Python y Ciencia de Datos (segunda parte)

Algoritmos supervisados

Regresión

Regresión Lineal Simple

Regresión significa que se predice un valor numérico, en lugar de una "clase" como se vió anteriormente. RLS se utiliza en Machine Learning y en estadística. Modela la relación entre una variable escalar dependiente “y” y una o más variables explicativas “X”. La idea es dibujar una recta que indicará la tendencia del conjunto de datos.

El objetivo de la regresión es producir un modelo que represente el "mejor ajuste" a algunos datos observados. Normalmente, el modelo es una función que describe algún tipo de curva, que está determinada por un conjunto de parámetros (por ejemplo, pendiente e intersección).

"Mejor ajuste" significa que hay un conjunto óptimo de parámetros de acuerdo con un criterio de evaluación que elegimos.

El modelo de regresión intenta predecir el valor de una variable, conocida como variable dependiente, variable de respuesta o etiqueta, utilizando los valores de otras variables, conocidas como variables independientes, variables explicativas o características.

La regresión simple tiene una etiqueta utilizada para predecir una característica. La regresión múltiple utiliza dos o más variables de entidad.

En forma matemática, el objetivo de la regresión es encontrar una función de algunas características X que predice el valor de la etiqueta y. Ésta función se puede escribir de la siguiente manera:

$$\hat{y} = f(X)$$

El desafío en la regresión es aprender la función $f(X)$ para que las predicciones de $\hat{y}$ sean precisas. En otras palabras, entrenamos el modelo para minimizar la diferencia entre nuestros pronósticos $\hat{y}$ y los valores de etiqueta conocidos y

El caso más simple de regresión lineal se conoce como regresión simple, ya que hay una sola característica. La función $f(X)$ es lineal en los coeficientes del modelo. Para un solo vector de características x

La ecuación de regresión lineal se escribe de la siguiente manera:

$$\hat{y} = a \cdot x + b$$

Los coeficientes del modelo son a, que denominamos como la pendiente, y b, que denominamos como la intercepción. Observe que esto es solo la ecuación de una línea recta para una variable.

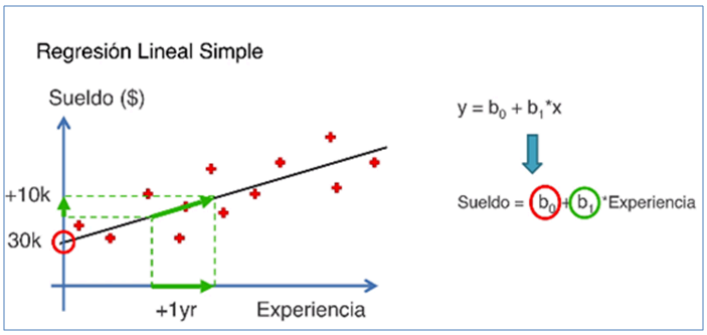
¿cuáles son los mejores valores de a y b? En regresión lineal, a y b se eligen para minimizar el error cuadrado entre las predicciones y las etiquetas conocidas. Esta cantidad se conoce como SSE (Sum Squared Error), o sea la suma de las diferencias al cuadrado entre cada observación. Para n casos, la SSE se calcula de la siguiente manera:

$$\begin{aligned} MSE &= \sum_{i=1}^{n} \left(f(x_i) - y_i \right)^2 \\ &= \sum_{i=1}^{n} \left(\hat{y}_i - y_i \right)^2 \\ &= \sum_{i=1}^{n} \left(a \cdot x_i + b - y_i \right)^2 \end{aligned}$$

Más información (https://es.wikipedia.org/wiki/Suma_residual_de_cuadrados)

El enfoque de regresión que minimiza la ESS se conoce como el **método de mínimos cuadrados**.

Ejemplo:



- b_0 sería el sueldo inicial, 30k, y
- b_1 es el incremento de sueldo que tengo en un año de experiencia, en el ejemplo es 10k

El algortimo de Regresión Lineal Simple utiliza el método de los mínimos cuadrados para hallar la mejor recta que se ajusta a los datos. Es decir, de todas las rectas se queda con aquella que minimiza los cuadrados de las diferencias entre el dato real y la predicción.

Más información (https://www.varsitytutors.com/hotmath/hotmath_help/spanish/topics/line-of-best-fit)

Importación de los datos

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

df= pd.read_csv('archs/Datos_de_salarios.csv')
df.head()
```

Out[1]:

	AniosExperiencia	Salario
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0

In [2]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   AniosExperiencia  30 non-null     float64
 1   Salario           30 non-null     float64
dtypes: float64(2)
memory usage: 608.0 bytes
```

In [3]:
```
df.describe()
```

Out[3]:

	AniosExperiencia	Salario
count	30.000000	30.000000
mean	5.313333	76003.000000
std	2.837888	27414.429785
min	1.100000	37731.000000
25%	3.200000	56720.750000
50%	4.700000	65237.000000
75%	7.700000	100544.750000
max	10.500000	122391.000000

In [4]:
```
X = df.iloc[:, :-1].values
y = df.iloc[:, 1].values
df.shape
```

Out[4]:
```
(30, 2)
```

In [5]:
```
X, y
```

Out[5]:
```
(array([[ 1.1],
        [ 1.3],
        [ 1.5],
        [ 2. ],
        [ 2.2],
        [ 2.9],
        [ 3. ],
        [ 3.2],
        [ 3.2],
        [ 3.7],
        [ 3.9],
        [ 4. ],
        [ 4. ],
        [ 4.1],
        [ 4.5],
        [ 4.9],
        [ 5.1],
        [ 5.3],
        [ 5.9],
        [ 6. ],
        [ 6.8],
        [ 7.1],
        [ 7.9],
        [ 8.2],
        [ 8.7],
        [ 9. ],
        [ 9.5],
        [ 9.6],
        [10.3],
        [10.5]]),
 array([ 39343.,  46205.,  37731.,  43525.,  39891.,  56642.,  60150.,
         54445.,  64445.,  57189.,  63218.,  55794.,  56957.,  57081.,
         61111.,  67938.,  66029.,  83088.,  81363.,  93940.,  91738.,
         98273., 101302., 113812., 109431., 105582., 116969., 112635.,
        122391., 121872.]))
```

In [6]:
```
1 df.describe()
```

Out[6]:

	AniosExperiencia	Salario
count	30.000000	30.000000
mean	5.313333	76003.000000
std	2.837888	27414.429785
min	1.100000	37731.000000
25%	3.200000	56720.750000
50%	4.700000	65237.000000
75%	7.700000	100544.750000
max	10.500000	122391.000000

Aquí tendremos como:

- **variable independiente (X)** o matriz de características, los años de experiencia por un lado, y
- el sueldo es la **variable dependiente (y)** aquella que el modelo de regresión lineal va a intentar predecir.

```
Dado el archivo detalle_autos2.csv

1. Cargar los datos.
2. Obtener información y descripción estadística.
3. Eliminar las columnas name,year, fuel,seller_type,transmission y owner
4. Detectar valores faltantes y outliers e imputar si es necesario.
```

Graficamos

In [7]:
```
1 df.plot(x='AniosExperiencia', y='Salario', style='o')
2 plt.title('Salario vs Años de Experiencia')
3 plt.ylabel('Salario')
4 plt.xlabel('Años de Experiencia')
5 plt.show()
```

Como se esperaba, los datos siguen una tendencia en línea recta. Sin embargo, hay cierta dispersión de estos datos como resultado del ruido distribuido normalmente.

```
Dado el archivo detalle_autos2.csv

1. Graficar.
2. Obtener conclusiones.
```

Dividir en train y test

In [8]:
```
1 from sklearn.model_selection import train_test_split
2 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 1/3, random_state = 0)
```

Comparar

In [9]:
```
df_aux = pd.DataFrame({'X_train': X_train.flatten(), 'y_train': y_train.flatten()})
df_aux.head()
```

Out[9]:

	X_train	y_train
0	2.9	56642.0
1	5.1	66029.0
2	3.2	64445.0
3	4.5	61111.0
4	8.2	113812.0

Se tienen los valores exactos de lo que ganan los individuos, las observaciones que formarán parte del proceso de entrenamiento. Con estos datos se creará el modelo de regresión lineal y luego se comprobará si el modelo se comporta correctamente o no, utilizando los elementos de testing.

X_test se suministrará al modelo que se ha creado con los elementos de entrenamiento y se observará si la predicción que elabora el modelo con estos datos de test se asemeja o no al vector y_test. Es la idea global del algoritmo de RLS.

In [10]:
```
df_aux = pd.DataFrame({'X_test': X_test.flatten(), 'y_test': y_test.flatten()})
df_aux
```

Out[10]:

	X_test	y_test
0	1.5	37731.0
1	10.3	122391.0
2	4.1	57081.0
3	3.9	63218.0
4	9.5	116969.0
5	8.7	109431.0
6	9.6	112635.0
7	4.0	55794.0
8	5.3	83088.0
9	7.9	101302.0

Dado el archivo detalle_autos2.csv

1. Dividir el dataset.
2. Obtener las comparaciones necesarias.

In [11]:
```
from sklearn.linear_model import LinearRegression
'''
Con LinearRegression creamos el regresor para el modelo
'''
regression = LinearRegression()
regression.fit(X_train, y_train)
```

Out[11]: LinearRegression()

Se brinda al modelo los datos de entrenamiento. Entonces, utilizando la técnica de los mínimos cuadrados, basado en la experiencia de aprendizaje, se creó un modelo y éste será capaz de predecir, con nuevos años de experiencia que tenga un trabajador en la empresa, cuál es el sueldo que le corresponde.

La función **predict()** se encarga de hacer las predicciones.

In [12]:
```
'''
Al aplicar .predict(), pasa el regresor como argumento y obtiene la respuesta predicha correspondiente.
'''

y_pred = regression.predict(X_test)
y_pred
```

Out[12]:
```
array([ 40835.10590871, 123079.39940819,  65134.55626083,  63265.36777221,
       115602.64545369, 108125.8914992 , 116537.23969801,  64199.96201652,
        76349.68719258, 100649.1375447 ])
```

intercept_ y coef_ son atributos de 'regression'. El siguiente código ilustra cómo obtener b_0 y b_1.

Puede notarse que .intercept_ es un escalar, mientras que .coef_ es un vector, intercept_ muestra el punto donde la línea de regresión estimada cruza el eje y. Denota que el modelo predice la respuesta 26816.19 cuando x = 0

In [13]:
```
print(f"Intercepción del modelo: {regression.intercept_}")
```

```
Intercepción del modelo: 26816.192244031183
```

Ejecutar la siguiente instrucción devuelve la pendiente de la recta. El valor b_1 = 9345.94 significa que la respuesta predice ese valor cuando x aumenta en 1.

In [14]:
```
print(f'Pendiente del modelo: {regression.coef_}')
```

```
Pendiente del modelo: [9345.94244312]
```

In [15]:
```
y_pred2 = regression.intercept_ + regression.coef_ * X_test
df_aux = pd.DataFrame({'Predicción': y_pred.flatten(), 'Predicción con intercept_ y coef_': y_pred2.flatten()})
df_aux
```

Out[15]:

	Predicción	Predicción con intercept_ y coef_
0	40835.105909	40835.105909
1	123079.399408	123079.399408
2	65134.556261	65134.556261
3	63265.367772	63265.367772
4	115602.645454	115602.645454
5	108125.891499	108125.891499
6	116537.239698	116537.239698
7	64199.962017	64199.962017
8	76349.687193	76349.687193
9	100649.137545	100649.137545

Ahora se pueden predecir las observaciones nuevas, que son las del conjunto de pruebas X_test que no ha sido utilizado en la creación del modelo lineal.

In [16]:
```
df_aux = pd.DataFrame({'Actual': y_test.flatten(), 'Predicción': y_pred.flatten()})
df_aux
```

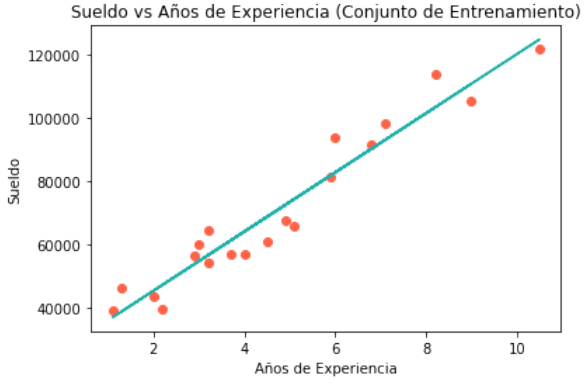
Out[16]:

	Actual	Predicción
0	37731.0	40835.105909
1	122391.0	123079.399408
2	57081.0	65134.556261
3	63218.0	63265.367772
4	116969.0	115602.645454
5	109431.0	108125.891499
6	112635.0	116537.239698
7	55794.0	64199.962017
8	83088.0	76349.687193
9	101302.0	100649.137545

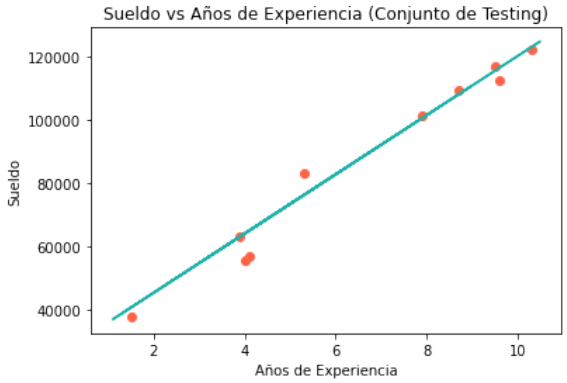
Dado el archivo detalle_autos2.csv

1. Crear el regresor para estos datos.
2. Aplicar predict() y emitir los datos.
3. Calcular y emitir la pendiente y la intercepción.
4. Comparar los datos actuales con la predicción.

In [17]:
```
plt.scatter(X_train, y_train, color = "#FF6347")
plt.plot(X_train, regression.predict(X_train), color = "#20B2AA")
plt.title("Sueldo vs Años de Experiencia (Conjunto de Entrenamiento)")
plt.xlabel("Años de Experiencia")
plt.ylabel("Sueldo")
plt.show()
```



In [18]:
```
plt.scatter(X_test, y_test, color = "#FF6347")
plt.plot(X_train, regression.predict(X_train), color = "#20B2AA")
plt.title("Sueldo vs Años de Experiencia (Conjunto de Testing)")
plt.xlabel("Años de Experiencia")
plt.ylabel("Sueldo")
plt.show()
```



Basado en este modelo lineal, predicción de sueldo por el puesto al que se aspira en base a la los años de experiencia

In [19]:
```
regression.predict([[6.5]])
```

Out[19]:
```
array([87564.81812433])
```

Dado el archivo detalle_autos2.csv

1. Crear el grafico con X_train, y_train
2. Crear el grafico con X_test, y_test
3. Predecir el precio de un vehículo en base a su kilometraje.

Evaluar el rendimiento del modelo

In [20]:
```
from sklearn import metrics
```

Con el modelo entrenado, es hora de evaluar el rendimiento. Esto se hace utilizando el conjunto de datos de prueba, de modo que no haya fuga de información del entrenamiento del modelo.

Como primer paso, se calcula un conjunto de métricas de rendimiento. Hay muchas métricas posibles utilizadas para la evaluación de modelos de regresión. Generalmente, estas métricas son funciones del valor residual, o diferencia entre el valor o puntuación prevista y el valor real de la etiqueta:

$$r_i = f(x_i) - y_i = \hat{y}_i - y_i$$

- Error cuadrático medio o MSE:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(f(x_i) - y_i)^2$$

El error cuadrático medio es idéntico a la varianza de los residuos (con un ligero sesgo), es la media de los errores al cuadrado.

Más información (https://es.wikipedia.org/wiki/Error_cuadr%C3%A1tico_medio)

In [21]:
```
print('Error cuadrático medio (MSE):', metrics.mean_squared_error(y_test, y_pred))
```

```
Error cuadrático medio (MSE): 21026037.329511296
```

- Raíz cuadrada del error cuadrático medio o RMSE,

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(f(x_i) - y_i)^2}$$

La raíz cuadrada del error cuadrático medio es idéntica a la desviación estándar de los residuos (con un ligero sesgo), está en las mismas unidades que los valores de la etiqueta.

Más información (https://es.wikipedia.org/wiki/Ra%C3%ADz_del_error_cuadr%C3%A1tico_medio)

In [22]:
```
print('Raíz cuadrada del error cuadrático medio (RMSE) :', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
Raíz cuadrada del error cuadrático medio (RMSE) : 4585.4157204675885
```

- Error medio absoluto o MAE,

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|f(x_i) - y_i|$$

Dónde || es el operador de valor absoluto. La interpretación similar a la raíz cuadrada del error cuadrático medio Puede encontrar esta medida más intuitiva ya que es simplemente el promedio de la magnitud de los residuos.

In [23]:
```
print('Error Medio Absoluto (MAE):', metrics.mean_absolute_error(y_test, y_pred))
```

```
Error Medio Absoluto (MAE): 3426.4269374307078
```

- Error absoluto mediano,

$$Median\ Absolute\ Error = Median\Big(\sum_{i=1}^{N}|f(x_i) - y_i|\Big)$$

El error absoluto mediano es una medida robusta del parámetro de ubicación de los residuos absolutos. Si esta medida es significativamente diferente del error absoluto medio, es probable que haya valores atípicos en los residuos.

Más información (https://es.wikipedia.org/wiki/Error_absoluto_medio)

In [24]:
```
print('Error absoluto mediano  = ', metrics.median_absolute_error(y_test, y_pred))
```

```
Error absoluto mediano  =  2235.2302275105103
```

- R squared or R^2 también conocido como coeficiente de determinación,

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

dónde,
$SS_{res} = \sum_{i=1}^{N} r_i^2$, es la suma de los residuos al cuadrado,
$SS_{tot} = \sum_{i=1}^{N} y_i^2$, es la suma de los valores de etiqueta al cuadrado.

En otras palabras, R^2 es la medida de la reducción en la suma de los valores al cuadrado entre los valores brutos de la etiqueta y los residuos. Si el modelo no ha reducido la suma de cuadrados de las etiquetas $R^2 = 0$. Por otro lado, si el modelo se ajusta perfectamente a los datos entonces $R^2 = 1$

In [25]:
```
print('R^2', metrics.r2_score(y_test, y_pred))
```

```
R^2 0.9749154407708353
```

- R cuadrado ajustado o R^2_{adj}, es R^2 ajustado por grados de libertad en el modelo,

$$R^2_{adj} = 1 - \frac{var(r)}{var(y)} = 1 - \frac{\frac{SS_{res}}{(n-p-1)}}{\frac{SS_{tot}}{(n-1)}}$$

dónde,
$var(r)$ = la varianza de los residuos, $var(y)$ = la variación de las etiquetas, n = número de muestras o casos, p = número de parámetros del modelo.

La interpretación de R^2_{adj} es lo mismo que R^2. En muchos casos habrá poca diferencia. Sin embargo, si el número de parámetros es significativo con respecto al número de casos, R^2 dará una medida demasiado optimista del rendimiento del modelo. En general, la diferencia entre R^2_{adj} y R^2 se vuelve menos significativo a medida que el número de casos n crece. Sin embargo, incluso para los modelos de 'big data' puede haber una diferencia significativa si hay un gran número de parámetros del modelo.

In [26]:
```
r2 = metrics.r2_score(y_test, y_pred)
r2_adj = r2 - (y_test.shape[0] - 1)/(y_test.shape[0] - 2 - 1) * (1 - r2)
print('R^2 ajustado', r2_adj)
```

```
R^2 ajustado 0.942663864619052
```

El R cuadrado y el R cuadrado ajustado, el valor más grande que pueden tomar es 1.

```
Dado el archivo detalle_autos2.csv

1. Obtener el Error cuadrático medio o MSE
2. Obtener la Raíz cuadrada del error cuadrático medio (RMSE)
3. Obtener el Error Medio Absoluto (MAE)
4. Obtener el Error absoluto mediano
5. Obtener el R^2
6. Obtener el R^2 ajustado
```