- Error de validación: 4433.1342
- Error de prueba: 4724.1788

Ahora, recordemos que el análisis tiene que ser realizado sobre los datos obtenidos del conjunto de entrenamiento y el de validación, a partir de ello se realizan los siguientes análisis:

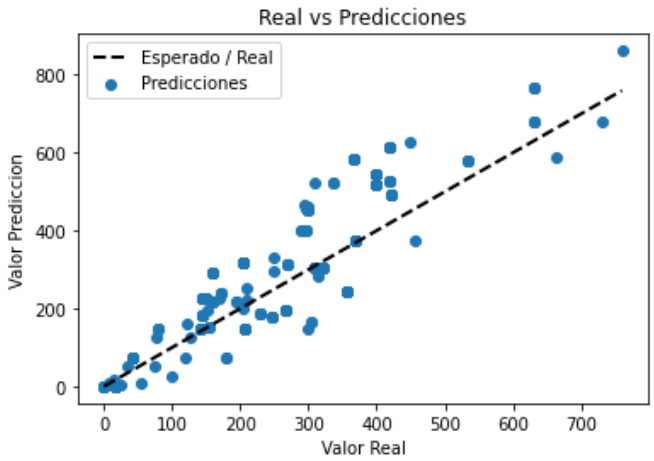
Bias / Sesgo

Diagnóstico: Bias Alto.

Como se puede observar los errores tanto para el conjunto de entrenamiento como el de validación y prueba resultan ser bastante altos, esto nos sugiere sobre la existencia de un **bias alto.** A continuación, se muestra una tabla comparativa de una muestra de 10 registros donde se visualiza la existencia de sesgos altos.

	Real	Prediccion	Diferencia / Sesgo
0	400	520.0	-120.0
1	400	520.0	-120.0
2	308	306.0	2.0
3	141	146.0	-5.0
4	141	146.0	-5.0
5	141	146.0	-5.0
6	0	1.0	-1.0
7	630	682.0	-52.0
8	630	682.0	-52.0
9	159	218.0	-59.0

En la siguiente grafica se observa el sesgo de una manera más visual; si nuestros valores predichos fueran iguales a los esperados obtendríamos una línea perfecta, como la línea punteada que se observa a lo largo de la gráfica, sin embargo, observamos que los puntos de predicción se encuentran lejanos a dicha línea, se hace notorio el sesgo existente.

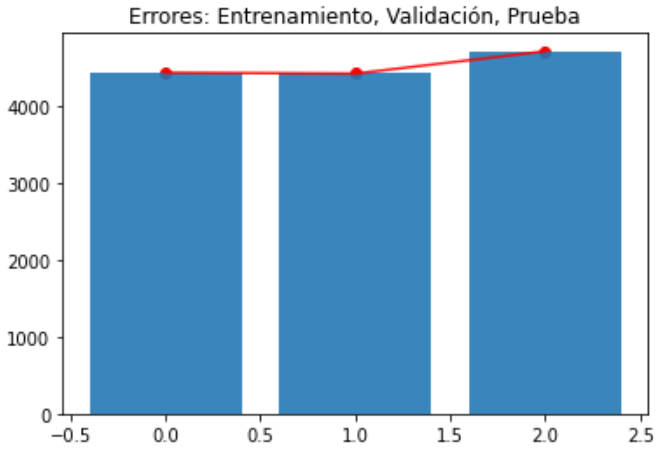


Variación

Diagnóstico: Variación Baja.

Respecto a la variación, al observar los errores podemos ver que es baja comparada con la numeraria del sesgo, pues los errores se mantienen dentro de un muy buen rango, la diferencia es considerablemente baja si comparamos los 3 errores (entrenamiento, prueba, validación) lo cual sugiere que el modelo tiene un comportamiento similar para todos los conjuntos de datos.

La variación baja se hace notar a partir de la siguiente gráfica donde podemos notar que la mayoría de nuestras predicciones se mantienen de manera constante dentro de un rango de error.



Las 3 barras corresponden a los errores obtenidos para entrenamiento, validación y prueba respectivamente, con la línea roja podemos apreciar que al aumento es nada notorio entre los errores de entrenamiento y validación, aunque si bien aumenta un poco para el de prueba la variación es realmente pequeña comparada con el gran sesgo que se obtuvo.

Nivel de ajuste del modelo

Diagnóstico: Underfitt.

Finalmente podemos decir que esta relación de alto bias y una variación baja, nos indica la existencia de underfitt en el modelo. Se puede mencionar que este comportamiento es esperable, pues se está empleando un modelo de regresión lineal, los cuales suelen tener este tipo evaluaciones debido a que suelen ser bastante simples; cuando se emplea una sola variable independientes estos modelos pueden ser visualizados a través de una simple gráfica de una pendiente que difícilmente se adaptaran a los comportamientos representados por funciones polinomiales.

Optimización del Modelo (Técnica de regularización y ajuste de parámetros)

A raíz de ello, y como se vio a lo largo del curso, para manejar este problema del bajo ajuste del modelo a los datos es necesario aumentar la complejidad del modelo, para ello se añadieron características que brindaran mayor información para la generación de un modelo más ajustado. Por otro lado, se realizó un ajuste de parámetros más preciso a través de prueba y error y los datos recolectados anteriormente.

En el siguiente cuadro de código se aprecia el crecimiento del modelo a uno más complejo, se pasó d utilizar las 2 variables con mayor correlación a utilizar toda información disponible, es decir, el uso de las 4 variables independientes.

```
# Funcion de Hipotesis
h = lambda x0, x1, x2, x3, theta: theta[0] + theta[1] *x0 + theta[2] *x1 + theta[3] *x2 + theta[4] *x3

# Funcion auxiliar (parcial) de la funcion de costo
j_i = lambda x0, x1, x2, x3, y, theta: (h(x0, x1, x2, x3, theta)-y)**2
```

En la siguiente imagen se muestra el ajuste de los parámetros, para lograr llegar a estos valores fue necesario probar distintos valores de Alpha iniciales, una vez que se encontraron unos que daban buenos resultados se asignaron como valores iniciales y se otorgó un learning rate bajo para que el modelo se ajustara de manera precisa sin realizar pasos agigantados pues ya nos encontramos cerca de una numeraria aceptable.

```
# Parametros e hiper-parametros
theta = [0,4,8,3,-2]
alpha = 0.0000000065

n_iter = 10000
```

Modelo Optimizado

Nuevamente se muestra un pedazo de código representativo, así como la impresión de los valores de alpha calculdados:

```
    # Calculo de los nuevos thetas (coeficientes del modelo)
    theta[0] = theta[0] - alpha / n_train * sum_j0
    theta[1] = theta[1] - alpha / n_train * sum_j1
    theta[2] = theta[2] - alpha / n_train * sum_j2
    theta[3] = theta[3] - alpha / n_train * sum_j3
    theta[4] = theta[4] - alpha / n_train * sum_j4

# Impresion de los thetas obtenidos finalmente
print('Thetas calculados:', theta)
```

```
Thetas calculados: [0.0008566099408843958, 4.70270171297117, 8.254737312971152, 3.2281203250923927, -2.0047326860894197]
```

De este modo se obtuvo el siguiente modelo:

$$Y = 0.0008 + 4.7027 + 8.2547 + 3.2281 - 2.0047$$

Análisis Comparativo

Comencemos observando los errores para cada uno de los conjuntos de datos:

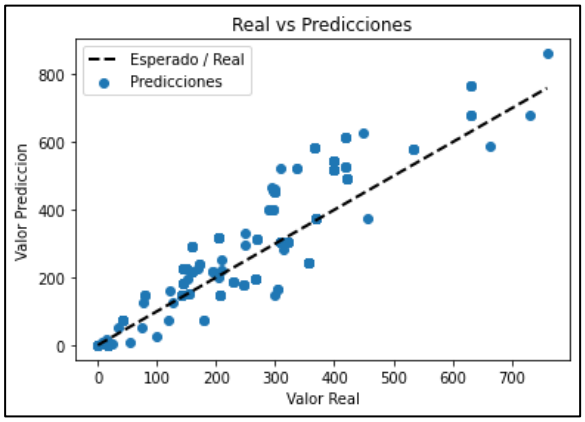
- Error de entrenamiento: 242.7300
- Error de validación: 242.6779
- Error de prueba: 214.9874

Podemos observar que el sesgo de nuestro modelo bajo notablemente gracias que se realizó un modelo más complejo, en la siguiente gráfica podemos observar evidentemente que los datos de predicción se encuentran mucho más apegados a la recta deseada

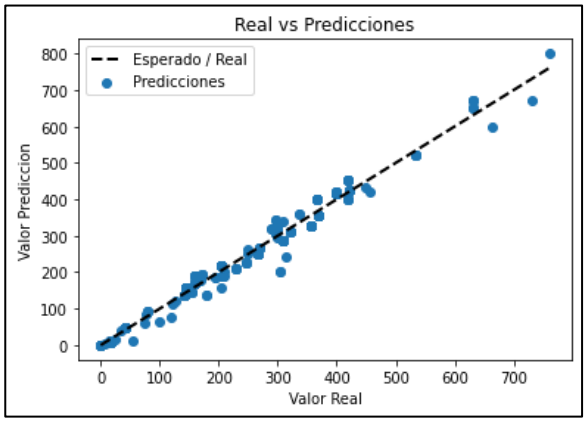
Bias / Sesgo

El bias para el nuevo modelo resulta ser mucho más bajo, alrededor de 20 veces más bajo si comparamos los errores del primero modelo con los del nuevo. La gran diferencia se puede notar sobre todo en la comparativa de las gráficas siguientes:

Primer Modelo



Modelo Optimizado



Como podemos observar en el gráfico del modelo optimizado las predicciones se acercan mucho más a la recta esperada, en cambio en la gráfica dada por el primer modelo se observa un mayor alejamiento por parte de los puntos, esto corresponde al sesgo existente, mismo que se aprecia en las siguientes tablas.

Primer Modelo

	Real	Prediccion	Diferencia / Sesgo
0	400	520.0	-120.0
1	400	520.0	-120.0
2	308	306.0	2.0
3	141	146.0	-5.0
4	141	146.0	-5.0
5	141	146.0	-5.0
6	0	1.0	-1.0
7	630	682.0	-52.0
8	630	682.0	-52.0
9	159	218.0	-59.0

Modelo Optimizado

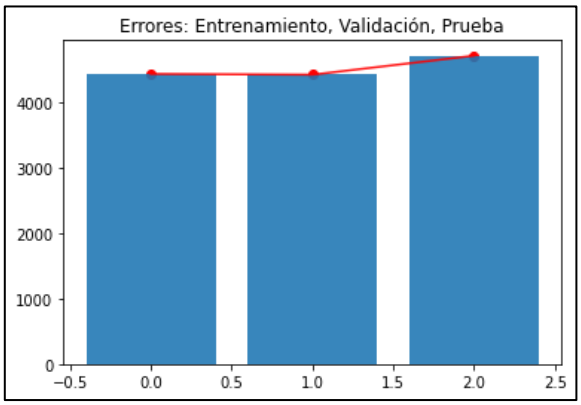
	Real	Prediccion	Diferencia / Sesgo
0	400	415.0	-15.0
1	400	415.0	-15.0
2	308	289.0	19.0
3	141	136.0	5.0
4	141	136.0	5.0
5	141	136.0	5.0
6	0	-0.0	0.0
7	630	671.0	-41.0
8	630	671.0	-41.0
9	159	168.0	-9.0

Si bien en las tablas no es tan notoria la diferencia a primera vista, se puede ir al código y revisar estas tablas más a fondo si se imprimen las tablas completas, las tablas anteriores corresponden solo a 10 muestras por lo que es difícil englobar un resultado preciso, sin embargo, aun sigue siendo notoria la gran diferencia de sesgo en algunas de las predicciones.

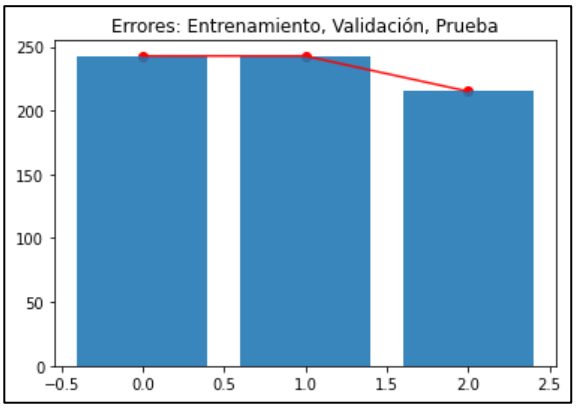
Variación

Respecto a la variación se puede observar nuevamente que los errores se mantienen dentro de un buen rango, no existe una variación muy grande para ninguno de los modelos.

Primer Modelo



Modelo Optimizado



Esto es un buen signo, pues se logró mantener una variación baja, lo cual es ideal en un buen modelo.

Nivel de ajuste del modelo

Finalmente, tras realizar la comparación, se ha de mencionar que la optimización ha logrado lo que se busca en un buen modelo, un bias y una variación baja, a raíz de ello podemos decir que:

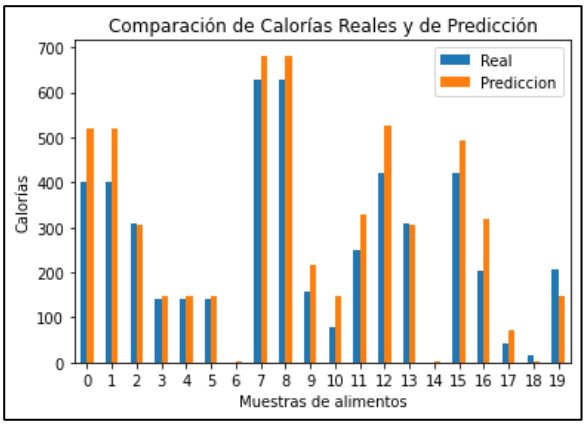
Diagnóstico de Primer Modelo: Underfitt.

Diagnóstico Modelo Optimizado: Fitt.

Comparativa Final Gráfica

A través de las siguientes gráficas se realiza una comparativa final de los resultados entre modelos y las predicciones correspondientes a cada uno de ellos.

Primer Modelo



Modelo Optimizado