Fase 4: modelación de los datos Implementación del código con nuestra base de datos

```
#importamos las librerías a utilizar
import pandas as pd
#designamos la variable donde se van a almacenar nuestros datos de excel
datos_consumo = pd.read_excel('datos_nutricionales.xlsx')

#comprobamos que los datos se hayan cargado correctamente utilizando head
datos_consumo.head()
```

	Fecha (dd/mm/aa)	Momento	Nombre alimento	Calorías (kcal)	Carbohidratos (g)	Lípidos/grasas (g)	P
0	2022-02-18	desayuno	2 Burritos de pollo	542	48.68	16.62	
1	2022-02-18	desayuno	1 L de jugo de piña	136	128.72	1.20	
2	2022-02-18	snack	2 huevos revueltos	199	1.96	15.21	
3	2022-02-18	snack	1 chocolate	70	5.00	5.00	
4	2022-02-18	comida	1 ensalada de pollo	318	17.55	11.80	

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

Momento	Fecha (dd/mm/aa)	Nombre alimento	Calorías (kcal)	Carbohidratos (g)	Lípidos/grasas (g)	Pro
cena	61	61	61	61	61	
cena	1	1	1	1	1	

```
#obtenemos la estadística descriptiva con describe()
datos_consumo.describe()
```

	Calorías (kcal)	Carbohidratos (g)	Lípidos/grasas (g)	Proteína (g)	Sodio (mg)
count	286.000000	286.000000	286.000000	286.000000	286.000000
mean	227.472028	29.708178	10.153357	17.311783	294.643846
std	193.817307	47.706899	11.960306	22.929374	501.776394
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	103.000000	1.960000	1.050000	1.300000	2.000000
50%	156.000000	11.000000	6.900000	12.580000	62.000000
75%	372.000000	33.090000	15.210000	22.780000	211.000000
max	1008.000000	536.000000	60.040000	109.160000	2196.000000

Seleccionando los datos

```
datos_seleccionados = datos_consumo.iloc[:,3:8]# : selecciona todas las filas y 3:8(-
```

```
datos_seleccionados.info
```

```
<bound method DataFrame.info of      Calorías (kcal)  Carbohidratos (g)  Lípidos
0                542             48.68              16.62         46.98
1                136            128.72               1.20          3.60
2                199              1.96              15.21         13.01
3                 70              5.00               5.00          1.00
4                318             17.55              11.80         34.81
..               ...               ...                ...           ...
281              513             31.59              27.70         26.50
282              113             15.00               5.00          2.00
283              146             11.03               7.93          7.86
284               94             22.13               0.25          1.28
285               61             11.00               1.00          2.00

     Sodio (mg)
0        1354.0
1          20.0
2         211.0
```

```
3           5.0
4         459.0
..          ...
281      1227.0
282        62.0
283        98.0
284         2.0
285        85.0

[286 rows x 5 columns]>
```

```
datos_seleccionados.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 286 entries, 0 to 285
Data columns (total 5 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Calorías (kcal)     286 non-null    int64  
 1   Carbohidratos (g)   286 non-null    float64
 2   Lípidos/grasas (g)  286 non-null    float64
 3   Proteína (g)        286 non-null    float64
 4   Sodio (mg)          286 non-null    float64
dtypes: float64(4), int64(1)
memory usage: 11.3 KB
```

Limpiamos los datos

```
#buscamos valores nulos y obtenemos true o false si es que hay o no
datos_seleccionados.isnull().values.any()
```

```
False
```

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

```
#validamos que no tengamos datos nulos
dataset.isnull().sum() #todos deben de dar cero
```

```
Calorías (kcal)       0
Carbohidratos (g)     0
Lípidos/grasas (g)    0
Proteína (g)          0
Sodio (mg)            0
dtype: int64
```

Preparando los datos

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

```
Index(['Calorías (kcal)', 'Carbohidratos (g)', 'Lípidos/grasas (g)',
       'Proteína (g)', 'Sodio (mg)'],
      dtype='object')
```

```
X = dataset[['Carbohidratos (g)', 'Lípidos/grasas (g)', 'Proteína (g)', 'Sodio (mg)']

y = dataset['Calorías (kcal)'].values # variable dependiente
```

Ahora dividimos nuestros datos en un conjunto de entrenamiento (80%) y un conjunto de prueba (20%). Con los datos de entrenamiento realizamos el aprendizaje automático y con los datos de prueba realizamos la validación:

```
from sklearn.model_selection import train_test_split # importamos la herramienta para

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

Modelación de los datos

```
from sklearn.linear_model import LinearRegression # importamos la clase de regresión

modelo_regresion = LinearRegression() # modelo de regresión
```

```
#usamos fit para ajustaar los datos
modelo_regresion.fit(X_train, y_train) # aprendizaje automático con base en nuestros
```

```
LinearRegression()
```

En este punto, el algorítmo ya ha aprendido cuales son los coeficientes de X óptimos para satisfacer el modelo. Para verlos copiamos el siguiente código:

```
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 coefientes y sus valores; por cada unidad del coeficente, su
```

	Coeficientes
Carbohidratos (g)	0.616400

```
#probamos los datos con un conjunto prueba
y_pred = modelo_regresion.predict(X_test)
```

Sodio (mg)	0.024000

```
#Comparamos el modelo de predicción con los valores actuales
validacion = pd.DataFrame({'Actual': y_test, 'Predicción': y_pred, 'Diferencia': y_te

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

muestra_validacion # desplegamos esos 25 valores
```

	Actual	Predicción	Diferencia
0	168	209.740470	-41.740470
1	199	241.567877	-42.567877
2	72	64.856379	7.143621

```
#Usamos la función describe() para obtener la estadística descriptiva de la columna D

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

```
count     58.000000
mean     -22.755592
std       90.813676
min     -188.384501
25%      -42.567877
50%      -28.243325
75%       25.024259
max      253.151135
Name: Diferencia, dtype: float64
```

Calculamos el coeficiente de determinación para comprobar nuestro modelo

13	126	75.447756	50.552244

```
from sklearn.metrics import r2_score # importamos la métrica R cuadrada (coeficiente

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

```
0.7765372832699708
```

17	542	443.757612	98.242388

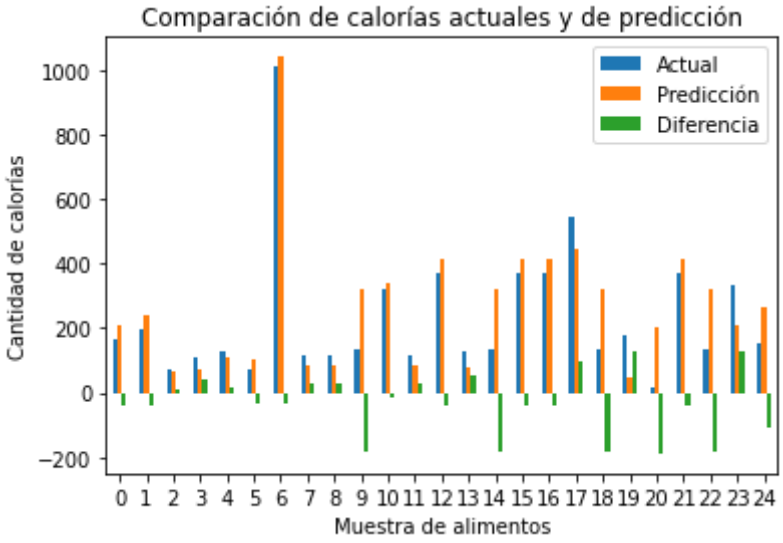
Notamos que nuestro coeficiente de determinación es menor al 90%, lo que nos indica que no es muy acertado

Validación de los datos (gráfica)

Haz doble clic (o pulsa Intro) para editar

```
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 q

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

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

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

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



Parte 2: Modelación de los datos ¿en qué consiste la Fase 4: modelación de los datos?

- La Fase 4 consiste en la Visualización de Datos, la cual ha hecho que sea más fácil el poder determinar cuál de los modelos matemáticos sería el más acorde a un problema específico. Dentro de los formas que se puede generar una visualización de los datos se encuentran: por relaciones, coposición, comparación, distribución. De la misma forma que, en esta fase, se trata de establecer un modelo de regresión lineal para crear así un modelo matemático. Para comprobar la vaalidéz del modelo, se realizan pruebas de hi`ótesis, las cuales son la hipótesis nula y la hipótesis alternaitva, que tiene como objetivo, minimizar el error.

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

- Realicé varias corridaas de los datos, tanto para asegurarme de que estos estuvieran completos,así como el código estuviera bien implementado

¿Cómo los resolviste los problemas que se presentaron?

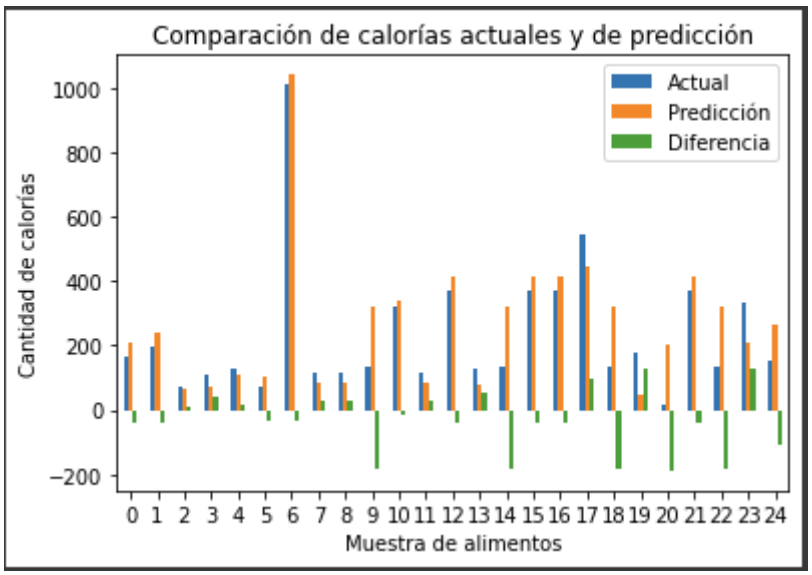
- No se presentaron muchos problemas, pero algunos que tuve, en su mayoría era por no haber escrito bien algunas partes de código o inlcuso por no haber cargado la hoja de datos de excel adecuadamente

¿Qué resultados arrojó el análisis? Incluye imagen de cada resultado y explica cada uno de los resultados: Estadística descriptiva: conjunto de valores que puede tomar una determinada característica de la población Coeficientes de regresión: indica que tan acertado es nuestro modelo Valores actuales y de predicción: se compara nuestro modelo con los valores de predicción con los actuales, para tener una mejor visualizasón de los datos Coeficiente de determinación r2: nos indica qué tanto se ajusta nuestro modelo a los datos

Representación de los datos anteriores con base en nuestro conjunto de datos

count	58.000000
mean	-22.755592
std	90.813676
min	-188.384501
25%	-42.567877
50%	-28.243325
75%	25.024259
max	253.151135

Gráfica de nuestro modelo



¿Cuáles son tus conclusiones de la modelación?

- no es un modelo muy preciso ya que, su coeficiente de determinación posee un valor del 77%, sin embargo puede ser útil.

✓ 1 s completado a las 11:24