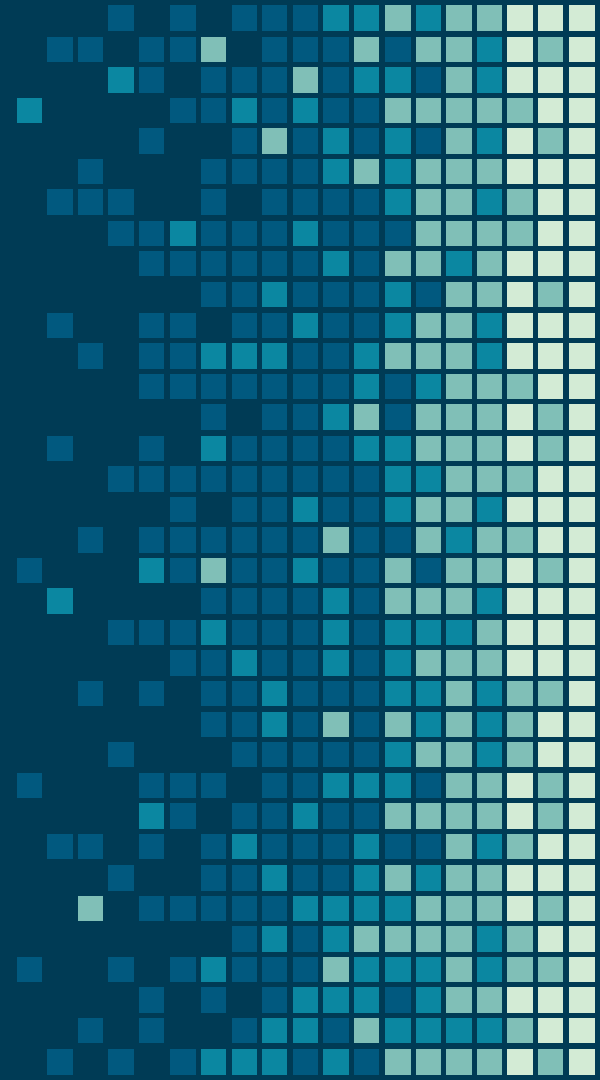


Aprendizaje Automático

Ing. Luis Martínez, MSc

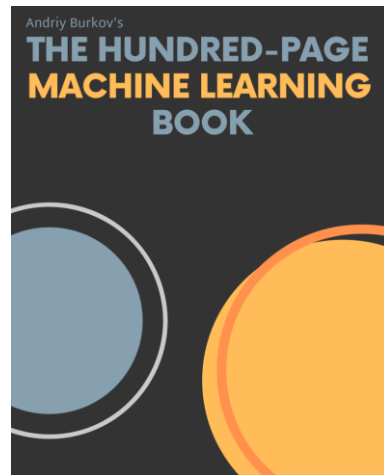
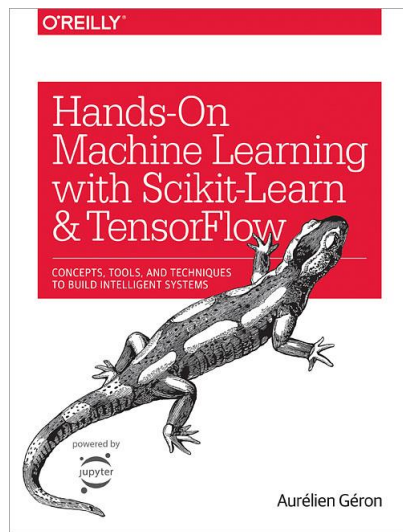
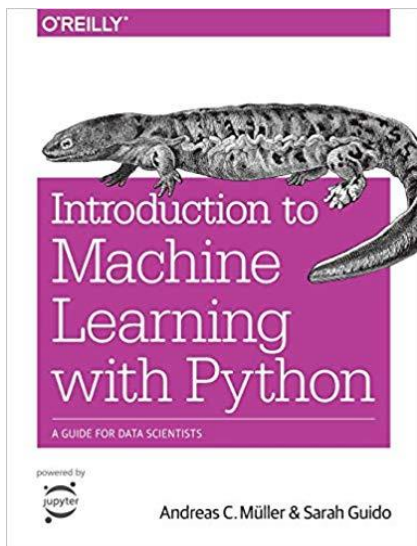


Contenido

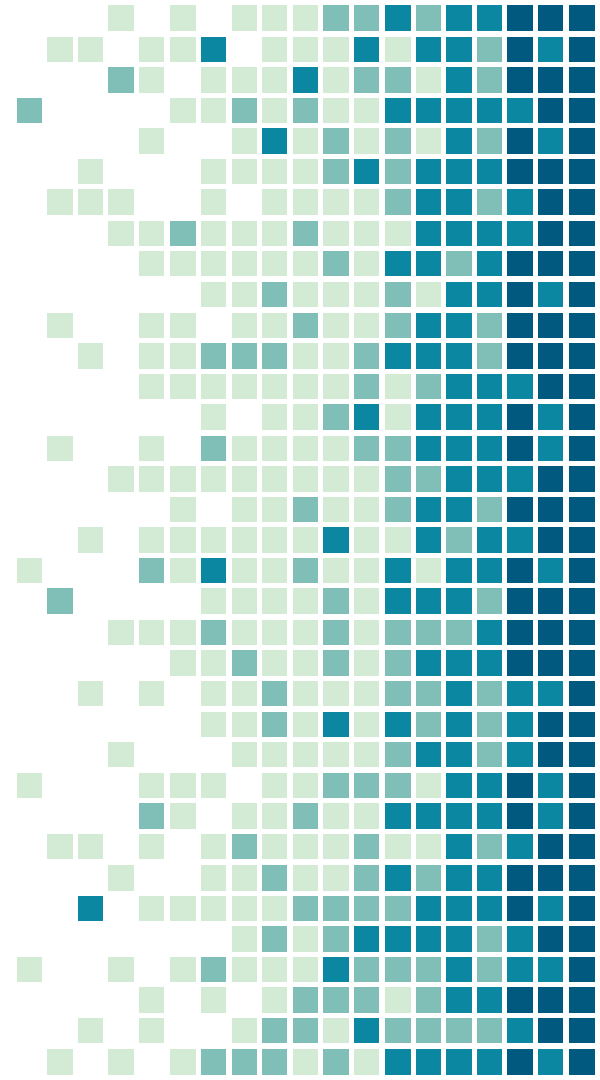
- Introducción
- Generalidades de los Modelos
- Clasificación
- Regresión
- Clustering
- Reglas de asociación



Libros sugeridos para lectura personal



1. ¿Qué es Aprendizaje Automático?



Programación Tradicional

```
[11] def spam_filter(email):  
    """Function that labels an email as 'spam' or 'not spam'"""  
    if 'Act now!' in email.contents:  
        label = 'spam'  
    elif 'hotmail.com' in email.sender:  
        label = 'spam'  
    elif email.contents.count('$') > 20:  
        label = 'spam'  
    else:  
        label = 'not spam'  
  
    return label
```

“ *Campo de estudio que le da a la computadora la habilidad de aprender sin ser explícitamente programadas.*

*- Arthur Samuel**

“ Es un programa de computador que aprende de la **Experiencia E** , respecto a alguna **tarea T** y con medida de **rendimiento P** , si el desempeño sobre la tarea T , medido por P , mejora con la experiencia E .

- Tom Mitchell

Breve Pregunta

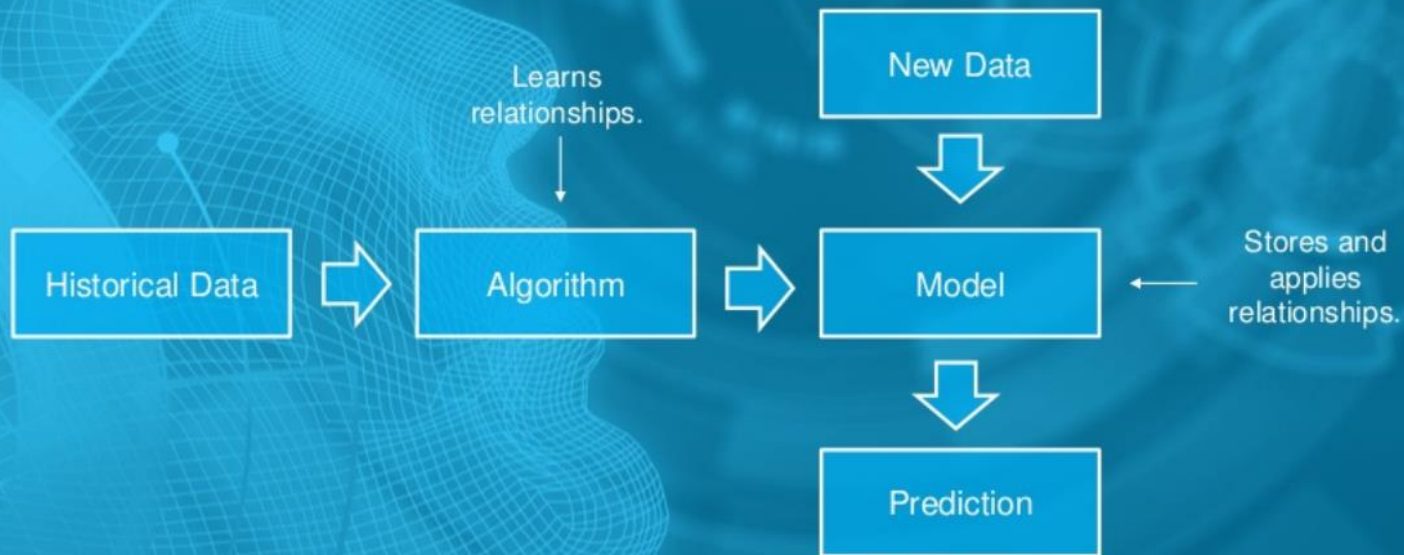
Tu programa válida transacciones fraudulentas o no. El cliente deja saber que es fraudulenta cuando llama a reportar que fue así (de esta manera, nuestro programa aprende).

¿Cuál es la tarea T?

¿Cuál es la experiencia E?

¿Cuál es la medida P?



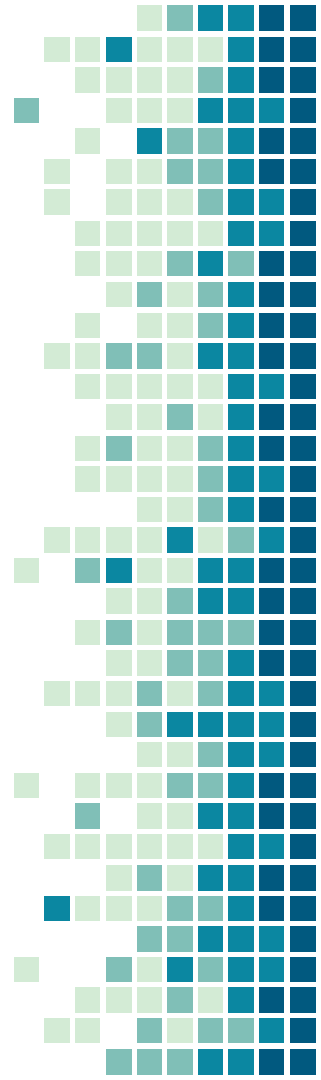
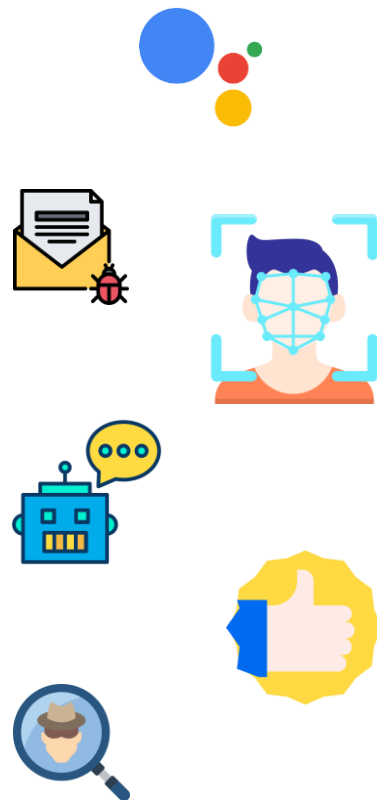


2. Usos de Aprendizaje Automático



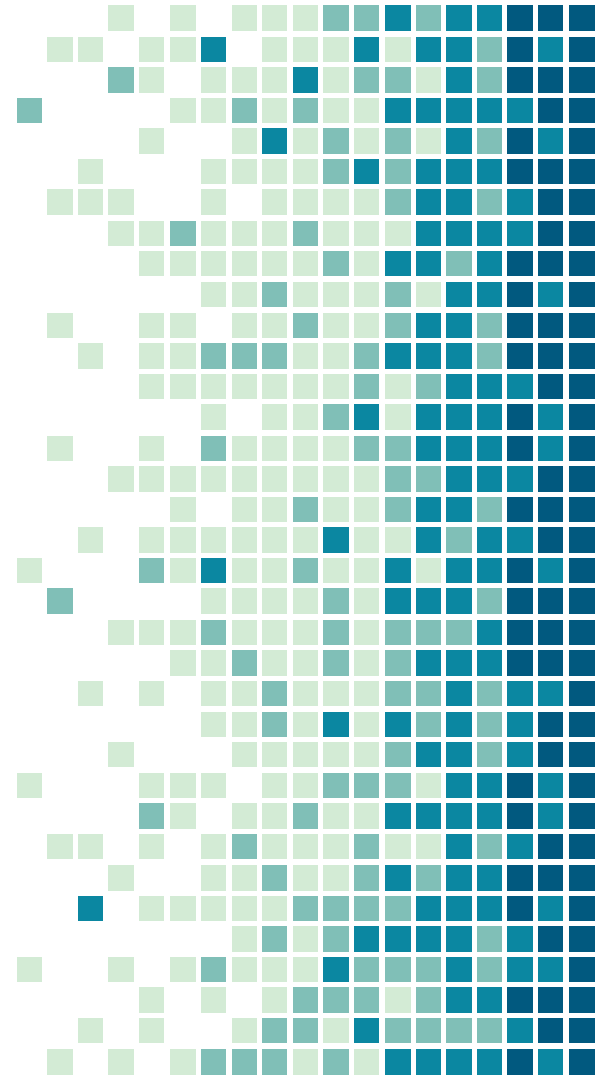
Algunos casos:

- Customer Churn
- Facilitador de Prestamos
- Asistentes Personales Virtuales
- Predicción de Tráfico
- Reconocimiento facial
- Detección de SPAM y Malware
- Chatbots
- Recomendador de Productos
- Entre otros...

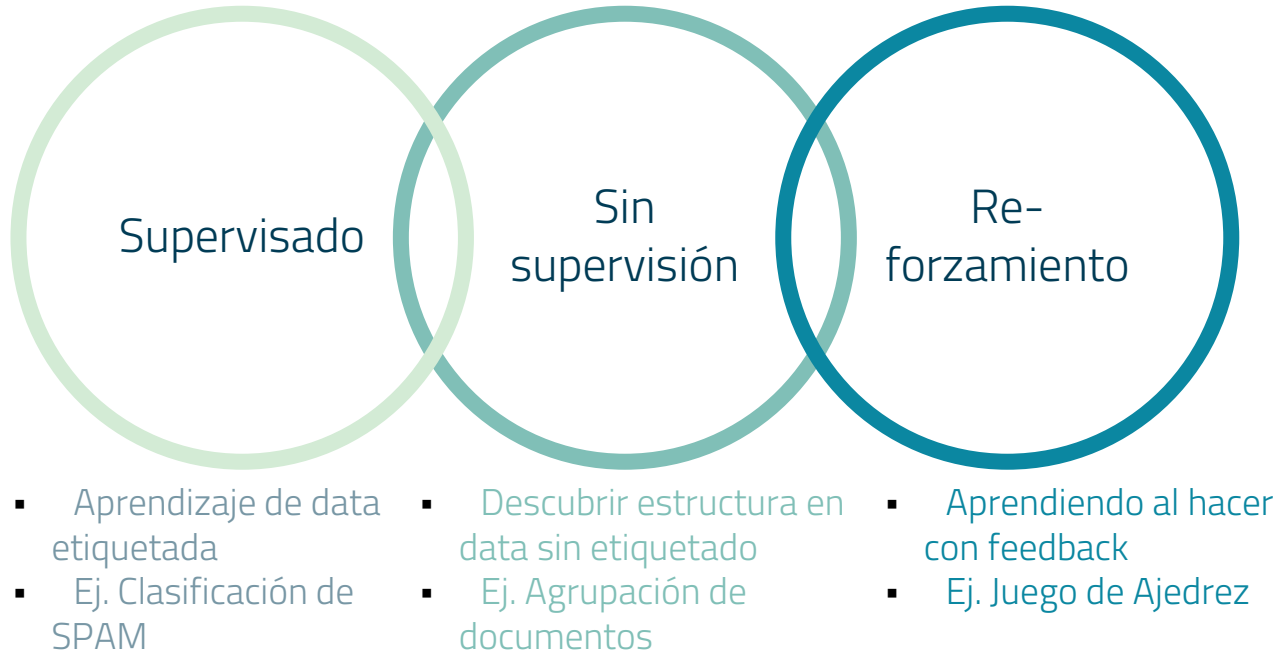


3.

Tipos de Aprendizaje Automático



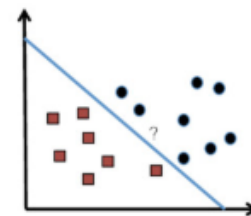
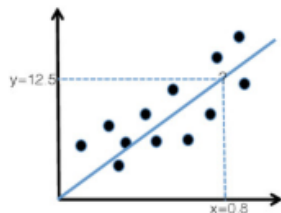
Tipos de Aprendizaje



Supervisado

Clasificación

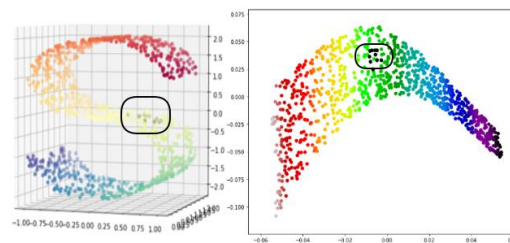
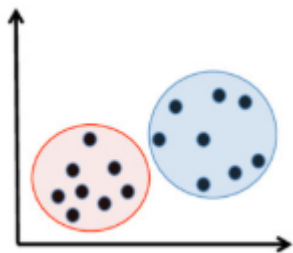
Regresión



Sin
Supervisión

Reducción

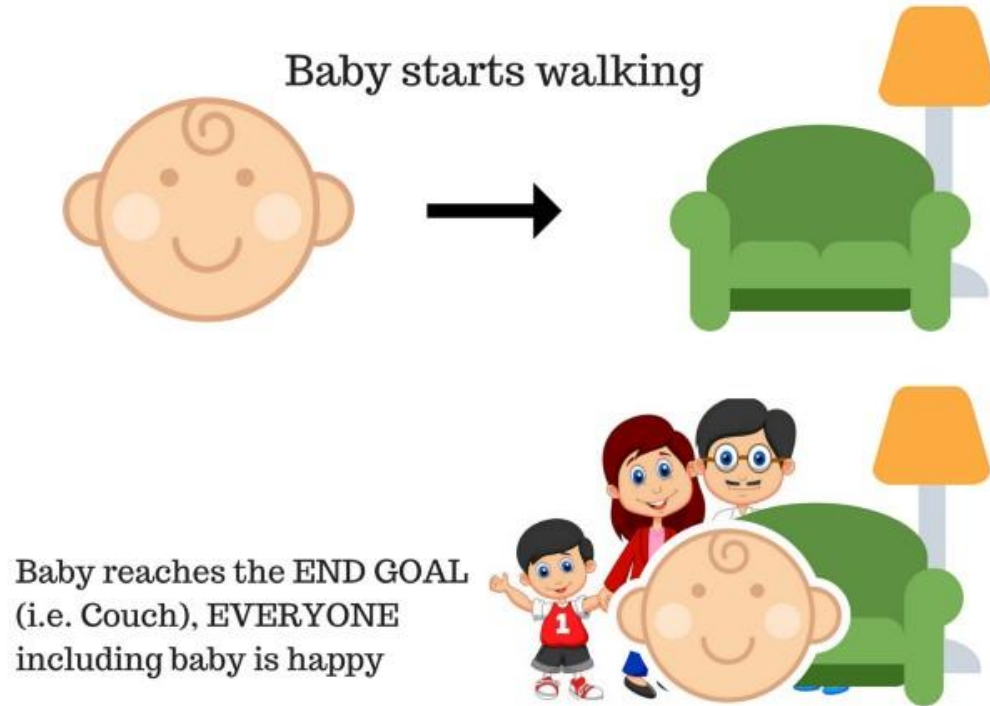
Clustering



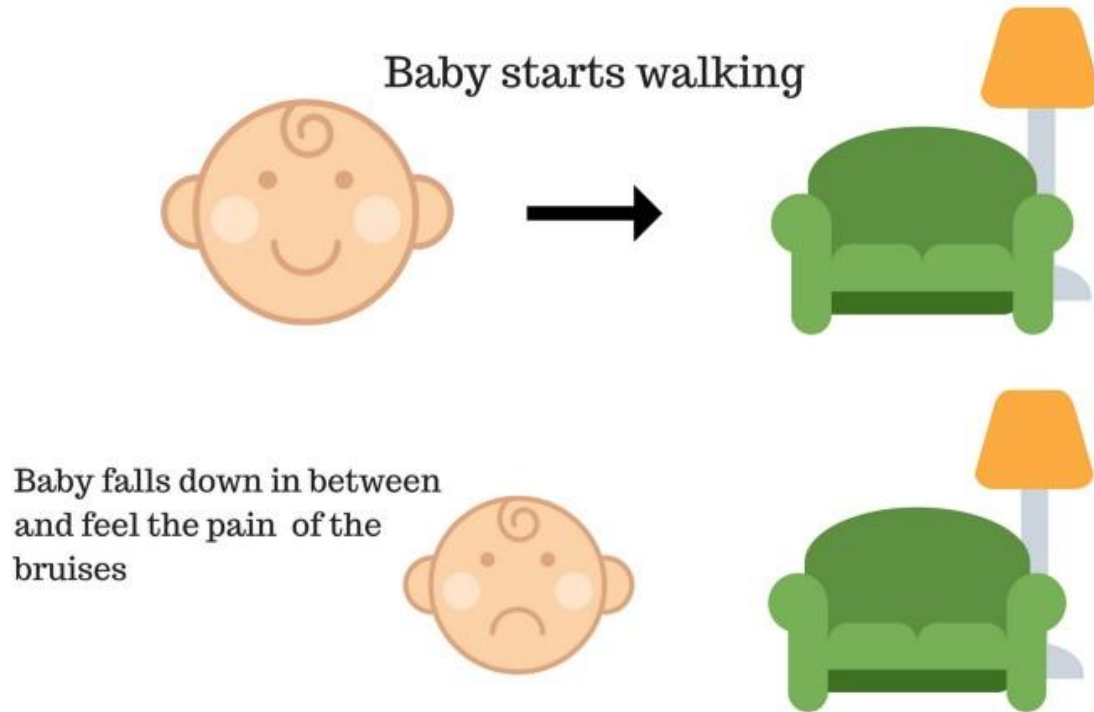
Reinforcement Learning



Reinforcement Learning



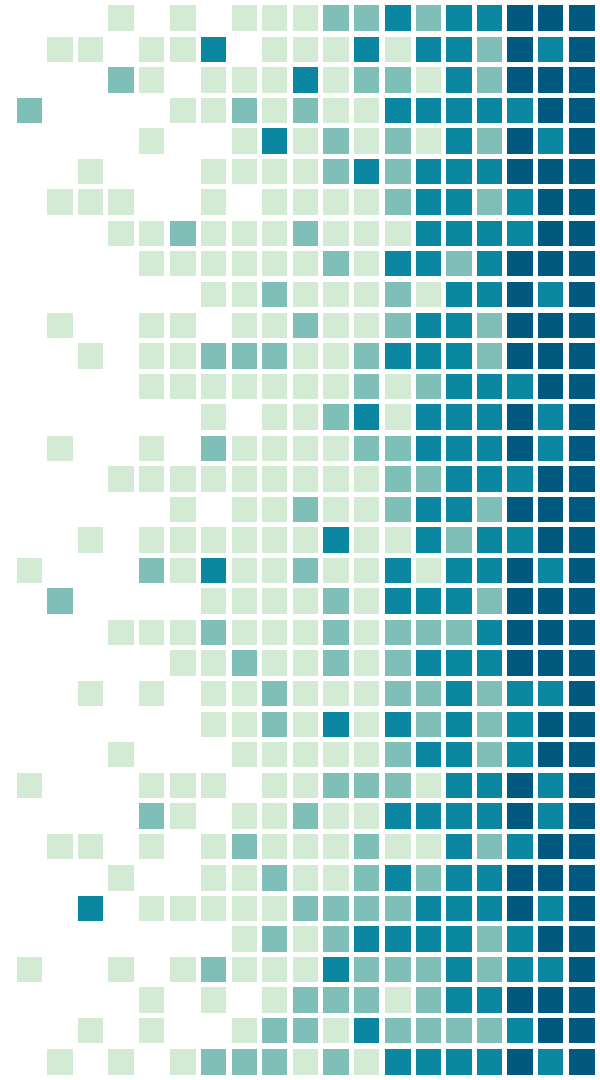
Reinforcement Learning



Datos como numpy o pandas
Características continuas
No valores perdidos en data

4. Scikit-learn

Librería para Machine Learning



Funciones de algebra linear
Fancy Indexing
Mayor Rápidez

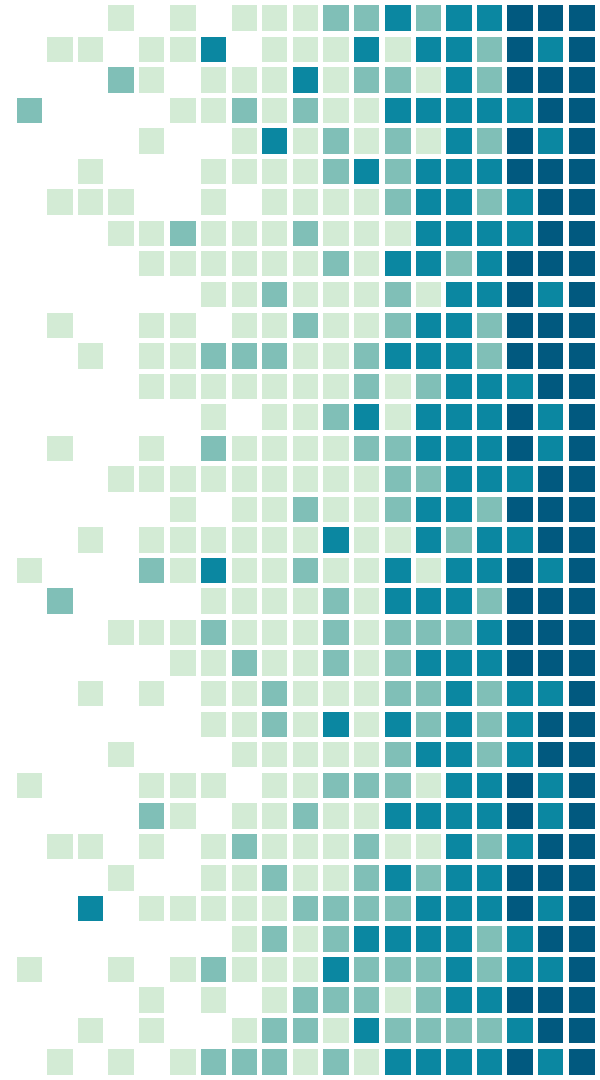
5. NumPy

Construida alrededor de un arreglo C



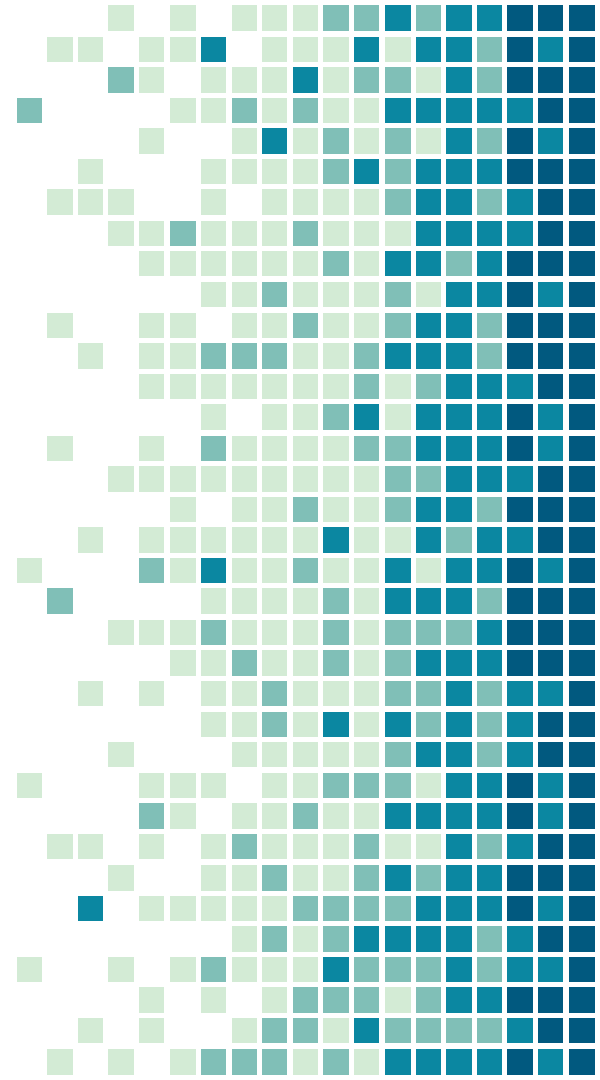
6. Pandas & Matplotlib

Manipulación de datos, análisis y
visualizaciones



7. Representación de Data

IRIS Data



Clasificación de Flores (IRIS)

Iris-Setosa



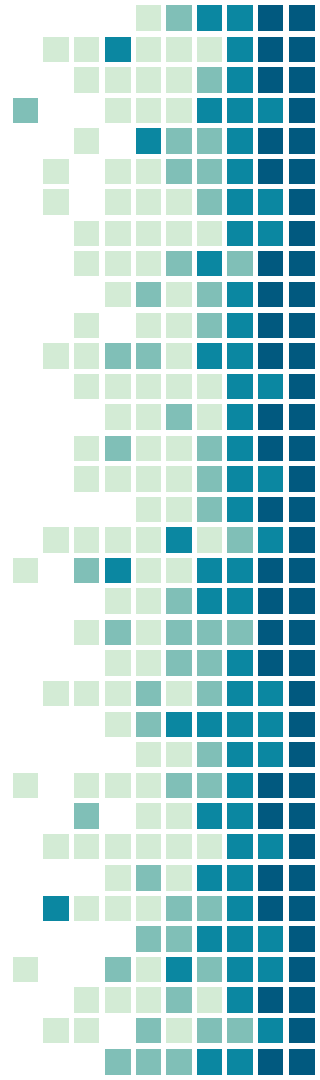
Iris-Versicolor



Iris-Setosa

Datos de la base de Flores Iris

- Consiste en 150 ejemplos (flores individuales) que tienen 4 características: largo del sépalo, ancho del sépalo, largo de pétalos y ancho de pétalos (todo en cm).
- Data es accesible gratis desde [UCI Machine Learning Repository](#)

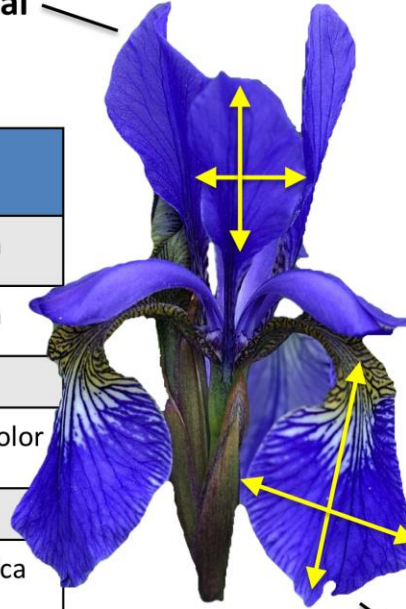


Representación

Ejemplos, casos, observaciones...

	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

Petal

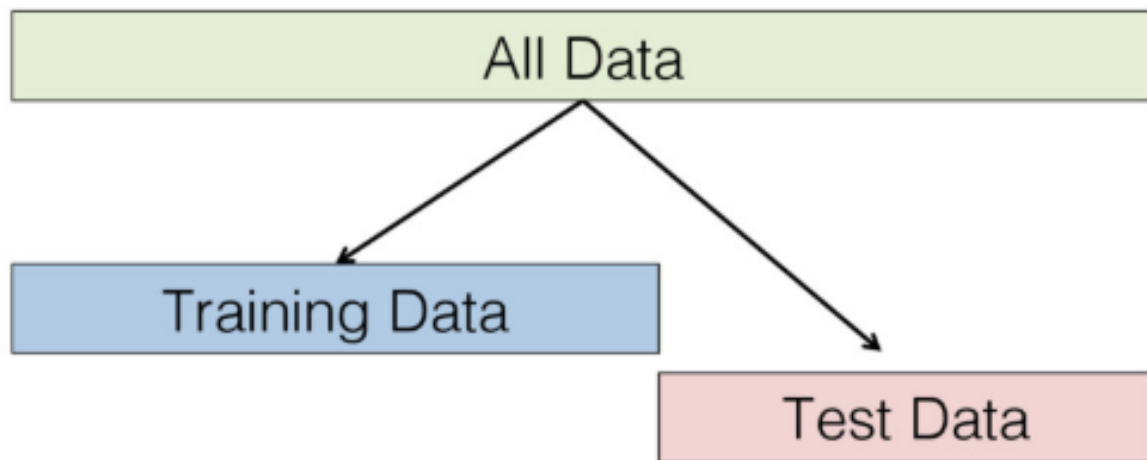


Sepal

Clase, Y, etiqueta, objetivo...

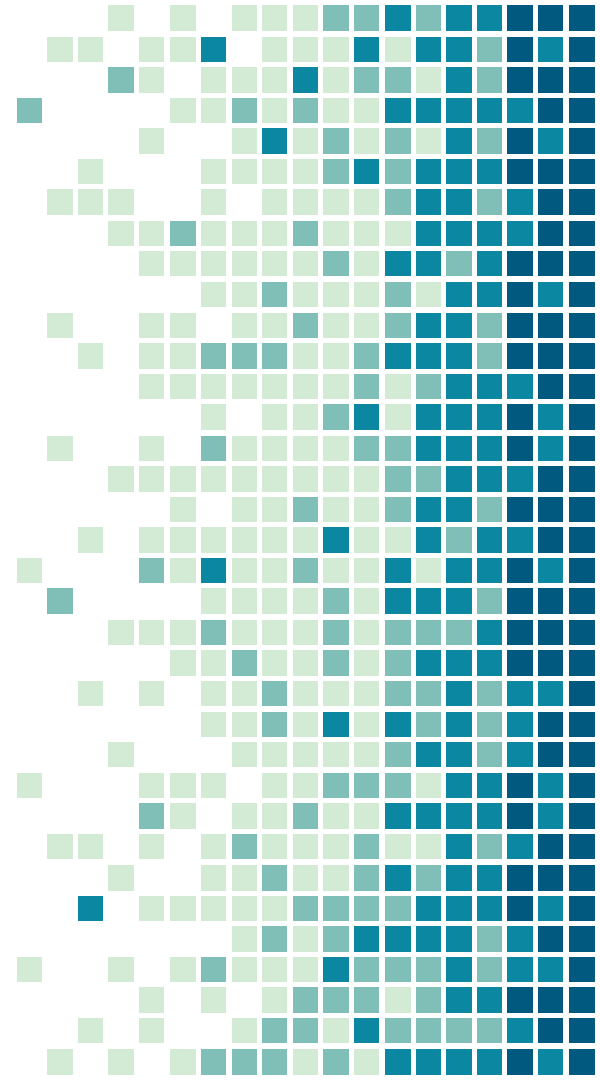
Atributos, características, dimensiones...

Base de Entrenamiento y Prueba

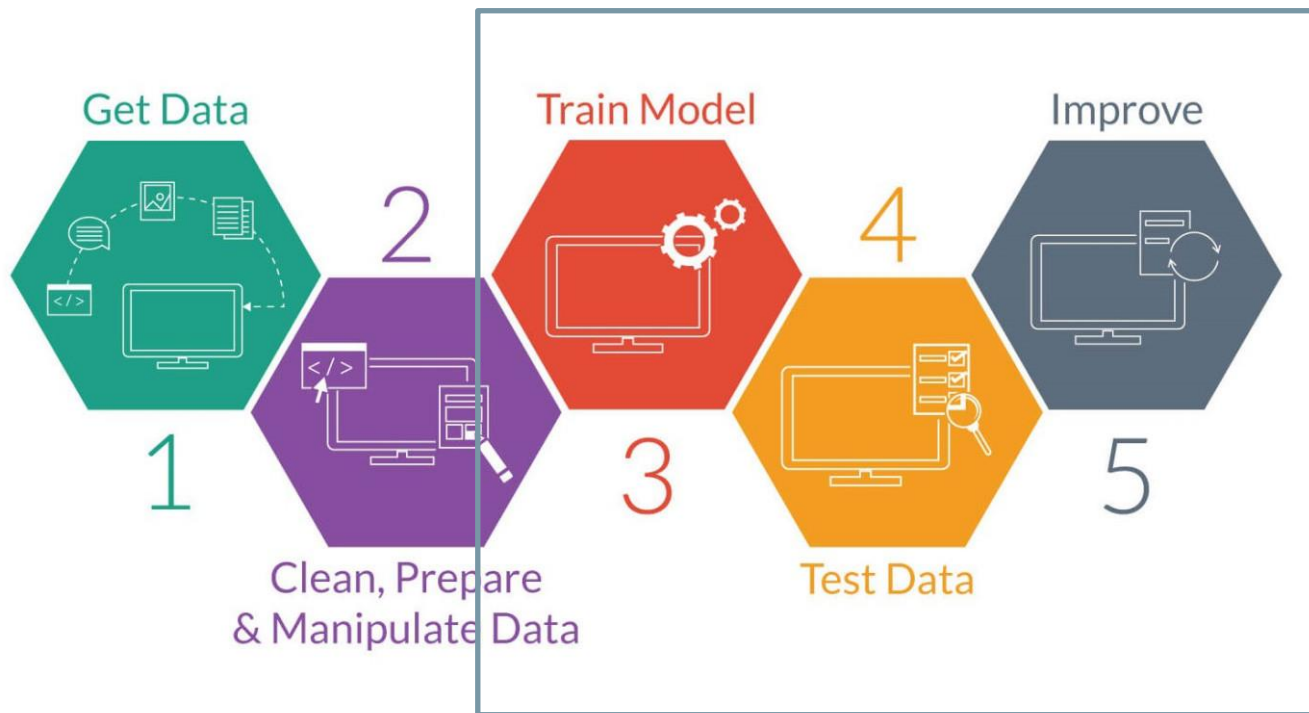


6.

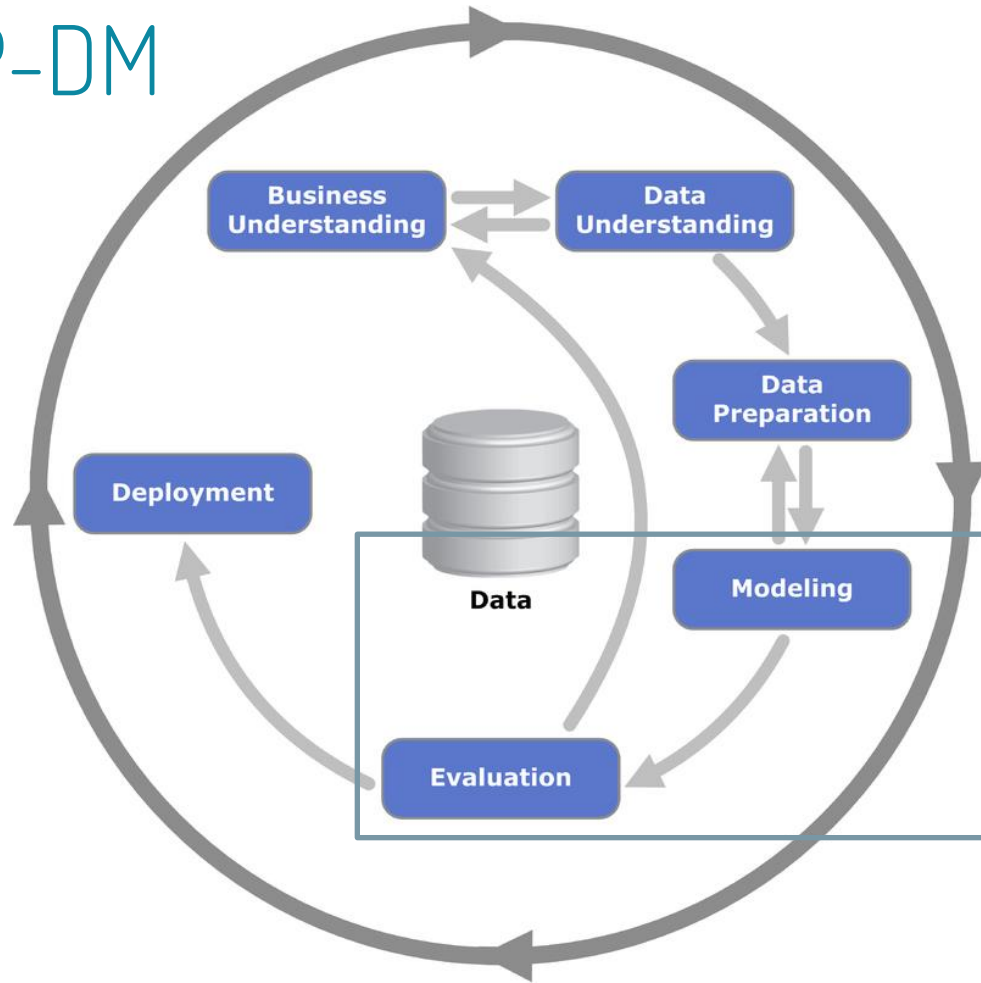
Flujo de un Proyecto de Aprendizaje Automático



Flujo



Flujo: CRISP-DM



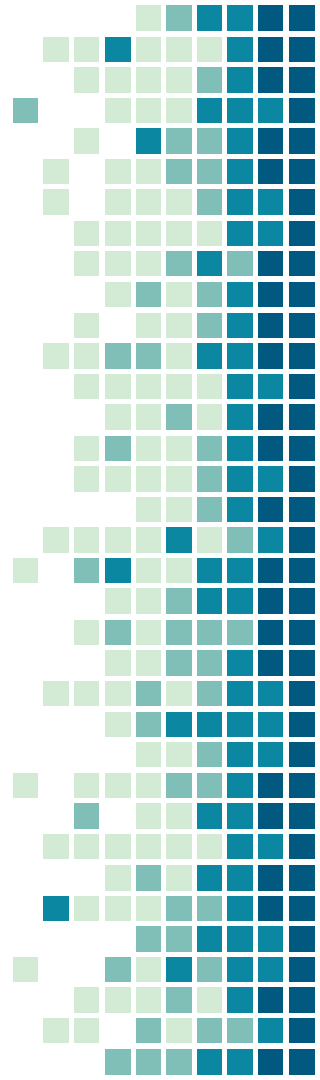
¿Qué pregunta (s) estoy tratando de responder? ¿Creo que los datos recopilados pueden responder esa pregunta?

¿Cuál es la mejor manera de formular mis preguntas como un problema de aprendizaje automático?

¿He recopilado suficientes datos para representar el problema que quiero resolver?

¿Qué características de los datos extraje, y estas permitirán la correcta predicciones?

¿Cómo mediré el éxito en mi solicitud?



1. Obtención de Datos

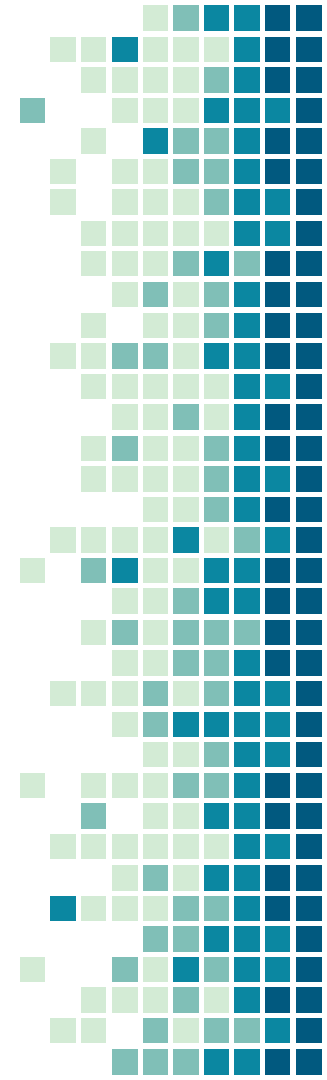


kaggle

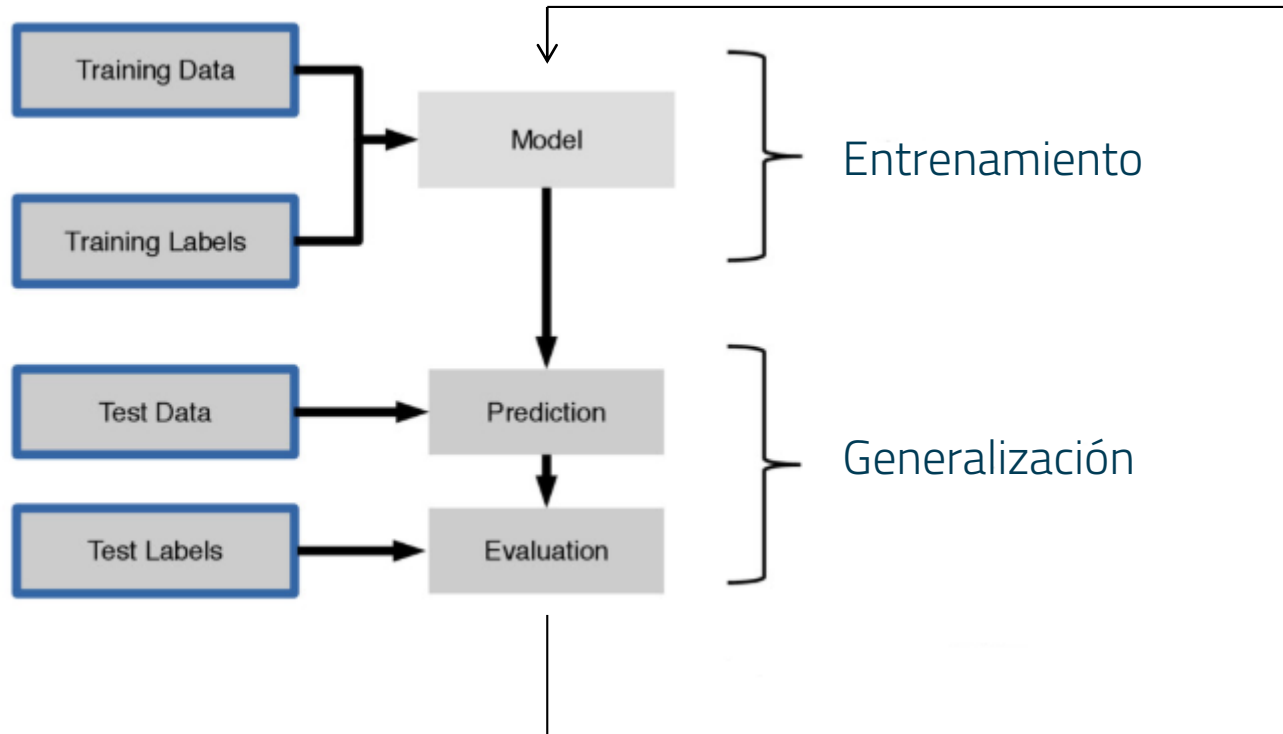


2. Pre-procesamiento de Datos

- Limpieza de datos (valores perdidos, etc)
- Análisis Exploratorios
- Creación de atributos (características)
- Reducción de atributos (compresión de dimensiones)



Flujo de trabajo en Supervisado




Estimadores en Scikit-learn

- Algoritmos son implementados como clases estimadoras
- Cada Algoritmo está bien documentado, usos y ejemplos

```
[1] from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor
    from sklearn.neighbors import KNeighborsClassifier, KNeighborsRegressor
    from sklearn.ensemble import RandomForestClassifier, GradientBoostingRegressor
    from sklearn.svm import SVC, SVR
    from sklearn.linear_model import LinearRegression, LogisticRegression
```

Estimadores en Scikit-learn



[Home](#) [Installation](#) [Documentation](#) [Examples](#)

[Previous
sklearn.svm.SVC](#) [Next
sklearn.tree.DecisionTreeClassifier](#) [Up
API Reference](#)

[scikit-learn v0.21.3](#)
[Other versions](#)

Please **cite us** if you use the software.

[sklearn.tree.DecisionTreeClassifier](#)
Examples using `sklearn.tree.DecisionTreeClassifier`

sklearn.tree.DecisionTreeClassifier

```
class sklearn.tree.DecisionTreeClassifier(criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None, presort=False)
```

[\[source\]](#)

A decision tree classifier.

Read more in the [User Guide](#).

Parameters:

- criterion** : *string, optional (default="gini")*
The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.
- splitter** : *string, optional (default="best")*
The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the best random split.
- max_depth** : *int or None, optional (default=None)*
The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.
- min_samples_split** : *int, float, optional (default=2)*
The minimum number of samples required to split an internal node:
 - If int, then consider min_samples_split as the minimum number.
 - If float, then min_samples_split is a fraction and `ceil(min_samples_split * n_samples)` are the minimum number of samples for each split.

Estimadores en Scikit-learn

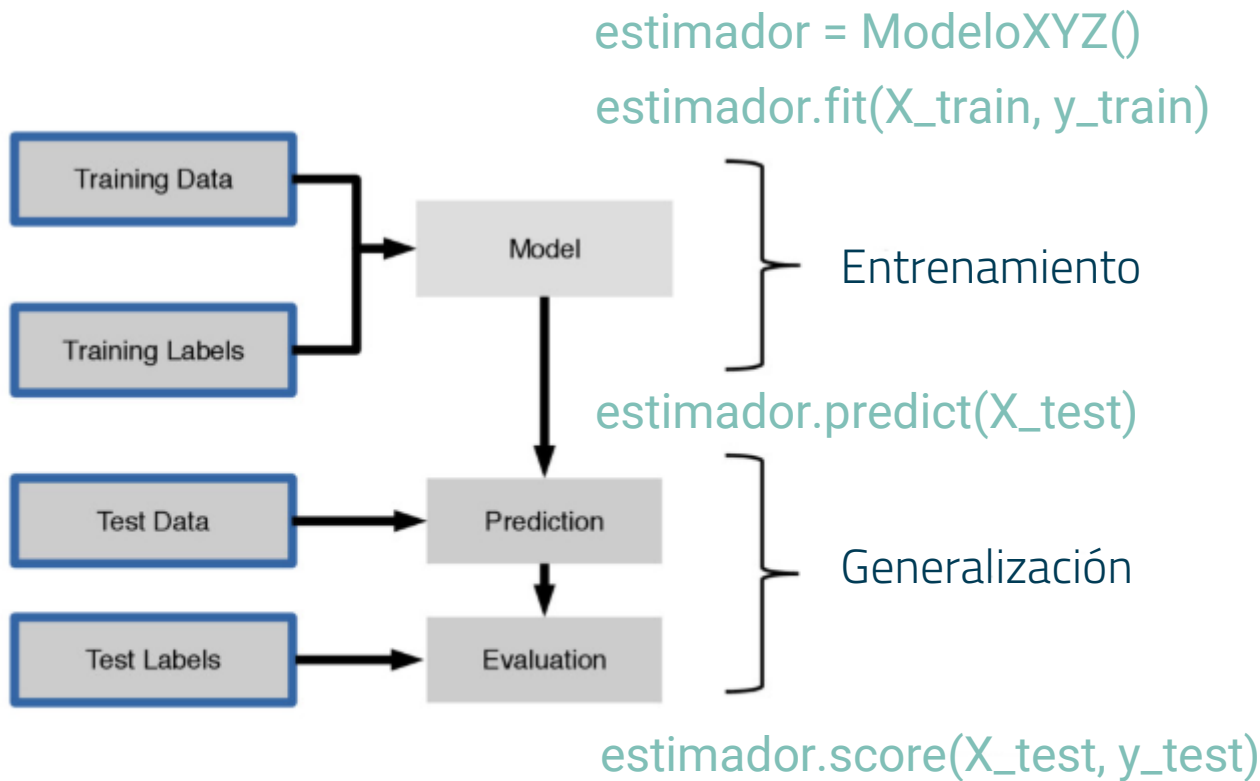
Examples

```
>>> from sklearn.datasets import load_iris
>>> from sklearn.model_selection import cross_val_score
>>> from sklearn.tree import DecisionTreeClassifier
>>> clf = DecisionTreeClassifier(random_state=0)
>>> iris = load_iris()
>>> cross_val_score(clf, iris.data, iris.target, cv=10)
...
array([ 1.        ,  0.93... ,  0.86... ,  0.93... ,  0.93... ,
        0.93... ,  0.93... ,  1.        ,  0.93... ,  1.        ])
```

Methods

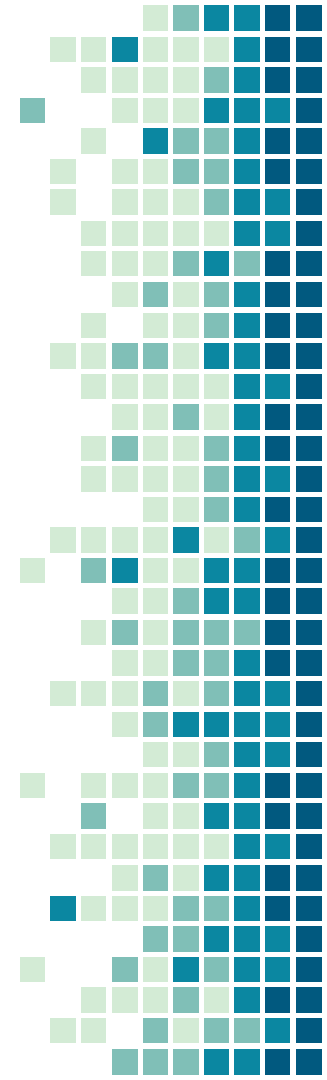
<code>apply</code> (self, X[, check_input])	Returns the index of the leaf that each sample is predicted as.
<code>decision_path</code> (self, X[, check_input])	Return the decision path in the tree
<code>fit</code> (self, X, y[, sample_weight, ...])	Build a decision tree classifier from the training set (X, y).
<code>get_depth</code> (self)	Returns the depth of the decision tree.
<code>get_n_leaves</code> (self)	Returns the number of leaves of the decision tree.
<code>get_params</code> (self[, deep])	Get parameters for this estimator.
<code>predict</code> (self, X[, check_input])	Predict class or regression value for X.
<code>predict_log_proba</code> (self, X)	Predict class log-probabilities of the input samples X.
<code>predict_proba</code> (self, X[, check_input])	Predict class probabilities of the input samples X.
<code>score</code> (self, X, y[, sample_weight])	Returns the mean accuracy on the given test data and labels.
<code>set_params</code> (self, **params)	Set the parameters of this estimator.

Flujo de trabajo en Supervisado



Medidas de Evaluación

- Precisión (Accuracy)
- Matrix de Confusión
- Area debajo de la Curva
- Valor F1
- Error Medio Absoluto
- Error Cuadrático Medio

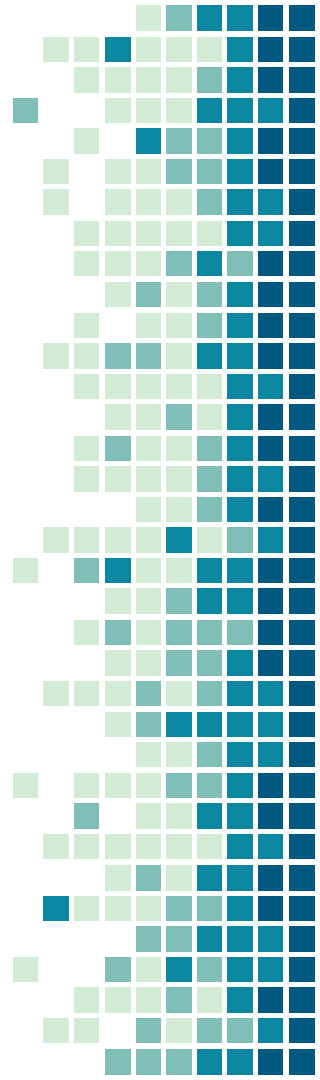


Precisión (Accuracy)

- Es lo que usualmente llamamos la Precisión (Accuracy) del Modelo

$$Accuracy = \frac{\text{Número de Predicciones Correctas}}{\text{Total número de Predicciones hechas}}$$

- Funciona si *existe número igual de ejemplos para cada clase.*

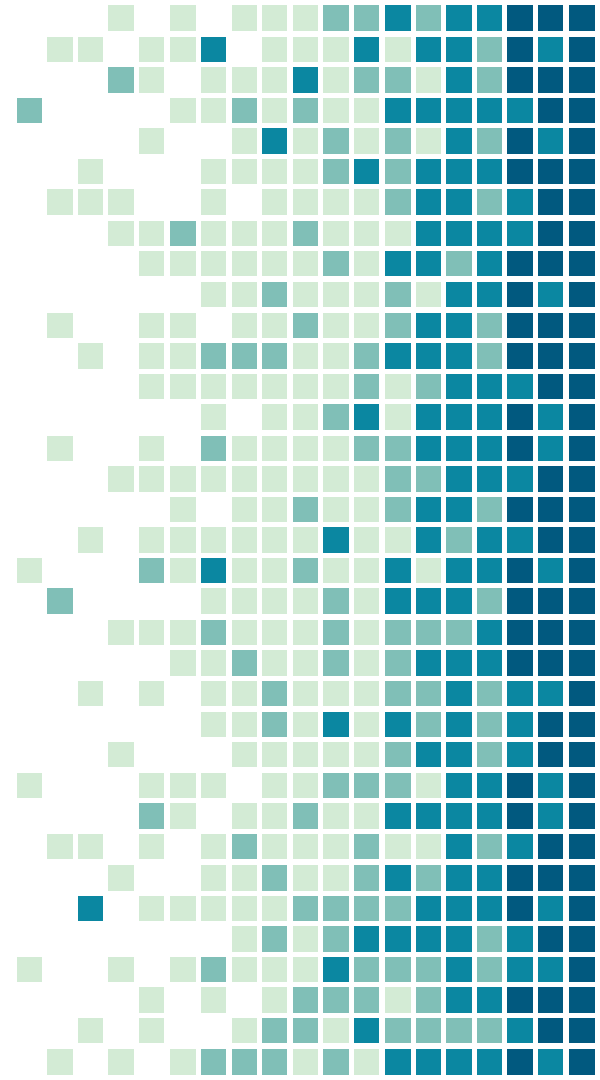


Matriz de Confusión

- Describe el rendimiento de nuestro modelo
- Funciona si existe número igual de ejemplos para cada clase.

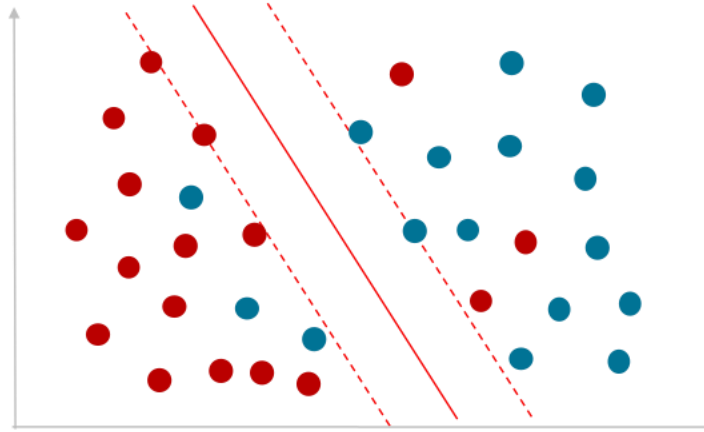
		Predicción	
		Positivos	Negativos
Observación	Positivos	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Negativos	Falsos Positivos (FP)	Verdaderos Negativos (VN)

6. Clasificación



Clasificación

- Decision Tree (Árbol de decisiones)
- RandomForest
- Support Vector Machine
- K-Nearest Neighbour
- Naive Bayes



Clasificación

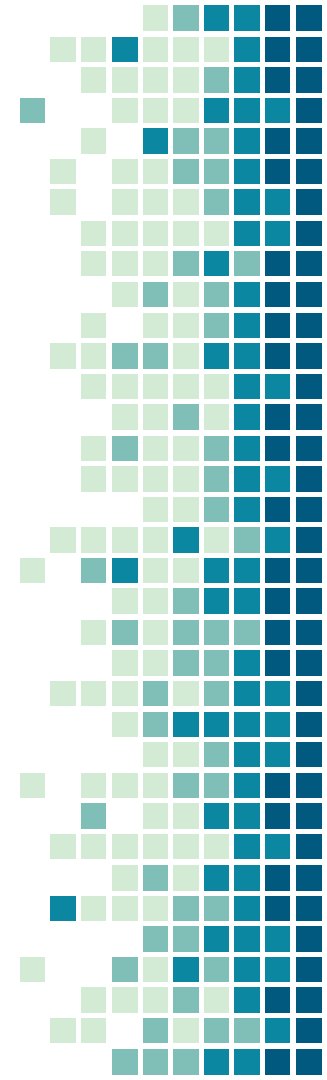
Dada una colección de filas (training set)

Encuentra un modelo que mapee las filas a una clase.

Por ejemplo:

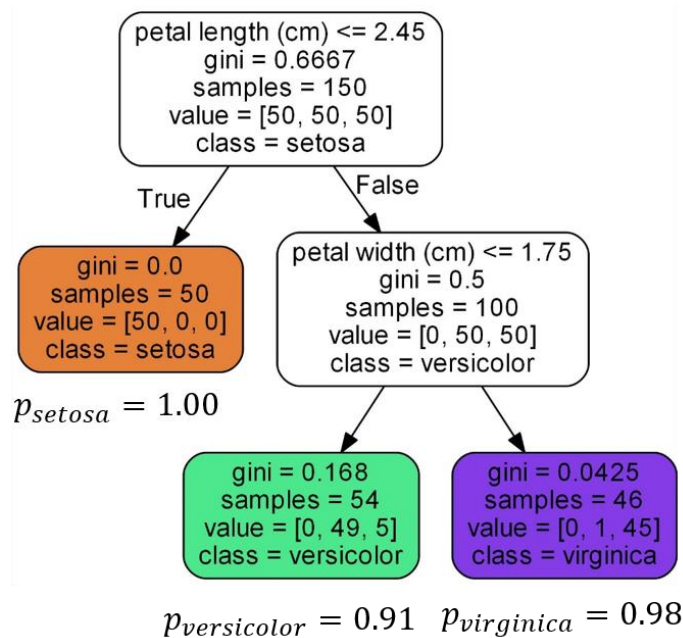
Predecir si un tumor es dañino o benigno.

Predecir si lloverá hoy o no. Predecir si un empleado renunciará o no. Predecir si es gato o no.

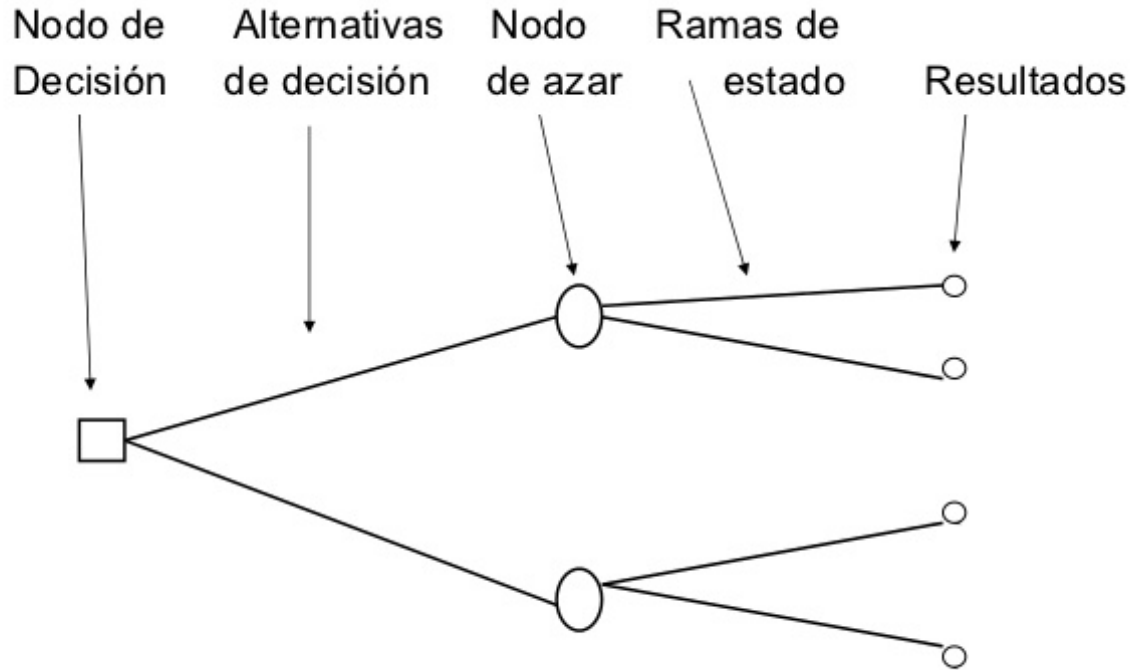


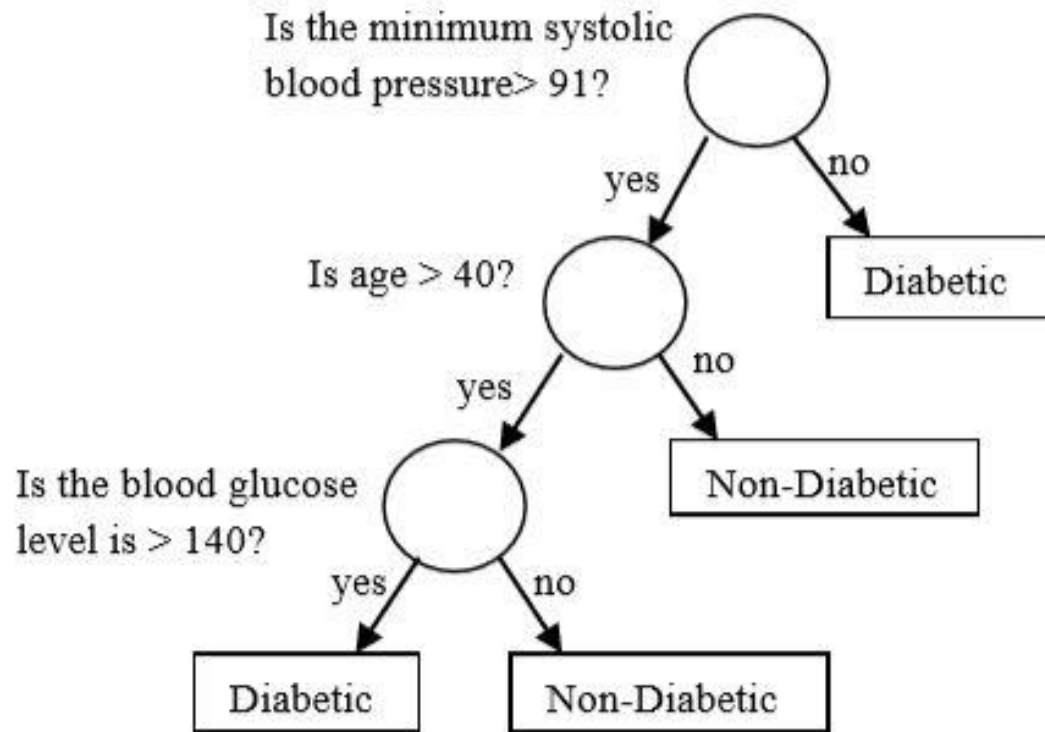
Árbol de Decisiones

- Idea es separar de manera secuencial una base de entrenamiento al preguntar un serie de preguntas

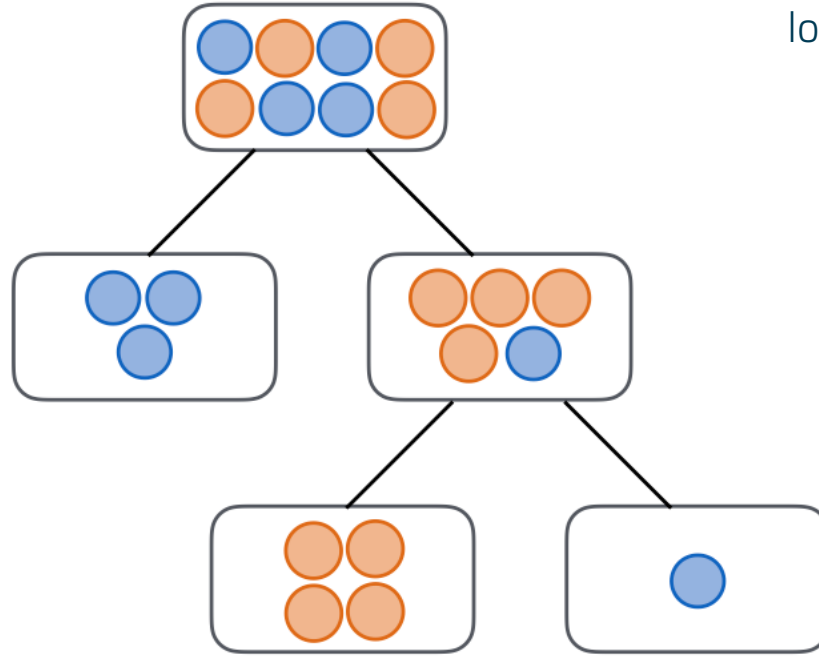


Partes de un árbol





División de Nodos

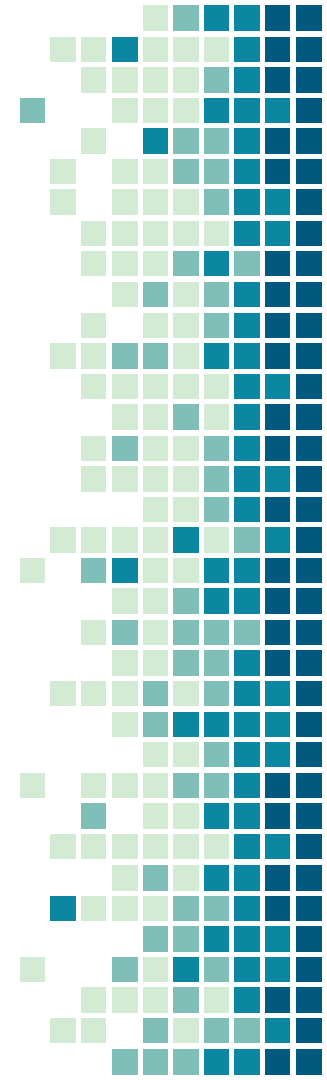


Idea es buscar que
los nodos sean lo
más puros
posibles

GINI
ENTROPY

Árbol de Decisiones

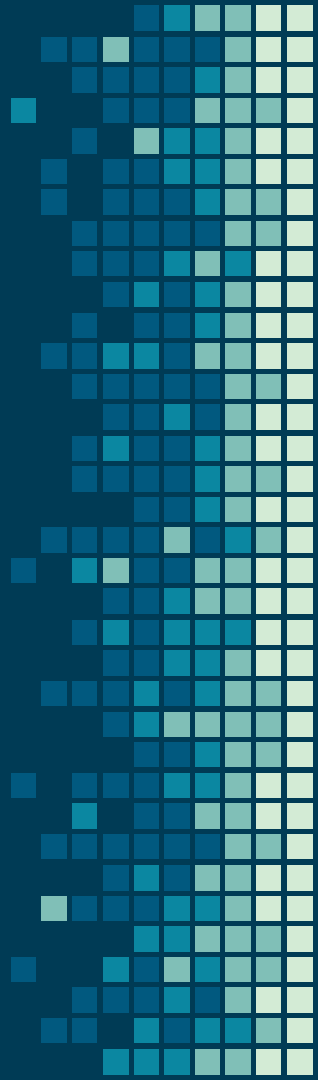
- Ventajas:
 1. Sencillo de entender y visualizar
 2. Pequeña preparación de datos
 3. Maneja datos numéricos y categóricos
- Desventajas:
 1. Crear arboles complejos que no generaliza bien
 2. Inestables (Un cambio en los datos puede generar nuevo árbol)





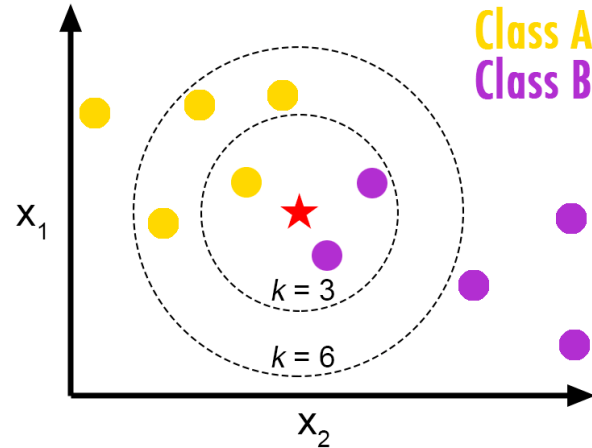
Notebook

Visitar Google Colab

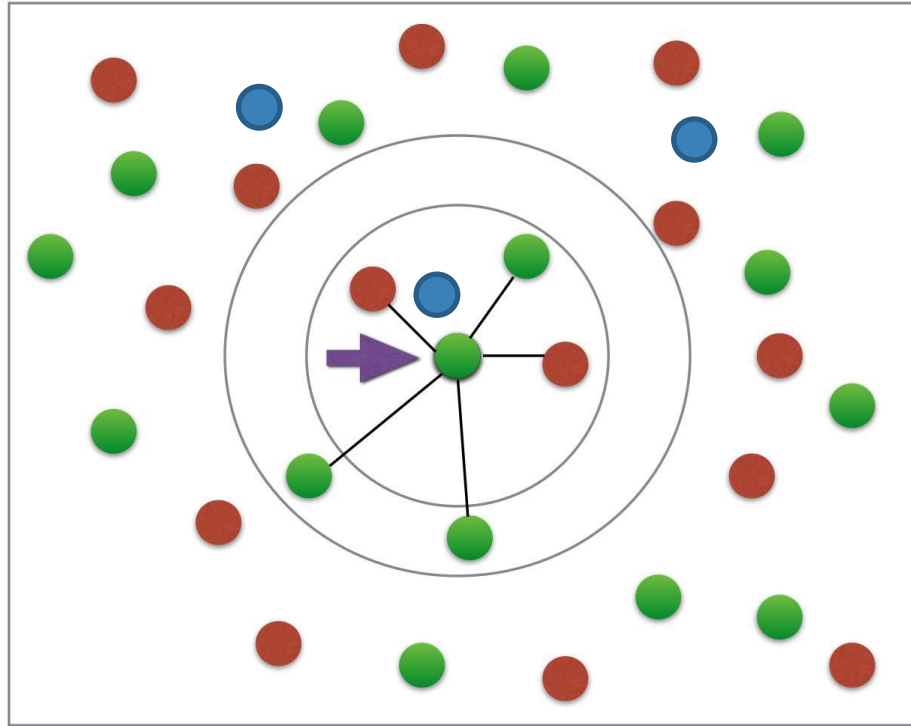


K-Nearest Neighbor

- Un objeto se clasifica por el voto mayoritario de sus vecinos, asignándose el objeto a la clase más común en sus k vecinos más cercanos (k es un numero entero positivo, típicamente pequeño)



K-Nearest Neighbor



Desventajas

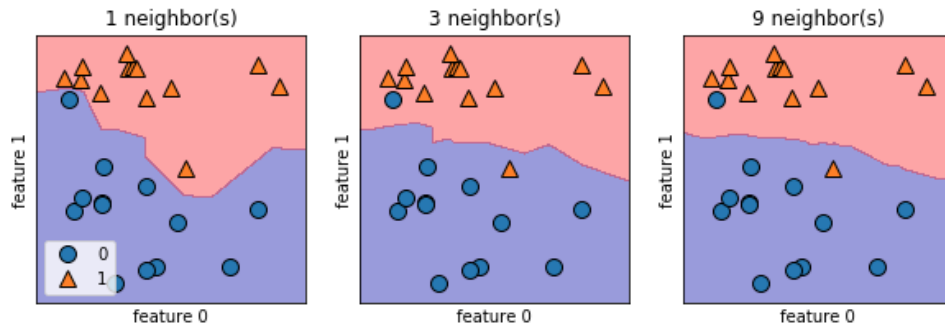
- Sensible a atributos irrelevantes
- Sensible a ruido
- Lento si hay mucha data de entrenamiento
- Necesita determina valor de K



¿Qué elegir como K?

- Largo K = Modelo menos complejo
- Pequeño K = Modelo más complejo

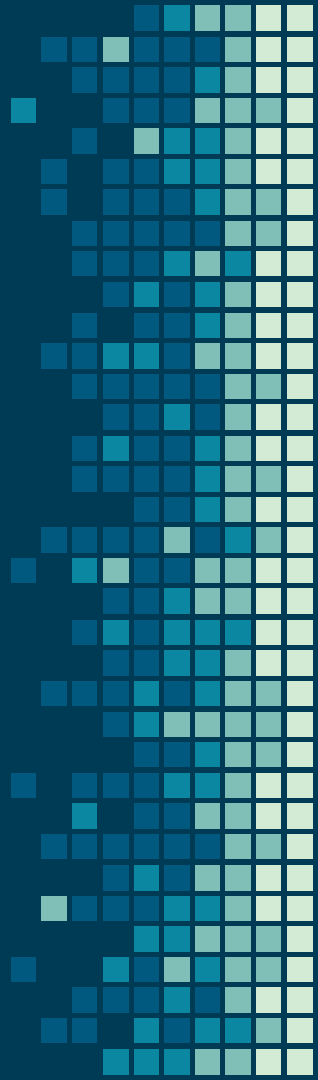
Común: K es impar si son 2 clases e igual a raíz cuadrada del número de clases



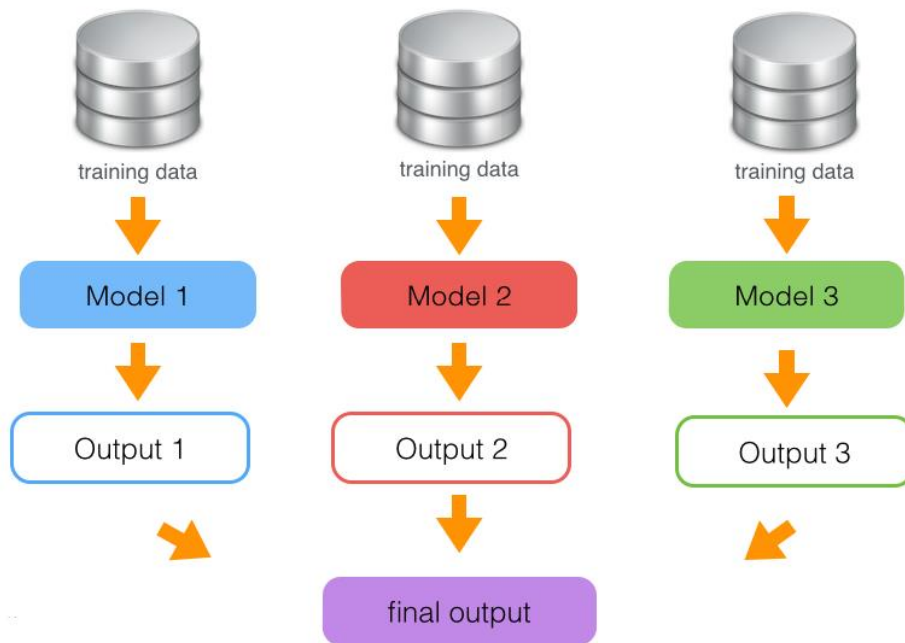


Notebook

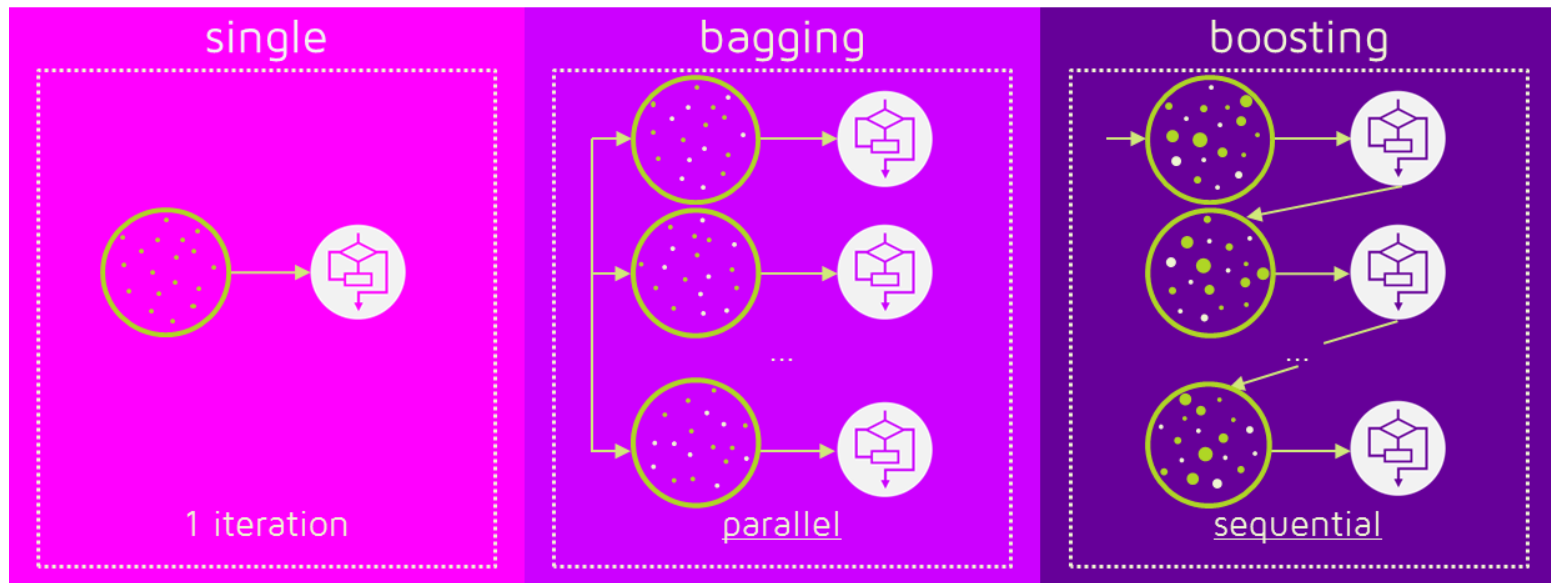
Visitar Google Colab



Ensamble

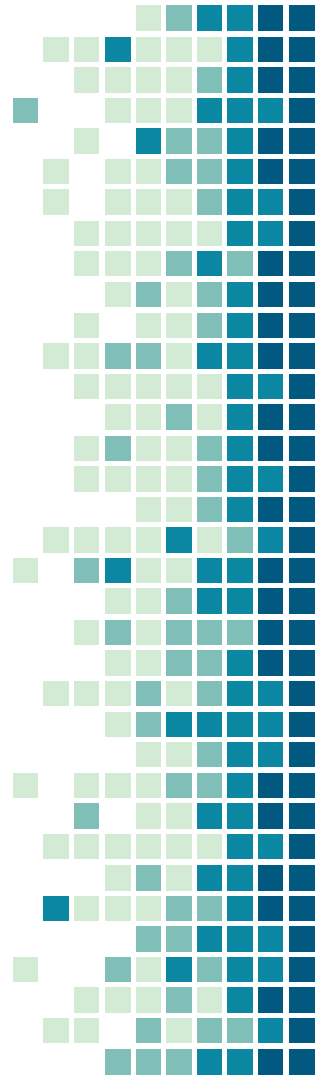


Ensamble

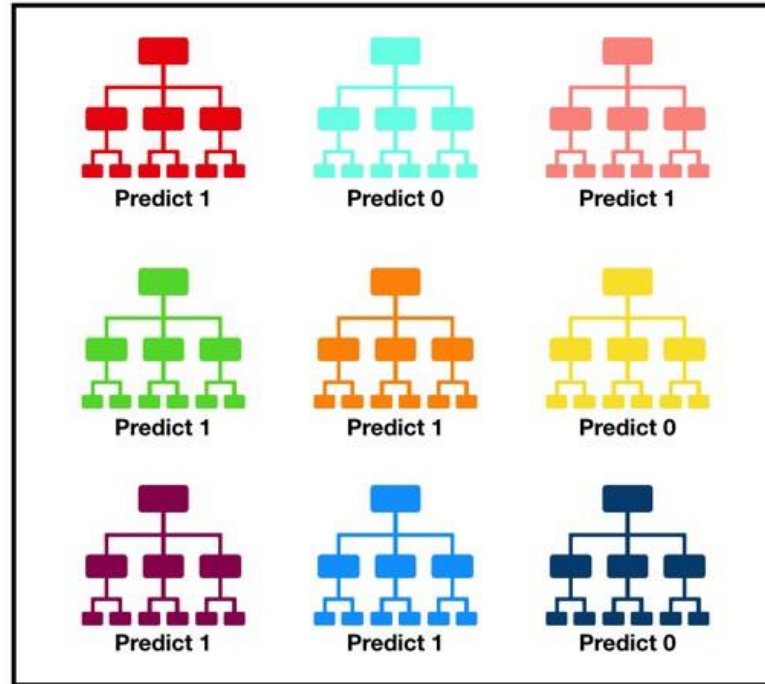


RandomForest

- Método de ensamblado que crea múltiples arboles y clasifica objetos basado en el voto de todos los arboles.
- Pros:
 - Maneja mucho datos con muchas dimensiones
 - Puede manejar datos perdidos
 - Reducción de Overfitting
- Cons:
 - Puede ser caja negra, poco control en modelo



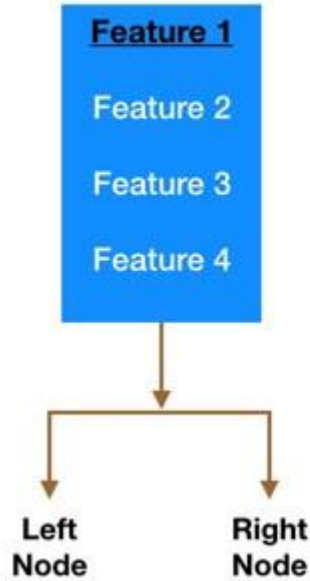
Random Forest



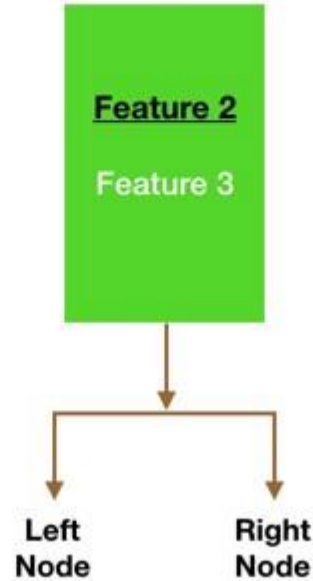
Tally: Six 1s and Three 0s
Prediction: 1

Random Forest

Decision Tree



**Random Forest
Tree 1**



**Random Forest
Tree 2**

