UNIVERSIDAD SANTO TOMÁS

PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA

T U N J A

VIGILADA MINEDUCACIÓN - SNIES 1732

ACREDITACIÓN INSTITUCIONAL DE ALTA CALIDAD MULTICAMPUS

Res. MEN No. 01456 del 29 de enero de 2016

Vigencia por seis años

200 BICENTENARIO DE INDEPENDENCIA 1819 - 2019

Formando personas que transforman

Bicentenario de la Independencia Nacional

UNIVERSIDAD SANTO TOMÁS
PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA
T U N J A
VIGILADA MINEDUCACIÓN - SNIES 1732

Faculty: **S**ystems engineer
Course: Deep Learning
Topic: Algoritmos de machine learning

Professor: Luis Fernando Castellanos Guarin
Email: Luis.castellanosg@usantoto.edu.co
Phone: 321-4582098

Formando personas que transforman

Bicentenario de la Independencia Nacional

UNIVERSIDAD SANTO TOMÁS
PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA

BICENTENARIO DE INDEPENDENCIA 1810 - 2010

¡Antes de continuar debemos analizar!

P1Tx_caricatura (Taller fuera de clase):

Leer los capítulos 1, 2, 3 y 8 del documento "**Como crear una mente**" de Ray Kurzweil, y dibujar una caricatura, donde puedan sintetizar lo leído.

El libro se encuentra en el campus virtual en la semana 1 sección "base documental"

UNIVERSIDAD SANTO TOMÁS
PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA
T U N J A
VIGILADA MINEDUCACIÓN - SNIES 1732

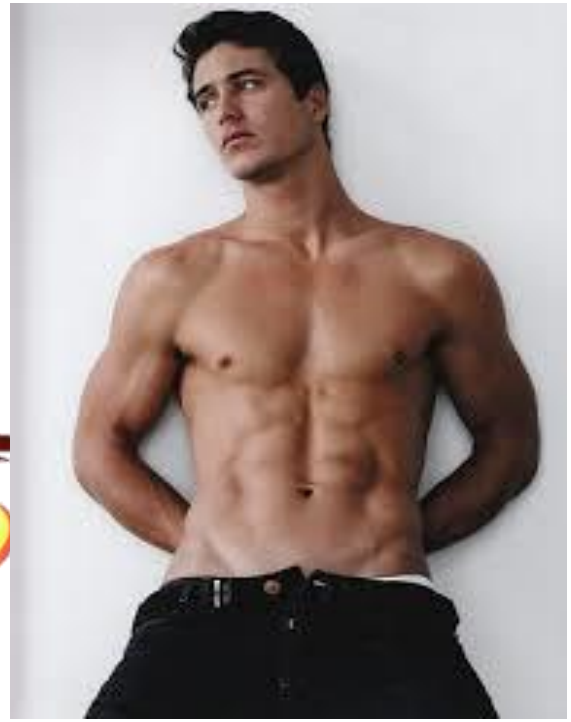
BICENTENARIO DE INDEPENDENCIA 1819 - 2019

Modelos para entender las realidades caóticas de nuestro universo

Formando personas que transforman

Bicentenario de la Independencia Nacional

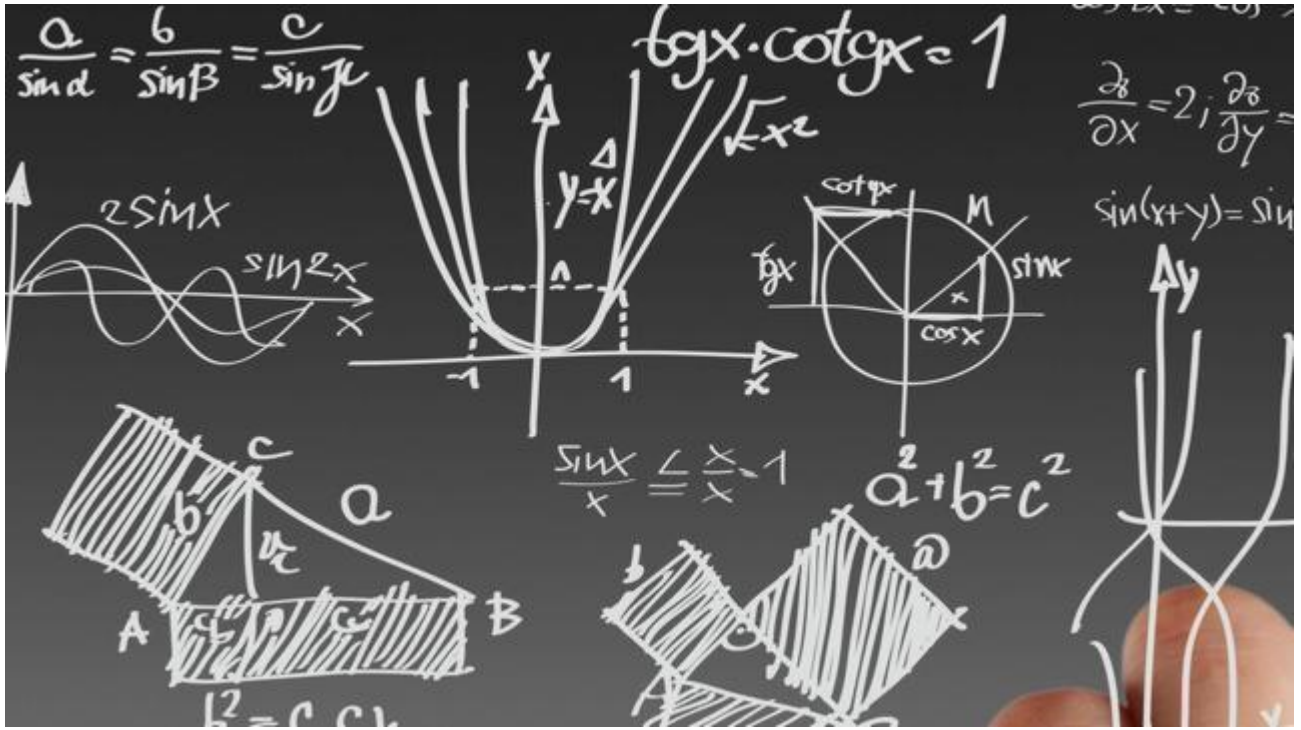
Con que tipo de modelos empezamos?



Ups… hablamos de modelos matemáticos

Lo siento NO hablaremos de estos modelos

Modelos matemáticos

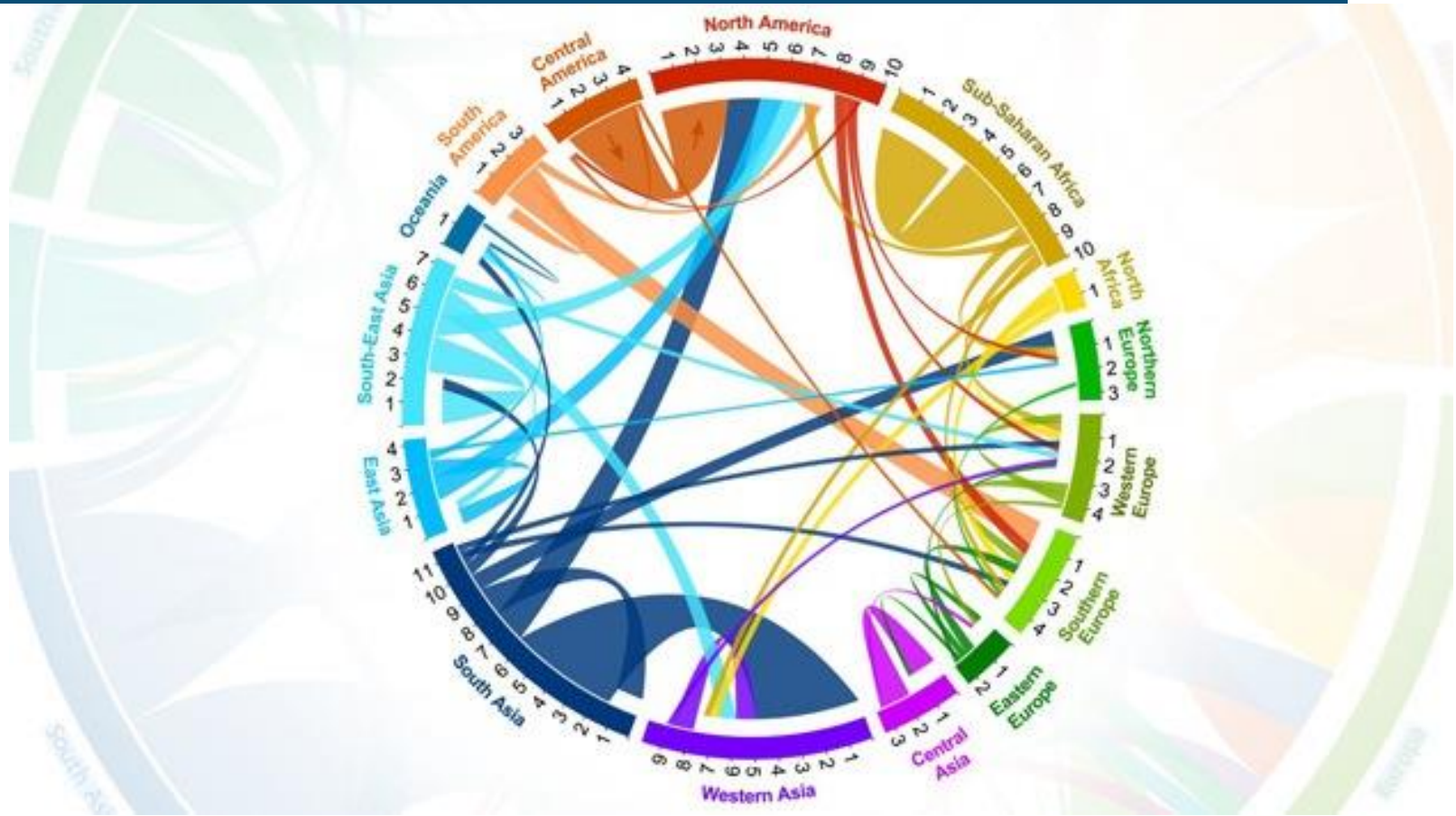


La inteligencia humana logra identificar patrones en medio del caos generado por un universo que esta en constante evolución, complejo, caótico y con mucho ruido.

Modelos matemáticos->patrones

Los humanos hemos logrado con nuestra propia evolución encontrar simetría y elegancia entre los patrones de nuestra realidad y usarlos para nuestro propio beneficio

Pero que es un modelo?

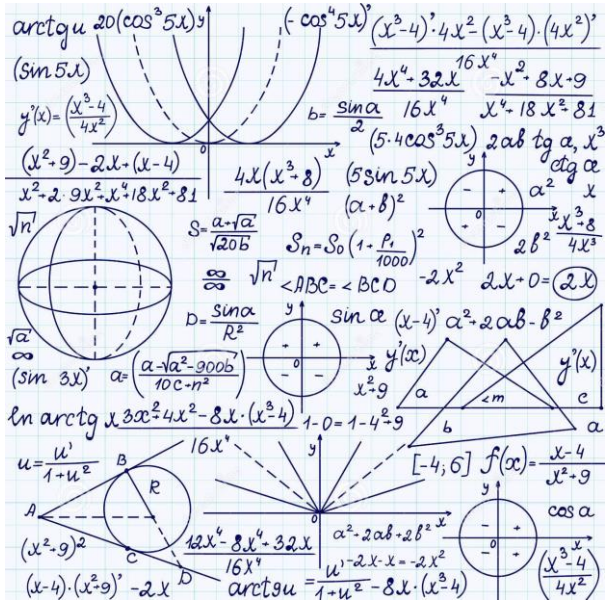


Un modelo es una construcción conceptual simplificada de una realidad mas compleja, y con ello entender mejor dicha realidad.

Ejemplos de modelos

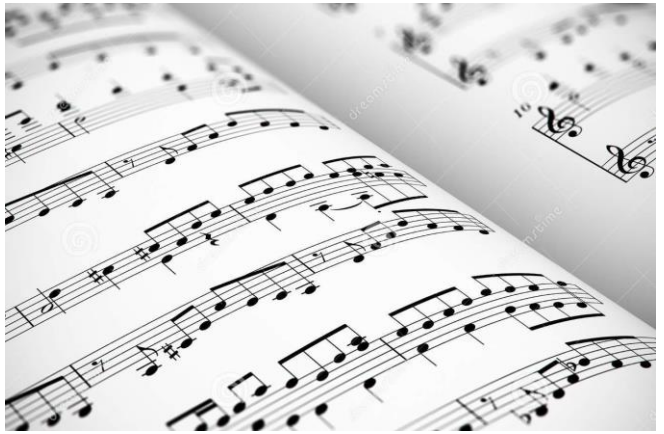
Mapas: representa una realidad tridimensional en un plano bidimensional

Ecuación Matemática/física : Representa el comportamiento de variables que explican la realidad ejemplo **E: mc²**

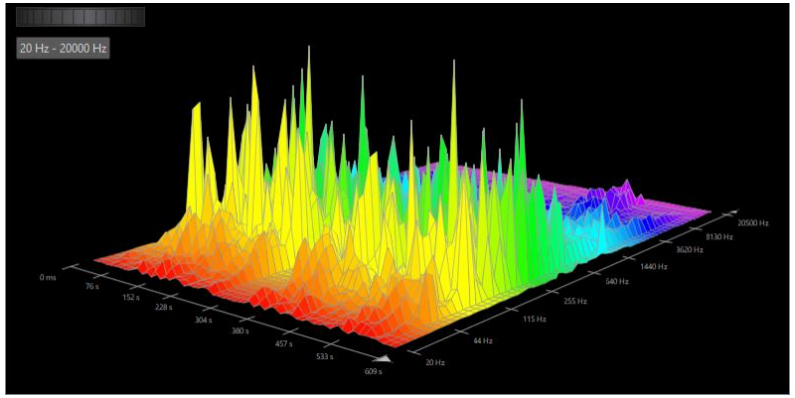


Ejemplos de modelos

Partituras musicales: representación de como los instrumentos musicales deben sincronizarse para siempre escucharse igual



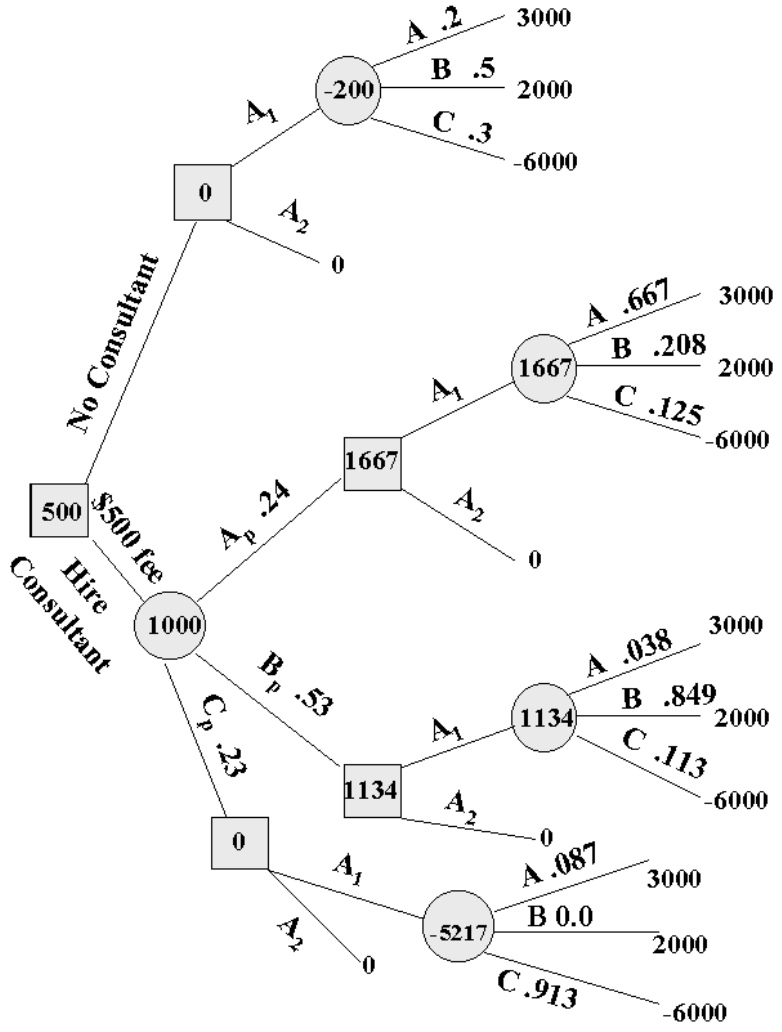
Espectros : Representación de frecuencias ya sea de sonidos, temperaturas, comportamientos de plantas, animales u otros aspectos rutinarios de nuestra realidad.



modelos-> probabilísticos

Usando la probabilidad (INFERENCIA): Permite resumir la incertidumbre sobre un tema ya sea por "*pereza*" o "*falta de conocimiento*" (0 a 100% de ocurrencia de un suceso).

Los modelos probabilísticos logran comprimir en base a la probabilidad muchas de las variables de una realidad haciendo fácil la gestión

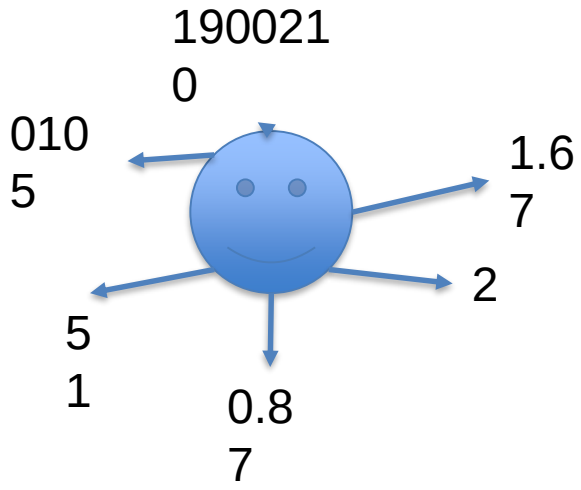


modelos-> Multidimencionales

En el universo conocido los objetos cuentan casi siempre con mas de una característica, ejemplo una persona, tiene:

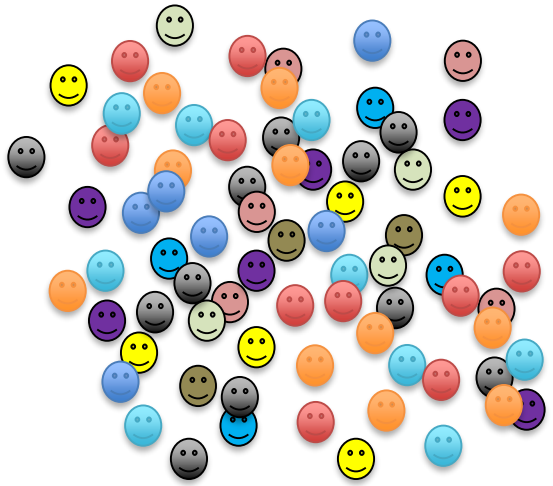
1. Fecha de nacimiento,
2. Estatura
3. Peso,
4. lugar de nacimiento,
5. Tipo de sangre
6. Porcentaje de glóbulos rojos en la sangre

y si cada una de sus características se considera una dimensión una sola persona seria un **punto multimendional**



Pero si estamos analizando la información de **100** personas con las mismas 6 características cada una , tendríamos un modelo muy complejo de poder imaginar en graficas de 2d y 3d que son las que el cerebro humano puede procesar

(**por eso necesitamos de las matemáticas**)



When machines have ideas

P1Tx_ensayo_When_machines_have_ideas (Taller fuera de clase):

Ver la presentación de **Ben Vigoda** sobre "**When machines have ideas**" y escribir un ensayo con las siguientes características:

- Tamaña Carta (con bordes: superior 3, izquierda 3, derecha 2, inferior 2)
- Una hoja, máximo 2
- Letra estilo Arial o Calibri y tamaño 12
- Interlineado: 1.5
- Alienación del texto: ajustado.

La presentación esta en esta URL: https://www.youtube.com/watch?v=PCs3vsoMZfY (obviamente esta en ingles).
Por si no saben quien es ese tal Ben Vigoda: https://tedxboston.org/speaker/vigoda

UNIVERSIDAD SANTO TOMÁS
PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA
T U N J A
VIGILADA MINEDUCACIÓN - SNIES 1732

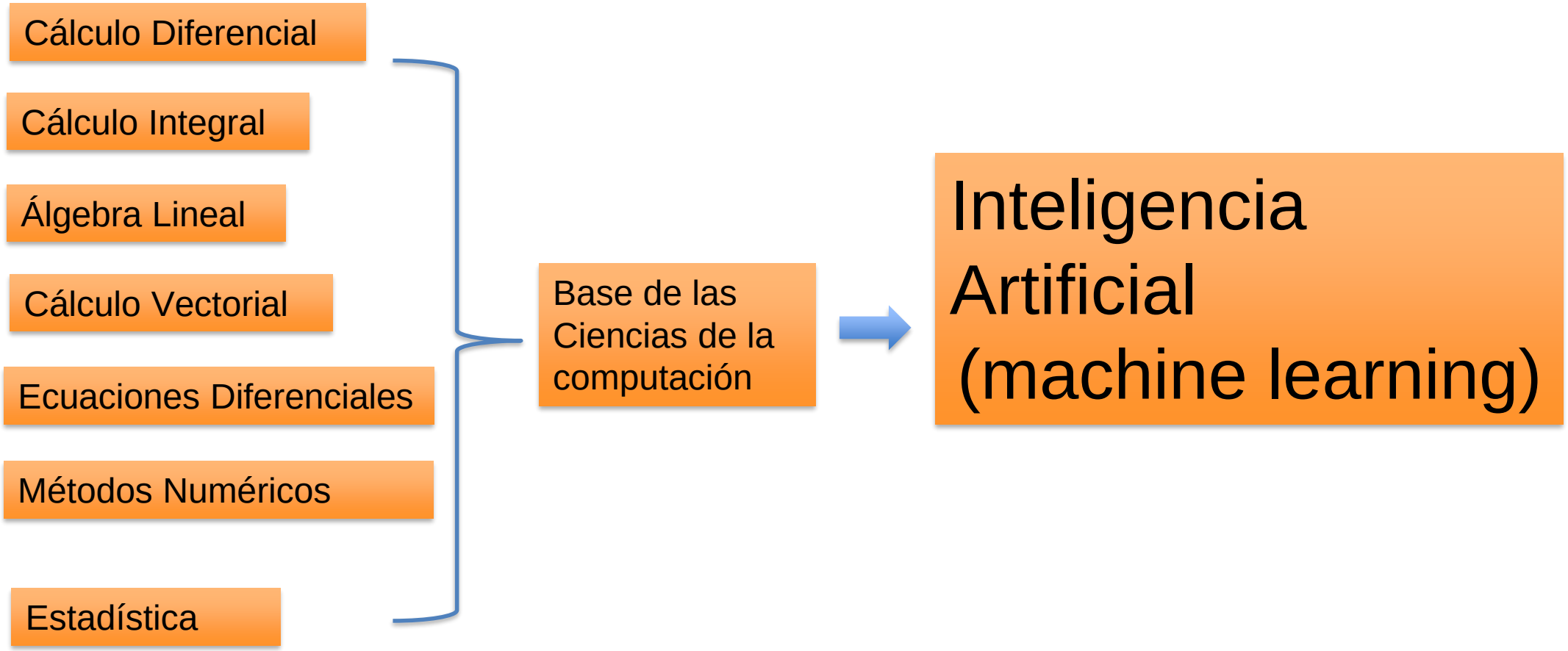
200 BICENTENARIO DE INDEPENDENCIA 1819 - 2019

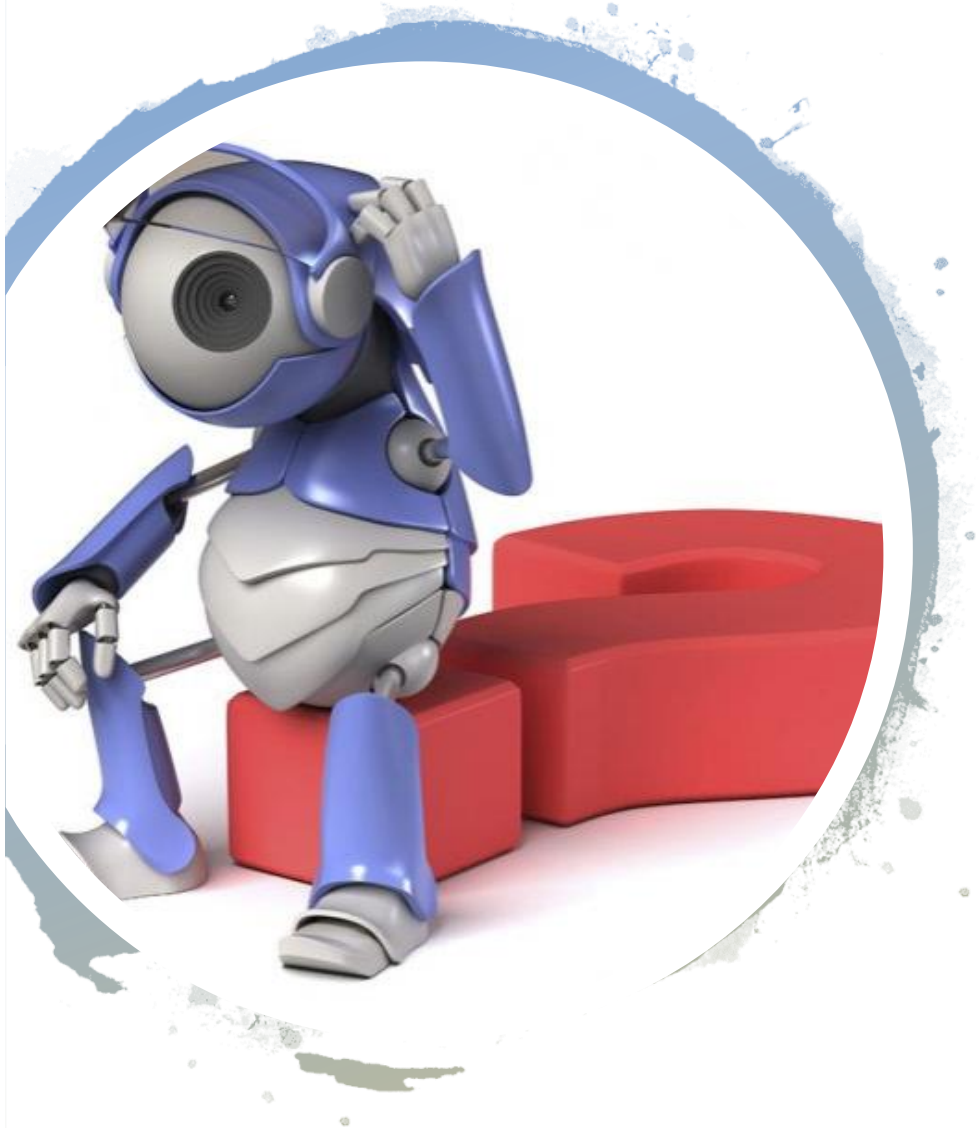
Machine learning y las matemáticas

Formando personas que transforman

Bicentenario de la Independencia Nacional

Machine learning y las matemáticas





Machine learning y las matemáticas

La gran tarea en Machine Learning es encontrar algoritmos que sean capaces de **aprender** sobre valores óptimos a partir de los datos con un porcentaje mínimo de **error**.

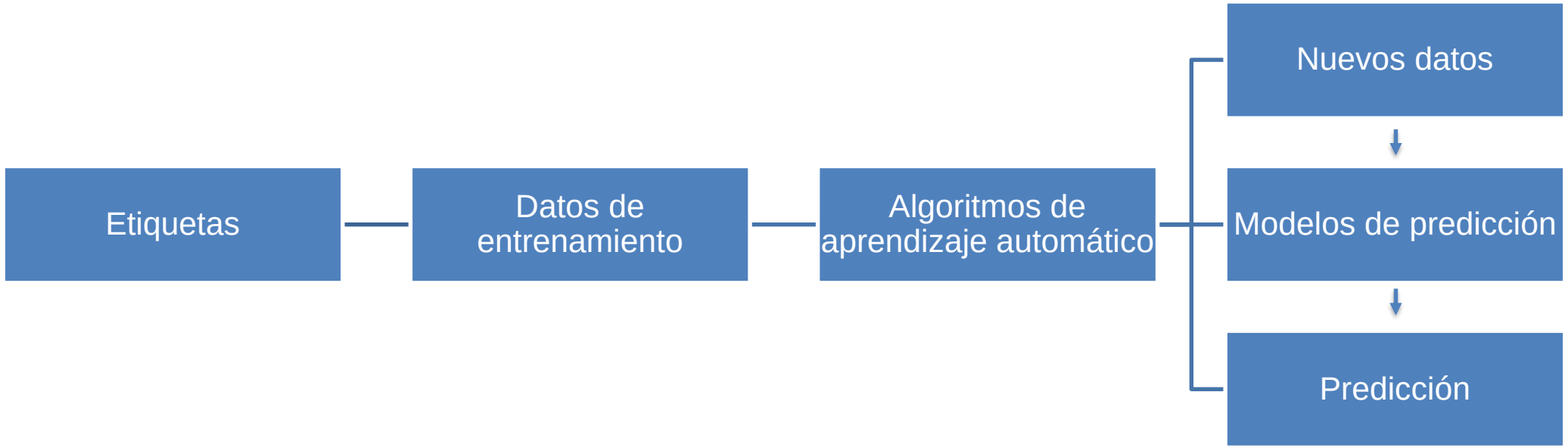
Aquello que no se mide, no se puede mejorar

¿Como enseñar a una maquina a hacer una tarea?

TIPOS DE APRENDIZAJE AUTOMÁTICO

Aprendizaje supervisado	• Datos etiquetados • Feedback directo • Predicción de resultados/futuro
Aprendizaje NO supervisado	• Sin etiquetas • Sin Feedback • Encontrar estructuras ocultas en los datos
Aprendizaje reforzado	• Proceso de decisión • Sistema de recompensa • Aprender series de acciones

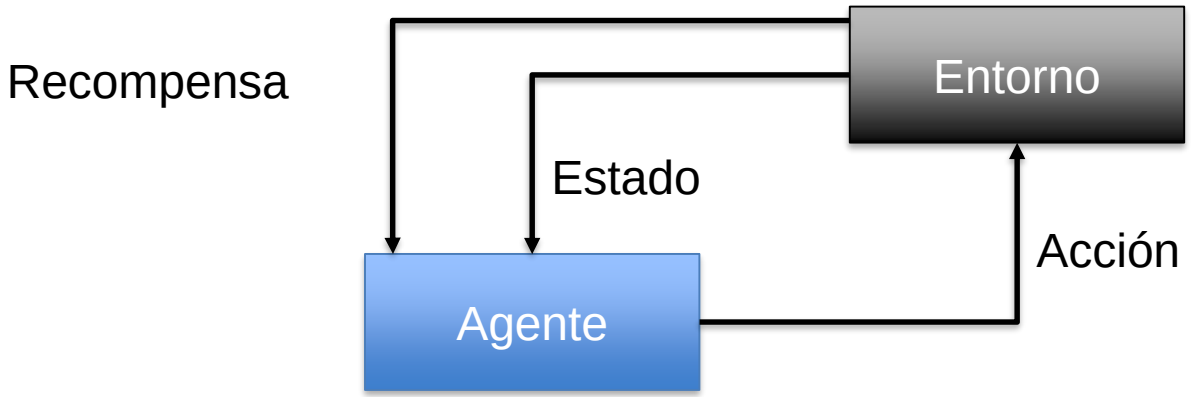
Predicciones con aprendizaje supervisado



El objetivo principal del aprendizaje supervisado es aprender un modelo, a partir de datos de entrenamiento etiquetados, que nos permite hacer predicciones sobre datos futuros o no vistos.

Predicciones con aprendizaje reforzado

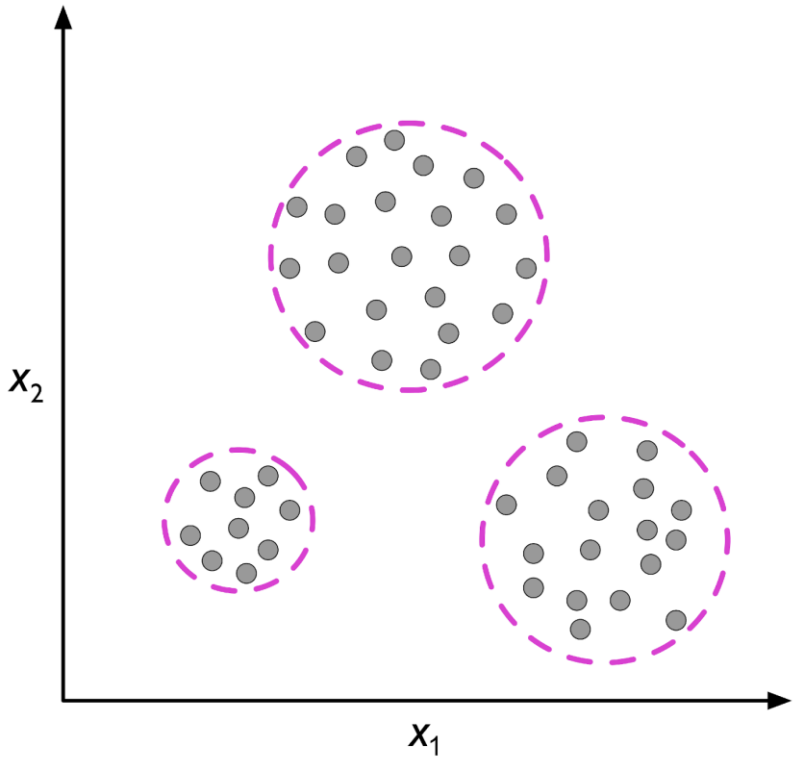
El objetivo es desarrollar un sistema (agente) que mejore su rendimiento basado en interacciones con el entorno. Como la información sobre el estado actual del entorno normalmente también incluye una señal de recompensa.



Ejemplo: Un motor de ajedrez. El “agente” elige entre una serie de movimientos según el estado del tablero (el entorno), y la recompensa se puede definir como “ganas” o “pierdes” al final del juego.

APRENDIZAJE SIN SUPERVISIÓN : Descubrir estructuras ocultas

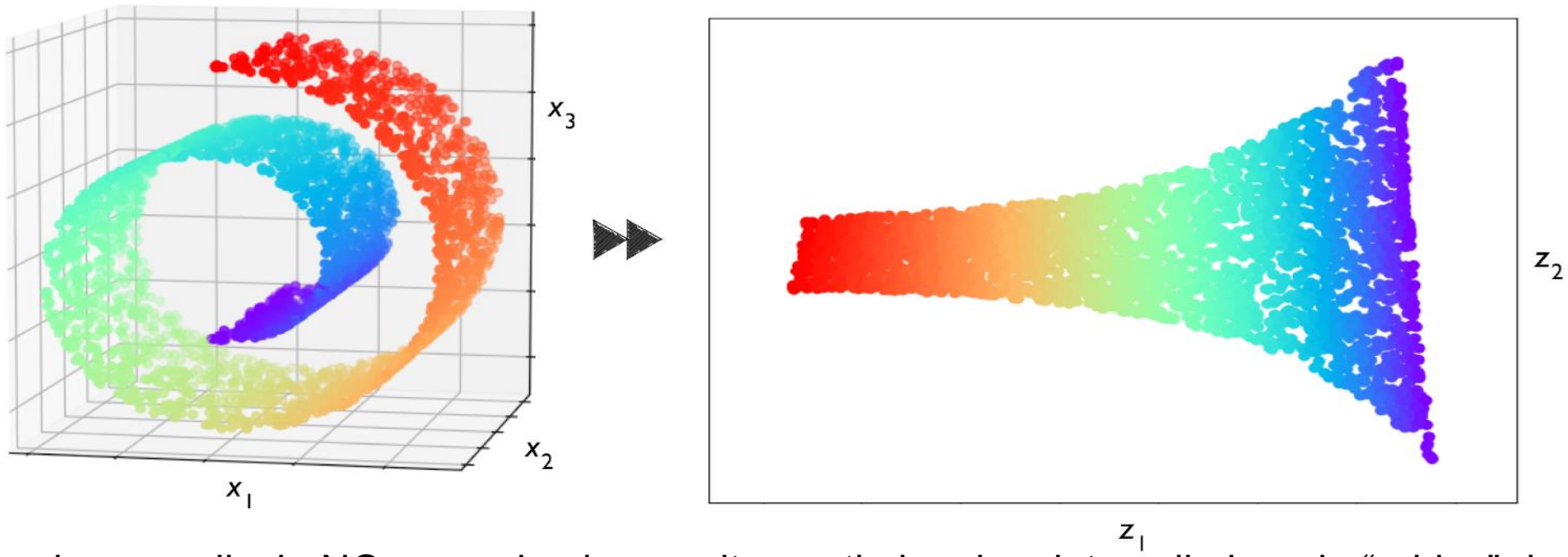
El objetivo es desarrollar un sistema (agente) que mejore su rendimiento basado en interacciones con el entorno. Como la información sobre el estado actual del entorno normalmente también incluye una señal de recompensa.



El agrupamiento es una técnica exploratoria de análisis de datos que nos permite organizar un montón de información en subgrupos significativos denominados **Clústers**

APRENDIZAJE SIN SUPERVISIÓN : Reducción de dimensionalidad

En el campo de la “computación científica” que requiere analizar una gran cantidad de datos lo que podría suponer un reto para el almacenamiento de la información al igual el rendimiento de los algoritmos típicos (SQL, ETL, Machine Learning).



Utilizando técnicas de aprendizaje NO supervisado permiten optimizar los datos eliminando “ruidos” de los datos

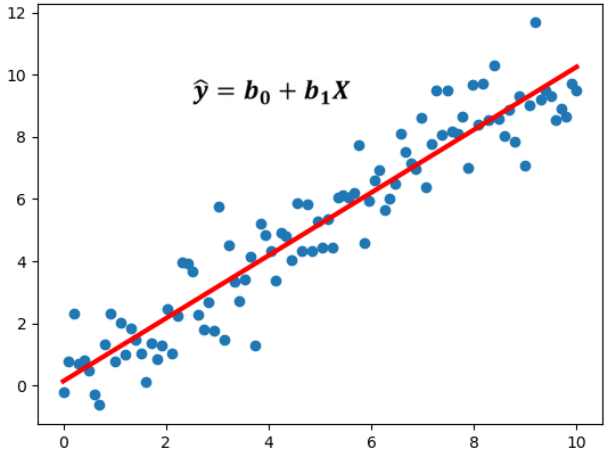
Machine learning y las matemáticas

Para entender algunos conceptos fundamentales es necesario que revisemos los siguientes videos hechos por un fanático de la I.A:

- Que es un modelo: https://www.youtube.com/watch?v=Sb8XVheowVQ
- Regresión lineal: https://www.youtube.com/watch?v=k964_uNn3l0
- Descenso del gradiente: https://www.youtube.com/watch?v=A6FiCDoz8_4

Regresión lineal

Es una técnica estadística utilizada para estudiar la relación entre variables(dos o mas).

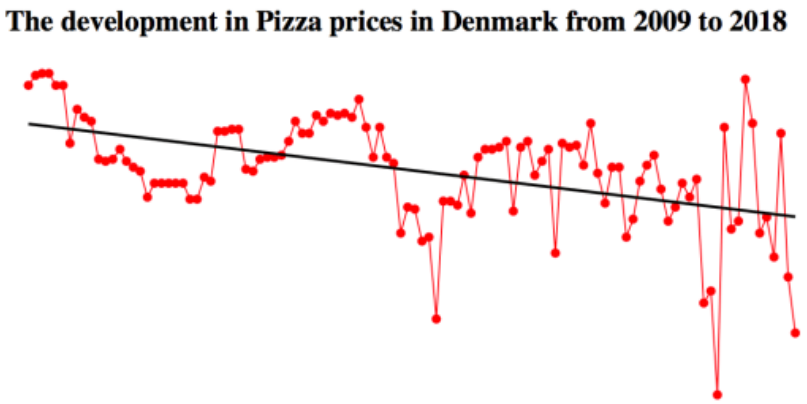


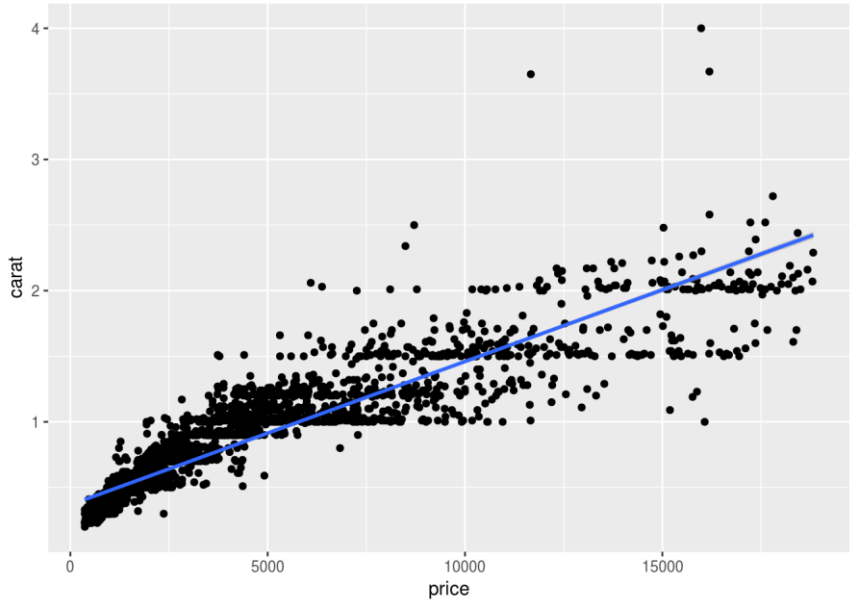
$$Y = mX + b$$

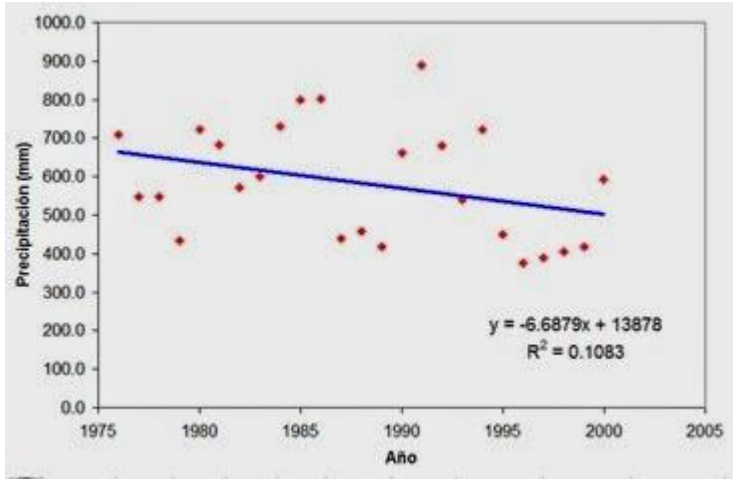
*Donde **Y** es el resultado, **X** es la variable, **m** la pendiente (o coeficiente) de la recta y **b** la constante o también conocida como el "**punto de corte con el eje Y**" en la gráfica (cuando X=0)*

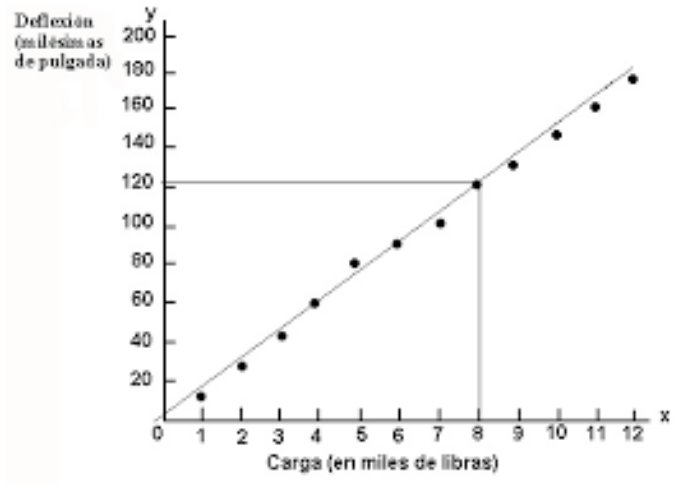
Pero también es un algoritmo de aprendizaje supervisado que se utiliza en Machine Learning

Ejemplos de regresión lineal









¿Cómo funciona el algoritmo de regresión lineal en Machine Learning?

Recordemos que los algoritmos de ***Machine Learning Supervisados***, aprenden por sí mismos y -en este caso- a obtener automáticamente esa “recta” que buscamos con la tendencia de predicción.

Para hacerlo se mide el error con respecto a los puntos de entrada y el valor “Y” de salida real.
El algoritmo deberá minimizar el coste de una función de ***error cuadrático*** y esos coeficientes corresponderán con la recta óptima.

Hay diversos métodos para conseguir minimizar el coste. Lo más común es utilizar una versión vectorial y la llamada **Ecuación Normal** que nos dará un resultado directo.

Sklearn

DATASET'S

- Herramientas simples y eficientes para la minería de datos y el análisis de datos.
- Construido en NumPy, SciPy y matplotlib
- Código abierto, utilizable comercialmente - licencia BSD

Sklearn
DATASET'S

La biblioteca sklearn proporciona una lista de "***conjuntos de datos de juguetes***" con el fin de probar algoritmos de aprendizaje automático. Los datos se devuelven de las siguientes:

- **load_boston()** Precios de la vivienda de Boston por regresión
- **load_iris()** El conjunto de datos de iris para la clasificación
- **load_diabetes()** El conjunto de datos de diabetes para regresión
- **load_digits()** Imágenes de dígitos para clasificación
- **load_linnerud()** El conjunto de datos linnerud para regresión multivariante
- **load_wine()** El conjunto de datos del vino para la clasificación
- **load_breast_cancer()** El conjunto de datos de cáncer de mama para clasificación

Imagina que se gana una beca para ir a estudiar en una de las universidades de **Big-Boston** (Boston-Cambridge-Quincy):

- Universidad de Harvard
- El Instituto Tecnológico de Massachusetts (MIT)
- La Universidad de Tufts

Recordemos que **Big-Boston** no solo es una de las Áreas metropolitanas más antiguas de estados unidos y una de las más pobladas con cerca de 4,5 millones de habitantes(2018), también tiene un alto grado de criminalidad, especialmente contra los extranjeros, principalmente **latinos**...

Entonces en que hacer?

Rechazar la beca por que me da miedo la criminalidad o busco una buena parte de Bostón donde podría quedarme?

Lo mejor es usar algo de inteligencia artificial

Que la ciencia nos ayude a mejorar nuestras decisiones usando un buen modelo de predicción.

¿Pero qué tipos de modelos de IA Existen?

- *Regresión lineal (Es método para encontrar el patrón con una "Mejor Línea de Ajuste")*
- *Regresión logística.*
- *Árboles de clasificación y regresión.*
- *K-means*
- *Redes bayesianas*
- *Máquinas de vectores soporte (VSM)*
- ***Deep learning***

DATASET'S en Phyton – load_Boston()

Empezaremos a trabajar con grandes cantidades de información denominada DATASET'S.

Para los ejercicios iniciales utilizaremos Dataset's de acceso publico creados por scikit-learn

Paso 1: importando librerías necesarias

```
import numpy as np            #Mejora el soporte para vectores y matrices
import pandas as pd           #Estructura de datos (Ciencia de datos)

import matplotlib.pyplot as plt #Para graficar
import seaborn as sns          #interfaz mejorada para dibujar gráficos estadísticos (basada en matplotlib)
```

Paso 2: cargamos los datos de la biblioteca scikit-learn

```
from sklearn.datasets import load_boston
boston_dataset = load_boston()
```

Paso 3: Conociendo los datos que tiene el dataset

```
print(boston_dataset.keys())
```

DATASET'S en Phyton – load_Boston()

Paso 4: Conociendo las caracteristicas que tienen los datos:

```
boston_dataset.DESCR
```

CRIM: Tasa de delincuencia per cápita por ciudad
ZN: Proporción de terrenos residenciales divididos en zonas para lotes de más de 25,000 pies cuadrados
INDUS: Proporción de acres comerciales no minoristas por ciudad
CHAS: Variable ficticia de Charles River (= 1 si el tramo limita con el río; 0 en caso contrario)
NOX: concentración de óxido nítrico (partes por 10 millones)
RM: Número medio de habitaciones por vivienda
EDAD: Proporción de unidades ocupadas por el propietario construidas antes de 1940
DIS: distancias ponderadas a cinco centros de empleo de Boston
RAD: Índice de accesibilidad a carreteras radiales
TAX/IMPUESTO: Tasa de impuesto a la propiedad de valor total por USD 10.000
PTRATIO: Proporción alumno/profesor por municipio
B: 1000 (Bk - 0,63)², donde Bk es la proporción de personas de ascendencia afroamericana por ciudad
LSTAT: porcentaje de la población de menor estatus (pobres)
***(Target)MEDV**: Valor medio de las viviendas ocupadas por sus propietarios en $ 1000*

DATASET'S en Phyton – load_Boston()

Paso 5: Creamos una tabla de datos usando pandas (facilita el manejo):

```
#creamos una tabla (tipo excel, con titulos para faciliar la manipulación)
boston = pd.DataFrame(boston_dataset.data, columns=boston_dataset.feature_names)
boston.head()   #imprimimos las primeras 5 filas
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

Nota: los valores de target no están en el dataset, por lo tanto es necesario agregarlo a la tabla

```
#Agregamos en la tabla los valores de target del dataset
boston['MEDV'] = boston_dataset.target
```

DATASET'S en Phyton – load_Boston()

P1Tx-1: **Analizando los datos del dataset:**

Usando pandas en LOAD_BOSTON, determine (15 minutos)

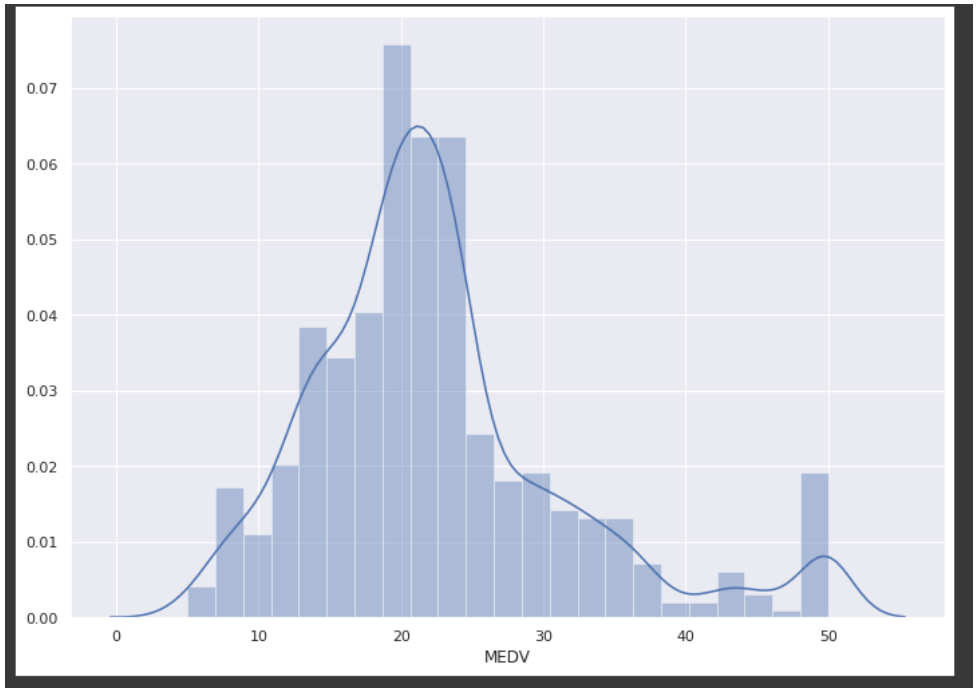
- Cuantos registros tiene el dataset (rows)
- Cuántos datos tiene cada registro (columns)
- Hay datos nulos (null) en el dataset?

DATASET'S en Phyton – load_Boston()

Paso 6: Graficando los datos del dataset

Primero revisemos la distribución de la variable de destino (target), para garantizar al distribución de los valores

```
sns.set(rc={'figure.figsize':(11.7,8.27)})  #tamaño del grafico
sns.distplot(boston['MEDV'])                #agregamos los datos
plt.show()                                  #visualizamos el grafico
```



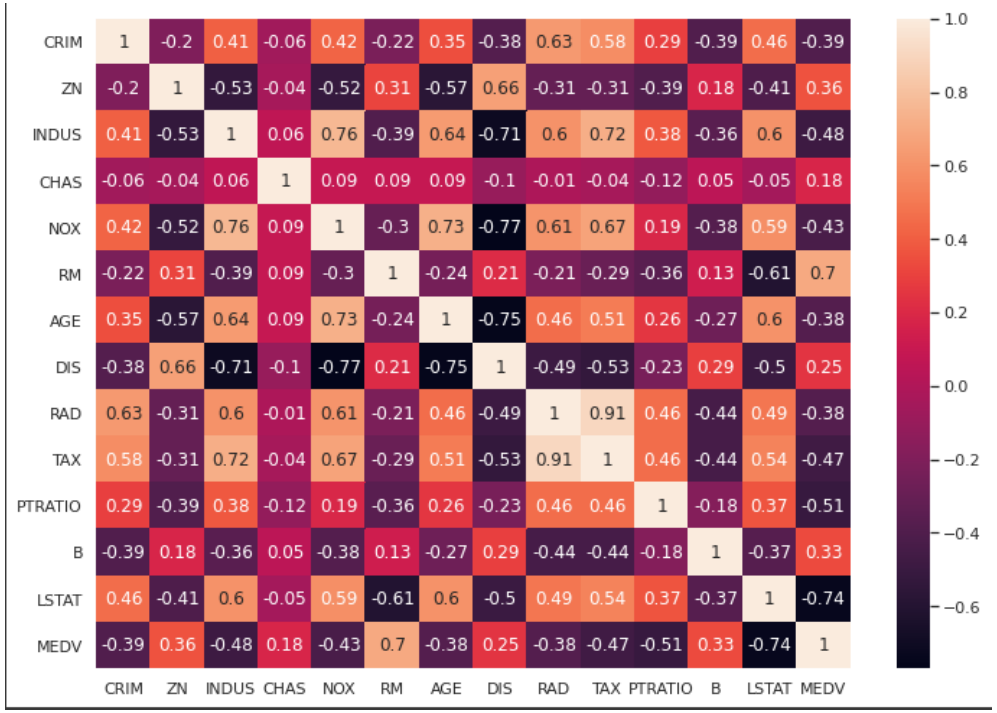
Los valores se distribuyen normalmente con pocos valores atípicos (*la única diferencia fuerte es con 50*)

DATASET'S en Phyton – load_Boston()

Paso 6: Graficando los datos del dataset (matriz de correlación de características)

para medir las relaciones lineales entre las variables y de esa forma determinar que valores son más practicos para entregar un modelo de regresión lin

```
#función de correlación de pandas (cercano a 1 es la mejor correlación, negativos la peor)
correlation_matrix = boston.corr().round(2)
# annot = True (para imprimir los valores dentro del cuadrado)
sns.heatmap(data=correlation_matrix, annot=True)
```



El coeficiente de correlación oscila entre -1 y 1. Si el valor está cerca de 1, significa que hay una fuerte correlación positiva entre las dos variables.

Cuando está cerca de -1, las variables tienen una fuerte correlación negativa.

DATASET'S en Phyton – load_Boston()

Paso 7: Seleccionar las características que tienen una alta correlación

Se deben seleccionar aquellas características que tienen una alta correlación (ya se positiva o negativa) con nuestra variable de destino (MEDV).

Entre 0.7 a 0.74 sea positivo o negativo:

MEDV <-> RM
MEDV <-> LSTAT

Se deben descartar las características que tengan multi-colinealidad (correlación utópica que solo se podría dar en laboratorio), son aquellas que tengan valores superiores 0,74:

RAD vs TAX
DIS vs AGE

Usaremos un gráfico de dispersión para ver cómo estas características varían

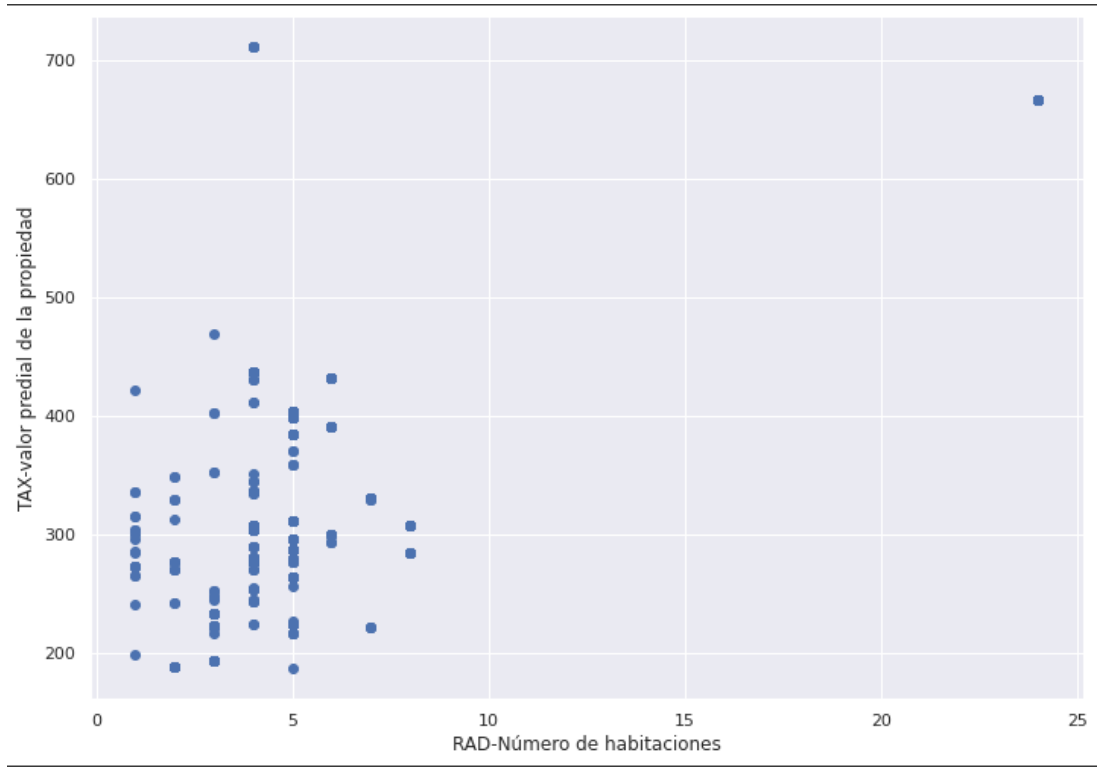
RM vs MEDV
LSTAT vs MEDV

DATASET'S en Phyton – load_Boston()

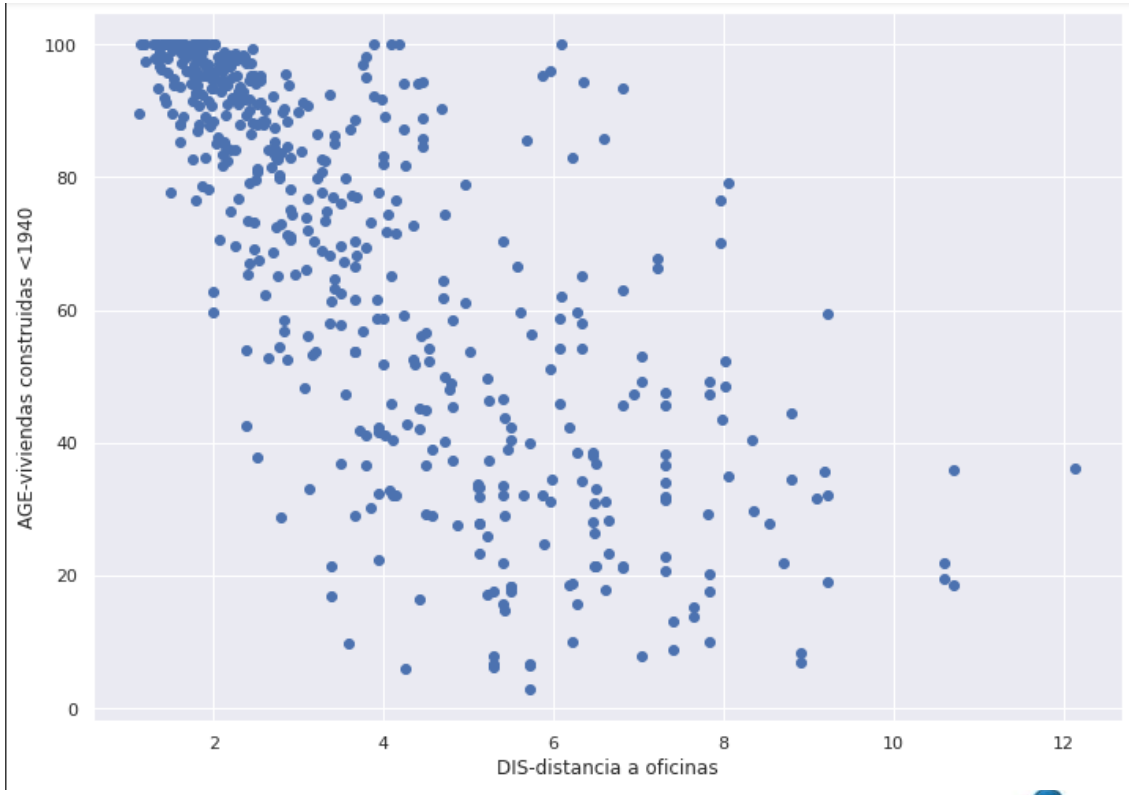
Paso 8: Graficando dispersión de variables DESCARTADAS

Para comprobar si es cierto grafiquemos también las que se descartan para ver como es la corelacción y por que se descartaron:

TAX - RAD



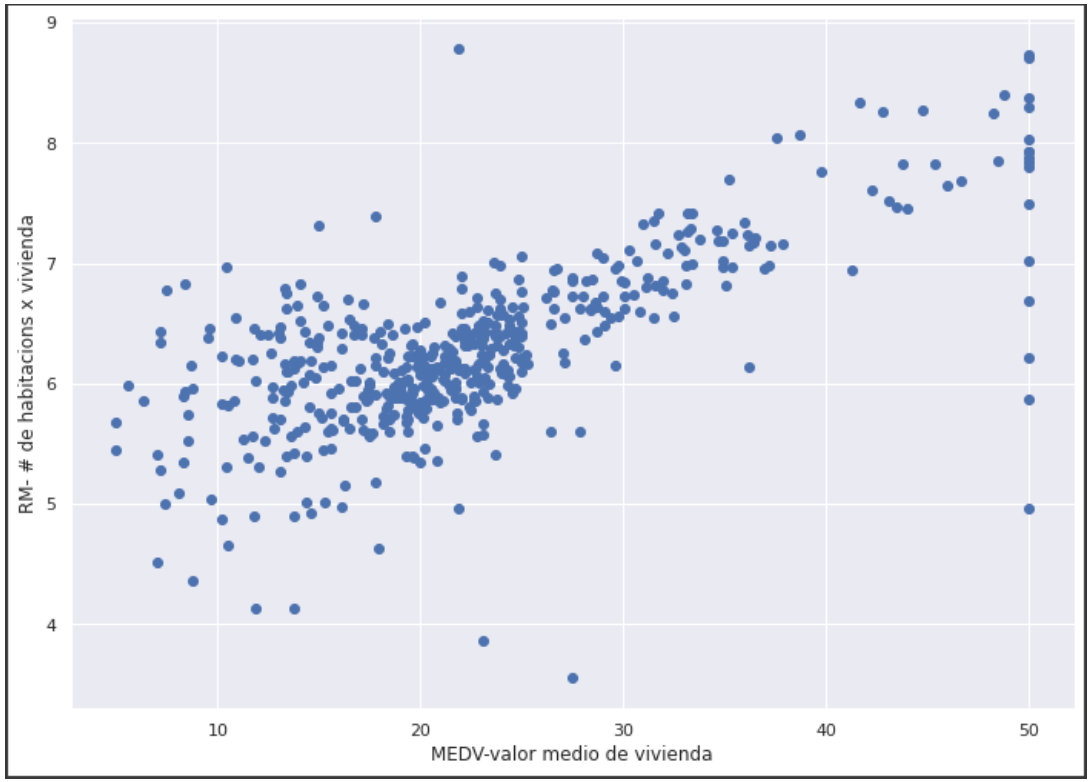
DIS - AGE



DATASET'S en Phyton – load_Boston()

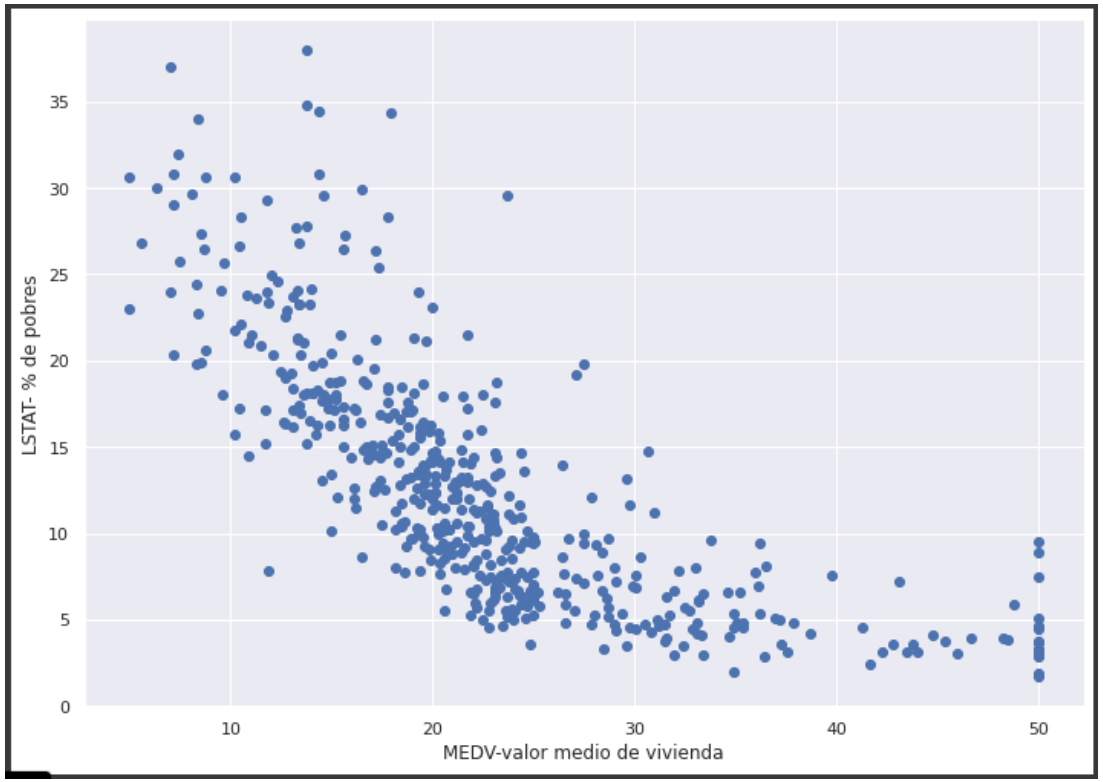
Paso 8: Graficando dispersión de variables seleccionadas

RM vs MEDV



Los precios aumentan a medida que el valor de **RM** aumenta linealmente. Hay pocos valores atípicos y los datos parecen estar limitados a 50.

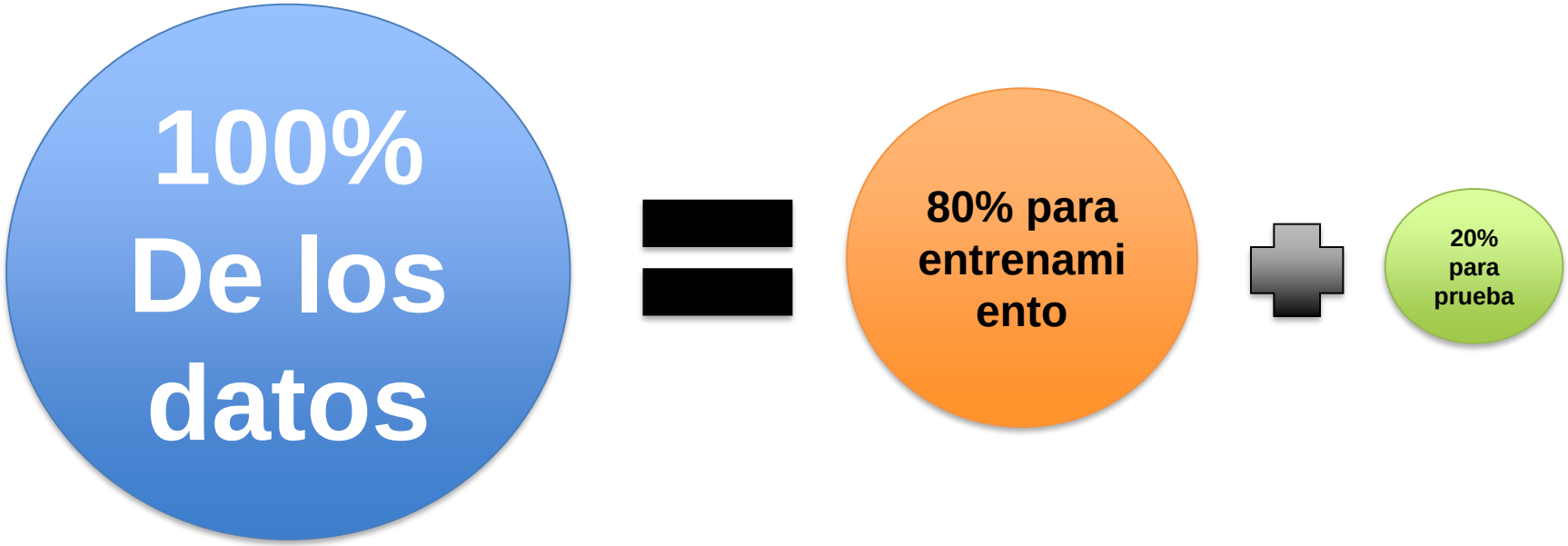
LSTAT vs MEDV



Los precios tienden a disminuir con un aumento en LSTAT (porcentaje de pobre)

Aplicando a LOAD BOSTON un algoritmo de regresión lineal

Paso 8: Separar datos (entrenamiento, test)



Lo primero que haremos es separar los datos en entrenamiento y prueba lo hacemos utilizando la instrucción **train_test_split**

Aplicando a LOAD BOSTON un algoritmo de regresión lineal

Paso 8: Preparar datos (X y Y) = y=mX+b

Concatenamos las columnas que seleccionamos para entrenar:
X = LSTAT y RM
Y = MEDV

```
#Entrenando con una sola variable en X

X = pd.DataFrame(np.c_[boston['LSTAT']], columns = ['LSTAT'])
X = pd.DataFrame(np.c_[boston['RM']], columns = ['RM'])

#entrenando con dos variables en X
X = pd.DataFrame(np.c_[boston['LSTAT'], boston['RM']], columns = ['LSTAT','RM'])
Y = boston['MEDV']
```

Paso 9: Separar datos en entrenamiento (80%) y test(20%)

```
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state=5)
print("x80%: "+str(X_train.shape) +", x20%: "+str(X_test.shape))
print("y80%: "+str(Y_train.shape) +", y20%: "+str(Y_test.shape))
```

Aplicando a LOAD BOSTON un algoritmo de regresión lineal

Paso 10: entrenando un modelo de regresión lineal

Usando los datos de entrenamiento los aplicamos al algoritmo de regresión lineal.

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

lin_model = LinearRegression()
lin_model.fit(X_train, Y_train)
```

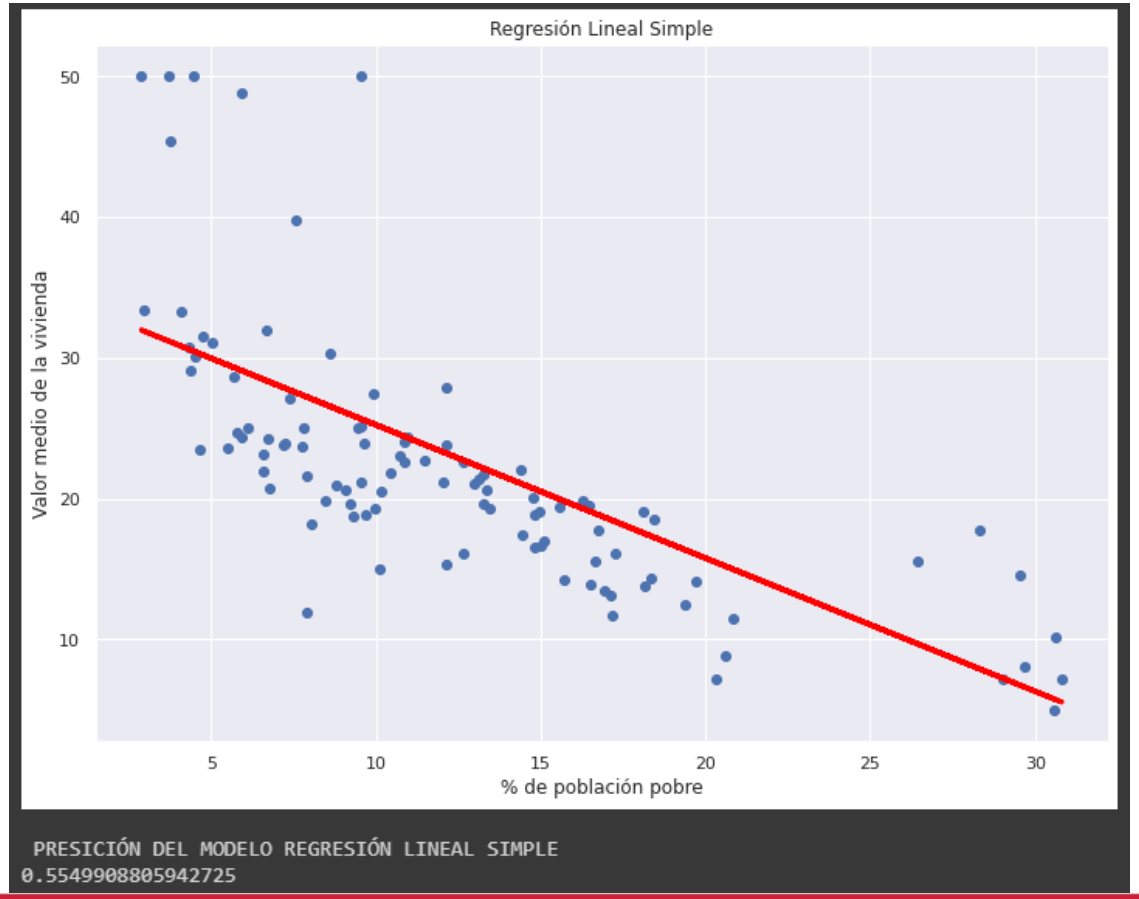
Paso 11: evaluando el modelo

```
# poner a prueba la maquina (modelo)
Y_pred = lin_model.predict(X_test)
plt.scatter(X_test['RM'], Y_test)
plt.plot(X_test, Y_pred, color='red', linewidth=3)
plt.title('Regresión Lineal Simple')
plt.xlabel('Número de habitaciones')
plt.ylabel('Valor medio')
plt.show()
print('\n PRESICIÓN DEL MODELO REGRESIÓN LINEAL SIMPLE')
print(lin_model.score(X_train, Y_train))
```

Aplicando a LOAD BOSTON un algoritmo de regresión lineal

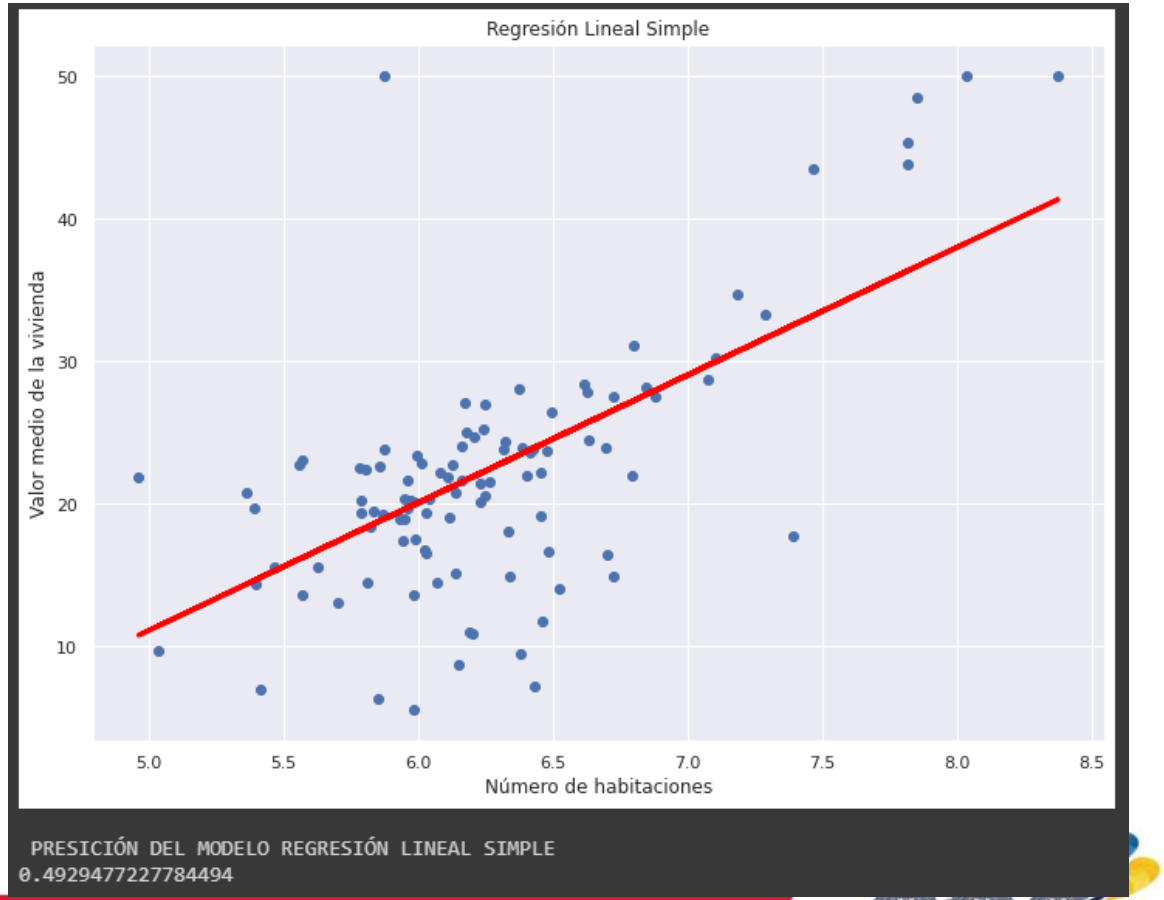
Entrenando un modelo con una sola variable MEVD Vs

Predicción con LSTAT (% de pobres)




```
 PRESICIÓN DEL MODELO REGRESIÓN LINEAL SIMPLE
0.5549908805942725
```

Predicción con RM (# de habitaciones)

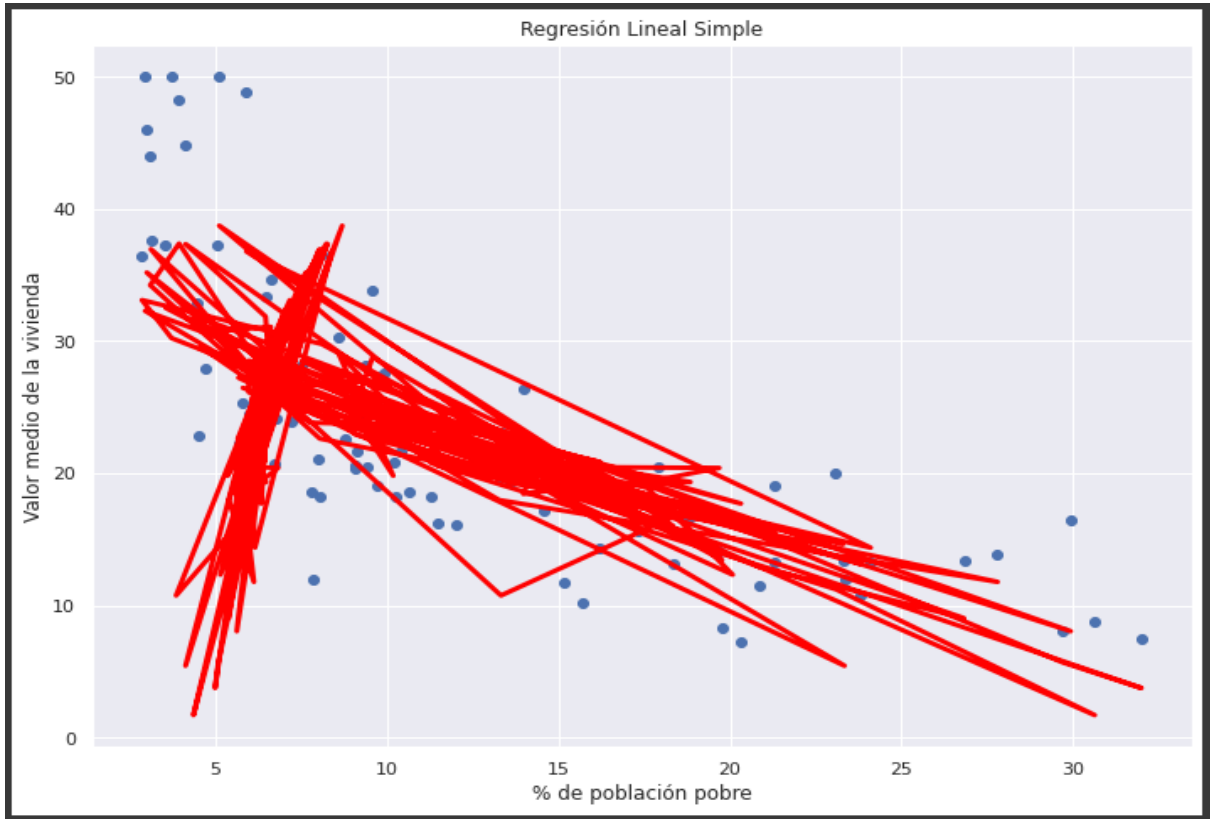



```
 PRESICIÓN DEL MODELO REGRESIÓN LINEAL SIMPLE
0.4929477227784494
```

Aplicando a LOAD BOSTON un algoritmo de regresión lineal

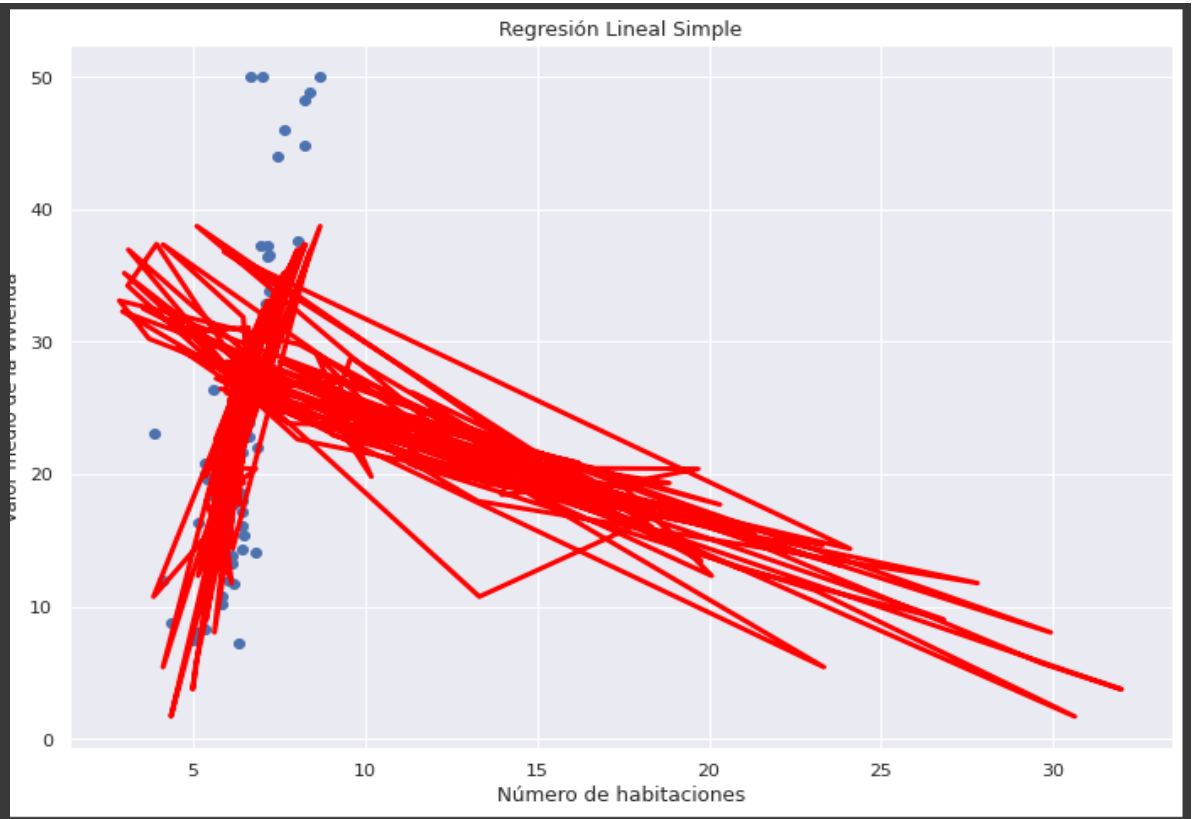
Entrenando un modelo con una sola variable MEVD Vs (LSTAT + RM)

Predicción con LSTAT (% de pobres)



PRESICIÓN DEL MODELO REGRESIÓN LINEAL SIMPLE
0.6224960438968672

Predicción con RM (# de habitaciones)



PRESICIÓN DEL MODELO REGRESIÓN LINEAL SIMPLE
.6224960438968672

Un buen modelo debe arrojar una precisión superior al 80% (siempre).

De lo contrario hay que buscar otro algoritmo

Aplicando a LOAD BOSTON algoritmo FULL

```
import numpy as np          #Mejora el soporte para vectores y matrices
import pandas as pd         #Estructura de datos (Ciencia de datos)

import matplotlib.pyplot as plt #Para graficar
import seaborn as sns        #interfaz de alto nivel para dibujar gráficos estadísticos (basada en matplotlib)

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

from sklearn.datasets import load_boston

boston_dataset = load_boston()
#creamos una tabla (tipo excel con PANDAS, con titulos para faciliar la manipulación)
boston = pd.DataFrame(boston_dataset.data, columns=boston_dataset.feature_names)
#Agregamos en la tabla los valores de target del dataset
boston['MEDV'] = boston_dataset.target
#entrenando con dos variables en X
X = pd.DataFrame(np.c_[boston['LSTAT'], boston['RM']], columns = ['LSTAT','RM'])
Y = boston['MEDV']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2)
#Entrenando el modelo
lin_model = LinearRegression()
lin_model.fit(X_train, Y_train)
# poner a prueba la maquina (modelo)
Y_pred = lin_model.predict(X_test)
plt.scatter(X_test['LSTAT'], Y_test)
plt.plot(X_test, Y_pred, color='red', linewidth=3)
plt.title('Regresión Lineal Simple')
#plt.xlabel('Número de habitaciones')
plt.xlabel('% de población pobre')
plt.ylabel('Valor medio de la vivienda')
plt.show()
print('\n PRESICIÓN DEL MODELO REGRESIÓN LINEAL SIMPLE')
print(lin_model.score(X_train, Y_train))
```

¿Y que pasaría si yo entrenada con todo?

Sin descartar ninguna variable

Aplicando a LOAD BOSTON un algoritmo de regresión lineal (ENTRENADO CON TODO)

Librerías y volviendo todo a tabla de panda

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
boston_data = datasets.load_boston()
boston_df = pd.DataFrame(boston_data.data, columns=boston_data.feature_names)
boston_df.head()
```

Separar los datos (TRAIN, TEST)

```
scalar = StandardScaler()
Y = boston_data.target
X = boston_df.values            #tomaremos todos los columnas para entrenar
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2)
print("TRAIN--> X: {} - Y: {}".format(X_train.shape,y_train.shape))
print("TEST--> X: {} - Y: {}".format(X_test.shape,y_test.shape))
```

Aplicando a LOAD BOSTON un algoritmo de regresión lineal (ENTRENADO CON TODO)

Aplicar regresión lineal

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
#entrenamos
regressor.fit(X_train, y_train)
#predecimos
pred = regressor.predict(X_test)
```

visualizar la regresión

```
#visualizar la predicción en los datos de testeo
plt.scatter(y_test, pred)
plt.plot([y.min(), y.max()], [y.min(), y.max()], c='r', lw=2)
plt.show()
print("Precisión del modelo: "+str(regressor.score(X_test, y_test)))
```

Aplicando a LOAD BOSTON un algoritmo de regresión lineal (ENTRENADO CON TODO)

¿empeora o mejora?

¿Se puede mejorar aún más?