Regresión Lineal

Ejemplo. Predecir el costo de un diamante (price) dado su peso en quilates (carat).

1. Carga y preparación de datos

   Utilizamos un dataset de Seaborn (diamonds).

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

   Mostramos los datos

```
df_diamonds = sns.load_dataset('diamonds')
df_diamonds.head()
```

	carat	cut	color	clarity	depth	table	price	x	y	z
0	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75

   Este data set **no tiene** valores nulos en las columnas. Sin embargo, en un flujo de trabajo real de Data Science debes tener cuidado con estos valores y decidir qué hacer con ellos, puedes rellenarlos o eliminarlos.

   Para visualizarlos puedes utilizar los métodos isna() y contabilizarlos con sum()

```
df_diamonds.isna().sum()
```

```
carat      0
cut        0
color      0
clarity    0
depth      0
table      0
price      0
x          0
y          0
z          0
dtype: int64
```

Dividimos los datos en un bloque para **entrenar** y otro para **testear**. La finalidad de hacer esto es evitar el *over-fitting*.

Nota: Se considera como buena práctica usar entre el 60% y 80% de los datos, mientras que para la parte de testing entre un 40% y 20%.

Para este ejemplo se utiliza una relación 70–30.

Obtenemos la variable dependiente e independiente. (X –> peso en quilates, Y –> precio)

```
x = df_diamonds['carat'].values.reshape(-1, 1)
y = df_diamonds['price'].values.reshape(-1, 1)
```

Usando Scikit-Learn dividimos los datos

```
X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size=0.3, random_state=0)
```

El método `train_test_split` recoge los datos de forma aleatoria. Entonces, el argumento `random_state` sirve para indicar si se trabaja con los datos aleatorios iniciales siempre o se escoge nuevos.
Valores que pueden asignarse:

- `random_state = 0.` Escoge los mismos datos.
- `random_state = None`. Itera y escoge nuevos datos cada vez.

2. Implementar el modelo de regresión lineal.

Una vez separados los datos de entrenamiento y testing se crea el modelo de machine learning.

```
regressor = LinearRegression()
regressor.fit(X_train, Y_train) #Creates model
```

Una vez que el modelo esta creado se pueden realizar predicciones con `regressor.predict()` y pasar como argumentos los valores que quisieras predecir.

```
y_pred = regressor.predict(X_train)
```

3. Evaluar el modelo

Para evaluar el desempeño del modelo en el conjunto de prueba se utiliza el *error cuadrático medio* y el *coeficiente de determinación*.

```
from sklearn.metrics import mean_squared_error, r2_score

print('Coefficients: \n', regressor.coef_)
print('Independent term: \n', regressor.intercept_)
print("Mean squared error: %.2f" % mean_squared_error(Y_train, y_pred))
```

```
Coefficients:
 [[7741.37904828]]
Independent term:
 [-2247.70447085]
Mean squared error: 2408781.08
```

Para cuantificar la precisión de los resultados se pueden utilizar cualquiera de estos dos métodos `regressor.score()` o bien `r2_score()`.

```
print(regressor.score(X_test, Y_test))
print('Variance score: %.2f' % r2_score(Y_train, y_pred))
```

```
0.8509771156910653
Variance score: 0.85
```

4. Graficar los resultados

```
sns.set_theme()
fig, ax = plt.subplots(1, 2, figsize=(10, 10), sharey=True)
ax[0].scatter(X_train, Y_train)
ax[0].plot(X_train, regressor.predict(X_train), c='g')
ax[0].set_title('Carat vs price (training)')
ax[0].set_xlabel('Carat')
ax[0].set_ylabel('Price, USD')
ax[1].scatter(X_test, Y_test)
ax[1].plot(X_train, regressor.predict(X_train), c='g')
ax[1].set_title('Carat vs price (testing)')
ax[1].set_xlabel('Carat')
ax[1].set_ylabel('Price, USD')
plt.suptitle('Linear Regression Model')
```

```
Text(0.5, 0.98, 'Linear Regression Model')
```

Para ingresar otra variable independiente (z) que representará la profundidad en milímetro de un diamante se puede utilizar el siguiente código.

```
xx = df_diamonds[['carat', 'z']].values
yy = df_diamonds['price'].values.reshape(-1, 1)
XX_train, XX_test, YY_train, YY_test = train_test_split(xx, yy, test_size=0.3, random_state=0)
regressor.fit(XX_train, YY_train)
regressor.score(XX_test, YY_test)
```

```
0.8552958757573261
```

5. Escribir una conclusión

El procedimiento que usamos fue, básicamente:

- Separar las variables dependiente e independiente.
- Separar los datos en set de entrenamiento y set de prueba.
- Entrenar el modelo con el set de entrenamiento.
- Validar el modelo con el set de prueba y calcular el score.

Se recomienda tener en cuenta siempre la documentación de Scikit-Learn.

Ejercicio. Diabetes

Se desea determinar la progresión de la enfermedad mediante un modelo lineal, con base en su IMC.

1. Carga y preparación de datos
   Utilizamos el set de datos de la librería de Scikit learn “diabetes”

```
from sklearn.datasets import load_diabetes
from sklearn.model_selection import train_test_split
```

2. Implementar el modelo de regresión lineal.

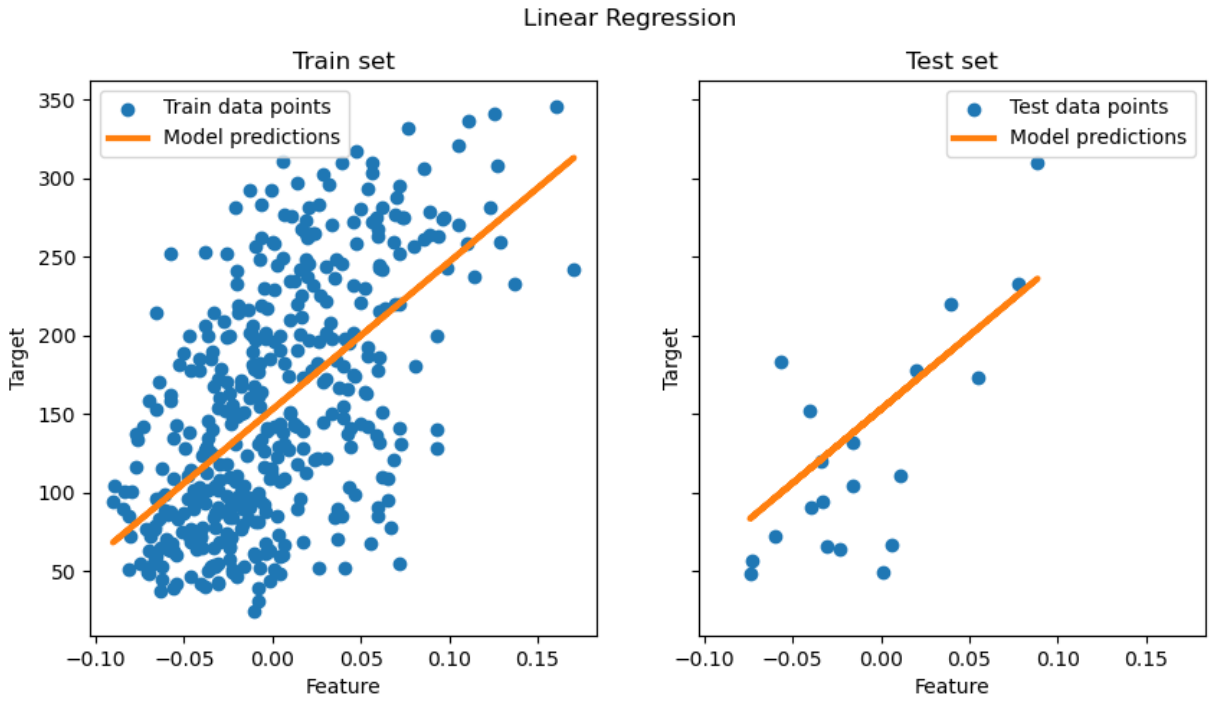
3. Evaluar el modelo

   Resultado esperado

```
Mean squared error: 2548.07
Coefficient of determination: 0.47
```

4. Graficar los resultados

   Resultado esperado



5. Escribir una conclusión