

```
In [1]: import numpy as np           # Librerías
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing, svm
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression # Paquete Regresión Linear
```

```
In [2]: # Modelo from scratch Regresión Lineal
```

```
In [3]: class LinearR() :  
  
    def __init__( self, learn_r, itera ) :  
  
        self.learn_r = learn_r  
  
        self.itera = itera  
        return None  
  
    def fit( self, X, Y ) :      # Model training  
  
        self.m, self.n = X.shape  
  
        self.W = np.zeros( self.n ) # Inicialización variables pesos  
                                     #  $y = w*x + b$   
        self.b = 0  
  
        self.X = X  
        self.Y = Y  
  
        for i in range( self.itera ) :    # Gradiente descendente  
  
            self.update_w()  
  
        return self  
  
    def update_w( self ) :      # Ajuste pesos: gradiente descendente  
  
        Y_pred = self.predict( self.X )  
  
        dW = - ( 2 * ( self.X.T ).dot( self.Y - Y_pred ) ) / self.m    # Cálculo  
        Lo gradiente  
        db = - 2 * np.sum( self.Y - Y_pred ) / self.m  
  
        self.W = self.W - self.learn_r * dW    # Ajuste pesos  
  
        self.b = self.b - self.learn_r * db  
  
        return self  
  
    def predict( self, X ) :      # Función H
```

```
return X.dot(self.W) + self.b
```

```
In [4]: df = pd.read_csv( "Salaries.csv" )    # Lectura datos

X = df.iloc[:, :-1].values

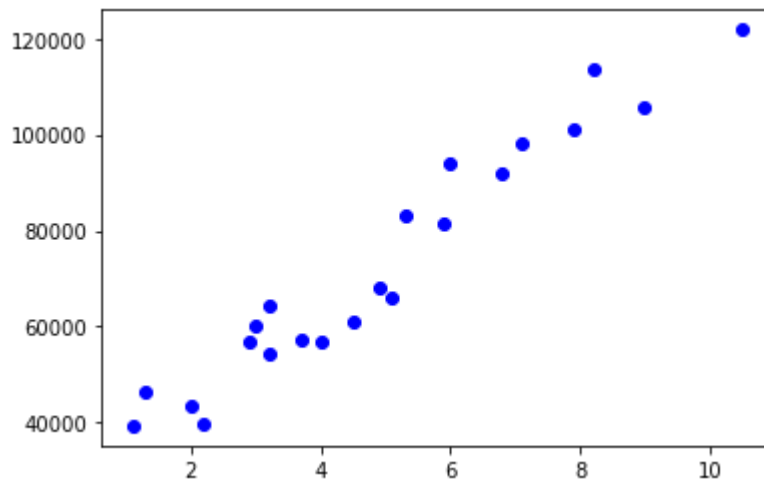
Y = df.iloc[:, 1].values

X_train, X_test, Y_train, Y_test = train_test_split(    # Train & test set
X, Y, test_size = 0.25, random_state = 0 )

plt.scatter( X_train, Y_train, color = 'blue' )

model = LinearR( learn_r = 0.01, itera = 1000 )    # Training
model.fit(X_train, Y_train)

Y_pred = model.predict(X_test)    # Predicción
```



```
In [5]: print( "Predicción ", np.round( Y_pred[:3], 2 ) )    # Resultados

print( "Reales ", Y_test[:3] )

print( "W ", round( model.W[0], 2 ) )

print( "b ", round( model.b, 2 ) )
```

```
Predicción [ 40726.6 123893.62 65298.68]
Reales [ 37731 122391 57081]
W 9450.8
b 26550.41
```

In [6]:

```
plt.scatter( X_test, Y_test, color = 'red' )    # Figuras de visualización
plt.plot( X_test, Y_pred, color = 'yellow' )
plt.title( 'Salario vs Experiencia' )
plt.xlabel( 'Experiencia(años)' )
plt.ylabel( 'Salario' )
plt.show()
```

In [7]: *# Modelo paquete Regresión Lineal*

```
In [8]: df2 = pd.read_csv('Salaries.csv')
df_bi = df2[['YearsExperience', 'Salary']]

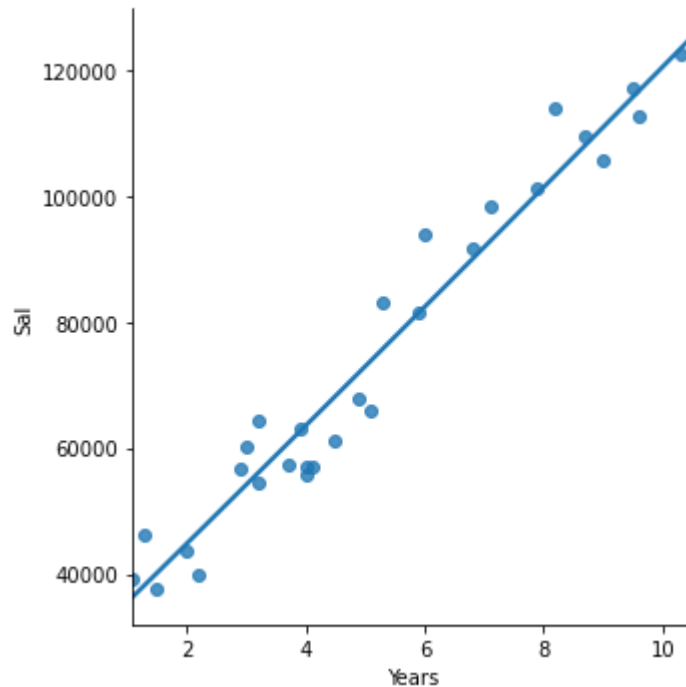
df_bi.columns = ['Years', 'Sal']
df_bi.head()
```

Out[8]:

	Years	Sal
0	1.1	39343
1	1.3	46205
2	1.5	37731
3	2.0	43525
4	2.2	39891

```
In [9]: sns.lmplot(x="Years", y="Sal", data = df_bi, order = 2, ci = None) # Visualización datos
```

```
Out[9]: <seaborn.axisgrid.FacetGrid at 0x2183d46cd90>
```



```
In [10]: X = np.array(df_bi['Years']).reshape(-1, 1)
y = np.array(df_bi['Sal']).reshape(-1, 1)

X_train2, X_test2, y_train2, y_test2 = train_test_split(X, y, test_size = 0.25)
# Train y Test separación

model2 = LinearRegression() # Llamamos a la Librería

model2.fit(X_train2, y_train2) # Train

print(model2.score(X_test2, y_test2))

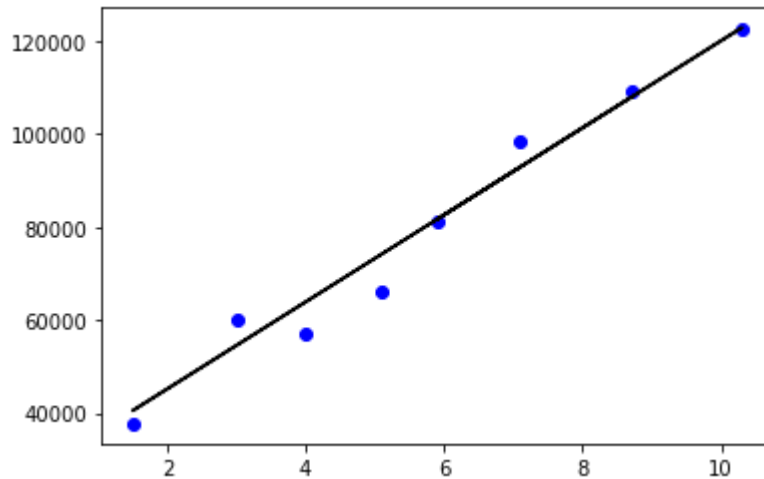
0.9685312453017993
```

```
In [11]: y_pred2 = model2.predict(X_test2)      # Predicción
print(model2.score(X_test2, y_test2)) # Score

plt.scatter(X_test2, y_test2, color='b')
plt.plot(X_test2, y_pred2, color='k')

plt.show() # Visualización LR
```

0.9685312453017993



```
In [12]: # Análisis y comparación modelos
```

```
In [13]: from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Modelo from scratch
mae = mean_absolute_error(y_true=Y_test, y_pred=Y_pred)
mse = mean_squared_error(y_true=Y_test, y_pred=Y_pred)
rmse = mean_squared_error(y_true=Y_test, y_pred=Y_pred, squared=False)
r2 = r2_score(y_true=Y_test, y_pred=Y_pred)

print("MAE:", mae)
print("MSE:", mse)
print("RMSE:", rmse)
print("R2:", r2)
```

MAE: 3425.468800564794
MSE: 21807636.097934574
RMSE: 4669.864676619075
R2: 0.9785123298086768

```
In [14]: # Modelo Librería
mae = mean_absolute_error(y_true=y_test2,y_pred=y_pred2)
mse = mean_squared_error(y_true=y_test2,y_pred=y_pred2)
rmse = mean_squared_error(y_true=y_test2,y_pred=y_pred2,squared=False)
r2 = r2_score(y_true=y_test2,y_pred=y_pred2)

print("MAE:",mae)
print("MSE:",mse)
print("RMSE:",rmse)
print("R2:",r2)
```

```
MAE: 3902.5781344374855
MSE: 23202059.514357246
RMSE: 4816.851618470019
R2: 0.9685312453017993
```

Al utilizar regresión lineal en ambos casos, nuestro análisis de errores resultó muy parecidos para los dos modelos. Con la visualización de los datos, al inicio del programa, se puede observar que se comportan de manera bastante lineal, Esto, aparte de suponer que nuestro modelo de regresión lineal será bien ajustado, nos indica que el grado de varianza debe ser bajo.

También, al ver los valores de MAE y RMSE, se puede concluir que, para el segundo modelo, probablemente se tenga menor varianza que para el primero, ya que el primero tiene un MAE menor, pero un RMSE casi igual al del segundo. La diferencia mayor entre estos valores indica una mayor varianza.

Al hablar del sesgo de nuestro modelo, resulta tener un nivel muy bajo. Esto es indicado por el R2, que nos dice que nuestro modelo tiene un buen ajuste. El bajo nivel del sesgo se debe al comportamiento lineal que presentan los datos, permitiendo un buen ajuste de nuestro modelo. El MAE que se presenta es bajo y nuestro R2 es alto, por lo que el sesgo es de un nivel bajo.

Al trabajar con una base de datos reducida y que presentó un comportamiento bastante lineal, el *fit* o ajuste de nuestro modelo no fue un problema. Aún así, se podrían presentar problemas de *underfitt* si se utiliza nuestro modelo con más datos de prueba, al haber sido entrenado con muy pocos datos.

Se utilizó *train-test-split*, con un 25% de testing para evitar el *underfitt*. Inicialmente, se utilizó un testing de 1/3 de los datos, pero se obtuvieron mejores resultados con el presentado finalmente.