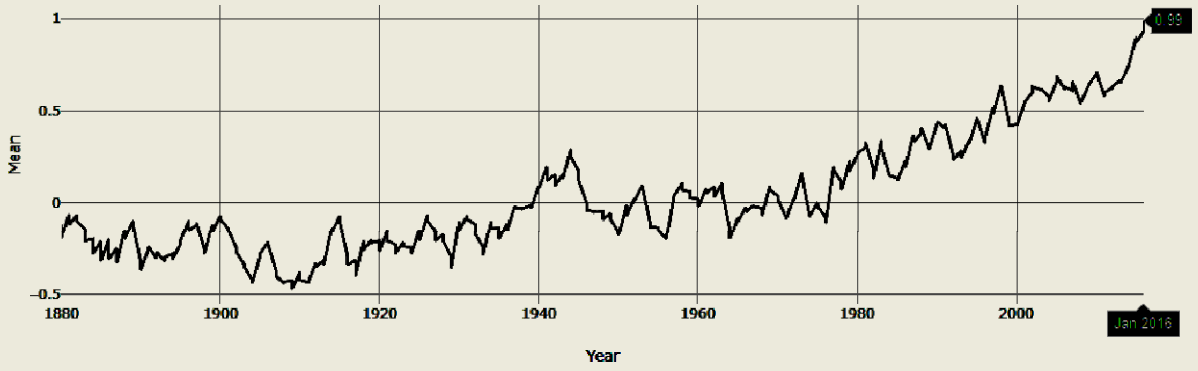
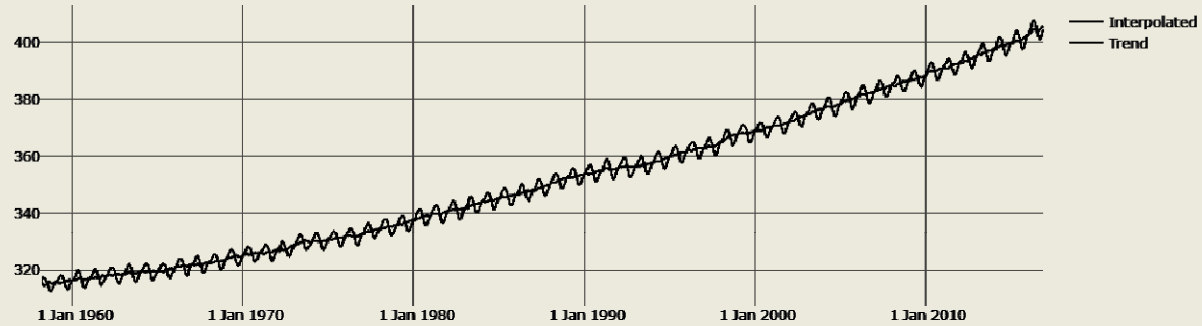
ML_REGRESION_01	Modelo de Regresión Lineal	ML

La temperatura media del planeta Tierra y la concentración de dióxido de carbono CO2 de la atmósfera presentan una tendencia ascendente durante las últimas décadas.



Anomalía de la temperatura media mensual en grados Celsius relativos a un periodo base

Fuente: *https://datahub.io/core/global-temp#resource-annual*



Tendencia del dióxido de carbono (CO2) de la atmósfera

Fuente: *https://pkgstore.datahub.io/core/co2-ppm/co2-annmean-mlo_json/data/31185d494d1a6f6431aee8b8004b6164/co2-annmean-mlo_json.json*

El problema a realizar con esta práctica consiste en realizar un modelo de aprendizaje supervisado basado en regresión, verificar si la relación entre la temperatura y CO2 se ajusta bien a un modelo de regresión lineal o no, y utilizarlo para predecir valores futuros.

SOLUCIÓN

Importar las librerías necesarias para realizar la práctica.

```
# Importar librerías
import urllib.request, json
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

Descargar y leer la información de partida (Datasets) y leer la temperatura y co2

```
# Información de partida (Datasets)
# Fuente:

# Temperatura: https://datahub.io/core/global-temp#data
# Temperatura anual media (formato json)
# https://pkgstore.datahub.io/core/global-
temp/annual_json/data/529e69dbd597709e36ce11a5d0bb7243/annual_json.json
temp_url="https://pkgstore.datahub.io/core/global-
temp/annual_json/data/529e69dbd597709e36ce11a5d0bb7243/annual_json.json"

# CO2: https://datahub.io/core/co2-ppm#data
# Concentración anual media (formato json)
# https://pkgstore.datahub.io/core/co2-ppm/co2-annmean-
mlo_json/data/31185d494d1a6f6431aee8b8004b6164/co2-annmean-mlo_json.json
co2_url="https://pkgstore.datahub.io/core/co2-ppm/co2-annmean-
mlo_json/data/31185d494d1a6f6431aee8b8004b6164/co2-annmean-mlo_json.json"

# lectura de la información de los ficheros JSON
with urllib.request.urlopen(temp_url) as url:
    temp_data = json.loads(url.read().decode())

with urllib.request.urlopen(co2_url) as url:
    co2_data = json.loads(url.read().decode())
```

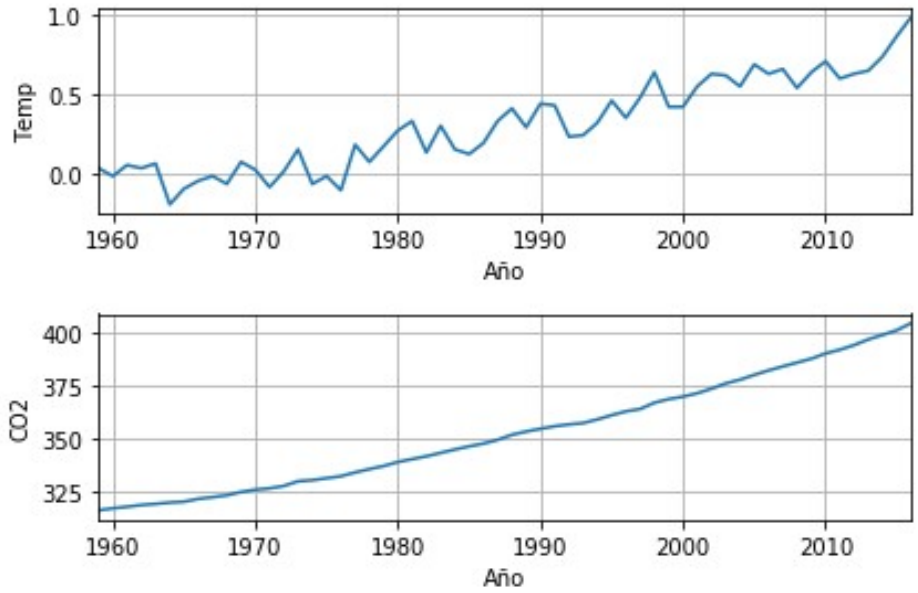
Registrar los valores en listas.

```
# Registro de las variables (listas) de temperatura y co2
temp=[]
co2=[]
year=[]

ntemp=len(temp_data)
nco2=len(co2_data)

# Registro de temperaturas desde el año 1880
for i in range(ntemp):
    if temp_data[i]["Source"]=="GISTEMP":
        # Se utiliza la temperatura media en superficie (NASA)
        # GISTEMP:  https://data.giss.nasa.gov/gistemp/
```

```
        temp.append(temp_data[i]["Mean"])
        year.append(temp_data[i]["Year"])

# Las listas de temperatura y años están en orden decreciente (de 2016 a
1880)
# Las ordenamos en orden creciente
temp.reverse()
year.reverse()
# y nos quedamos con la serie desde 1959 hasta 2016
# En total son 58 registros
temp=temp[1959-1880:2016-1880+1]
year=year[1959-1880:2016-1880+1]

# Registro de CO2 desde el año 1959
for i in range(nco2):
    co2.append(co2_data[i]["Mean"])
```

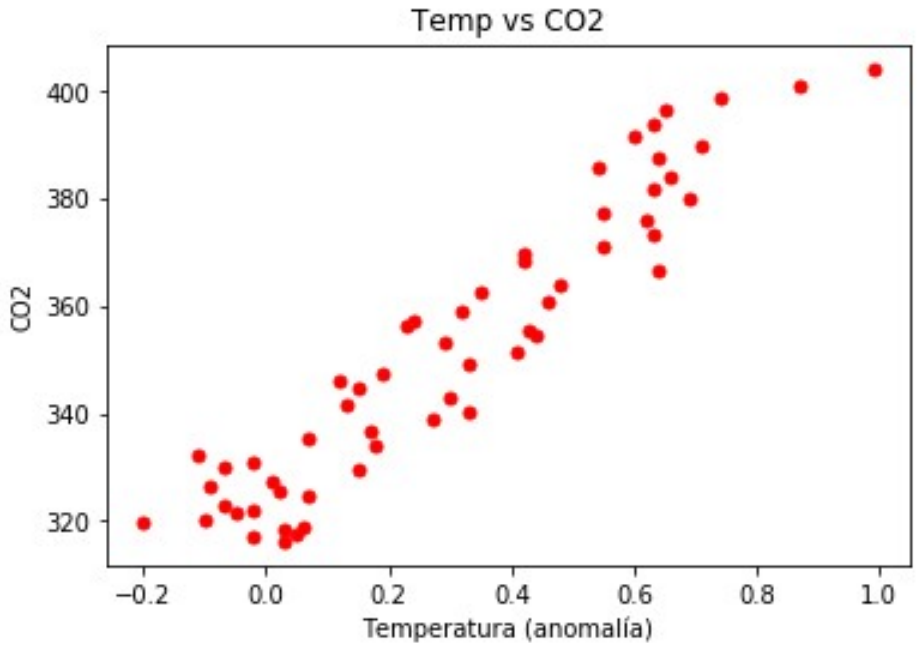
Visualizar la temperatura y CO2 por años

```
# Visualizamos la temperatura y CO2
fig, axs = plt.subplots(2,1)

# Temperatura
axs[0].plot(year,temp)
axs[0].set_xlim(1959,2016)
axs[0].set_xlabel("Año")
axs[0].set_ylabel("Temp")
axs[0].grid(True)

# CO2
axs[1].plot(year,co2)
axs[1].set_xlim(1959,2016)
axs[1].set_xlabel("Año")
axs[1].set_ylabel("CO2")
axs[1].grid(True)

fig.tight_layout()
plt.show()
```



Representar la temperatura frente al CO2

```
# Representamos la temperatura frente al CO2

l = plt.plot(temp,co2,'ro')
plt.setp(l, markersize=5)
plt.title("Temp vs CO2")
plt.xlabel("Temperatura (anomalía)")
plt.ylabel("CO2")
plt.show()
```



Realizar el modelo de regresión lineal. Previamente convertimos las listas de temperatura y CO2 en un DataFrame utilizando la librería Pandas.

```
# REGRESIÓN LINEAL (APRENDIZAJE SUPERVISADO)

# Utilizar la serie temporal de temperaturas y CO2 de 1959 hasta el año 2016
# para construir un modelo de regresión lineal

# Creamos un Dataframe con los datos utilizando la librería Pandas
datos={'temp':temp,'co2':co2}
df=pd.DataFrame(datos,columns=['temp','co2'])

# Asignamos las variables X (atributos) e y (etiquetas)
X=df[['temp']]
y=df[['co2']]
```

Importar la librería 'sklearn' para realizar el modelo de regresión.

```
# importamos las librerías para realizar regresión lineal
# Utilizamos sklearn (http://scikit-learn.org/stable/)
# Aprendizaje Supervisado: http://scikit-
learn.org/stable/supervised_learning.html#supervised-learning
# Ejemplo de Regresión Lineal: http://scikit-
learn.org/stable/auto_examples/linear_model/plot_ols.html#sphx-glr-auto-
examples-linear-model-plot-ols-py
```

```
from sklearn import linear_model
from sklearn.metrics import mean_squared_error, r2_score
```

Dividimos el conjunto de datos para entrenamiento y test

```
# Dividimos el conjunto de datos para entrenamiento y test
# Elegimos a priori el 70 % (40 registros) para entrenamiento
# y el resto 30 % (18 registros) para test

X_train = np.array(X[:40])
y_train = np.array(y[:40])

X_test = np.array(X[40:])
y_test = np.array(y[40:])
```

Realizar el ajuste del modelo de regresión lineal, y la predicción para los datos de entrenamiento.

```
# Creamos el objeto de Regresión Lineal
regr=linear_model.LinearRegression()

# Entrenamos el modelo
regr.fit(X_train,y_train)

# Realizamos predicciones sobre los atributos de entrenamiento
y_pred = regr.predict(X_train)
```

Obtener los parámetros del modelo de regresión.

```
# Recta de Regresión Lineal (y=t0+t1*X)
# Pendiente de la recta
t1=regr.coef_
print('Pendiente: \n', t1)
# Corte con el eje Y (en X=0)
t0=regr.intercept_
print('Término independiente: \n', t0)

# Ecuación de la recta
print('La recta de regresión es: y = %f + %f * X'%(t0,t1))
```

```
Pendiente:
 [[68.69130866]]
Término independiente:
 [327.32809716]
La recta de regresión es: y = 327.328097 + 68.691309 * X
```

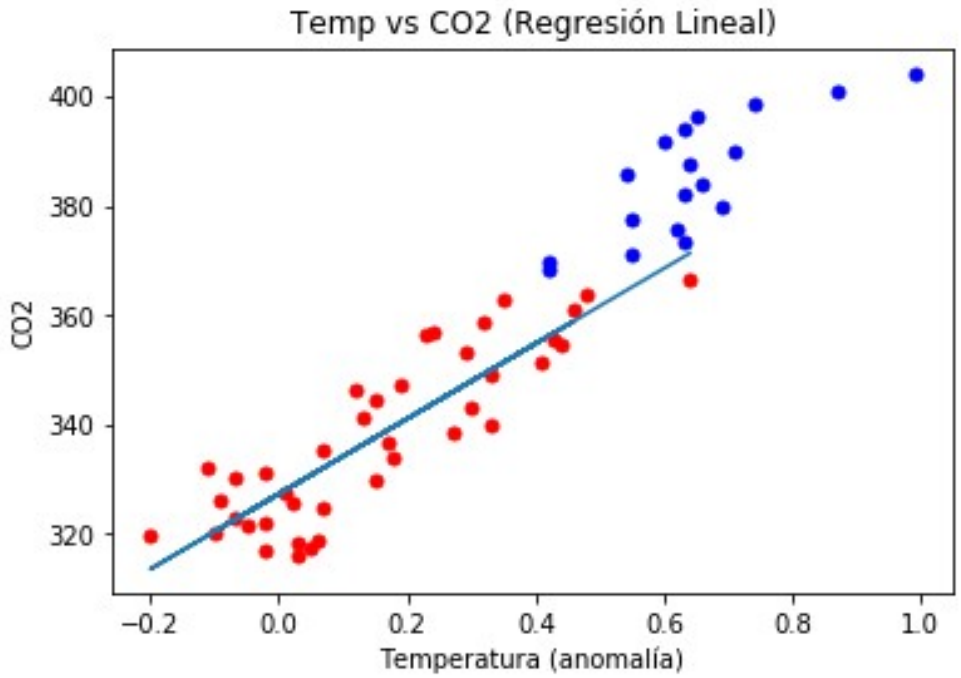
Cálculo del error (pérdida) del ajuste del modelo de regresión

```
# Error (pérdida)
print("Cálculo del error o pérdida del modelo de regresión lineal")
# Error Cuadrático Medio (Mean Square Error)
print("ECM : %.2f" % mean_squared_error(y_train, y_pred))
# Puntaje de Varianza. El mejor puntaje es un 1.0
print('Coeficiente Correlacción: %.2f' % r2_score(y_train, y_pred))
```

```
Cálculo del error o pérdida del modelo de regresión lineal
ECM : 57.17
Coeficiente Correlacción: 0.76
```

Visualizar la recta de regresión y los puntos de entrenamiento y test

```
# Dibujamos la recta de regresión
tr = plt.plot(X_train,y_train,'ro')
plt.setp(tr, markersize=5)

te = plt.plot(X_test,y_test,'bo')
plt.setp(te, markersize=5)

plt.title("Temp vs CO2 (Regresión Lineal)")
plt.xlabel("Temperatura (anomalía)")
plt.ylabel("CO2")

plt.plot(X_train,y_pred)
plt.show()
```



Comprobar con los valores de test y prueba si el ajuste es bueno o no.

```
# Con el modelo de regresión ajustado con los valores de entrenamiento
# se aplica a los valores para test y validación
y_pred_test = regr.predict(X_test)
# Comprobar el error del modelo con los valores para test
# Error (pérdida)
print("Cálculo del error o pérdida del modelo de regresión lineal")
# Error Cuadrático Medio (Mean Square Error)
print("ECM : %.2f" % mean_squared_error(y_test, y_pred_test))
# Puntaje de Varianza. El mejor puntaje es un 1.0
print('Coeficiente Correlacción: %.2f' % r2_score(y_test, y_pred_test))
```

```
Cálculo del error o pérdida del modelo de regresión lineal
ECM : 227.81
Coeficiente Correlacción: -0.94
```

Utilizar el modelo para predecir valores de la concentración de CO2

```
# Predecir la concentración de CO2 para una anomalía de 0.8
y_pred2 = regr.predict(0.8)
print('La predicción de CO2 para una anomalía de 0.8°C es: ',y_pred2)
```

```
La predicción de CO2 para una anomalía de 0.8°C es:  [[382.28114408]]
```