**2. Linear Regression**

<https://www.youtube.com/watch?v=8jazNUpO3lQ>

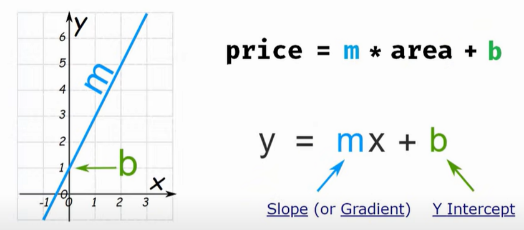
¿Por qué LINEAR REGRESION? Porque un se desea línea recta de tendencia, pero se debe probar varias posibles líneas rectas, y al final se escoge la que genera MENOS suma de errores, de modo que el algoritmo prueba varias posibilidades hasta que nos da la mejor línea recta.

<https://github.com/codebasics/py/blob/master/ML/1_linear_reg/1_linear_regression.ipynb>

<https://github.com/codebasics/py/tree/master/ML/1_linear_reg>

<https://github.com/codebasics/py>

|  |  |
| --- | --- |
|  | area price  2600 550000  3000 565000  3200 610000  3600 680000  4000 725000  We try different lines  We choose the line with less errors |



import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn import linear\_model

df = pd.read\_csv("homeprices.csv")

|  |  |
| --- | --- |
| **#EXPLORAR SI ES UNA REGRESION LINEAL**  %matplotlib inline  plt.xlabel('area')  plt.ylabel('price')  plt.scatter(df.area, df.price, color='red', marker='+') |  |

x = df[['area']]

y = df.price

reg = linear\_model.LinearRegression()

reg.fit(x,y)

reg.predic(3300)

output

array([628715.7534])

**CALCULE PENDIENTE E INTERCEPTO**

m = reg.coef\_

b = reg.intercept\_

# y = m(x) + b

m(3300) + b # 628715.7534

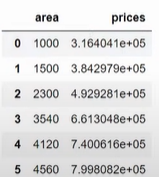
El algoritmo de ML probo varias líneas rectas, y escogió la pendiente e intercepto con el mínimo error

**TIENE UNA LISTA DE AREAS, ASIGNE SU PRECIO**

d= pd.read\_csv('areas.csv')

P = reg.predict(d)

d['price'] = p



#DOWNLOAD

df.to\_csv(‘prediccion.csv’)

%matplotlib inline

plt.xlabel('area', fontsize=20)

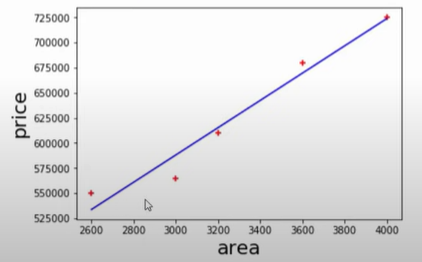
plt.ylabel('price', fontsize=20)

#puntos como '+'

plt.scatter(df.area, df.price, color='red', marker='+')

#linea azul

plt.scatter(df.area, reg.predict(df[['area']]), color='blue')



print(‘pendiente ’ + m)

print(‘intercepto ‘+b)

ACCURACY

*# Accuracy for the model calculate MAE, MSE, RMSE*

print(reg**.**mean\_absolute\_error(y\_true, y\_pred))

print(reg**.**mean\_squared\_error(y\_true, y\_pred))

print(np**.**sqrt(reg**.**mean\_squared\_error(y\_true, y\_pred)))

10.0

150.0

12.2474487139

MSE is more popular than MAE because MSE "punishes" larger errors. But, RMSE is even more popular than MSE because RMSE is interpretable in the "y" units.

model**.**score(X,y)

Out[21]:

0.9573929037221873