



UNIVERSIDAD
NACIONAL
DE COLOMBIA

CLASIFICACIÓN Y RECONOCIMIENTO DE PATRONES

JOHN W. BRANCH

Profesor Titular

Departamento de Ciencias de la Computación y de la Decisión
Director del Grupo de I+D en Inteligencia Artificial – GIDIA

jwbranch@unal.edu.co

<https://github.com/jwbranch/ClasificacionYReconocimientoDePatrones>

<https://www.coursera.org/programs/unal-iuukt>

Descripción del Curso

La descripción, la agrupación, la clasificación y el reconocimiento, de patrones de forma automática, son problemas importantes en una gran variedad de aplicaciones de ingeniería, psicología, medicina, economía, biologías, etc. El problema consiste en asignar automáticamente a una clase una muestra según las mediciones realizadas sobre la muestra. En el curso se estudiará la teoría necesaria para resolver este problema, y se aplicará la teoría en aplicaciones de diferentes áreas.

OBJETIVOS DEL CURSO

Identificar los componentes principales de un sistema de Clasificación y Reconocimiento de Patrones, a partir de **información visual.**

Estudiar los conceptos básicos de extracción de características, selección de características, clasificación y evaluación de desempeño

Implementar los algoritmos asociados con las distintas etapas y modelos de un sistema de Clasificación y Reconocimiento de Patrones.

Aplicar técnicas para establecer y comparar el desempeño de los clasificadores.

Aplicar métodos de Clasificación y Reconocimiento de Patrones para resolver un problema real.

Contenido

1. Introducción.
2. Adquisición, Procesamiento y Etiquetado de Datos
3. Selección de Características.
4. Extracción de Características.
5. Clasificación Supervisada.
6. Clasificación No Supervisada
7. Evaluación de Desempeño.
8. Aplicaciones y Casos de Éxito.

CLASIFICACIÓN Y RECONOCIMIENTO DE PATRONES

Selección y Extracción de Características

Reducción de la Dimensionalidad

Jorge E. Espinosa

Profesor

Departamento de Ciencias de la Computación y de la Decisión
Investigador del Grupo de I+D en Inteligencia Artificial – GIDIA

jeespinosao@unal.edu.co

Contenido

1. Generalidades
2. Por qué la Reducción de la Dimensionalidad ?
3. Principales técnicas de reducción de la dimensionalidad
 - a. Selección de Características
 - b. Extracción de Características
4. Reducción de la dimensionalidad en Python
 - a. Selección Vs Extracción
5. t-SNE
6. **Lab 1** ☺
7. La Maldición de la Dimensionalidad (The curse of dimensionality)
8. Correlación de Valores
9. **Lab 2** ☺

Contenido

10. Regresión Lineal

11. Regresión Logística

a. RL para eliminación de Características

12. Eliminación de Características hacia atrás

a. RFE – Recursive Feature Elimination

13. Árboles de Decisión

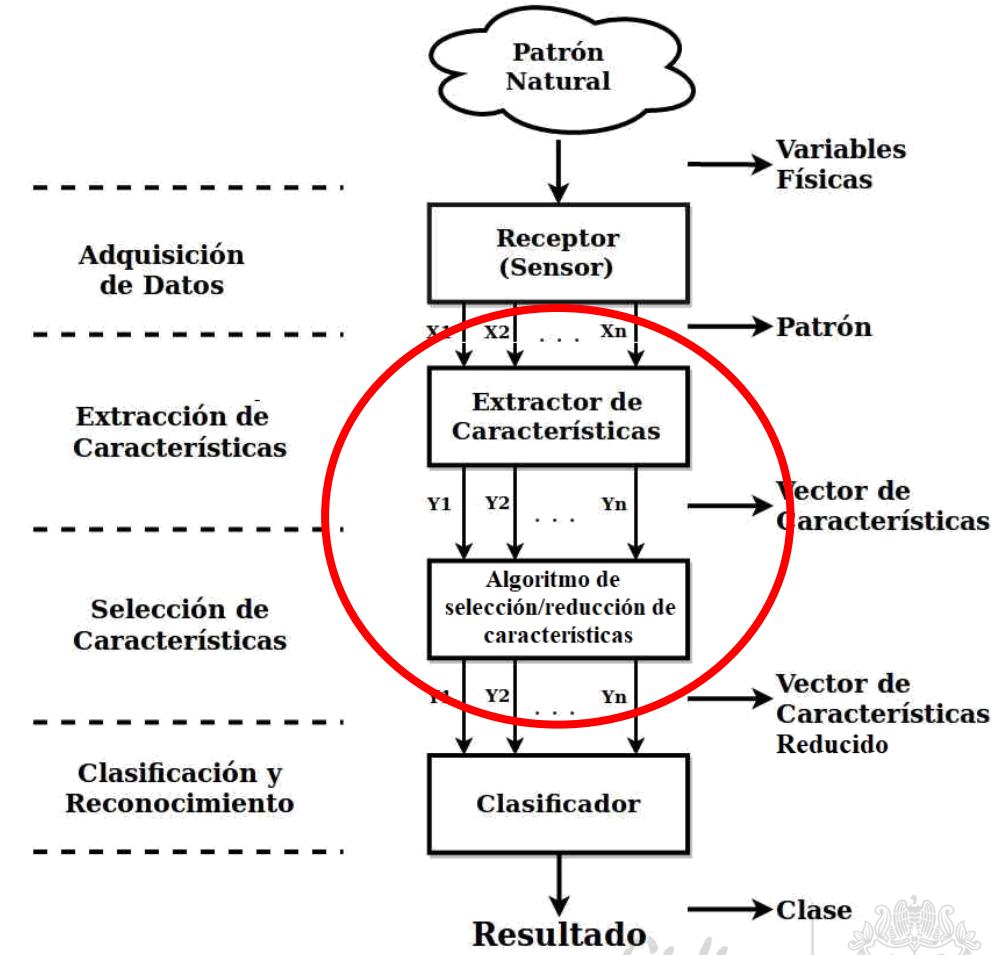
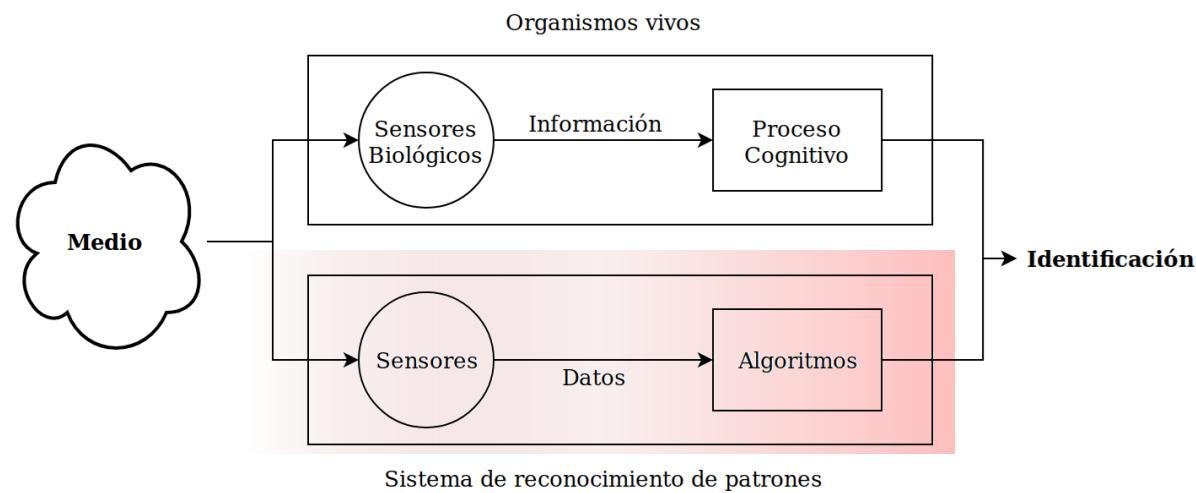
a. Random Forest –Bosques Aleatorios

b. Random Forest para la Eliminación de características

Contenido

- 14. Regresión Lineal Multivariable
 - a. Gradiente descendiente para múltiples variables
 - b. Ecuación Normal
 - c. Regresión Multivariable en Python
- 15. El regresor LASSO
- 16. Regresor LASSO CV
- 17. Combinación de varios selectores de características.
- 18. Lab 3 ☺

Etapas Sistema de Clasificación y Reconocimiento de Patrones

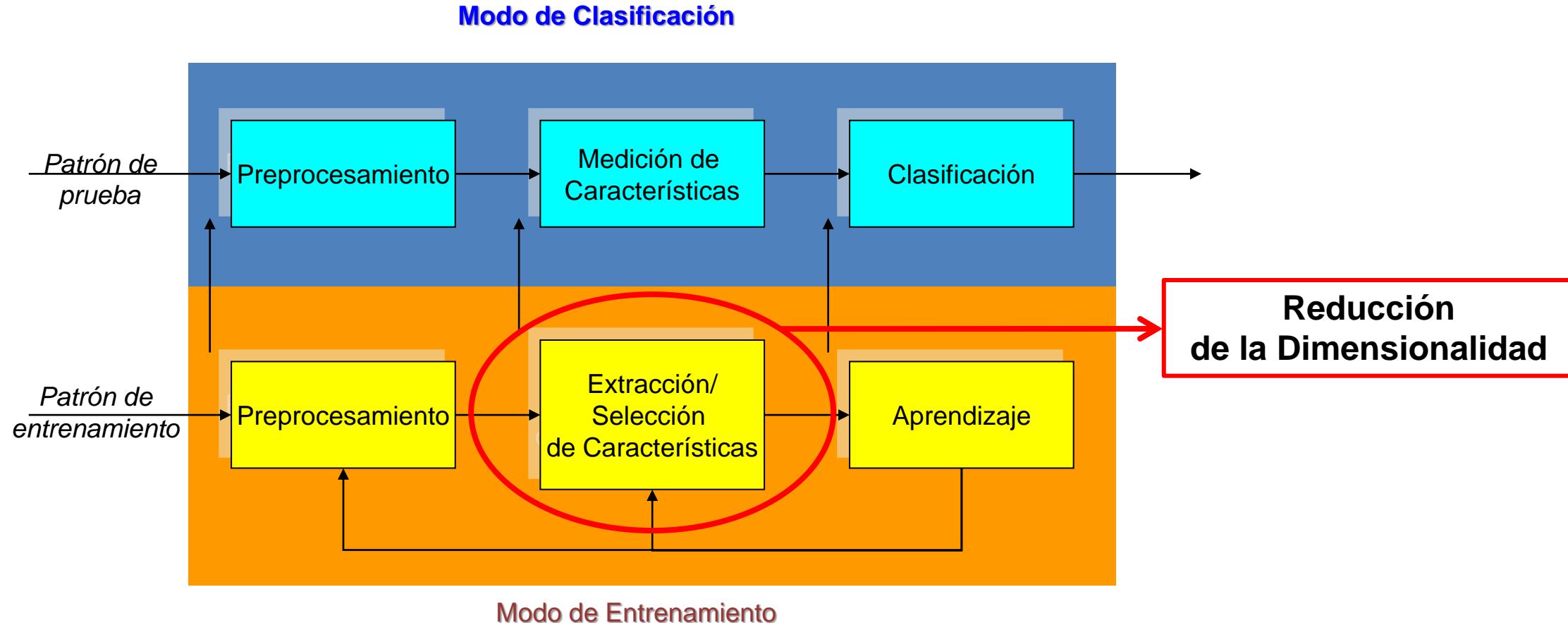


Representación de un Patrón

Un patrón es representado por un conjunto de *d* **características**, o atributos, visto como un vector *d*-dimensional *vector de características*.

$$\mathbf{x} = (x_1, x_2, \dots, x_d)^T$$

Dos procesos para la Implementación de sistemas de reconocimiento de Patrónes



Por qué la Reducción de la Dimensionalidad ?

En la actualidad es muy fácil **recopilar Datos**

Los datos **no son únicamente recopilados para análisis**

Los datos se acumulan con una velocidad sin precedentes

El **preprocesamiento de los datos** (Reduciendo Dimensionalidad?) resulta *efectivo* en el reconocimiento de patrones y el aprendizaje automático

La **reducción de la dimensionalidad** es una técnica efectiva en la optimización de la información

Por qué la Reducción de la Dimensionalidad ?

La mayoría de las técnicas de aprendizaje automático y de reconocimiento de patrones **pueden no ser efectivas** en datos con alta dimensionalidad

La Maldición de la dimensionalidad

La precisión y la eficiencia de la consulta se degradan rápidamente a medida que aumenta la dimensión.

La dimensión **intrínseca** puede ser pequeña.

Por ejemplo, el número de genes responsables de cierto tipo de enfermedad puede ser pequeño.

Por qué la Reducción de la Dimensionalidad ?

Visualización: proyección de datos multidimensionales en 2D o 3D.

Compresión de datos: almacenamiento y recuperación eficientes.

Eliminación de ruido: efecto positivo en la precisión de la consulta.

Aplicaciones de la Reducción de la Dimensionalidad

Gestión de la relación con el cliente

Extracción de textos

Recuperación de imágenes

Análisis de datos de microarrays

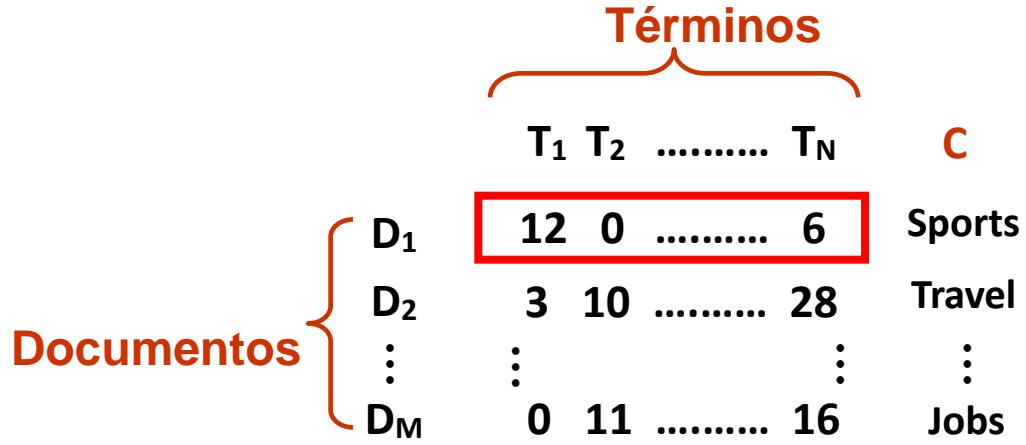
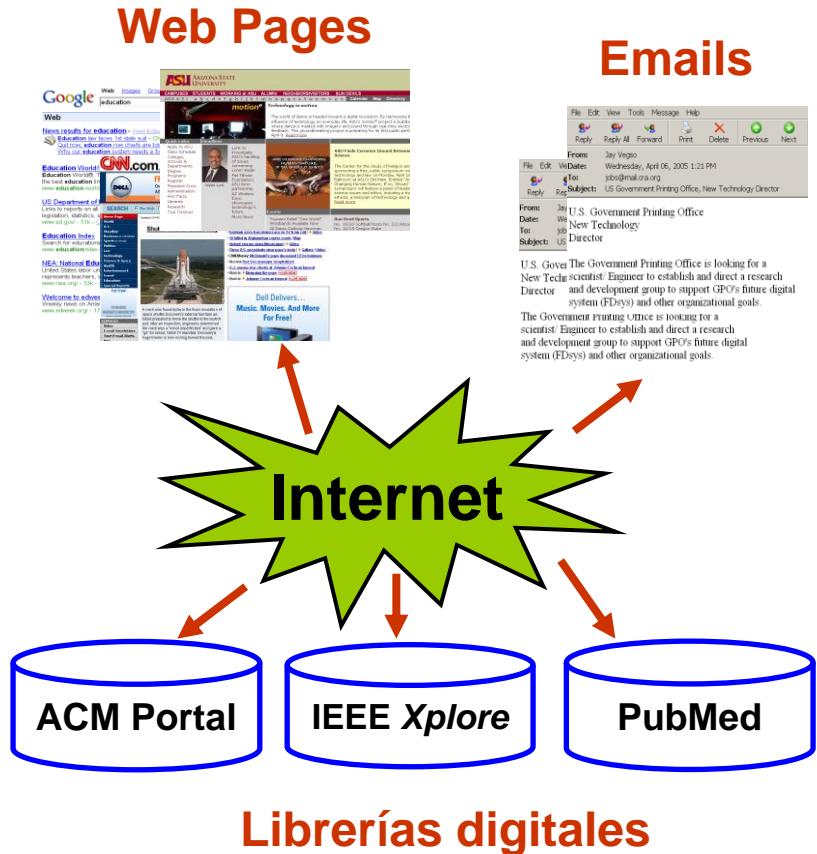
Clasificación de proteínas

Reconocimiento facial

Reconocimiento de dígitos escritos a mano

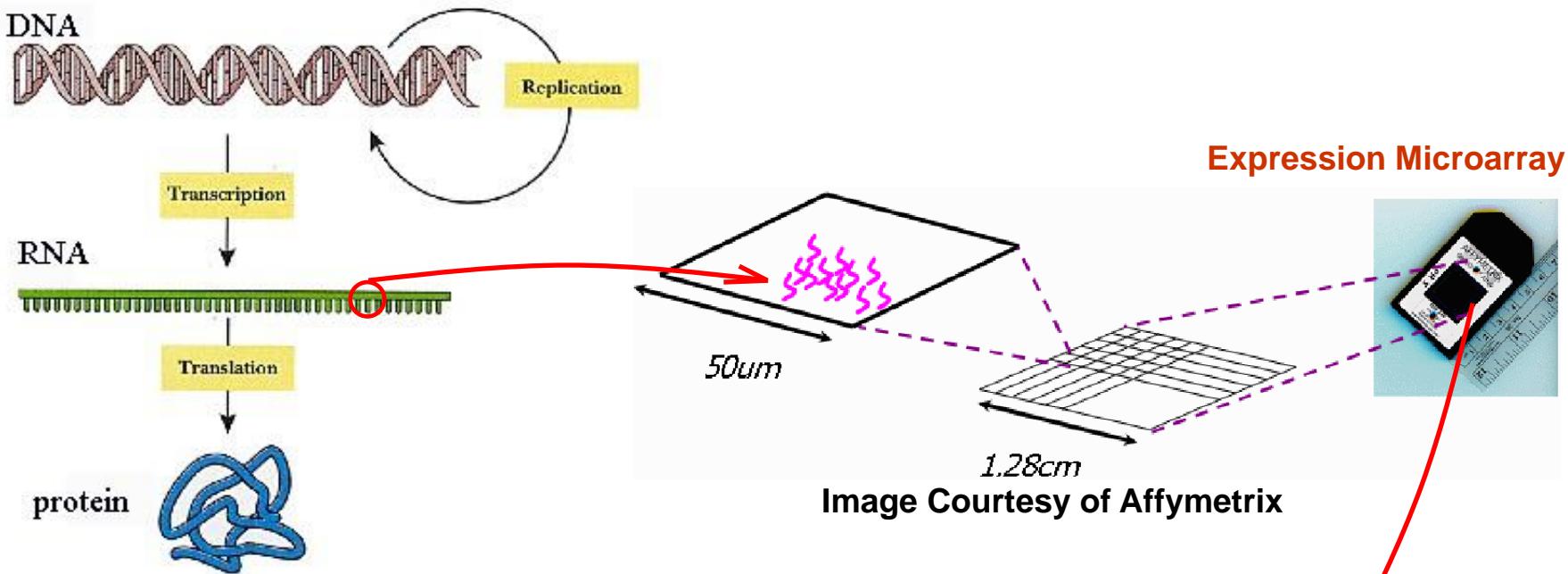
Detección de intrusiones

Clasificación de documentos



- **Tarea:** clasificar documentos sin etiqueta en categorías
- **Desafío:** miles de términos
- **Solución:** Aplicar reducción de la dimensionalidad

Análisis de microarrays de expresión genética



Tarea: Clasificar nuevas muestras en tipos de enfermedades conocidas (diagnóstico de enfermedades)

Desafío: miles de genes, pocas muestras

Solución: aplicar reducción de dimensionalidad

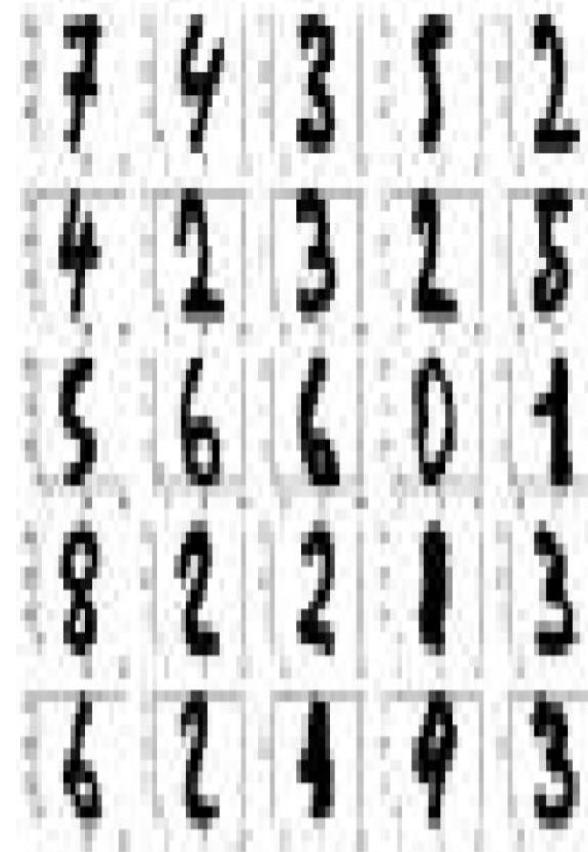
Gene Sample \	M23197_at	U66497_at	M92287_at	...	Class
Sample 1	261	88	4778	...	ALL
Sample 2	101	74	2700	...	ALL
Sample 3	1450	34	498	...	AML
.
.
.

Expression Microarray Data Set

Otros tipos de datos multidimensionales



Imágenes de rostros



Dígitos manuscritos

Principales técnicas de reducción de dimensionalidad

Selección de características

Definición

Objetivos

Extracción de características

Definición

Objetivos

Diferencias entre las dos técnicas

Selección de características

Definición

Un proceso que elige un subconjunto óptimo de características de acuerdo con una función objetivo

Objetivos:

Para reducir la dimensionalidad y eliminar el ruido.

Para mejorar el desempeño del reconocimiento de patrones

Velocidad de aprendizaje

Precisión predictiva

Simplicidad y comprensibilidad de los resultados de reconocimiento de patrones.

Extracción de características

La extracción de características se refiere al mapeo de los datos originales de alta dimensión en un espacio de menor dimensión

Dado un conjunto de puntos de datos de p variables $\{x_1, x_2, \dots, x_n\}$
Computar su representación en una dimensión más reducida:

$$x_i \in \Re^d \rightarrow y_i \in \Re^p \quad (p \ll d)$$

El criterio para la extracción de características puede ser diferente de acuerdo a las diferentes configuraciones de problemas.

Configuración **no supervisada**: minimice la pérdida de información

Entorno **supervisado**: maximizar la discriminación de clase

Selección de características vs. Extracción de características

Selección de las características

Solo se selecciona un subconjunto de las características originales.

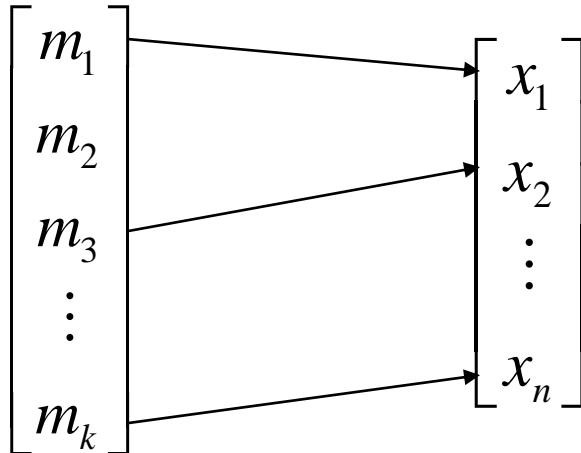
Extracción de características

Se utilizan todas las características originales.

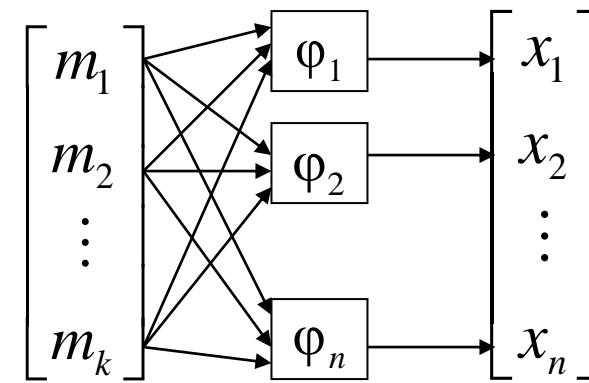
Las características transformadas son combinaciones lineales (en la mayoría de los casos) de las características originales.

Métodos de Reducción de la Dimensionalidad

Selección de Características



Extracción de Características



El problema puede expresarse como la optimización de los parámetros $\varphi(\theta)$ del extractor de características.

Métodos supervisados: la función objetivo es un criterio de separabilidad (discriminabilidad) de ejemplos etiquetados, por ejemplo, análisis de discriminación lineal (LDA).

Métodos no supervisados: se busca una representación dimensional más baja que conserve las características importantes de los datos de entrada, por ejemplo, análisis de componentes principales (PCA).

Datos Organizados

Name	Type	HP	Attack	Defense	Speed	Generation
Bulbasaur	Grass	45	49	49	45	1
Ivysaur	Grass	60	62	63	60	1
Venusaur	Grass	80	82	83	80	1
Charmander	Fire	39	52	43	65	1
Charmeleon	Fire	58	64	58	80	1

Datos Organizados

Name	Type	HP	Attack	Defense	Speed	Generation
Bulbasaur	Grass	45	49	49	45	1
Ivysaur	Grass	60	62	63	60	1
Venusaur	Grass	80	82	83	80	1
Charmander	Fire	39	52	43	65	1
Charmeleon	Fire	58	64	58	80	1

Datos Organizados

Name	Type	HP	Attack	Defense	Speed	Generation
Bulbasaur	Grass	45	49	49	45	1
Ivysaur	Grass	60	62	63	60	1
Venusaur	Grass	80	82	83	80	1
Charmander	Fire	39	52	43	65	1
Charmeleon	Fire	58	64	58	80	1

Datos Organizados (comando shape)

Name	Type	HP	Attack	Defense	Speed	Generation
Bulbasaur	Grass	45	49	49	45	1
Ivysaur	Grass	60	62	63	60	1
Venusaur	Grass	80	82	83	80	1
Charmander	Fire	39	52	43	65	1
Charmeleon	Fire	58	64	58	80	1

```
pokemon_df.shape
```

(5, 7)

Cuando Utilizamos Reducción de Características?

Name	Type	HP	Attack	Defense	Speed	Generation
Bulbasaur	Grass	45	49	49	45	1
Ivysaur	Grass	60	62	63	60	1
Venusaur	Grass	80	82	83	80	1
Charmander	Fire	39	52	43	65	1
Charmeleon	Fire	58	64	58	80	1

Datos Organizados (comando describe())

	HP	Attack	Defense	Speed	Generation
count	5.0	5.0	5.0	5.0	5.0
mean	56.4	61.8	59.2	66.0	1.0
std	15.9	13.0	15.4	14.7	0.0
min	39.0	49.0	43.0	45.0	1.0
25%	45.0	52.0	49.0	60.0	1.0
50%	58.0	62.0	58.0	65.0	1.0
75%	60.0	64.0	63.0	80.0	1.0
max	80.0	82.0	83.0	80.0	1.0

```
pokemon_df.describe()
```

Datos Organizados (comando describe())

	HP	Attack	Defense	Speed	Generation
count	5.0	5.0	5.0	5.0	5.0
mean	56.4	61.8	59.2	66.0	1.0
std	15.9	13.0	15.4	14.7	0.0
min	39.0	49.0	43.0	45.0	1.0
25%	45.0	52.0	49.0	60.0	1.0
50%	58.0	62.0	58.0	65.0	1.0
75%	60.0	64.0	63.0	80.0	1.0
max	80.0	82.0	83.0	80.0	1.0

```
pokemon_df.describe()
```

Selección de Características

income	age	favorite color
10000	18	Black
50000	47	Blue
20000	40	Blue
30000	29	Green
20000	22	Purple

Selección de Características

income	age	favorite color
10000	18	Black
50000	47	Blue
20000	40	Blue
30000	29	Green
20000	22	Purple

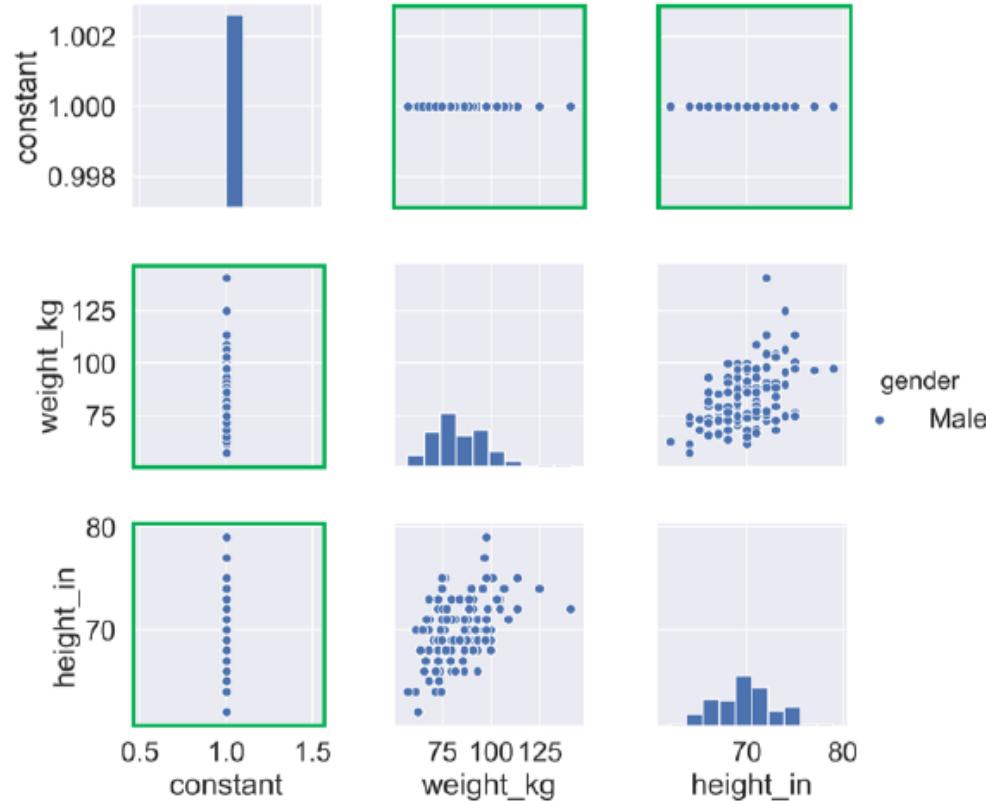


income	age
10000	18
50000	47
20000	40
30000	29
20000	22

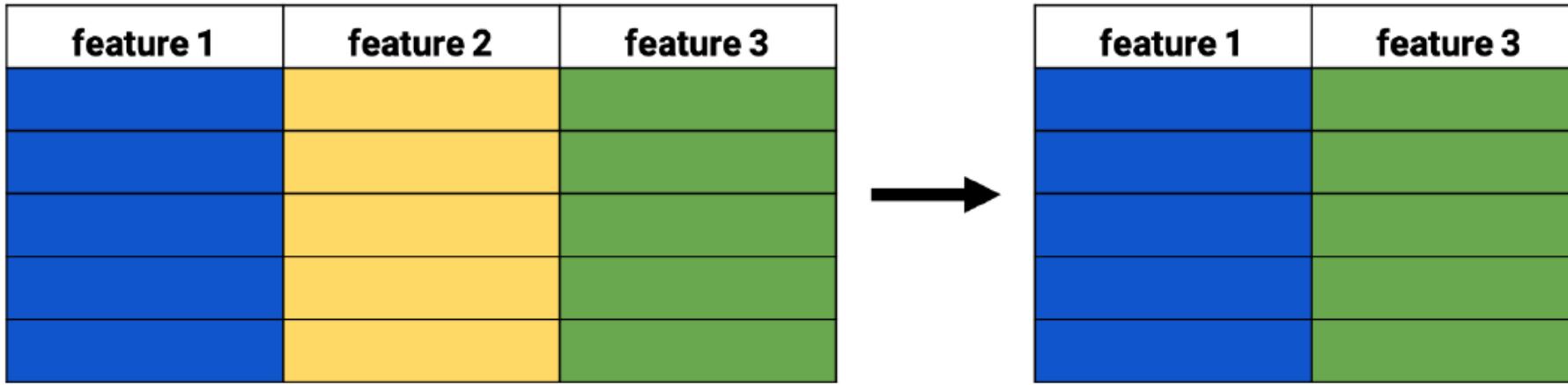
```
insurance_df.drop('favorite color', axis=1)
```

Selección de Características a partir del análisis de los datos

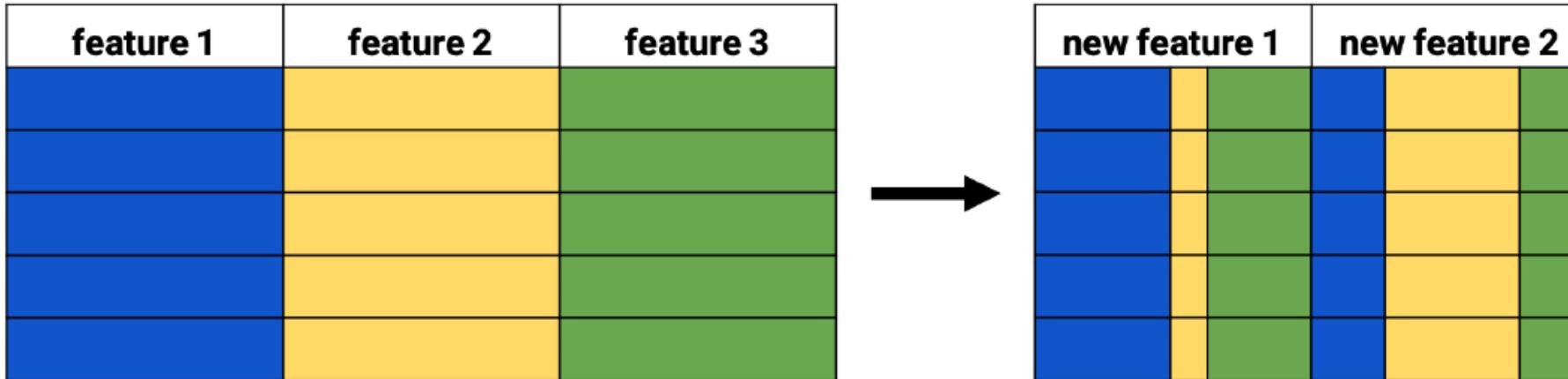
```
sns.pairplot(ansur_df, hue="gender", diag_kind='hist')
```



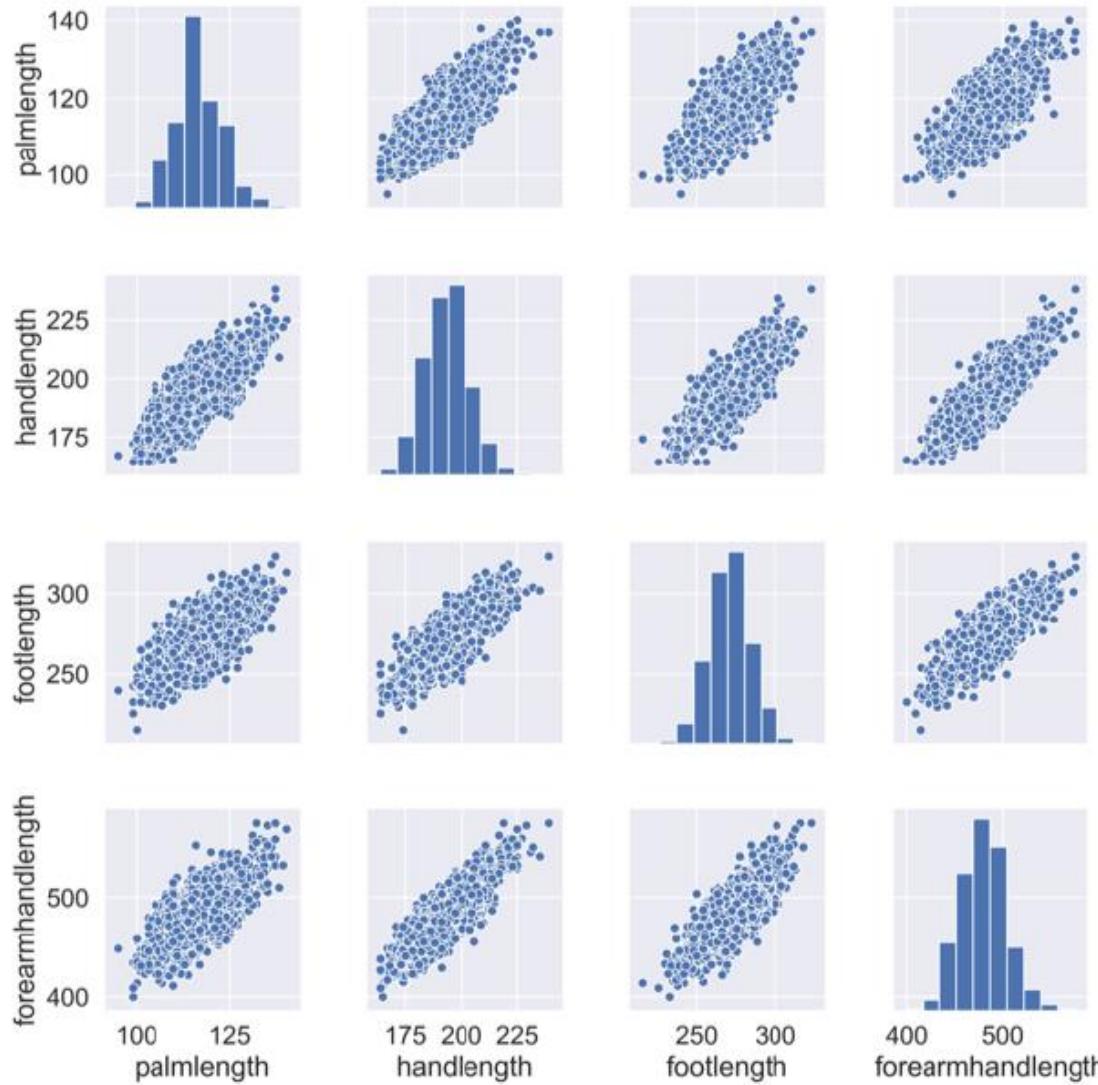
Selección de Características



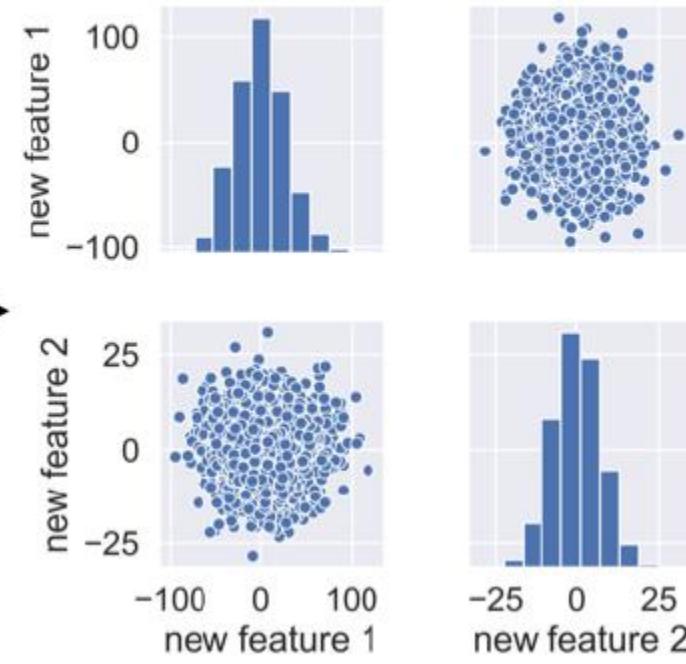
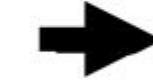
Extracción de Características



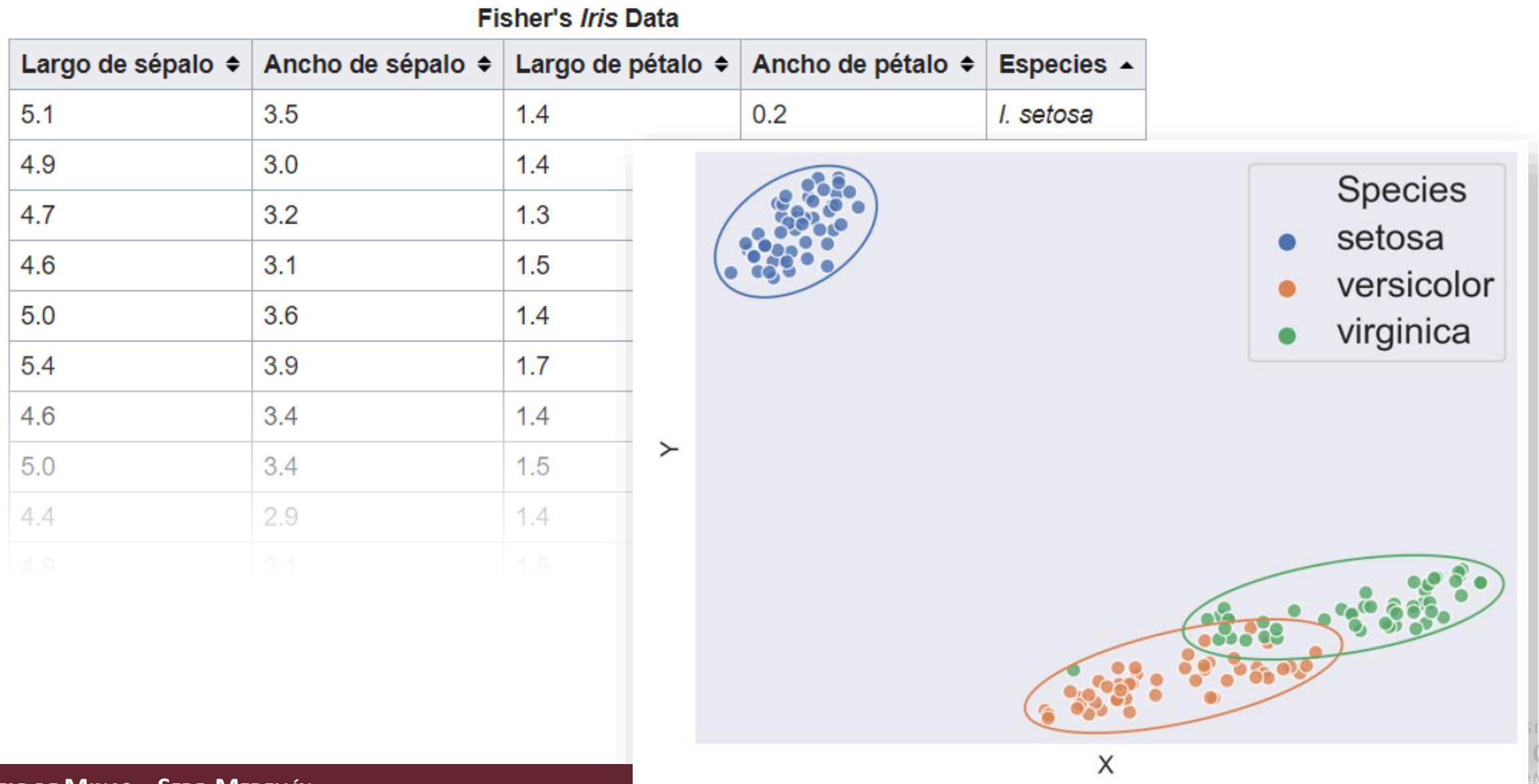
Extracción de Características a partir del análisis de los datos



Usando PCA



Como hacemos para visualizar datos multidimensionales?



t-SNE resulta ideal para visualizar datos en 2D o 3D

= *t-distributed stochastic neighbor embedding*

(*incrustación vecina estocástica distribuida en t*)

t-SNE intenta reproducir vecindades de datos de alta D en una imagen 2D o 3D mediante:

1. Definir una distribución de probabilidad sobre pares de objetos de alta D tal que los objetos "similares" tienen una alta probabilidad de ser recogidos, mientras que "diferentes" los objetos tienen una probabilidad extremadamente pequeña de ser recogidos.
2. Definir una distribución de probabilidad similar sobre los puntos en el mapa de dimensionalidad reducida.
3. Minimizando la divergencia Kullback-Leibler entre las dos distribuciones variando las ubicaciones de los puntos en el mapa de baja D, es decir
minimizar:

$$\sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

Probabilidad de que i y j sean cercanos en el espacio multidimensional
Probabilidad de que i y j sean cercanos en el espacio reducido

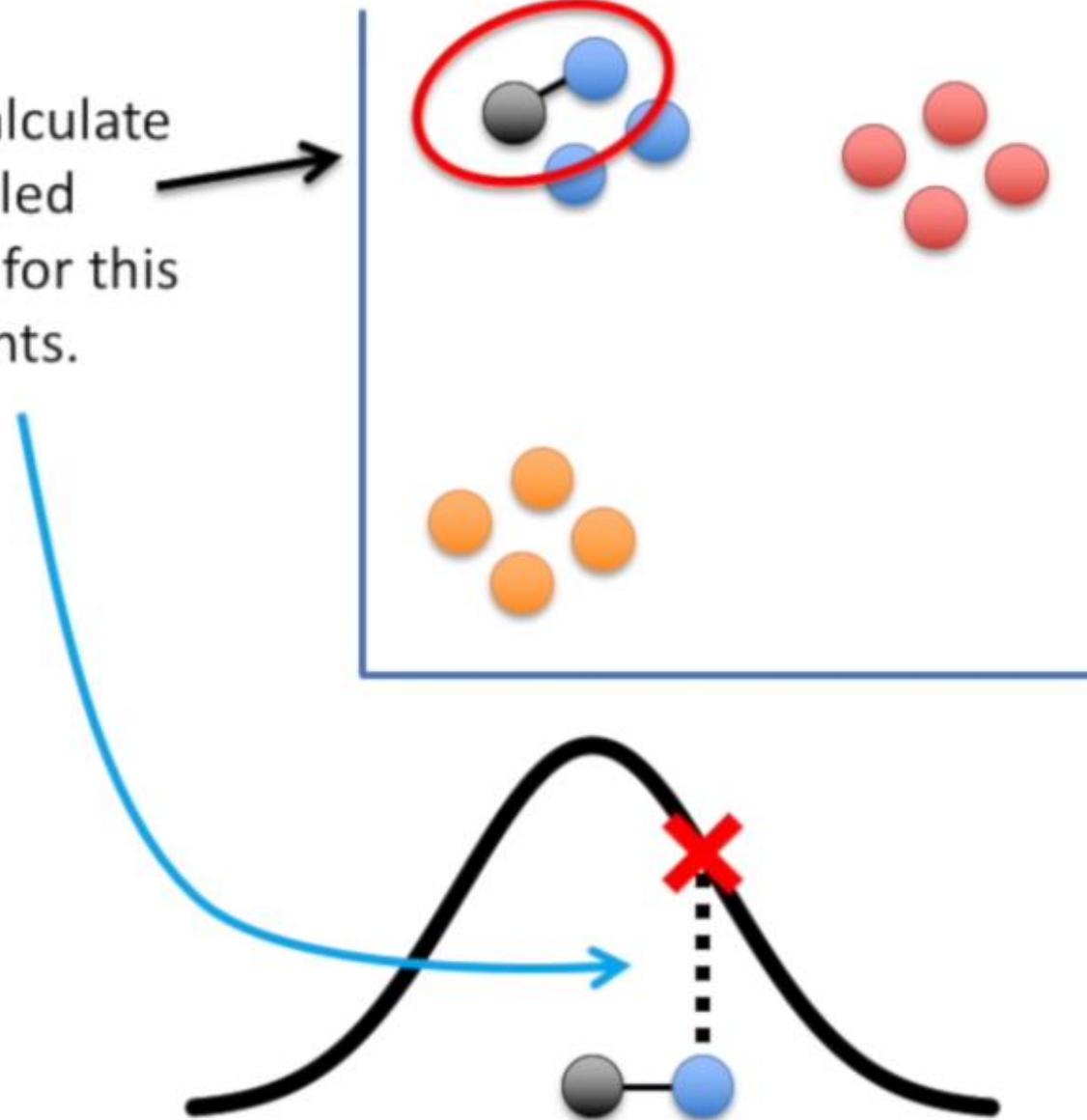
Suma sobre todos los puntos

van der Maaten & Hinton, Visualizing High-Dimensional Data Using t-SNE.
Journal of Machine Learning Research 9: 2579–2605 (Nov 2008)
en Inteligencia Artificial



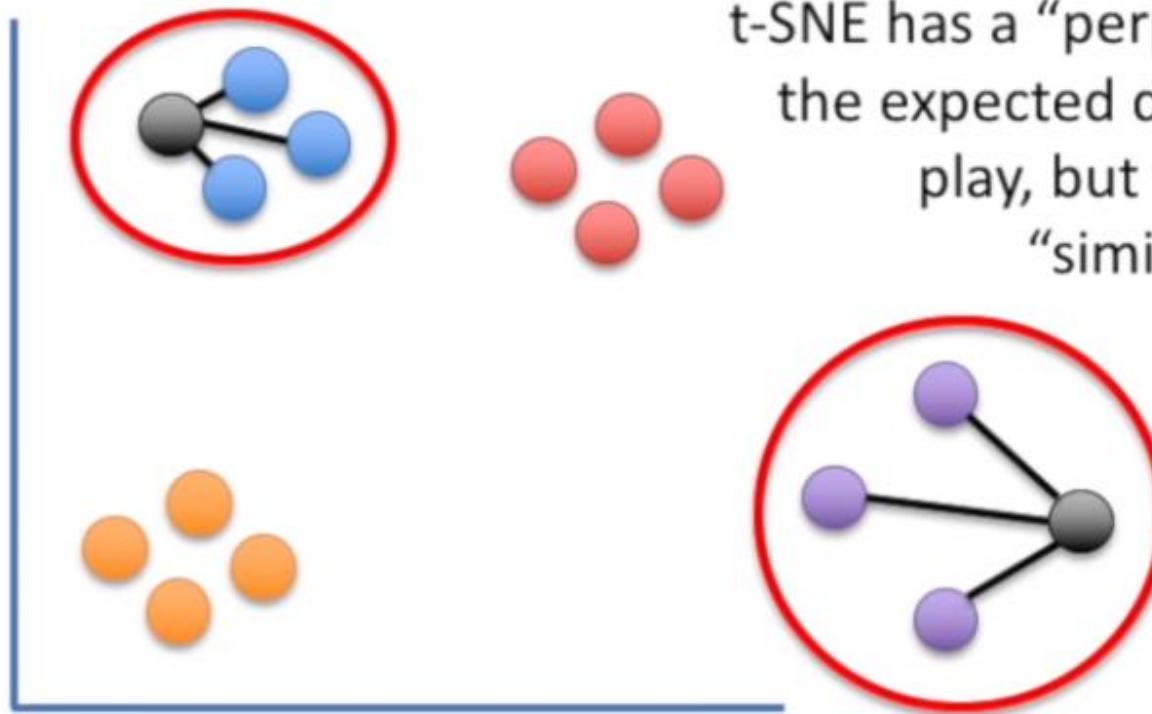
Como funciona T-SNE

Now we calculate
the “unscaled
similarity” for this
pair of points.



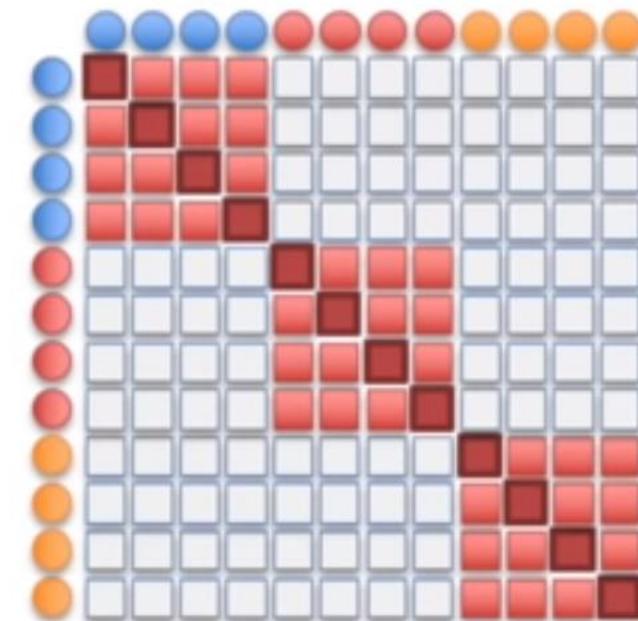
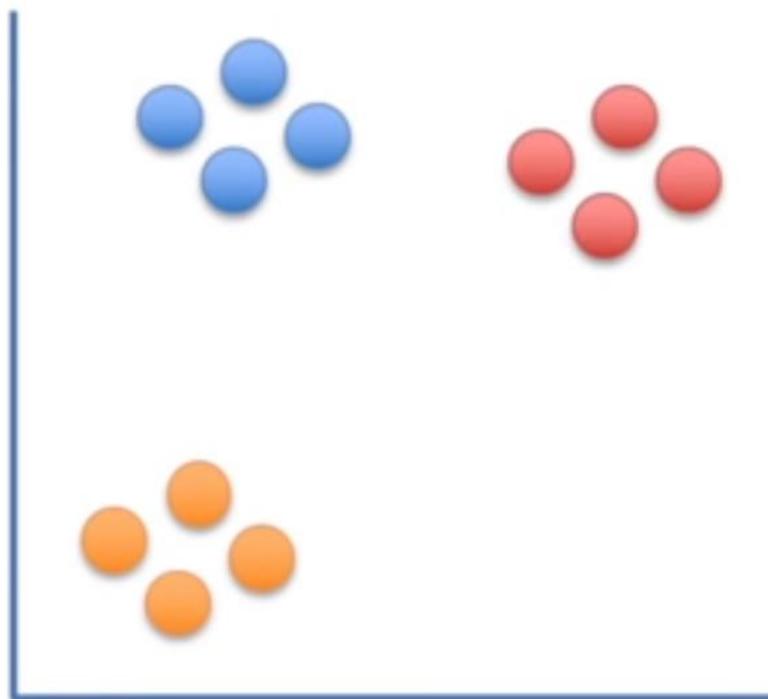
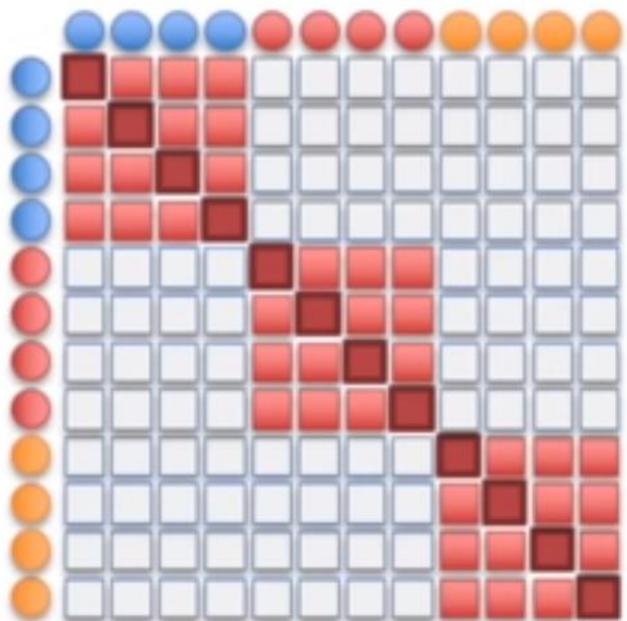
Como funciona T-SNE

The reality is a little more complicated, but only slightly.



t-SNE has a “perplexity” parameter equal to the expected density, and that comes into play, but these clusters are still more “similar” than you might expect.

Como funciona T-SNE



t-SNE moves the points a little bit at a time, and each step it chooses a direction that makes the matrix on the left more like the matrix on the right.



It uses small steps, because it's a little bit like a chess game and can't be solved all at once. Instead, it goes one move at a time.

Laboratorio 1

<https://colab.research.google.com/drive/14OhBGGNsAGOUDQn69KvY75C97pNkTchL?authuser=2>



Exploración de Datos Multidimensionales

Este Laboratorio es Basado en el Curso de Dimensionality Reduction de DATACAMP®

En este laboratorio se presentará el concepto de reducción de dimensionalidad y aprenderá cuándo y por qué esto es importante. Aprenderá la diferencia entre la selección de características y la extracción de características y aplicará ambas técnicas para la exploración de datos. Al final se termina con una lección sobre t-SNE, una poderosa técnica de extracción de características que le permitirá visualizar un conjunto de datos de alta dimensión.

Encontrar el número de dimensiones en un conjunto de datos

Se ha cargado una muestra más grande del conjunto de datos de Pokemon como el marco de datos de Pandas `pokemon_df`.

```
: # import required Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
pokemon_df=pd.read_csv("pokemon-c.csv")#reading a dataset in a dataframe using pandas
```

¿Cuántas dimensiones o columnas hay en este conjunto de datos?

```
: pokemon_df.shape
for col in pokemon_df.columns:
    print(col)
```

```
HP
Attack
Defense
Generation
Name
Type
Legendary
```

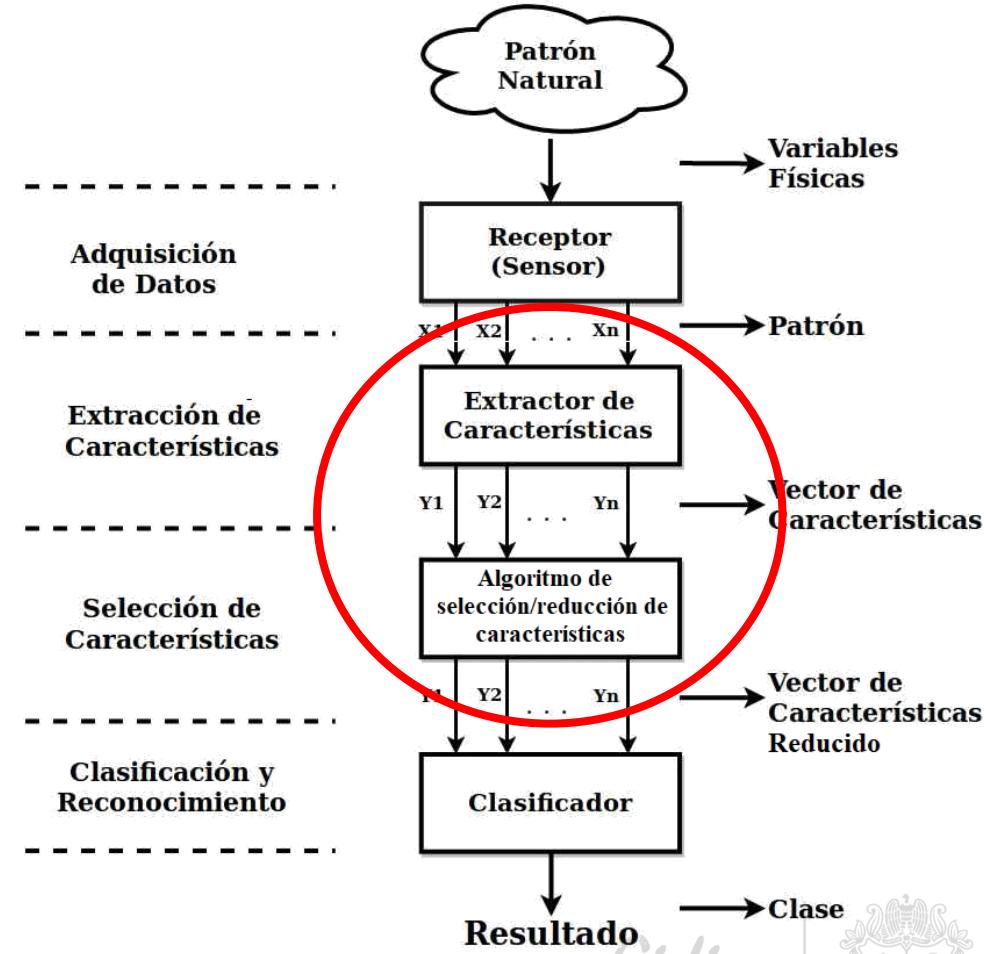
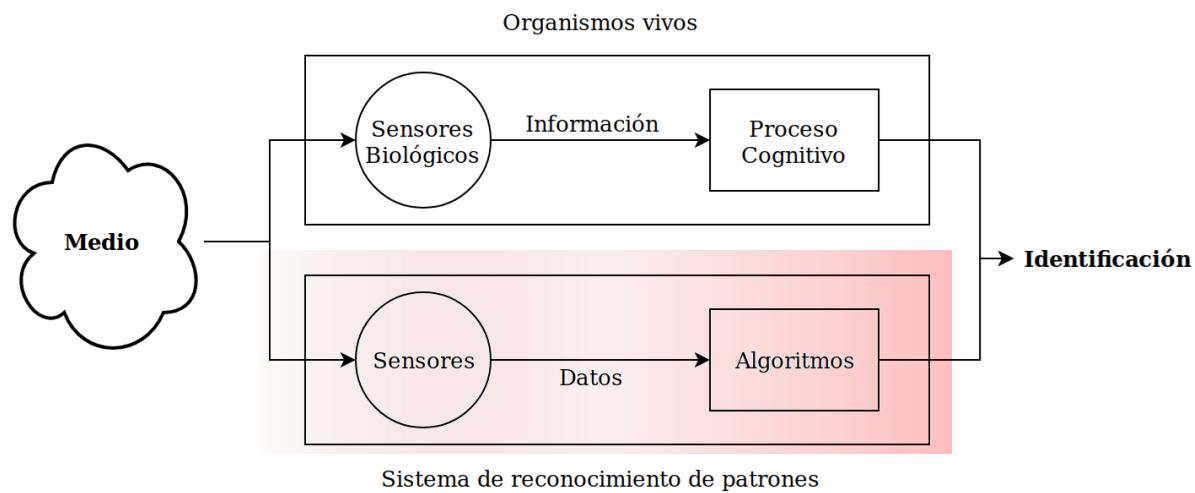
```
: pokemon_df.shape
(160, 7)
```

Eliminar características sin variación

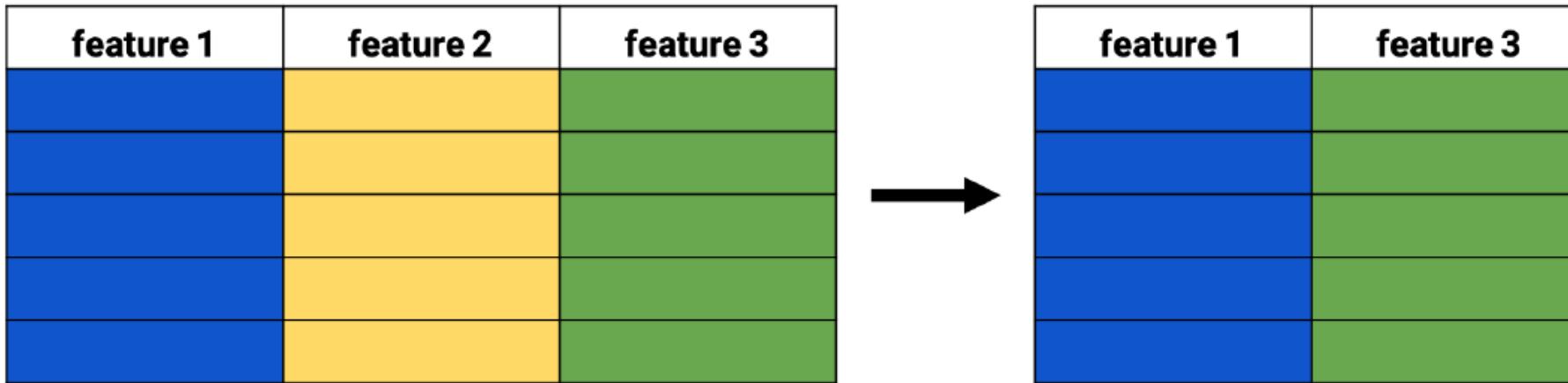
Se ha cargado una muestra del conjunto de datos de Pokemon como `pokemon_df`. Para tener una idea de qué características tienen poca variación, debe usar el Shell de IPython para calcular estadísticas de resumen en esta muestra. Luego ajuste el código para crear un conjunto de datos más pequeño y fácil de entender. Utilice el `.describe()` método para buscar la función numérica sin variación y elimine su nombre de la lista asignada a `number_cols`.

Labs-ExploracionMultiD

Etapas Sistema de Clasificación y Reconocimiento de Patrones



Selección de Características



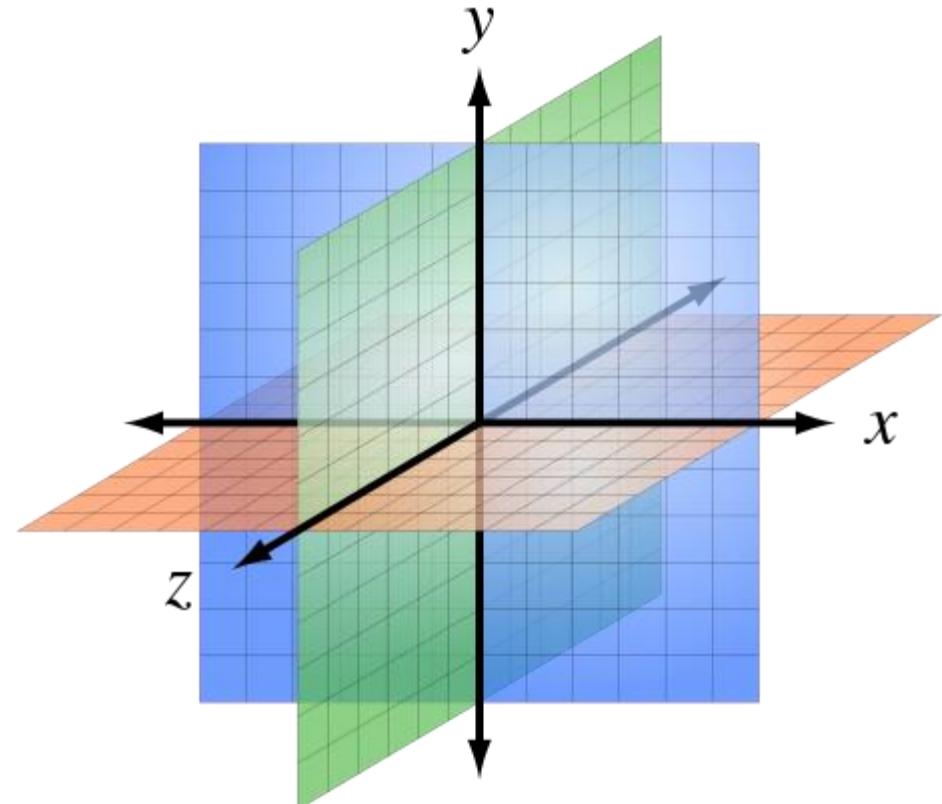
Extracción de Características



La maldición de la dimensionalidad (The Curse of Dimensionality)

Cuando el número de características es muy grande en relación con el número de observaciones en su conjunto de datos, ciertos algoritmos luchan por entrenar modelos efectivos.

Esto se llama la "**Maldición de la dimensionalidad**", y es especialmente relevante para los algoritmos de agrupamiento que se basan en cálculos de distancia.



La maldición de la dimensionalidad (Analogía)

“Digamos que tiene una línea recta de 100 metros de largo y dejó caer una moneda en algún lugar. No sería muy difícil de encontrar. Caminas a lo largo de la línea y te lleva dos minutos.

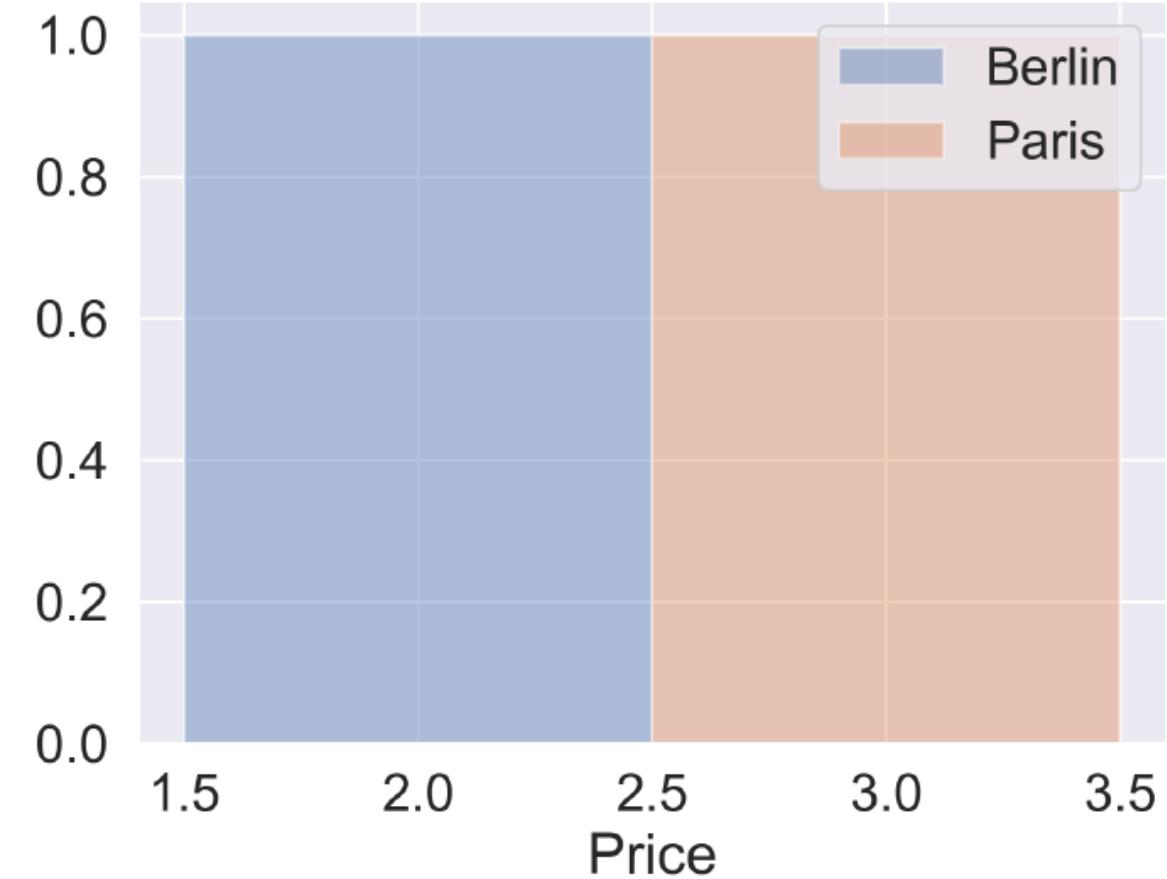
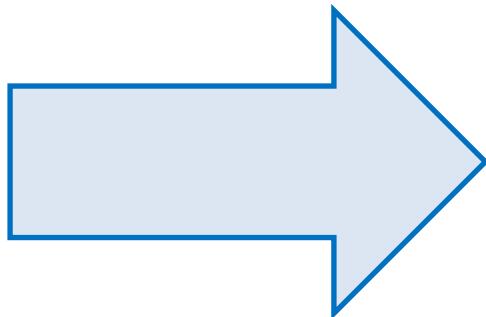
Ahora supongamos que tiene un cuadrado de 100 metros a cada lado y dejó caer un centavo en algún lugar. Sería bastante difícil, como buscar en dos campos de fútbol unidos. Podría llevar días.

Ahora un cubo de 100 metros de arista. Es como buscar en un edificio de 30 pisos del tamaño de un estadio de fútbol. Ugh

La dificultad de buscar en el espacio se vuelve mucho más difícil a medida que tienes más dimensiones.”

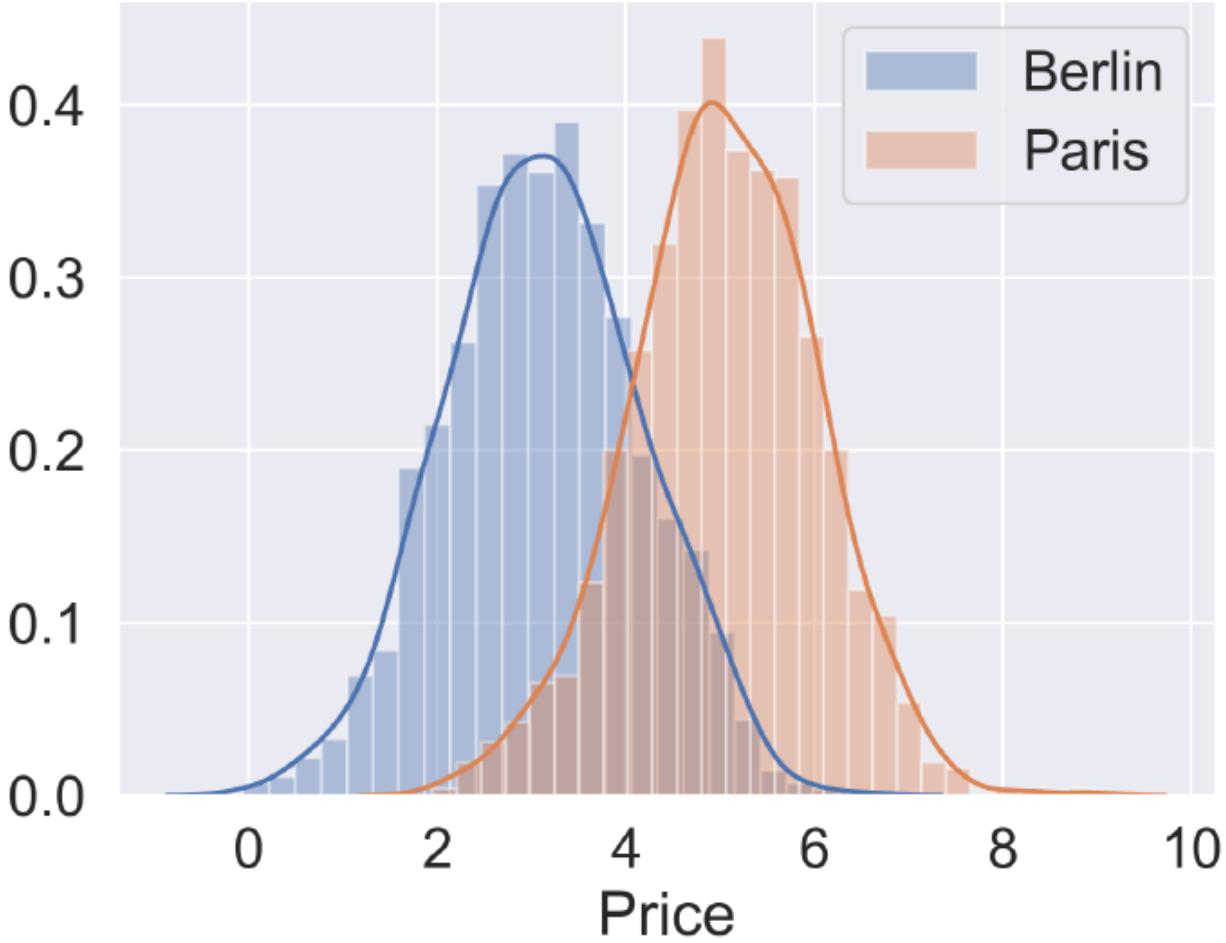
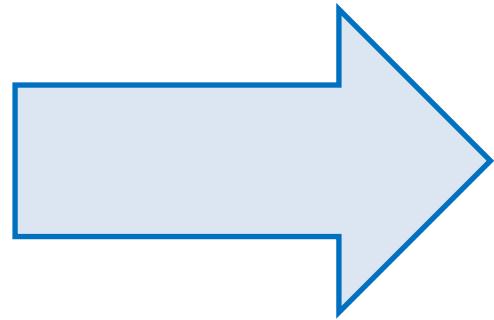
La maldición de la dimensionalidad (Ejemplo)

City	Price
Berlin	2
Paris	3



La maldición de la dimensionalidad (Ejemplo)

City	Price
Berlin	2.0
Berlin	3.1
Berlin	4.3
Paris	3.0
Paris	5.2
...	...



La maldición de la dimensionalidad

Construimos un Clasificador

Separate the feature we want to predict from the ones to train the model on.

```
y = house_df['City']  
  
X = house_df.drop('City', axis=1)
```

Perform a 70% train and 30% test data split

```
from sklearn.model_selection import train_test_split  
  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

La maldición de la dimensionalidad

Construimos un Clasificador

Create a Support Vector Machine Classifier and fit to training data

```
from sklearn.svm import SVC  
  
svc = SVC()  
  
svc.fit(X_train, y_train)
```

La maldición de la dimensionalidad

Evaluamos la calidad de la Predicción

```
from sklearn.metrics import accuracy_score  
  
print(accuracy_score(y_test, svc.predict(X_test)))
```

0.826

```
print(accuracy_score(y_train, svc.predict(X_train)))
```

0.832

La maldición de la dimensionalidad

Incrementamos dimensiones (Características)

City	Price	n_floors	n_bathroom	surface_m2
Berlin	2.0	1	1	190
Berlin	3.1	2	1	187
Berlin	4.3	2	2	240
Paris	3.0	2	1	170
Paris	5.2	2	2	290
...



La maldición de la dimensionalidad

Se debe asegurar que existe una cantidad significativa de ejemplos por cada dimensión adicional que se maneja en nuestro problema.

De otra manera puede haber **sobreajuste (Overfitting)**. No se garantiza la generalización.

Por esto es importante identificar de que manera podemos eliminar características (**feature selection**) que poco aporten en nuestra búsqueda de nuestro modelo de reconocimiento de patrones.

Características (Dimensiones) con valores faltantes o Poca Varianza

En algunos casos debemos identificar estrategias para completar datos faltantes. (Usar el valor medio o por ejemplo (**mean**) - El Valor más frecuente (**mode**))

Para los valores que presenten poca varianza podemos eliminarlos. (Para esto es importante **normalizar la varianza** en todo el dataset)

También es muy útil identificar el valor de **correlación** en los datos.

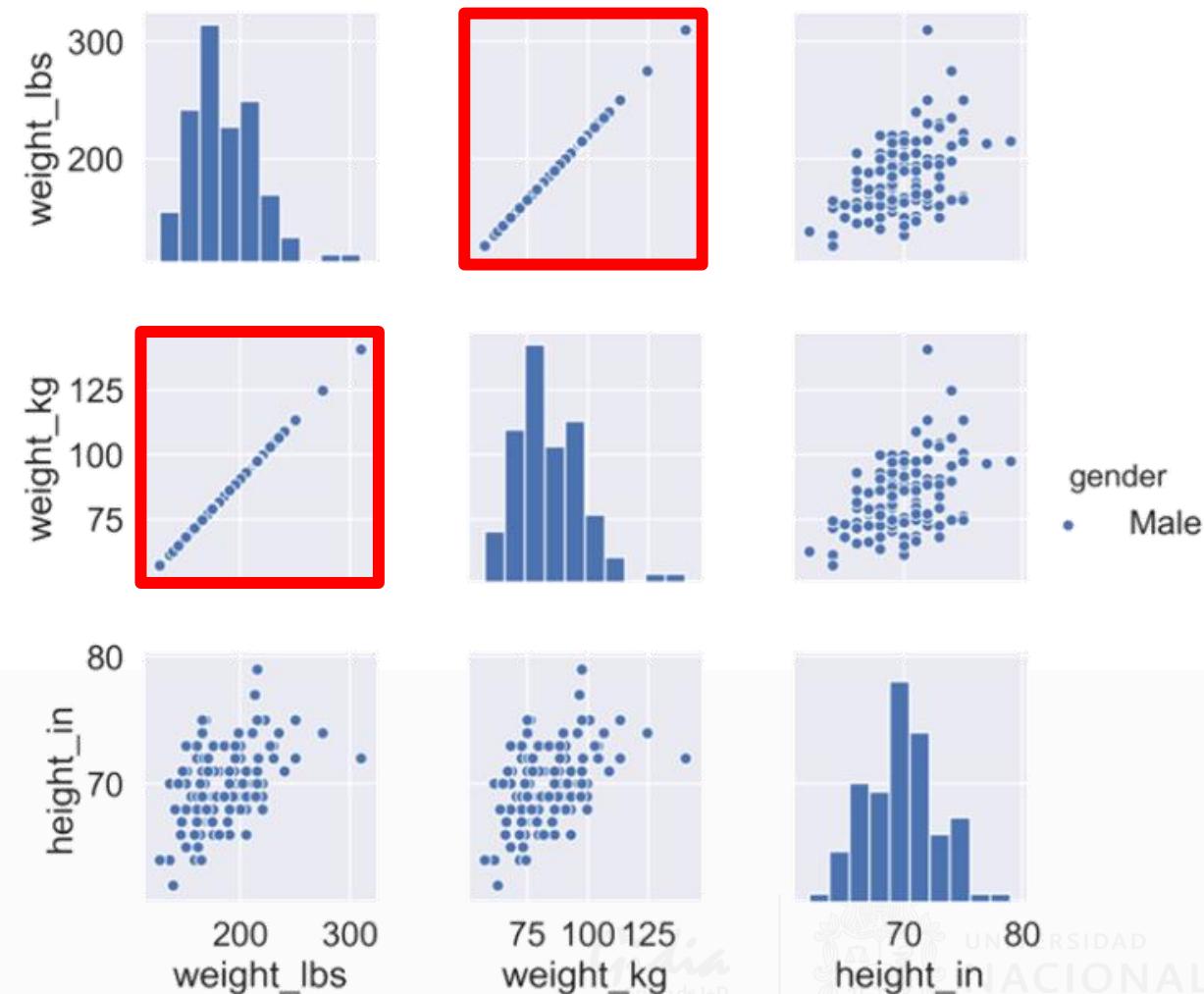
Correlación de Valores

Correlación en parejas

```
sns.pairplot(ansur, hue="gender")
```

$$Cov(X, Y) = \frac{\sum_1^n (x_i - \bar{x})(y_i - \bar{y})}{n}$$

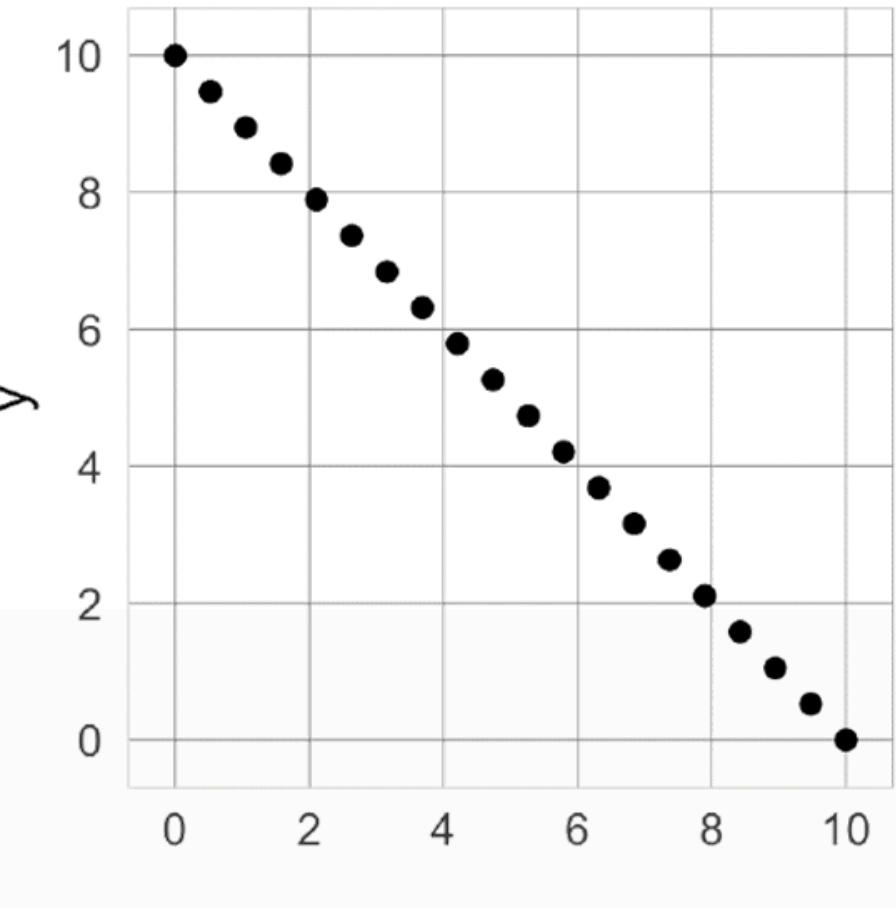
$$\rho_{xy} = \frac{Cov_{xy}}{\sigma_x \sigma_y}$$



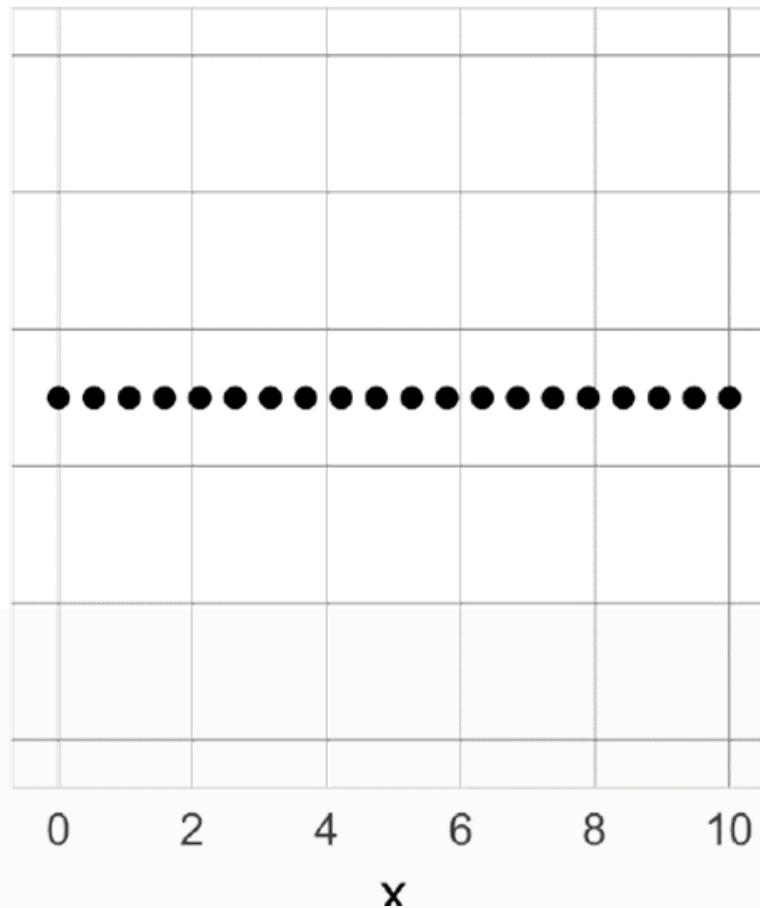
Correlación de Valores

Coeficiente de correlación

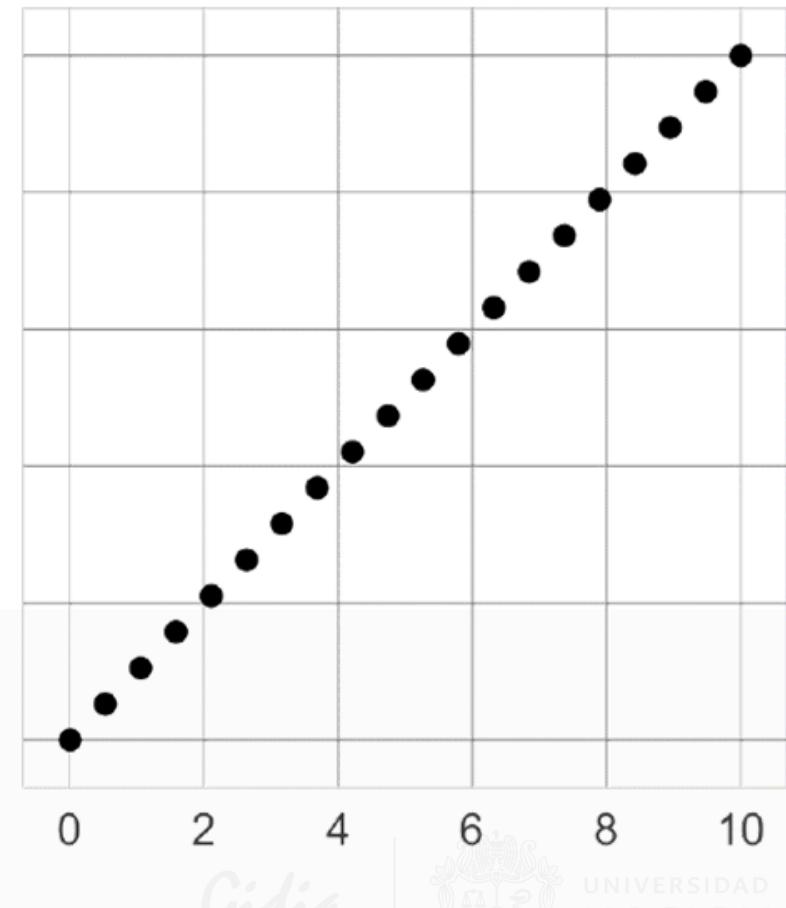
$r = -1$



$r = 0$



$r = 1$



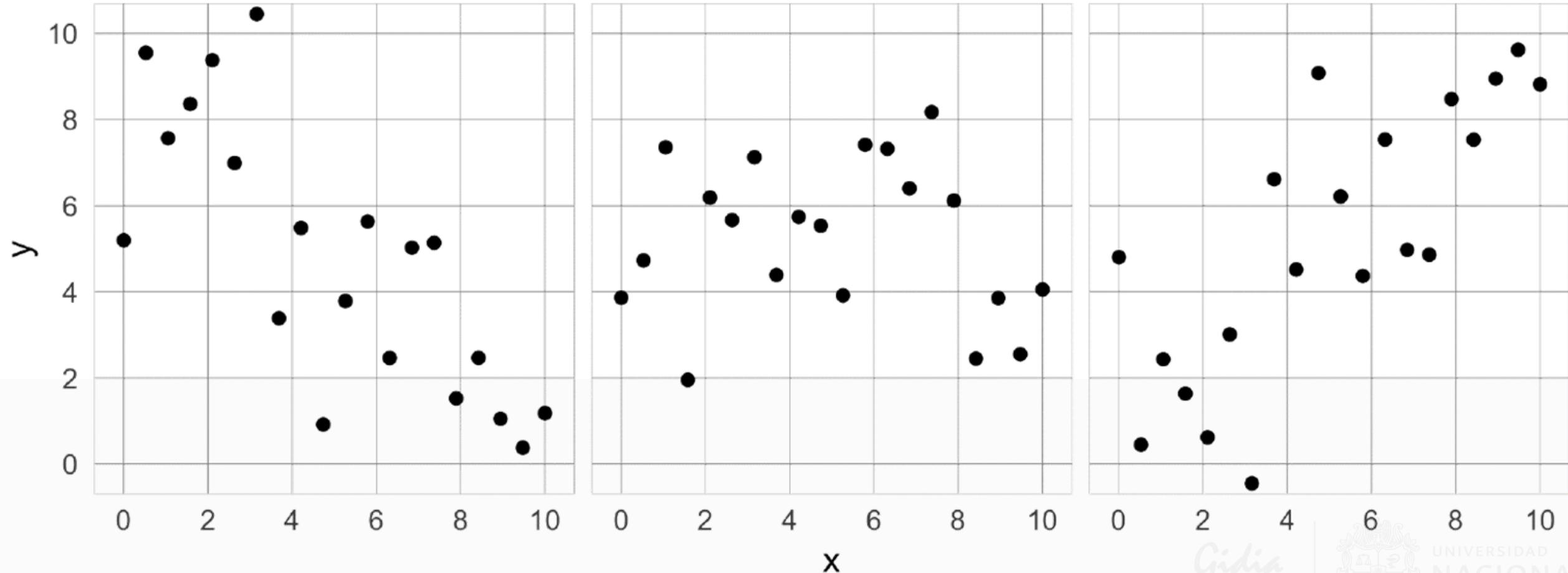
Correlación de Valores

Coeficiente de correlación

$r = -0.88$

$r = 0.05$

$r = 0.88$



Correlación de Valores

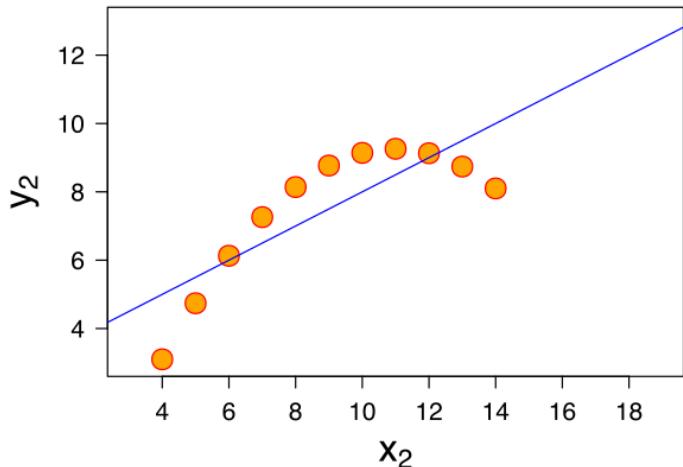
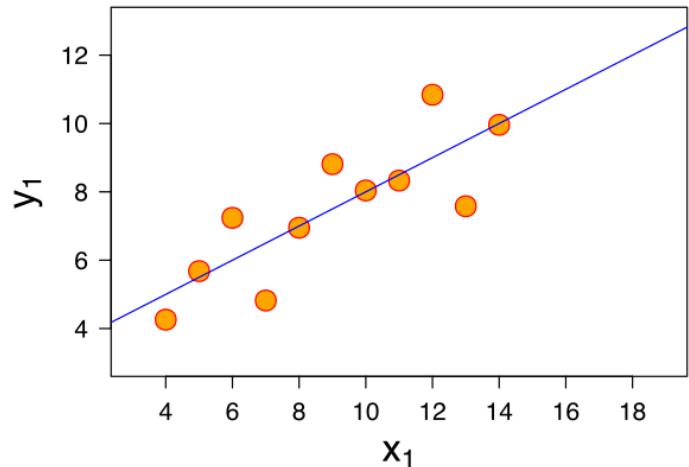
Matriz de Correlación

```
weights_df.corr()
```

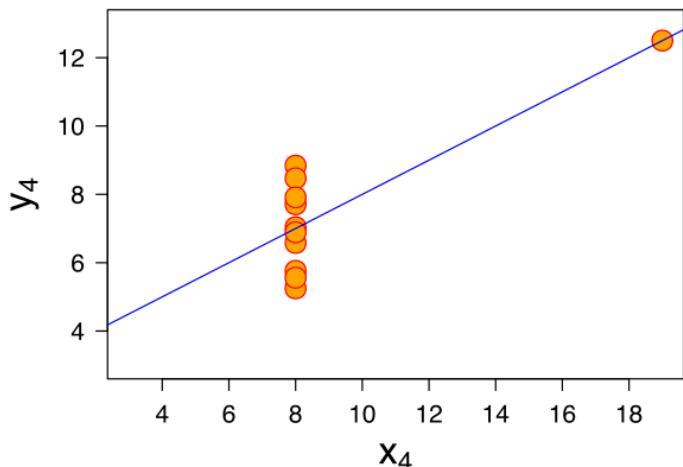
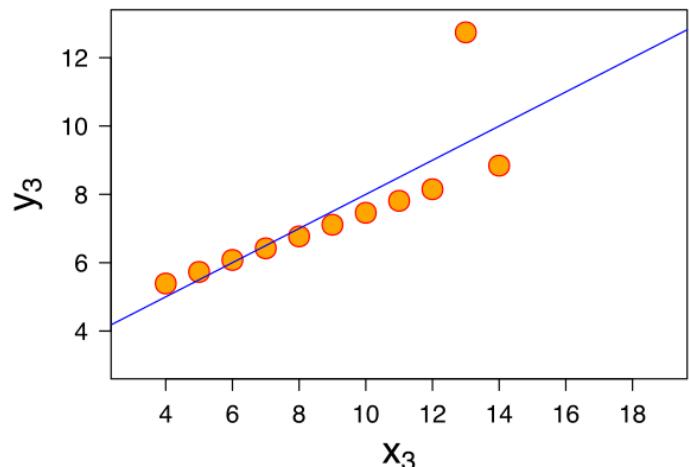
	weight_lbs	weight_kg	height_in
weight_lbs	1.00	1.00	0.47
weight_kg	1.00	1.00	0.47
height_in	0.47	0.47	1.00

Correlación de Valores

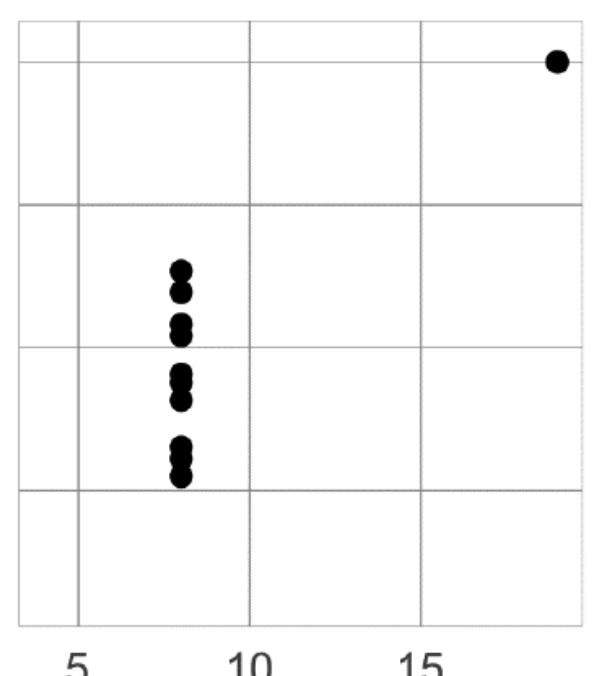
Problemas con la Correlación



= 0.82



$r = 0.82$

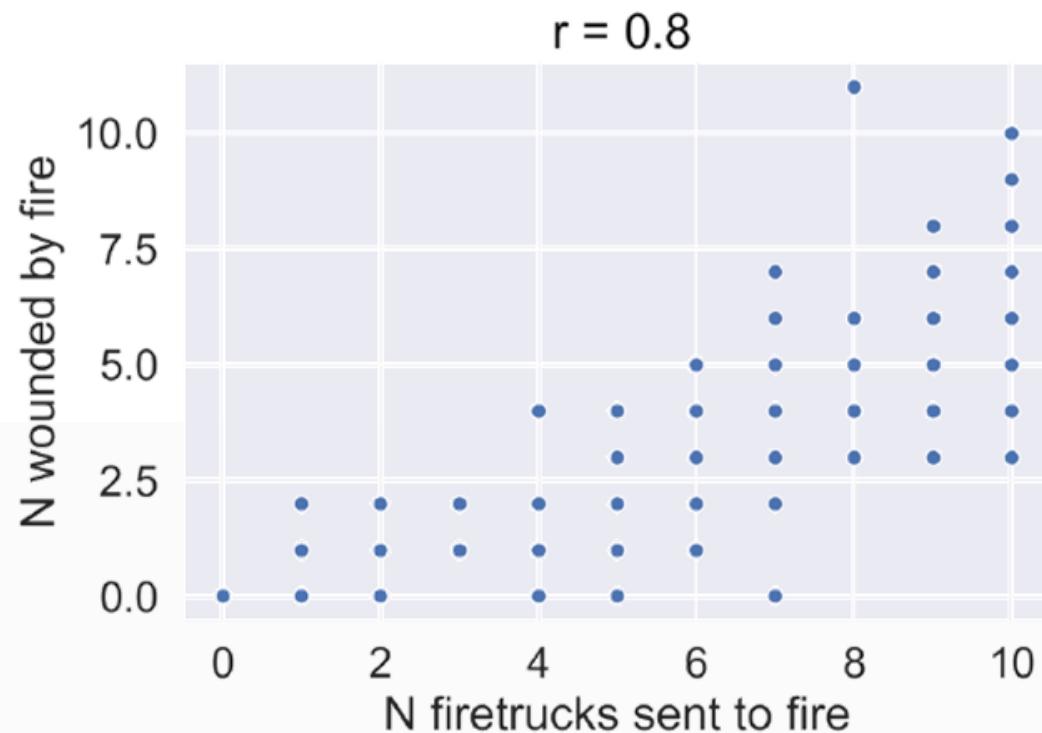


Cuarteto de Anscombe

Correlación de Valores

Problemas con la Correlación

```
sns.scatterplot(x="N firetrucks sent to fire",  
y="N wounded by fire", data=fire_df)
```



Causación

Laboratorios Restantes

<https://drive.google.com/drive/u/2/folders/1nNtp03GXdH8FGcVVFXEpPXpt624CxrJ0>

Reemplazo de valores faltantes y Filtrado de Baja Varianza

Considere una variable en nuestro conjunto de datos donde todas las observaciones tienen el mismo valor, digamos 1. Si usamos esta variable, ¿cree que puede mejorar el modelo que construiremos? La respuesta es no, porque esta variable tendrá cero varianza.

Entonces, necesitamos calcular la varianza de cada variable que se nos da. Luego, eliminar las variables que tienen una varianza baja en comparación con otras variables en nuestro conjunto de datos. La razón para hacerlo, como mencionamos anteriormente, es que las variables con una varianza baja no afectarán la variable objetivo.

```
# import required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
train=pd.read_csv("Train_data.csv")
# To show rows and columns
train.shape
```

(8523, 12)

Primero, identificamos los datos faltantes en cada dimensión

```
train.isna().sum()
```

Item_Identifier	0
Item_Weight	1463
Item_Fat_Content	0
Item_Visibility	0
Item_Type	0
Item_MRP	0
Outlet_Identifier	0
Outlet_Establishment_Year	0
Outlet_Size	2410
Outlet_Location_Type	0
Outlet_Type	0
Item_Outlet_Sales	0
dtype:	int64

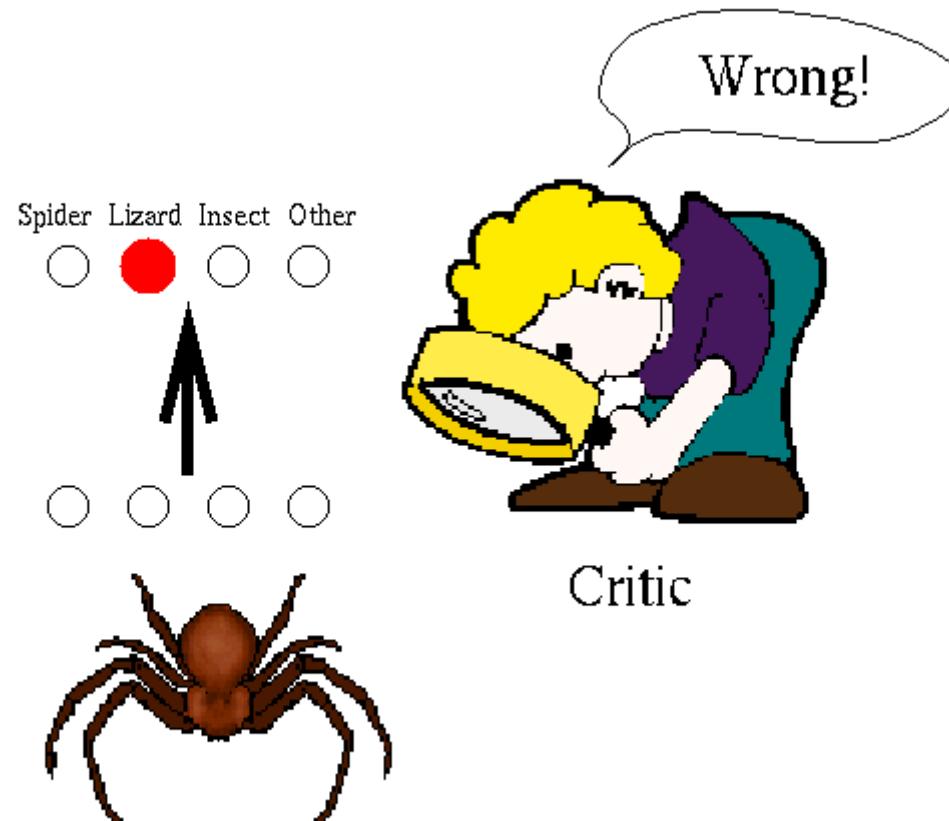
Reemplazaremos los valores faltantes en la columna Item_Weight usando el valor medio de las observaciones conocidas de Item_Weight. Para la columna Outlet_Size, usaremos el modo de los valores Outlet_Size conocidos para reemplazar los valores faltantes:

```
train['Outlet_Size'].mode()[0]
'Medium'
```

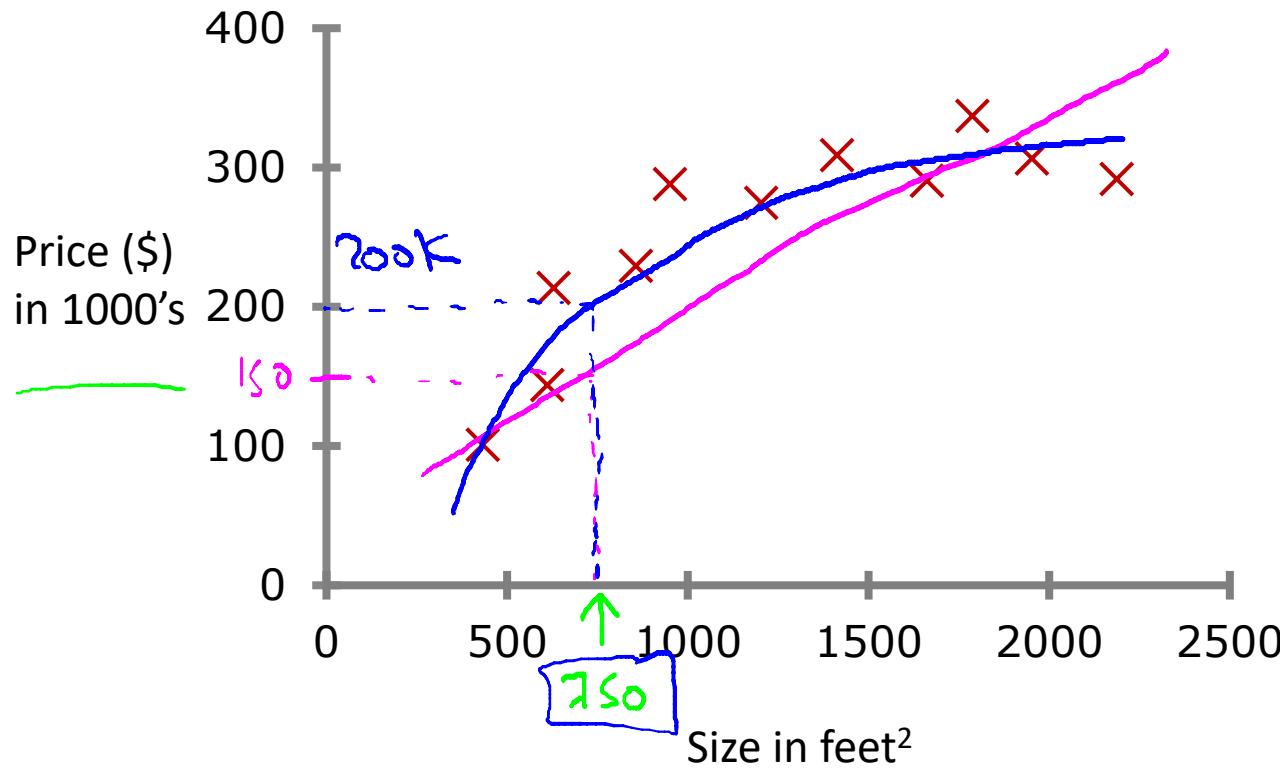
Basic_labs



Supervised Learning



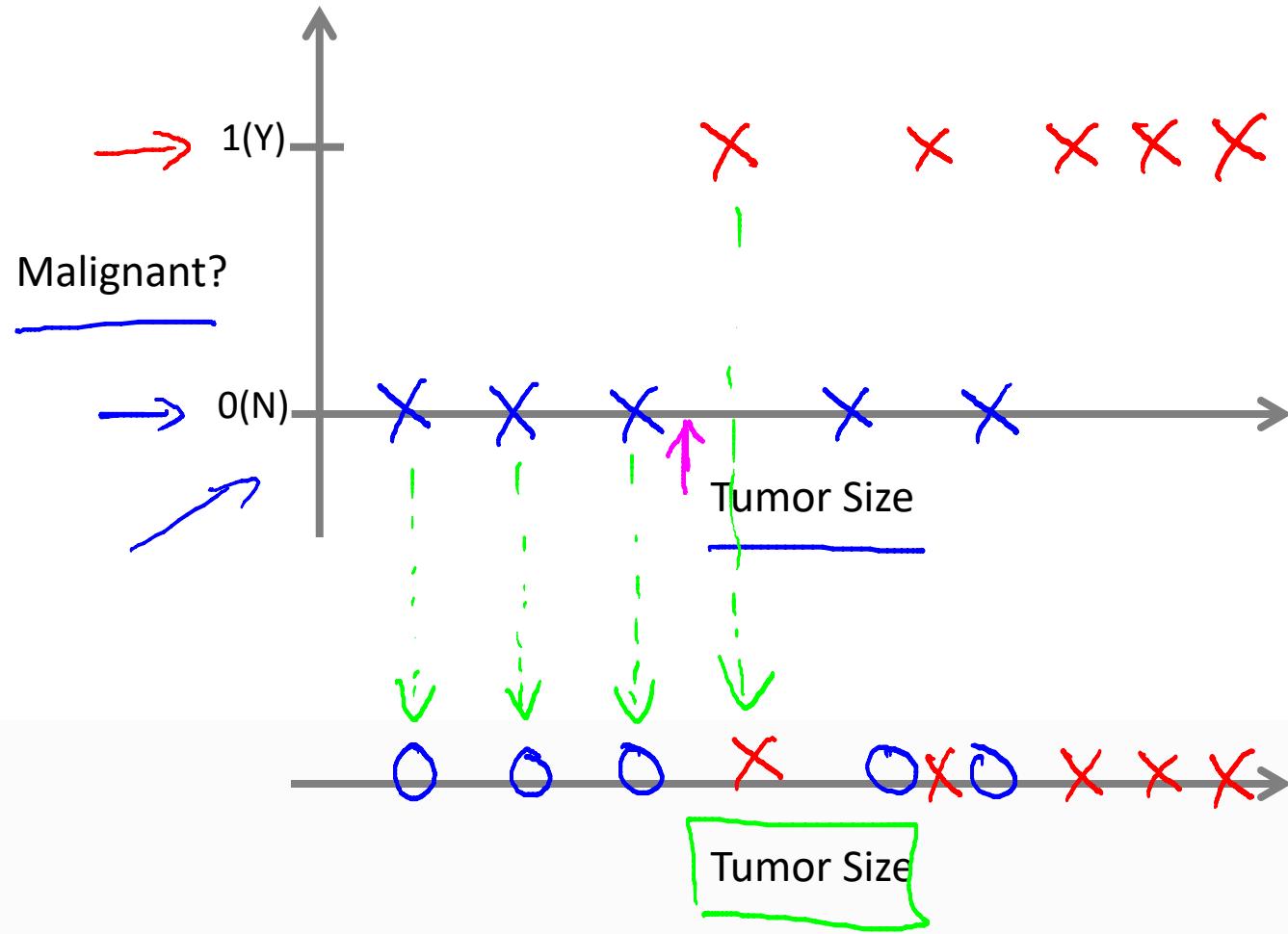
House Price prediction



Supervised Learning
"right answers" given

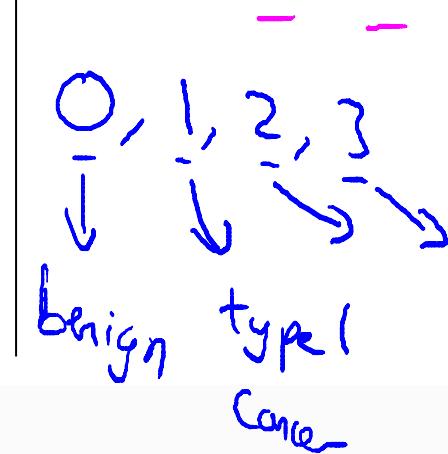
Regression: Predict continuous valued output (price)

Breast cancer (malignant, benign)

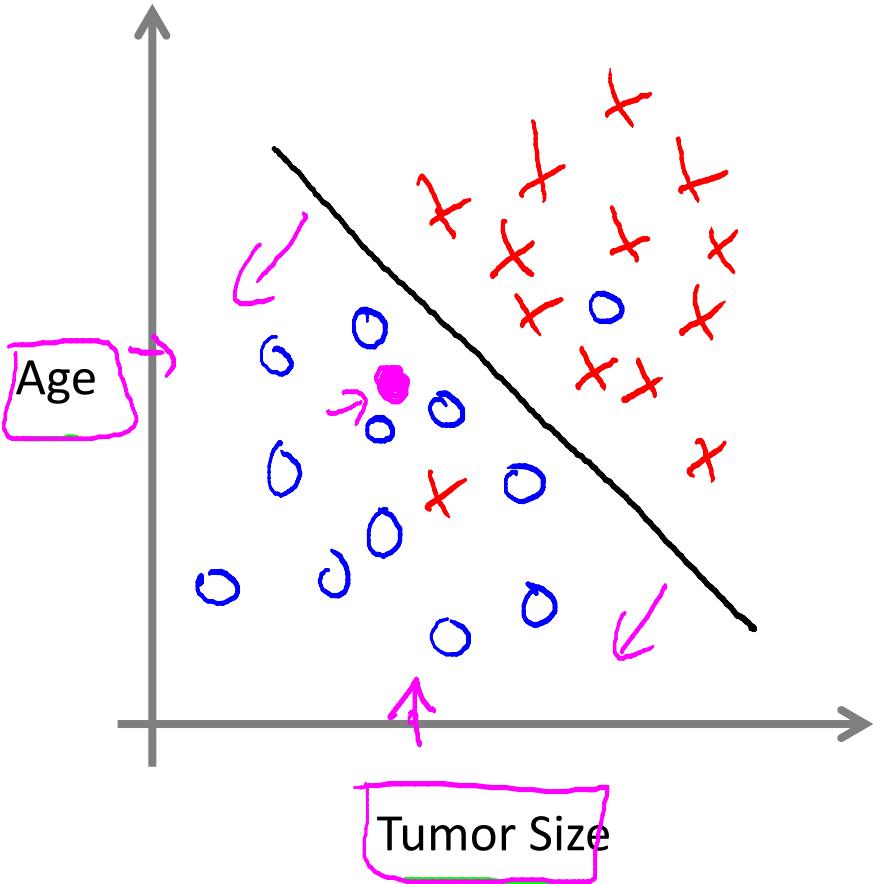


Classification

Discrete valued output
(0 or 1)



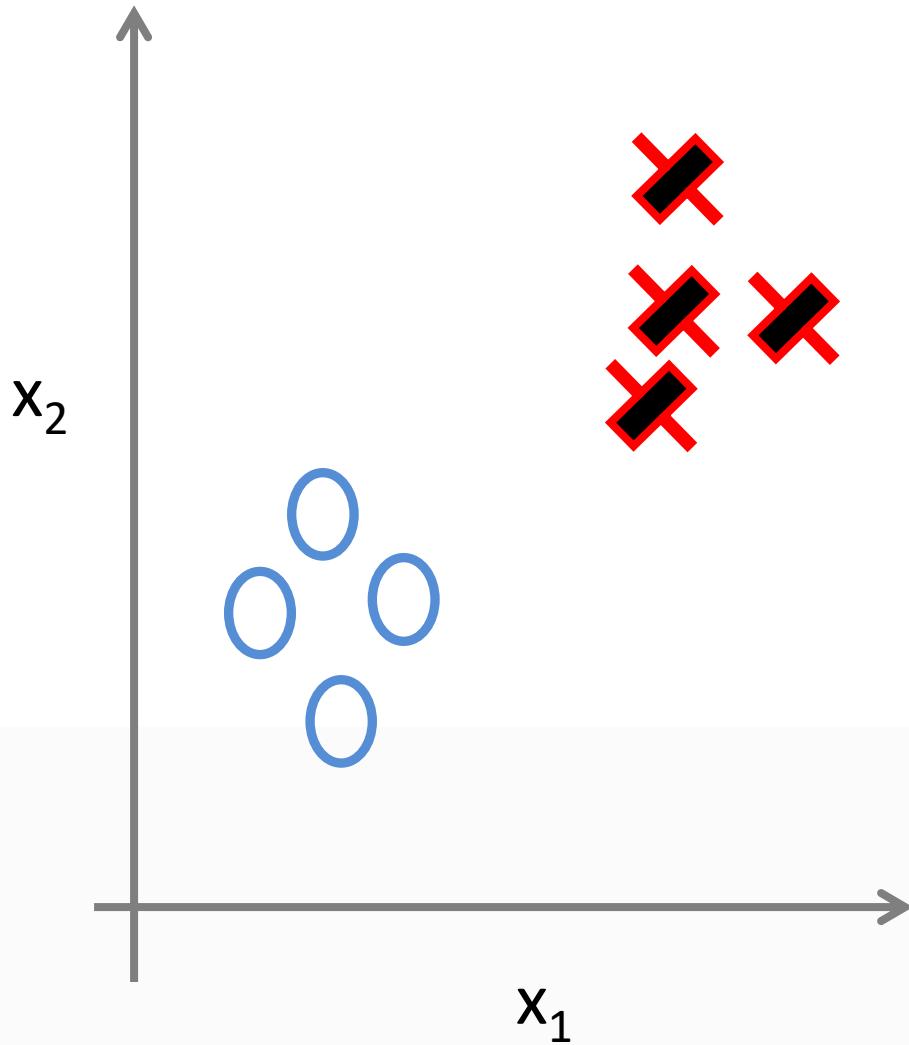
What if I have more features?



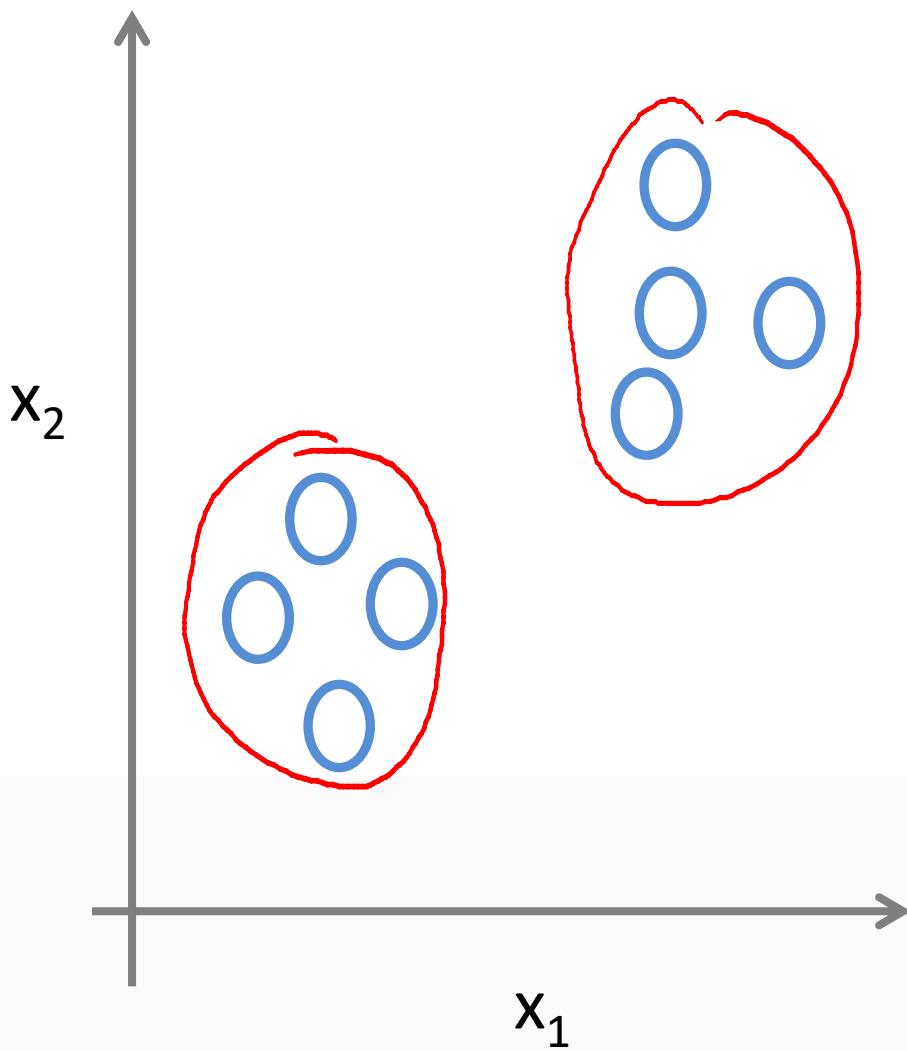
- Clump Thickness
- Uniformity of Cell Size
- Uniformity of Cell Shape

...

In Supervised Learning...



But in unsupervised learning

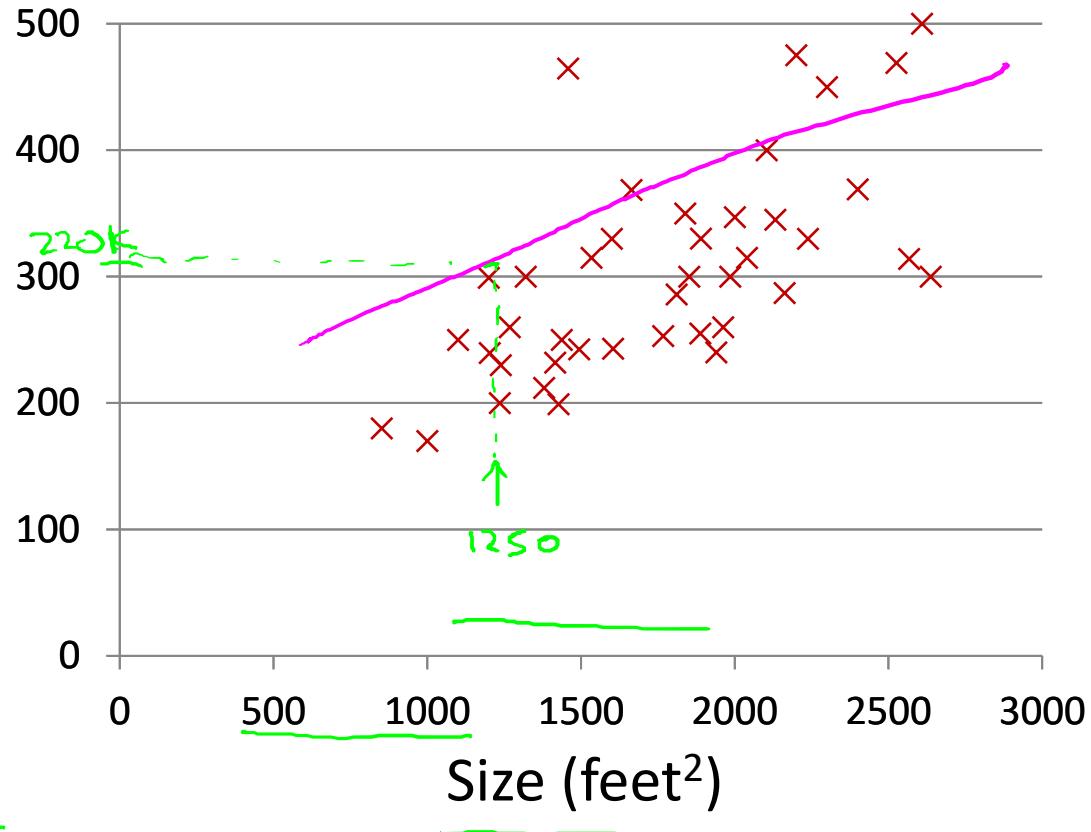


Linear regression with one variable

Model representation

Housing Prices (Portland, OR)

Price
(in 1000s
of dollars)



Supervised Learning

Given the “right answer” for each example in the data.

Regression Problem

Predict real-valued output

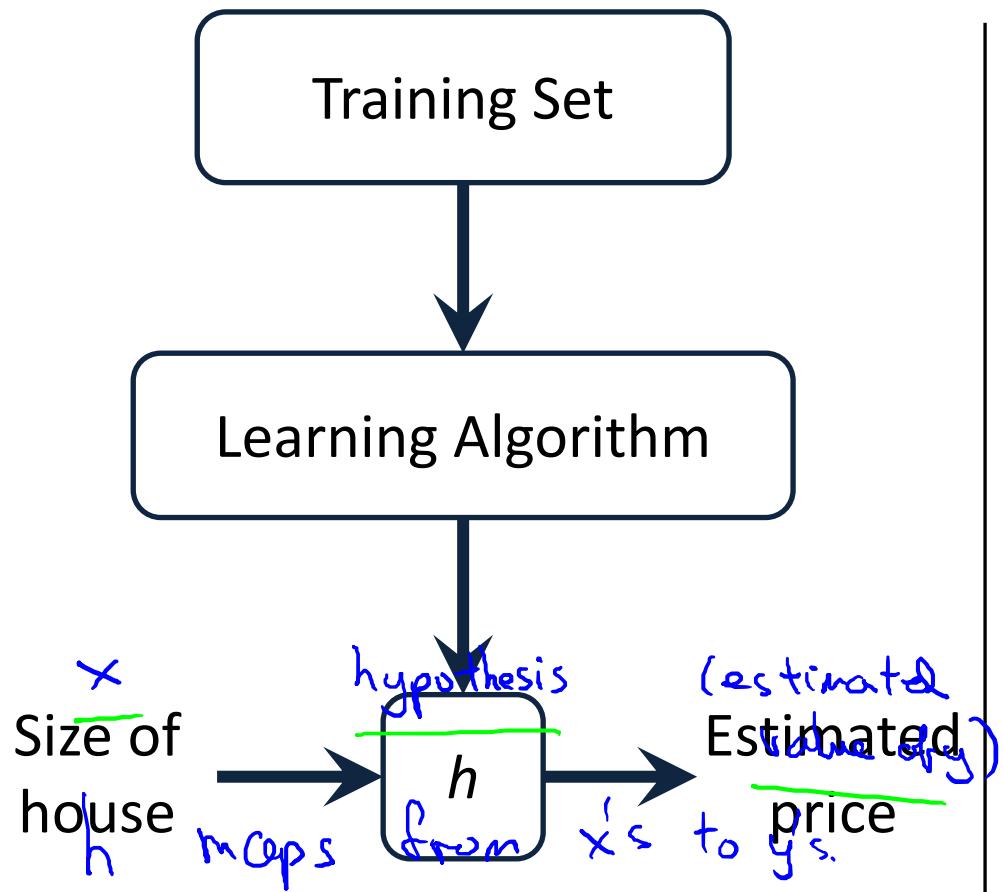
Training set of housing prices (Portland, OR)

Size in feet ²	Price (\$) in 1000's (y)
(x) → 2104	460
→ 1416	232
1534	315
852	178
...	...

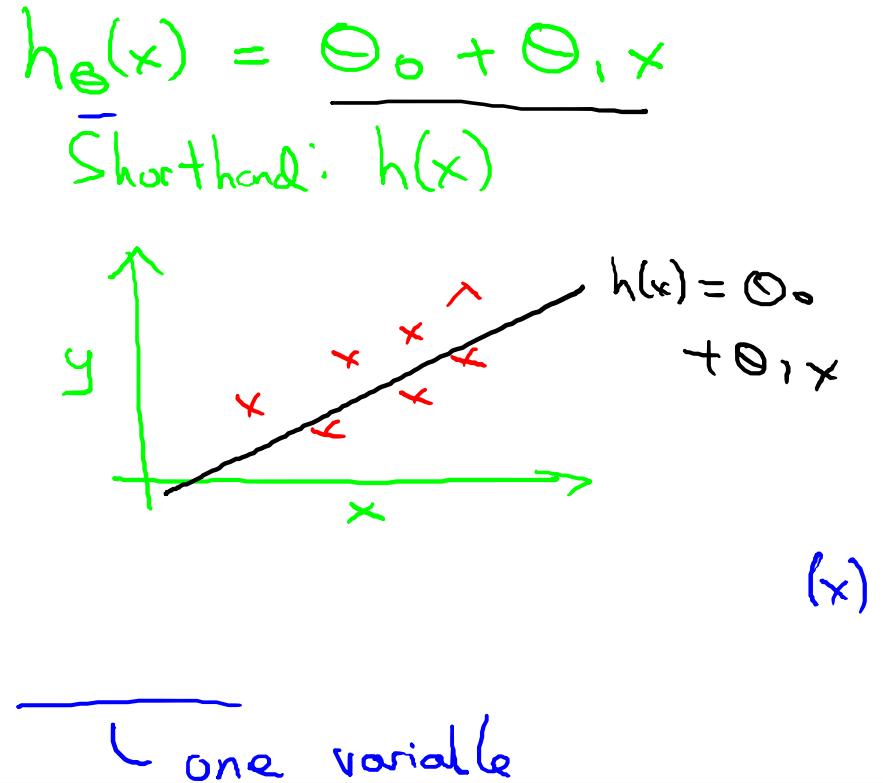
Notation:

- m = Number of training examples
- x 's = "input" variable / ~~features~~
- y 's = "output" variable / "target" variable
- (x, y) - one training example
- $(x^{(i)}, y^{(i)})$ - ith training example

$$\left\{ \begin{array}{l} x^{(1)} = 2104 \\ x^{(2)} = 1416 \\ y^{(1)} = 460 \end{array} \right.$$



How do we represent h ?



Linear regression with one variable.
Univariate linear regression.

Linear regression with one variable

Cost function

Training Set

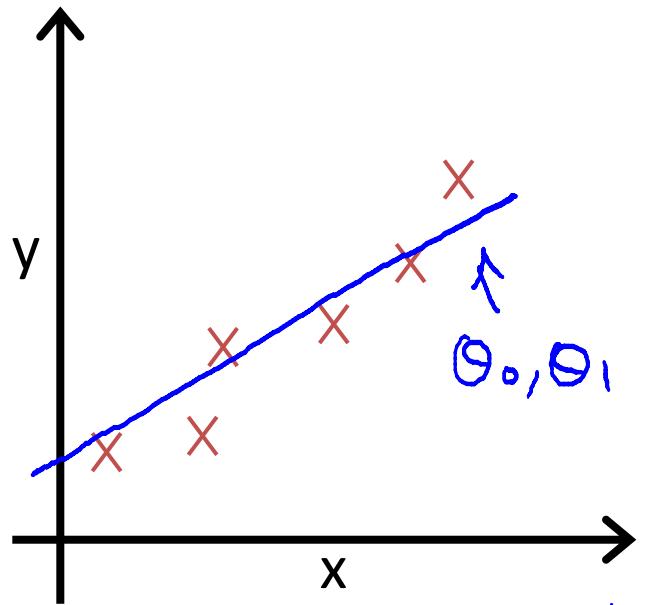
Size in feet ² (x)	Price (\$) in 1000's (y)
2104	460
1416	232
1534	315
852	178
...	...

$m = 47$

Hypothesis: $\underline{h_{\theta}(x) = \theta_0 + \theta_1 x}$

θ_i 's: Parameters

How to choose θ_i 's ?



#training examples

$$\underset{\theta_0, \theta_1}{\text{minimize}} \quad \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2$$

\uparrow

$$h_\theta(x^{(i)}) = \theta_0 + \theta_1 x^{(i)}$$

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2$$

Idea: Choose θ_0, θ_1 so that
 $h_\theta(x)$ is close to y for our
training examples

(x, y)

x, y

minimize $J(\theta_0, \theta_1)$

θ_0, θ_1

Cost function

Squared error function



Hypothesis:

$$h_{\theta}(x) = \theta_0 + \theta_1 x$$

Parameters:

$$\underline{\theta_0, \theta_1}$$

Cost Function:

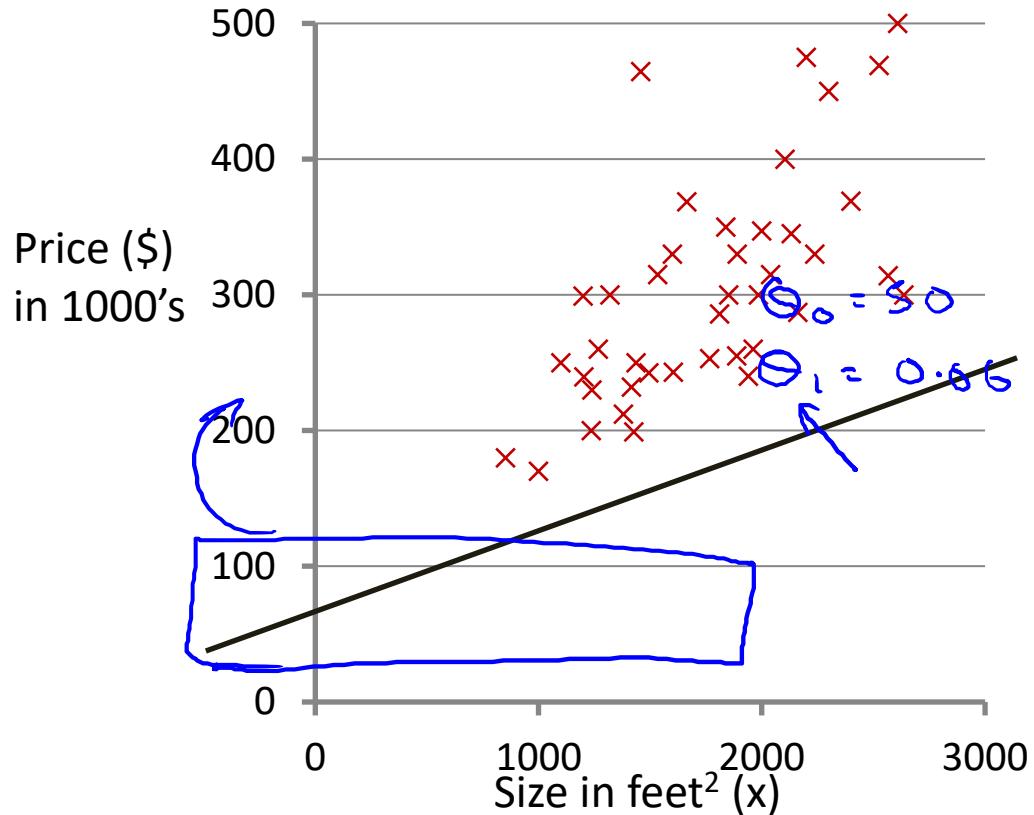
$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m \left(h_{\theta}(x^{(i)}) - y^{(i)} \right)^2$$

Goal:

$$\underset{\theta_0, \theta_1}{\text{minimize}} J(\theta_0, \theta_1)$$

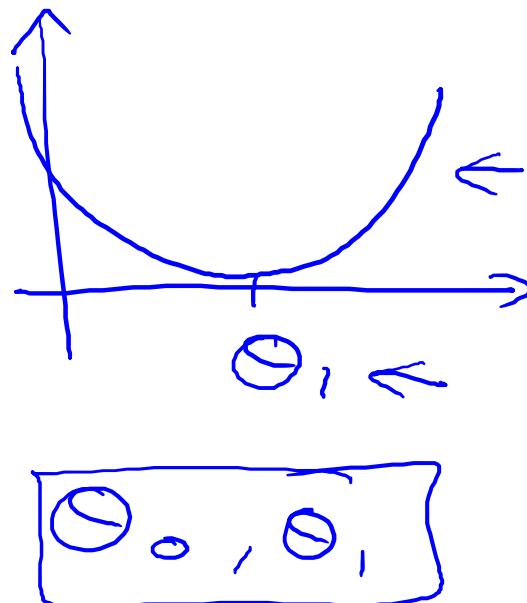
$$h_{\theta}(x)$$

(for fixed θ_0, θ_1 , this is a function of x)

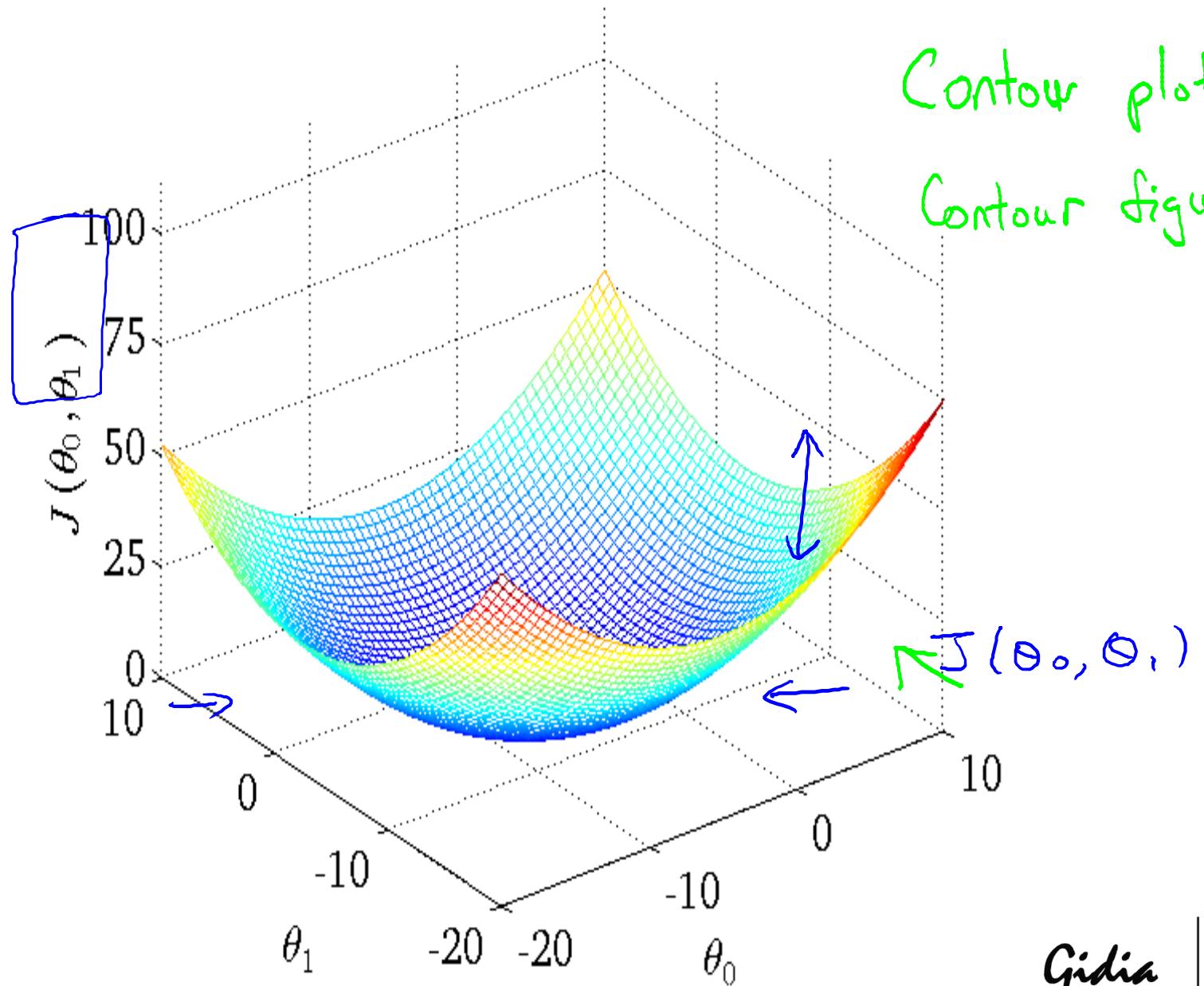


$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)

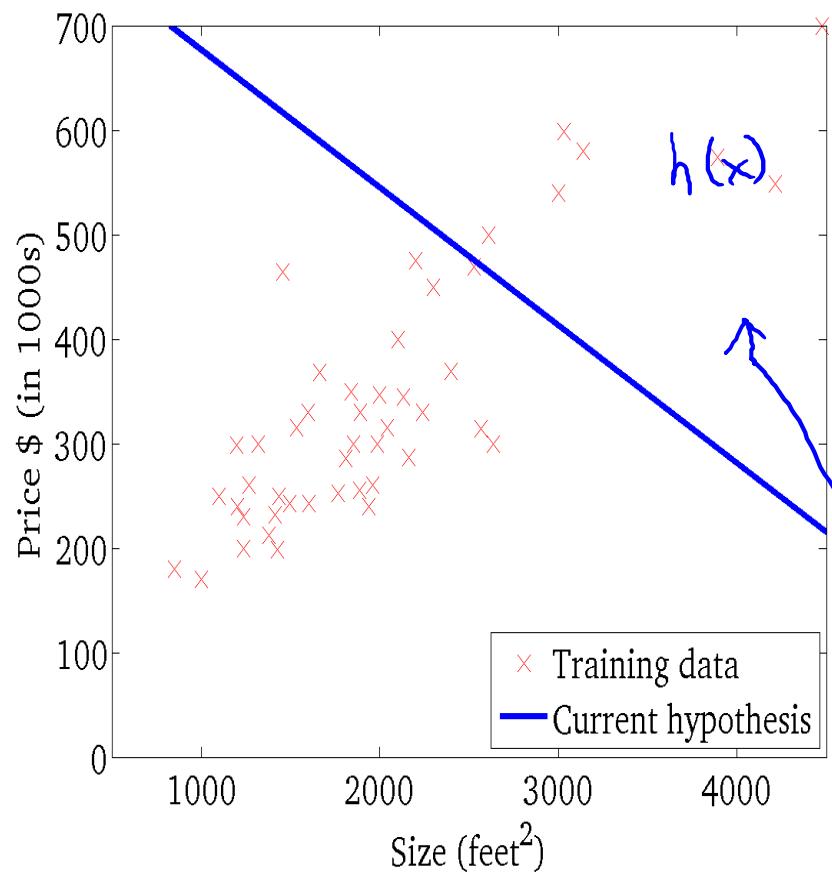


Contour plots
Contour figures -



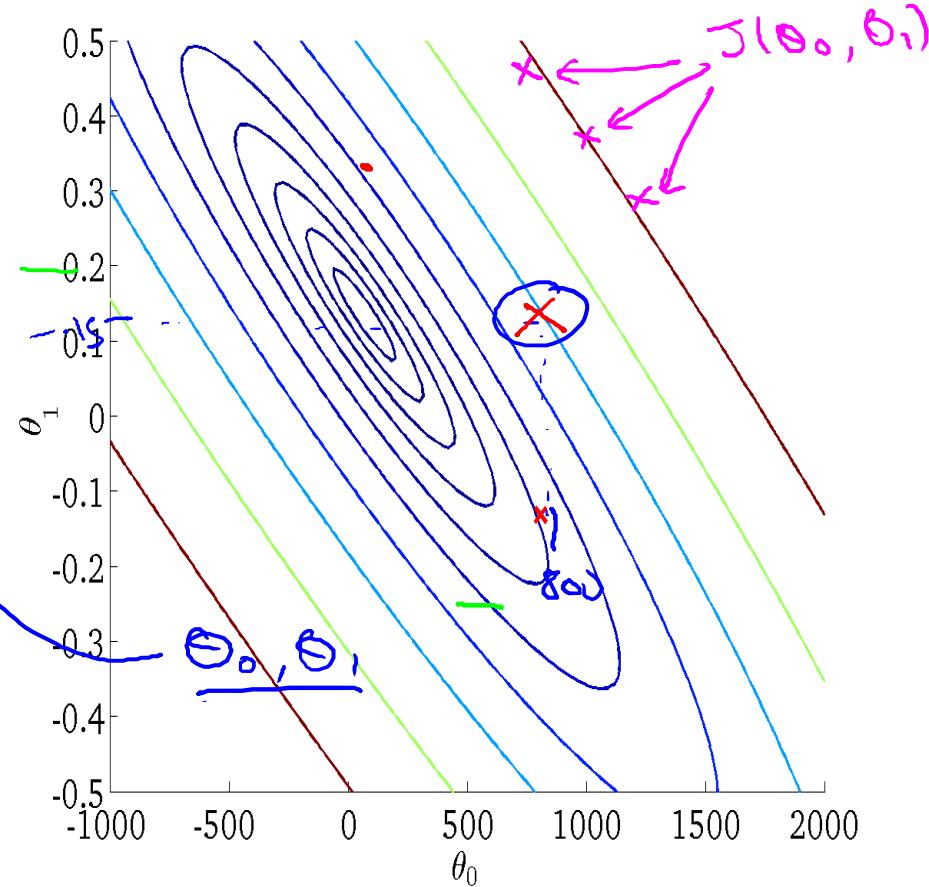
$$h_{\theta}(x)$$

(for fixed θ_0, θ_1 , this is a function of x)



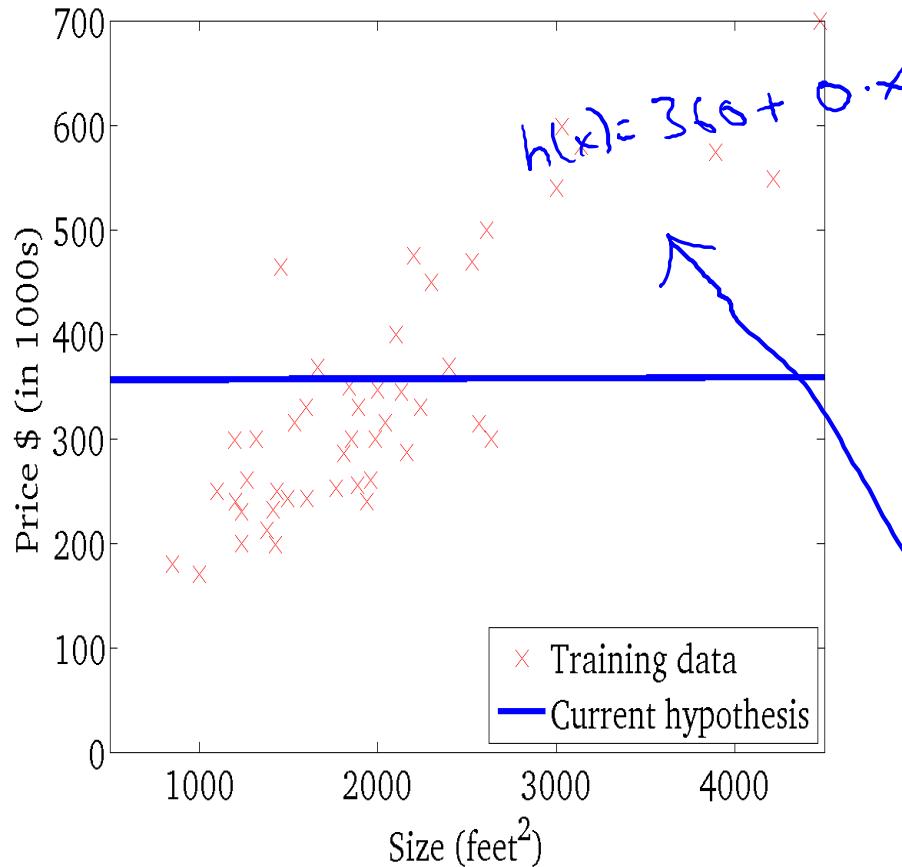
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



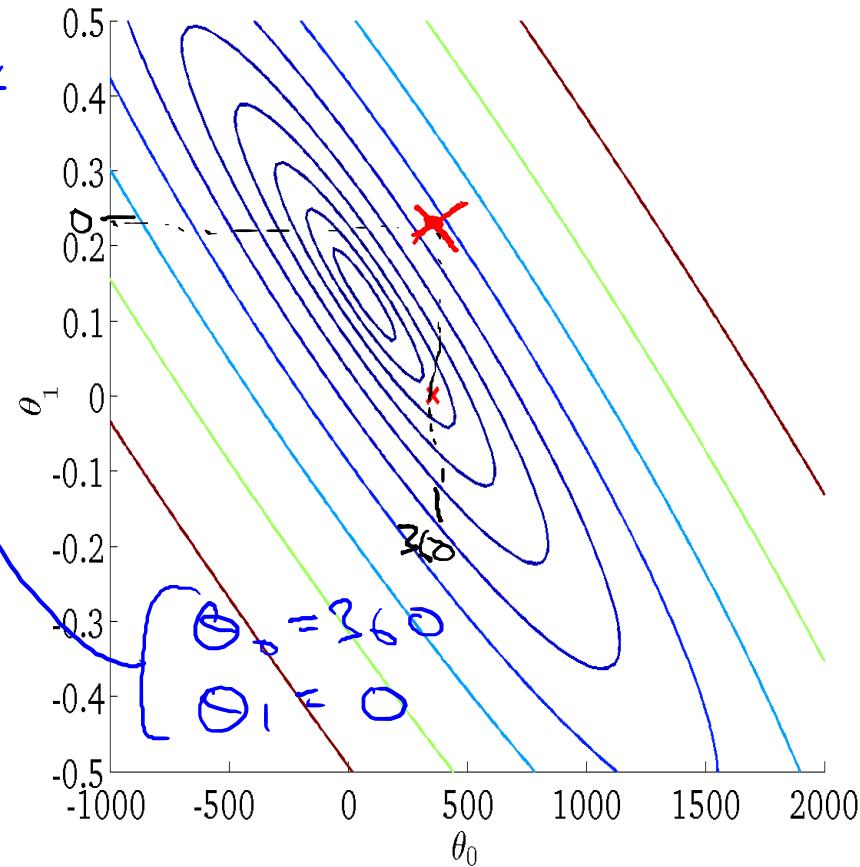
$$h_{\theta}(x)$$

(for fixed θ_0, θ_1 , this is a function of x)

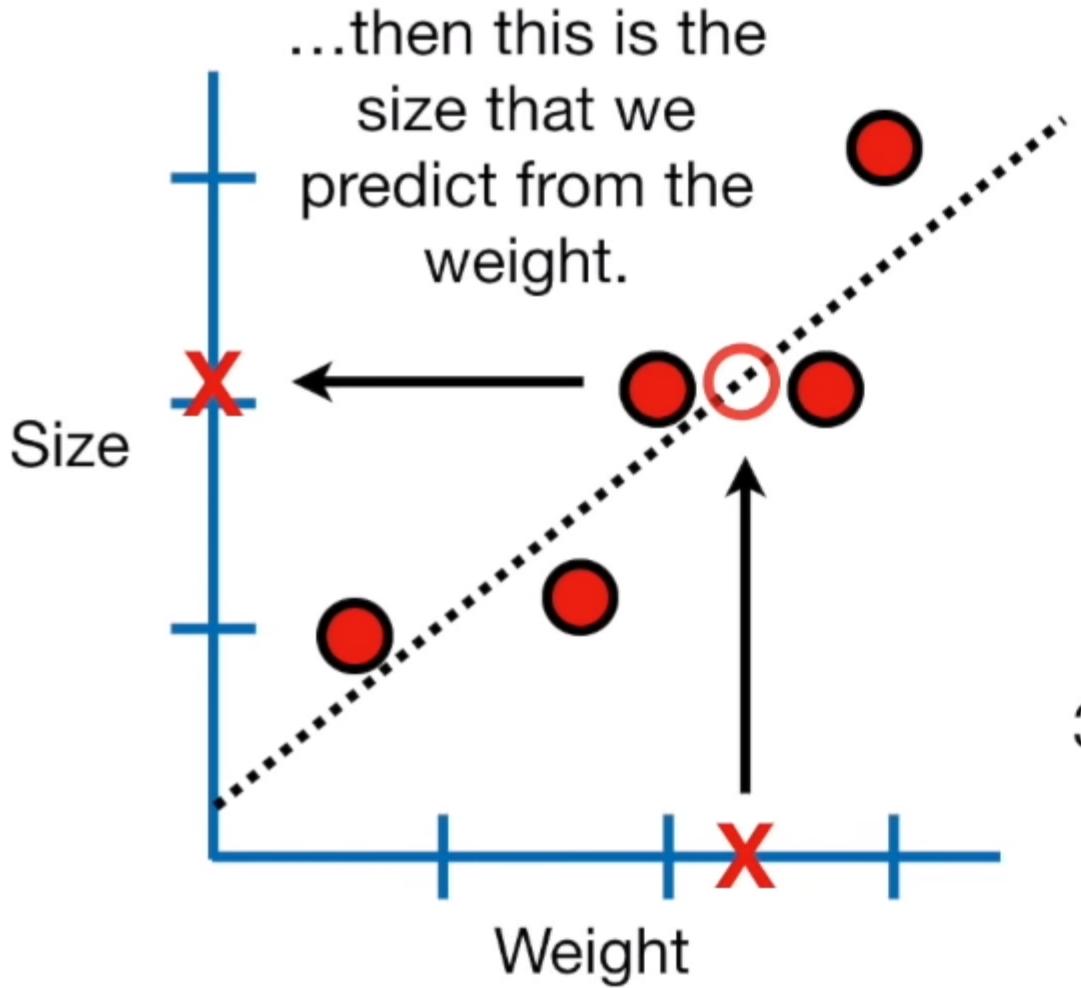


$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)

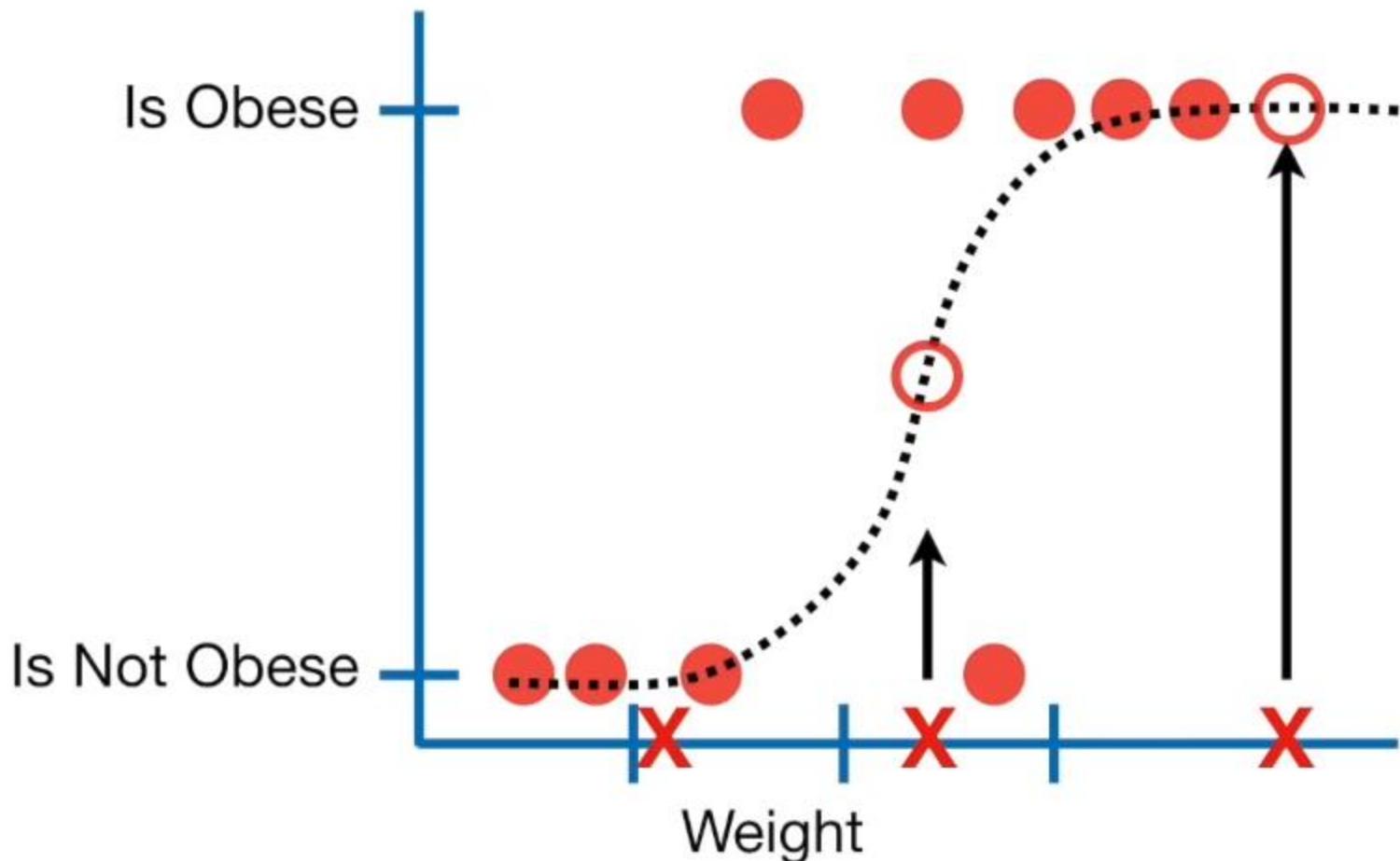


Recordemos: Regresión Lineal



Recordemos: Regresión Logística

Although logistic regression tells the probability that a mouse is obese or not, it's usually used for classification.



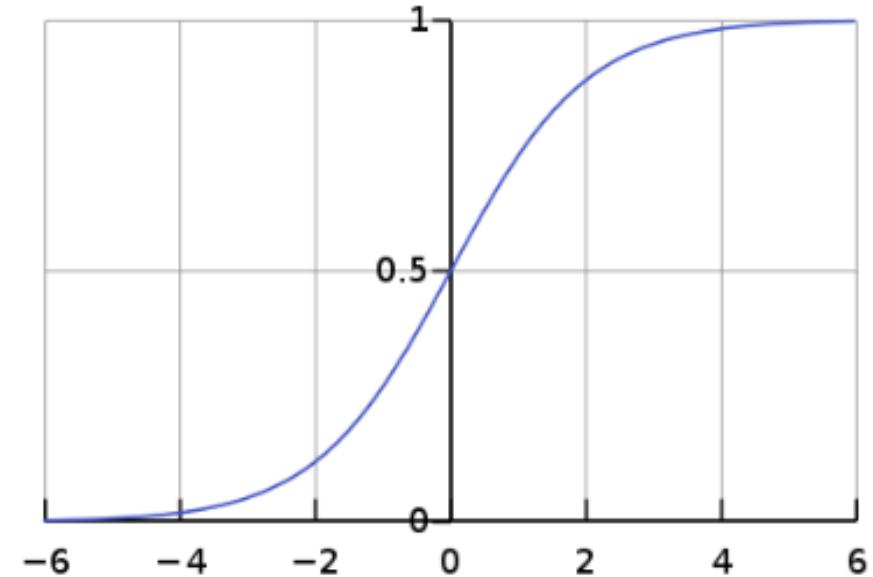
Recordemos: Regresión Logística

$$y = \begin{cases} 1 & \beta_0 + \beta_1 x + \varepsilon > 0 \\ 0 & \text{else} \end{cases}$$

$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}}$$

$$t = \beta_0 + \beta_1 x$$

$$p(x) = \sigma(t) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$



Recordemos: Regresión Logística

Así como en la regresión lineal , podemos correlacionar modelos simples:

La **Obesidad** de los ratones se puede predecir en función del **Peso**

O modelos más complejos::

Obesidad predicha en función del **Peso + Genotipo**

O aun más complejos:

Obesidad predicha en función del **Peso + Genotipo + Edad + Signo Zodiacal**

Podemos analizar si el efecto de determinada variable (Característica) es significativamente diferente de 0

Selección de características a través del desempeño del modelo

Ansur dataset sample

Gender	chestdepth	handlength	neckcircumference	shoulderlength	earlength
Female	243	176	326	136	62
Female	219	177	325	135	58
Male	259	193	400	145	71
Male	253	195	380	141	62

Regresión Logística en Python

Pre-procesamiento de los datos:

```
from sklearn.model_selection import train_test_split  
  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)  
  
from sklearn.preprocessing import StandardScaler  
  
scaler = StandardScaler()  
  
X_train_std = scaler.fit_transform(X_train)
```

Standardization:

$$z = \frac{x - \mu}{\sigma}$$

with mean:

$$\mu = \frac{1}{N} \sum_{i=1}^N (x_i)$$

and standard deviation:

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \mu)^2}$$

Regresión Logística en Python

Creamos un modelo de regresión Logística:

```
from sklearn.linear_model import LogisticRegression  
from sklearn.metrics import accuracy_score  
  
lr = LogisticRegression()  
lr.fit(X_train_std, y_train)  
  
X_test_std = scaler.transform(X_test)  
  
y_pred = lr.predict(X_test_std)  
print(accuracy_score(y_test, y_pred))
```

0.99

Regresión Logística en Python

Podemos inspeccionar los coeficientes asociados a cada dimensión:

```
print(lr.coef_)
```

```
array([[-3.   ,  0.14,  7.46,  1.22,  0.87]])
```

```
print(dict(zip(X.columns, abs(lr.coef_[0]))))
```

```
{'chestdepth': 3.0,
'handlength': 0.14,
'neckcircumference': 7.46,
'shoulderlength': 1.22,
'earlength': 0.87}
```

Regresión Logística en Python

Eliminamos las características que contribuyen poco al modelo:

```
X.drop('handlength', axis=1, inplace=True)  
  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)  
  
lr.fit(scaler.fit_transform(X_train), y_train)  
  
print(accuracy_score(y_test, lr.predict(scaler.transform(X_test))))
```

0.99

Eliminación de Características hacia atrás

(Backward Feature Elimination)

Como automatizamos el proceso de eliminación de características?

1. Primero tomamos todas las n características presentes en nuestro conjunto de datos y entrenamos el modelo usándolas.
2. Luego calculamos el rendimiento del modelo.
3. Ahora, calculamos el rendimiento del modelo después de eliminar cada característica (n veces), es decir, eliminamos una variable cada vez y entrenamos el modelo con las n-1 características restantes
4. Identificamos la variable cuya eliminación ha producido el cambio más pequeño (o nulo) en el rendimiento del modelo, y luego descartamos esa variable
5. Repita este proceso hasta que no se pueda descartar ninguna variable

Recursive Feature Elimination (RFE) en Python

```
from sklearn.feature_selection import RFE  
  
rfe = RFE(estimator=LogisticRegression(), n_features_to_select=2, verbose=1)  
rfe.fit(X_train_std, y_train)
```

```
Fitting estimator with 5 features.  
Fitting estimator with 4 features.  
Fitting estimator with 3 features.
```

Al descartar características, estamos afectando los coeficientes de las características que quedan.

Recursive Feature Elimination (RFE) en Python

Analizando los resultados tenemos:

```
X.columns[rfe.support_]
```

```
Index(['chestdepth', 'neckcircumference'], dtype='object')
```

```
print(dict(zip(X.columns, rfe.ranking_)))
```

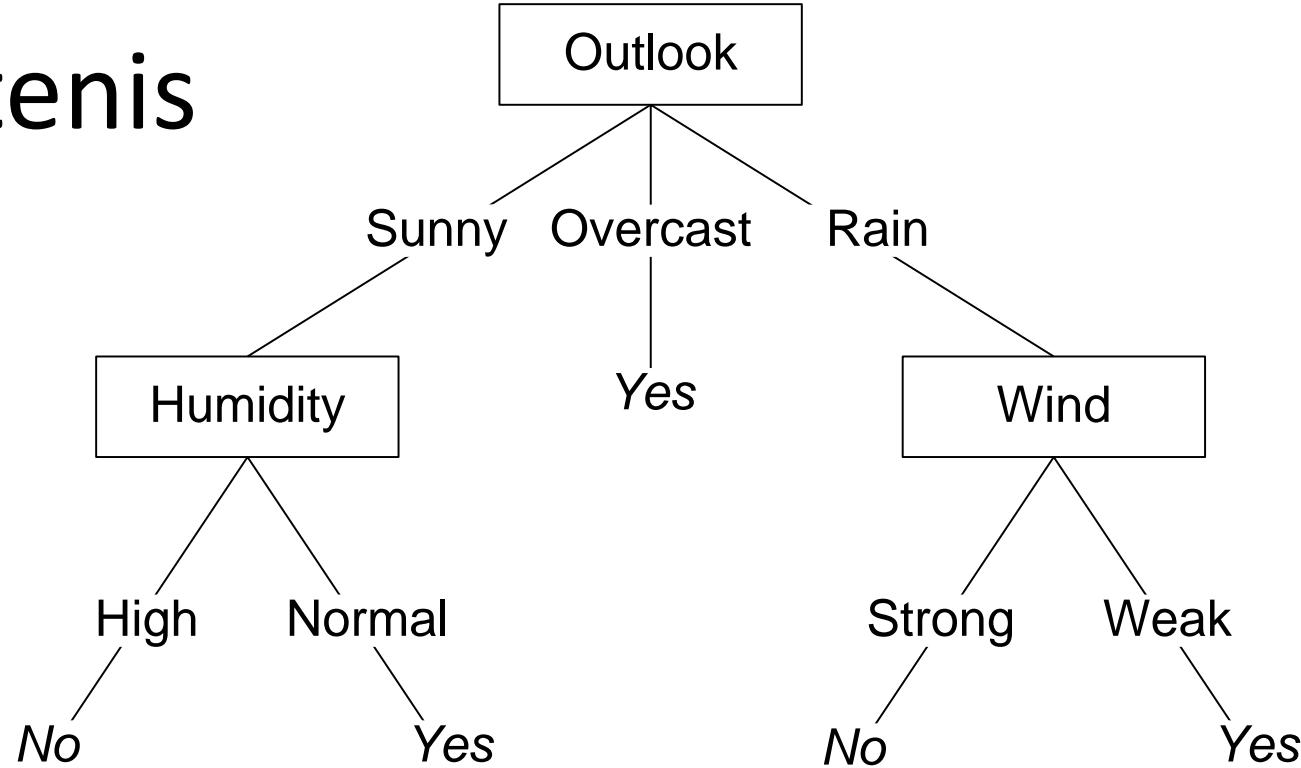
```
{'chestdepth': 1,  
 'handlength': 4,  
 'neckcircumference': 1,  
 'shoulderlength': 2,  
 'earlength': 3}
```

```
print(accuracy_score(y_test, rfe.predict(X_test_std)))
```

```
0.99
```

Recordemos: Árboles de Decisión

Jugar tenis



Recordemos: Árboles de Decisión

Jugar Tennis

Disyunción de conjunciones:

(Outlook = Sunny **And** Humidity = Normal)

Or (Outlook = Overcast)

Or (Outlook = Rain **And** Wind = Weak)

Recordemos: Arboles de Decisión

Información:
reducción de la incertidumbre (Entropía)

$$I(E) = \log_2 \frac{1}{p(x)} = -\log_2 p(x)$$

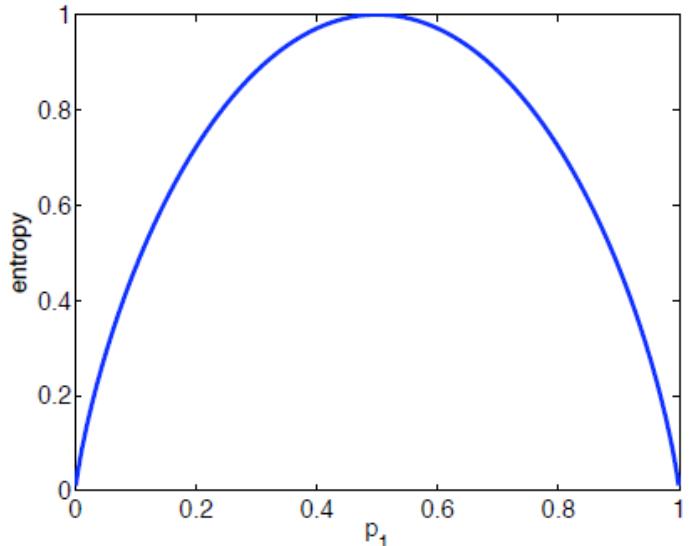
Si la probabilidad de que ocurra este evento es pequeña y sucede
La cantidad de información recibida es grande.

Observando que el lanzamiento de la
moneda cayó CARA!!

$$\longrightarrow I = -\log_2 1/2 = 1$$

Observando que el lanzamiento del dado
arrojó el número 6.

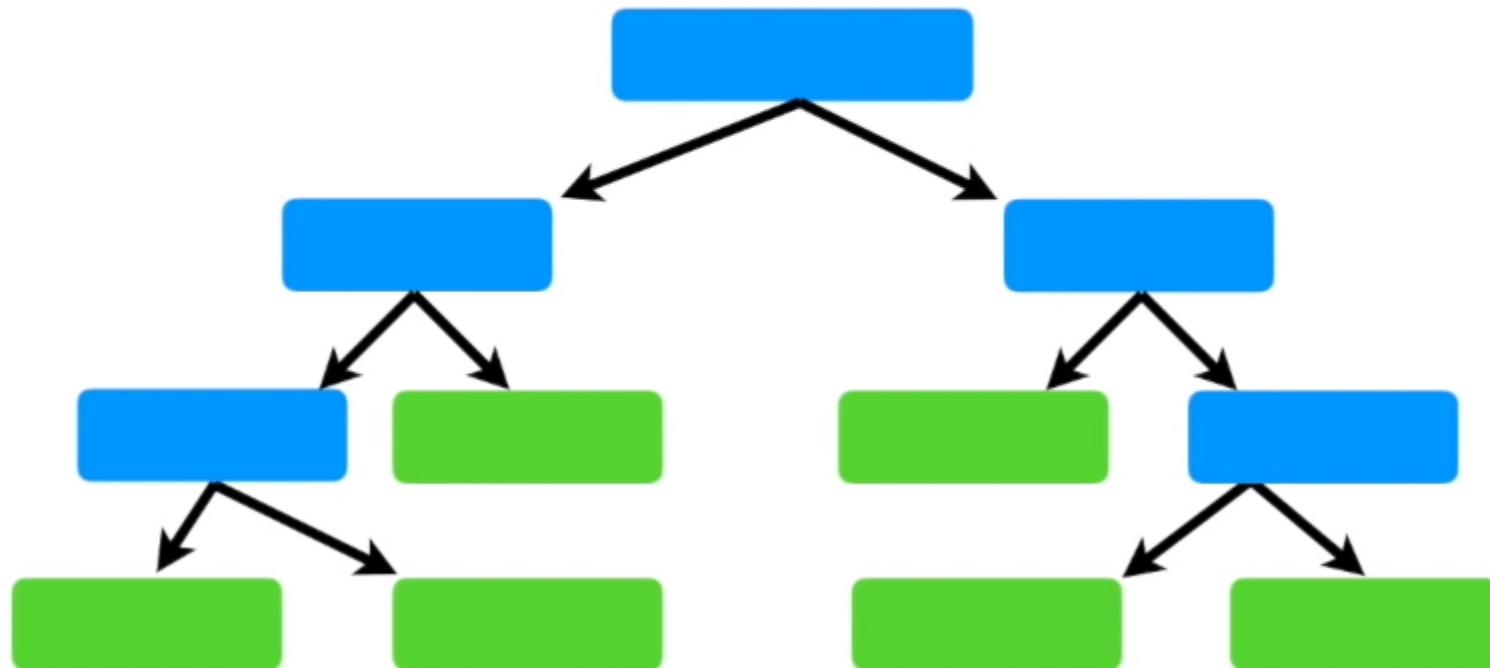
$$\longrightarrow I = -\log_2 1/6 = 2.58$$



Random Forests – Bosques Aleatorios

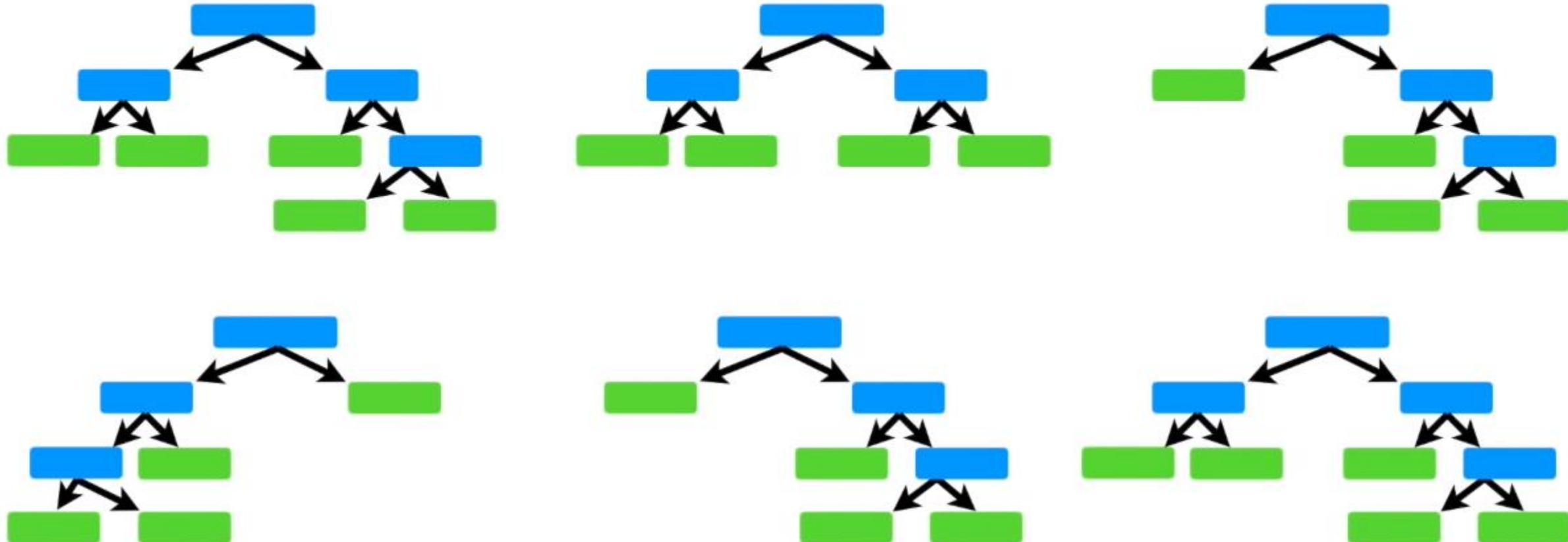
Surgen como solución a problemas de precisión de los árboles de decisión

In other words, they work great with the data used to create them, but **they are not flexible when it comes to classifying new samples.**



Random Forests – Bosques Aleatorios

The good news is that **Random Forests** combine the simplicity of decision trees with flexibility resulting in a vast improvement in accuracy.



Random Forests – Bosques Aleatorios

Paso 1: Crear un “bootstraped” dataset

Random Forests – Bosques Aleatorios

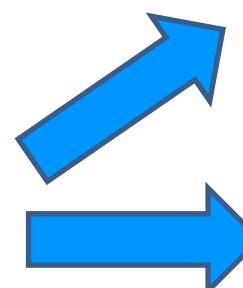
Paso 1: Crear un “bootstraped” dataset

Original Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

Bootstrapped Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes



Random Forests – Bosques Aleatorios

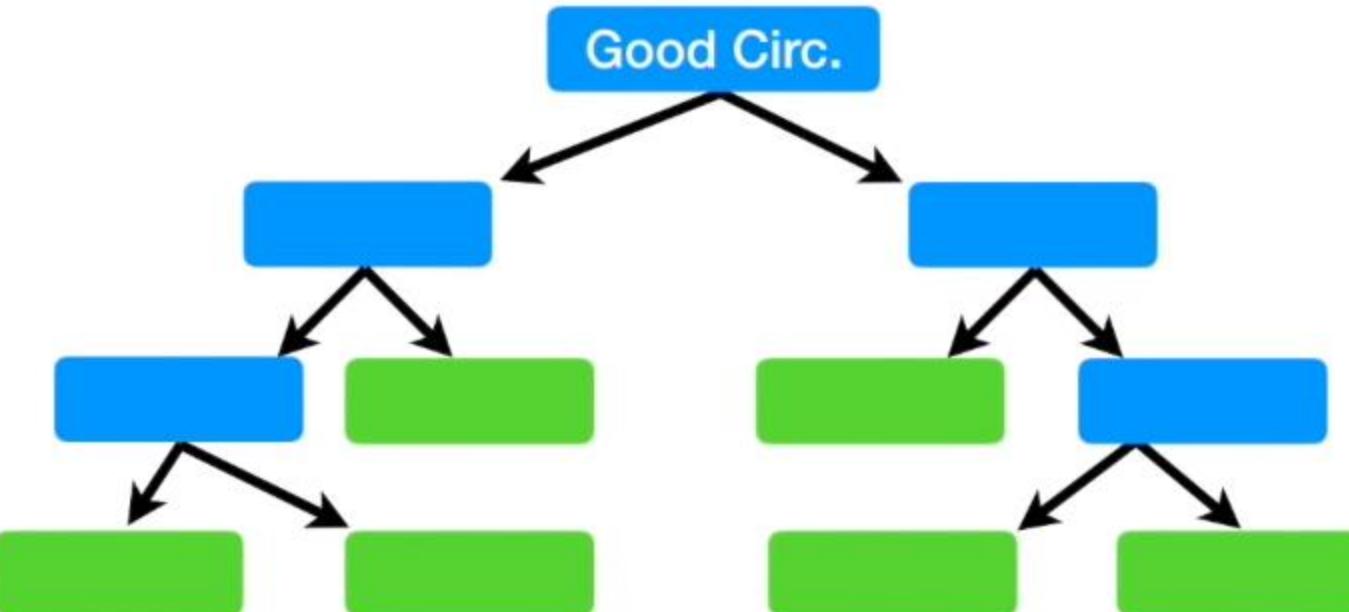
Paso 2: Crear un árbol de decisión usando el “boostaped” dataset, pero usando únicamente un sub-conjunto de características (Columnas) a cada paso.

Random Forests – Bosques Aleatorios

Paso 2: Crear un árbol de decisión usando “bootstraped” dataset

We built a tree...

- 1) Using a bootstrapped dataset
- 2) Only considering a random a subset of variables at each step.



Bootstrapped Dataset

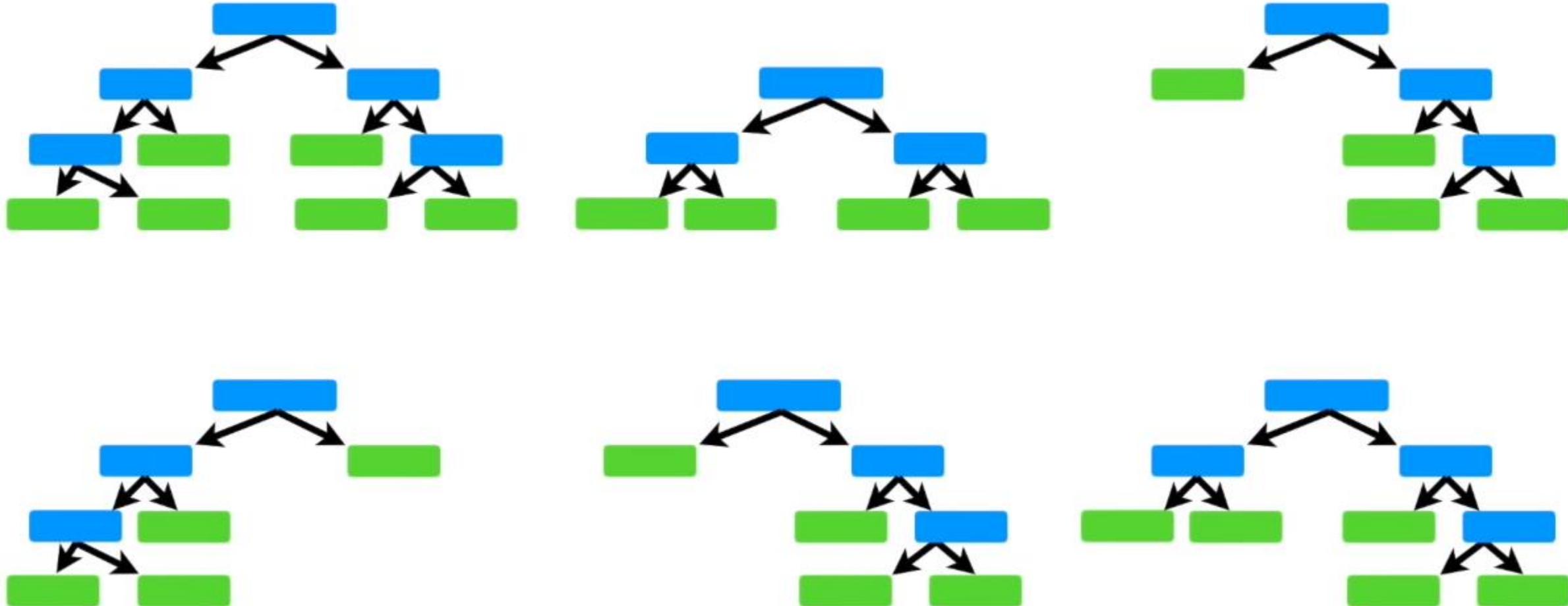
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

Random Forests – Bosques Aleatorios

Paso 3: Creamos un nuevo “boostraped” dataset y construimos otros arboles de decisión usando este nuevo “boostraped” dataset, y usando otra selección aleatoria de sub-conjuntos de características (Columnas) a cada paso.

Random Forests – Bosques Aleatorios

Paso 3: Creamos varios árboles de decisión usando nuevos “bootstrapped” datasets



Random Forests – Bosques Aleatorios - Evaluación

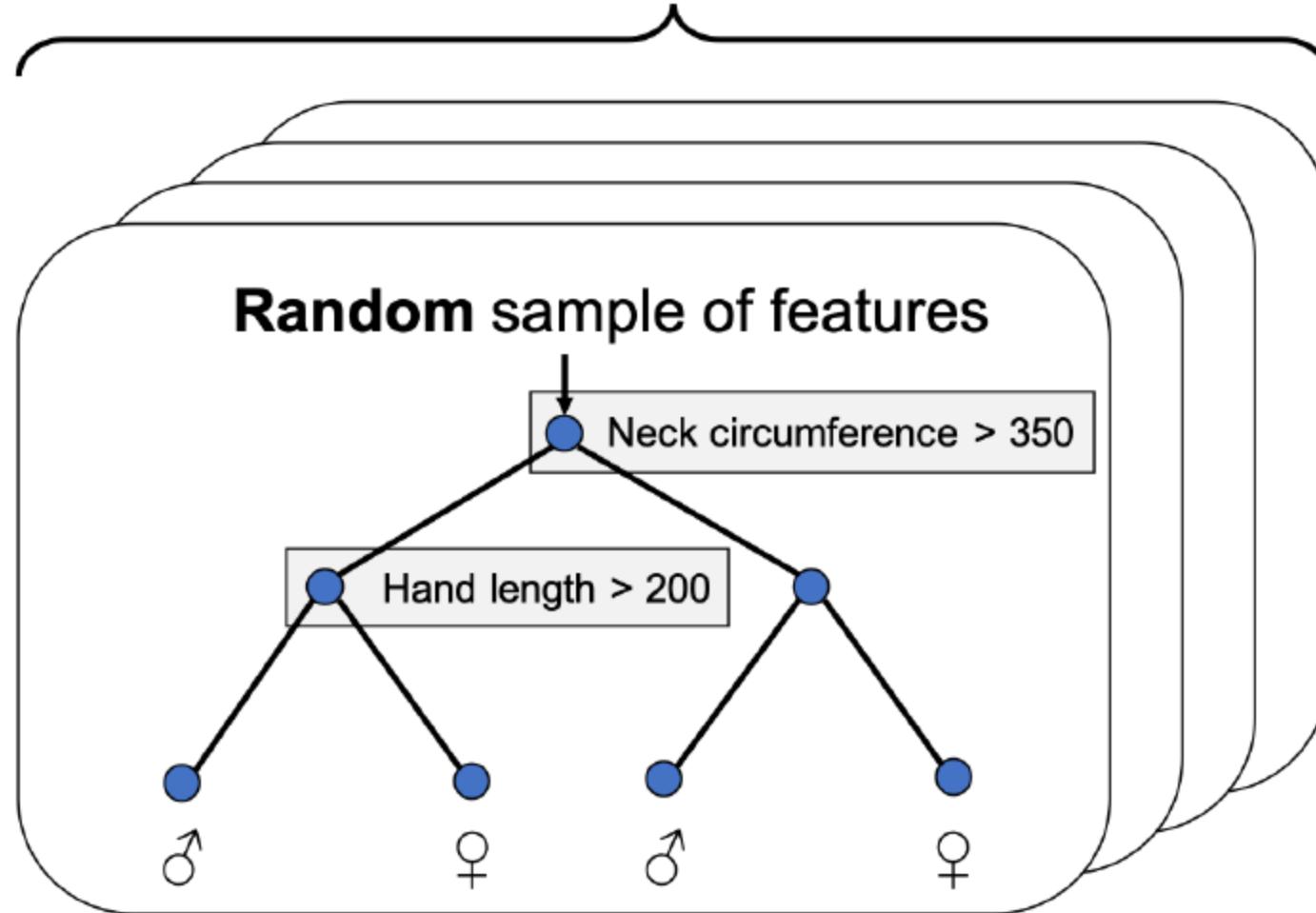
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	No	168	YES

In this case, “**Yes**” received the most votes, so we will conclude that this patient has heart disease.



Random Forests – Bosques Aleatorios – En Python

Forest of decision trees



Random Forests – Bosques Aleatorios – En Python

Valores según importancia de las características

```
rf = RandomForestClassifier()
```

```
rf.fit(X_train, y_train)
```

```
print(rf.feature_importances_)
```

```
array([0.    , 0.    , 0.    , 0.    , 0.    , 0.    , 0.    , 0.04, 0.    , 0.01, 0.01,
       0.    , 0.    , 0.    , 0.    , 0.01, 0.01, 0.    , 0.    , 0.    , 0.    , 0.05,
       ...
       0.    , 0.14, 0.    , 0.    , 0.    , 0.06, 0.    , 0.    , 0.    , 0.    , 0.    ,
       0.    , 0.07, 0.    , 0.    , 0.01, 0.    ])
```

```
print(sum(rf.feature_importances_))
```

1.0

Random Forests – Bosques Aleatorios – En Python

Importancia de las características como selector de características

```
mask = rf.feature_importances_ > 0.1
```

```
print(mask)
```

```
array([False, False, ..., True, False])
```

```
X_reduced = X.loc[:, mask]
```

```
print(X_reduced.columns)
```

```
Index(['chestheight', 'neckcircumference', 'neckcircumferencebase',
       'shouldercircumference'], dtype='object')
```

RFE con Random Forests – en Python

```
from sklearn.feature_selection import RFE  
  
rfe = RFE(estimator=RandomForestClassifier(),  
           n_features_to_select=6, verbose=1)  
  
rfe.fit(X_train,y_train)
```

```
Fitting estimator with 94 features.  
Fitting estimator with 93 features  
...  
Fitting estimator with 8 features.  
Fitting estimator with 7 features.
```

```
print(accuracy_score(y_test, rfe.predict(X_test))
```

```
0.99
```

RFE con Random Forests – en Python

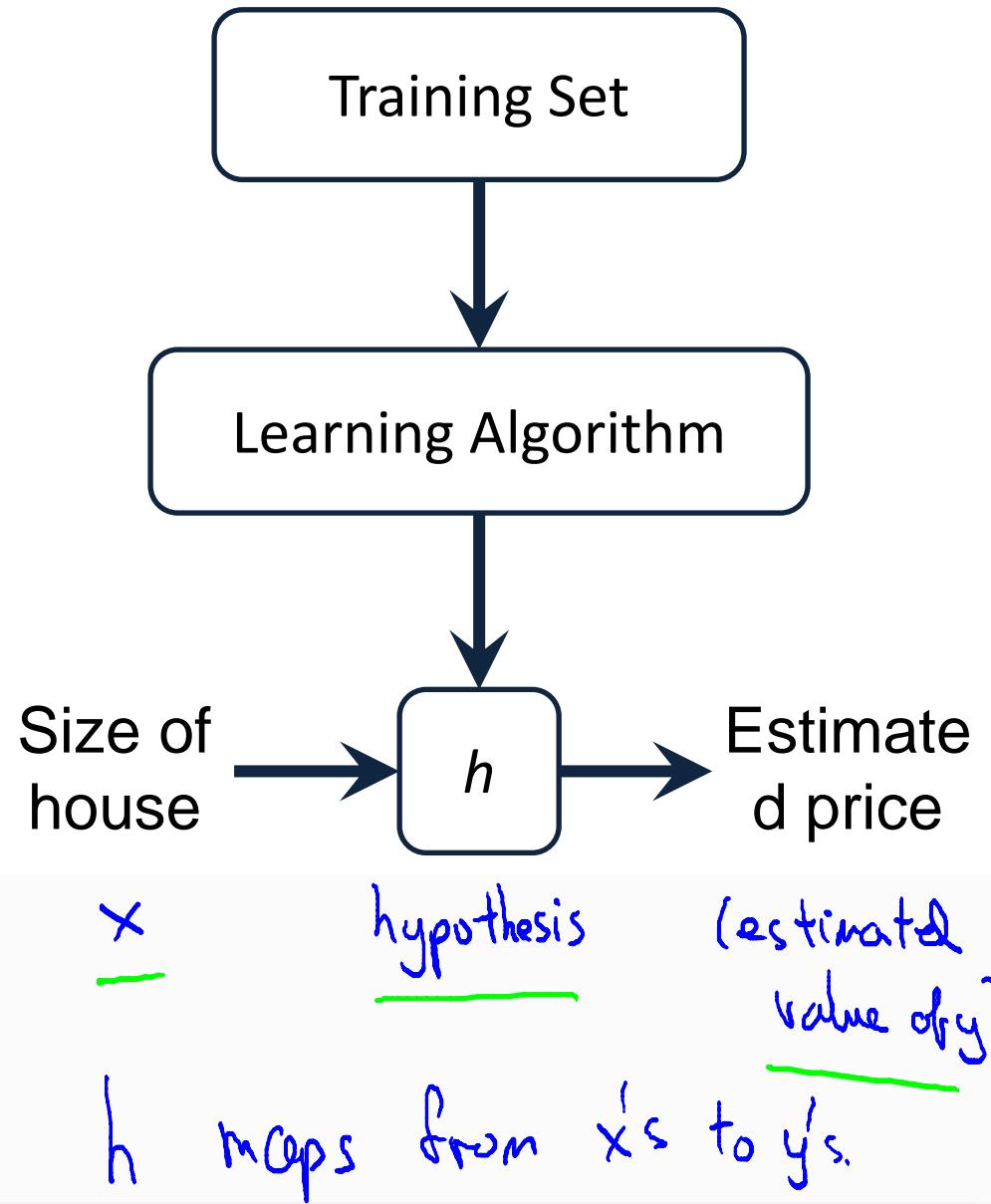
```
from sklearn.feature_selection import RFE  
  
rfe = RFE(estimator=RandomForestClassifier(),  
           n_features_to_select=6, verbose=1)  
  
rfe.fit(X_train,y_train)
```

```
Fitting estimator with 94 features.  
Fitting estimator with 93 features  
...  
Fitting estimator with 8 features.  
Fitting estimator with 7 features.
```

```
print(X.columns[rfe.support_])
```

```
Index(['biacromialbreadth', 'handbreadth', 'handcircumference',  
       'neckcircumference', 'neckcircumferencebase', 'shouldercircumference'], dtype='object')
```

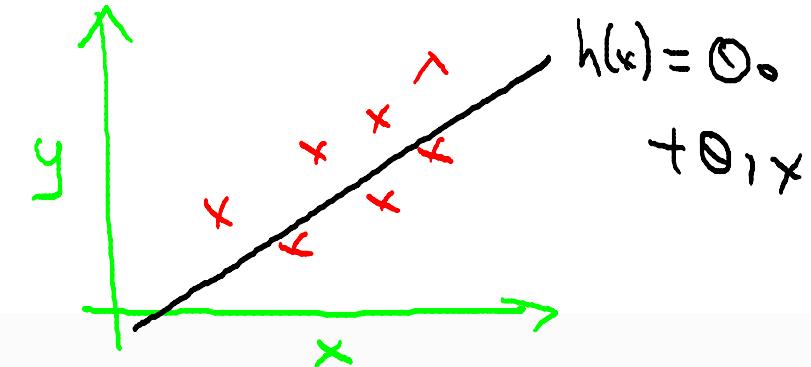
Regresión Lineal Multivariable



How do we represent h ?

$$h_{\theta}(x) = \underline{\underline{\theta_0 + \theta_1 x}}$$

Shorthand: $\underline{\underline{h(x)}}$



Linear regression with one variable.
Univariate linear regression.

One variable

Regresión Lineal Multivariable

Multiple features (variables).

Size (feet ²) → x	Price (\$1000) y ←
2104	460
1416	232
1534	315
852	178
...	...
$h_{\theta}(x) = \theta_0 + \theta_1 x$	

Regresión Lineal Multivariable

Multiple features (variables).

Size (feet ²)	Number of bedrooms	Number of floors	Age of home (years)	Price (\$1000)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...

Regresión Lineal Multivariable

Multiple features (variables).

Size (feet ²) → x_1	Number of bedrooms → x_2	Number of floors → x_3	Age of home (years) → x_4	Price (\$1000) y
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...

Notation:

→ n = number of features

$$n=4$$

→ $x^{(i)}$ = input (features) of i^{th} training example.

→ $x_j^{(i)}$ = value of feature j in i^{th} training example.

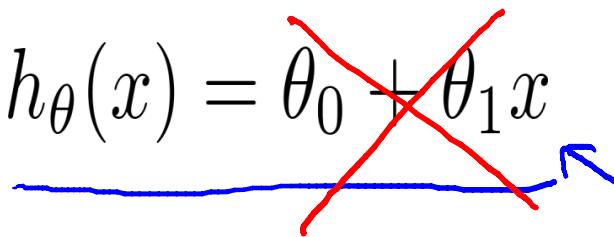
$$\underline{x}^{(2)} = \begin{bmatrix} 1416 \\ 3 \\ 2 \\ 40 \end{bmatrix}$$

$$x_3^{(2)} = 2$$

Regresión Lineal Multivariable

Hypothesis:

Previously:

$$h_{\theta}(x) = \theta_0 + \theta_1 x$$


$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$$

E.g. $\underline{h_{\theta}(x)} = \underline{\underline{\theta_0}} + \underline{\underline{\theta_1}x_1} + \underline{\underline{\theta_2}x_2} + \underline{\underline{\theta_3}x_3} + \underline{\underline{\theta_4}x_4}$

↑ ↑ ↑ ↑
age

Regresión Lineal Multivariable

$$\rightarrow h_{\theta}(x) = \underline{\theta_0} + \underline{\theta_1}x_1 + \underline{\theta_2}x_2 + \cdots + \underline{\theta_n}x_n$$

For convenience of notation, define $x_0 = 1$ ($x_0^{(i)} = 1$)

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1}$$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1}$$

$$\begin{aligned} h_{\theta}(x) &= \underline{\theta_0x_0 + \theta_1x_1 + \cdots + \theta_nx_n} \\ &= \boxed{\theta^T x} \end{aligned}$$

θ^T
 $(n+1) \times$
matrix
 $\theta^T x$

x

Multivariate linear regression. 

Linear Regression with multiple variables

Gradient descent for multiple variables

Regresión Lineal Multivariable

Hypothesis: $h_{\theta}(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

Parameters: $\underline{\theta_0, \theta_1, \dots, \theta_n}$ Θ n+1 - dimensional vector

Cost function: $J(\underline{\theta_0, \theta_1, \dots, \theta_n}) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$
 $J(\Theta)$

Regresión Lineal Multivariable

Hypothesis: $h_{\theta}(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

Parameters: $\underline{\theta_0, \theta_1, \dots, \theta_n}$ Θ $n+1$ -dimensional vector

Cost function:

$$J(\underline{\theta_0, \theta_1, \dots, \theta_n}) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

$J(\Theta)$

Gradient descent:

Repeat {

$$\rightarrow \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\underline{\theta_0, \dots, \theta_n}) \quad J(\Theta)$$

}

↑ (simultaneously update for every $j = 0, \dots, n$)



UNIVERSIDAD
NACIONAL
DE COLOMBIA

Gradient Descent

Previously ($n=1$):

Repeat {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) \underbrace{\frac{\partial}{\partial \theta_0} J(\theta)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)} \quad (\text{simultaneously update } \theta_0, \theta_1)$$

}

New algorithm ($n \geq 1$):

Repeat {

$$\quad \quad \quad \downarrow \frac{\partial}{\partial \theta_j} J(\theta)$$

$$\rightarrow \theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

(simultaneously update θ_j for
 $j = 0, \dots, n$)

}

$$\rightarrow \theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\rightarrow \theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)}$$

$$\rightarrow \theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_2^{(i)}$$

...

Linear Regression with multiple variables

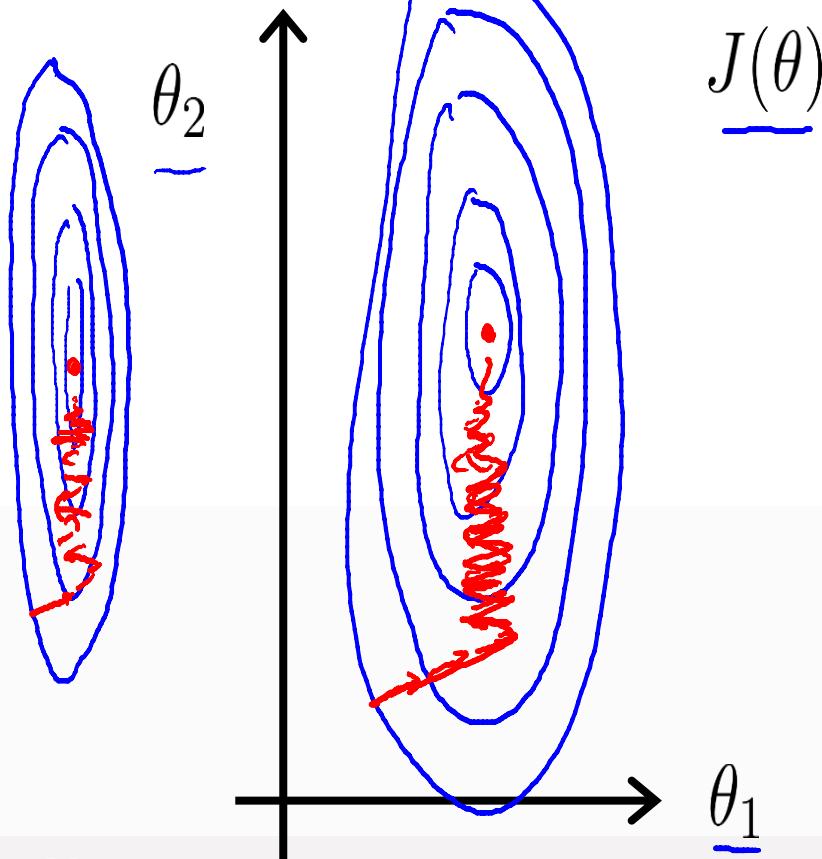
Gradient descent in practice I: Feature Scaling

Feature Scaling

Idea: Make sure features are on a similar scale.

E.g. x_1 = size (0-2000 feet²)

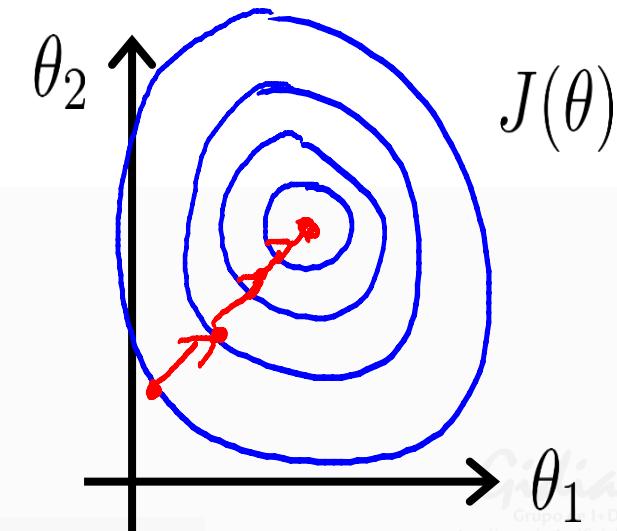
x_2 = number of bedrooms (1-5)



$$\rightarrow x_1 = \frac{\text{size (feet}^2)}{2000}$$

$$\rightarrow x_2 = \frac{\text{number of bedrooms}}{5}$$

$$0 \leq x_1 \leq 1 \quad 0 \leq x_2 \leq 1$$



Feature Scaling

Get every feature into approximately a $-1 \leq x_i \leq 1$ range.

$$x_0 = 1$$

$$-1 \leq x_i \leq 1$$

$$6 \leq x_1 \leq 3 \quad \checkmark$$

$$-2 \leq x_2 \leq 0.5 \quad \checkmark$$

$$-100 \leq x_3 \leq 100 \quad \times$$

$$-0.0001 \leq x_4 \leq 0.0001 \quad \times$$

$$-3 \text{ to } 3 \quad \checkmark$$

$$-\frac{1}{3} \text{ to } \frac{1}{3} \quad \checkmark$$

Mean normalization

Replace x_i with $\frac{x_i - \mu_i}{\sigma_i}$ to make features have approximately zero mean
(Do not apply to $x_0 = 1$).

E.g.

$$x_1 = \frac{\text{size} - 1000}{2000}$$

Average size = 100

$$x_2 = \frac{\#bedrooms - 2}{5 - 4}$$

1 - 5 bedrooms

$$\rightarrow -0.5 \leq x_1 \leq 0.5, -0.5 \leq x_2 \leq 0.5$$

$x_1 \leftarrow$

$$\frac{x_1 - \mu_1}{\sigma_1}$$

avg value
of x_1
in training
set

range (max-min)

(or standard deviation)

$$x_2 \leftarrow \frac{x_2 - \mu_2}{\sigma_2}$$

Linear Regression with multiple variables

Gradient descent in practice II: Learning rate

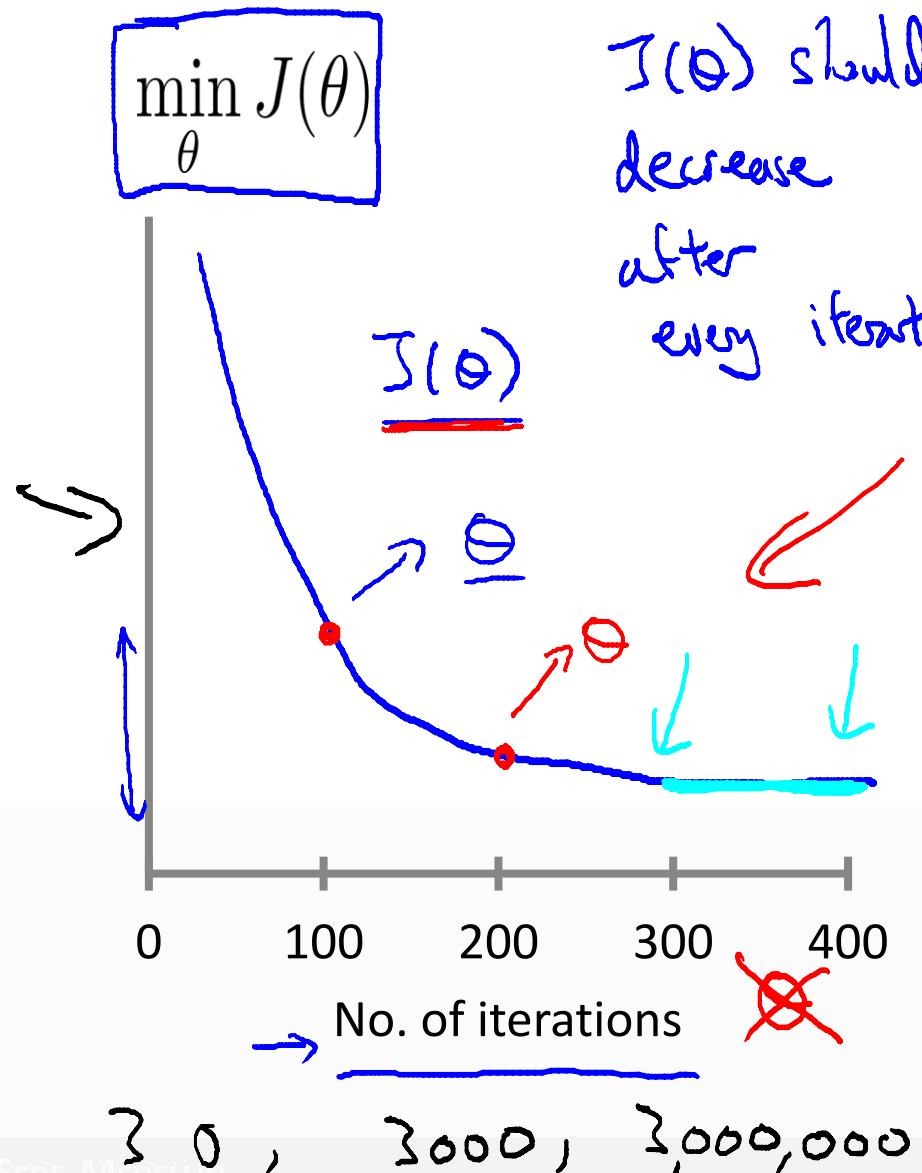
Regresión Lineal Multivariable

Gradient descent

$$\rightarrow \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

- “Debugging”: How to make sure gradient descent is working correctly.
- How to choose learning rate α .

Making sure gradient descent is working correctly.

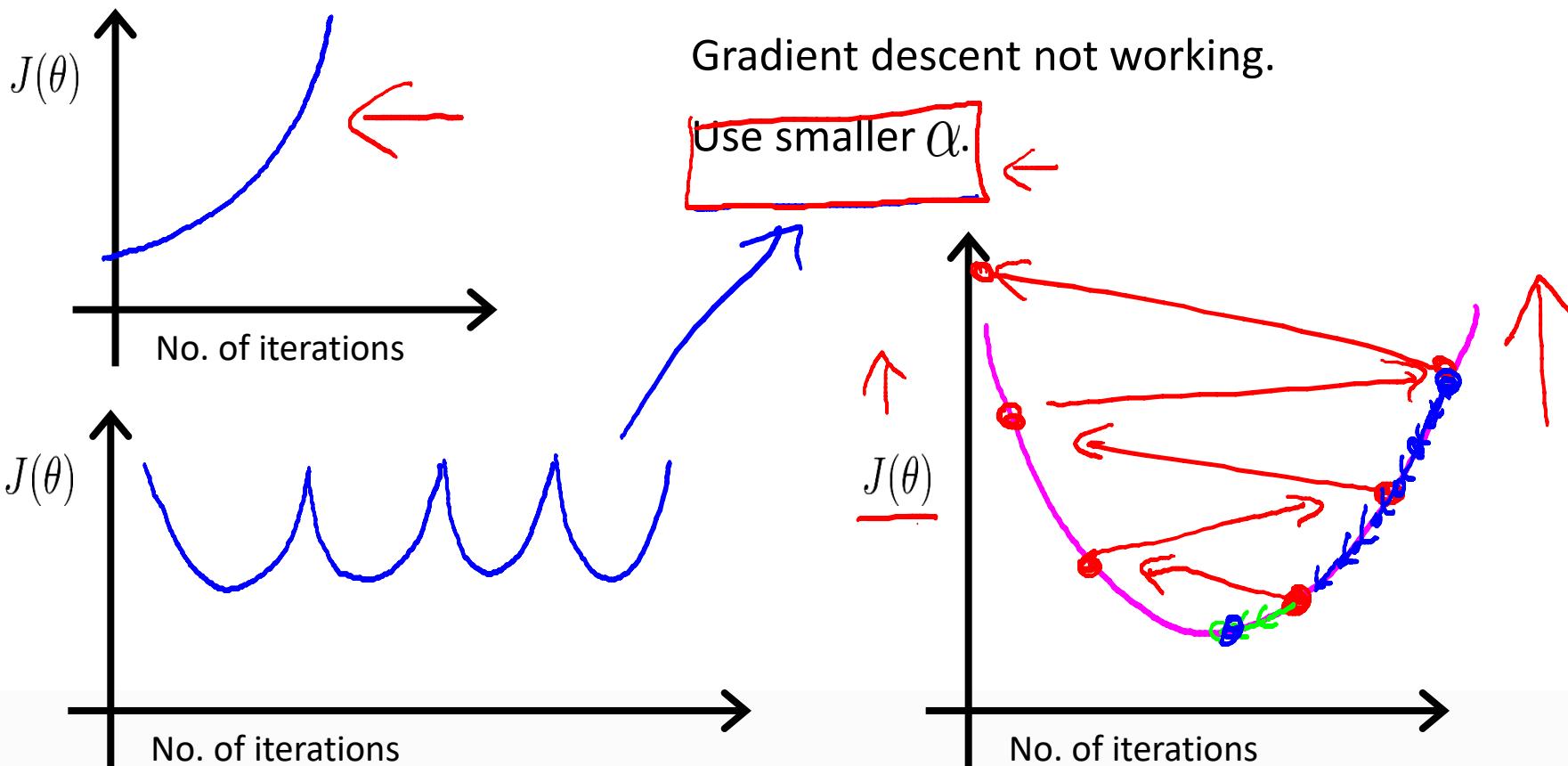


→ Example automatic convergence test:

→ Declare convergence if $\frac{J(\theta)}{\Sigma}$ decreases by less than 10^{-3} in one iteration.

$$10^{-3}$$

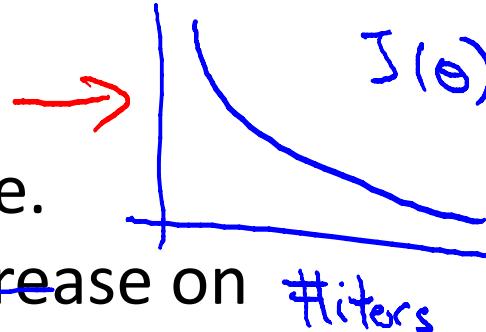
Making sure gradient descent is working correctly.



- For sufficiently small α , $J(\theta)$ should decrease on every iteration.
- But if α is too small, gradient descent can be slow to converge.

Summary:

- If α is too small: slow convergence.
- If α is too large: $J(\theta)$ may not decrease on every iteration; may not converge.



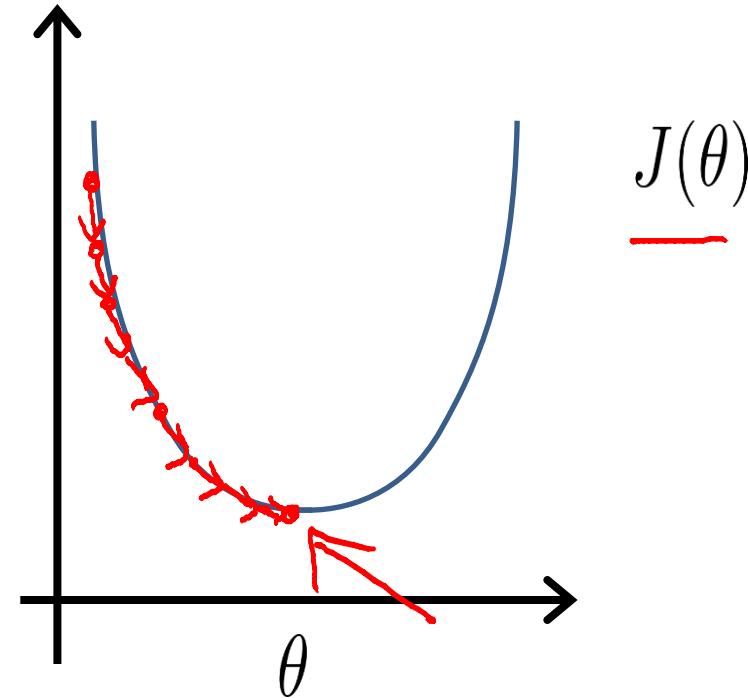
(Slow converge
also possible)

To choose α , try

$$\dots, \underbrace{0.001}_{\uparrow}, \underbrace{0.003}_{\approx 3x}, \underbrace{0.01}_{\approx 10x}, \underbrace{0.03}_{\approx 3x}, \underbrace{0.1}_{\approx 3x}, \underbrace{0.3}_{\approx 3x}, \underbrace{1}_{\approx 10x}, \dots$$

Regresión Lineal Multivariable

Gradient Descent



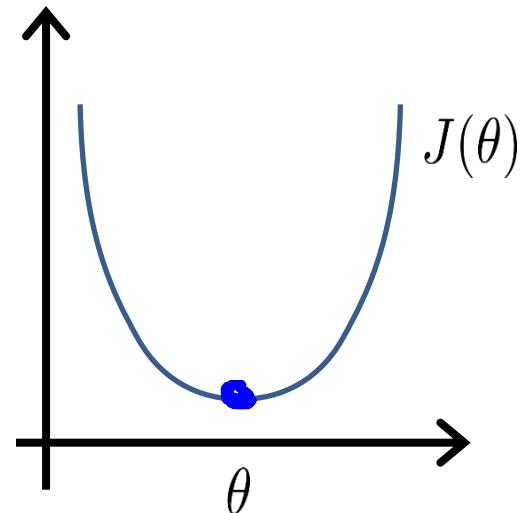
Normal equation: Method to solve for θ analytically.

Intuition: If 1D ($\theta \in \mathbb{R}$)

$$\rightarrow J(\theta) = a\theta^2 + b\theta + c$$

$$\frac{\partial}{\partial \theta} J(\theta) = \dots \stackrel{\text{set}}{=} 0$$

Solve for θ



$$\theta \in \mathbb{R}^{n+1}$$

$$J(\theta_0, \theta_1, \dots, \theta_m) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2$$

$$\frac{\partial}{\partial \theta_j} J(\theta) = \dots \stackrel{\text{set}}{=} 0 \quad (\text{for every } j)$$

Solve for $\theta_0, \theta_1, \dots, \theta_n$

Examples: $m = 4$.

	Size (feet ²)	Number of bedrooms	Number of floors	Age of home (years)	Price (\$1000)
	x_1	x_2	x_3	x_4	y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix}$$

$m \times (n+1)$

$$y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

m -dimensional vector

$$\theta = (X^T X)^{-1} X^T y$$

Examples: $m = 5$.

x_0	Size (feet²) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price (\$1000) y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178
1	3000	4	1	38	540

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \\ 1 & 3000 & 4 & 1 & 38 \end{bmatrix} \quad y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \\ 540 \end{bmatrix}$$

$$\Theta = (X^T X)^{-1} X^T y$$

m examples $(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})$; n features.

$$\underline{x^{(i)}} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix} \in \mathbb{R}^{n+1}$$

\times =

(design matrix)

$$\begin{bmatrix} \cdots & (x^{(1)})^\top & \cdots \\ \cdots & (x^{(2)})^\top & \cdots \\ \vdots & \vdots & \vdots \\ \cdots & (x^{(m)})^\top & \cdots \end{bmatrix}$$

E.g. If $\underline{x^{(i)}} = \begin{bmatrix} 1 \\ x_1^{(i)} \end{bmatrix}$

\times =

$$X = \begin{bmatrix} 1 & x_1^{(1)} \\ 1 & x_1^{(2)} \\ \vdots & \vdots \\ 1 & x_1^{(m)} \end{bmatrix}$$

$m \times 2$

$$y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

$\Theta = (X^T X)^{-1} X^T y$

$m \times (n+1)$

$$\theta = \boxed{(X^T X)^{-1} X^T y}$$



$(X^T X)^{-1}$ is inverse of matrix $X^T X$.

Set A = $X^T X$

$$\boxed{(X^T X)^{-1}} = A^{-1}$$

Octave: $\text{pinv}(\boxed{X' * X}) * X' * y$

$$\underline{\text{pinv}(X^T * X) * X^T * y}$$

$$\theta = \boxed{\theta} (X^T X)^{-1} X^T y$$

$$\min_{\theta} J(\theta)$$

$$X' \quad X^T$$

~~Feature Scaling~~

$$0 \leq x_1 \leq 1$$

$$0 \leq x_2 \leq 1000$$

$$0 \leq x_3 \leq 10^{-5} \quad \checkmark$$

m training examples, n features.

Gradient Descent

- • Need to choose α .
- Needs many iterations.
- • Works well even when n is large.

$$n = 10^6$$

Normal Equation

- • No need to choose α .
- Don't need to iterate.
- • Need to compute
- $(X^T X)^{-1}$ $n \times n$ $O(n^3)$
- Slow if n is very large.

$$n = 100$$

$$n = 1000$$

$$\dots n = 10000$$

Regresión Lineal Multivariable

Normal equation

$$\theta = (X^T X)^{-1} X^T y$$

- What if $X^T X$ is non-invertible? (singular/degenerate)
- Python: `np.linalg.pinv(a)`

Regresión Lineal Multivariable

What if $X^T X$ is non-invertible?

- Redundant features (linearly dependent).

E.g. x_1 = size in feet²

$\underline{x_1}$ = size in m²

~~x_2~~

$$\underline{x_1} = (3.28)^2 \underline{x_2}$$

$$l_m = 3.28 \text{ feet}$$

$$\rightarrow \underline{m = 10} \leftarrow$$

$$\rightarrow \underline{n = 100} \leftarrow$$

$$\underline{\Theta \in \mathbb{R}^{101}}$$

- Too many features (e.g. $m \leq n$).

- Delete some features, or use regularization.

↓ later

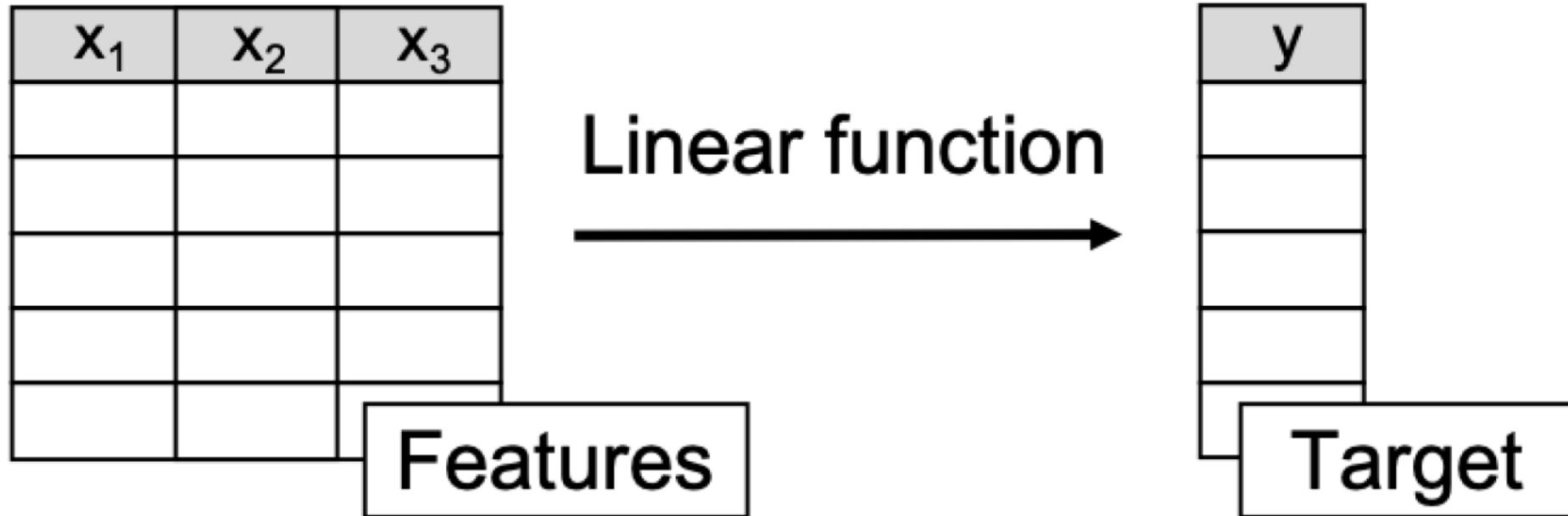
Gidia
Grupo de I+D
en Inteligencia Artificial



UNIVERSIDAD
NACIONAL
DE COLOMBIA

Regresión Lineal Multivariable

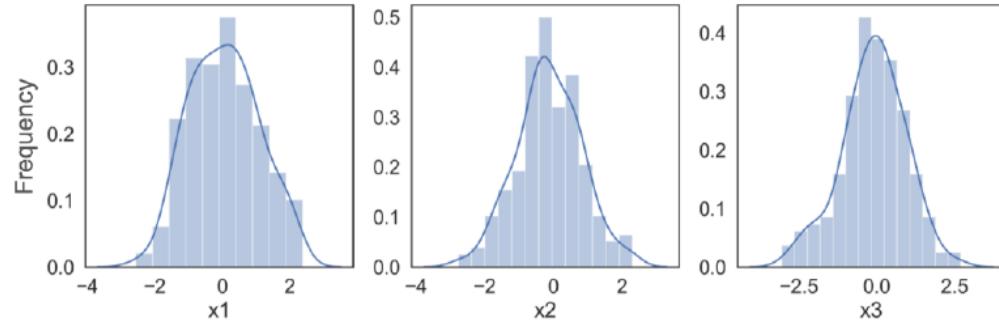
Linear model concept



Regresión Lineal Multivariable

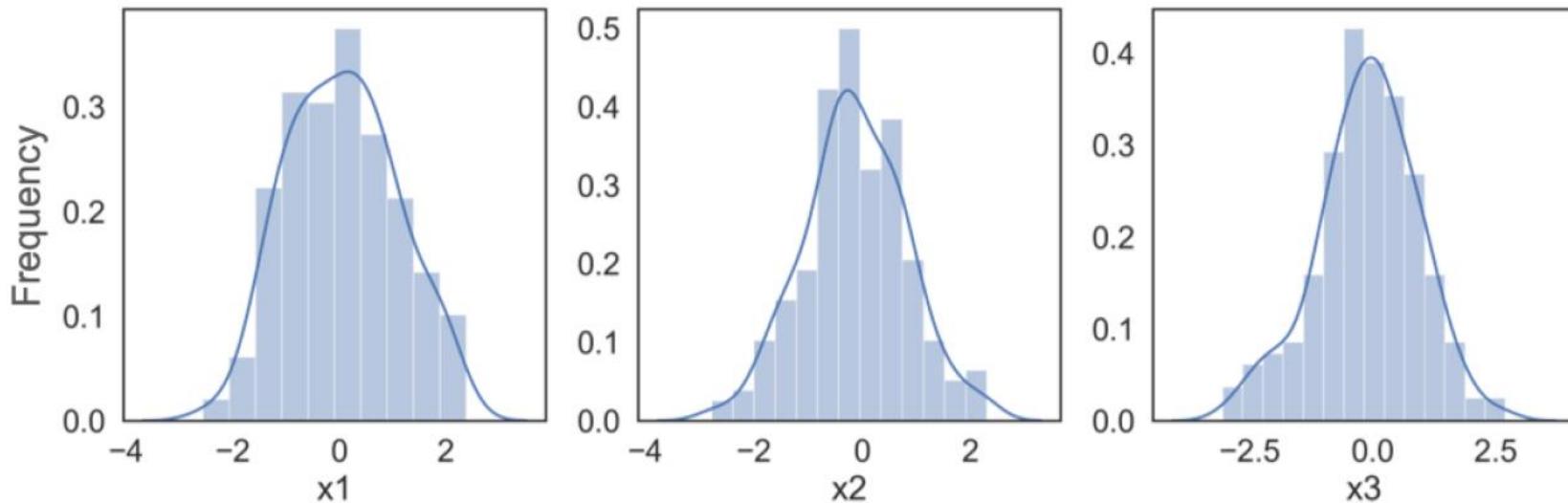
Creating our own dataset

x1	x2	x3
1.76	-0.37	-0.60
0.40	-0.24	-1.12
0.98	1.10	0.77
...



Regresión Lineal Multivariable

Creating our own dataset



Creating our own target feature:

$$y = 20 + 5x_1 + 2x_2 + 0x_3 + \text{error}$$

Regresión Lineal Multivariable en Python

```
from sklearn.linear_model import LinearRegression  
  
lr = LinearRegression()  
lr.fit(X_train, y_train)  
  
# Actual coefficients = [5 2 0]  
print(lr.coef_)
```

```
[ 4.95  1.83 -0.05]
```

```
# Actual intercept = 20  
print(lr.intercept_)
```

```
19.8
```

Regresión Lineal Multivariable en Python

```
# Calculates R-squared  
print(lr.score(X_test, y_test))
```

0.976

$$\text{R-Square} = 1 - \frac{\sum(Y_{\text{actual}} - Y_{\text{predicted}})^2}{\sum(Y_{\text{actual}} - Y_{\text{mean}})^2}$$

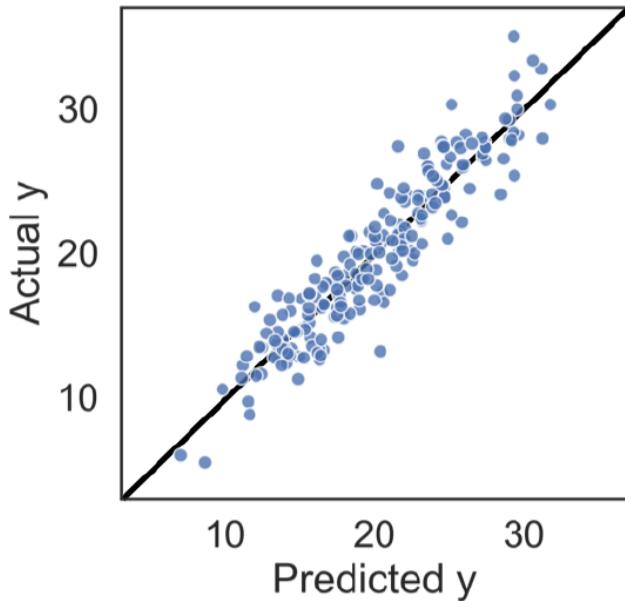
Regresión Lineal Multivariable en Python

```
from sklearn.linear_model import LinearRegression  
  
lr = LinearRegression()  
lr.fit(X_train, y_train)  
  
# Actual coefficients = [ 5 2 0 ]  
print(lr.coef_)
```

```
[ 4.95  1.83 -0.05]
```

Regresión Lineal Multivariable en Python

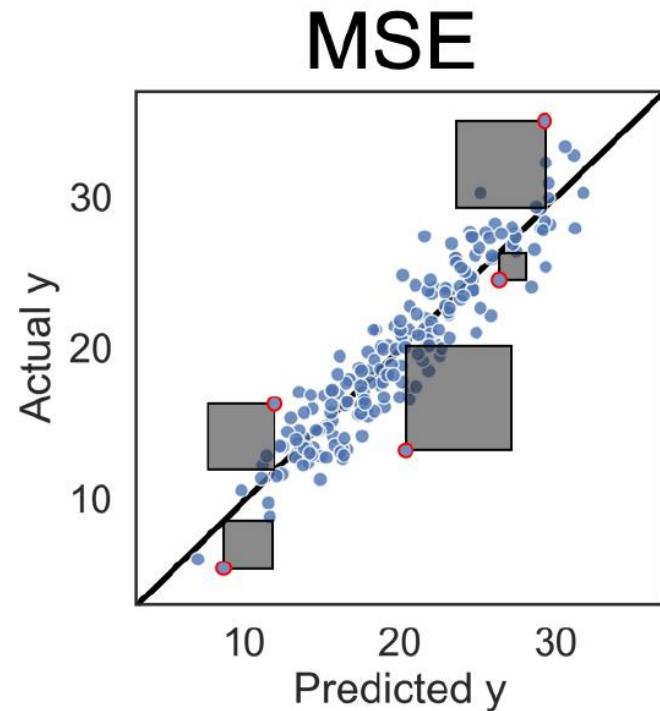
Loss function: Mean Squared Error



$$J(\theta_0, \theta_1, \dots, \theta_n) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2$$

Regresión Lineal Multivariable en Python

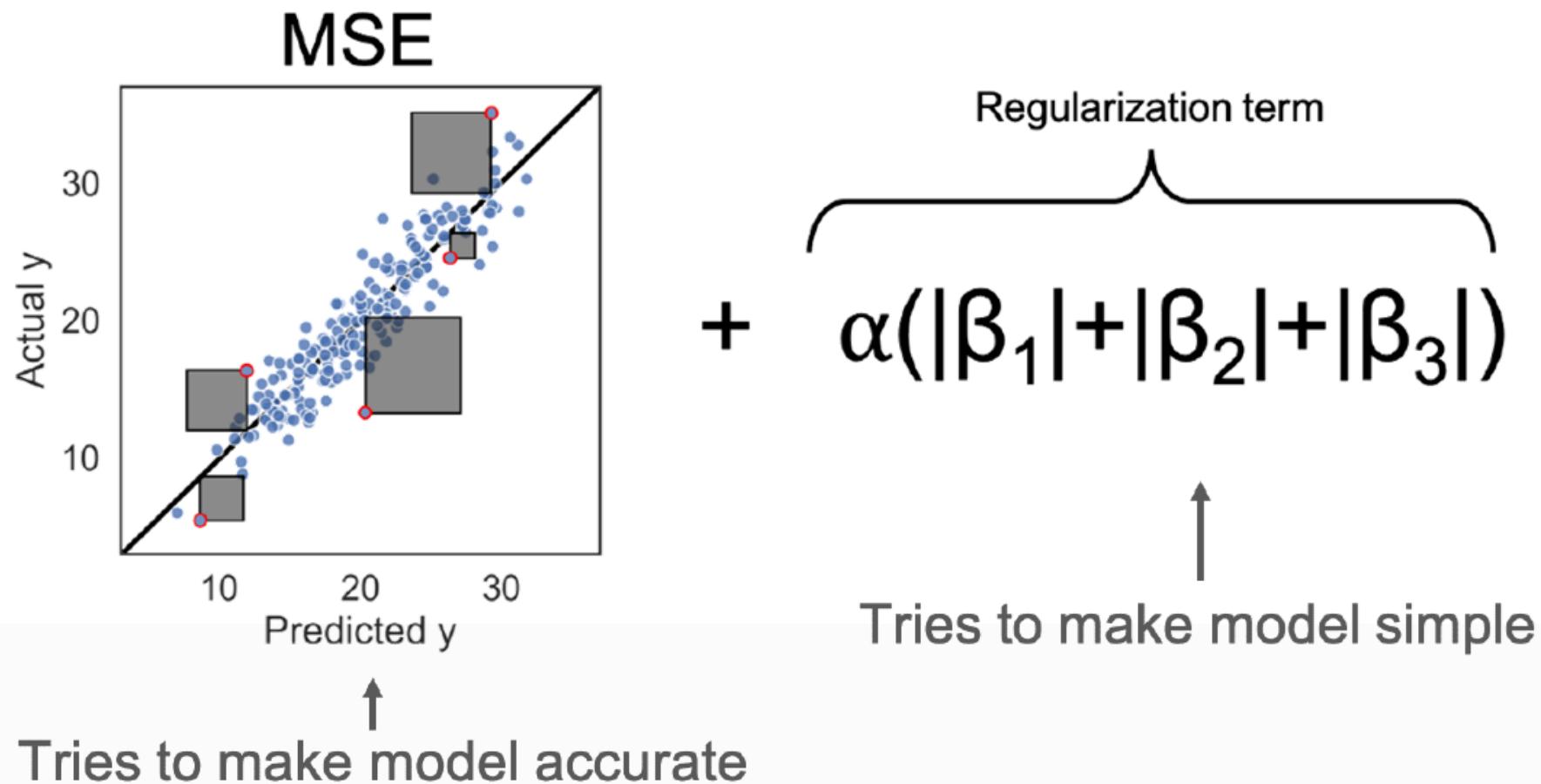
Loss function: Mean Squared Error



$$J(\theta_0, \theta_1, \dots, \theta_n) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2$$

Regresión Lineal Multivariable en Python - Regularizando

Adding regularization



¹ alpha, when it's too low the model might overfit, when it's too high the model might become too simple and inaccurate. One linear model that includes this type of regularization is called Lasso, for least absolute shrinkage and Selection operator

Regresión Lineal Multivariable en Python - Regularizando

Adding regularization

$$J(\theta) = \frac{1}{2m} \left[\sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2 + \alpha \sum_{j=1}^n \theta_j^2 \right]$$

Regularization

¹ alpha, when it's too low the model might overfit, when it's too high the model might become too simple and inaccurate. One linear model that includes this type of regularization is called Lasso, for least absolute shrinkage and Selection operator

Regresión Lineal Multivariable en Python - Regularizando Lasso regressor

```
from sklearn.linear_model import Lasso

la = Lasso(alpha=0.05)
la.fit(X_train, y_train)

# Actual coefficients = [5 2 0]
print(la.coef_)
```

```
[ 4.91  1.76  0. ]
```

```
print(la.score(X_test, y_test))
```

```
0.974
```

Combinando varios selectores en Python

LassoCV regressor

```
mask = lcv.coef_ != 0  
  
print(mask)
```

```
[ True True False ]
```

```
reduced_X = X.loc[:, mask]
```

Combinando varios selectores en Python

Recordemos:

- Así como los bosques aleatorios (Random Forest) son una combinación de árboles de decisión
- Podemos hacer también una combinación de modelos para la selección de características

Combinando varios selectores en Python

Combining the feature selectors

```
import numpy as np

votes = np.sum([lcv_mask, rf_mask, gb_mask], axis=0)

print(votes)
```

```
array([3, 2, 2, ..., 3, 0, 1])
```

```
mask = model_votes >= 2

reduced_X = X.loc[:, mask]
```

Laboratorio 3

<https://colab.research.google.com/drive/1Zlahrce5dIIHAOEKIhhrgobmv11dgaOb?authuser=2>



Seleccionando Características para mejorar el desempeño

Este Laboratorio es Basado en el Curso de Dimensionality Reduction de DATACAMP®

En este laboratorio implementaremos técnicas para granatizar el desempeño de nuestro modelo de reconocimiento de patrones, apicando técnicas de selección de características.

FeatureSelection

Contruyendo un clasificador para detectar diabetes

En este laboratorio se utilizará el conjunto de datos sobre diabetes de la tribu indígena Pima para predecir si una persona tiene diabetes mediante regresión logística. Hay 8 características y un objetivo (Y) en este conjunto de datos. Los datos se han dividido en un conjunto de entrenamiento y otro de prueba. A continuación se cargarán como X_train, y_train, X_test, y_test.

```
In [136]: # import required Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
X_train=pd.read_csv("Pima_X_train.csv")#reading a dataset in a dataframe using pandas
print("X_train",X_train.shape)

X_test=pd.read_csv("Pima_X_test.csv")#reading a dataset in a dataframe using pandas
print("X_test",X_test.shape)

y_train = X_train['result']
print("y_train",y_train.shape)

y_test = X_test['result']
print("y_test",y_test.shape)

X_train = X_train.drop('result', axis=1)
print("X_train droped",X_train.shape)

X_test = X_test.drop('result', axis=1)
print("X_test droped",X_test.shape)
```

Gracias !!!



© Man Bouncing Question Mark Towards Doctor - Artist: [Art Glazer](#)