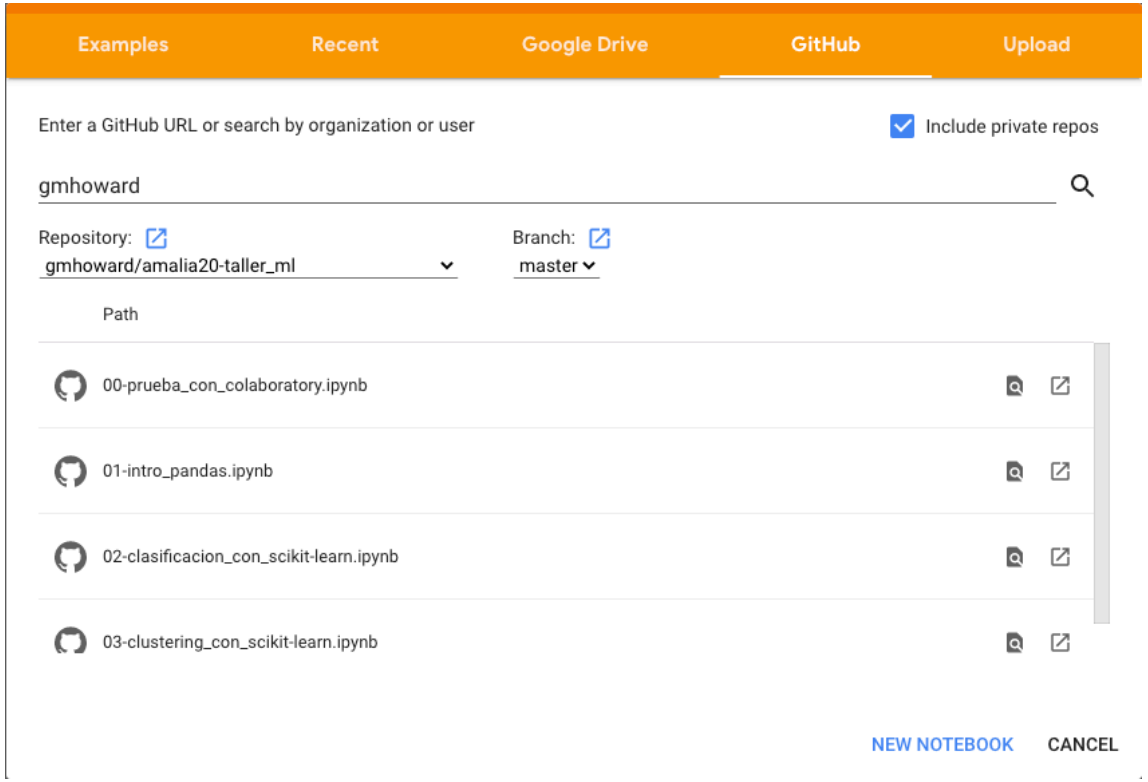
Estoy lista(o) para participar?

1. No hay necesidad de instalar ningún software en su computadora, sólo usarás tu navegador web
2. Visita Google Colab (https://colab.research.google.com/)
3. Baja el material del taller, disponible en el sitio de Github: https://github.com/gmhoward/amalia20-taller_ml



Taller de Ciencia de Datos y Machine Learning con Python

AMALIAconf 2020

Octubre 24, 2020

AMALIA CONF by COMUNIDAD DOJO

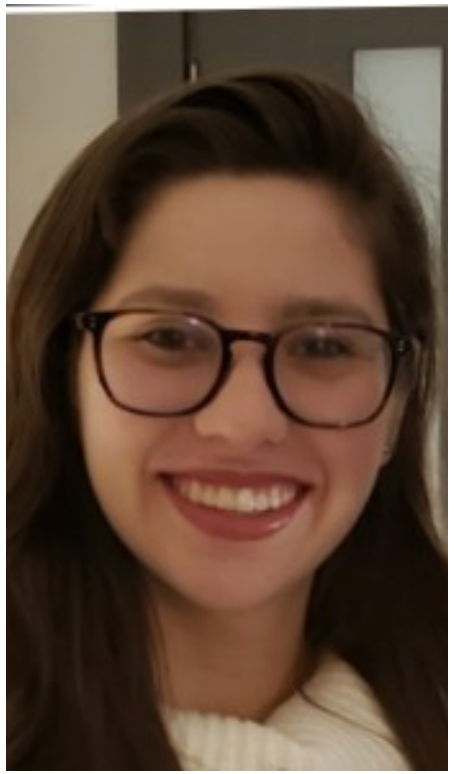
https://github.com/gmhoward/amalia20-taller_ml

Introducciones y Logística

Equipo del Taller

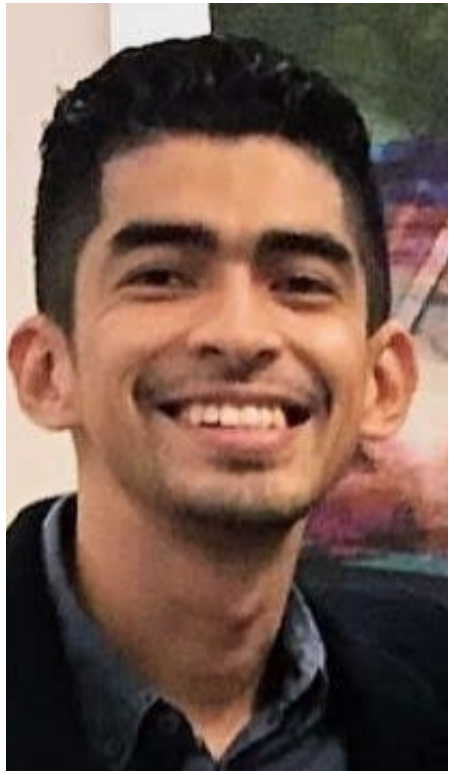
Gaspar

Adriaan



Eliana

Yiseika



Arnulfo

Agenda del Día

- Tiempos son estimados, haremos lo mejor posible

HORA	MODULO
9:00 AM	Introducción, objetivos del taller, logística, motivación
9:10 AM	Introducción a ML con Python (ecosistema PyData)
9:30 AM	Laboratorio 1: Análisis Exploratorio de Datos con Pandas
9:55 AM	Break
10:00 AM	Algoritmos Supervisados para ML (problemas de clasificación)
10:25 AM	Laboratorio 2: Clasificando flores y correo spam
10:55 AM	Break
11:00 AM	Algoritmos No Supervisados y Técnicas de Evaluación
11:25 AM	Laboratorio 3: Identificación de muestras (clustering) y evaluación de algoritmos
11:55 AM	Break
12:00 PM	Conclusiones, resumen

Expectativa

- Por favor participa y haz preguntas… escríbela en el chat room
- Sigue los ejemplos y ejercicios que se presenten en la clase
- Los laboratorios incluyen:
  - Código a ejecutar. Revísalo y juega con él.
  - Preguntas para repasar los temas cubiertos, intenta contestarlas
- No importa si tiene conocimientos o experiencia en ciencia de datos, hoy es un buen día para explorar el tema

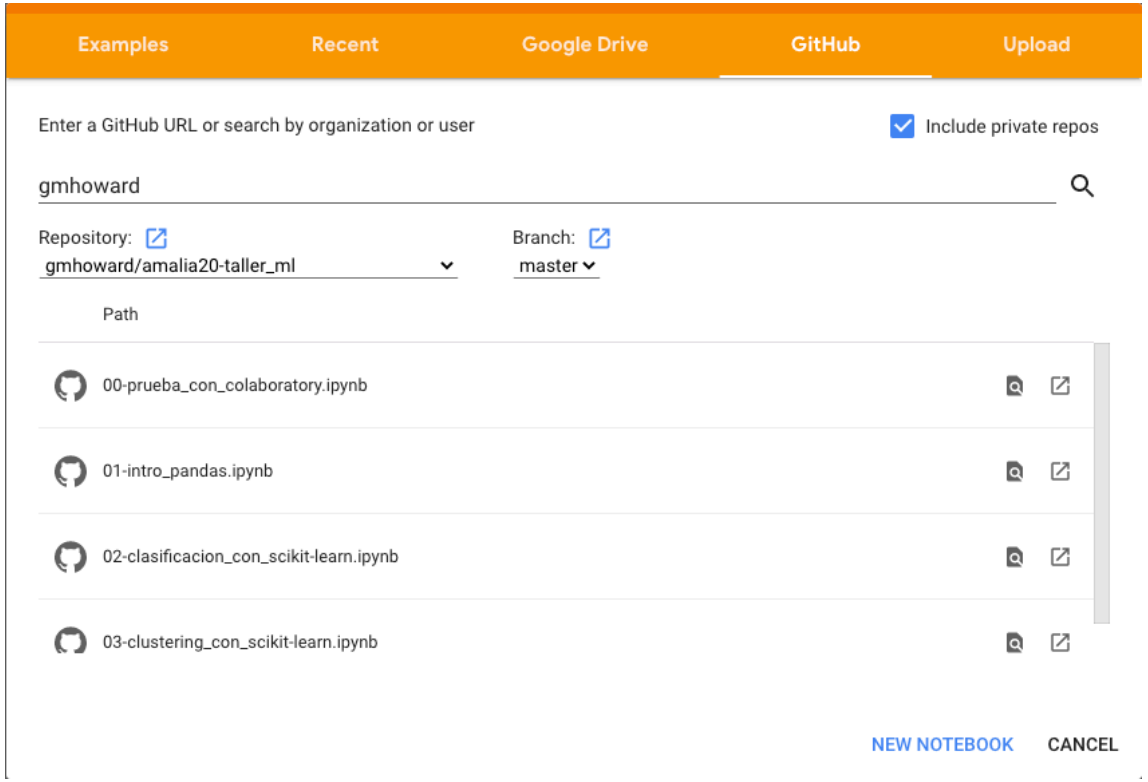
¿Por qué asistir al Taller?

- Introducción a la agenda de AmaliaConf
- Aumentar mi conocimiento para evaluar ofertas de trabajo que requieren aprendizaje de maquinas o ciencia de datos
- Conocimiento general, todo el mundo habla del tema
- Tengo muchos datos y quisiera hacer mi propio análisis
- Crear un sistema que me permita resolver un problema especifico

- Tiempo del taller es principalmente utilizado en ejercicios
- Mientras más haces, más aprendes!

¿Estoy lista(o) para participar?

1. No hay necesidad de instalar ningún software en su computadora, sólo usarás tu navegador web
2. Visita Google Colab (https://colab.research.google.com/)
3. Baja el material del taller, disponible en el sitio de Github: https://github.com/gmhoward/amalia20-taller_ml

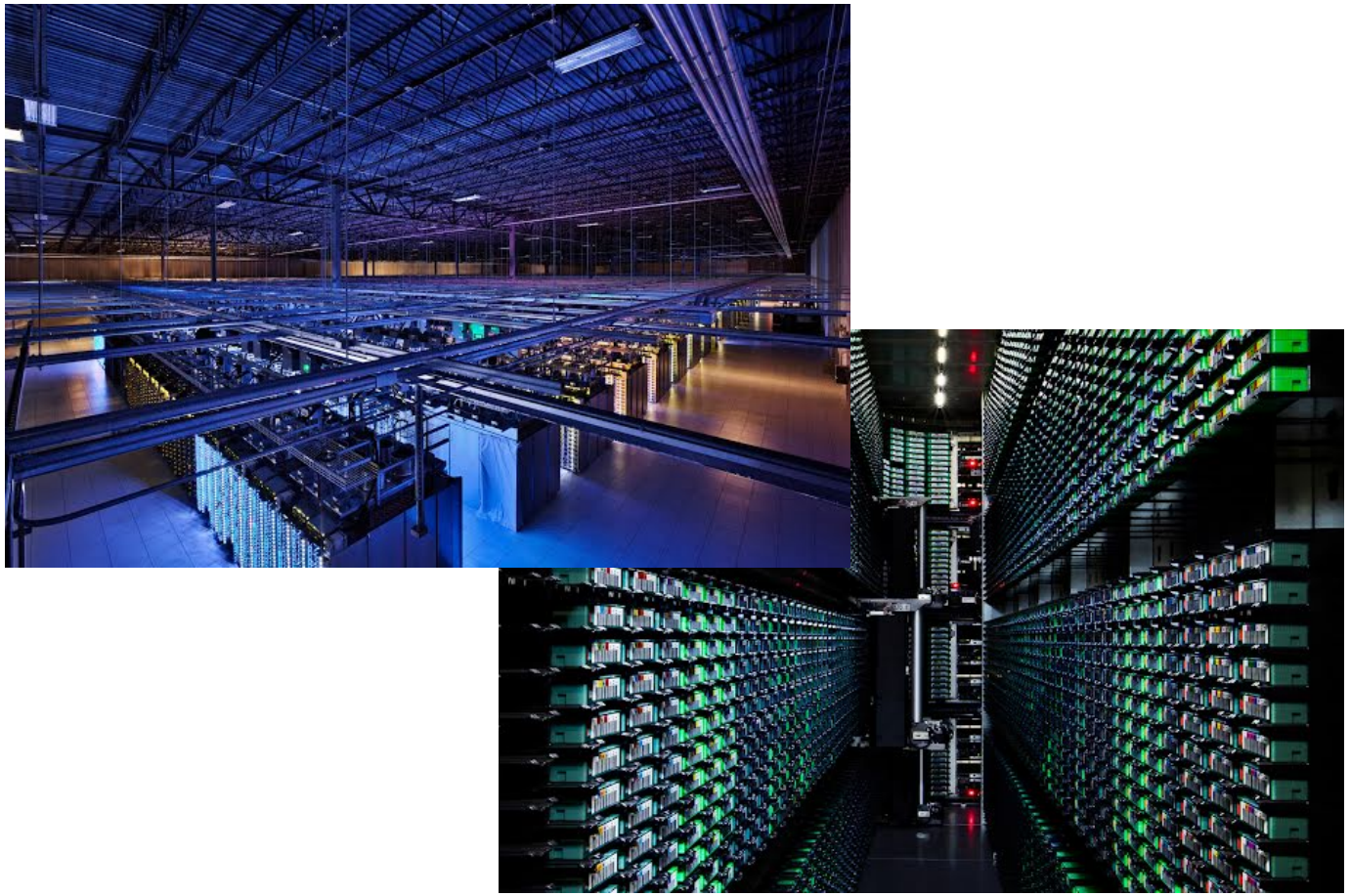


Introducción a Machine Learning y Ciencia de Datos

¿Cuánta data se produce al año?

16.3 ZB
(Zettabytes*)

* 1 Zettabyte = 1 trillion Gigabytes = 1 million Petabytes = 10^{21} bytes



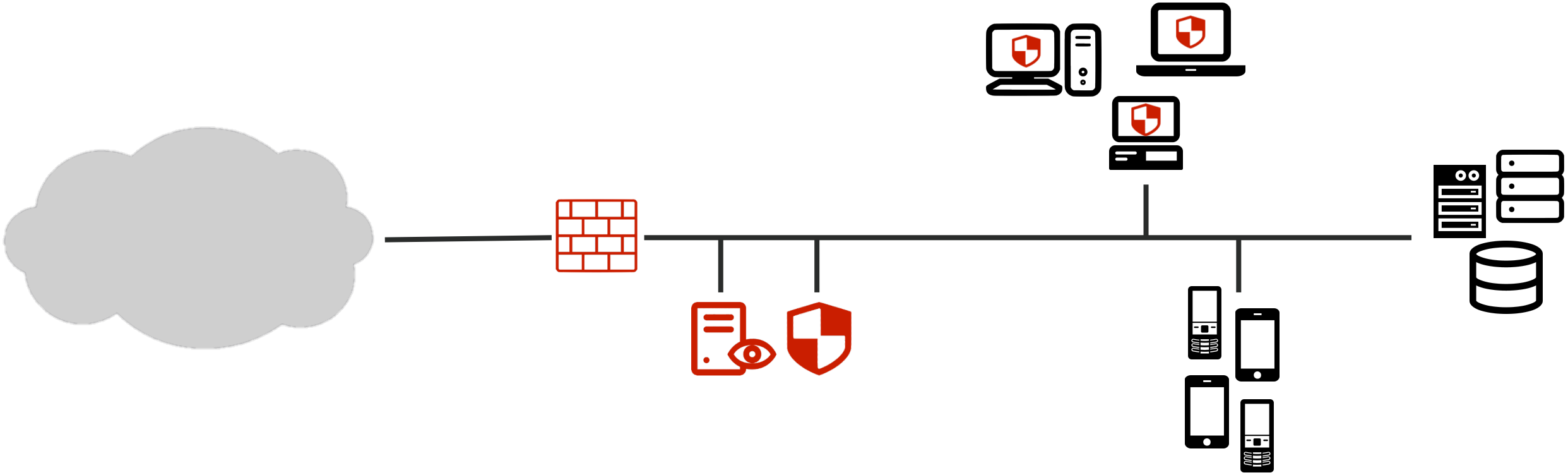
Centro de Datos de Google (Iowa y Carolina del Sur, EEUU)

¿Cuánta data puede almacenar un cerebro?

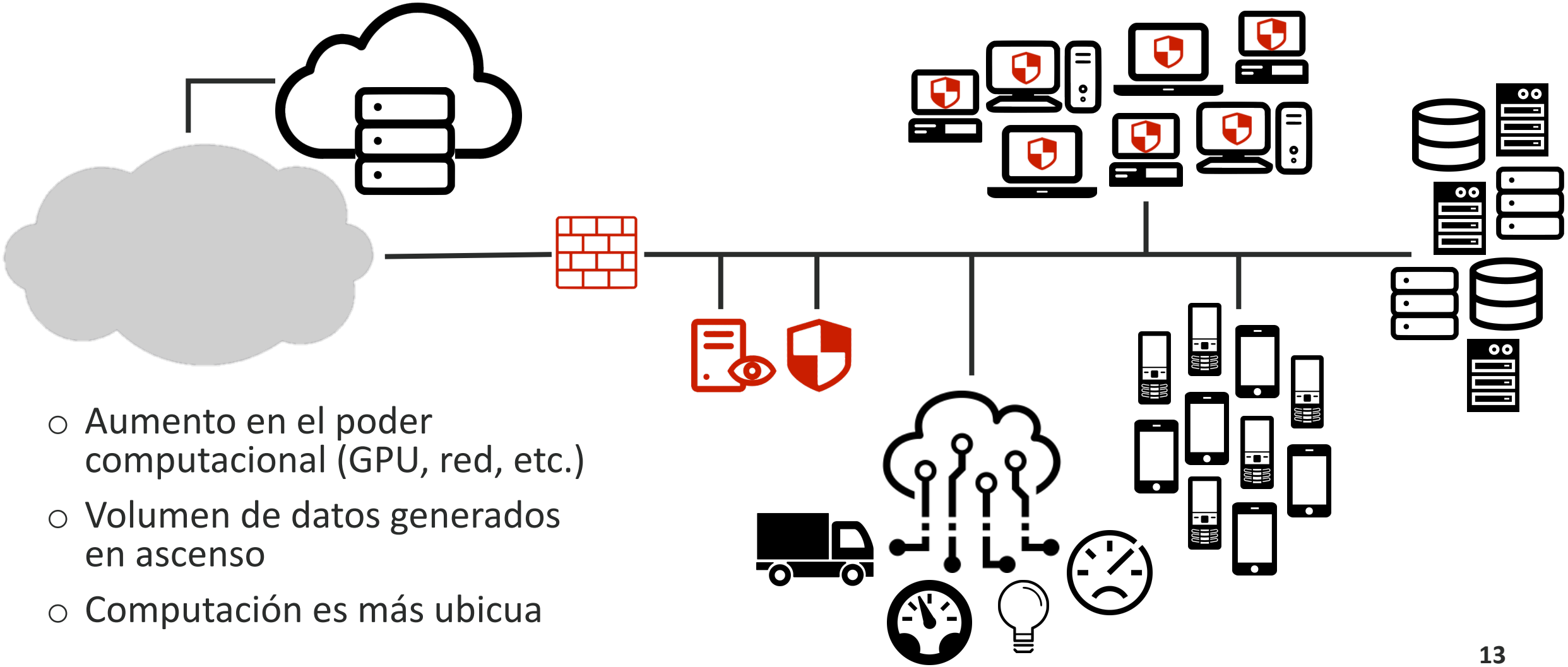
2.5 PB
(Petabytes)

1 Zettabyte = 1 trillion
Gigabytes = 1 million
Petabytes = 10^{21} bytes

Escenario Informático (2000s)



Escenario Informático (2020)



- Aumento en el poder computacional (GPU, red, etc.)
- Volumen de datos generados en ascenso
- Computación es más ubicua

Escenario Informático (2020s)

Data Science al rescate!

- Amenazas creciendo en costo, complejidad y severidad
- Frecuencia con la cual se necesita realizar un análisis ad-hoc también en aumento
- Reportes estandarizados, aunque útiles, no son suficientes
- **¿Cómo analizar el diluvio de datos? Imposible hacerlo de forma manual**

¿Qué es Aprendizaje de Máquinas (ML)?

- Usemos un ejemplo: ¿cómo identificar una cara?

¿Qué es Aprendizaje de Máquinas (ML)?

- Usemos un ejemplo: ¿cómo identificar una cara?

Irving Saladino

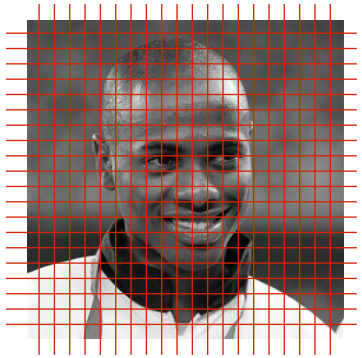
- Solución equivocada: programar a mano (definir que es cada objeto)
- **Solución correcta:** Hacer que el programa aprenda, mostrándole muchos ejemplos de lo que queremos
  - En realidad, se escribe la estructura del programa y durante el aprendizaje se ajustan los parámetros internos (θ)

Estrategia de aprendizaje de A.I.

Ejemplo basado en lección de Sam Roweis (http://videolectures.net/mlss06tw_roweis_mlpgm/)

¿Qué es Aprendizaje de Máquinas (ML)?

- ¿Cómo representar la información? Necesitamos seleccionar un método que identifique las regularidades o estructura de la data



$x = [127,254,3,18, \dots]$

Representación de una entrada (cara)

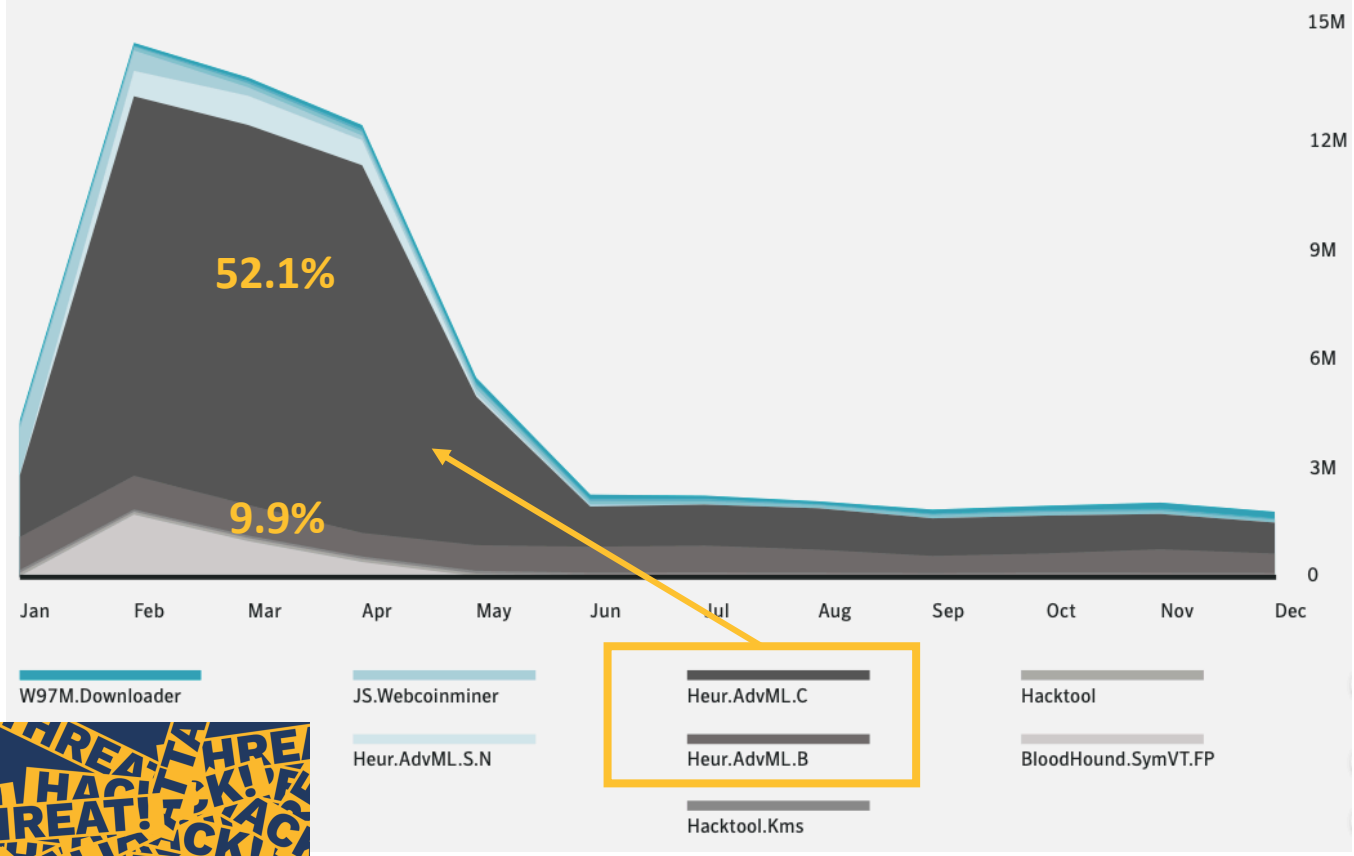
- Nuestro programa necesita entonces:
  1. Función matemática para calcular salida: $\boldsymbol{Z} = f(\boldsymbol{X}, \boldsymbol{\theta})$
  2. Función de Error: $L(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z})$
- Reto es decidir como representar $\boldsymbol{X}, \boldsymbol{Y}$, y $f()$

$\boldsymbol{\theta}$ = parámetros a ajustar

Cuan bien estamos aprendiendo

Top 10 Detecciones de Malware 2019

- Según reporte de seguridad de Symantec
- La lista dominada por mecanismos de ML, las cuales detectan nuevas formas de malware genérico
  - 62% de detecciones
- En 2018, Symantec detectó Wannacry semanas antes, gracias al módulo de ML



Funciones Canónicas de ML

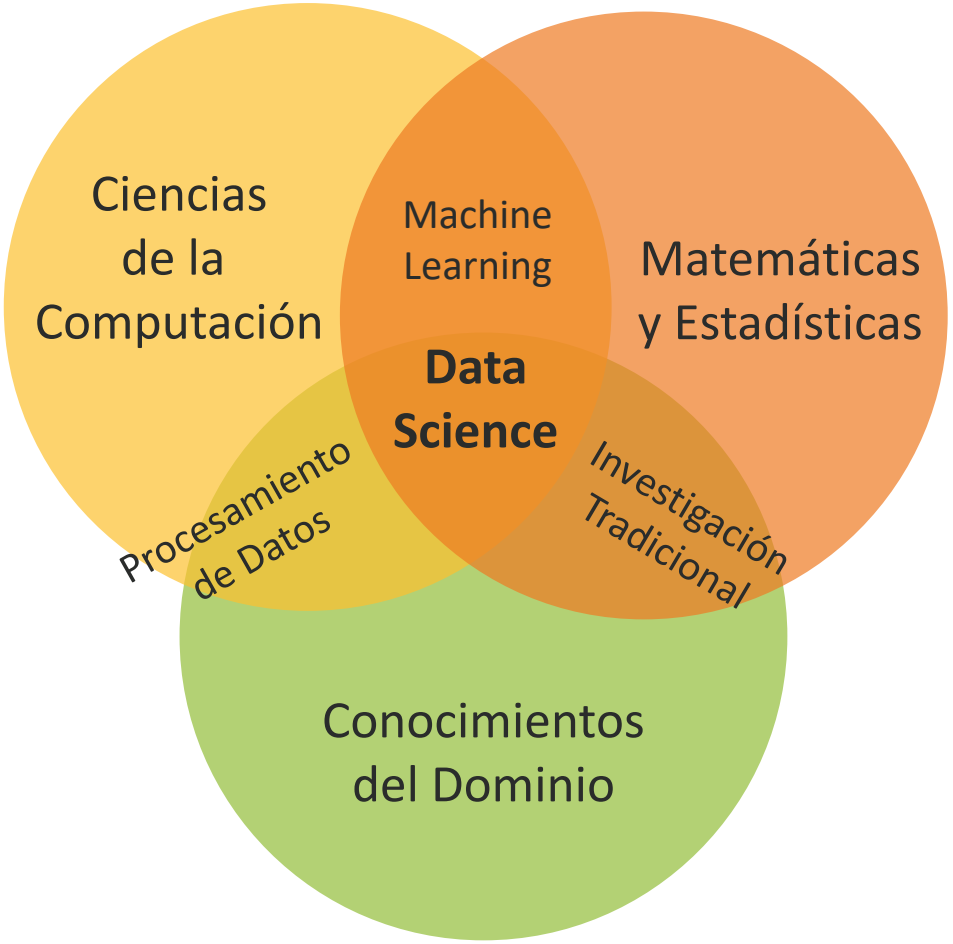
- $\boldsymbol{X}$= Entradas, $\boldsymbol{y}$= respuestas correctas, $\boldsymbol{Z}$= salida del programa

Estrategia de Aprendizaje	Descripción
Supervisado (Supervised Learning)	Dado ejemplos de entradas ($\boldsymbol{X}$) y su correspondiente salidas ($\boldsymbol{y}$), predecir salidas de futuras entradas Ej.: clasificación, regresión
No supervisado (Unsupervised Learning)	Dado solamente entradas ($\boldsymbol{X}$), descubrir estructura, características, etc. Ej.: clustering, outlier detection, compresión
Descubrimiento de Reglas (Rule Learning)	Descubrir subconjuntos comunes de mediciones
Con Reforzamiento (Reinforcement Learning)	Maximizar recompensa en base a acciones

Principales estrategias de ML

¿Qué es Ciencia de Datos?

- Es un campo interdisciplinario para:
  - recolectar, organizar, analizar, interpretar y representar datos,
  - usando ciencias de la computación,
  - con un conocimiento del área de interés
- Objetivo es obtener conocimiento de (grandes) conjuntos de datos, eficiente e inteligentemente



Fuente: http://drewconway.com/zia/2013/3/26/the-data-science-venn-diagram

Python

- Lenguaje de programación orientado a objetos
- Funciona para “scripting” y operaciones de misión critica
- Interpretado pero compilado bytecode
- Sintaxis simple pero robusta
- Lenguaje “fácil de leer”
- “Baterías incluidas”: sinnúmero de librerías disponibles
- Usar versión 3.5 o superior

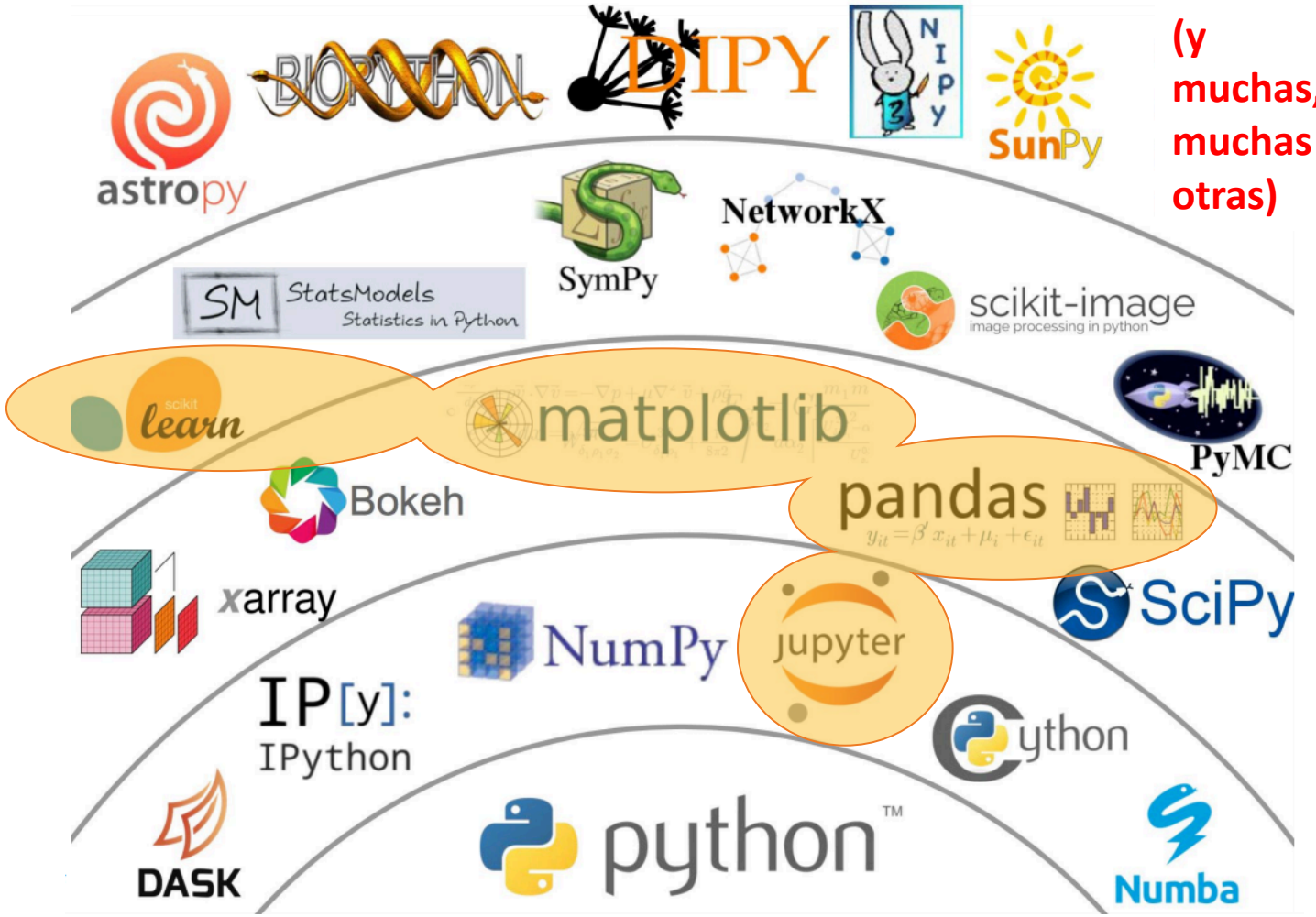
```
print("Hello World!")
```

Python

- Usar versión 3.5 o superior
- Versiones 2.x expiraron (sin soporte)

Python version	Maintenance status	First released	End of support	Release schedule
3.8	bugfix	2019-10-14	2024-10	PEP 569
3.7	security	2018-06-27	2023-06-27	PEP 537
3.6	security	2016-12-23	2021-12-23	PEP 494
3.5	security	2015-09-13	2020-09-13	PEP 478
2.7	end-of-life	2010-07-03	2020-01-01	PEP 373

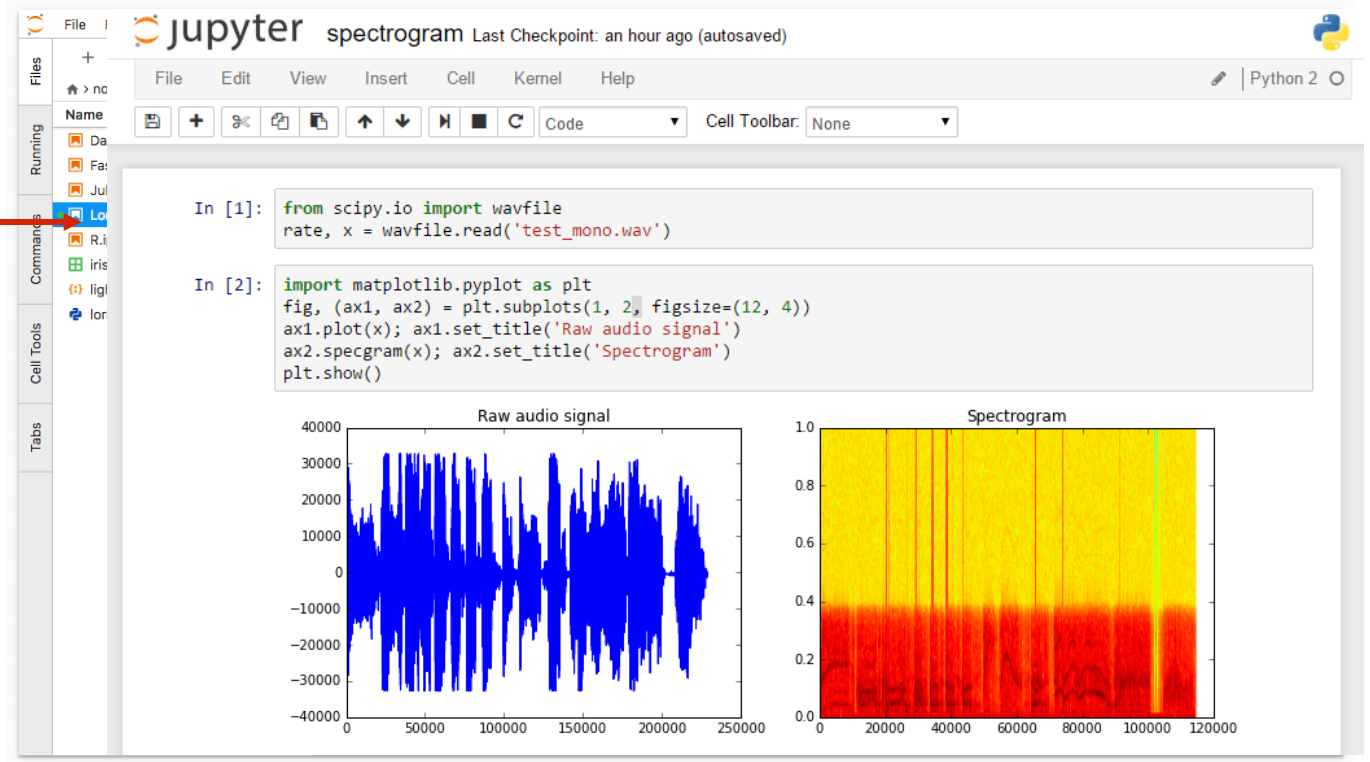
Ecosistema Científico de Python



(y muchas, muchas otras)

Jupyter Notebooks

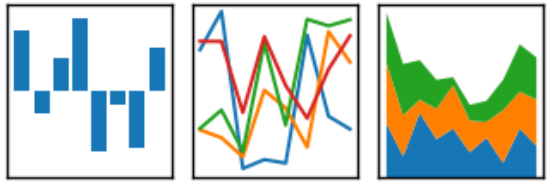
- Herramienta para computación interactiva
  - Conversación con una computadora sobre la data
- Ecosistema Jupyter
  - Ipython
  - Jupyter Notebook
  - Widgets
  - JupyterHub
  - JupyterLab



Pandas

pandas

$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$



- Provee estructuras de datos (matriz) y herramientas de manipulación
- Basado en Numpy, otra librería muy útil
- Capacidades similares a hojas de cálculo y bases de datos relacionales
- Robusta funcionalidad de indexado que permite fácil manejo de datos

	_id	_index	_score	_source.category_id	_source.type_id	_source.user_name	_source.version	_type
0	51d7f0a0-53c1-11e8-d3fa-00000018db99	atp_cmd_results-fdrfulldump-5a99025495f04e6a95a2c34eb139306b-2018-05-23	1.0	5	8002	SYSTEM	1.0.0	event
1	51f93440-53c1-11e8-c6d6-00000018db9a	atp_cmd_results-fdrfulldump-5a99025495f04e6a95a2c34eb139306b-2018-05-23	1.0	5	8007	SYSTEM	1.0.0	event

Scikit-learn

- Librería de machine learning, incluye versiones eficientes de un gran número de algoritmos comunes
- Posee un API uniforme y claro y muy buena documentación

```
model.fit(X_train, y_train)
model.predict(X_test)
```

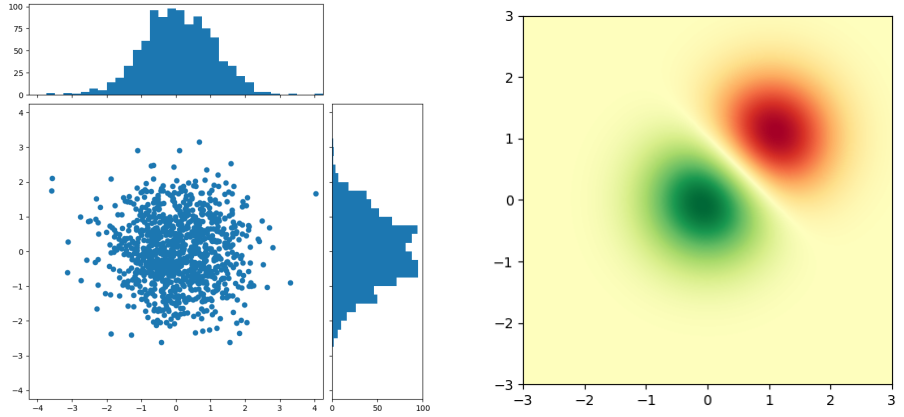
- Produce un nuevo vector de predicciones (y)
- Aplica para algoritmos de clasificación, regresión y clustering

```
model.transform(X_test)
```

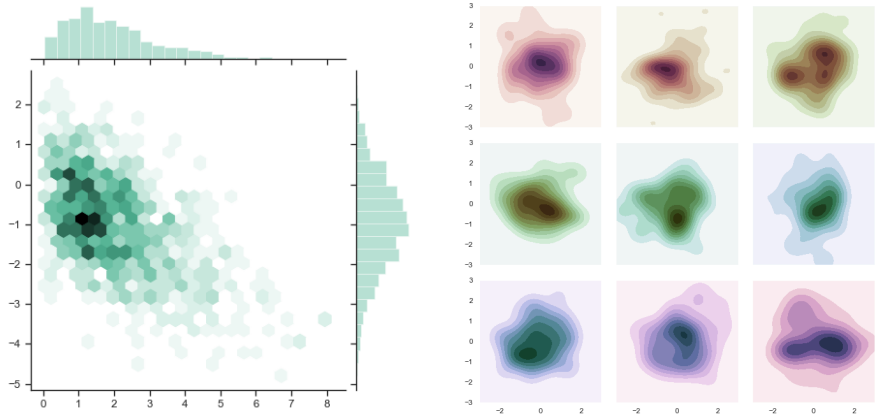
- Produce una nueva representación de X
- Aplica para algoritmos de pre-procesamiento, reducción de dimensiones, selección de características

Matplotlib

- Librería para la visualización 2D de datos (también 3D)
- Multi-plataforma, trabaja en scripts, Ipython, Jupyter, servidores web
- Posee módulo `pyplot` para interface tipo MATLAB
- Interface y estilo luce un poco desactualizado → Seaborn y Altair al rescate (otras librerías de visualización)



Matplotlib



Seaborn

Filosofía Open-Source

- Toda la documentación de los programas a utilizar está disponible en el Internet. Úsala!

Libreria	Versión Mas Reciente	Documentación
Python	3.8.5	https://docs.python.org/3/library/index.html
Numpy	1.19	https://numpy.org/doc/
Pandas	1.1.0	https://pandas.pydata.org/docs/
Scikit-learn	0.23	https://scikit-learn.org/stable/
Matplotlib	3.3.1	https://matplotlib.org/contents.html
Jupyter Notebook	6.1.3	https://jupyter-notebook.readthedocs.io/en/stable/

Conclusiones

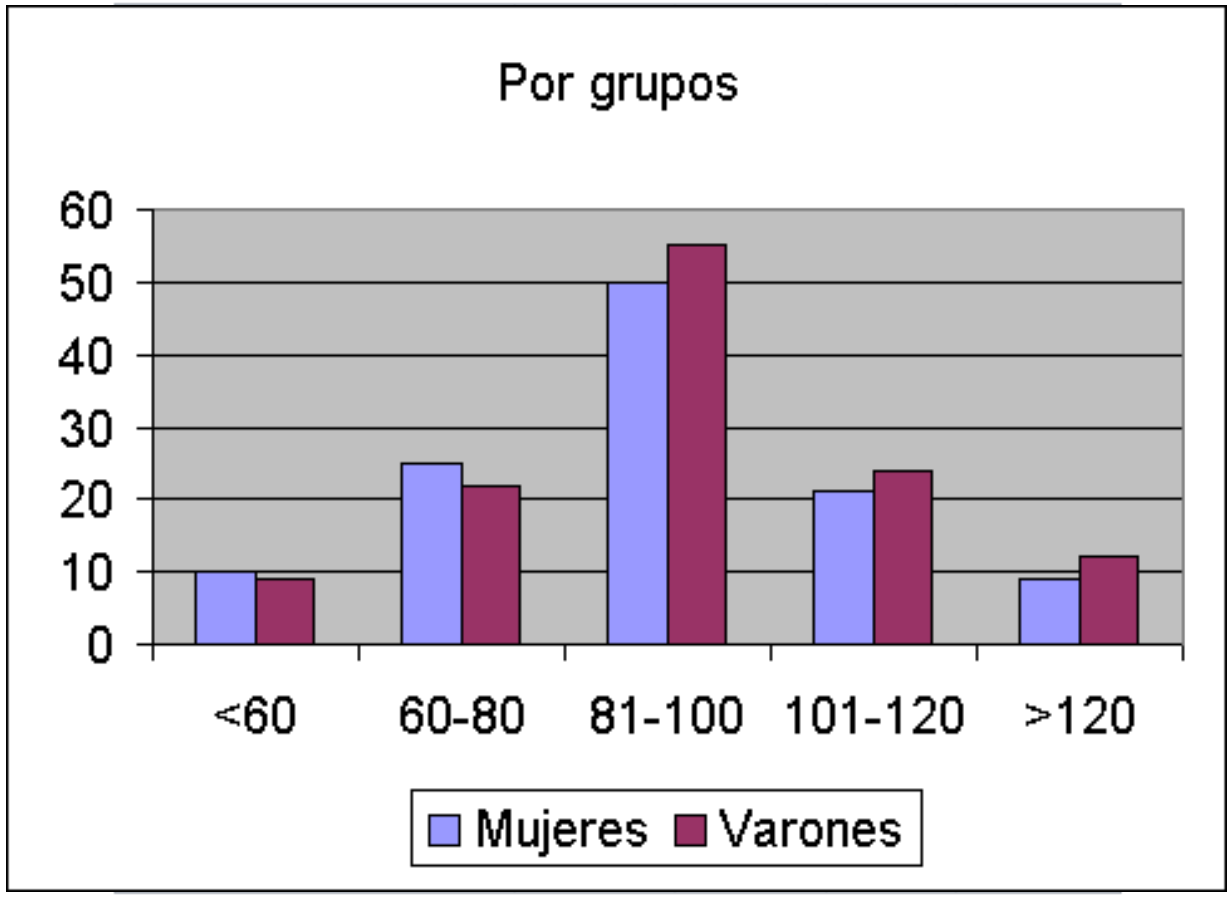
- Data Science es una caja de herramientas
  - No es posible analizar manualmente los datos a nuestro alcance
- Data Science incluye Machine Learning (ML)
  - ML es el conjunto de algoritmos que permiten que los programas aprendan a partir de los datos
- Python, buen lenguaje para aprender programación y data science

Preguntas?

Análisis Exploratorio de Datos con Librería Pandas

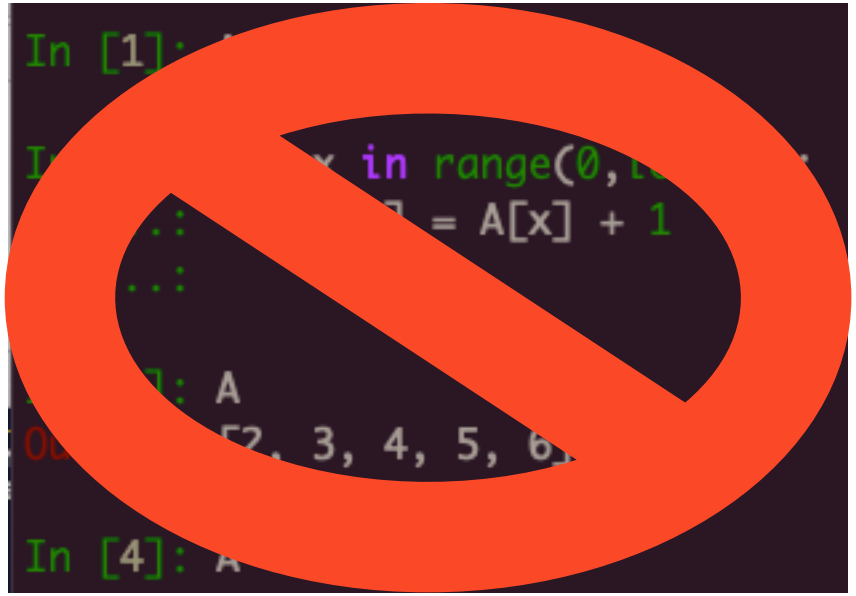
Exploración de Datos en Pandas

- **Análisis Exploratorio de Datos (EDA):** Concepto de estadística, analizando conjuntos de datos para detector principales características, usualmente con métodos visuales
  - No es necesario un modelo estadístico
  - Qué nos puede decir la data?



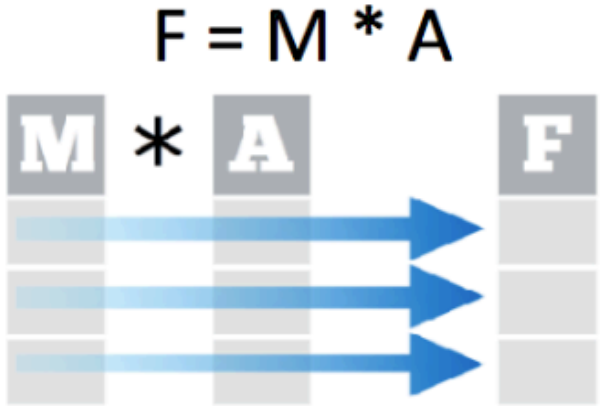
Exploración de Datos en Pandas

- Pandas ofrece ***estructuras de datos vectorizadas***, permitiendo realizar operaciones ***sobre todos los datos a la vez***
- Olvídate de los loops!



Exploración de Datos en Pandas

- Pandas ofrece ***estructuras de datos vectorizadas***, permitiendo realizar operaciones ***sobre todos los datos a la vez***
- Uso de matrices!



Cada **variable** es almacenada en su propia **columna**

&

Cada **observación** o **muestra** es almacenada en su propia **fila**

Pandas: Estructuras de Datos

Dimensiones	Nombre	Descripción
1	Series	Vector indexado
1	Timeseries	Series que usa tiempo como índice
2	DataFrame	Matriz indexada
3	Panel	Estructura de datos 3D (mutable)

- Las estructuras más populares son Series y Dataframes

Pandas: Series

- Una `Series` es como una lista o diccionario de Python pero **MEJOR!**
- Es el principal elemento sobre el cual existen las otras estructuras de datos

```
> misDatos = pd.Series( <data>, index=<index>)
```

```
In [6]: import pandas as pd

In [7]: series1 = pd.Series(['a','b','c','d'])

In [8]: series1
Out[8]:
0    a
1    b
2    c
3    d
dtype: object
```

Usando una lista de Python

```
In [11]: series2 = pd.Series({'nombre':'Gaspar',
    ...: 'apellido':'Modelo',
    ...: 'pais':'Panama'})

In [12]: series2
Out[12]:
nombre      Gaspar
apellido    Modelo
pais        Panama
dtype: object
```

Usando un diccionario de Python

Pandas: Series

- Olvídate de los loops!

```
In [1]: A = [1,2,3,4,5]

In [2]: for x in range(0,len(A)):
   ...:     A[x] = A[x] + 1
   ...:

In [3]: A
Out[3]: [2, 3, 4, 5, 6]

In [4]: A += 1
```

- Vectorización al rescate!

```
In [13]: series1 = pd.Series([1,2,3,4,5])

In [14]: series1 += 1

In [15]: series1
Out[15]:
0    2
1    3
2    4
3    5
4    6
dtype: int64
```

Pandas: DataFrame

- Un `DataFrame` es un conjunto de `Series`
- Permite que se almacenen datos de distintos tipos en una misma estructura

```
> df = pd.DataFrame( <data>, index=<index>, <column_names> )
```

- Se pueden importar archivos en múltiples formatos
  - Texto (CSV), Clipboard, Excel, JSON, HTML, SQL, Python Pickle, SAS, SPSS, Parquet, ORC, Google BigQuery, STATA

```
In [2]: df = pd.read_csv('portafolio.csv')

In [3]: df
Out[3]:
  nombre  acciones  precio
0     AA       100   32.20
1    IBM        50   91.10
2    CAT       150   83.44
3   MSFT       200   51.23
4     GE        95   40.37
5   MSFT        50   65.10
6    IBM       100   70.44
```

Pandas: DataFrame

- Métodos útiles para calcular estadísticas y explorar los datos

```
In [10]: df
Out[10]:
  nombre  acciones  precio estrategia
0     AA       100   32.20   mantener
1    IBM        50   91.10     vender
2    CAT       150   83.44     vender
3   MSFT       200   51.23   mantener
4     GE        95   40.37     vender
5   MSFT        50   65.10     vender
6    IBM       100   70.44   mantener
```

- info()

```
In [11]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7 entries, 0 to 6
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   nombre      7 non-null      object
 1   acciones    7 non-null      int64
 2   precio      7 non-null      float64
 3   estrategia  7 non-null      object
dtypes: float64(1), int64(1), object(2)
memory usage: 352.0+ bytes
```

- describe()

```
In [12]: df.describe()
Out[12]:
         acciones     precio
count    7.000000   7.000000
mean   106.428571  61.982857
std     53.596819  21.829700
min     50.000000  32.200000
25%     72.500000  45.800000
50%    100.000000  65.100000
75%    125.000000  76.940000
max    200.000000  91.100000
```

Pandas: DataFrame

- Métodos útiles para calcular estadísticas y explorar los datos

```
In [10]: df
Out[10]:
  nombre  acciones  precio estrategia
0     AA       100   32.20   mantener
1    IBM        50   91.10     vender
2    CAT       150   83.44     vender
3   MSFT       200   51.23   mantener
4     GE        95   40.37     vender
5   MSFT        50   65.10     vender
6    IBM       100   70.44   mantener
```

- value_counts()

```
In [13]: df['estrategia'].value_counts()
Out[13]:
vender      4
mantener    3
Name: estrategia, dtype: int64
```

- isnull()

```
In [14]: df.isnull().any()
Out[14]:
nombre        False
acciones      False
precio        False
estrategia    False
dtype: bool
```

Laboratorio 1: Análisis Exploratorio de Datos con Librería Pandas

Break (5 min)

Algoritmos Supervisados de ML

Scikit-learn

- Librería de machine learning, incluye versiones eficientes de un gran número de algoritmos comunes
- Posee un API uniforme y claro y muy buena documentación

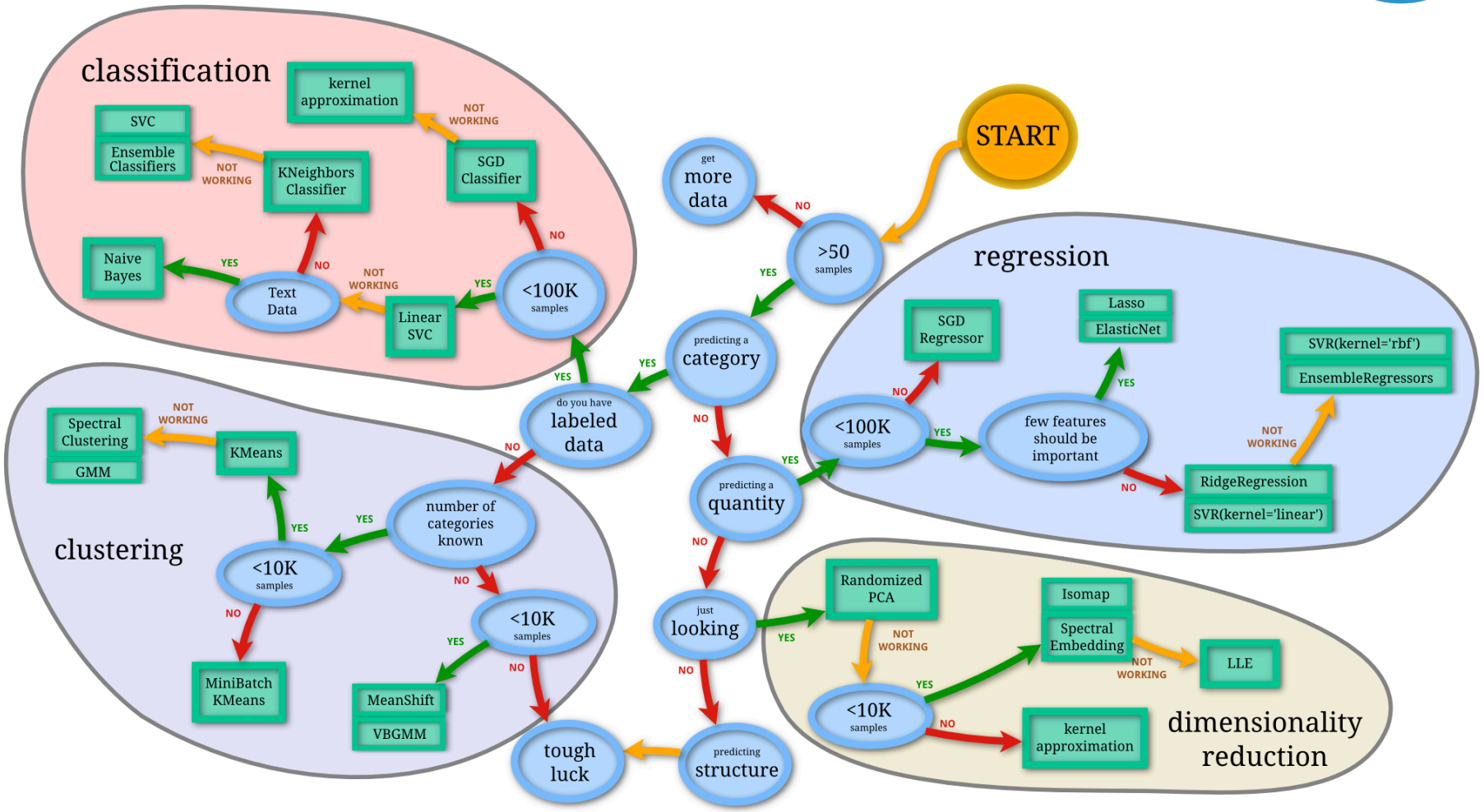
```
model.fit(X_train, y_train)
model.predict(X_test)
```

- Produce un nuevo vector de predicciones (y)
- Aplica para algoritmos de clasificación, regresión y clustering

```
model.transform(X_test)
```

- Produce una nueva representación de X
- Aplica para algoritmos de pre-procesamiento, reducción de dimensiones, selección de características

Scikit-learn



Scikit-learn

- Secciones de la Guía del Usuario (https://scikit-learn.org/stable/user_guide.html)
  - ***Supervised Learning***
  - ***Unsupervised Learning***
  - ***Model selection and evaluation***
  - Inspection
  - ***Visualizations***
  - Dataset transformations
  - ***Dataset loading utilities***
  - Computing with scikit-learn

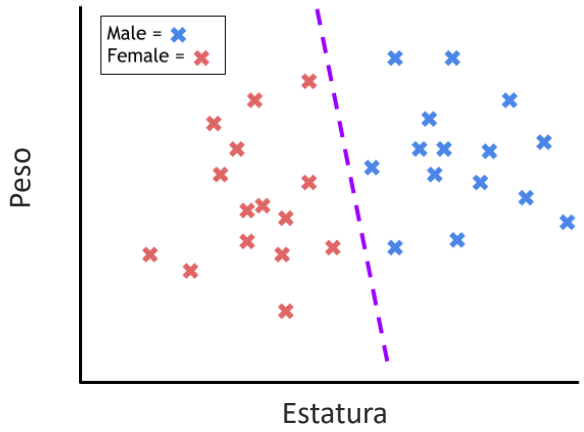
Algoritmos Supervisados

- Conjunto más popular de ML, algoritmos aprenden usando ejemplos
- Al entrenar un modelo supervisado, los datos de entrenamiento consisten de muestras (X) y sus correspondientes respuestas (Y)

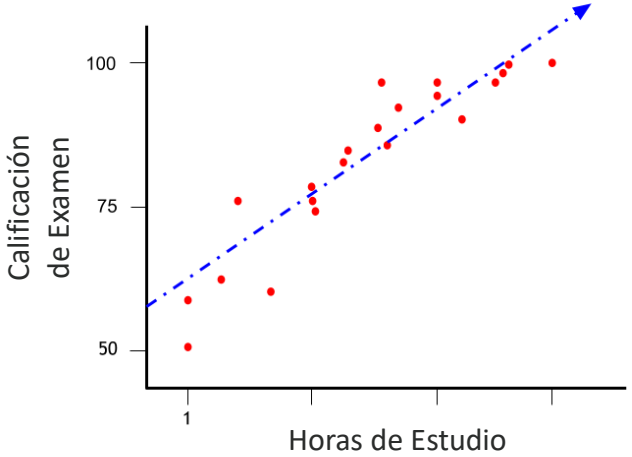
$$\boldsymbol{Z} = f(\boldsymbol{X}, \theta) \qquad L(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z})$$

- Tipos de problemas a resolver con algoritmos supervisados:

Clasificación

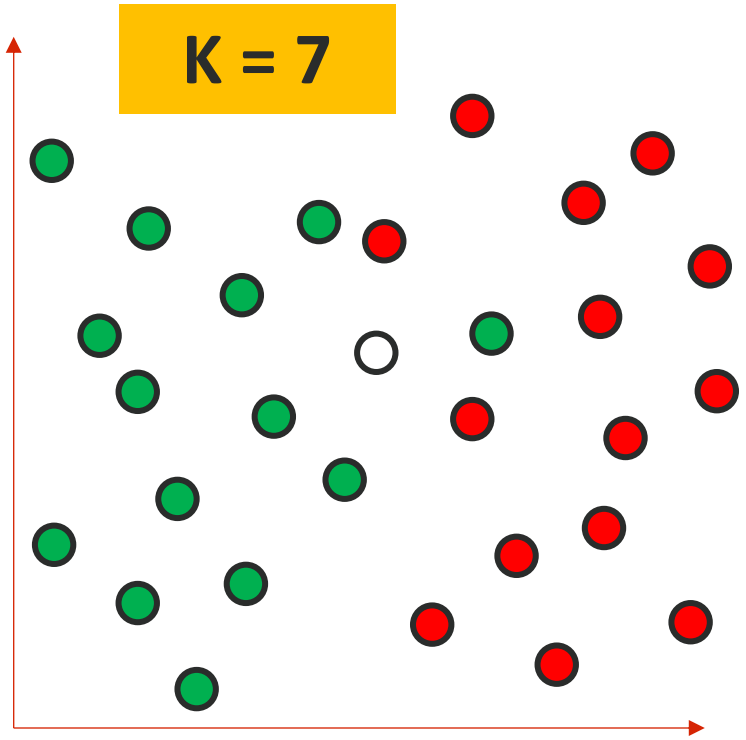


Regresión



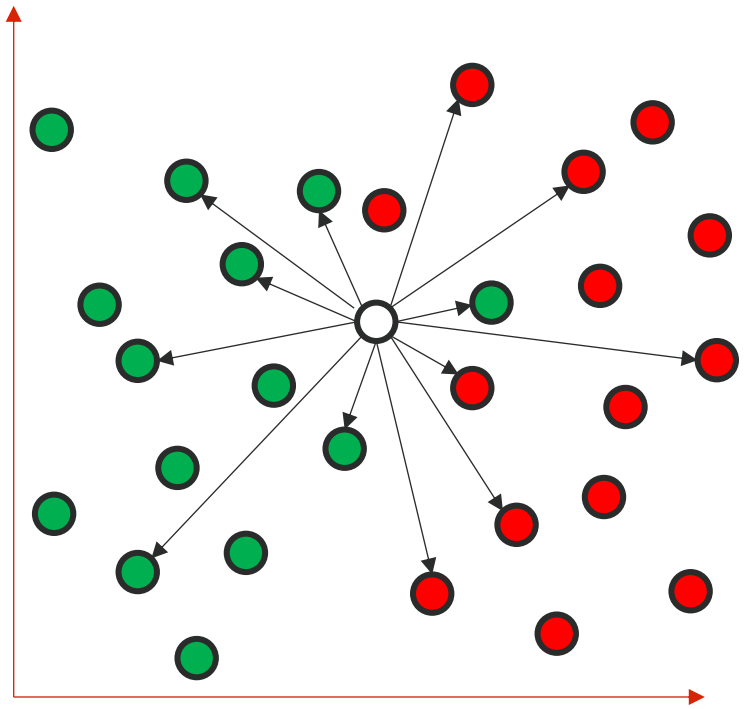
Algoritmo KNN (K-Nearest Neighbors)

- Algoritmo de clasificación, basado en distancia y de tipo no-paramétrico
- Puede ser usado para clasificación o para regresión
- Pasos para KNN (clasificación)
  1. Selecciona un valor para K (número de vecinos a considerar)



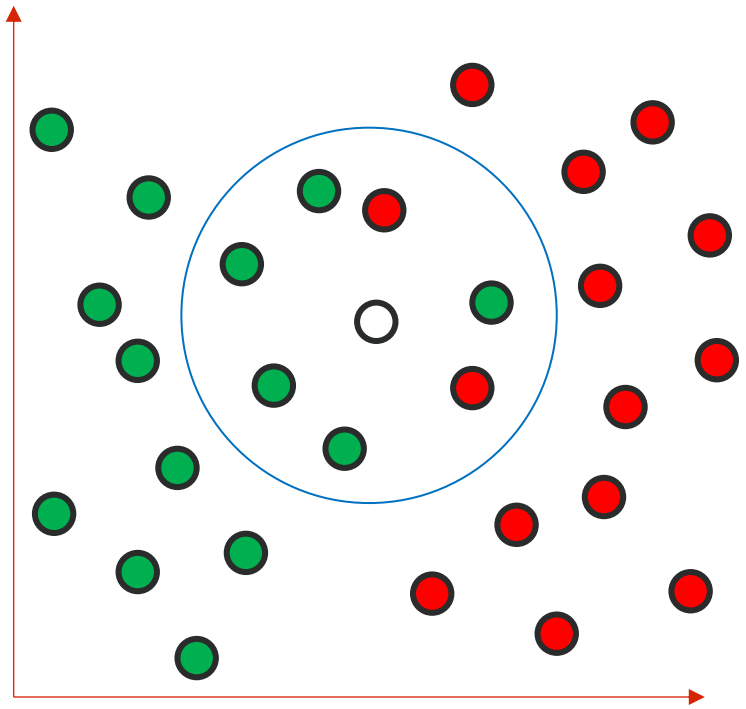
Algoritmo KNN (K-Nearest Neighbors)

- Algoritmo de clasificación, basado en distancia y de tipo no-paramétrico
- Puede ser usado para clasificación o para regresión
- Pasos para KNN (clasificación)
  1. Selecciona un valor para K (número de vecinos a considerar)
  2. Calcula la distancia entre la(s) muestra(s) a clasificar y el resto de los datos (todos!)



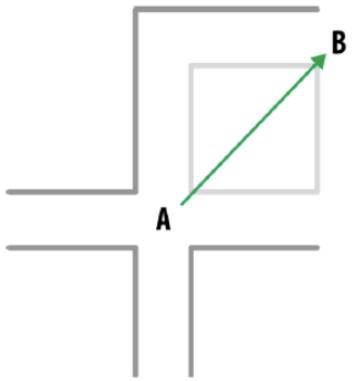
Algoritmo KNN (K-Nearest Neighbors)

- Algoritmo de clasificación, basado en distancia y de tipo no-paramétrico
- Puede ser usado para clasificación o para regresión
- Pasos para KNN (clasificación)
  1. Selecciona un valor para K (número de vecinos a considerar)
  2. Calcula la distancia entre la(s) muestra(s) a clasificar y el resto de los datos
  3. Determinar la clase de la(s) muestra(s) en base a los K vecinos más cercanos (votación)
- Algoritmo conocido como *aprendiz perezoso* (lazy learner)

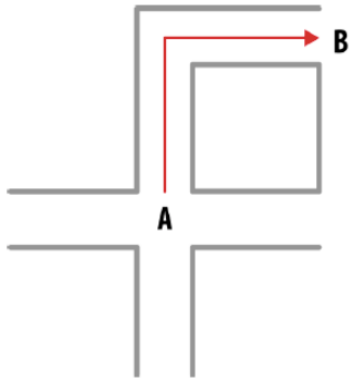


Algoritmo KNN (K-Nearest Neighbors)

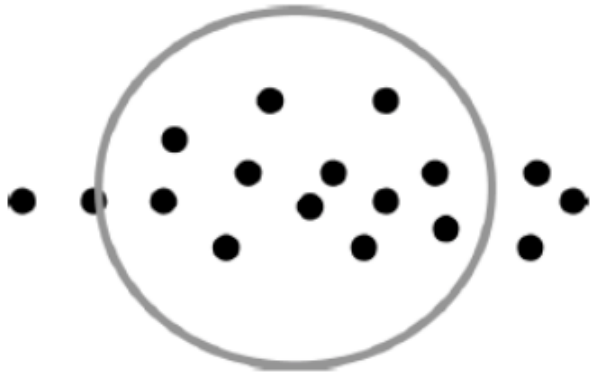
- Cómo determinar el valor de K? (parámetro de ajuste)
  - No hay fórmula exacta (sugerencia: considerar $\left\| K = \sqrt{no_muestras} \right\|$)
  - Necesario probar múltiples valores
- Cuál formula de distancia a utilizar?
  - Depende de los tipos de datos a procesar



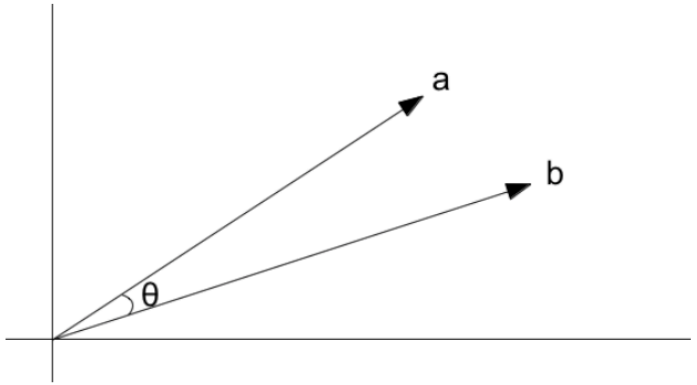
Distancia Euclidiana



Distancia Manhattan



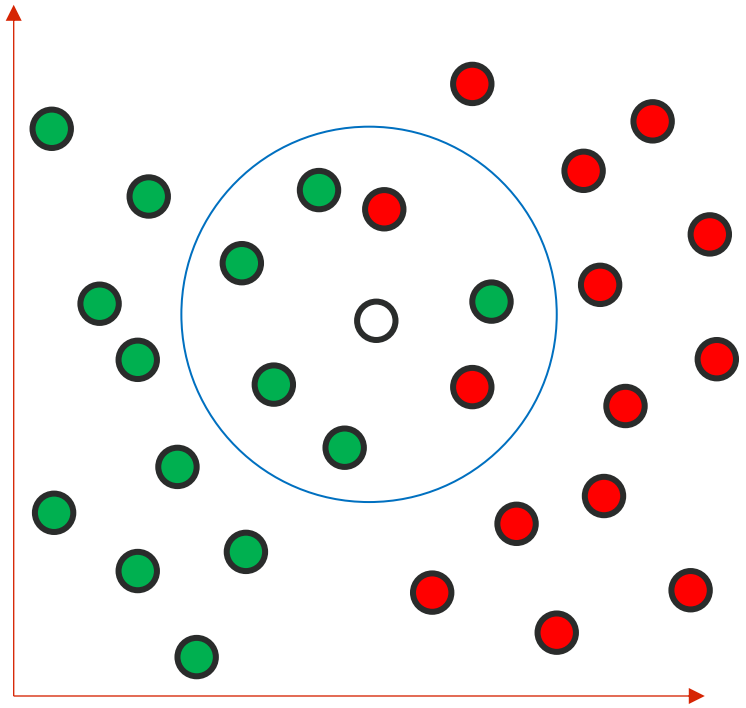
Distancia Mahalaobnis (variación estadística)



Similaridad por Coseno

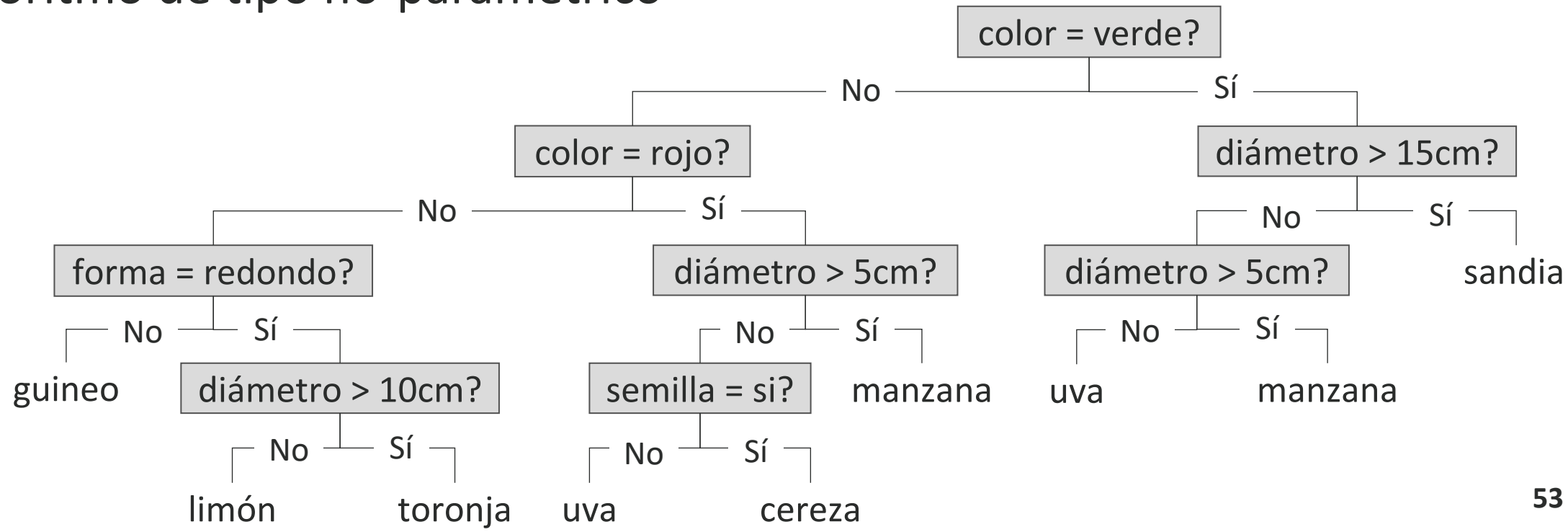
Algoritmo KNN (K-Nearest Neighbors)

- Algoritmo simple pero poderoso si es bien aplicado
  - Utiliza concepto intuitivo (distancia) que puede facilitar interpretación de resultados
- Es computacionalmente intensivo, aumento de memoria a medida que crece el conjunto de datos a evaluar y el número de dimensiones
  - Maldición de la dimensionalidad



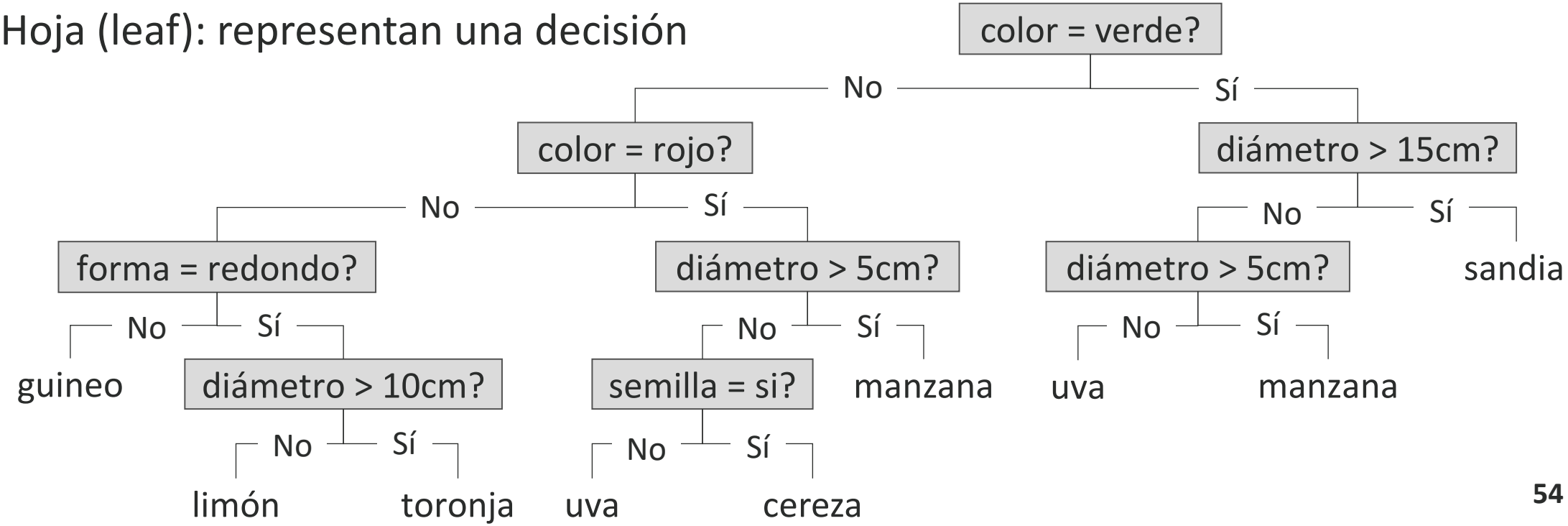
Algoritmo Árbol de Decisión

- Uno de los métodos más prácticos de ML, bastante intuitivo
  - Modelo entrenado luce como serie de condiciones if-else, arregladas en un árbol
- Algoritmo de tipo no-paramétrico



Algoritmo Árbol de Decisión

- Está compuesto por tres tipos de nodos:
  - Decisión: tienen dos o más ramas
  - Raíz (root): nodo en nivel superior
  - Hoja (leaf): representan una decisión

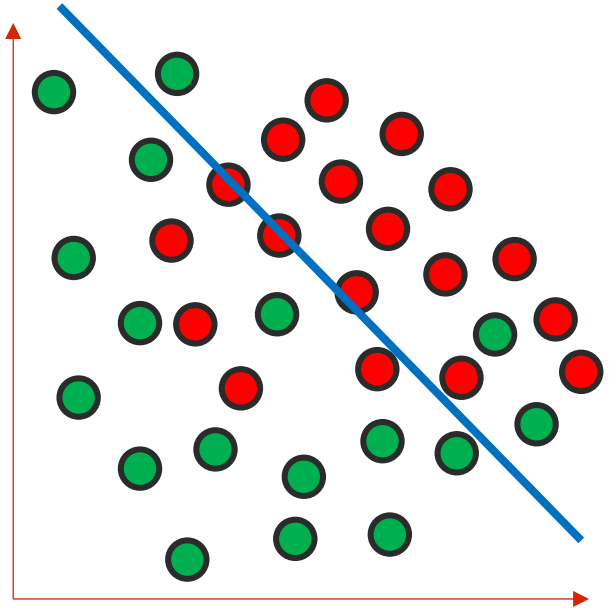


Algoritmo Árbol de Decisión

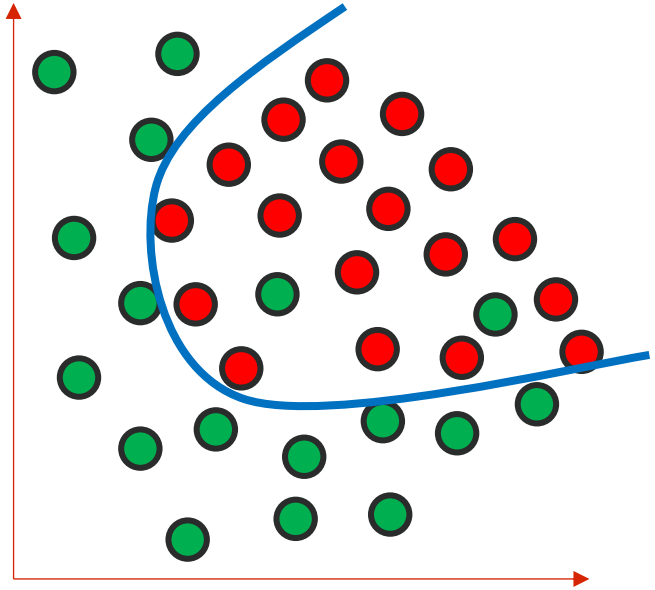
- Pasos para crear árbol de decisión (clasificación):
  1. Determinar el nodo raíz del árbol
     - Necesario calcular la *ganancia de información* para cada regla candidata, usando *impureza Gini* o *Entropía*
  2. Dividir los nodos de decisión (hijos) recursivamente
     - Considera las reglas candidatas no utilizadas
     - Algoritmo se detiene si no quedan reglas o la impureza es cero
  3. Nodos que no pueden ser divididos son los nodos hojas
- Implementación en *scikit-learn* es el algoritmo CART (Classification and Regression Trees)

Problema de Sobreajuste (Overfit) o Desajuste (Underfit) de un Modelo

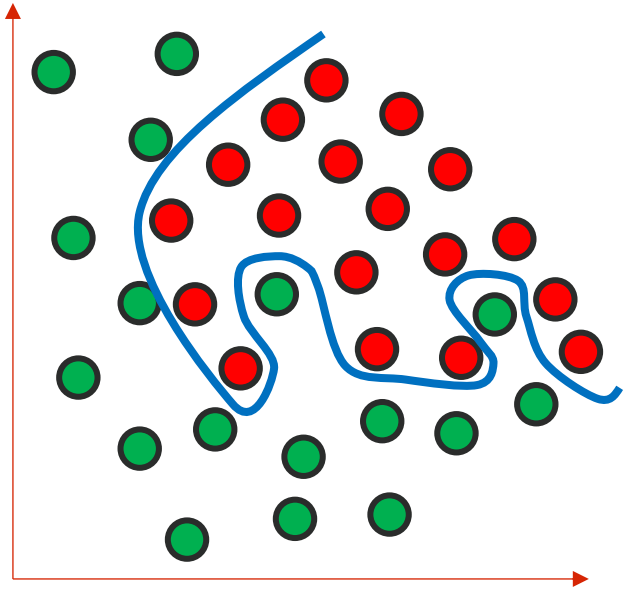
- Sobreajuste y desajuste son fenómenos que indican que el modelo no aprendió correctamente de los datos
  - Generalizamos demasiado (underfit) o aprendemos demasiado (overfit)



Desajuste (underfit)



Ajuste Apropiado



Sobreajuste (overfit)

Validación Cruzada de Modelos

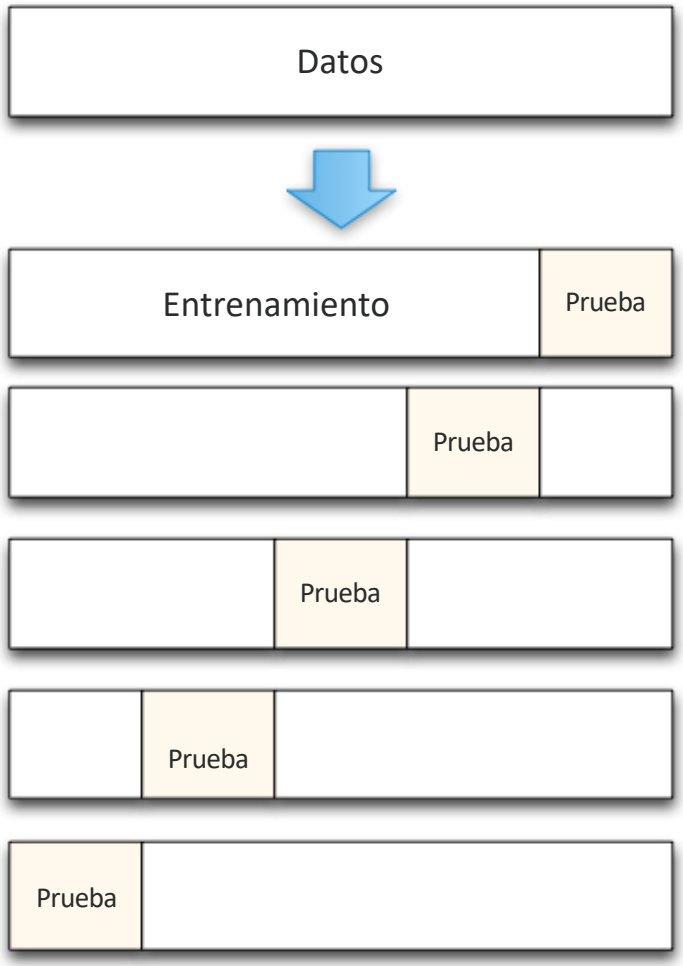
- Conjunto de técnicas que dividen los datos en dos conjuntos: entrenamiento (train) y evaluación (test)
  - Aprender los parámetros de un modelo y evaluarlo sobre los mismo datos es un error! (produce sobreajuste)
- División sencilla (75/25 por defecto)

```
>>> from sklearn.model_selection import train_test_split
>>> X_train, X_test, y_train, y_test = train_test_split(
...     X, y, test_size=0.33, random_state=42)
```

- Uso de validación cruzada (para estimación mas robusta)

```
>>> from sklearn.model_selection import cross_val_score
>>> clf = svm.SVC(kernel='linear', C=1)
>>> scores = cross_val_score(clf, X, y, cv=5)
>>> scores
array([0.96..., 1.  ..., 0.96..., 0.96..., 1.        ])
```

usa estrategia estratificada



Preguntas?

Laboratorio 2: Algoritmos Supervisados de ML

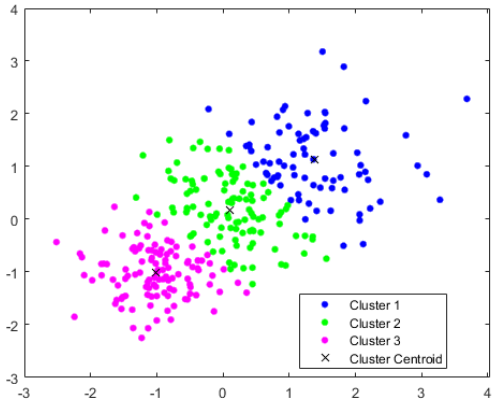
Break (5 min)

Algoritmos No Supervisados de ML

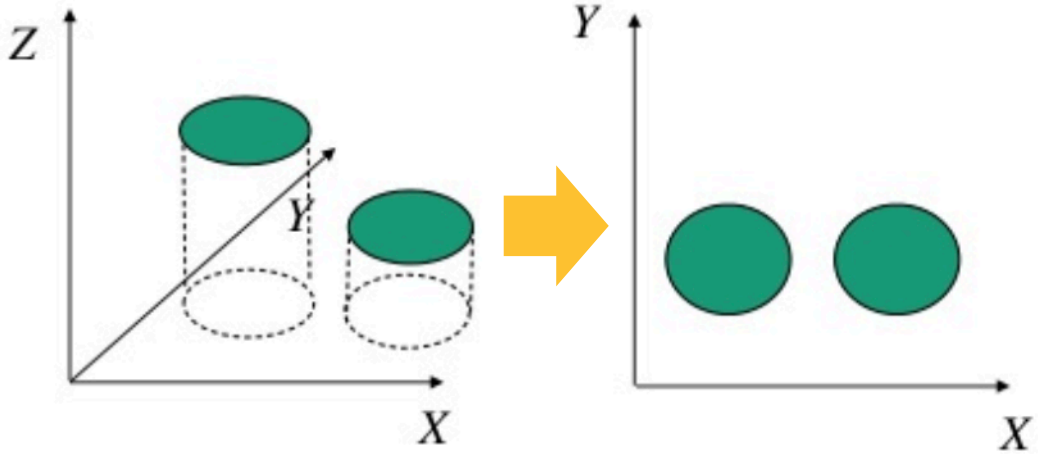
Algoritmos No Supervisados

- ¿Qué hacer cuando no tienes etiquetas para tus datos? ¿Qué hacer cuando tienes exceso de dimensiones?
- Tipos de problemas a resolver con algoritmos no supervisados:
- Clustering:
  - Útil para aprender sobre dominio del problema
  - No existe un sólo “mejor” método de clustering

Agrupar datos (clustering)



Reducir Dimensiones (manifold learning)



K Means (Clustering)

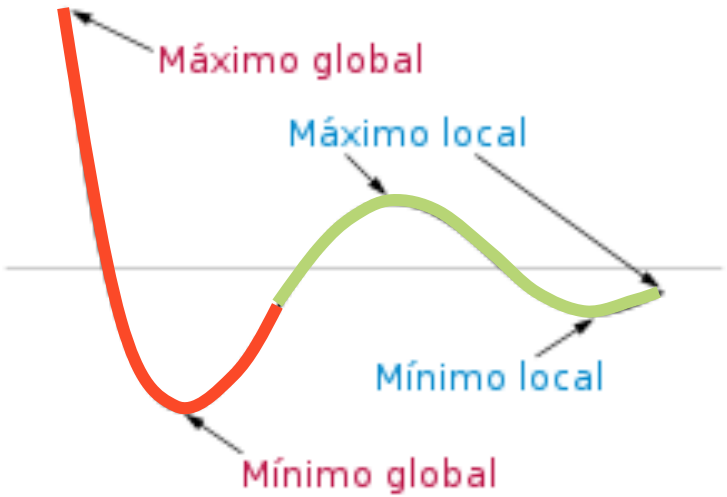
- Objetivo es encontrar grupos en los datos, con el número de grupos definido por el parámetro K y basados en la *similitud* de las muestras provistas
- Pasos de Kmeans:
  - Seleccionar los centroides iniciales
  - Asignar cada muestra al centroide más cercano
  - Crear nuevos centroides tomando el valor promedio de todas las muestras asignadas al centroide anterior
  - Repetir hasta convergencia
- La salida del algoritmo es:
  - Centros de los clusters K
  - Etiquetas para cada muestra, correspondiente a los clusters que pertenecen

K Means (Clustering)

- Objetivo es encontrar grupos en los datos, con el número de grupos definido por el parámetro K y basados en la *similitud* de las muestras provistas
- Pasos de Kmeans:
  - Seleccionar los centroides iniciales
  - Asignar cada muestra al centroide más cercano
  - Crear nuevos centroides tomando el valor promedio de todas las muestras asignadas al centroide anterior
  - Repetir hasta convergencia
- La salida del algoritmo es:
  - Centros de los clusters K
  - Etiquetas para cada muestra, correspondiente a los clusters que pertenecen

K Means (Clustering)

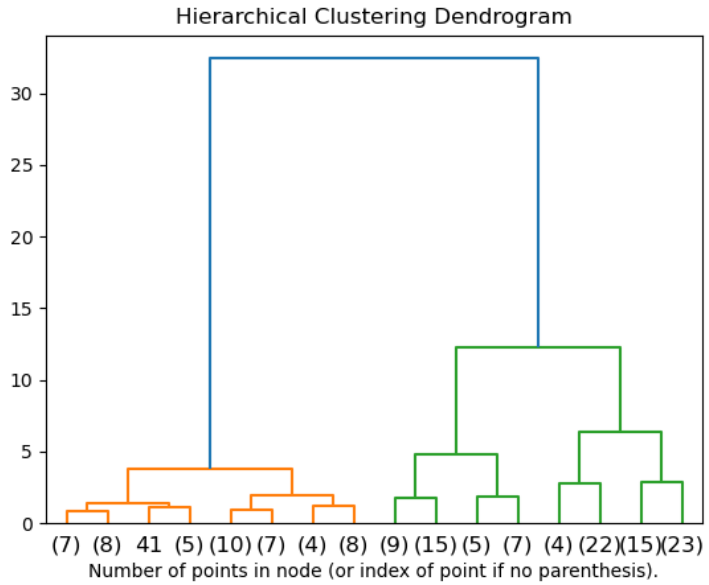
- Es uno de los más rápidos algoritmos para clustering pero tiene problemas de mínimo local
  - Útil ejecutarlo varias veces
- Separa los datos en K grupos de igual varianza, minimizando un criterio conocido como *inercia*
  - *Inercia* es una medida de que tan coherente son los grupos (clusters)
  - *Inercia* asume que los clusters son convexos
  - *Inercia* no es un métrico normalizado, puede crear problemas en espacios de altas dimensiones (numero de características)



- *Mini-Batch K-Means:* versión modificada que actualiza los centroides en base a subgrupos de las muestras, no todo el conjunto de datos

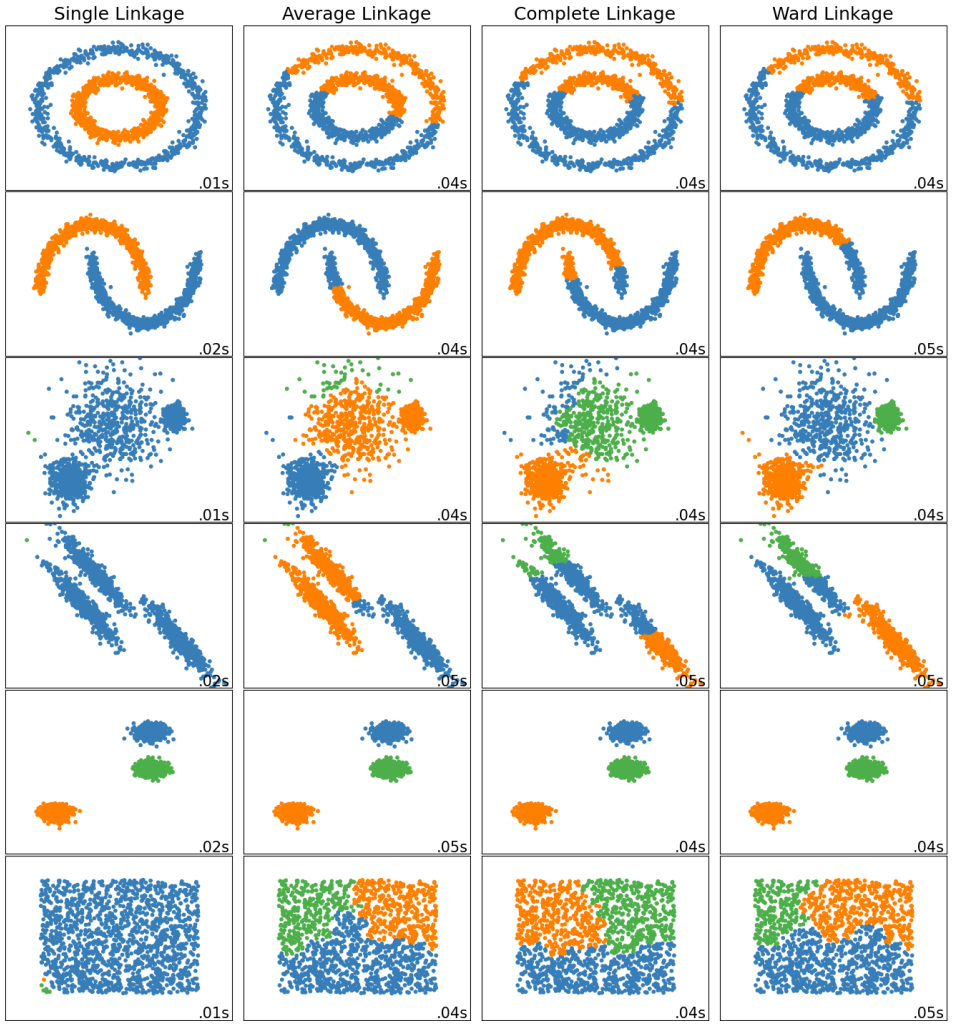
Hierarchical Agglomerative Clustering

- Utiliza una estrategia ”de abajo hacia arriba” (bottom up)
- Cada observación comienza en su propio cluster, los cuales son agrupados de acuerdo a un ***criterio de enlace (linkage criteria)***
  - **Ward:** minimiza suma de diferencias cuadradas en todos los clusters (similar a k-means)
  - **Enlace máximo (complete):** minimiza distancia máxima entre muestras de pares de clusters
  - **Enlace promedio (average):** minimiza distancia promedio entre todas las muestras de pares de clusters
  - **Enlace único (single):** minimiza distancia entre las muestras más cercanas de pares de clusters



Hierarchical Agglomerative Clustering

- Computacionalmente costoso cuando no se definen restricciones de conectividad
- Tiene una tendencia de ‘ricos se hacen más ricos’ que produce clusters de tamaño desigual
  - Single linkage es la peor estrategia, Ward da tamaños más regulares
  - Single linkage puede ser calculado eficientemente por lo que es útil en conjuntos de datos grandes



Otros Algoritmos de Clustering

- Affinity propagation
- BIRCH
- DBSCAN
- Mini-Batch K-Means
- Mean Shift
- OPTICS
- Spectral Clustering
- Gaussian Mixture Model

Preguntas?

Evaluación de Modelos de ML

Validando Modelos Supervisados

- Matriz de confusión
- Precisión
- Recall (recuerdo, memoria)
- Exactitud (accuracy)
- Curva ROC

- Excelente referencia: https://en.wikipedia.org/wiki/Precision_and_recall
- Advertencia: términos tienen distintos significados, dependiendo de área (ciencias, estadísticas, ingenierías)

Qué son Ciertos Positivos y Falsos Positivos?

- Caso típico: clasificación
- Ejemplo: usted quiere detectar perros en fotos
  - Divide las fotos en dos grupos, las que tiene perros y las que no tiene perros
  - Usted tiene dos posibles predicciones

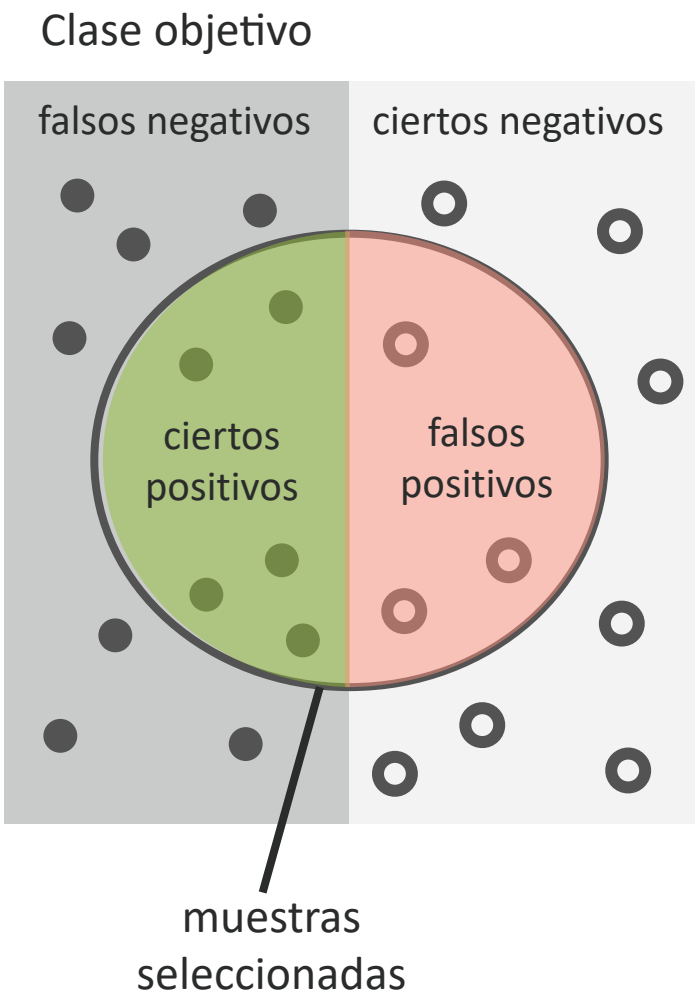
		VERDAD	
		Clase Positiva	Clase Negativa
PREDICCION	Es perro	**Cierto Positivo (True Positive)**	**Falso Positivo (False Positive) Error Tipo I**
	No es perro		

Matriz de Confusión

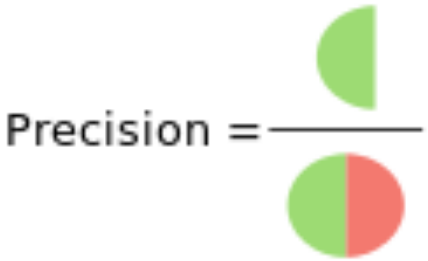
- Todo problema de clasificación se puede resumir a dos clases (positiva y negativa)
- Dos tipos de errores:
  - Tipo I: Falso Positivo
  - Tipo II: Falso Negativo

		VERDAD	
		Clase Positiva	Clase Negativa
PREDICCION	Es perro	Cierto Positivo (True Positive)	Falso Positivo (False Positive) Error Tipo I
	No es perro	Falso Negativo (Falso Negativo) Error Tipo II	Cierto Negativo (True Negative)

Precisión y Recall



Cuantas muestras seleccionadas pertenecen a la clase objetivo?



$$\text{Precisión} = \frac{CP}{CP + FP}$$

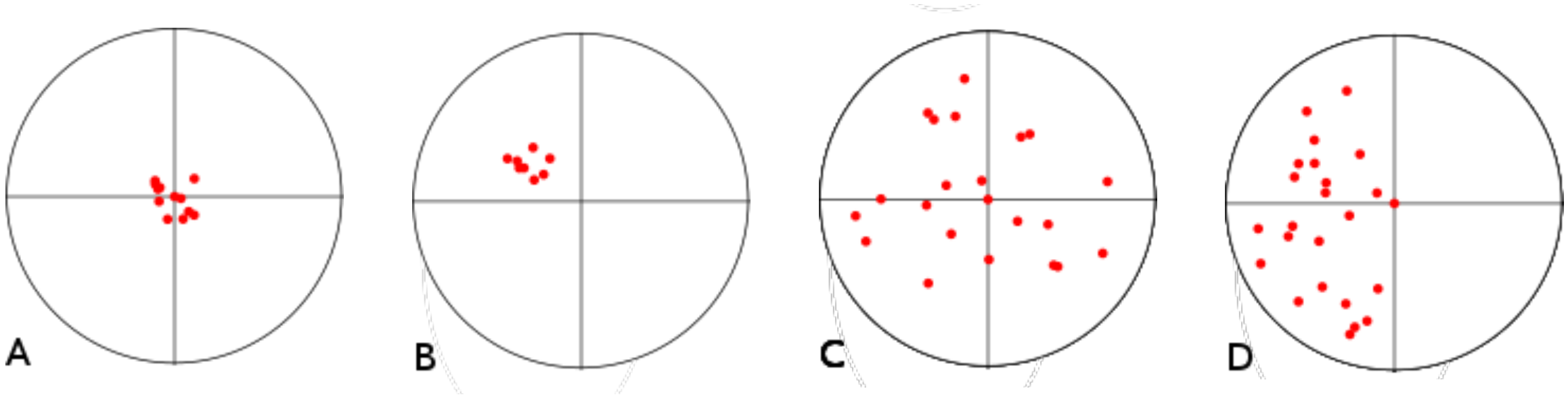
Cuantas muestras de la clase objetivo fueron seleccionadas?

$$\text{Recall} = \frac{CP}{CP + FN}$$

Otros nombres:

- Taza de aciertos (hit rate)
- Razón de ciertos positivos (TPR)

Alta o baja Precisión y Recall?



A	B	C	D
Alta precisión	Alta precisión	Baja precisión	Baja precisión
Alto recall	Bajo recall	Alto recall	Bajo recall

Accuracy (Exactitud)

$$\text{Exactitud} = \frac{CP + CN}{CP + FP + CN + FN}$$

- Buen métrico para comenzar pero puede ocultar problemas en el modelo, especialmente cuando hay problemas por ***desbalance de clases***
- Cual es el valor de exactitud en el siguiente caso:

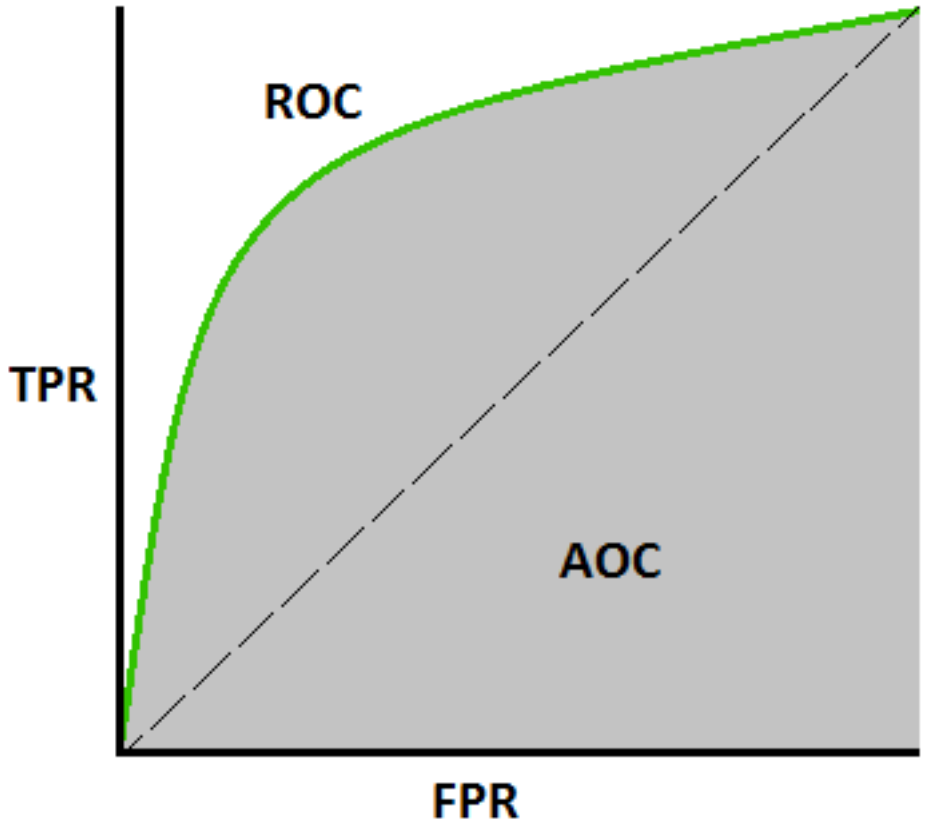
CLASE 1 (OBJETIVO)	10 muestras
CLASE 2	1000 muestras

CP	1
FN	9
CN	950
FP	50

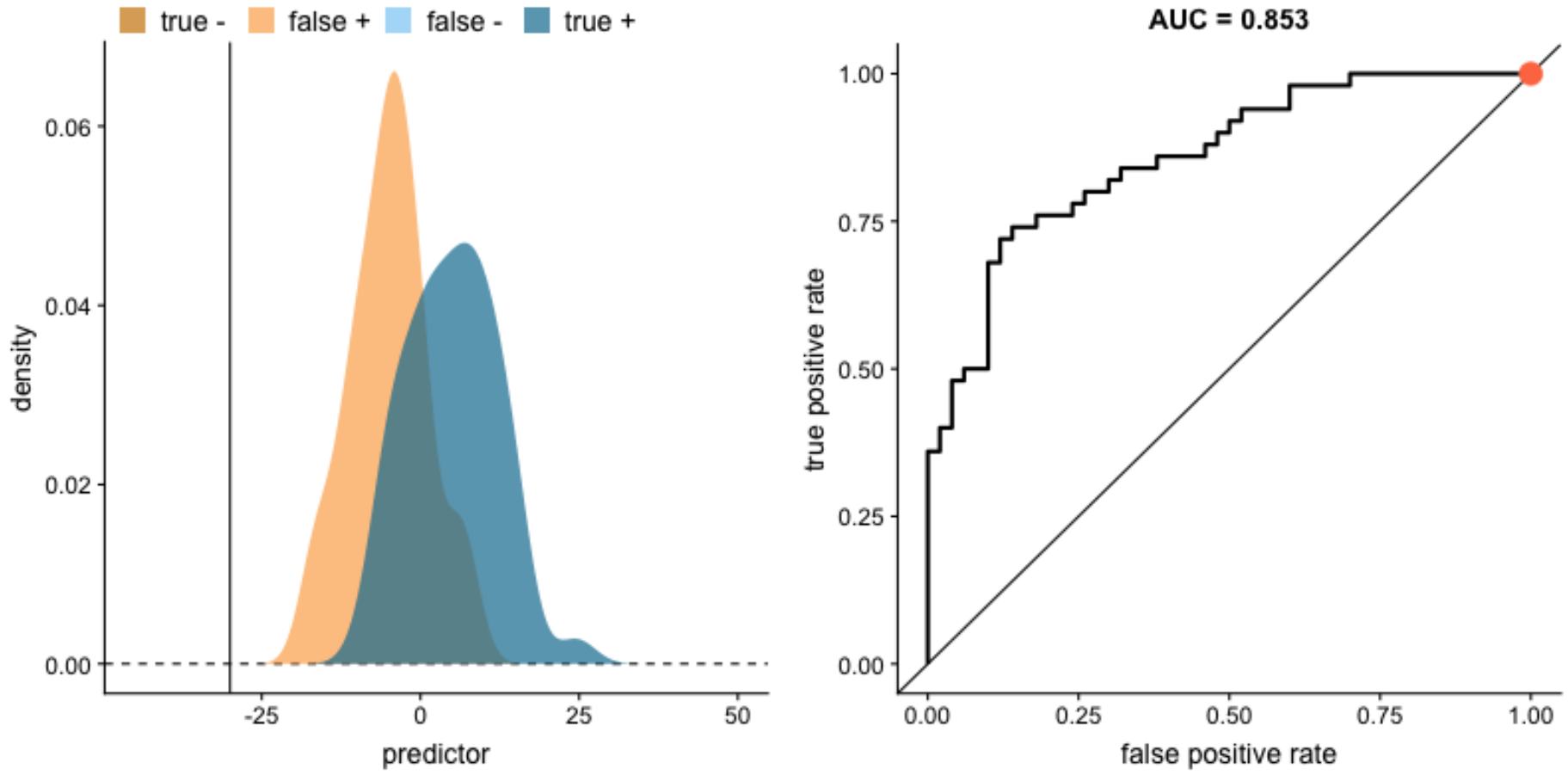
$$\text{Exactitud} = \frac{1 + 950}{1 + 9 + 950 + 50} = 0.942$$

Curva ROC

- ROC = característica operativa del receptor
- Muestra el rendimiento de un modelo sobre predicciones binarias
- Se grafica al calcular la razón de falsos positivos (FPR) y razón de ciertos positivos (TPR), para cada valor posible de corte
- El área bajo la curva (AUC) también es usado como métrico

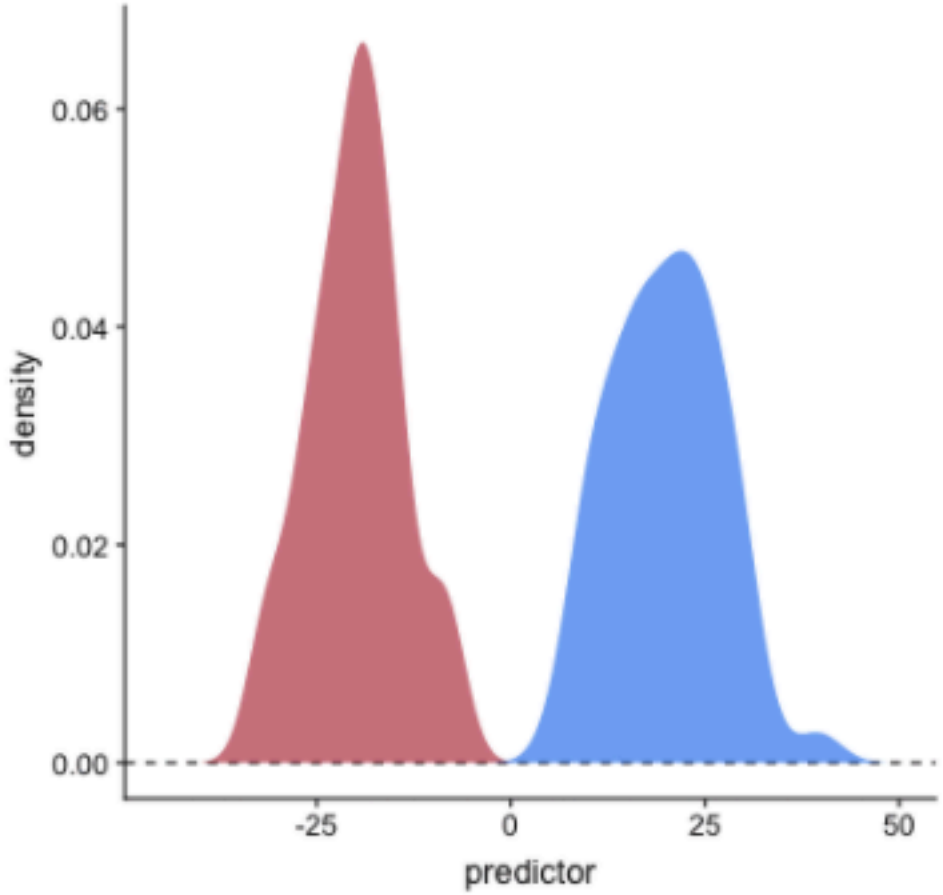


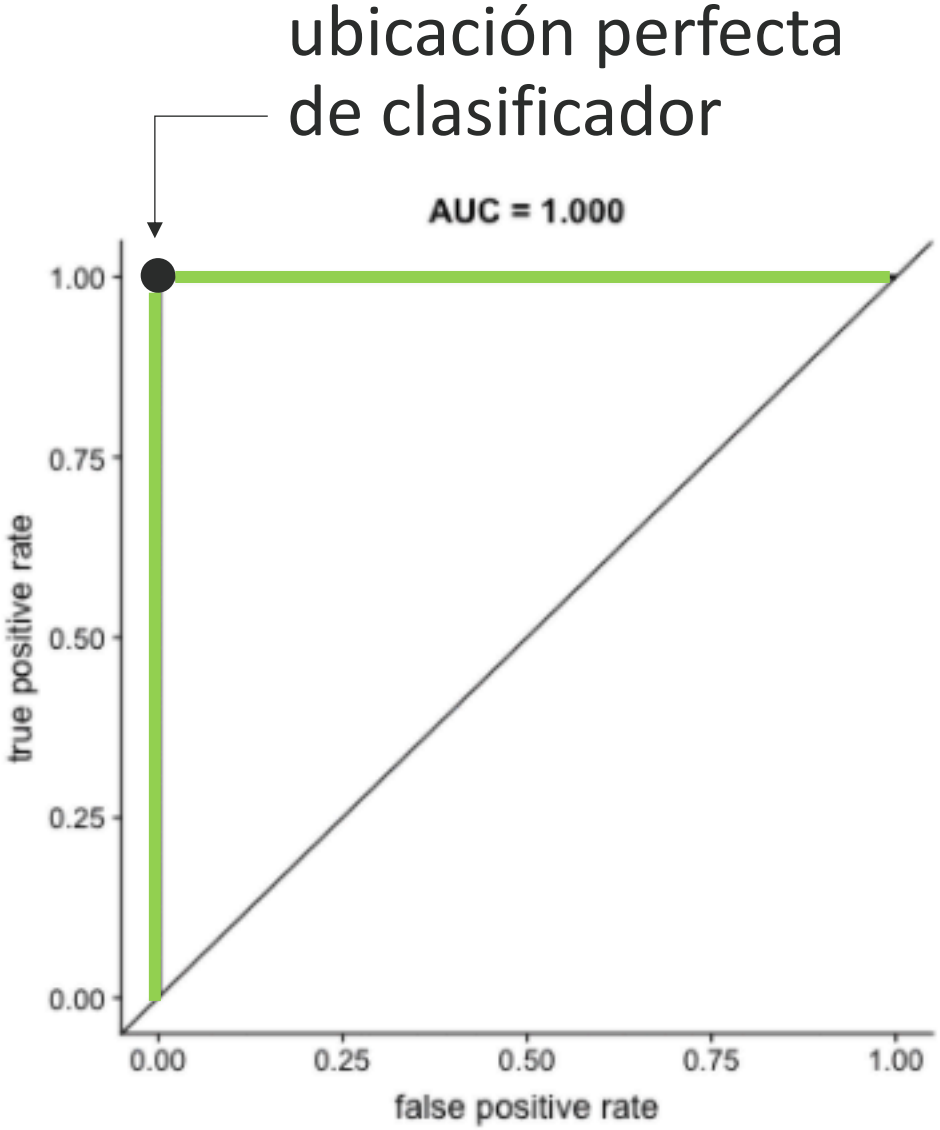
Curva ROC



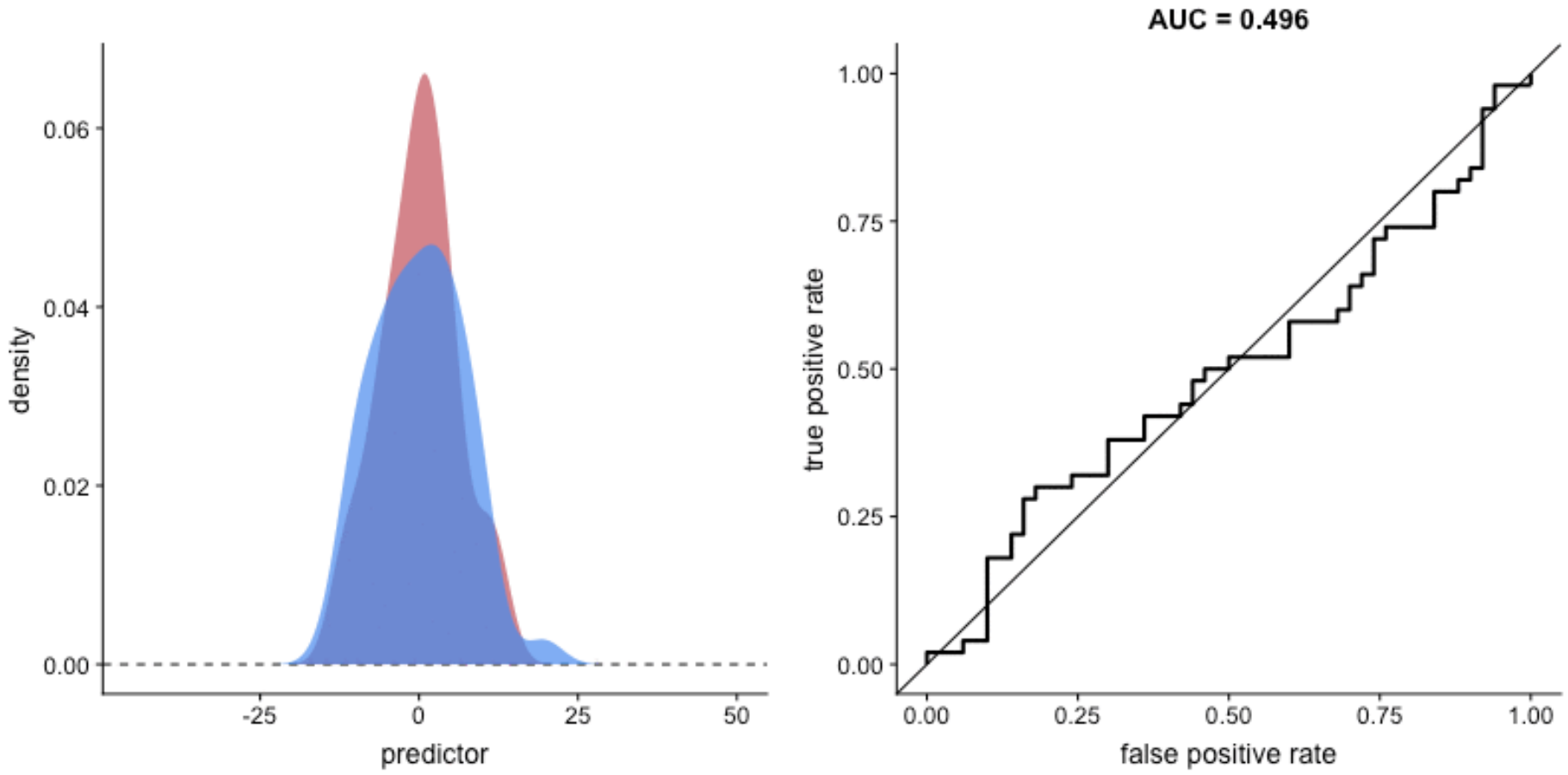
Fuente: https://github.com/dariyasydykova/open_projects/tree/master/ROC_animation

Curva ROC


0.06
0.04
0.02
0.00
density
-25
0
25
50
predictor


ubicación perfecta
de clasificador
AUC = 1.000
1.00
0.75
0.50
0.25
0.00
true positive rate
0.00
0.25
0.50
0.75
1.00
false positive rate

Curva ROC



Fuente: https://github.com/dariyasydykova/open_projects/tree/master/ROC_animation

Evaluación de Modelos en scikit-learn

Scoring	Function
Classification	
'accuracy'	metrics.accuracy_score
'balanced_accuracy'	metrics.balanced_accuracy_score
'average_precision'	metrics.average_precision_score
'neg_brier_score'	metrics.brier_score_loss
'f1'	metrics.f1_score
'f1_micro'	metrics.f1_score
'f1_macro'	metrics.f1_score
'f1_weighted'	metrics.f1_score
'f1_samples'	metrics.f1_score
'neg_log_loss'	metrics.log_loss
'precision' etc.	metrics.precision_score
'recall' etc.	metrics.recall_score
'jaccard' etc.	metrics.jaccard_score
'roc_auc'	metrics.roc_auc_score
'roc_auc_ovr'	metrics.roc_auc_score
'roc_auc_ovo'	metrics.roc_auc_score
'roc_auc_ovr_weighted'	metrics.roc_auc_score
'roc_auc_ovo_weighted'	metrics.roc_auc_score

```
>>> from sklearn.metrics import classification_report
>>> y_true = [0, 1, 2, 2, 0]
>>> y_pred = [0, 0, 2, 1, 0]
>>> target_names = ['class 0', 'class 1', 'class 2']
>>> print(classification_report(y_true, y_pred, target_names=target_names))
              precision    recall  f1-score   support

     class 0       0.67      1.00      0.80         2
     class 1       0.00      0.00      0.00         1
     class 2       1.00      0.50      0.67         2

    accuracy                           0.60         5
   macro avg       0.56      0.50      0.49         5
weighted avg       0.67      0.60      0.59         5
```

- https://scikit-learn.org/stable/modules/model_evaluation.html

Laboratorio 3
Algoritmos No Supervisados de ML

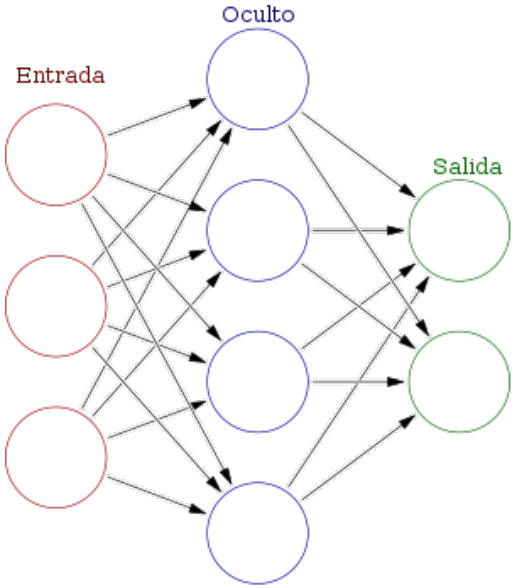
Break (5 min)

Siguientes pasos y Conclusiones

Deep Learning

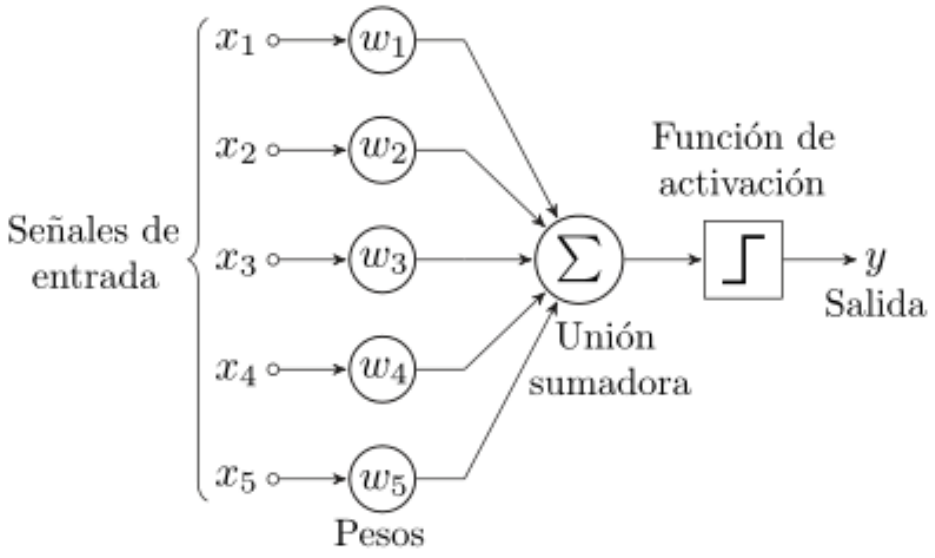
- Basados en redes neuronales, cuyo desarrollo comenzó en los 1940s
  - Neurona (perceptrón)
  - Algoritmo de retropropagación (backpropagation)
- Un modelo posee múltiples capas de unidades de procesamiento, cada una utilizando la capa anterior como su entrada.
- Popularidad creciente debido a la explosión de datos disponibles, aumento del poder computacional y avances algorítmicos

Red Neuronal

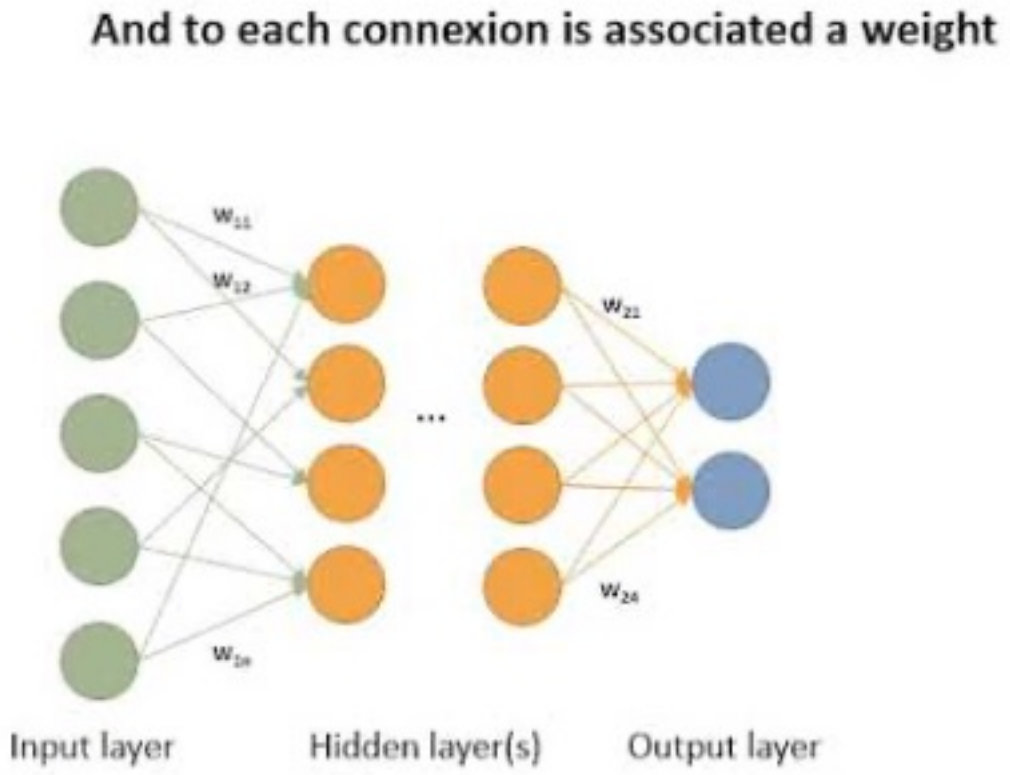


Deep Learning

Perceptron (Neurona)

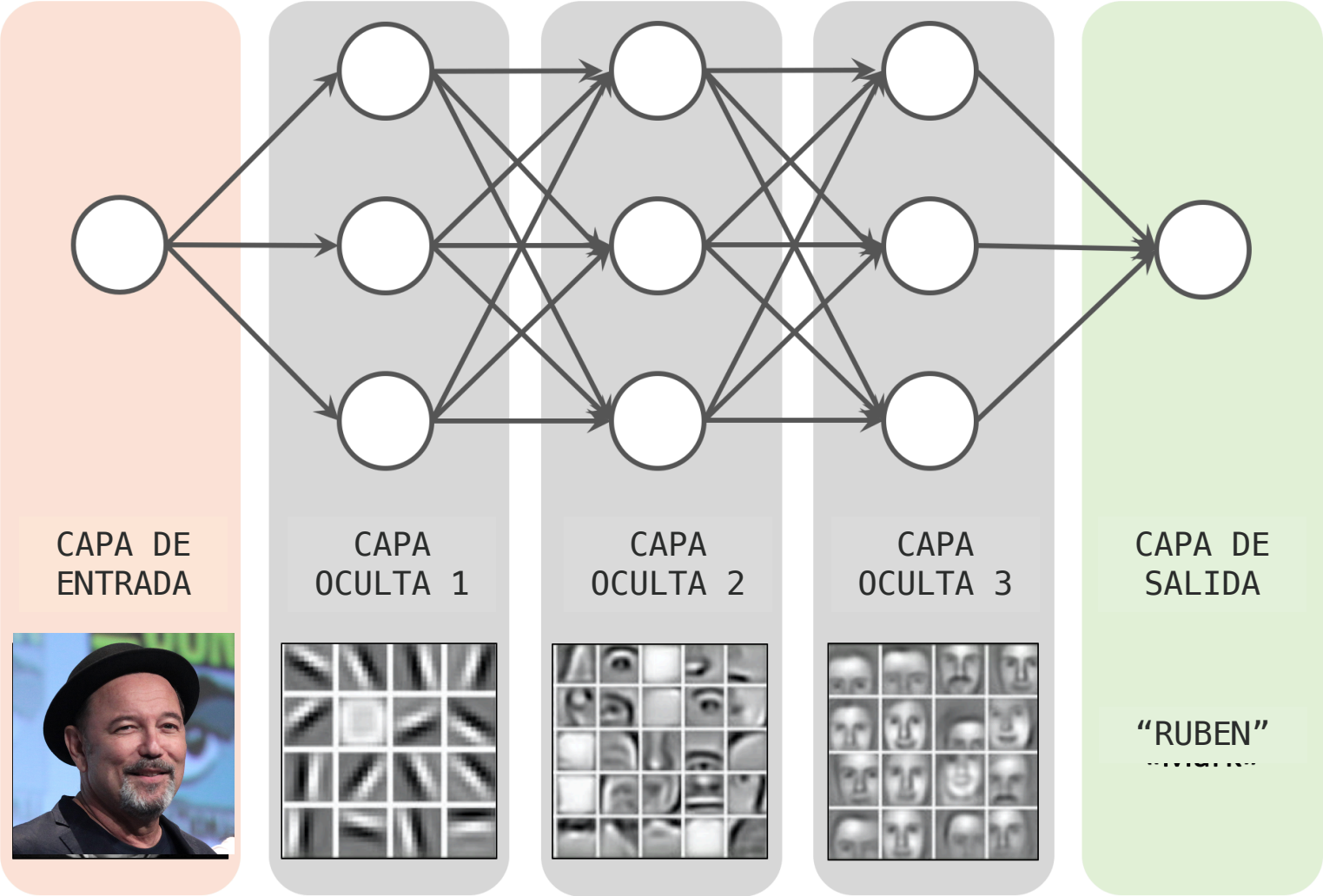


Modelo de Deep Learning



Deep Learning

- Consideran no sólo las características originales de los datos, sino también combinaciones de estos



Arquitecturas de Deep Learning

ARQUITECTURA	DESCRIPCION	EJEMPLOS DE USO
Feed-Forward Neural Network	• Arquitectura básica • Cada capa conectada hacia adelante	Usualmente primer arquitectura a probar
Convolutional Neural Network (CNN)	• Entrada que alimenta cada neurona esta delimitada por una ventana	Reconocimiento de imágenes, clasificación de imágenes, NLP
Autoencoder	• Busca comprimir y descomprimir con mínima diferencia entre entrada y salida	Comprimir imágenes
Generative Adversarial Network (GAN)	• Dos redes (generativa y discriminante) que compiten entre si para mejorar	
Recurrent Neural Network (RNN)	• Cuando el orden importa	Reconocimiento de escritura y habla, traducción de lenguaje, análisis de series de tiempo
Residual Network (ResNet)	• Usa conexiones que funcionan como saltos entre capas	Para resolver problemas complejos

Deep Learning

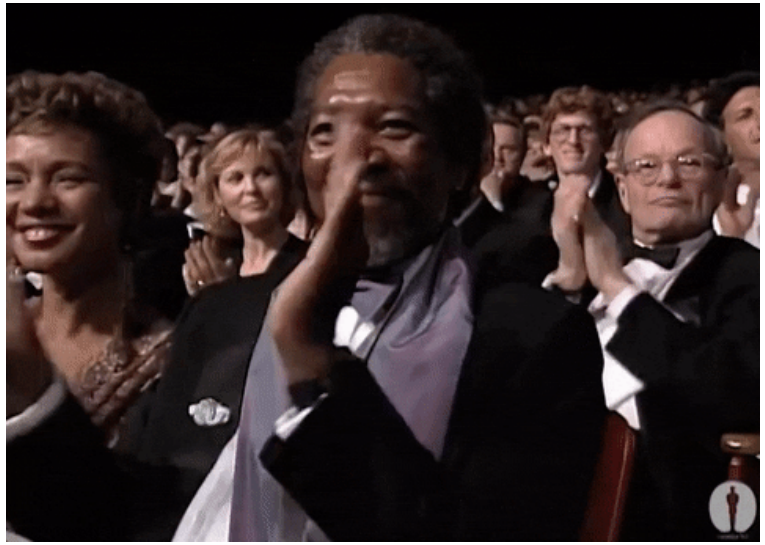
- Debido al calculo intensivo de parámetros, se utilizan procesadores especiales (FPGA, GPU, DLP)
  - Google Colab te permite usar servidores con estos procesadores, gratis!
- Principales librerías disponibles en Python: Tensorflow, Pytorch, Keras, Theano, MXnet
  - Scikit-learn no incluye algoritmos de deep learning
- Aunque muy poderoso, deep learning no es la solución a todo problema (al menos, por ahora)

Teorema Universal de Aproximación: dados suficientes neuronas y los parámetros correctos, una red neuronal puede básicamente emular cualquier tipo de comportamiento.

Preguntas?

Conclusiones

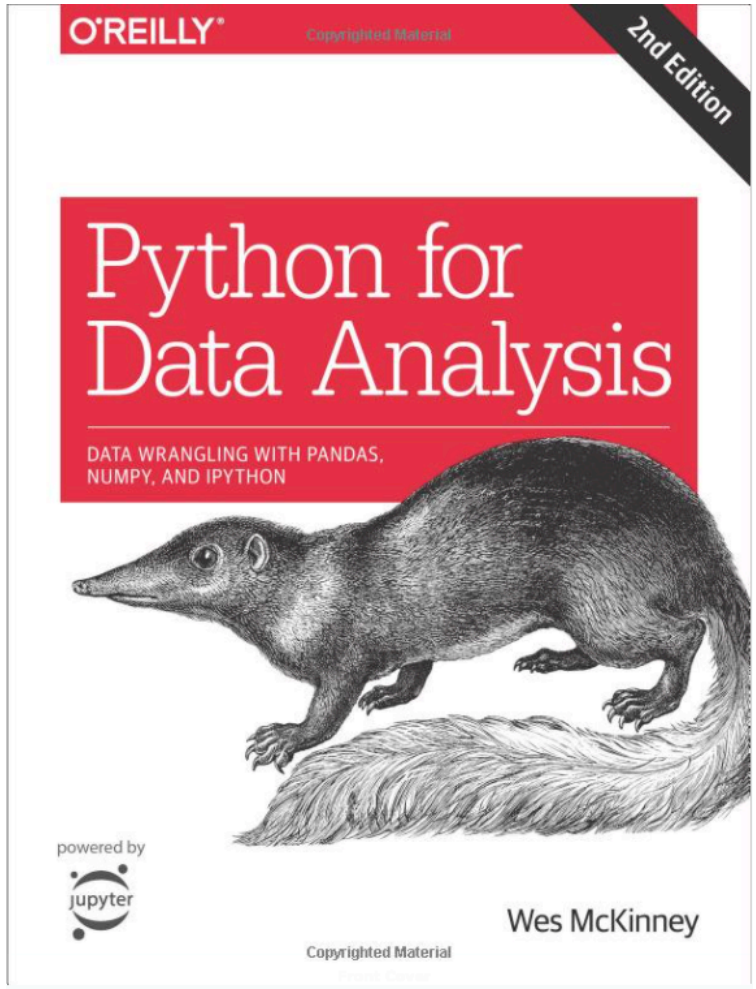
- Hemos cubierto bastante material de data science hoy, felicidades!
  - Análisis exploratorio de datos
  - Algoritmos de aprendizaje supervisado
  - Algoritmos de aprendizaje no supervisado
  - Complejidad del modelo (sobreajuste y desajuste)
  - Algoritmos de clustering
  - Métricos para evaluación de modelos
  - Un poco de deep learning!
- Pero esto es sólo el comienzo

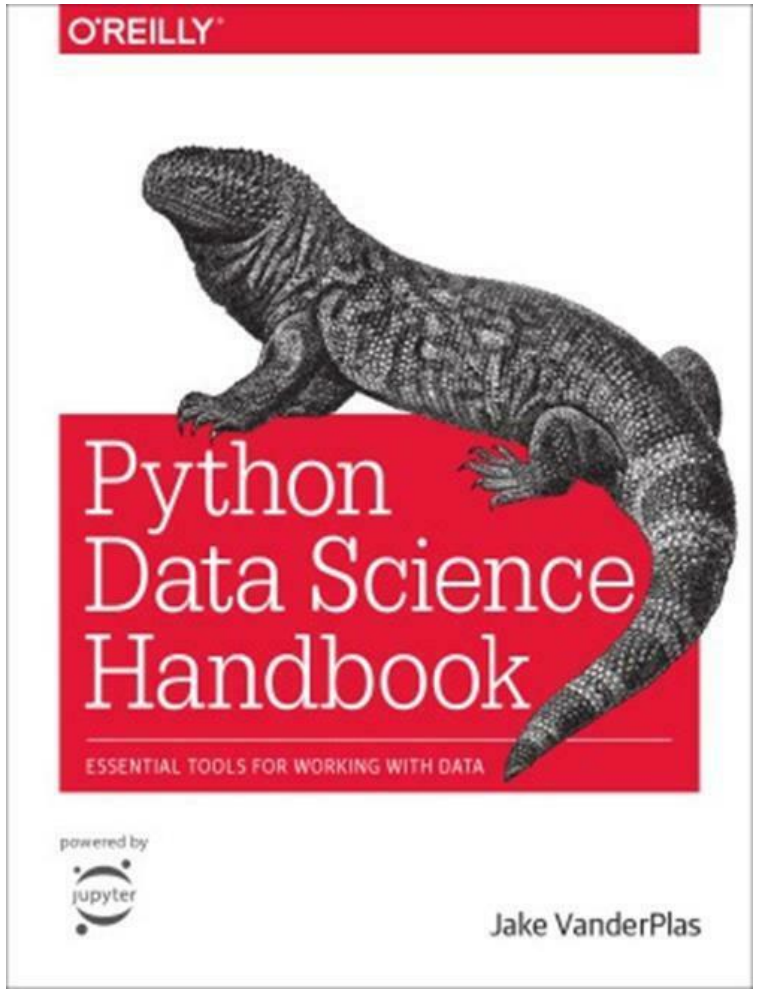


Conclusiones

- Python no es el único lenguaje disponible. Otra opción: Lenguaje R
- Algoritmos de ML hacen bien una sola tarea
  - Están abiertos para configuración con parámetros
  - Cerrados para modificación de su función (corazón del algoritmo)
- Siempre considera la sustitución de Liskov
  - Puedes reemplazar los algoritmos, hacen lo mismo pero de forma diferente
- Importancia de las características que utilizas
- Desbalance de clases (casi) siempre presente
- Se escéptico a los modelos perfectos, son señal de que algo anda mal
- Ventajas de experto del área/tema de interés

Referencias



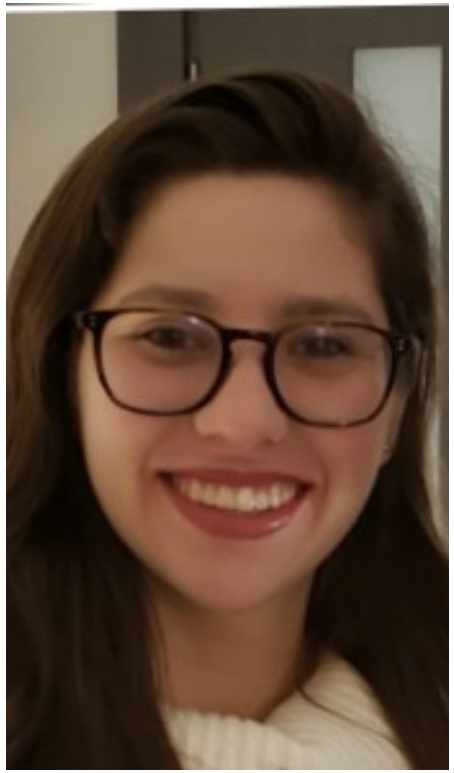


Referencias

- Tutoriales de PyData y PyCon → https://pyvideo.org
- 660 cursos gratuitos sobre programación y Python → http://bit.ly/2DO5UkW (https://medium.freecodecamp.org)
- Jupyter Notebooks sobre seguridad, por Click Security → https://github.com/ClickSecurity/data_hacking
  - Versión en Español de caso DGA → https://github.com/gmhoward/dojo-pty-2018/
- Towards Data Science Blog → http://towardsdatascience.com
- Curso de Deep Learning → http://www.fast.ai
- https://machinelearningknowledge.ai/ (tutoriales sobre Pandas y Numpy)
- Para revisar conceptos → Wikipedia!

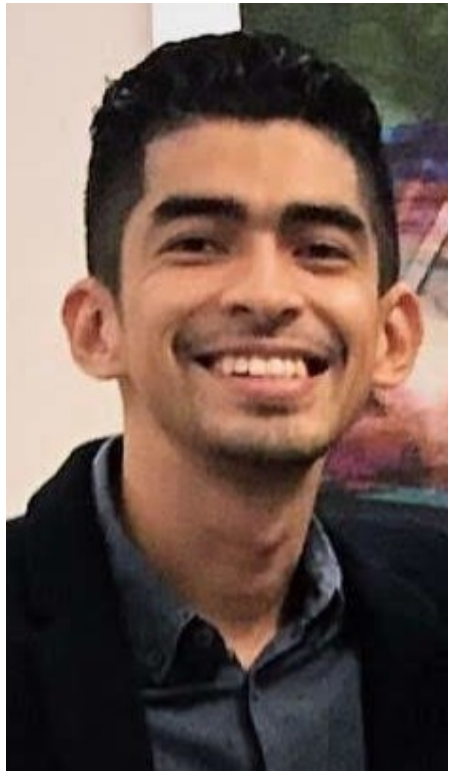
Gracias al equipo!

Adriaan



Eliana

Yiseika



Arnulfo

Muchas gracias!

gaspar@acm.org
@gmodeloh