SolidQ

Think Big. Move Fast.

Módulo 3: Entendiendo los algoritmos de Machine Learning

Primera Parte

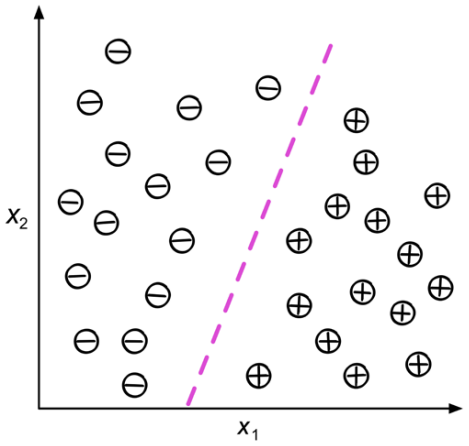
Agenda

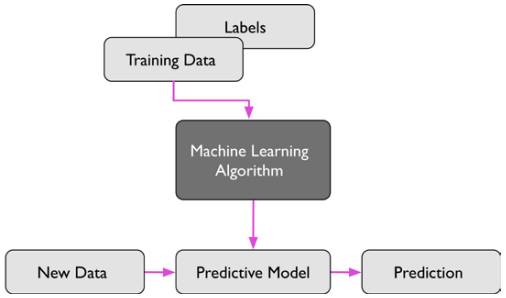
- Introducción a los algoritmos
- Datos de Entrenamiento y Prueba
- Clasificación
- Regresión
- Clustering
- Anomalías
- Recomendaciones
- Conjuntos de datos No-balanceados
- Como interpretar modelos

Introducción a Algoritmos

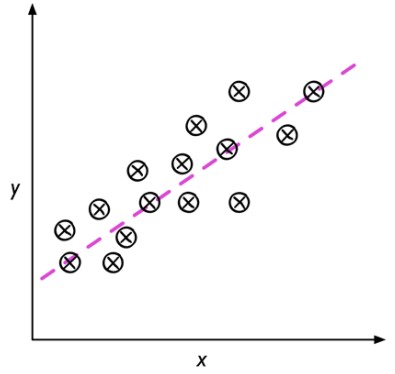
Haciendo predicciones con aprendizaje supervisado

- Clasificación para predecir etiquetas de clases
  - Binaria
  - Multiclase



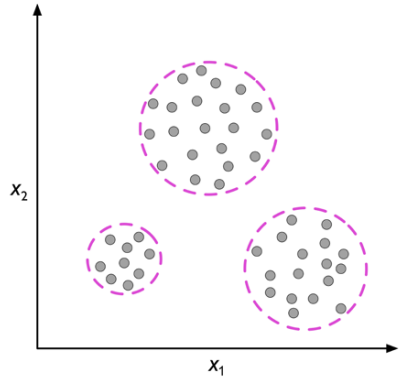


- Regresión para predecir salidas continuas



Descubriendo estructuras ocultas con aprendizaje no supervisado

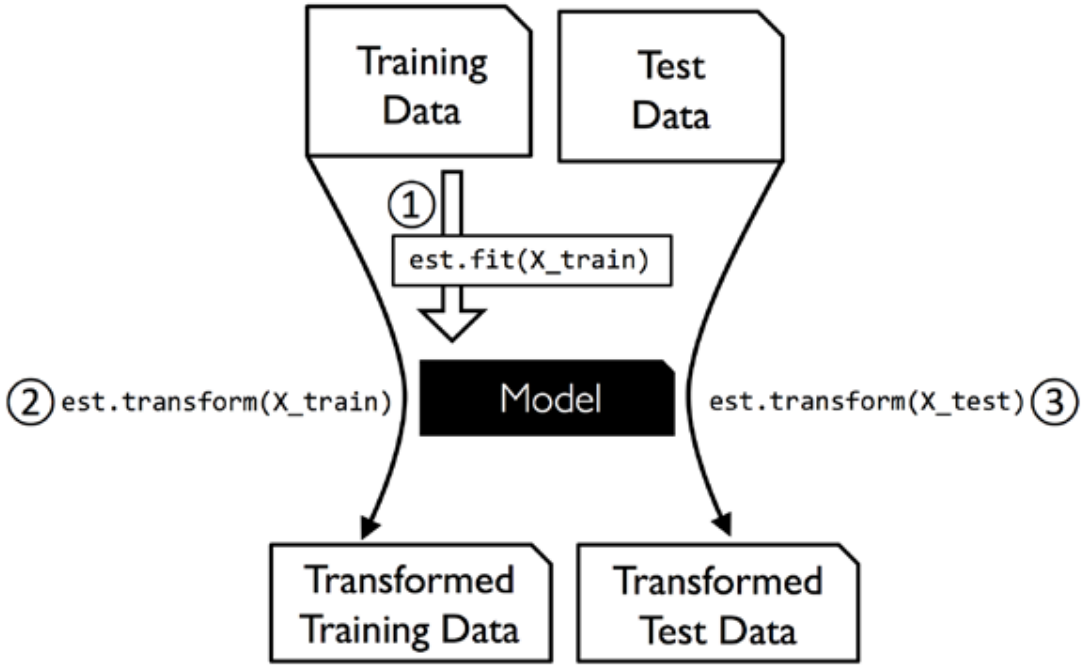
- Encontrando subgrupos con Clustering



- Reducción de la dimensionalidad para compresión de datos

Demo 03 – A: El primer algoritmo

- Scikit-learn



Datos de Entrenamiento y Prueba

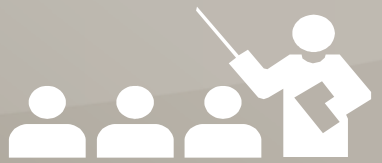
¿Por qué?

- Debemos de dejar datos que algoritmo no conozca para poder evaluar su rendimiento
- Debemos de intentar no introducir sesgos a la hora de realizar esta división
- Tercer grupo: Datos de seguridad
  - Entreno el modelo con un conjunto de datos de entrenamiento
  - Pruebo con el conjunto de datos de prueba
  - Entreno el modelo con datos de entrenamiento + prueba
  - Compruebo rendimiento con Datos de seguridad

¿Cómo?

- Submódulo sklearn.model_selection
- train_test_Split
  - Validación de entrada
  - next(ShuffleSplit().split(X, y))

Demo 03- B Train y Test



Dividiendo nuestro conjunto de datos

Regresión

Regresión

- Para predecir resultados con valores reales:
  - ¿Cuántos clientes llegarán a nuestro sitio web la próxima semana?
  - ¿Cuántos televisores venderemos el año que viene?
  - ¿Podemos predecir los ingresos de alguien desde sus clics de navegación a través de la información?

Regresión

- Cada observación se representa como un conjunto de números

Representamos una persona :

[	5	3	120	12	1	0 …..	]
[	0	0	89	5	1	1 …..	]
[	1	0	20	0	0	1 …..	]
		:					

Income

84
32
-10
:

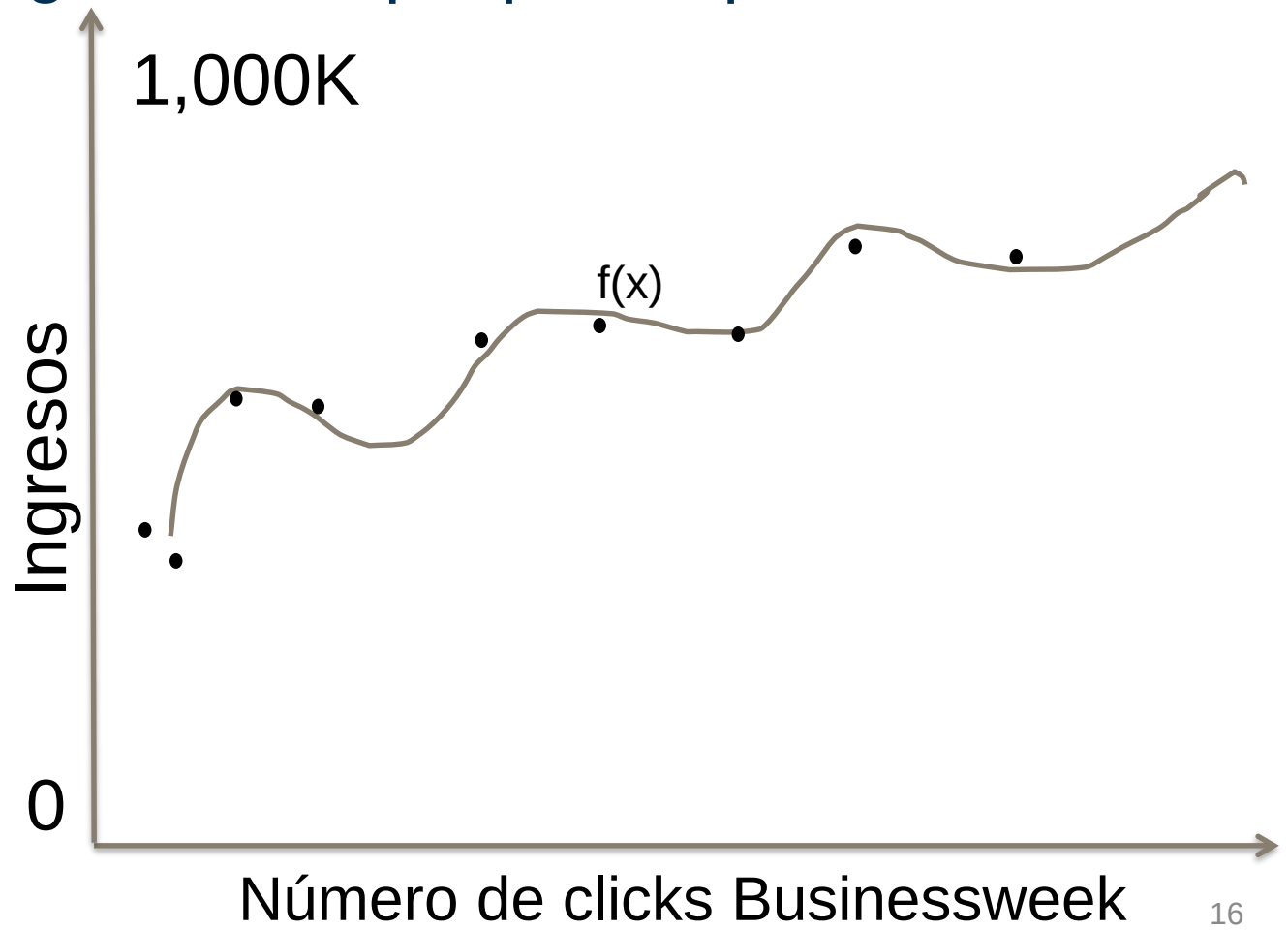
Características, llamadas X

Etiquetas, llamdas Y

Regresión

- Formalmente, dado un conjunto de entrenamiento (x_i, y_i) para i=1…n, queremos crear un modelo de regression f que pueda predecir la etiqueta y para un nuevo x

f(x) = function(Número de clics Businessweek)

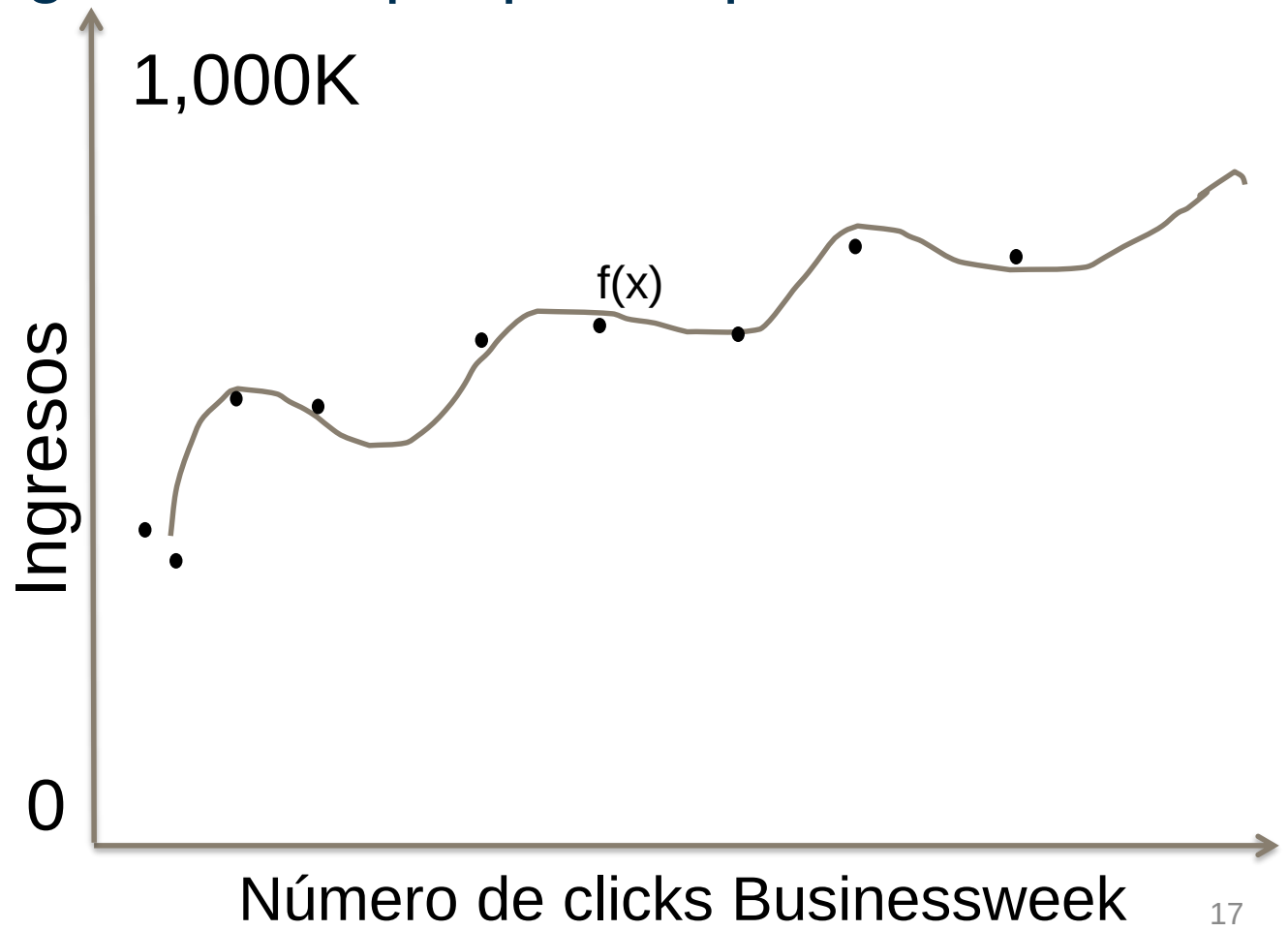


Regresión

- Formalmente, dado un conjunto de entrenamiento (x_i, y_i) para i=1…n, queremos crear un modelo de regression f que pueda predecir la etiqueta y para un nuevo x

f(x) = function(Número de clics Businessweek)

(Overfitting?)

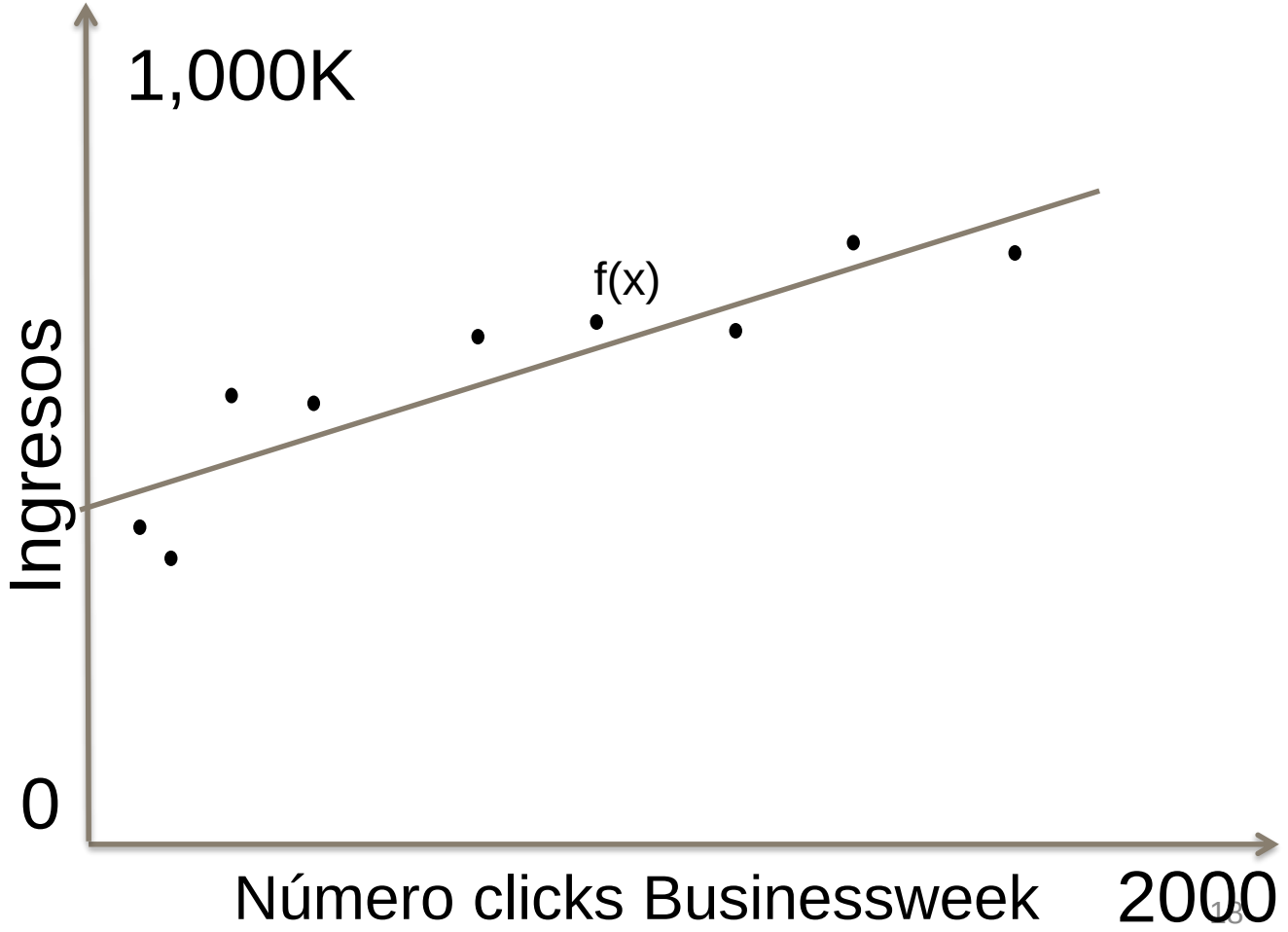


Regresión

- Formalmente, dado un conjunto de entrenamiento (x_i, y_i) para i=1…n, queremos crear un modelo de regression f que pueda predecir la etiqueta y para un nuevo x

f(x) = function(Número de clics Businessweek)
= 5K*Número clicks Businessweek + 100K

(Underfitting?)



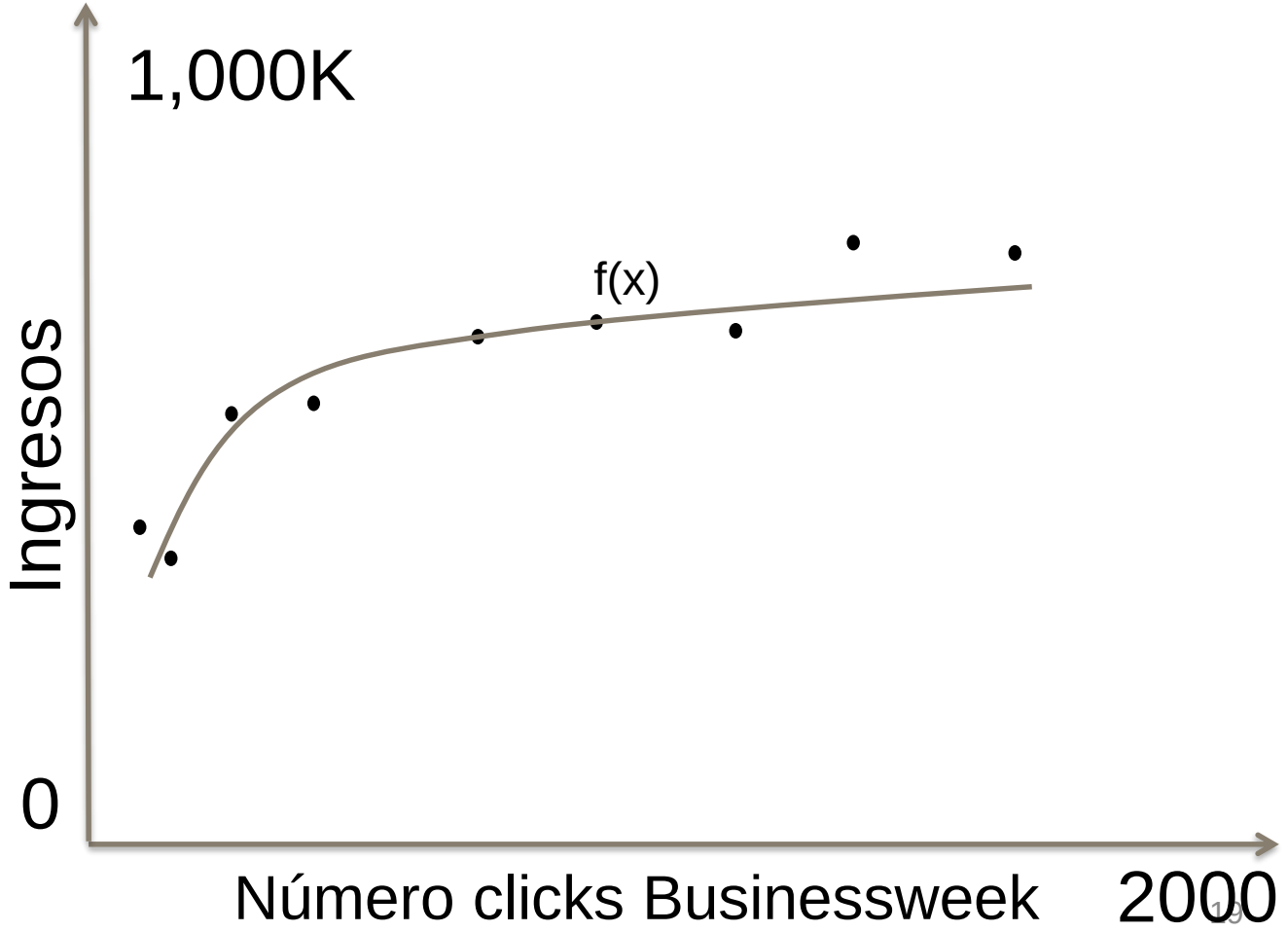
Regresión

- Formalmente, dado un conjunto de entrenamiento (x_i, y_i) para i=1…n, queremos crear un modelo de regression f que pueda predecir la etiqueta y para un nuevo x

f(x) = function(Number of Businessweek clicks)

Correcto?

Hablermos más tarde sobre ello



Regresión

- Formalmente, dado un conjunto de entrenamiento (x_i, y_i) para i=1…n, queremos crear un modelo de regression f que pueda predecir la etiqueta y para un nuevo x

Estimated income:

f(x) = function(Number of visits to upscale furniture websites, Number of Businessweek clicks, Number of distinct people emailed per day, Number of purchases of over 5K within the last month, Number of visits to airlines, etc.)

For instance,

f(x) = 3*Number of visits to upscale furniture websites
+10*Number of Businessweek clicks
+100*Number of distinct people emailed per day
+2*Number of purchases of over 5K within the last month
+10*Number of visits to airlines

But f(x) could be much more complicated

Aplicaciones de la Regresión

- Predecir cantidades monetarias
- Predecir el consumo o demanda de productos / energía

Algoritmos de Regresión

- **Regresión Lineal**
- **K-nn**
- Random Forest

Regresión Lineal

- Es un modelo lineal
  - Suma de variables ponderadas que predice un valor de salida a partir de una instancia de entrada
  - Hallar una línea recta que mejor encaje en un conjunto dado de datos
- Instancia de entrada – vector de características:

$$\boldsymbol{x} = (xx_0, xx_1, \ldots, xx_{nn})$$

- Salida de la predicción: $\hat{y} = \widehat{w_0} x_0 + \widehat{w_1} x_1 + \cdots \widehat{w_n} x_n + \hat{b}$
- Parámetros a estimar:

$\widehat{\boldsymbol{w}} = (\widehat{w_0}, \cdots, \widehat{w_n})$: feature weights/ model coefficients

$\widehat{\boldsymbol{b}}$: constant bias term / intercept

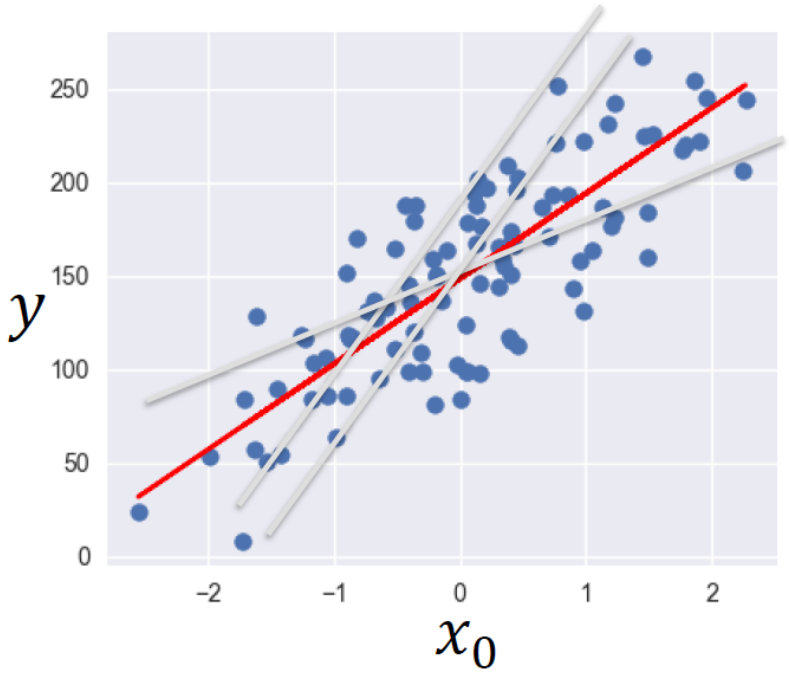
Regresión Lineal

- Modelo con una única variable:

Input instance: $\boldsymbol{x} = (x_0)$

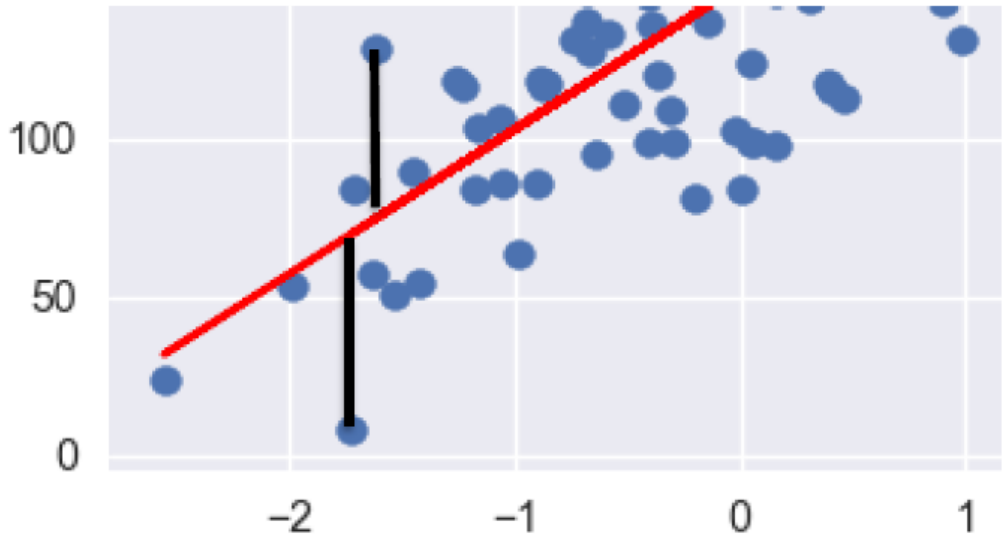
Predicted output: $\hat{y} = \widehat{w_0} x_0 + \hat{b}$

Parameters to estimate: $\widehat{w_0}$ (slope), $\hat{b}$ (y-intercept)



Regresión Lineal *"Least Squares"*

- *Ordinary Least Squares*
- Encontrar *w* y *b* que minimice la media al cuadrado del error del modelo lineal: la suma de las diferencias entre el objetivo predicho y los valores objetivos actuales al cuadrado
- No tenemos parámetros para controlar la complejidad del modelo



Regresión lineal

- ¿Cómo se estiman los parámetros w y b?
  - Se estima de los datos de entrenamiento
  - Existen diferentes modos de estimarlos
    - Los diferentes modos dependen de diferentes criterios “fit” y los objetivos y modos de controlar la complejidad del modelo
  - El algoritmo encuentra los parámetros que optimizan una función objetiva, típicamente para minimizar alguna función de pérdida de los valores predichos versus los valores actuales
- RSS→ suma de las diferencias al cuadrado
  - *Mean Square Error*

Regresión lineal en Scikit-Learn

```
from sklearn.linear_model import LinearRegression

X_train, X_test, y_train, y_test =
      train_test_split(X_R1, y_R1, random_state = 0)

linreg = LinearRegression().fit(X_train, y_train)

print("linear model intercept (b): {}".format(linreg.intercept_))
print("linear model coeff (w): {}".format(linreg.coef_))
```

```
linear model coeff (w): [ 45.70870465]
linear model intercept (b): 148.44575345658873
R-squared score (training): 0.679
R-squared score (test): 0.492
```

linreg.coef_ linreg.intercept_

$$\hat{y} = \boldsymbol{w_0} x_0 + b$$

Demo 03 C- Algoritmos de Regresión

- Regresión Lineal

k-NN Regresor

- Necesita cuatro cosas
  - Una métrica de distancia
  - A cuántos vecinos mira?
  - Función de peso opcional en los puntos vecinos
  - Método para agregar las clases de los puntos vecinos

k-NN regresor

- Complejidad del modelo
  - Nº de vecinos: número de vecinos más cercanos (k) a considerar.
  - Por defecto 5
- Encaje del model (Model fitting)
  - Metrica: función de distancia entre dos puntos de datos
  - Por defecto euclídea p=2
- Cómo se agregan
  - Voto de mayoría simple (la clase con el mayor número de representantes entre los vecinos)

Evaluar Modelos de Regresión

- Score de regresión R^2
  - Mide como de bien un modelo basado en predicción encaja con los datos
  - El score es un valor entre 0 y 1
    - 0 corresponde con un modelo constante que predice el valor medio de todos los valores de entrenamiento
    - 1 corresponde a la predicción perfecta
  - Se conoce también como *Coeficiente de determinación*
- Mean Squared Error (MSE)
  - Valor medio de SSE(Sum of Squared Errors) que se minimiza para encajar el modelo
  - Útil para comparar diferentes modelos de regresión o para optimizar sus parámetros

Demo 03 C- Algoritmos de Regresión

- k-NN

Clasificación

Clasificación

- El objetivo es definir una *etiqueta de clase*
  - Se utiliza para identificar valores diferentes
  - Una opción de entre un conjunto finito
    - Binaria (intentando responder sí o no)
      - *Clase positiva / clase negativa*
    - Multiclase
- La etiqueta (valor a predecir) tiene un valor de -1 para *Falso* y 1 para *Verdadero*

Clasificación

- Cada observación se representa por un conjunto de números (características)

Una alcantarilla se representa: [5 3 120 12 1 0 …..] -1
[0 0 89 5 1 1 …..] 1
[1 0 20 0 0 1 …..] -1

Características, llamadas X

Etiquetas, llamadas Y

Clasificación

Alcantarilla se representa: [5 3 120 12 1 0 …..]

Nº eventos ultimo año
Nº eventos serios ultimo año
Nª cables eléctricos
Nºcables anteriores a 1930
Vented cover?
¿Inspeccionada?

Datos de 2014 y anteriores
la etiqueta es 1 si explotó en 2015

Definición formal

- Formalmente, dado un conjunto de entrenamiento, (xi,yi) para i=1…n, queremos crear un modelo de clasificación f que pueda predecir la etiqueta *y* para un nuevo *x*

Alcantarilla se representa por: [1925 15]

Año de instalación del cable más viejo

Nº eventos ultimo año

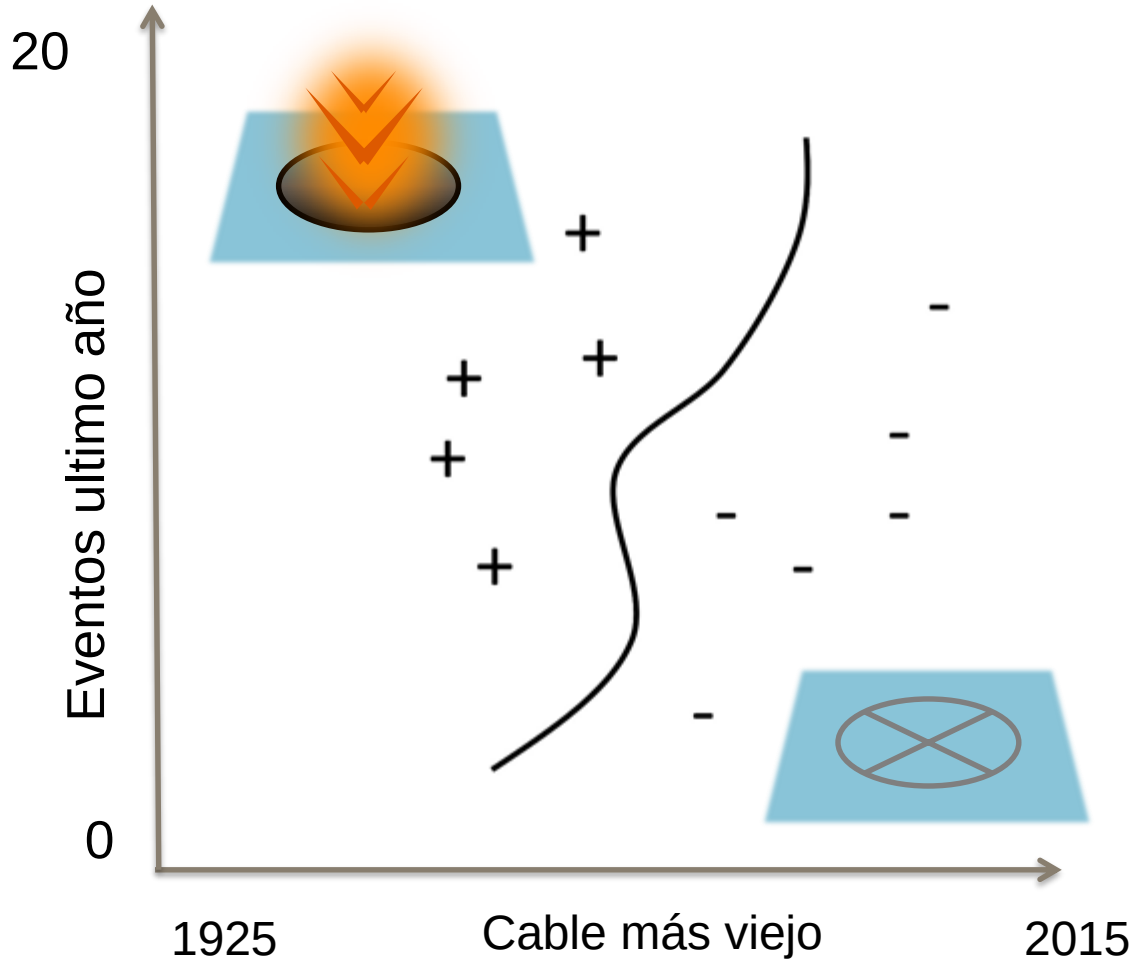
Definición formal

- Formalmente, dado un conjunto de entrenamiento, (xi,yi) para i=1…n, queremos crear un modelo de clasificación f que pueda predecir la etiqueta y para un nuevo x

Alcantarilla se representa por: [1925 15]

Año de instalación del cable más viejo

Nº eventos ultimo año



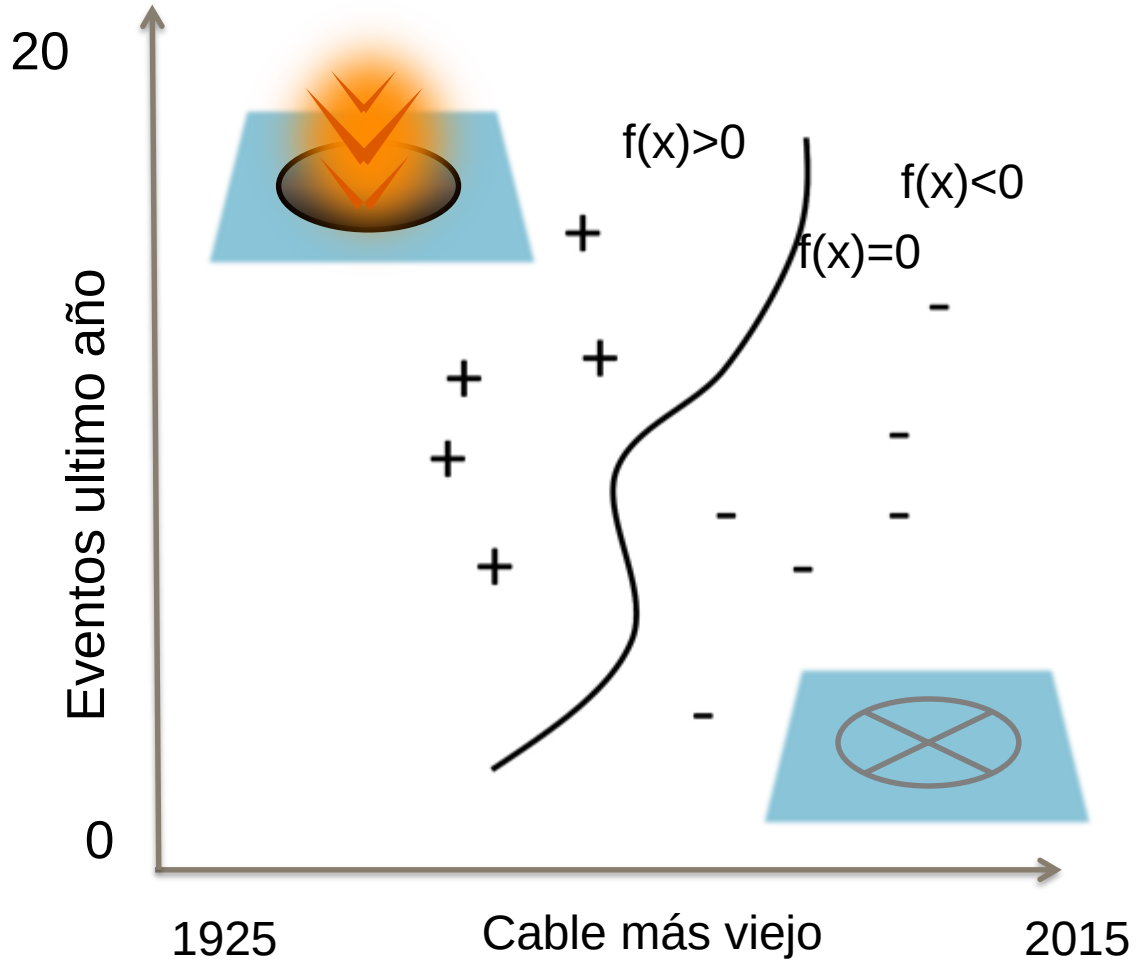
Definición formal

- Formalmente, dado un conjunto de entrenamiento, (xi,yi) para i=1…n, queremos crear un modelo de clasificación f que pueda predecir la etiqueta y para un nuevo x

Alcantarilla se representa por: [1925 15]

Año de instalación del cable más viejo

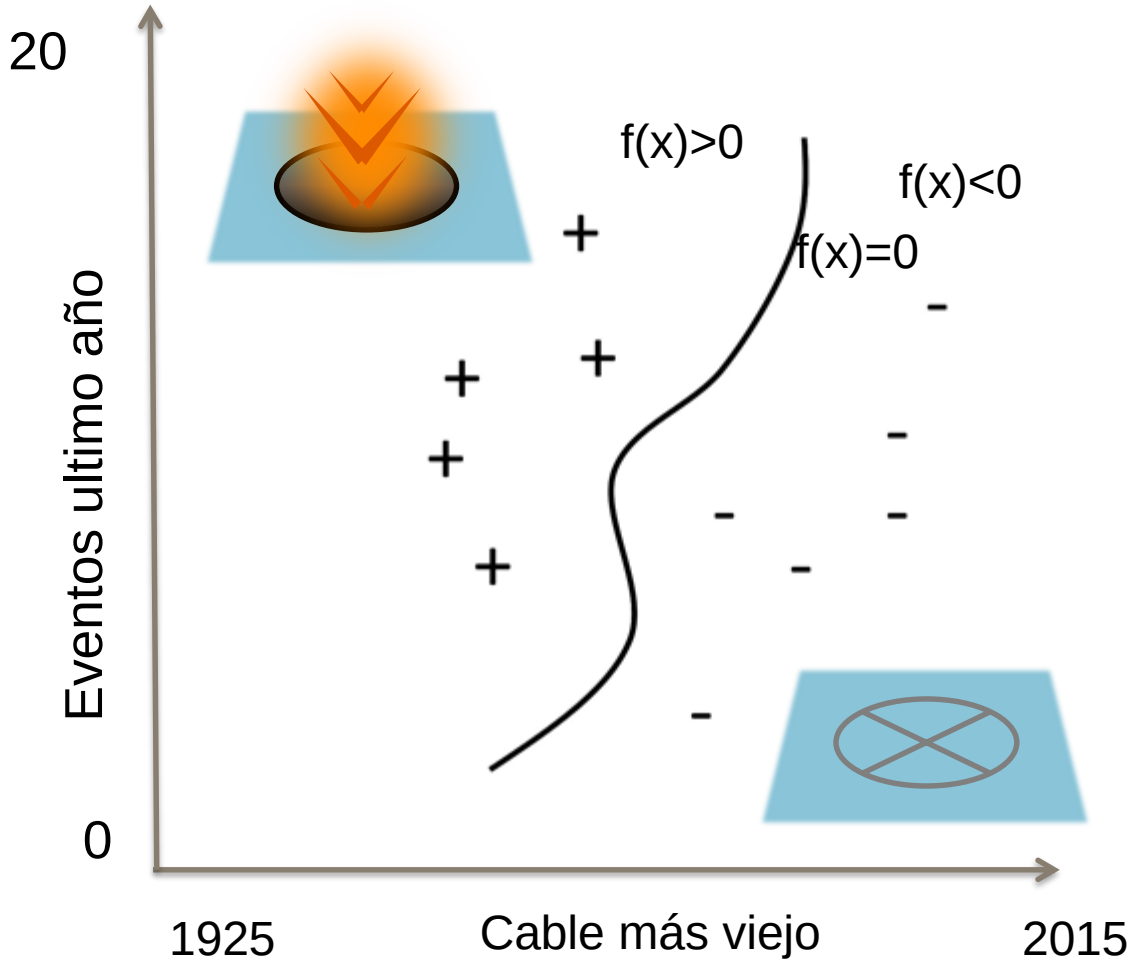
Nº eventos ultimo año



Definición formal

- Formalmente, dado un conjunto de entrenamiento, (xi,yi) para i=1…n, queremos crear un modelo de clasificación f que pueda predecir la etiqueta y para un nuevo x

f(x) = function(Eventos ultimo año, Cable más viejo)



Algoritmos de Clasificación

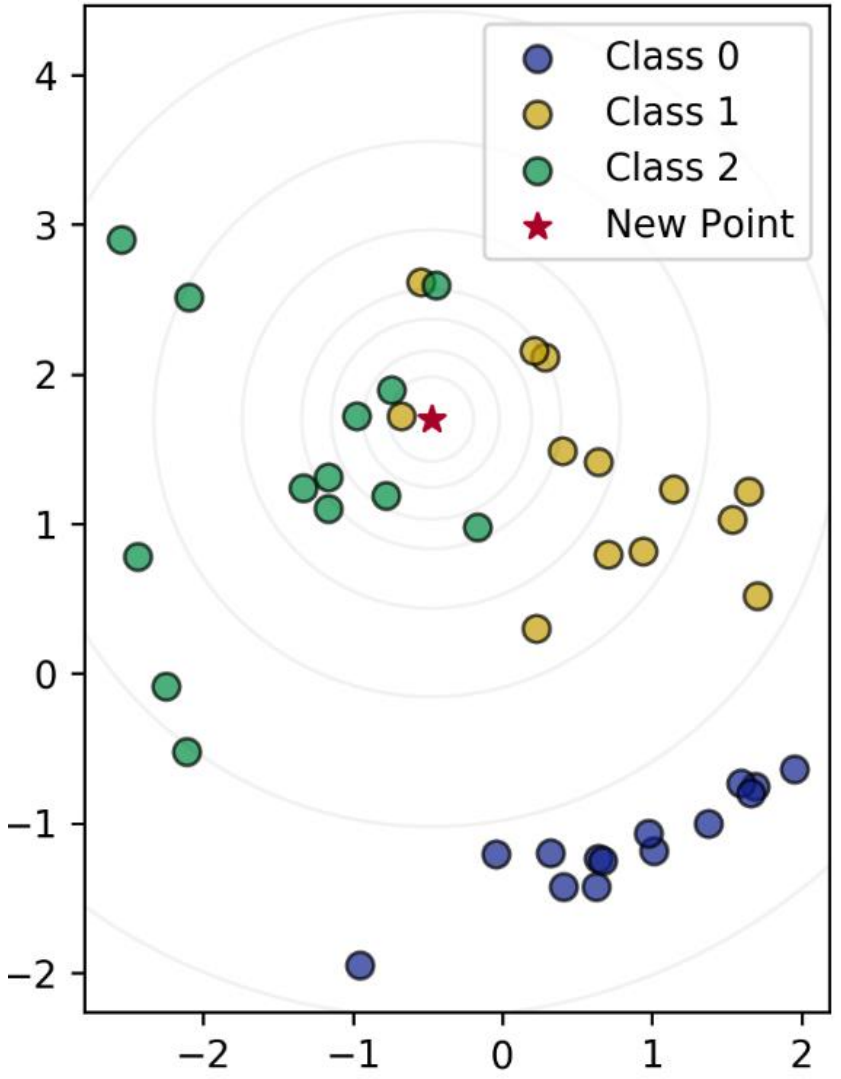
- Regresión Logística
- Naive Bayes
- SVM (Máquinas de vectores de soporte)
- **K-nn**
- **Árboles de Decisión**

k-NN Clasificador

- Dado un conjunto de entrenamiento X_train con etiquetas y_train, y teniendo una nueva instancia x_test para clasificar:
  1. Encontrar las instancias más parecidas (les llamaremos X_NN) ta x_test que están en X_train
  2. Obtener las etiquetas y_NN para las instancias X_NN
  3. Predecir la etiqueta para x_test combinando las etiquetas y_NN, por ejemplo a través de voto de mayoría simple

k-NN Clasificador

- Usando los datos para las clases 0,1 y 2 ¿qué clase asignará un clasificador k-NN a un nuevo punto para k=1 y para k=3?

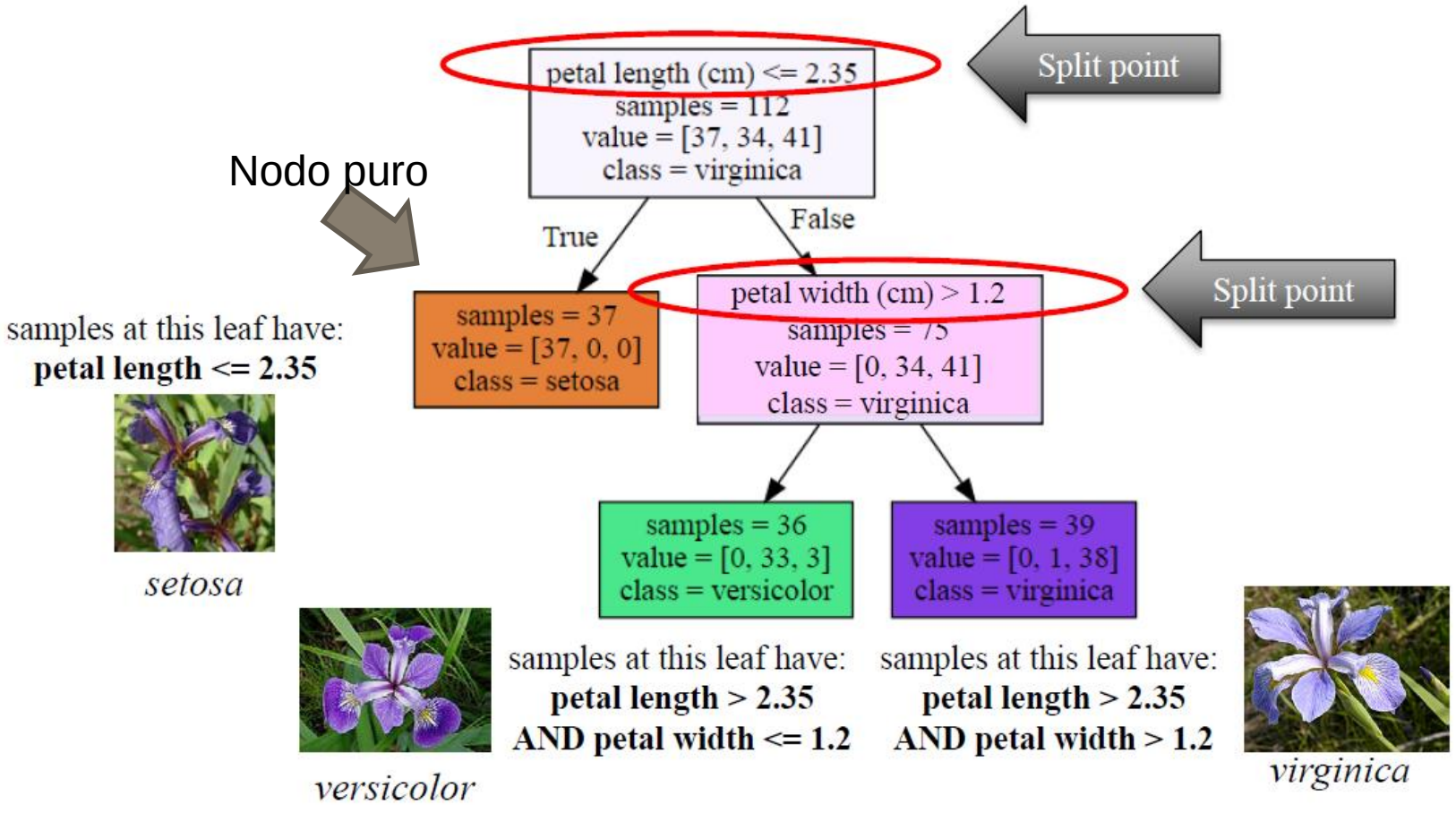


Árboles de Decisión

- ¿Por qué árboles?
  - Interpretables, intuitivos, modelos lógicos que mimetizan el razonamiento humano
  - Es no-lineal, por lo tanto más potente que los lineales
- Cómo se construyen?
  - Empezamos en la copa
  - Se hace crecer dividiendo las características una por una. Para hacer la división se mira “como de impuro” es el nodo
  - Se asignan nodos hoja por mayoría en la hoja
  - Al final, volvemos atrás y podamos hojas para reducir overfitting
- Necesitamos un conjunto de datos balanceado

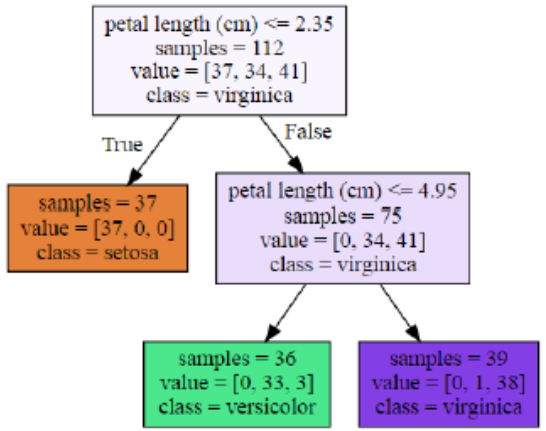
Ejemplo: el dataset iris

- 150 flores
- 3 especies
- 50 ejemplos / especie

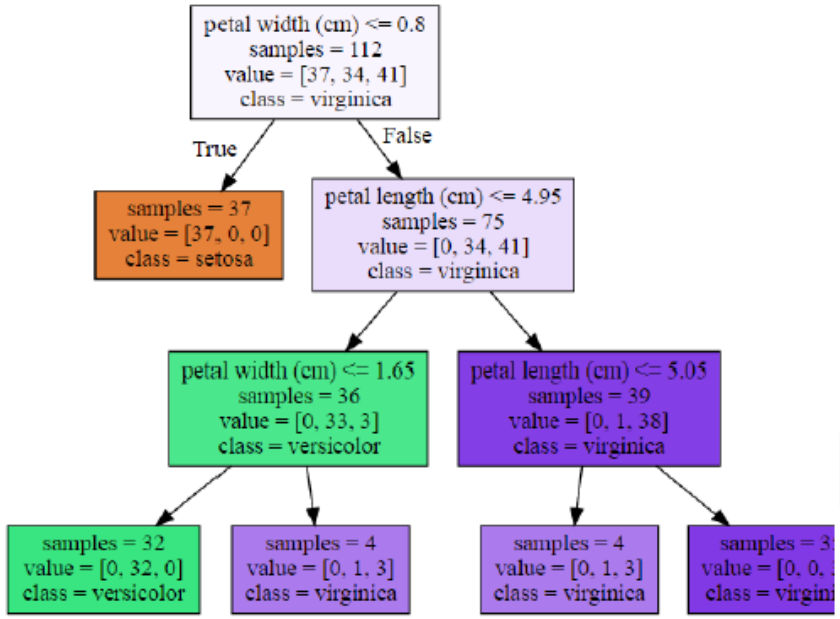


Árboles de decisión: Parámetros

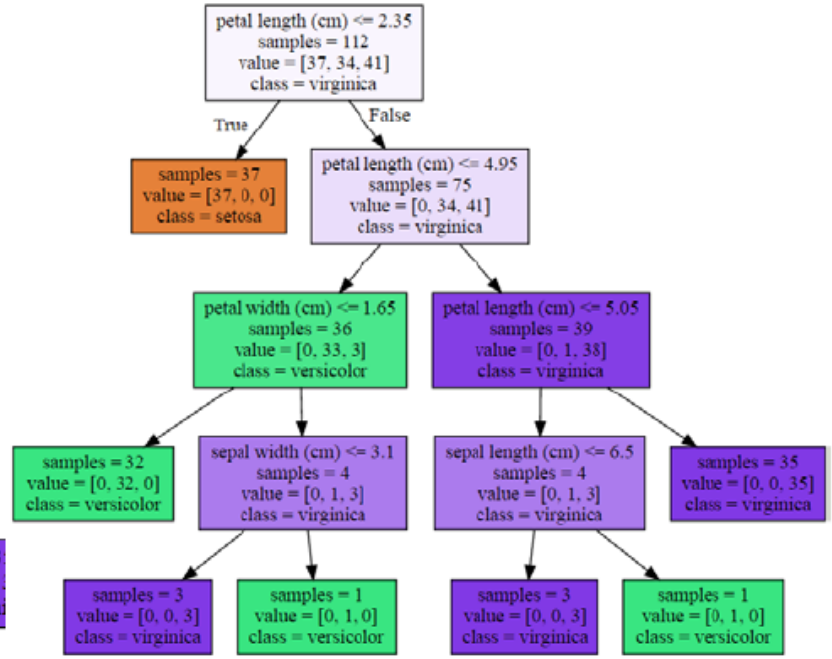
max_depth = 2



max_depth = 3



max_depth = 4



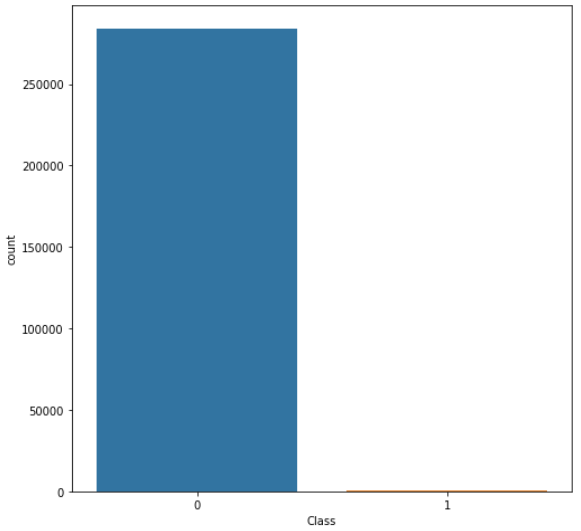
- Max_Depth
- Min_simples_leaf
- Max_simples_leaf

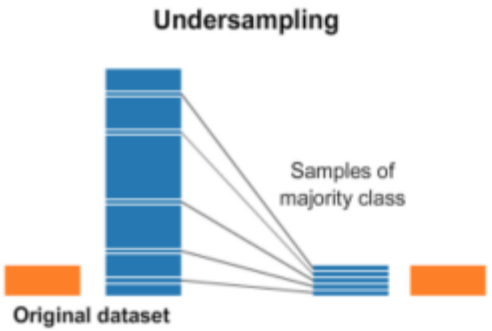
Demo 03 – D: Algoritmos de Clasificación

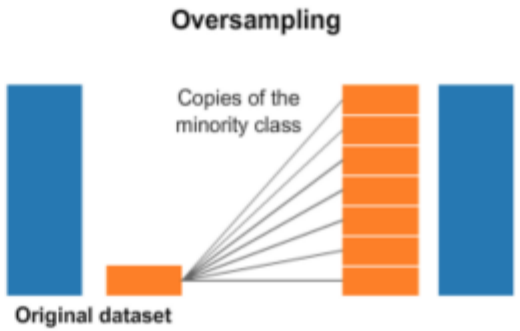
- k-NN Clasificador
- Árboles de Decisión

El gran problema de la Clasificación

- Datos no balanceados
  - Resampling
    - Oversampling: **SMOTE**
  - Métodos de ensamblado
    - Librería imblearn
      - **BalancedBaggingClassifier**







Aprendizaje Supervisado - Resumen

- "Supervisado" significa que los datos de entrenamiento tienen etiquetas de verdad para aprender de ellas. La clasificación y la regresión son problemas de aprendizaje supervisados.
- La clasificación (supervisada) a menudo tiene etiquetas + 1 o-1.
- La regresión (supervisada) tiene etiquetas numéricas.
- Los algoritmos de aprendizaje supervisado son mucho más fáciles de evaluar que los no supervisados.

Lab 03 Analizando los datos del Titanic

Analizando con Árbol de decisión

www.solidq.com

info@solidq.com