Introducción al Aprendizaje Automático Modelos de Clasificación

MARIA DE LOS ANGELES CONSTANTINO GONZÁLEZ

Tecnológico de Monterrey

Tipos de Aprendizaje

Aprendizaje Supervisado

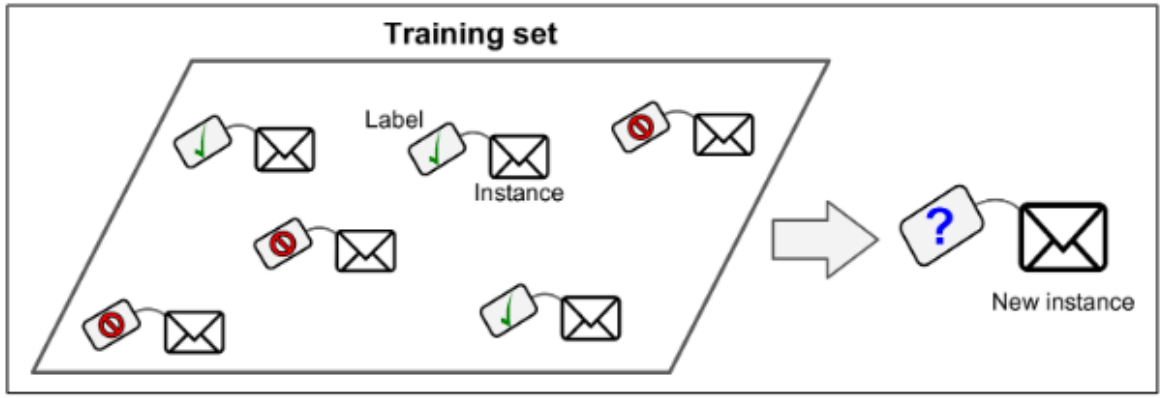


Figure 1-5. A labeled training set for supervised learning (e.g., spam classification)

Figure 1-6. Regression

Aprendizaje No Supervisado

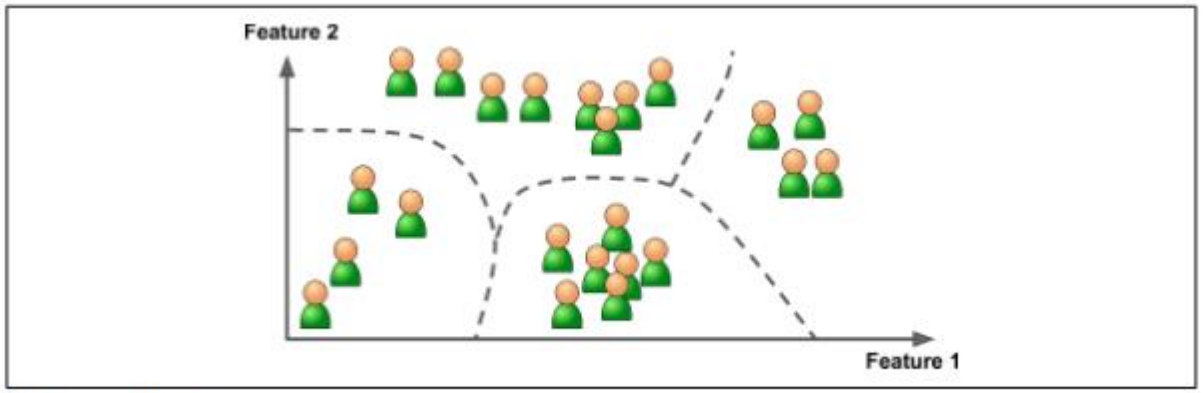
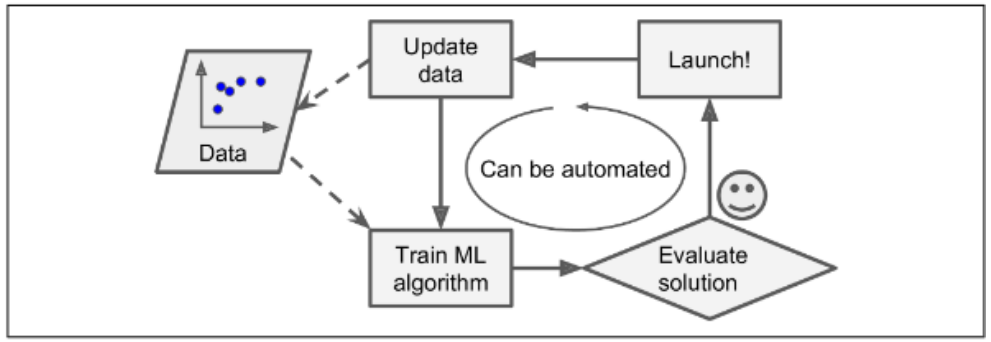


Figure 1-8. Clustering

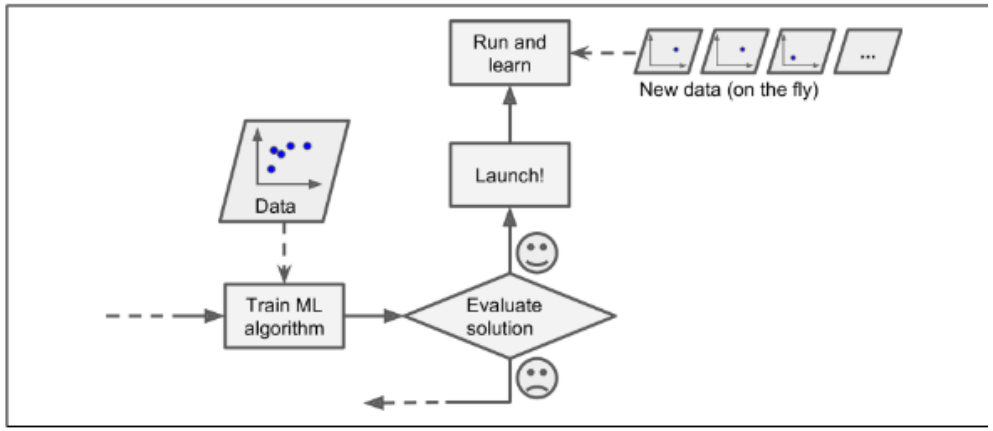
Aprendizaje batch o en línea

Batch (offline learning)



- Se entrena usando todos los datos disponibles.
- Generalmente toma mucho tiempo y recursos
- Una vez que ya es entrenado se utiliza en producción.
- Si se tienen nuevos datos, se debe reentrenar el sistema.
- Generalmente se entrena cada día o semana, mes.

Online - Incremental Learning



- Se entrena incrementalmente, alimentando datos secuencialmente, individualmente o en pequeños grupos.
- El sistema puede aprender sobre nuevos datos al momento que llegan.
- Se utiliza en sistemas que se deben adaptar al cambio rápidamente.

Aprendizaje basado en instancias y basado en modelos

Aprendizaje basado en Instancias (basado en memoria)

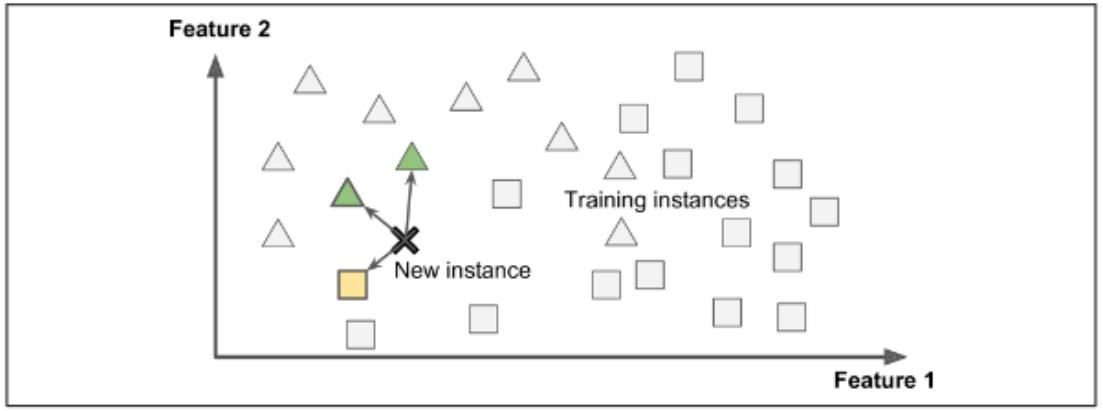


Figure 1-15. Instance-based learning

El sistema aprende los ejemplos de entrenamiento de memoria y generalizan a nuevas instancias en función de alguna medida de similitud.

Aprendizaje basado en modelos

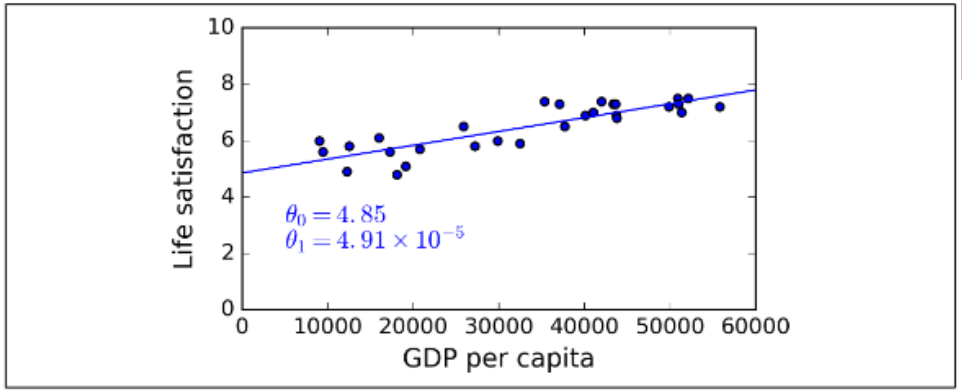
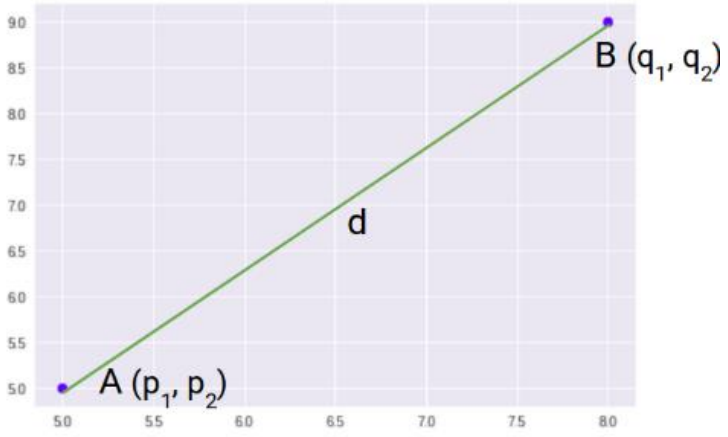


Figure 1-19. The linear model that fits the training data best

Se construye un modelo a partir de los ejemplos y se utiliza ese modelo para hacer predicciones

Medidas de Distancia

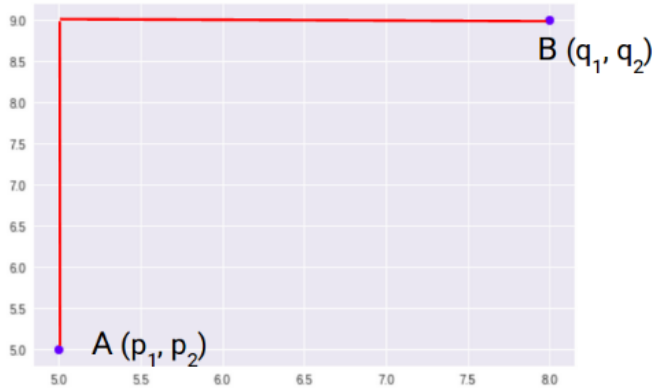
Distancia Euclidiana:
Distancia más corta entre dos puntos



$$d = ((p_1 - q_1)^2 + (p_2 - q_2)^2)^{1/2}$$

$$D_e = \left(\sum_{i=1}^{n} (p_i - q_i)^2 \right)^{1/2}$$

Distancia de Manhattan
Suma de las diferencias absolutas entre los puntos en todas sus dimensiones.



$$d = |p_1 - q_1| + |p_2 - q_2|$$

$$D_m = \sum_{i=1}^{n} |p_i - q_i|$$

Distancia de Minkowski
Forma generalizada de la Euclidiana y la de Manhattan

$$D = \left(\sum_{i=1}^{n} |p_i - q_i|^p \right)^{1/p}$$

p=1: Manhattan
p=2: Euclidiana

https://www.analyticsvidhya.com/blog/2020/02/4-types-of-distance-metrics-in-machine-learning/

Medidas de Distancia

Distancia Hamming

Permite calcular la “distancia” entre dos strings de la misma longitud, comparándolos caracter por caracter.

Ejemplo:

e u c l i d e a n
m a n h a t t a n

Distancia Hamming: 7

Entre más grande es el valor de la distancia haming los strings son más diferentes (y viceversa)

Clasificadores

Un clasificador es un algoritmo que recibe como entrada cierta información de un objeto, e indica la categoría o clase a que pertenece de entre un número acotado de clases posibles.

- Datos: Conjunto de datos (también llamados ejemplos, instancias o casos) descritos por:
  - *k* atributos: $A_1, A_2, \ldots A_k$.
  - Una clase: Cada ejemplo se etiqueta con una clase pre-definida.
- Meta: Aprender un modelo de clasificación a partir de los datos para predecir las clases de nuevos casos/instancias (futuros o pruebas)

Ejemplo: datos de aplicación para el otorgamiento de préstamos

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	No
2	young	false	false	good	No
3	young	true	false	good	Yes
4	young	true	true	fair	Yes
5	young	false	false	fair	No
6	middle	false	false	fair	No
7	middle	false	false	good	No
8	middle	true	true	good	Yes
9	middle	false	true	excellent	Yes
10	middle	false	true	excellent	Yes
11	old	false	true	excellent	Yes
12	old	false	true	good	Yes
13	old	true	false	good	Yes
14	old	true	false	excellent	Yes
15	old	false	false	fair	No

Age	Has_Job	Own_house	Credit-Rating	Class
young	false	false	good	?

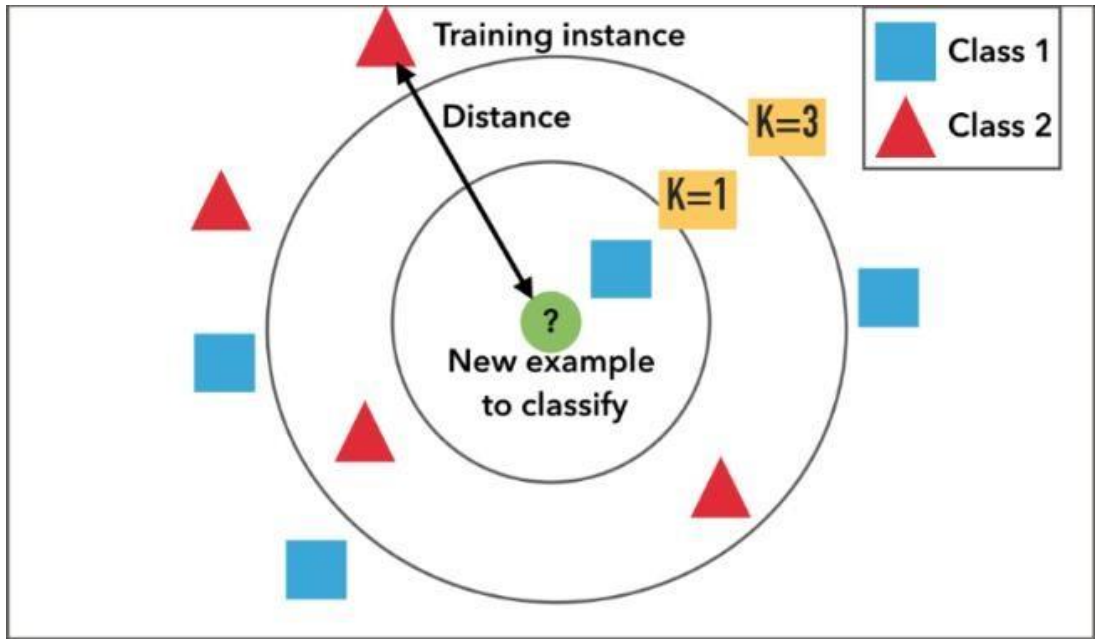
Predicción

Técnicas de Clasificación

- K-Nearest Neighbors (K-NN)
- Gaussian Naive-Bayes
- Máquina de Vectores de Soporte (SVM)
- Regresión Logística
- Árboles de Decisión
- Ensamble de Algoritmos
- Redes Neuronales

Clasificadores k-Nearest Neighbors (k-NN)

La idea de estos métodos es identificar los **k registros,** en el conjunto de datos de entrenamiento, que son **similares al nuevo registro** que se desea clasificar (es decir, registros que tienen valores cercanos a los valores de los predictores x1, x2, ..., XP de este nuevo registro).



En función de las clases a las que pertenecen los k registros más cercanos, al registro que se desea clasificar se le asigna la **clase mayoritaria de los k vecinos**.
Se busca que k sea impar para evitar empates.
Si k = 1, clase = 1
Si k = 3, clase = 2

https://medium.datadriveninvestor.com/knn-algorithm-and-implementation-from-scratch-b9f9b739c28f

Clasificador k-NN

- Obtiene información de similitudes entre los valores predictores (características)
- Para medir la similitud utiliza las **medidas de distancia** como distancia euclidiana, Manhattan, Minkwoski. Para comparar strings se utiliza la distancia de Hamming.
- Para igualar las escalas que pueden tener los distintos predictores, en la mayoría de los casos, los predictores deben **estandarizarse primero** antes de calcular una distancia euclidiana usando escalamiento de datos, tal como escalado estándar o Min-Max
- En el caso de que la **salida sea continua** en lugar de categórica, se calcula el promedio de los k vecinos más cercanos para determinar la predicción.

Clasificador k-NN

Ventaja

Simplicidad. En presencia de un conjunto de entrenamiento lo suficientemente grande, estos métodos funcionan muy bien, especialmente cuando cada clase se caracteriza por múltiples combinaciones de valores predictores.

Ejemplo:

En las bases de datos inmobiliarias, es probable que haya múltiples combinaciones de {tipo de vivienda, número de habitaciones, vecindario, precio de venta, etc.} que caracterizan las casas que se venden rápidamente frente a las que permanecen durante un largo período en el mercado.

Clasificador k-NN

Desventajas

- El tiempo para encontrar a los vecinos más cercanos en un gran conjunto de entrenamiento puede ser prohibitivo. Se puede considerar el reducir dimensiones (características)
- El número de registros requeridos en el conjunto de entrenamiento aumenta exponencialmente con el número de predictores p..
- **Aprendiz perezoso**: Para cada registro que se va a predecir, se calculan distancias desde todo el conjunto de registros de entrenamiento solo en el momento de la predicción. Este comportamiento prohíbe el uso de este algoritmo para la predicción en tiempo real de un gran número de registros simultáneamente.

K-NN Scikit Learn

class `sklearn.neighbors.` **`KNeighborsClassifier`** *(n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None, **kwargs)*

p : *integer, optional (default = 2)*

Power parameter for the Minkowski metric. When p = 1, this is equivalent to using manhattan_-distance (l1), and euclidean_distance (l2) for p = 2. For arbitrary p, minkowski_distance (l_p) is used.

Definir algoritmo
KNeighborsClassifier()

Predecir modelo
predict(x)

Entrenar modelo
fit(x, y)

```
from sklearn.neighbors import KNeighborsClassifier

x_entrenamiento = variablesIndependientes_entrenamiento
y_entrenamiento = variableDependiente_entrenamiento
x_prueba = variablesIndependientes_prueba
y_prueba = variableDependiente_prueba

algoritmo = KNeighborsClassifier()
algoritmo.fit(x_entrenamiento, y_entrenamiento)
algoritmo.predict(x_prueba)
```

https://aprendeia.com/k-vecinos-mas-cercanos-con-scikit-learn-machine-learning/

Clasificadores Naive Bayes

- Se basan en una técnica de clasificación estadística llamada **"teorema de Bayes",** proporcionando una forma de calcular la probabilidad 'posterior' de que ocurra un cierto evento *A,* dadas algunas probabilidades de eventos 'anteriores'.
- Se asume que las variables predictoras son independientes entre sí, es decir que el efecto de una característica particular en una clase es independiente de otras características (ingenuo)
- Es fácil de construir y particularmente útil para conjuntos de datos muy grandes.

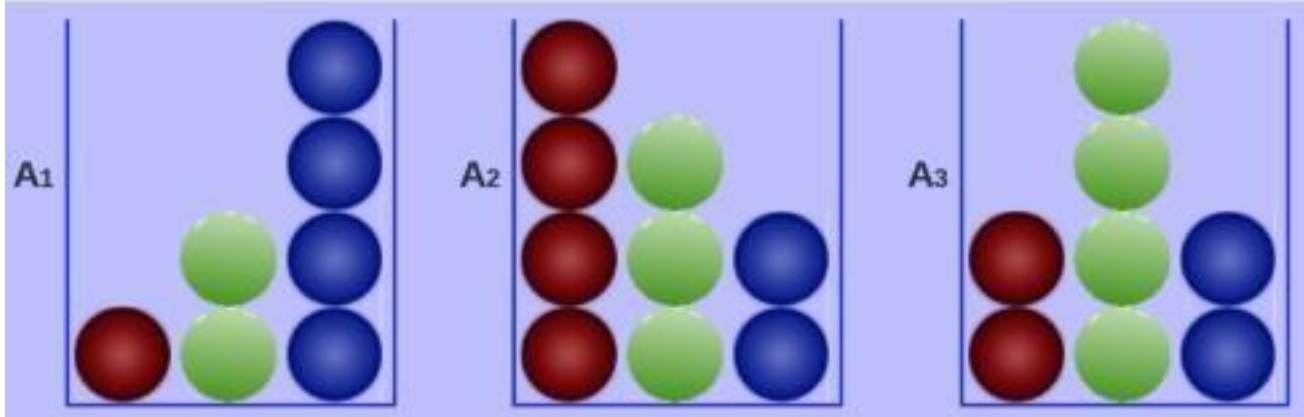
$$P(h \mid D) = \frac{P(D \mid h)P(h)}{P(D)}$$

P(h): es la probabilidad de que la hipótesis h sea cierta (independientemente de los datos).
P(D): probabilidad de los datos (independientemente de la hipótesis).
P(h|D): es la probabilidad de la hipótesis h dada los datos D. (probabilidad posterior)
P(D|h): es la probabilidad de los datos d dado que la hipótesis h era cierta.

Clasificador Naive – Bayes
Teorema de Bayes

La urna A_1 contiene: 1 bola roja, 2 verdes y 4 azules, la A_2: 4 bolas rojas, 3 verdes y 2 azules y la urna A_3 contiene: 2 bolas rojas, 4 verdes y 2 azules. Elegida una urna al azar se extrae una bola qe resulta ser **verde.**

¿Cuál es la probabilidad de que la bola extraida proceda de la urna A_3?



A priori $P(A_1) = P(A_2) = P(A_3) = \frac{1}{3} = 0{,}333$

$P(r/A_1) = \frac{1}{7}$ $P(v/A_1) = \frac{2}{7}$ $P(a/A_1) = \frac{4}{7}$ $P(r/A_2) = \frac{4}{9}$ $P(v/A_2) = \frac{3}{9}$ $P(a/A_2) = \frac{2}{9}$

$P(r/A_3) = \frac{2}{8}$ $P(v/A_3) = \frac{4}{8}$ $P(a/A_3) = \frac{2}{8}$ $\Rightarrow P(v) = \frac{1}{3}\frac{2}{7} + \frac{1}{3}\frac{3}{9} + \frac{1}{3}\frac{4}{8} = 0.373$

Aplicando el teorema de **Bayes** tenemos que:

$$P(A_3/v) = \frac{P(A_3)\cdot P(v/A_3)}{P(v)} = \frac{\frac{1}{3}\frac{4}{8}}{0.373} = 0.447$$

https://proyectodescartes.org/descartescms/blog/difusion/item/3270-miscelaneas-probabilidad-a-posteriori-teorema-de-bayes

Clasificador Naive-Bayes Algoritmo

1. Convertir el conjunto de datos en una tabla de frecuencias.
2. Calcular la probabilidad a priori para cada una de las etiquetas de clase.
3. Determinar la probabilidad de cada atributo con respecto a cada clase.
4. Calcular la probabilidad posterior de cada clase a partir de los atributos dados.
5. La clase con la probabilidad posterior más alta es el resultado de la predicción.

Clasificador Naive-Bayes

Ventajas: Simplicidad, eficiencia computacional, buen rendimiento de clasificación y capacidad para manejar variables categóricas directamente.

Es particularmente útil cuando se tiene un **gran número de registros.**

En caso de predictores numéricos se recomienda:

1) Discretizar (binned) los predictores numéricos y convertirlos a predictores categóricos.

2) Asumir que las variables tienen distribución gaussiana.

Naive Bayes Scikit Learn

```
class sklearn.naive_bayes. GaussianNB (priors=None, var_smoothing=1e-09)
```


```
from sklearn.naive_bayes import GaussianNB

x_entrenamiento = variablesIndependientes_entrenamiento
y_entrenamiento = variableDependiente_entrenamiento
x_prueba = variablesIndependientes_prueba
y_prueba = variableDependiente_prueba

algoritmo = GaussianNB()
algoritmo.fit(x_entrenamiento, y_entrenamiento)
algoritmo.predict(x_prueba)
```

https://aprendeia.com/k-vecinos-mas-cercanos-con-scikit-learn-machine-learning/

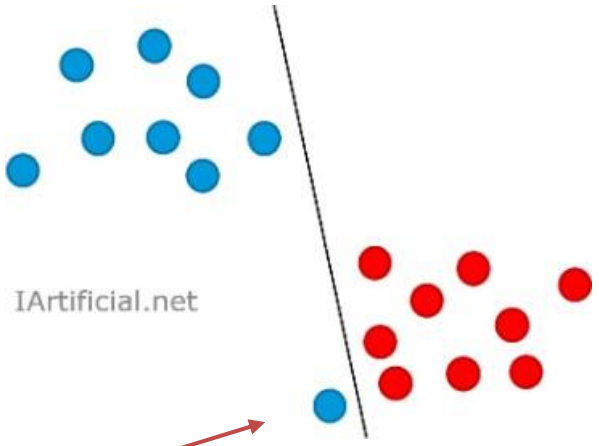
SVM: Máquinas de vectores de soporte (Support vector machines)

Este tipo de clasificadores asumen que se pueden establecer fronteras lineales en el espacio total de valores posibles de forma que, dependiendo de dónde se encuentre un conjunto de datos concreto, estará a alguno de los lados de la frontera, lo que lleva a su clasificación según el lado en que se encuentre.

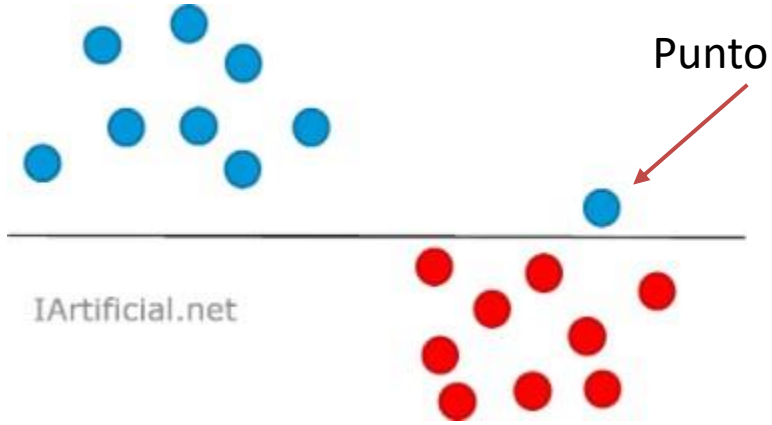
Construye un hiperplano en un espacio multidimensional para separar las clases

SVM

Formas equivocadas de clasificar a un punto nuevo

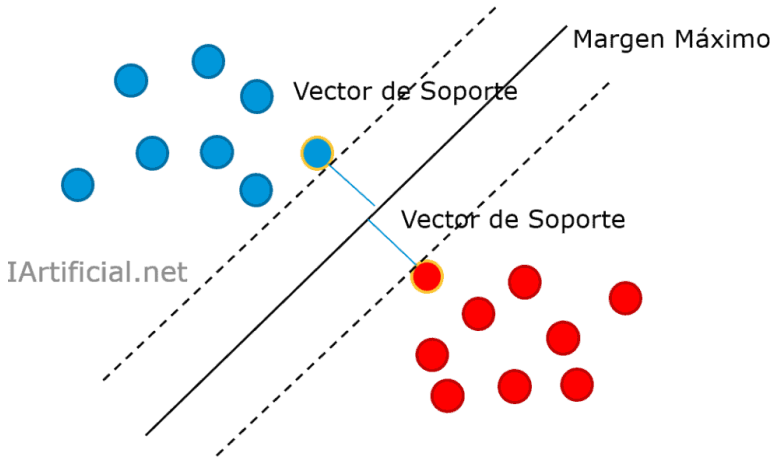


Punto nuevo



Vectores de Soporte: son los puntos de datos más cercanos al hiperplano

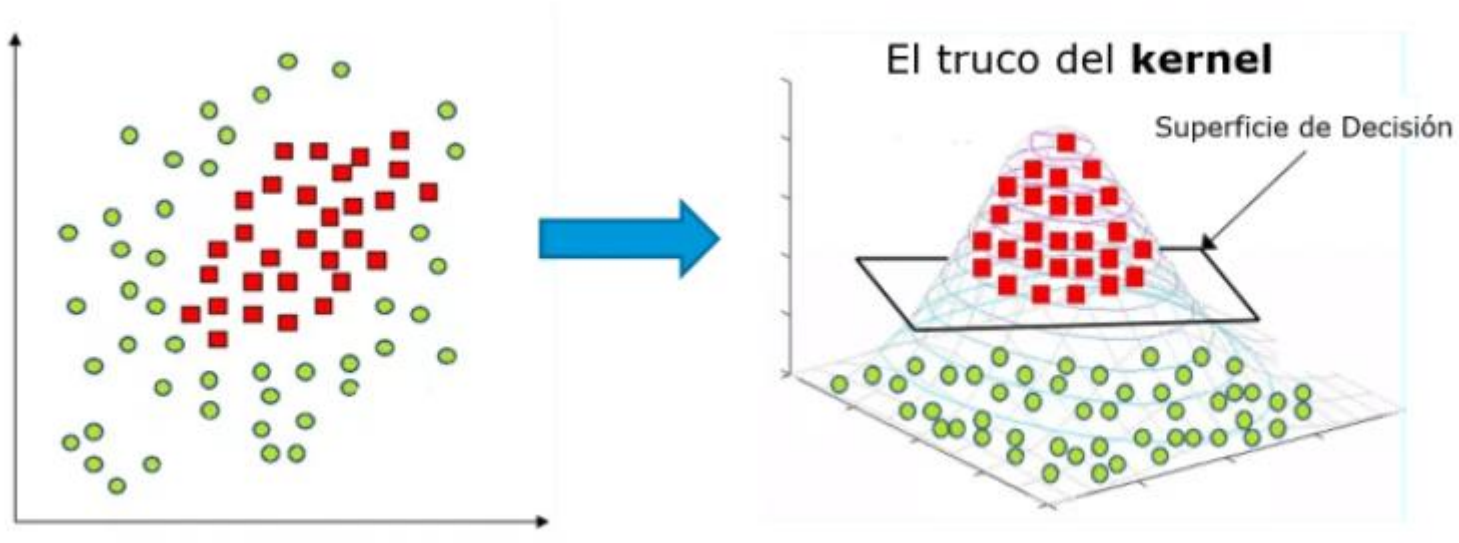
SVM: Encuentra la mejor separación posible entre clases. La clasificación óptima se realiza maximizando el margen de separación entre las clases



El SVM encuentra el hiperplano que maximiza el margen de separación entre clases.

SVM: Truco de núcleo (Kernel)

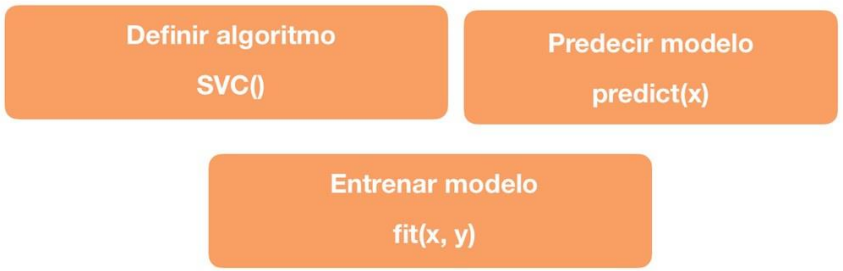
El truco del kernel consiste en inventar una dimensión nueva en la que podamos encontrar un hiperplano para separar las clases. En la siguiente figura vemos cómo al añadir una dimensión nueva, podemos separar fácilmente las dos clases con una superficie de decisión.



SVM
SVC: Clasificación, SVR: Regresión

```
class sklearn.svm. SVC (C=1.0, kernel='rbf', degree=3, gamma='auto_deprecated', coef0=0.0, shrinking=True,
probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1,
decision_function_shape='ovr', random_state=None)                                                    [s
```

C: parámetro de regularización que representa una clasificación errónea o un término de error. Indica cuánto error es soportable. Un valor menor de C crea un hiperplano de pequeño margen y un valor de mayor de C crea un hiperplano de mayor margen.
Kernel: Función a utilizar: lineal (linear), polinómica (poly), radial (RBF).




```python
from sklearn.svm import SVC

x_entrenamiento = variablesIndependientes_entrenamiento
y_entrenamiento = variableDependiente_entrenamiento
x_prueba = variablesIndependientes_prueba
y_prueba = variableDependiente_prueba

algoritmo = SVC()
algoritmo.fit(x_entrenamiento, y_entrenamiento)
algoritmo.predict(x_prueba)
```

https://aprendeia.com/k-vecinos-mas-cercanos-con-scikit-learn-machine-learning/

Regresión Logística

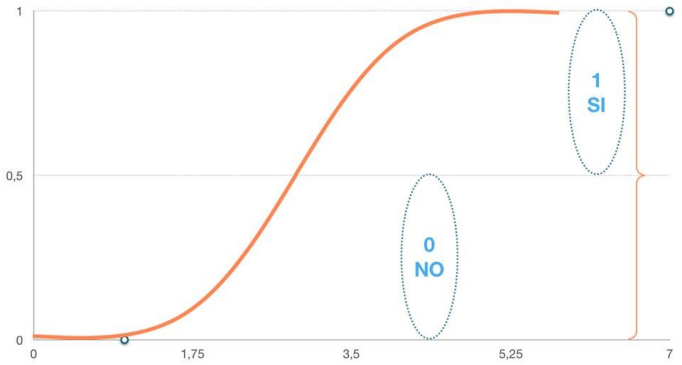
Algoritmo de clasificación basado en Estadística.

Diferencia entre Regresión Lineal y Regresión Logística:

- Regresión Logística: Variable de salida es binomial (clasificación binaria)
- Regresión Lineal: Variable de salida es numérica

A función logística también se le llama sigmoide.

$$f(y) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$



Por analogía, la regresión logística puede considerarse una extensión de los modelos de regresión lineal, con la particularidad de que el dominio de salida de la función está acotado al intervalo [0,1] y que el procedimiento de estimación, en lugar de mínimos cuadrados, utiliza el procedimiento de ***estimación máximo-verosímil***.

Regresión Logística

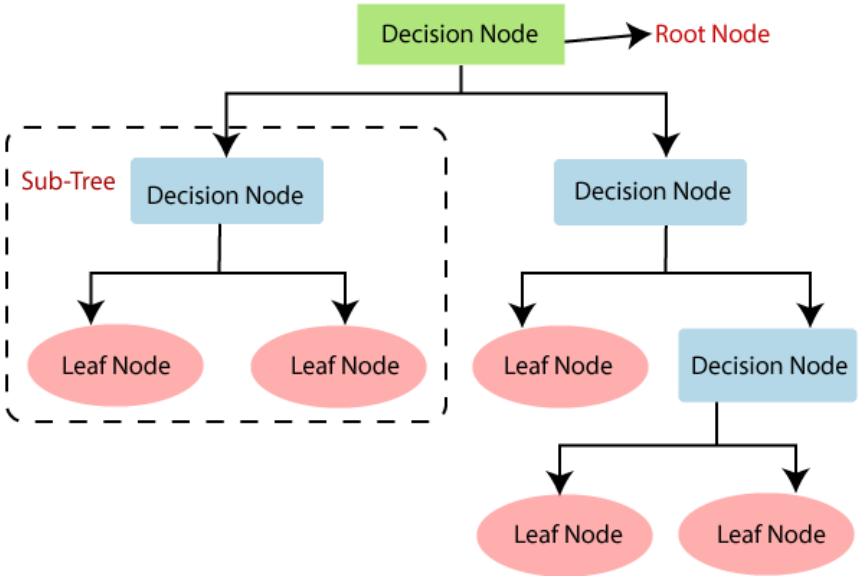
```
class sklearn.linear_model.LogisticRegression(penalty='l2', dual=False, tol=0.0001, C=1.0,
fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='liblinear',
max_iter=100, multi_class='ovr', verbose=0) ¶
```

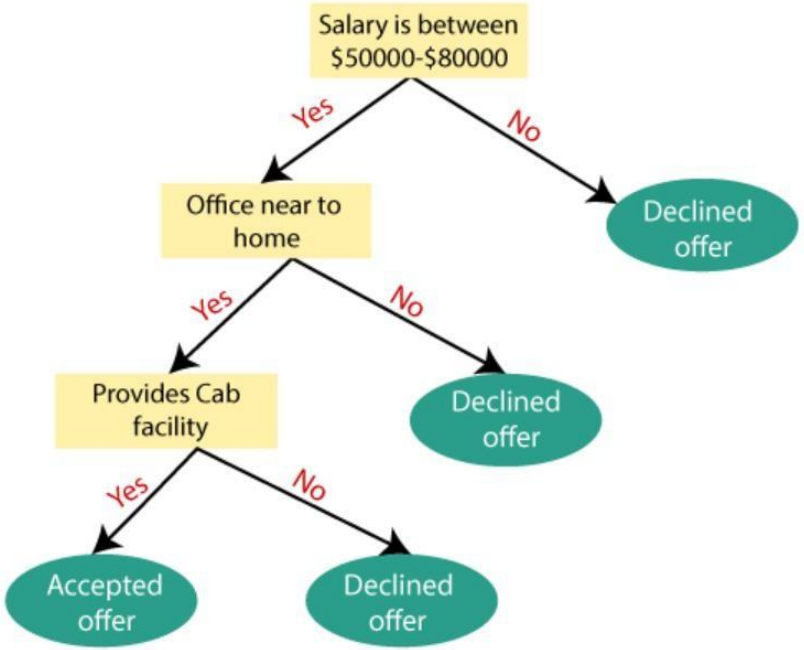
C: parámetro de regularización que representa una clasificación errónea o un término de error. Indica cuánto error es soportable.

```
from sklearn.linear_model import LogisticRegression

x_entrenamiento = variablesIndependientes_entrenamiento
y_entrenamiento = variableDependiente_entrenamiento
x_prueba = variablesIndependientes_prueba
y_prueba = variableDependiente_prueba

algoritmo = LogisticRegression()

algoritmo.fit(x_entrenamiento, y_entrenamiento)
algoritmo.predict(x_prueba)
```

Árboles de Decisión

- Emplea el método divide y vencerás
- Divide recursivamente un conjunto de entrenamiento hasta que la división consista de ejemplos de una clase.





A candidate who has a job offer and wants to decide whether he should accept the offer or No.

https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm

Árboles de Decisión
Algoritmo

1. Crear un nodo raiz y asignarle todos los datos de entrenamiento.
2. Seleccionar el mejor atributo para dividir los datos.
3. Agregar una rama al nodo raiz para cada valor dividido. Divide los datos mutuamente, subconjuntos exclusivos, a lo largo de la línea.
4. Repetir pasos 2 y 3 para cada hoja nodo hasta que se alcance un criterio de paro.

Árboles de Decisión

- Los algoritmos de árboles se diferencían en
  1. Criterios de partición
     - Qué variable, qué valor
  2. Criterios de paro
     - Cuando parar de construir el árbol.
  3. Poda (Método de generalización)
     - Pre-poda versus post-poda

- Los algoritmos de árboles de decision más populares incluyen:
  - ID3, C4.5, C5; CART; CHAID; M5

Árboles de Decisión

Ventajas:

Fácil de entender. La salida del árbol de decisión es muy fácil de entender, incluso para personas con antecedentes no analíticos, no se requiere ningún conocimiento estadístico para leerlos e interpretarlos.

Se requiere menos limpieza de datos. Requiere menos limpieza de datos en comparación con algunas otras técnicas de modelado. A su vez, no esta influenciado por los valores atípicos y faltantes en la data.

El tipo de datos no es una restricción. Puede manejar variables numéricas y categóricas.

https://aprendeia.com/aprendizaje-supervisado-decision-tree-classification/

Árboles de Decisión

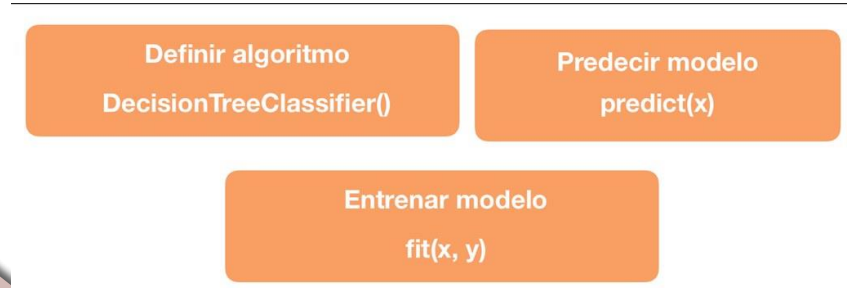
class `sklearn.tree.` **`DecisionTreeClassifier`** *(criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None,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

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.

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.




```
from sklearn.tree import DecisionTreeClassifier

x_entrenamiento = variablesIndependientes_entrenamiento
y_entrenamiento = variableDependiente_entrenamiento
x_prueba = variablesIndependientes_prueba
y_prueba = variableDependiente_prueba

algoritmo = DecisionTreeClassifier()
algoritmo.fit(x_entrenamiento, y_entrenamiento)
algoritmo.predict(x_prueba)
```

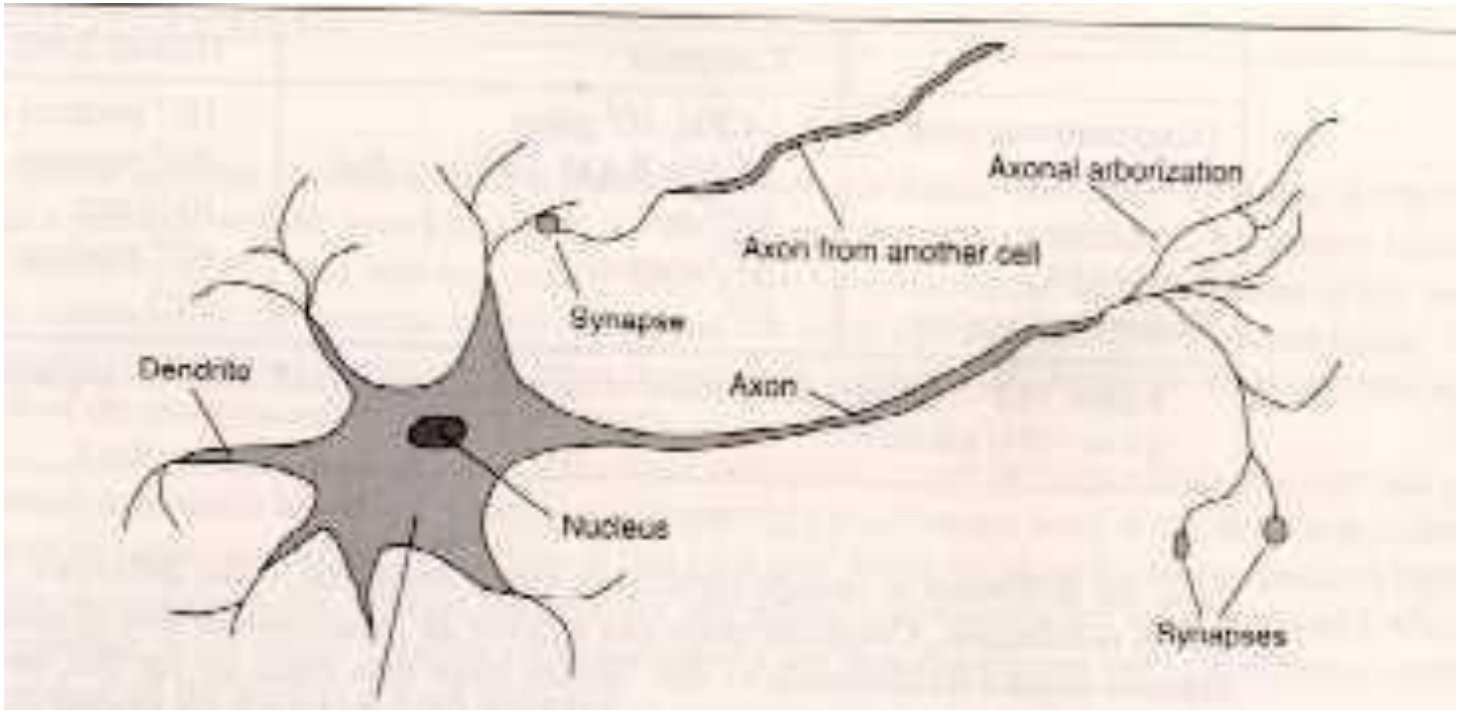
https://aprendeia.com/arboles-de-decision-clasificacion-scikit-learn-machine-learning/

Redes Neuronales

- Las Redes Neuronales son una representación del conocimiento totalmente diferente de las representaciones simbólicas.
- Están basadas en lo que se conoce sobre la estructura del cerebro humano.
- A esta representación del conocimiento se le conoce también como modelos conexionistas o sistemas de procesamiento distribuido paralelo.

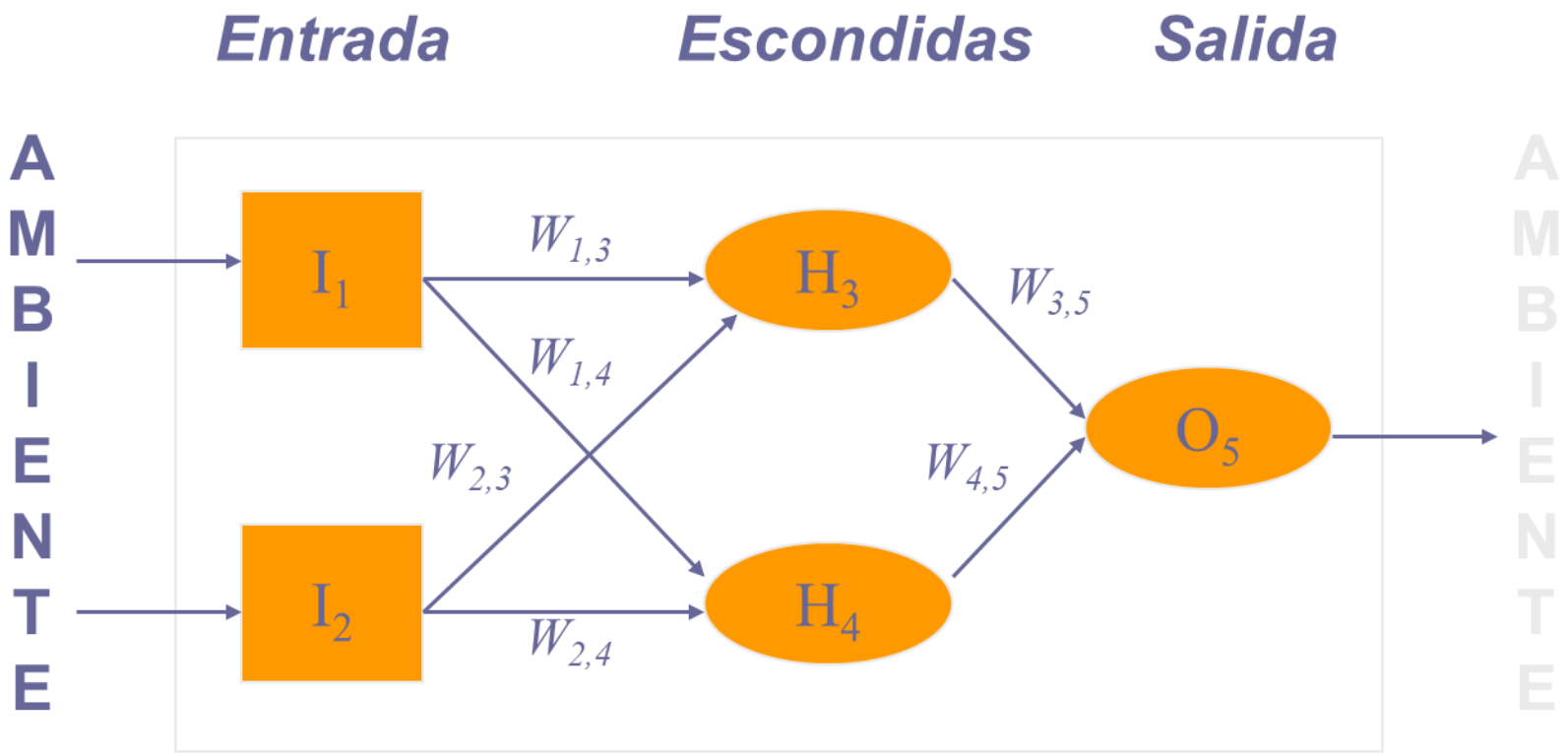
Redes Neuronales: Base teórica

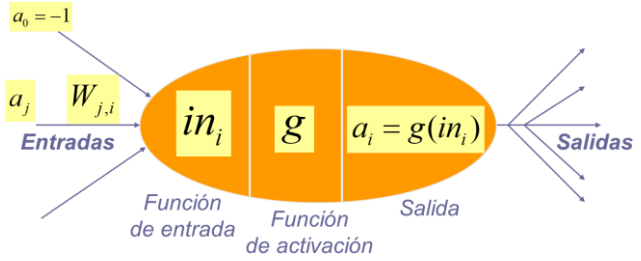
- Partes de una neurona o célula nerviosa



- Longitud del axon: 1 cm (hasta 1 m. en ciertos casos)
- Número de sinapsis: de una docena a cientos de miles

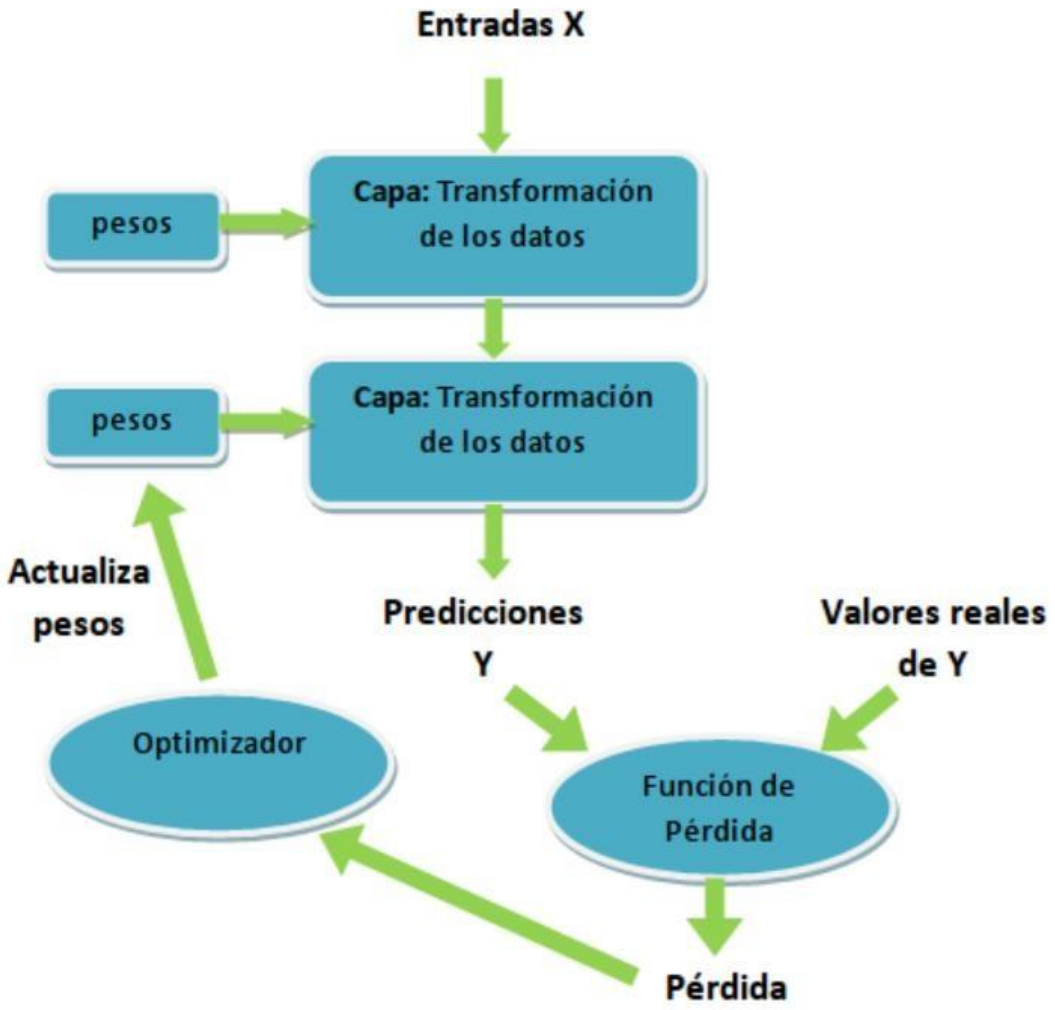
Redes Neuronales: Aprendizaje





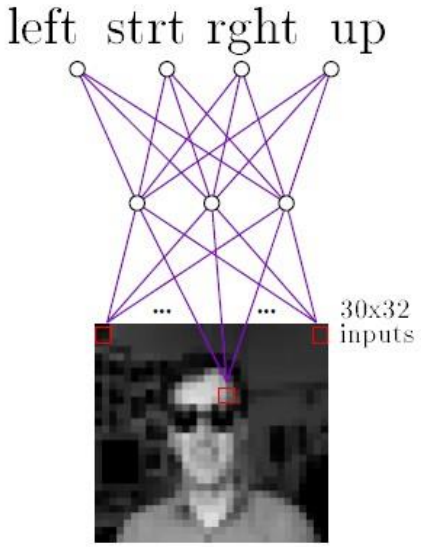
Aprendizaje = "puesta a tono" de los pesos para ajustarse al conjunto de entrenamiento

Algoritmo de Aprendizaje
Ajuste de pesos



https://relopezbriega.github.io/blog/2017/06/13/introduccion-al-deep-learning/

Ejemplo: Redes Neuronales para reconocimiento de imágenes



Typical input images

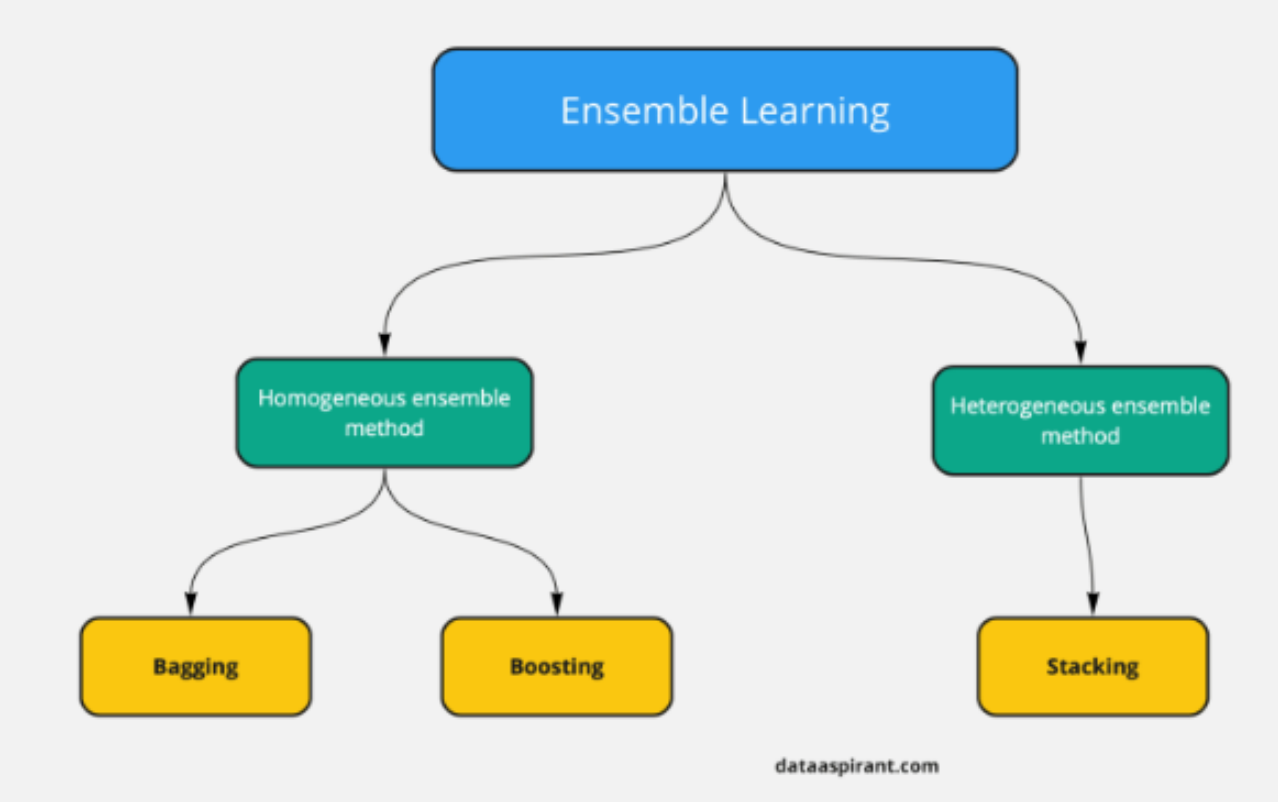
90% accurate learning head pose, and recognizing 1-of-20 faces

Ensamblado de Modelos para Analítica Predictiva

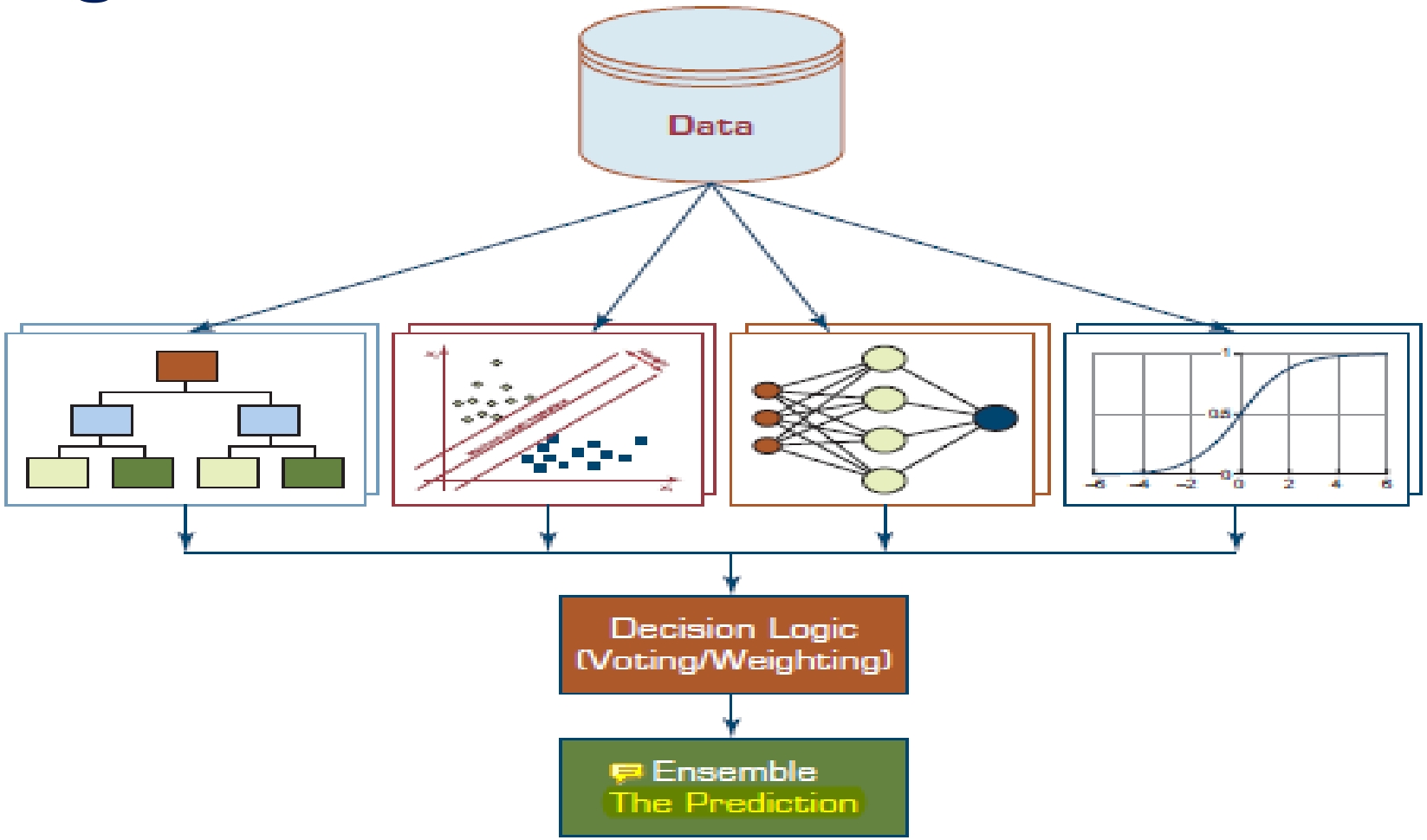
- Usar multiples algoritmos de aprendizaje para obtener un desempeño mejor que el que produce un solo algoritmo.
- Producen modelos de predicción más robustos y confiables.
- Tipos:

  Homogéneos: usan el mismo método de selección.

  Heterogéneos: usan diferentes métodos de selección sobre la misma base de datos.



Ensamble Heterogéneo



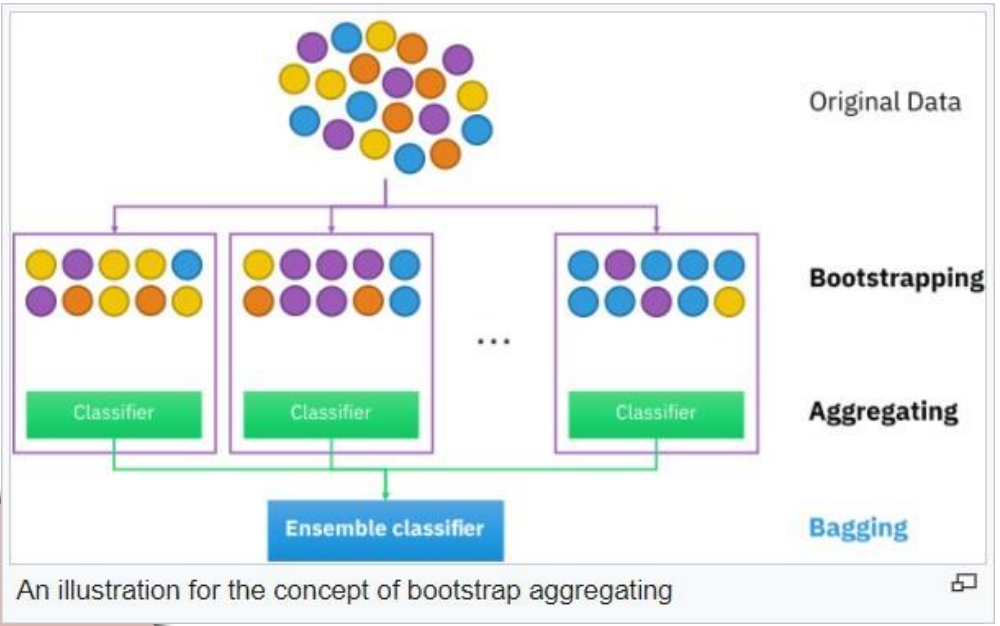
Ensamble Homogéneo de Modelos

Tipos:

1) BAGGing (Boostrap Aggregation)

Dada una muestra de datos, se extraen varias muestras, bootstrapped. Esta selección se realiza de manera aleatoria. Se entrenan los modelos de manera separada en forma paralela.

La decisión final se toma en base a la mayoría.



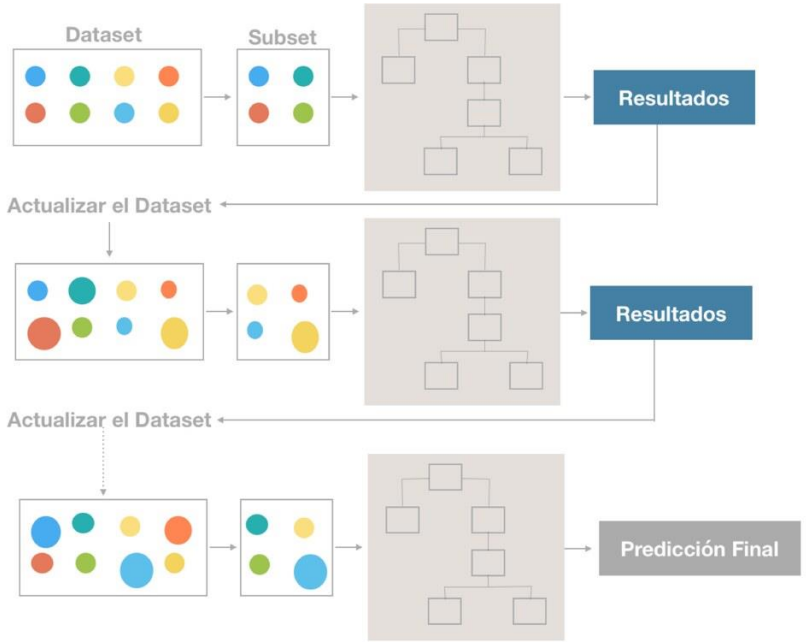
An illustration for the concept of bootstrap aggregating

2) Boosting

Aprendizaje secuencial.

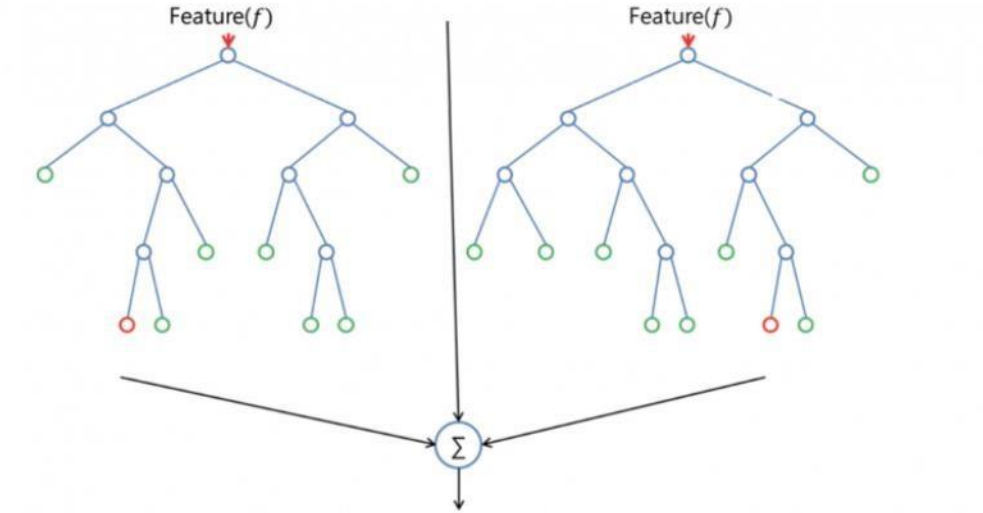
Se entrena un modelo con todo el conjunto de entrenamiento y los modelos posteriores se construyen ajustando valores de error residual.

La decisión final se obiene mediante promedio ponderado



Ensamble Homogéneo
Random Forest

- Es un tipo de método BAGGing.
- Es un ensamble de varios árboles de decision entrenados con el método bagging (boostrap aggregating), cuya idea general es que la combinación de modelos de aprendizaje mejore el resultado final.



Random Forest (Bosques Aleatorios)

```
class sklearn.ensemble.RandomForestClassifier(n_estimators=100, *, criterion='gini', max_depth=None,
min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None,
min_impurity_decrease=0.0, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0,
warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None) ¶                                    [s
```

n_estimators : *int, default=100*

The number of trees in the forest.

Definir algoritmo
RandomForestClassifier()

Predecir modelo
predict(x)

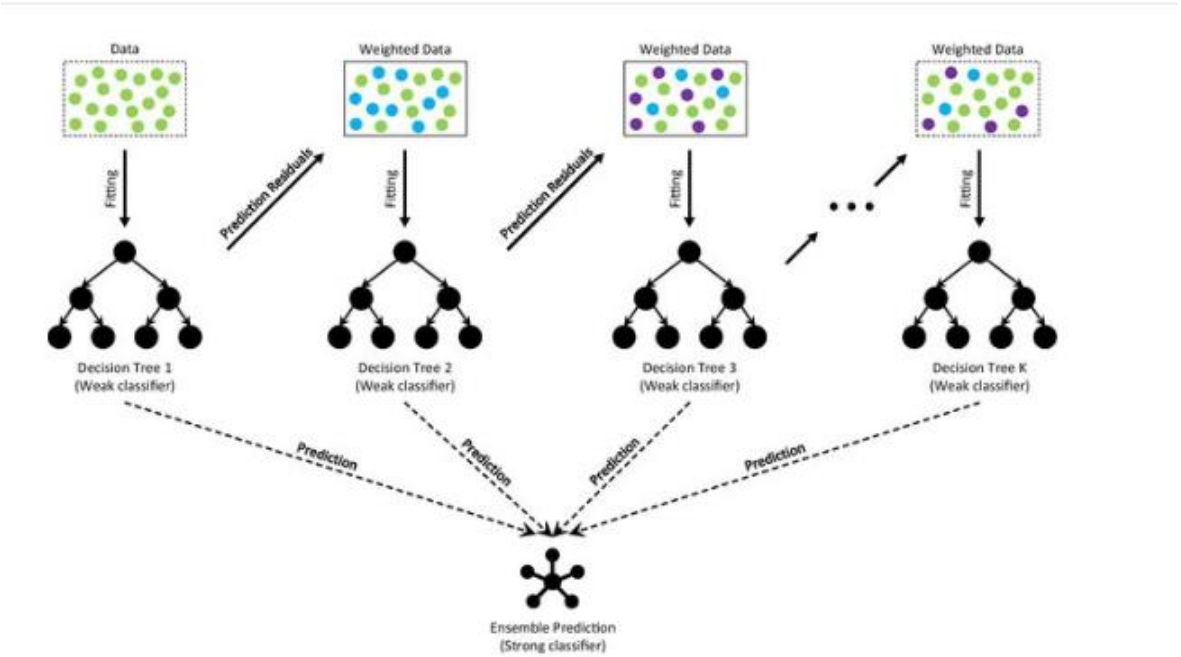
Entrenar modelo
fit(x, y)

```
from sklearn.ensemble import RandomForestClassifier

x_entrenamiento = variablesIndependientes_entrenamiento
y_entrenamiento = variableDependiente_entrenamiento
x_prueba = variablesIndependientes_prueba
y_prueba = variableDependiente_prueba

algoritmo = RandomForestClassifier()
algoritmo.fit(x_entrenamiento, y_entrenamiento)
algoritmo.predict(x_prueba)
```

https://aprendeia.com/

Ensamble Homogéneo
Gradient Boosting Classifier

- Es un tipo de método Boosting
- Combina muchos modelos de aprendizaje débiles para crear un modelo predictivo más fuerte, al mismo tiempo de tratar de reducir los residuos.



Ejemplo

Seguir tutorial en https://towardsdatascience.com/machine-learning-workflow-on-diabetes-data-part-01-573864fcc6b8

Dataset: Pima Indians Diabetes Database | Kaggle

Objetivo: predecir si una persona tiene o no diabetes basado en ciertas medidas diagnósticas:

pregnancies - Number of times pregnant.

glucose - Plasma glucose concentration a 2 hours in an oral glucose tolerance test

diastolic - Diastolic blood pressure (mm Hg).

Skill thickness- Skinfold thickness (mm). subcutaneous body fat

insulin – 2 Hour serum insulin (mu U/ml).

bmi – Body mass index (weight in kg/(height in m) ^2).

dpf - Diabetes pedigree function (likelihood of diabetes based on family history)

age - Age in years.

diabetes - “1” represents the presence of diabetes while “0” represents the absence of it. This is the target variable.

Referencias

Business Intelligence, Analytics, and Data Science: A Managerial Perspective, Ramesh Sharda, Dursun Delen, Efraim Turban, Pearson, 2018

Data mining for business analytics : concepts, techniques and applications in Python / Galit Shmueli, Peter C. Bruce, Peter Gedeck, Nitin R. Patel, 2020

IArtificial.net https://www.iartificial.net/maquinas-de-vectores-de-soporte-svm/

https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm

Introducción a Regresión Logística. https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148 2019.

https://aprendeia.com/