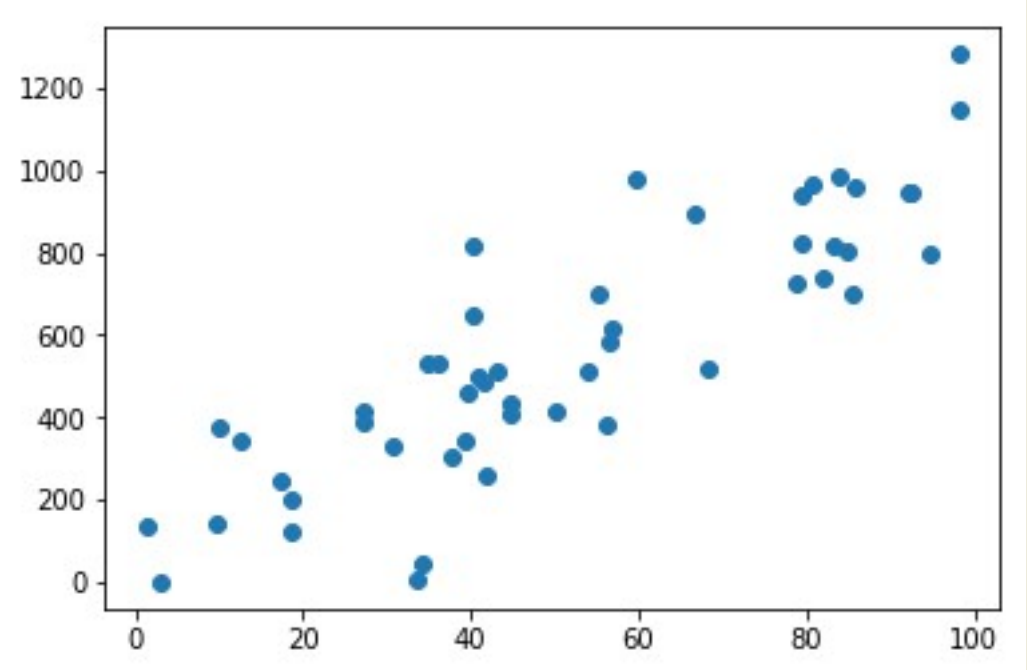
ML_REGRESION_02	Modelo de Regresión Lineal	ML

En esta práctica se genera una muestra aleatoria de 'n' valores (variable X) entre 0 y 100, así como las etiquetas 'y' teniendo en cuenta también un error aleatorio.



El problema a realizar con esta práctica consiste en realizar un modelo de aprendizaje supervisado basado en regresión, verificar si la relación entre el atributo (X) y la etiqueta (y) 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 numpy as np
import random
from sklearn import linear_model
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
```

Definir la función de generación de datos y los parámetros de la distribución

```
# Definir las funciones
# Generador de distribución de datos para regresión lineal simple
def generador_datos_simple(beta, muestras, desviacion):
    # Genera n (muestras) de valores de x aleatorios entre 0 y 100
    X = np.random.random(muestras) * 100
    # Genera un error aleatorio gaussiano con desviación típica (desviacion)
    e = np.random.randn(muestras) * desviacion
    # Obtener la etiqueta (y) real como x*beta + error
    y = X * beta + e
    return X.reshape((muestras,1)), y.reshape((muestras,1))

# Parámetros de la distribución
desviacion = 200
beta = 10
n = 50
X, y = generador_datos_simple(beta, n, desviacion)
```

Representar los datos generados

```
# Represento los datos generados
plt.scatter(X, y)
plt.show()
```

Realizar el modelo de regresión lineal. Dividimos los datos para entrenamiento y test

```
# Dividimos el conjunto de datos para entrenamiento y test
# Elegimos a priori el 70 % (n*0.7) para entrenamiento
# y el resto 30 % (n*0.3) para test
n_train=int(n*0.7)

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

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

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:
 [[8.97412436]]
Término independiente:
 [88.86663924]
La recta de regresión es: y = 88.866639 + 8.974124 * 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 : 25774.79
Coeficiente Correlacción: 0.70
```

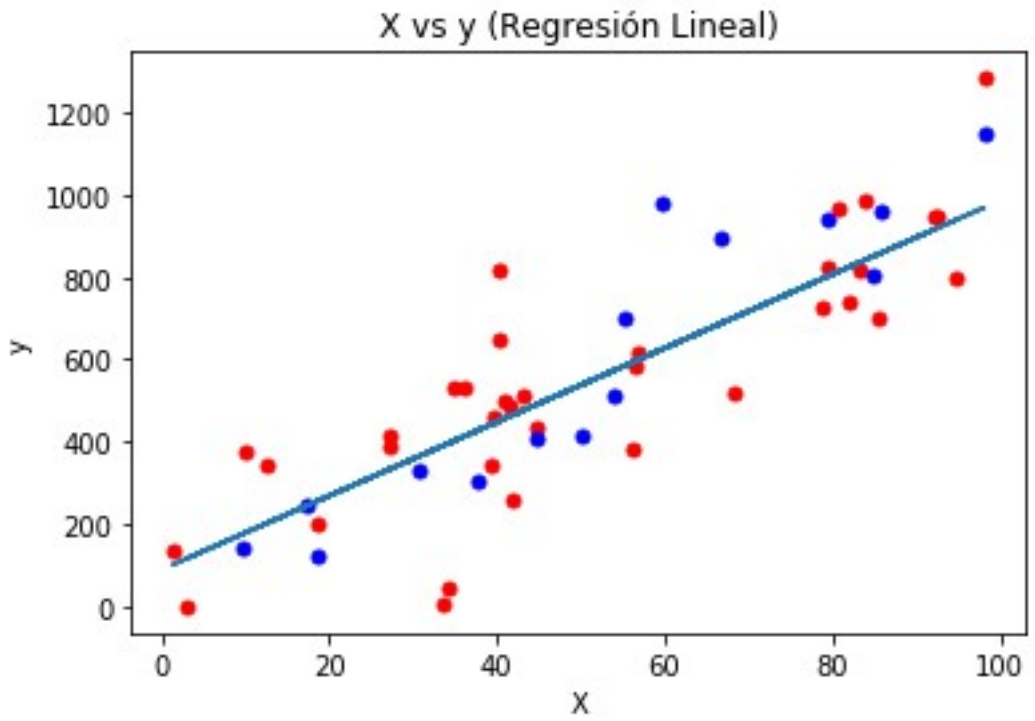
Visualizar la recta de regresión y los puntos de entrenamiento y test

```
# 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("X vs y (Regresión Lineal)")
plt.xlabel("X")
plt.ylabel("y")

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))

Error o pérdida del modelo de regresión lineal con datos para test
ECM : 20754.57
Coeficiente Correlacción: 0.81
```

Utilizar el modelo para predecir valores de la etiqueta 'y'

```
# Predecir la etiqueta 'y' para un valor X=50
y_pred2 = regr.predict(50)
print('La predicción de y para un valor X de 50 es: ',y_pred2)

La predicción de y para un valor X de 50 es:  [[461.91084362]]
```