

## Fase 4: Modelación de los datos

### Parte 1: Análisis de regresión en Python

```

import pandas as pd # importa la libreria pandas y la asigna a la variable pd

[ ] datos_consumo = pd.read_excel('A01253031_Registro-1_Avance.xlsx') # indicamos el nombre de nuestro archivo a ser leído

[ ] datos_consumo.head()

```

	Fecha (dd/mm/aa)	Momento	Nombre alimento	Calorías (kcal)	Carbohidratos (g)	Lípidos/grasas (g)	Proteína (g)	Sodio (mg)	Fuente
0	2022-08-17	Desayuno	Huevo con tortilla	303	23.0	17.0	18.0	233	MyFitnessPal
1	2022-08-17	Comida	Pollo asado	422	2.0	19.0	60.0	2	MyFitnessPal
2	2022-08-17	Cena	Quesadillas con frijoles	572	51.0	29.0	29.0	702	MyFitnessPal
3	2022-08-17	Snack	Nito	252	37.0	10.0	3.0	10	MyFitnessPal
4	2022-08-17	Snack	5 Picalfresas	100	25.0	0.0	0.0	225	MyFitnessPal

```

[ ] datos_consumo.groupby("Momento").count() # con la función groupby agrupamos los datos de la columna Momento y con count() los contamos para obtener subtotales

[ ] datos_consumo.describe()

```

	Fecha (dd/mm/aa)	Nombre alimento	Calorías (kcal)	Carbohidratos (g)	Lípidos/grasas (g)	Proteína (g)	Sodio (mg)	Fuente
Momento								
Cena	78	78	78	78	78	78	78	78
Comida	78	78	78	78	78	78	78	78
Desayuno	78	78	78	78	78	78	78	78
Desayuno	2	2	2	2	2	2	2	2
Snack	157	157	157	157	157	157	157	157

```

[ ]

```

	Calorías (kcal)	Carbohidratos (g)	Lípidos/grasas (g)	Proteína (g)	Sodio (mg)
count	391.000000	391.000000	391.000000	391.000000	391.000000
mean	327.138107	31.956266	15.480102	18.352174	499.051151
std	208.084178	25.044617	11.659986	14.591886	523.009822
min	40.000000	0.000000	0.000000	0.000000	0.000000
25%	165.000000	18.000000	7.000000	2.000000	180.000000
50%	300.000000	30.000000	13.000000	12.000000	320.000000
75%	406.000000	37.200000	21.950000	29.000000	702.000000
max	1061.000000	115.000000	68.000000	65.000000	3110.000000

```

[ ] datos_seleccionados = datos_consumo.iloc[:,3:8] # : selecciona todas las filas y 3:8(-1) selección columnas de la 4 a la 7

[ ] datos_seleccionados # desplegamos el dataframe

```

	Calorías (kcal)	Carbohidratos (g)	Lípidos/grasas (g)	Proteína (g)	Sodio (mg)
0	303	23.0	17.0	18.0	233
1	422	2.0	19.0	60.0	2
2	572	51.0	29.0	29.0	702
3	252	37.0	10.0	3.0	10
4	100	25.0	0.0	0.0	225
...	...	...	...	...	...
386	366	26.3	26.5	8.0	334
387	330	30.0	22.0	3.0	500
388	1052	115.0	49.0	36.0	1710
389	470	52.0	26.0	8.0	3110
390	327	52.0	5.3	17.7	653

391 rows x 5 columns

```
[ ] datos_seleccionados.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 391 entries, 0 to 390
Data columns (total 5 columns):
 #   Column              Non-Null Count  Dtype
---  -
 0   Calorias (kcal)      391 non-null   int64
 1   Carbohidratos (g)    391 non-null   float64
 2   Lipidos/grasas (g)  391 non-null   float64
 3   Proteina (g)         391 non-null   float64
 4   Sodio (mg)           391 non-null   int64
dtypes: float64(3), int64(2)
memory usage: 15.4 KB

[ ] datos_seleccionados.isnull().values.any() # buscamos valores nulos y obtenemos True o False dependiendo si hay o no

dataset = datos_seleccionados.dropna() # creamos un nuevo dataframe descartando los valores nulos o vacíos de nuestro dataframe datos_seleccionados

dataset.isnull().sum() # validamos que no tenemos valores nulos en ninguna columna, todos deben dar cero

Calorias (kcal)      0
Carbohidratos (g)    0
Lipidos/grasas (g)  0
Proteina (g)         0
Sodio (mg)           0
dtype: int64

dataset.columns # vemos los nombres de nuestras columnas para asignarlos a las variables

X = dataset[['Carbohidratos (g)', 'Lipidos/grasas (g)', 'Proteina (g)', 'Sodio (mg)']].values # variables independientes
y = dataset['Calorias (kcal)'].values # variable dependiente

[ ] from sklearn.model_selection import train_test_split # importamos la herramienta para dividir los datos de SciKit-Learn

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0) # asignación de los datos 80% para entrenamiento y 20% para prueba

[ ] from sklearn.linear_model import LinearRegression # importamos la clase de regresión lineal

modelo_regresion = LinearRegression() # modelo de regresión

[ ] modelo_regresion.fit(X_train, y_train) # aprendizaje automático con base en nuestros datos

LinearRegression()

[ ] x_columns = ['Carbohidratos (g)', 'Lipidos/grasas (g)', 'Proteina (g)', 'Sodio (mg)']
coeff_df = pd.DataFrame(modelo_regresion.coef_, x_columns, columns=['Coeficientes'])
coeff_df # despliega los coeficientes y sus valores; por cada unidad del coeficiente, su impacto en las calorías será igual a su valor

Coeficientes
Carbohidratos (g)    4.142930
Lipidos/grasas (g)  9.020501
Proteina (g)         3.985310
Sodio (mg)          -0.011956

[ ] y_pred = modelo_regresion.predict(X_test) # probamos nuestro modelo con los valores de prueba

validacion = pd.DataFrame({'Actual': y_test, 'Predicción': y_pred, 'Diferencia': y_test-y_pred}) # creamos un dataframe con los valores actuales y los de predicción

muestra_validacion = validacion.head(25) # elegimos una muestra con 25 valores

muestra_validacion # desplegamos esos 25 valores
```

	Actual	Predicción	Diferencia
0	442	442.775806	-0.775806
1	180	196.270902	-16.270902
2	343	339.926061	3.073939
3	572	576.454573	-4.454573
4	140	137.308400	2.691600
5	130	127.873612	2.126388
6	406	501.111034	-95.111034
7	130	127.873612	2.126388
8	572	576.454573	-4.454573
9	401	372.666362	28.333638
10	276	269.844275	6.155725
11	180	196.270902	-16.270902
12	122	114.111317	7.888683
13	111	104.384859	6.615141
14	276	269.844275	6.155725
15	140	137.308400	2.691600
16	252	250.959628	1.040472
17	130	123.901455	6.098545

```
[ ] validacion["Diferencia"].describe()
```

```
count    79.000000
mean      1.176466
std       42.356121
min      -95.111034
25%      -3.895230
50%       2.691600
75%       7.564434
max      285.238360
Name: Diferencia, dtype: float64
```

```
[ ] from sklearn.metrics import r2_score # importamos la métrica R cuadrada (coeficiente de determinación)
```

```
r2_score(y_test, y_pred) # ingresamos nuestros valores reales y calculados
```

```
0.9461940602031965
```

```
[ ] import matplotlib.pyplot as plt # importamos la librería que nos permitirá graficar
```

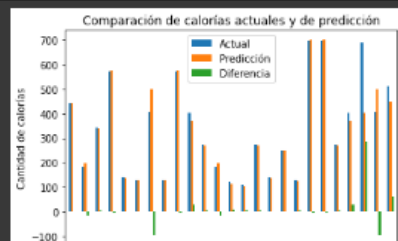
```
muestra_validacion.plot.bar(rot=0) # creamos un gráfico de barras con el dataframe que contiene nuestros datos actuales y de predicción
```

```
plt.title("Comparación de calorías actuales y de predicción") # indicamos el título del gráfico
```

```
plt.xlabel("Muestra de alimentos") # indicamos la etiqueta del eje de las x, los alimentos
```

```
plt.ylabel("Cantidad de calorías") # indicamos la etiqueta del eje de las y, la cantidad de calorías
```

```
plt.show() # desplegamos el gráfico
```



```
import pandas as pd # importa la librería pandas y la asigna a la
variable pd
```

```
datos_consumo = pd.read_excel('A01253031_Registro-1_Avance.xlsx') #
indicamos el nombre de nuestro archivo a ser leído
```

```
datos_consumo.groupby("Momento").count() # con la función groupby
agrupamos los datos de la columna Momento y con count() los contamos
para obtener subtotales
```

```
datos_consumo.describe()
```

```
datos_seleccionados = datos_consumo.iloc[:,3:8] # : selecciona todas
las filas y 3:8(-1) selecciona columnas de la 4 a la 7
```

```
datos_seleccionados # desplegamos el dataframe
```

```
datos_seleccionados.info()
```

```
datos_seleccionados.isnull().values.any() # buscamos valores nulos y
obtenemos True o False dependiendo si hay o no
```

```
dataset = datos_seleccionados.dropna() # creamos un nuevo dataframe
descartando los valores nulos o vacíos de nuestro dataframe
datos_seleccionados

dataset.isnull().sum() # validamos que no tenemos valores nulos en
ninguna columna, todos deben dar cero
dataset.columns # vemos los nombres de nuestras columnas para
asignarlos a las variables

X = dataset[['Carbohidratos (g)', 'Lípidos/grasas (g)', 'Proteína (g)',
'Sodio (mg)']].values # variables independientes

y = dataset['Calorías (kcal)'].values # variable dependiente
from sklearn.model_selection import train_test_split # importamos la
herramienta para dividir los datos de SciKit-Learn

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=0) # asignación de los datos 80% para
entrenamiento y 20% para prueba
from sklearn.linear_model import LinearRegression # importamos la clase
de regresión lineal

modelo_regresion = LinearRegression() # modelo de regresión
modelo_regresion.fit(X_train, y_train) # aprendizaje automático con
base en nuestros datos
x_columns = ['Carbohidratos (g)', 'Lípidos/grasas (g)', 'Proteína (g)',
'Sodio (mg)']
coeff_df = pd.DataFrame(modelo_regresion.coef_, x_columns,
columns=['Coeficientes'])
coeff_df # despliega los coeficientes y sus valores; por cada unidad del
coeficiente, su impacto en las calorías será igual a su valor
y_pred = modelo_regresion.predict(X_test) # probamos nuestro modelo con
los valores de prueba
validacion = pd.DataFrame({'Actual': y_test, 'Predicción': y_pred,
'Diferencia': y_test-y_pred}) # creamos un dataframe con los valores
actuales y los de predicción

muestra_validacion = validacion.head(25) # elegimos una muestra con 25
valores
```

```
muestra_validacion # desplegamos esos 25 valores
validacion["Diferencia"].describe()
from sklearn.metrics import r2_score # importamos la métrica R cuadrada
(coeficiente de determinación)

r2_score(y_test, y_pred) # ingresamos nuestros valores reales y
calculados
import matplotlib.pyplot as plt # importamos la librería que nos
permitirá graficar

muestra_validacion.plot.bar(rot=0) # creamos un gráfico de barras con
el dataframe que contiene nuestros datos actuales y de predicción

plt.title("Comparación de calorías actuales y de predicción") #
indicamos el título del gráfico

plt.xlabel("Muestra de alimentos") # indicamos la etiqueta del eje de
las x, los alimentos

plt.ylabel("Cantidad de calorías") # indicamos la etiqueta del eje de
las y, la cantidad de calorías

plt.show() # desplegamos el gráfico
```

Lo que hice fue básicamente importar la librería Pandas para poder analizar los datos, ya que la importe utilicé la función groupby, para agrupar los datos de la columna Momento y con la función count contamos la cantidad de momentos. Después obtuvimos una estadística descriptiva para poder seleccionar los datos. Hice una variable datos para asignarle únicamente los datos que iba a analizar, para proceder a limpiar los datos. Utilice varias funciones para buscar valores nulos y se creó un nuevo dataframe con los datos que no sean nulos. Después de ahí le asigne la variable X a los atributos de entrada y Y a los de salida. Dividí mis datos uno en un conjunto de entrenamiento y otro conjunto de prueba, modele los datos y ya nomas visualice los datos para hacer una comparación.

**Parte 2: Modelación de los datos**

Los datos que se analizan por lo general se procesan utilizando herramientas tecnológicas como lenguajes de programación, ya sea Python, Java, C++ entre otros. Casi siempre se ejecutan varios modelos y luego se deben ajustar dichos parámetros. Por lo general hay casos en los cuales hay modelos que más se adaptan a algunas ocasiones en específico, pero para poder saber con más exactitud se debe examinar los resultados de cada modelo utilizado y se debe evaluar el modelo. Para cada modelo se debe realizar una evaluación basado en los criterios que se tengan, esto podría ser muy útil como una base, luego para cada modelo se puede generar una lista de resultados y apoyarlo con el uso de gráficos para tener un mejor análisis de los resultados. Otro punto importante es que los resultados deben de tener un sentido lógico. De ahí seguiría clasificar los modelos según ciertos criterios, como lo podrían ser los objetivos, precisión del modelo y subjetivos y facilidad de uso o interpretación de los resultados. Con base a los resultados que se tienen es importante realizar una revisión exhaustiva de estos, consultar si es posible con otros analistas de datos, considerar si los resultados son fáciles de desplegar y por último comprobar si los resultados cumplen con los objetivos del problema o lo que se quiera resolver.

**1. ¿Cuántos intentos o corridas realizaste para obtener los resultados sin errores? Porqué**

Gracias a que desde que empecé a registrar los datos y me percate de que tenía que ser más objetivo con mis registros, yo creo que no ocupe más que 1 intento para obtener los resultados sin errores y yo creo que por eso es por lo que no ocupe otro intento.

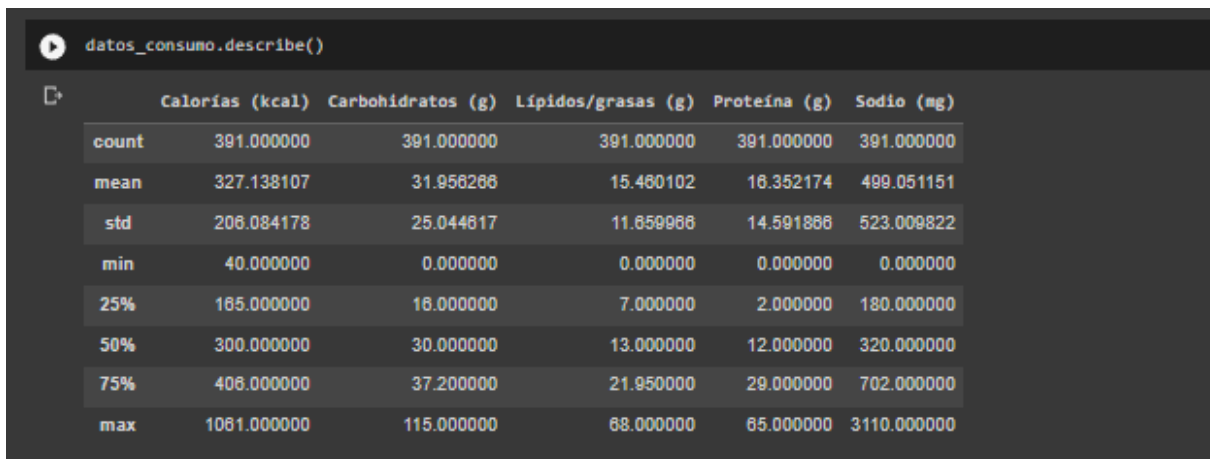
**2. ¿Cómo los resolviste los problemas que se presentaron?**

14 de Noviembre del 2022

Para poder resolver los problemas que se presentaron fue consultarlo con mis compañeros y comparar con los datos que a ellos les arrojaban y en base a eso, podía tener un mejor panorama del problema que tenía y así intentar replicar el camino que ellos realizaron para que mi análisis sea más exacto, aunque sus datos fueran diferentes.

### 3. ¿Qué resultados arrojó el análisis? Incluye imagen de cada resultado y explica cada uno de los resultados:

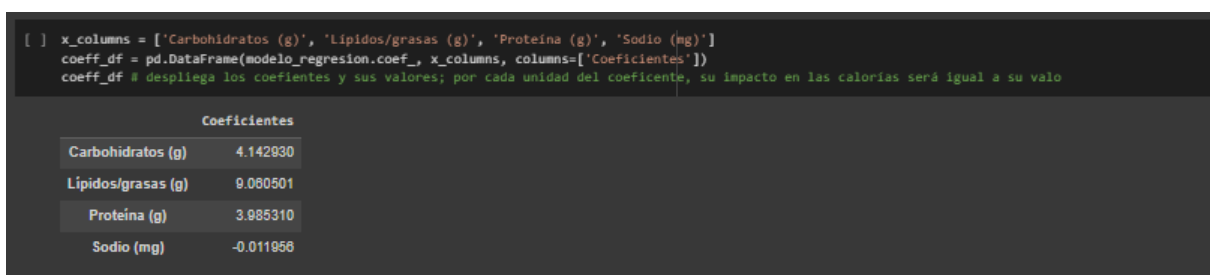
#### 1. Estadística descriptiva



	Calorías (kcal)	Carbohidratos (g)	Lípidos/grasas (g)	Proteína (g)	Sodio (mg)
count	391.000000	391.000000	391.000000	391.000000	391.000000
mean	327.138107	31.956266	15.460102	16.352174	499.051151
std	206.084178	25.044617	11.659968	14.591866	523.009822
min	40.000000	0.000000	0.000000	0.000000	0.000000
25%	165.000000	16.000000	7.000000	2.000000	180.000000
50%	300.000000	30.000000	13.000000	12.000000	320.000000
75%	406.000000	37.200000	21.950000	29.000000	702.000000
max	1061.000000	115.000000	68.000000	65.000000	3110.000000

Lo que se observa aquí es básicamente es una estadística descriptiva con solo los nutrientes en el cual se realizan varios análisis como el mínimo y el máximo entre otros resultados más que me ayudan al análisis que se quiere realizar.

#### 2. Coefficientes de regresión



```
[ ] x_columns = ['Carbohidratos (g)', 'Lípidos/grasas (g)', 'Proteína (g)', 'Sodio (mg)']
coeff_df = pd.DataFrame(modelo_regresion.coef_, x_columns, columns=['Coeficientes'])
coeff_df # despliega los coeficientes y sus valores; por cada unidad del coeficiente, su impacto en las calorías será igual a su valo
```

Coeficientes	
Carbohidratos (g)	4.142930
Lípidos/grasas (g)	9.060501
Proteína (g)	3.985310
Sodio (mg)	-0.011956

14 de Noviembre del 2022

Aquí el algoritmo ya ha aprendido cuáles son los coeficientes de X óptimos para satisfacer el modelo.

### 3. Valores actuales y de predicción

```
[ ] validation = pd.DataFrame({'Actual': y_test, 'Predicción': y_pred, 'Diferencia': y_test-y_pred}) # creamos un dataframe con los valores actuales y los de predicción
muestra_validation = validation.head(25) # elegimos una muestra con 25 valores
muestra_validation # desplegamos esos 25 valores
```

	Actual	Predicción	Diferencia
0	442	442.775806	-0.775806
1	180	196.270902	-16.270902
2	343	339.926061	3.073939
3	572	576.454573	-4.454573
4	140	137.308400	2.691600
5	130	127.873612	2.126388
6	406	501.111034	-95.111034
7	130	127.873612	2.126388
8	572	576.454573	-4.454573
9	401	372.666362	28.333638
10	276	269.844275	6.155725
11	180	196.270902	-16.270902
12	122	114.111317	7.888683
13	111	104.384859	6.615141
14	276	269.844275	6.155725
15	140	137.308400	2.691600
16	252	250.959528	1.040472
17	130	123.901455	6.098545
18	696	699.335887	-3.335887
19	696	699.335887	-3.335887
20	276	269.844275	6.155725
21	401	372.666362	28.333638
22	690	404.761640	285.238360
23	406	501.111034	-95.111034
24	511	448.904655	62.095345

Se generó una tabla con una comparación de los valores actuales y de predicción en el cual se muestran 25 valores y la diferencia que existe entre los valores.

### 4. Coeficiente de determinación r2

```
from sklearn.metrics import r2_score # importamos la métrica R cuadrada (coeficiente de determinación)
r2_score(y_test, y_pred) # ingresamos nuestros valores reales y calculados
```

0.9461940602031965

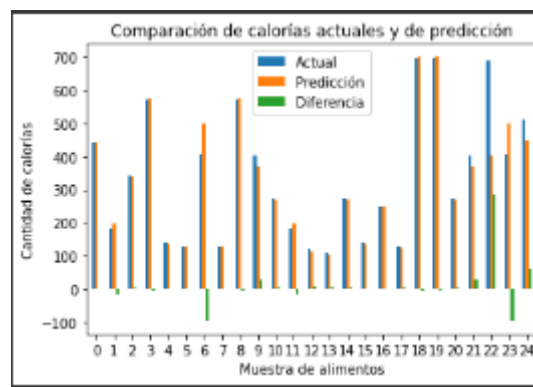
Luego aquí tenemos el coeficiente de determinación de r2, que no es muy útil, ya que entre mayor sea el R2, mejor será el ajuste del modelo a los



14 de Noviembre del 2022

datos. Se espera que sea un valor lo más cercano a 1 por lo cual a mi parecer es muy cercano.

## 5. Gráfica



Lo que obtuve en esta gráfica fue la comparación de calorías actuales y de predicción, en los cuales si se puede observar hay 3 colores en la tabla, una representa las calorías actuales, otro la predicción con naranja y por último el verde con la diferencia que existe entre los datos actuales y a los de predicción.

## 4. ¿Cuáles son tus conclusiones de la modelación?

En conclusión utilizar herramientas tecnológicas como Python son esenciales para la modelación de datos, ya que estos mismos nos ayudan a tener un mejor análisis de los datos y tener muchos mejores resultados que sean entendibles para poder resolver el problema que se intenta resolver.

Hector Gutierrez

A01253031

14 de Noviembre del 2022

Link google colab: <https://colab.research.google.com/drive/1kNiUYiZwulTU63b9e-OMhh6HloDREPa9?usp=sharing>