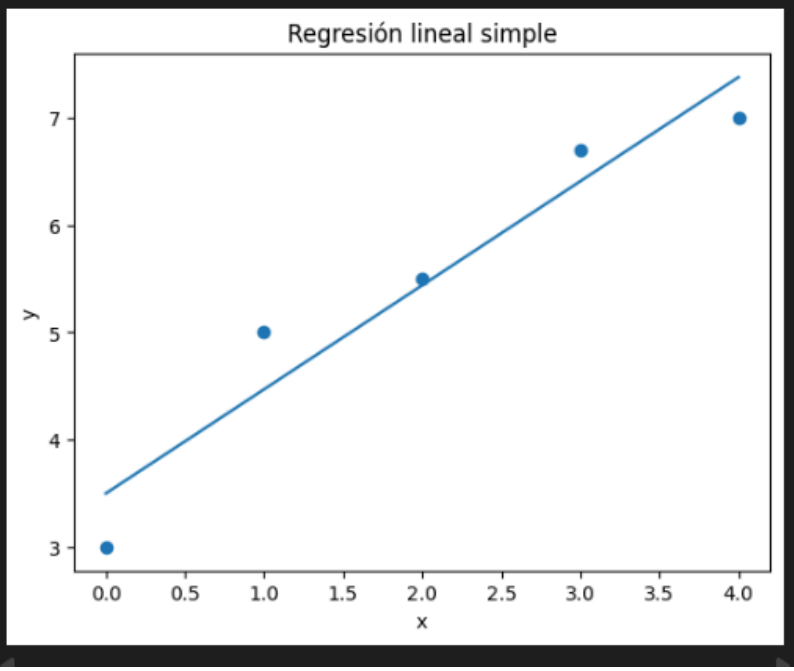
Guillermo García Hernández
2ºDAW 25/26
IA

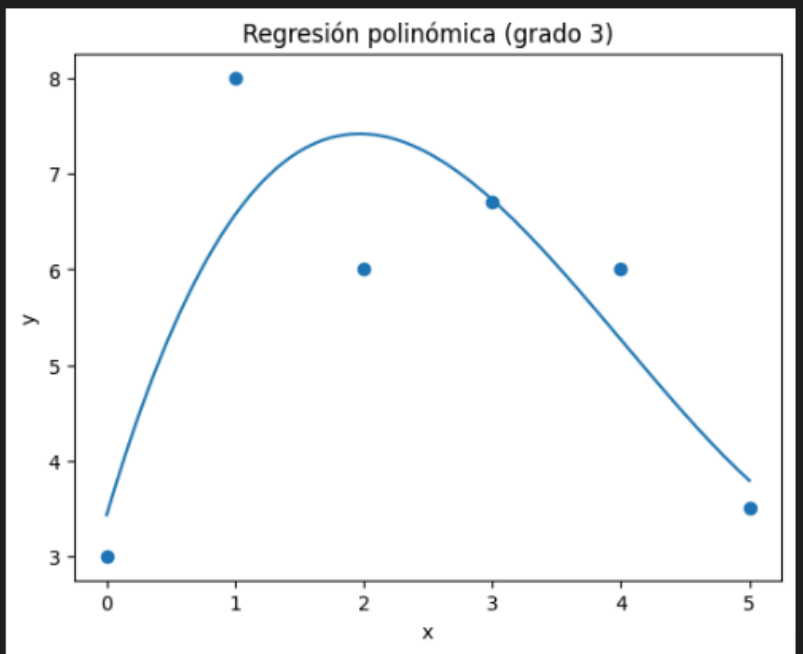
ML Regresiones

1 Lineal vs Polinómica

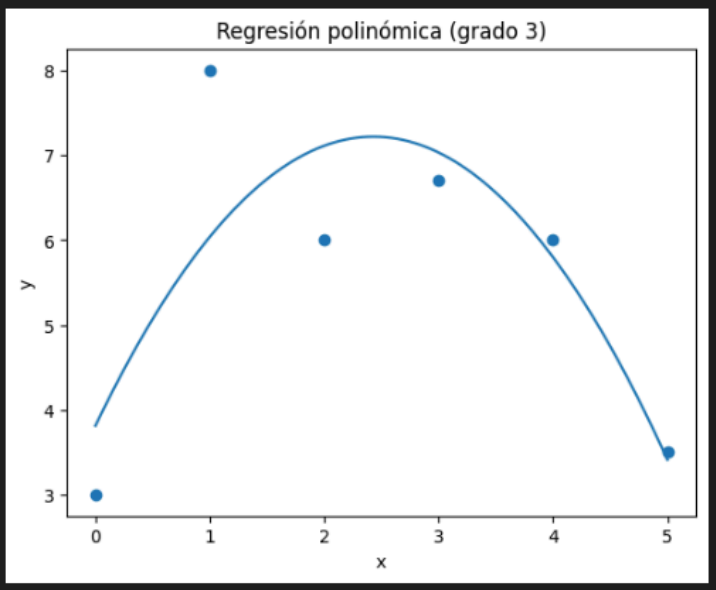
```
Pendiente: 0.97
Intercepto: 3.5
MSE: 0.1526
MAE: 0.352
R²: 0.925
```



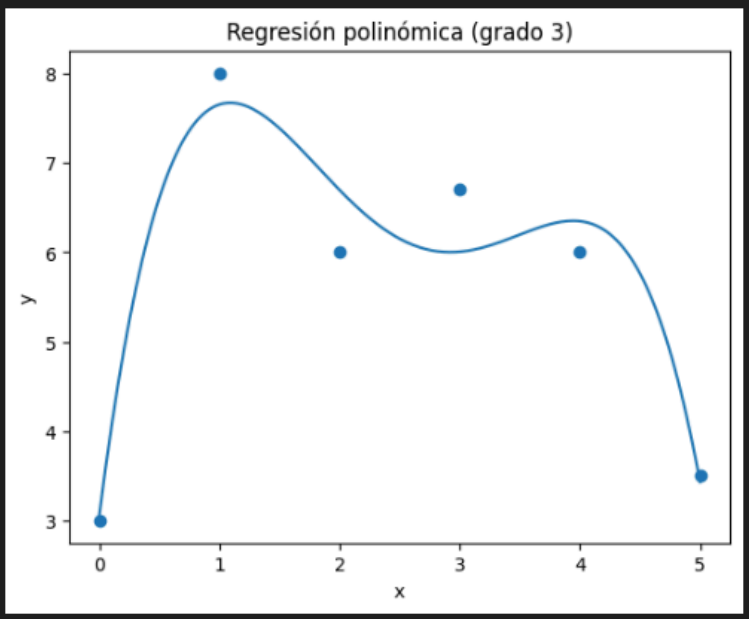

```
R² (grado 3): 0.7364
MSE (grado 3): 0.8097
Guillermo García Hernández
```



```
R² (grado 2): 0.6799
MSE (grado 2): 0.9835
Guillermo García Hernández
```




```
R² (grado 4): 0.9341
MSE (grado 4): 0.2025
Guillermo García Hernández
```



El sobreajuste aparece con el polinomio de grado 4.
Se nota porque el R^2 aumenta mucho (0.93) y el MSE baja, pero la curva se adapta demasiado a los datos.
En cambio, el modelo lineal o el de grado 3 son más equilibrados y representarían mejor una tendencia general.

2 Importancia de variables

```
MSE: 2848.3107
MAE: 41.5485
R²: 0.4849
```

	Variable	Coeficiente
4	s1	-918.502905
1	sex	-241.990907
9	s6	26.324582
0	age	47.749681
6	s3	116.950164
7	s4	269.492303
3	bp	381.562862
5	s2	508.257783
2	bmi	531.971063
8	s5	695.808117

El coeficiente indica cuánto influye cada variable en el resultado.
La variable s5 y bmi son las más importantes.
La variable s1 tiene un coeficiente negativo muy grande, lo que sugiere que cuando aumenta ese valor, la predicción disminuye.

3 Errores

```
Errores en train vs test:
MSE train: 2907.26
MSE test: 2848.31
MAE train: 44.05
MAE test: 41.55
Guillermo García Hernández
```

El error del train es un poco mayor, pero casi igual al del test.
Esto significa que el modelo no se ha sobreajustado y aprende bien en general.

4 Aplica StandardScaler en un Pipeline con la regresión polinómica

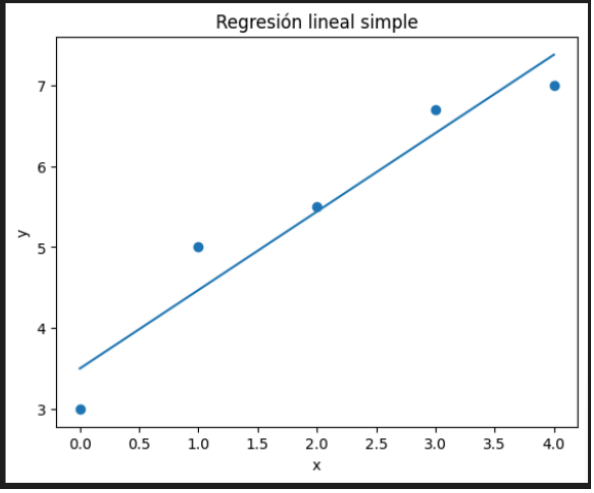
```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.preprocessing import StandardScaler

X_poly = np.array([0,1,2,3,4,5]).reshape(-1,1)
y_poly = np.array([3.0,8.0,6.0,6.7,6.0,3.5])

poly3 = make_pipeline(StandardScaler(), PolynomialFeatures(4), LinearRegression())
```

```
R² (grado 4): 0.9341
MSE (grado 4): 0.2025
Guillermo García Hernández
```

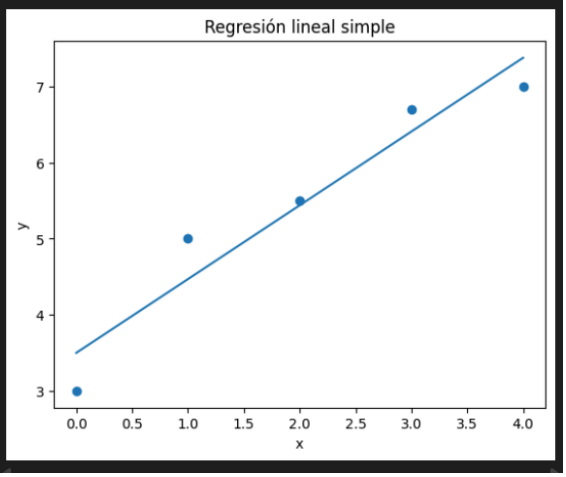
He añadido StandardScaler() al pipeline de la regresión polinómica antes de PolynomialFeatures(). Esto sirve para escalar los datos y mejorar la estabilidad numérica del modelo. En este caso, los resultados no cambian mucho.

5 Usa statsmodels y compara pendiente/intercepto con los de sklearn

```
Pendiente: 0.97
Intercepto: 3.5
MSE: 0.1526
MAE: 0.352
R²: 0.925
Skalearn
Guillermo García Hernández
```




```
Intercepto: 3.5000
Pendiente:  0.9700
statsmodels
Guillermo García Hernández
```



```python
import numpy as np, matplotlib.pyplo
import statsmodels.api as sm
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute

X = np.array([0,1,2,3,4]).reshape(-1,1)
y = np.array([3.0,5.0,5.5,6.7,7.0])
X_sm = sm.add_constant(X)
ols = sm.OLS(y, X_sm).fit()

lin = LinearRegression().fit(X, y)
y_pred = lin.predict(X)

print(f"Intercepto: {ols.params[0]:.4f}")
print(f"Pendiente:  {ols.params[1]:.4f}")
print("statsmodels")
print("Guillermo García Hernández")


plt.figure()
plt.scatter(X, y)
plt.plot(X, y_pred)
plt.title('Regresión lineal simple')
plt.xlabel('x'); plt.ylabel('y')
plt.show()
```

He repetido la regresión lineal simple usando statsmodels.
Los valores de pendiente e intercepto coinciden con los obtenidos con sklearn, ya que ambos usan mínimos cuadrados.