INTRODUCCIÓN AL APRENDIZAJE AUTOMÁTICO

APRENDIZAJE SUPERVISADO PARA REGRESIÓN Y CLASIFICACIÓN

SVM – Árboles de Decisión (TDIDT) - Random Forest

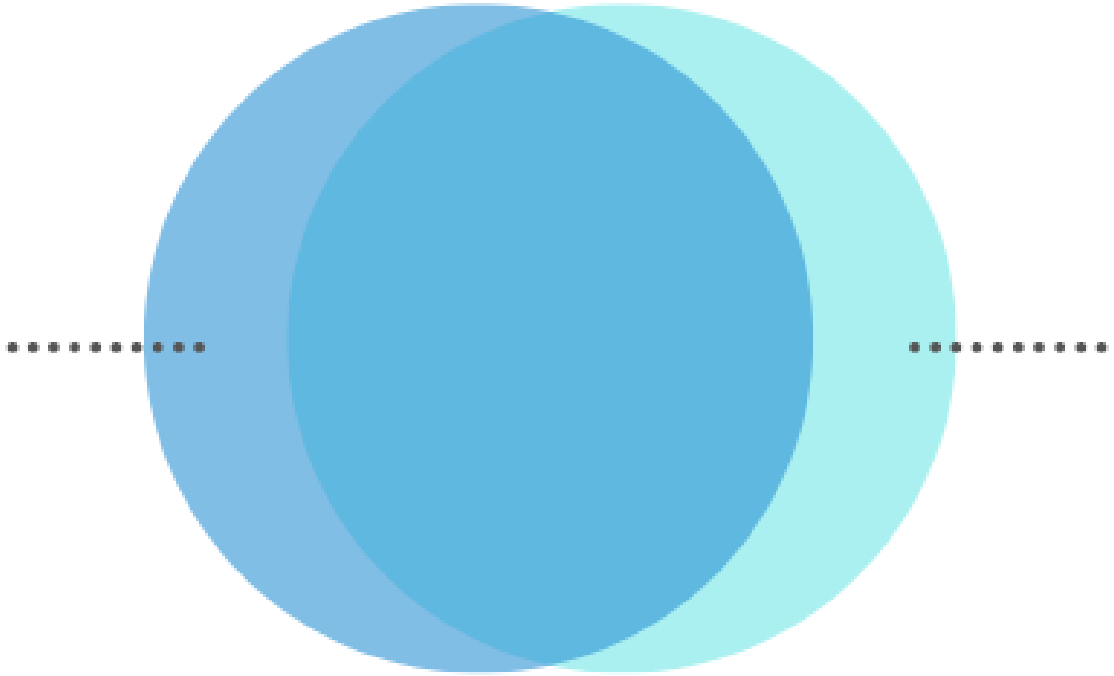
LAURA DIAZ DÁVILA – FRANCISCO TAMARIT

REGRESIÓN

CLASIFICACIÓN

GENERALIZAR PARA PREDECIR CON APRENDIZAJE SUPERVISADO

REGRESIONES Y
CLASIFICACIONES



REDES NEURONALES:
MLP – CNN
MLP

REGRESIÓN LINEAL MIULTIVARIADA

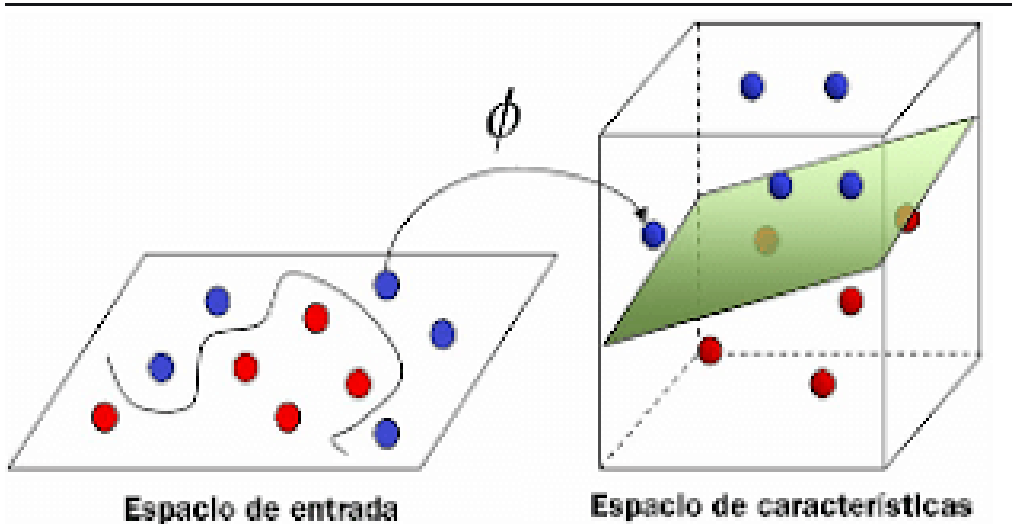
REGRESIÓN LOGÍSTICA

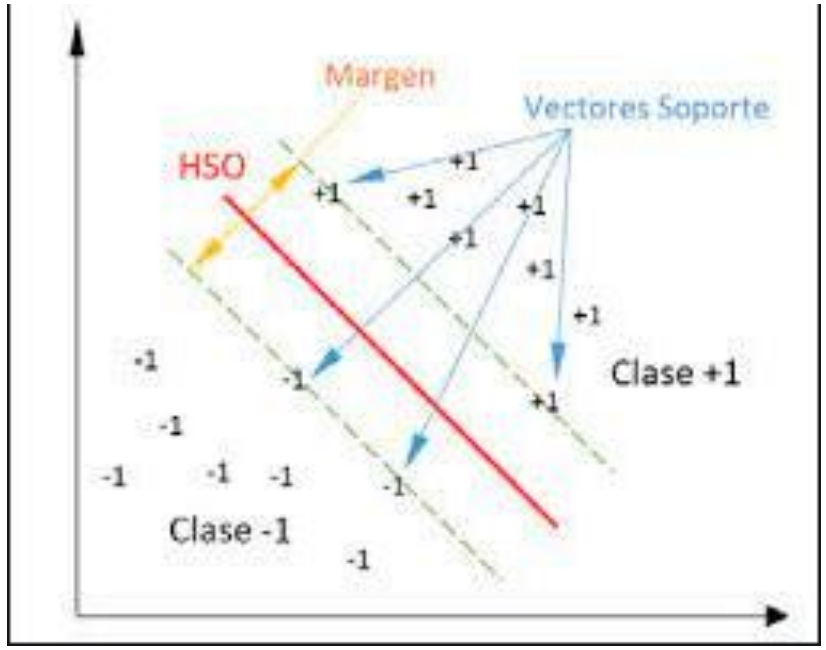
SUPPORT VECTOR MACHINES

ARBOLES DE DECISIÓN POR INDUCCIÓN (TOP DOWN INDUCTION DECISION TREE)

SUPPORT VECTOR MACHINES

Vladimir Vapnik

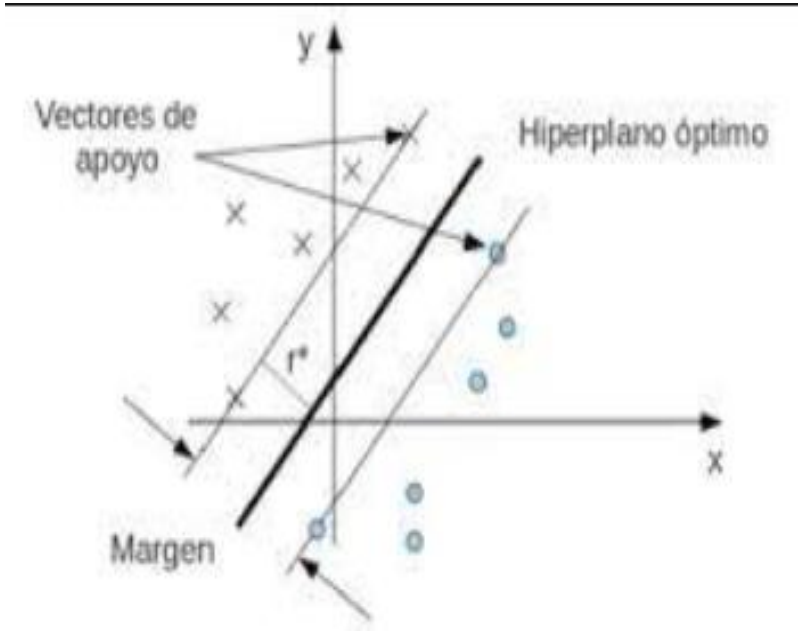




BASADO EN UN KERNEL

PRECISIÓN MUY ALTA EN RELACIÓN A LA REGRESIÓN LOGÍSTICA Y TDIDT

SVM: APLICACIONES



Detección y diagnóstico de fallas para la dinámica lateral de un automóvil

Diagnóstico clínico de la Enfermedad de Párkinson y el Temblor Esencial

Clasificación de células cervicales empleando rasgos del núcleo

Filtros de spam para correo electrónico

Reconocimiento de imágenes de satélites (DETECTAR qué partes contiene nubes, tierra, agua, hielo, etc.)

Knowledge Discovery in Database

Descubrimiento automatizado de patrones desconocidos

Predicción automatizada de tendencias y comportamientos

EJEMPLOS DE USOS ESPECÍFICOS:

SEGMENTACIÓN DE MERCADO

COMPORTAMIENTO DE PERFILES DEL CIUDADANO

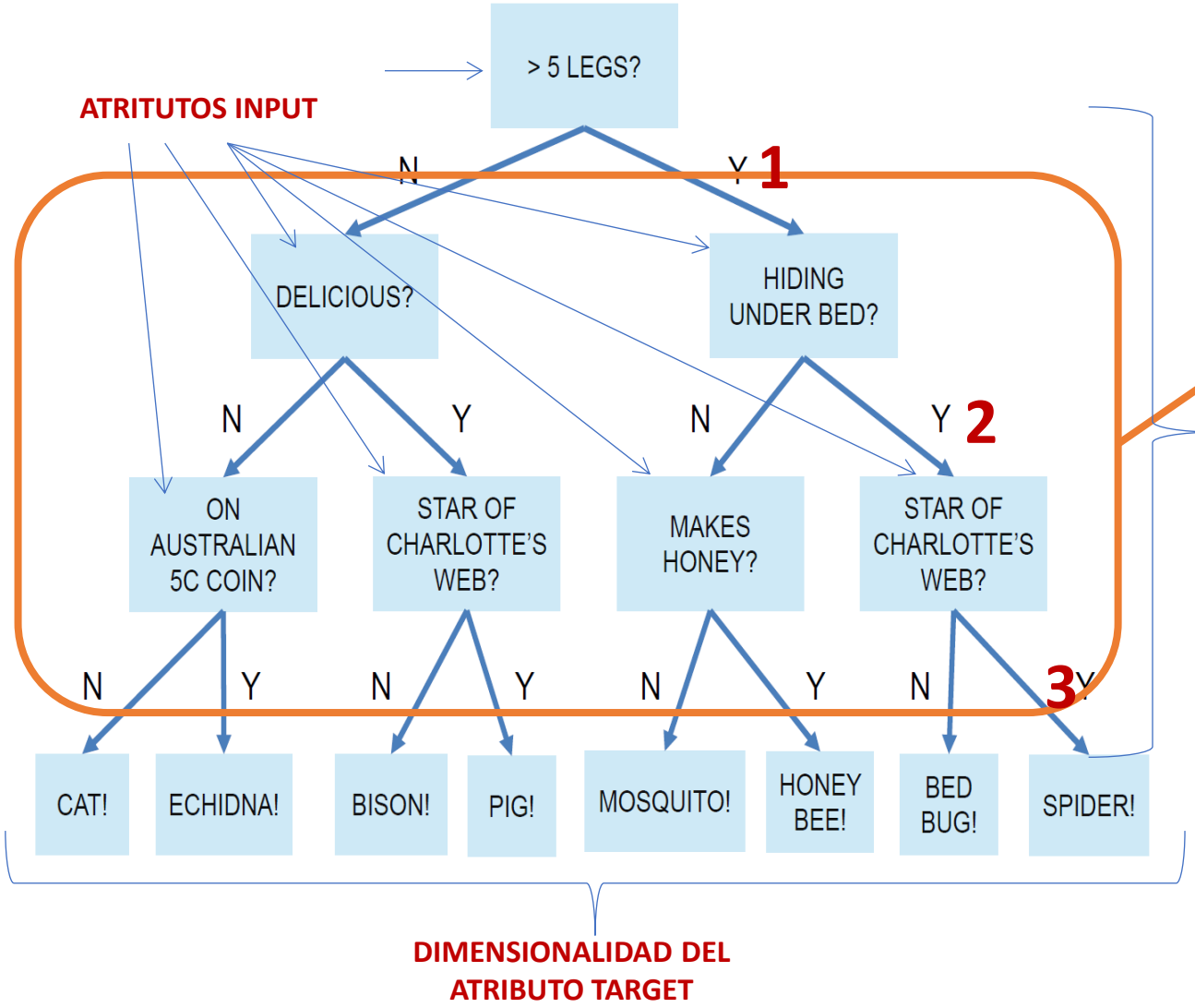
DETECCIÓN DE FRAUDES O DE DATOS ANÓMALOS

MARKETING DIRECTO

PERFIL SOCIOECONÓMICO DEL ESTUDIANTE EN RELACIÓN A SU RENDIMIENTO ACADÉMICO

Fuente: https://www.acadian-asset.com/viewpoints/machine-learning-in-quant-investing-revolution-or-evolution

ÁRBOLES DE DECISIÓN – RANDOM FOREST



LÓGICA BINARIA

Arbol *balanceado*:

- Para cada nodo interno, hay dos ramas.
- *Altura del árbol*: Longitud del camino más largo desde la raíz a una hoja.
- Para árboles balanceados de altura n, tenemos 2^n hojas:
  - n = 3 –> 8 hojas (el de la figura)
  - n = 10 –> 1,024 hojas
  - n = 20 –> aprox. 1 millón de hojas
  - ...

PROFUNDIDAD

Magnitud termodinámica que indica el grado de desorden molecular de un sistema

Entropía y Ganancia de información

Se encarga de medir la energía que NO es usada (Información)

S ↓

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

Tumba de Boltzmann en el Cementerio central de Viena con la fórmula de entropía.

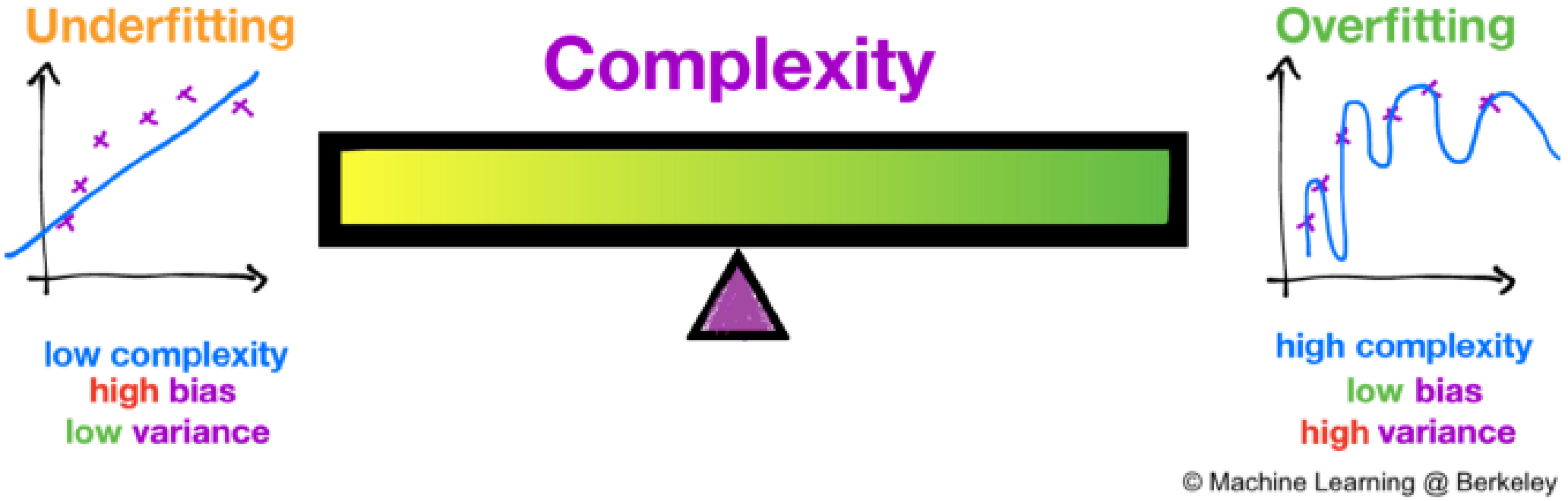
Cantidad de estados posibles

Probabilidad de que tome el i-ésimo micro estado.
Dsitribución de Boltzmann

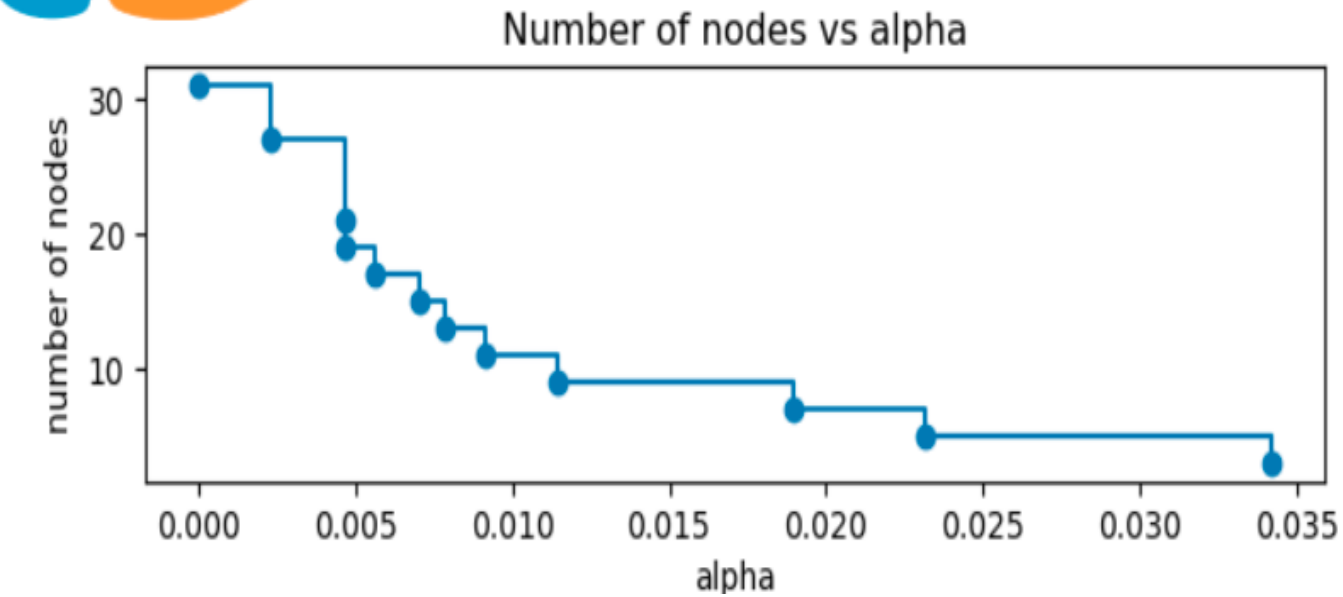
Fuente: https://www.cs.put.poznan.pl/jstefanowski/sed/DM-5-newtrees.pdf

¡PODA PARA DISMINUIR LA COMPLEJIDAD DEL ÁRBOL!

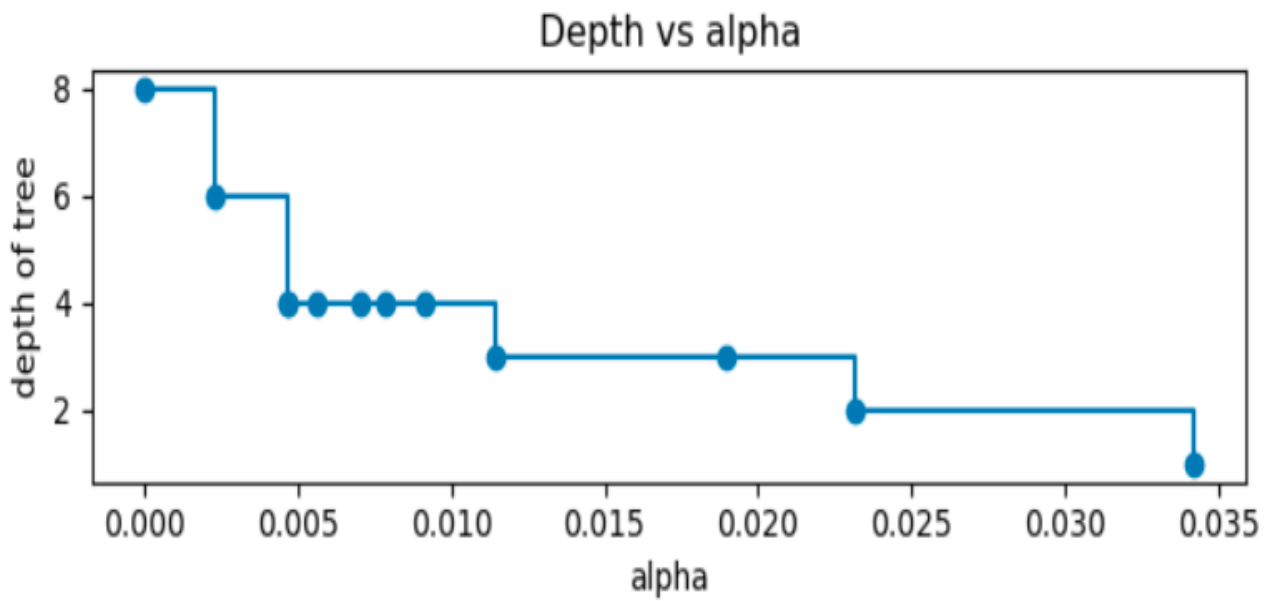
SKLEARN: $R_\alpha(T) = R(T) + \alpha|\widetilde{T}|$

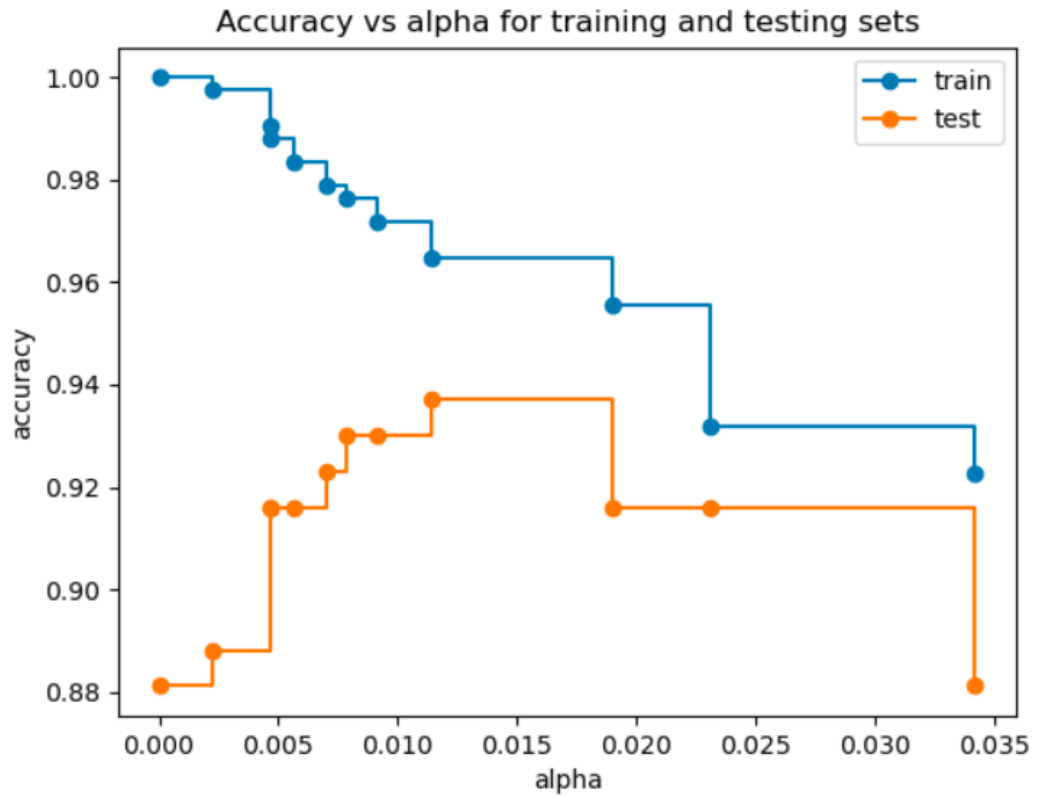


Fuente: https://ml.berkeley.edu/blog/posts/crash-course/part-4/



$$R_\alpha(T) = R(T) + \alpha|\widetilde{T}|$$

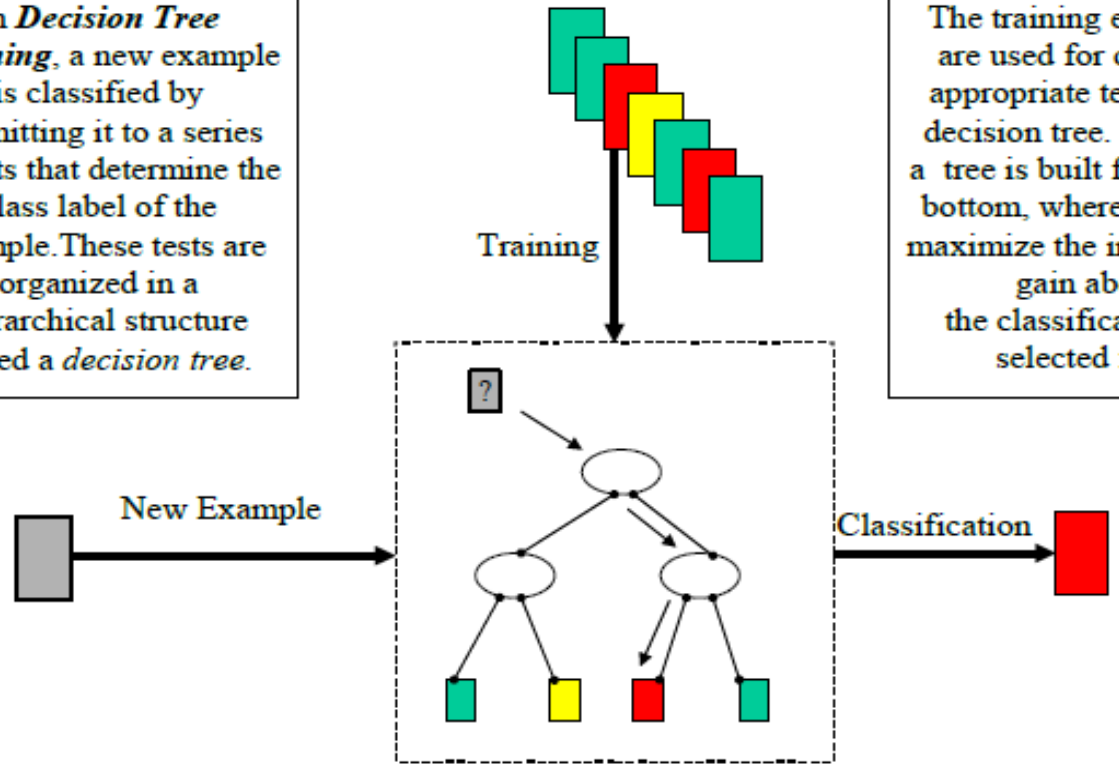




https://scikit-learn.org/stable/auto_examples/tree/plot_cost_complexity_pruning.html#sphx-glr-auto-examples-tree-

MODELOS DT: APRENDIZAJE SUPERVISADO

In ***Decision Tree Learning***, a new example is classified by submitting it to a series of tests that determine the class label of the example. These tests are organized in a hierarchical structure called a *decision tree*.



The training examples are used for choosing appropriate tests in the decision tree. Typically, a tree is built from top to bottom, where tests that maximize the information gain about the classification are selected first.

EN CADA ÉPOCA DEL ENTRENAMIENTO, EL ALGORITMO INTELIGENTE DECIDE CUÁL ES EL "NODO" (CARACTERÍSTICA) DE MAYOR GANANCIA DE INFORMACIÓN (MÍNIMA ENTROPÍA)

PARA CADA ÉPOCA, TODOS LOS ATRIBUTOS SON CANDIDATOS, AUNQUE HAYAN SIDO ELEGIDOS ANTERIORMENTE

EVOLUCIONA HASTA QUE LA ENTROPÍA ALCANCE LA TOLERANCIA FIJADA (HIPERPARÁMETRO)

NUNCA TODOS LOS EJEMPLOS QUEDAN COMPLETAMENTE REPRESENTADOS, LA COMPLEJIDAD DEL PROBLEMA LO IMPOSIBILITA

EJEMPLO

* Detectar características socioeconómicas de los estudiantes de asignaturas en contextos de masividad en la UNC –carreras de Ingeniería para

- Aprobo Inf en Cursada < 0,5000
  - Ingles < 0,5000 then Cluster_SOM_1 = c_som_2_2 (100,00 % of 381 examples)
  - Ingles >= 0,5000
    - Procedencia_Cordoba < 0,5000
      - Demora en Cursarla < 0,5000 then Cluster_SOM_1 = c_som_3_2 (53,68 % of 190 examples)
      - Demora en Cursarla >= 0,5000 then Cluster_SOM_1 = c_som_2_1 (99,64 % of 842 examples)
    - Procedencia_Cordoba >= 0,5000
      - Portugues < 0,5000
        - Demora en Cursarla < 0,5000 then Cluster_SOM_1 = c_som_3_2 (84,82 % of 494 examples)
        - Demora en Cursarla >= 0,5000 then Cluster_SOM_1 = c_som_1_2 (91,99 % of 1811 examples)
      - Portugues >= 0,5000
        - CumplePlan < 2,5000 then Cluster_SOM_1 = c_som_1_1 (96,90 % of 2127 examples)
        - CumplePlan >= 2,5000 then Cluster_SOM_1 = c_som_3_2 (65,32 % of 222 examples)
- Aprobo Inf en Cursada >= 0,5000 then Cluster_SOM_1 = c_som_3_1 (98,97 % of 584 examples)

Base de Datos de SIU_Guaraní, de alumnos de las carreras de Ingeniería de la Universidad Nacional de Córdoba, inscriptos en la materia Informática en el primer cuatrimestre de los años 2012-2013 relevada en Julio de 2014, más de 1500 registros.

Variables, trece en total:

- ❑ La fuente para costear estudios del alumno: de su propio trabajo, de su familia y/o de beca.
- ❑ Los últimos estudios alcanzados por su padre y madre.
- ❑ El género.
- ❑ La ubicación de procedencia (generando tres variables booleanas, si es argentino, si es de la Provincia de Córdoba, y si es de Córdoba).
- ❑ Si el alumno aprobó Informática durante la cursada.
- ❑ Si el alumno realizó la cursada de Informática acorde a lo establecido en el plan de estudios.
- ❑ Dos variables que determinan el rendimiento del alumno en su primer año de ingreso y su desempeño en el total de años cursados respecto al plan de estudios.

MODELOS DT

NODOS: ¡LÓGICA BINARIA!

ATRIBUTOS INPUT SELECCIONADOS EN CADA NIVEL

ATRIBUTOS INPUT CONTINUOS O DISCRETOS

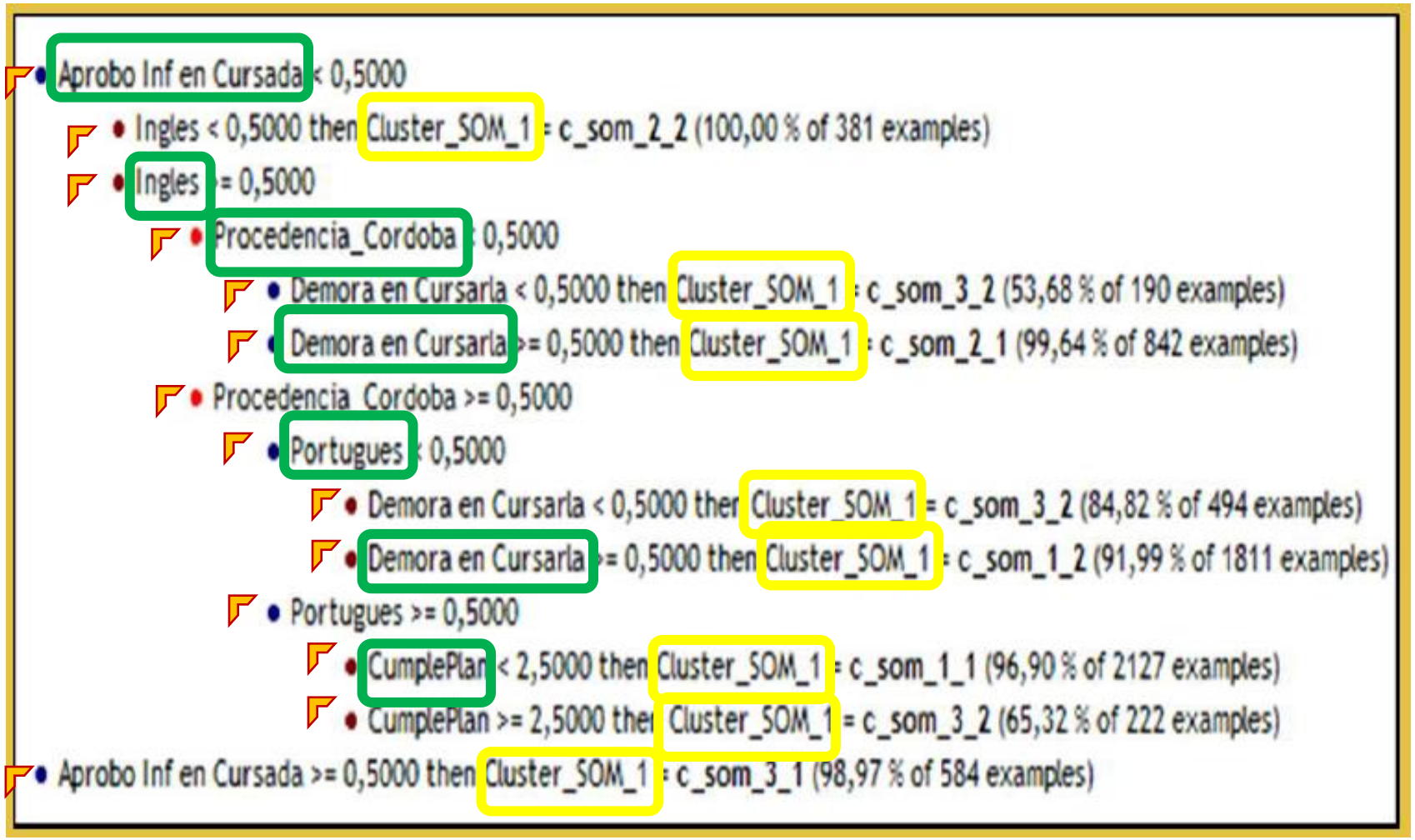
PRIMER NIVEL

SEGUNDO NIVEL

TERCER NIVEL

CUARTO NIVEL

QUINTO NIVEL

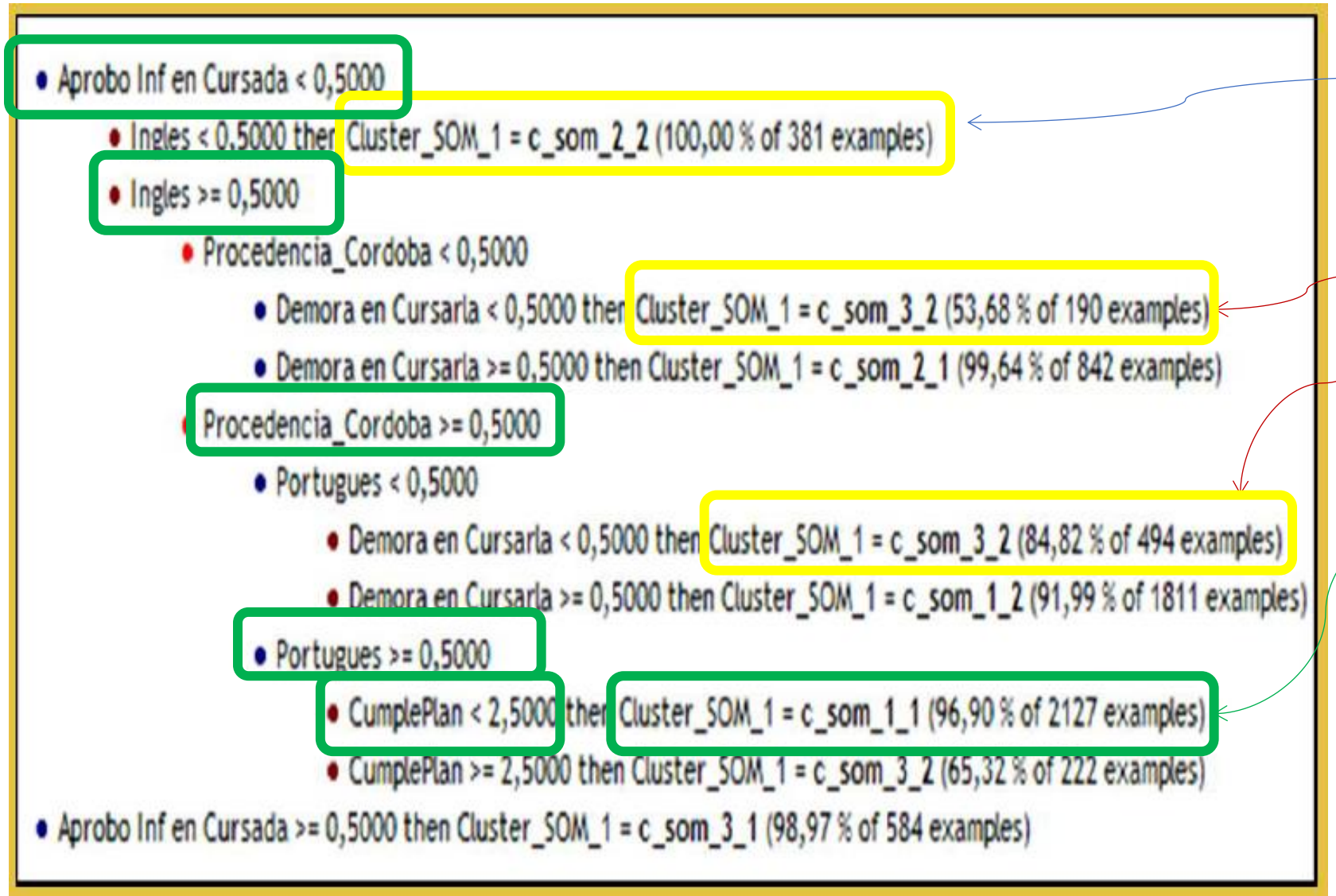


ATRIBUTO TARGET: CATEGÓRICO. POLITÓMICO O DICOTÓMICO

EN NUESTRO EJEMPLO: ¡6 CATEGORÍAS O CLASES!

MODELOS DT: INTERPRETACIONES

ACERCA DE SUS HOJAS Y LAS REGLAS



HOJA: EXPRESA UNA CATEGORÍA DEL ATRIBUTO TARGET

PUEDEN EXISTIR MÁS DE UNA REGLA PARA CADA CATEGORÍA DEL ATRIBUTO TARGET

SU TRAZA EXPRESA UNA REGLA: EXPRESIÓN CONJUNTIVA

TODAS LAS CATEGORÍAS ESTÁN REPRESENTADAS

SON EL ÚLTIMO NIVEL DEL ÁRBOL

MODELOS TDIDT
INTERPRETACIONES

QUE SEA POSIBLE COMPRENDER SUS RESULTADOS A TRAVÉS DE UNA EXPRESIÓN CONJUNTIVA IMPLICA QUE EL MODELO SEA EXPLICABLE. SIN EMBARGO, EL ALGORITMO INTELIGENTE USA, POR EJEMPLO, CRITERIOS DE ENTROPÍA PARA DECIDIR (SUBSIMBÓLICO)

MIENTRAS MÁS PROFUNDIDAD (HIPERPARÁMETRO) TENGA EL ÁRBOL, MÁS DENSO Y EXACTO RESULTARÁ (SOBREAJUSTE). UN ÁRBOL COMPLETO (INASIBLE) ES AQUÉL EN EL QUE TODOS SUS EJEMPLOS ESTÁN REPRESENTADOS

ES NECESARIO ENCONTRAR EL JUSTO EQUILIBRIO ENTRE INTERPRETABILIDAD Y DENSIDAD (COMPLEJIDAD) DEL MODELO

TODOS LOS EJEMPLOS TERMINAN ASIGNÁNDOSE A ALGUNA DE SUS HOJAS, SÓLO QUE NO QUEDAN COMPLETAMENTE REPRESENTADOS EN ELLAS

c) Descripción de las características de los alumnos según si cumplen o no el plan

- c2d_Aprobo Inf en Cursada_1 in [_1_0,00] then c2d_CumplePlan_1 = _1_3,00 (98,01 % of 653 examples)
- c2d_Aprobo Inf en Cursada_1 in [_2_1,00]
  - Edad < 20,5000
    - Madre Ultimos estudios grupos < 2,5000 then c2d_CumplePlan_1 = _1_3,00 (55,91 % of 127 examples)
    - Madre Ultimos estudios grupos >= 2,5000 then c2d_CumplePlan_1 = _2_4,00 (61,40 % of 272 examples)
  - Edad >= 20,5000 then c2d_CumplePlan_1 = _1_3,00 (65,13 % of 195 examples)

Error rate	0,2116
Values prediction	Confusion matrix

Value	Recall	1-Precision
_1_3,00	0,8641	0,1399
_2_4,00	0,5417	0,4500

	_1_3,00	_2_4,00	Sum
_1_3,00	744	117	861
_2_4,00	121	143	264
Sum	865	260	1125

Results

Attribute	Target	Input	Illustrative
c2d_Procedencia_Argentina_1	-	yes	-
c2d_Procedencia_Cordoba_1	-	yes	-
c2d_Procedencia_Cordoba_Capital_1	-	yes	-
c2d_CosteaEst_Trabajo_1	-	yes	-
c2d_CosteaEst_Familia_1	-	yes	-
c2d_CosteaEst_Beca_1	-	yes	-
c2d_pc casa_1	-	yes	-
c2d_PC universidad_1	-	yes	-
c2d_Internet casa_1	-	yes	-
c2d_Internet universidad_1	-	yes	-
c2d_internet para ocio_1	-	yes	-
c2d_Internet para capacitarse_1	-	yes	-
c2d_Demora en Cursarla_1	-	yes	-
c2d_Aprobo Inf en Cursada_1	-	yes	-
c2d_CumplePlan_1	yes	-	-
c2d_RitmoInicial_1	-	yes	-

- c2d_Aprobo Inf en Cursada_1 in [_1_0,00] then c2d_CumplePlan_1 = _1_3,00 (98,01 % of 653 examples)
- c2d_Aprobo Inf en Cursada_1 in [_2_1,00]

  - c2d_RitmoInicial_1 in [_1_0,00] then c2d_CumplePlan_1 = _1_3,00 (100,00 % of 9 examples)
  - c2d_RitmoInicial_1 in [_2_1,00] then c2d_CumplePlan_1 = _2_4,00 (66,67 % of 3 examples)
  - c2d_RitmoInicial_1 in [_3_2,00]
    - c2d_CosteaEst_Beca_1 in [_1_0,00]
      - c2d_internet para ocio_1 in [_1_0,00] then c2d_CumplePlan_1 = _1_3,00 (62,16 % of 74 examples)
      - c2d_internet para ocio_1 in [_2_1,00]
        - c2d_Internet universidad_1 in [_1_0,00]
          - c2d_Procedencia_Cordoba_1 in [_1_0,00] then c2d_CumplePlan_1 = _2_4,00 (51,72 % of 87 examples)
          - c2d_Procedencia_Cordoba_1 in [_2_1,00] then c2d_CumplePlan_1 = _1_3,00 (53,90 % of 141 examples)
        - c2d_Internet universidad_1 in [_2_1,00] then c2d_CumplePlan_1 = _1_3,00 (56,41 % of 117 examples)
    - c2d_CosteaEst_Beca_1 in [_2_1,00] then c2d_CumplePlan_1 = _2_4,00 (64,15 % of 53 examples)
  - c2d_RitmoInicial_1 in [_4_3,00] then c2d_CumplePlan_1 = _2_4,00 (60,00 % of 110 examples)

- Scikit-learn 1.2.dev0 (dev) documentation (ZIP 62.9 MB)
- Scikit-learn 1.1.2 (stable) documentation (ZIP 62.4 MB)
- Scikit-learn 1.0.2 documentation (ZIP 59.4 MB)
- Scikit-learn 0.24.2 documentation (ZIP 67.6 MB)
- Scikit-learn 0.23.2 documentation (PDF 51.0 MB)
- Scikit-learn 0.22.2 documentation (PDF 48.5 MB)
- Scikit-learn 0.21.3 documentation (PDF 46.7 MB)
- Scikit-learn 0.20.4 documentation (PDF 45.2 MB)
- Scikit-learn 0.19.2 documentation (PDF 42.2 MB)
- Scikit-learn 0.18.2 documentation (PDF 46.5 MB)
- Scikit-learn 0.17.1 documentation (PDF 46.0 MB)
- Scikit-learn 0.16.1 documentation (PDF 56.8 MB)
- Scikit-learn 0.15-git documentation

scikit-learn 1.1.2

1.10. Decision Trees

- 1.10.1. Classification
- 1.10.2. Regression
- 1.10.3. Multi-output problems
- 1.10.4. Complexity
- 1.10.5. Tips on practical use
- 1.10.6. Tree algorithms: ID3, C4.5, C5.0 and CART
- 1.10.7. Mathematical formulation
- 1.10.8. Minimal Cost-Complexity Pruning

1.11. Ensemble methods

- 1.11.1. Bagging meta-estimator
- 1.11.2. Forests of randomized trees
- 1.11.3. AdaBoost
- 1.11.4. Gradient Tree Boosting
- 1.11.5. Histogram-Based Gradient Boosting
- 1.11.6. Voting Classifier
- 1.11.7. Voting Regressor
- 1.11.8. Stacked generalization

https://scikit-learn.org/stable/supervised_learning.html#supervised-learning

https://scikit-learn.org/dev/versions.html

Definir algoritmo

DecisionTreeClassifier()

Predecir modelo

predict(x)

Entrenar modelo

fit(x, y)

1.10. Decision Trees

- 1.10.1. Classification
- 1.10.2. Regression
- 1.10.3. Multi-output problems
- 1.10.4. Complexity
- 1.10.5. Tips on practical use
- 1.10.6. Tree algorithms: ID3, C4.5, C5.0 and CART
- 1.10.7. Mathematical formulation
- 1.10.8. Minimal Cost-Complexity Pruning

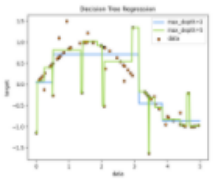
1.11. Ensemble methods

- 1.11.1. Bagging meta-estimator
- 1.11.2. Forests of randomized trees
- 1.11.3. AdaBoost
- 1.11.4. Gradient Tree Boosting
- 1.11.5. Histogram-Based Gradient Boosting
- 1.11.6. Voting Classifier
- 1.11.7. Voting Regressor
- 1.11.8. Stacked generalization

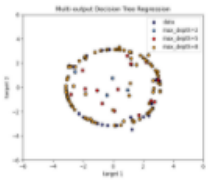
https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeCl

Decision Trees ¶

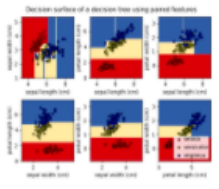
Examples concerning the `sklearn.tree` module.



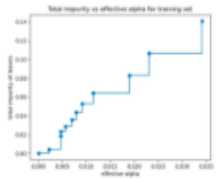
Decision Tree Regression



Multi-output Decision Tree Regression



Plot the decision surface of a decision tree on the iris dataset



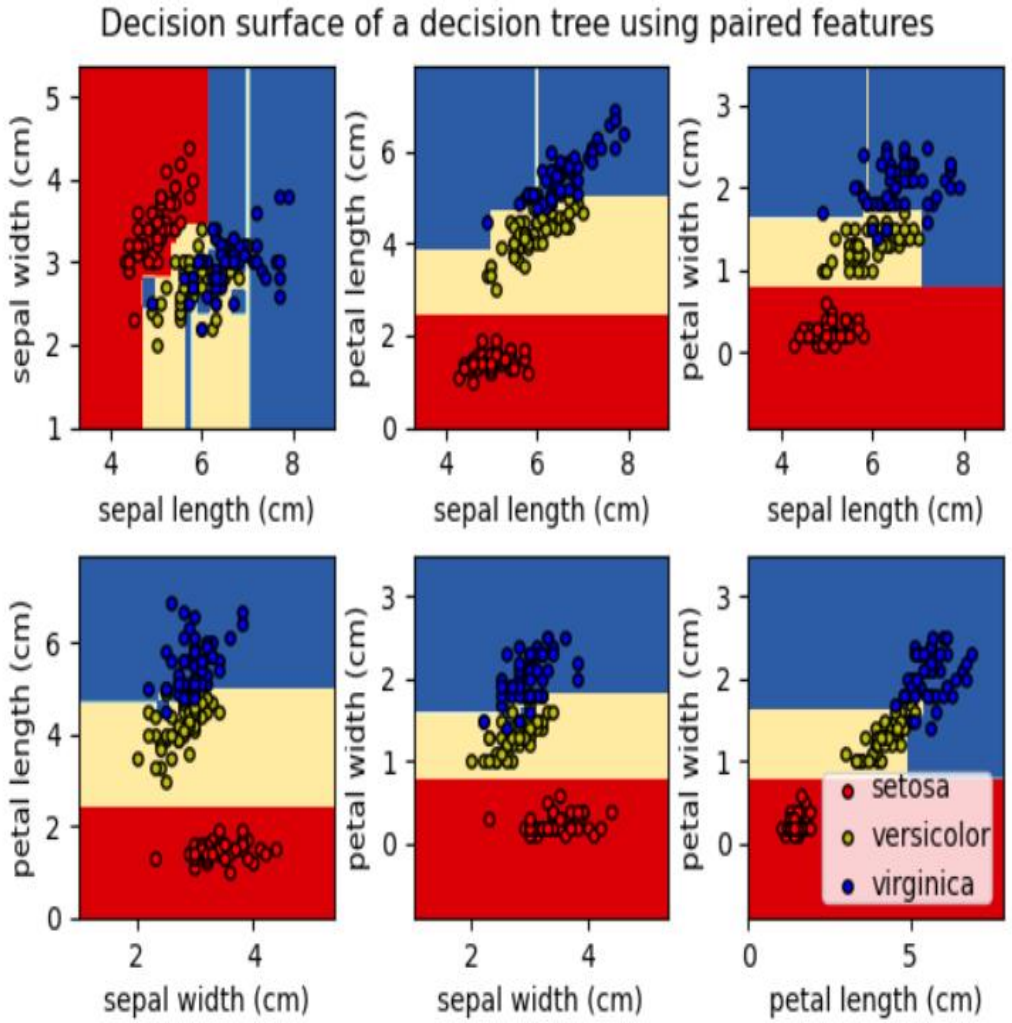
Post pruning decision trees with cost complexity pruning

Understanding the decision tree structure

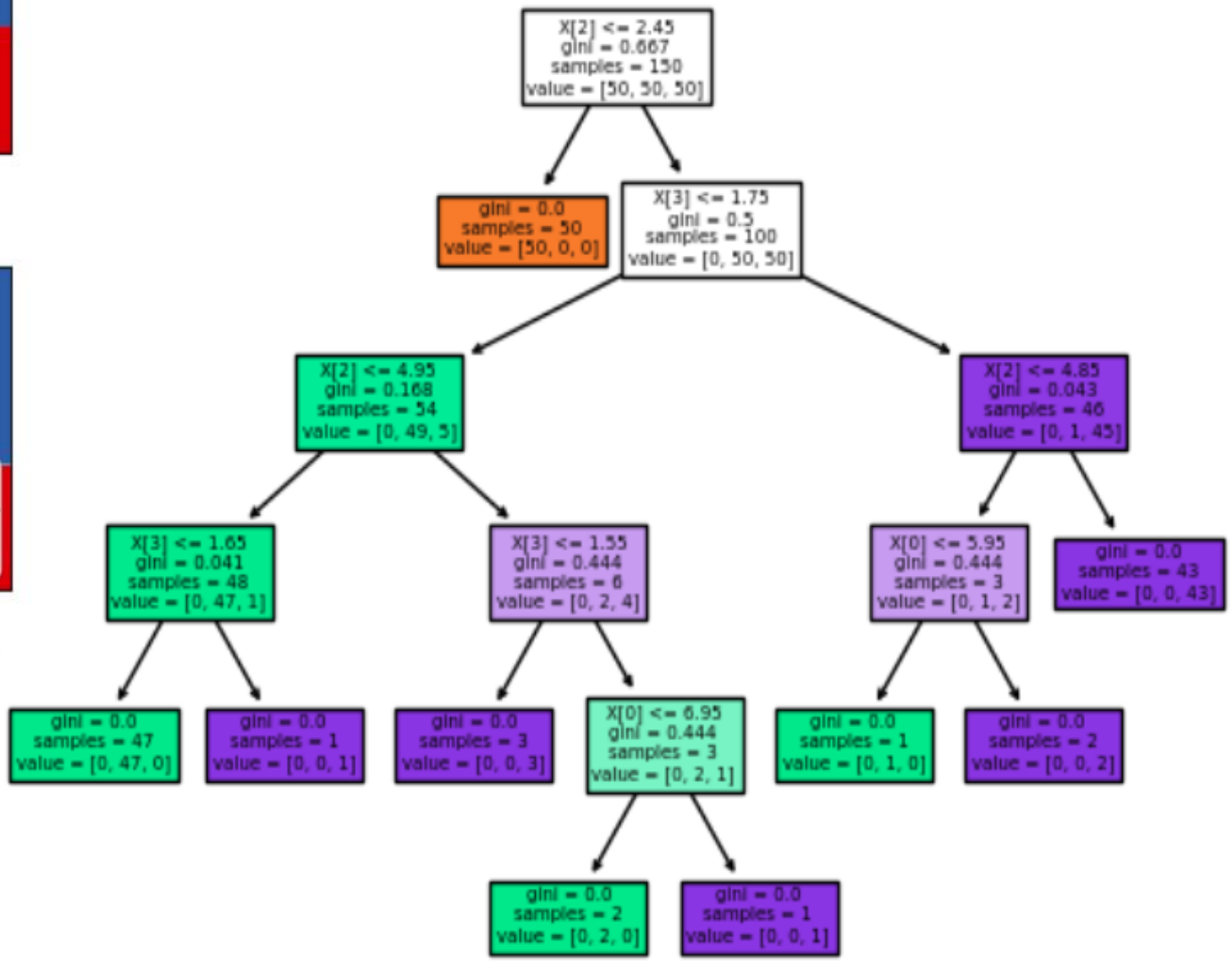
https://scikit-learn.org/dev/auto_examples/index.html#classif

MODELOS IDT

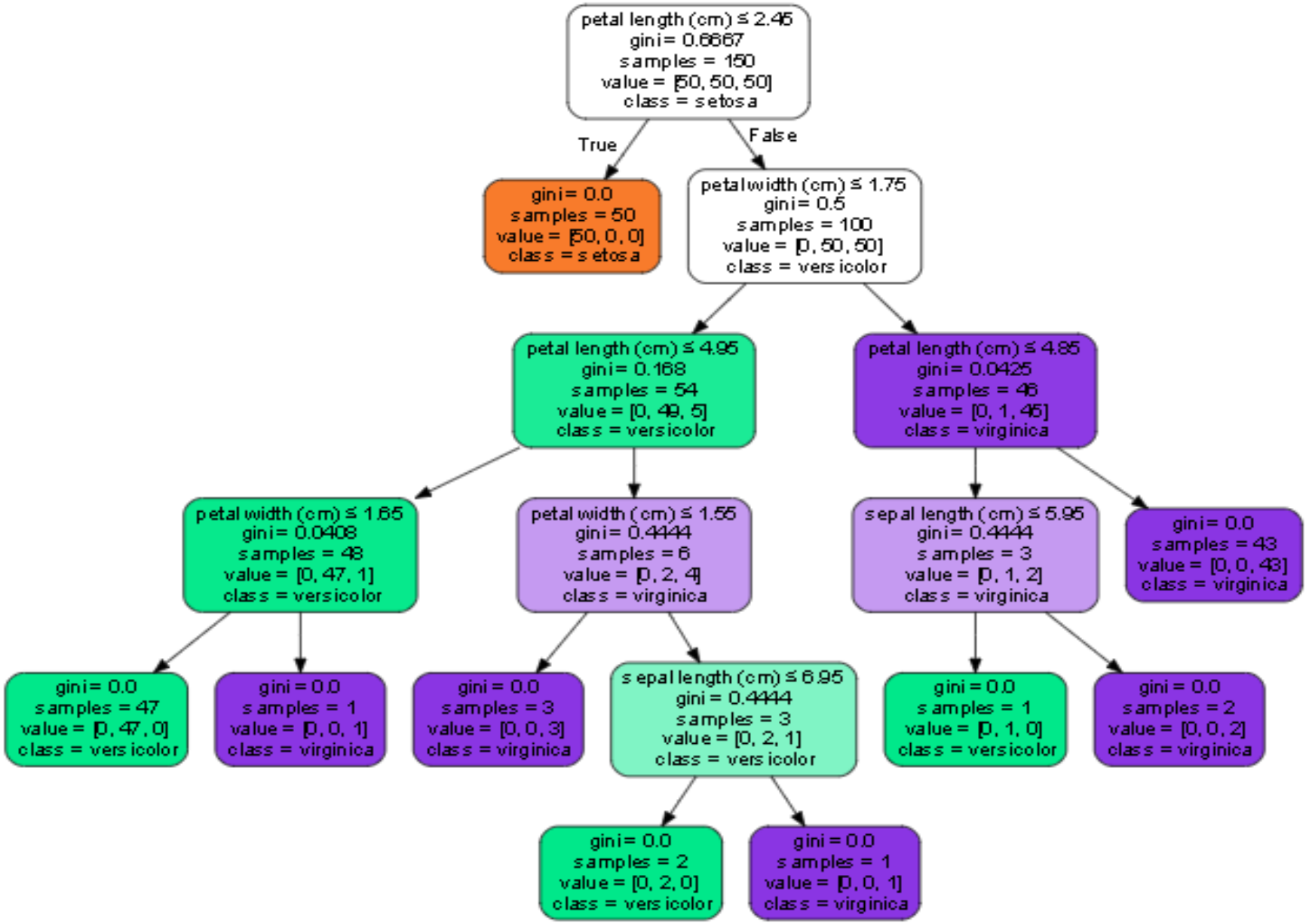
https://scikit-learn.org/stable/modules/tree.html



1.10.1. Classification

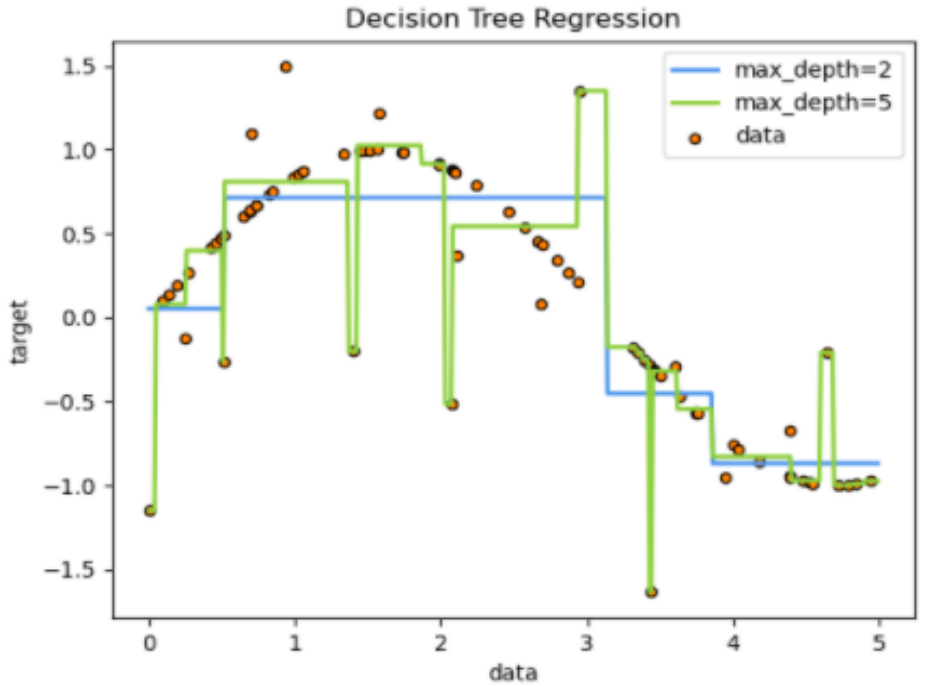



scikit
learn


petal length (cm) ≤ 2.45
gini = 0.6667
samples = 150
value = [50, 50, 50]
class = setosa
True
False
gini = 0.0
samples = 50
value = [50, 0, 0]
class = setosa
petal width (cm) ≤ 1.75
gini = 0.5
samples = 100
value = [0, 50, 50]
class = versicolor
petal length (cm) ≤ 4.95
gini = 0.168
samples = 54
value = [0, 49, 5]
class = versicolor
petal length (cm) ≤ 4.85
gini = 0.0425
samples = 46
value = [0, 1, 45]
class = virginica
petal width (cm) ≤ 1.65
gini = 0.0408
samples = 48
value = [0, 47, 1]
class = versicolor
petal width (cm) ≤ 1.55
gini = 0.4444
samples = 6
value = [0, 2, 4]
class = virginica
sepal length (cm) ≤ 5.95
gini = 0.4444
samples = 3
value = [0, 1, 2]
class = virginica
gini = 0.0
samples = 43
value = [0, 0, 43]
class = virginica
gini = 0.0
samples = 47
value = [0, 47, 0]
class = versicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = virginica
gini = 0.0
samples = 3
value = [0, 0, 3]
class = virginica
sepal length (cm) ≤ 6.95
gini = 0.4444
samples = 3
value = [0, 2, 1]
class = versicolor
gini = 0.0
samples = 1
value = [0, 1, 0]
class = versicolor
gini = 0.0
samples = 2
value = [0, 0, 2]
class = virginica
gini = 0.0
samples = 2
value = [0, 2, 0]
class = versicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = virginica

1.10.2. Regresión

The deeper the tree, the more complex the decision rules and the fitter the model.



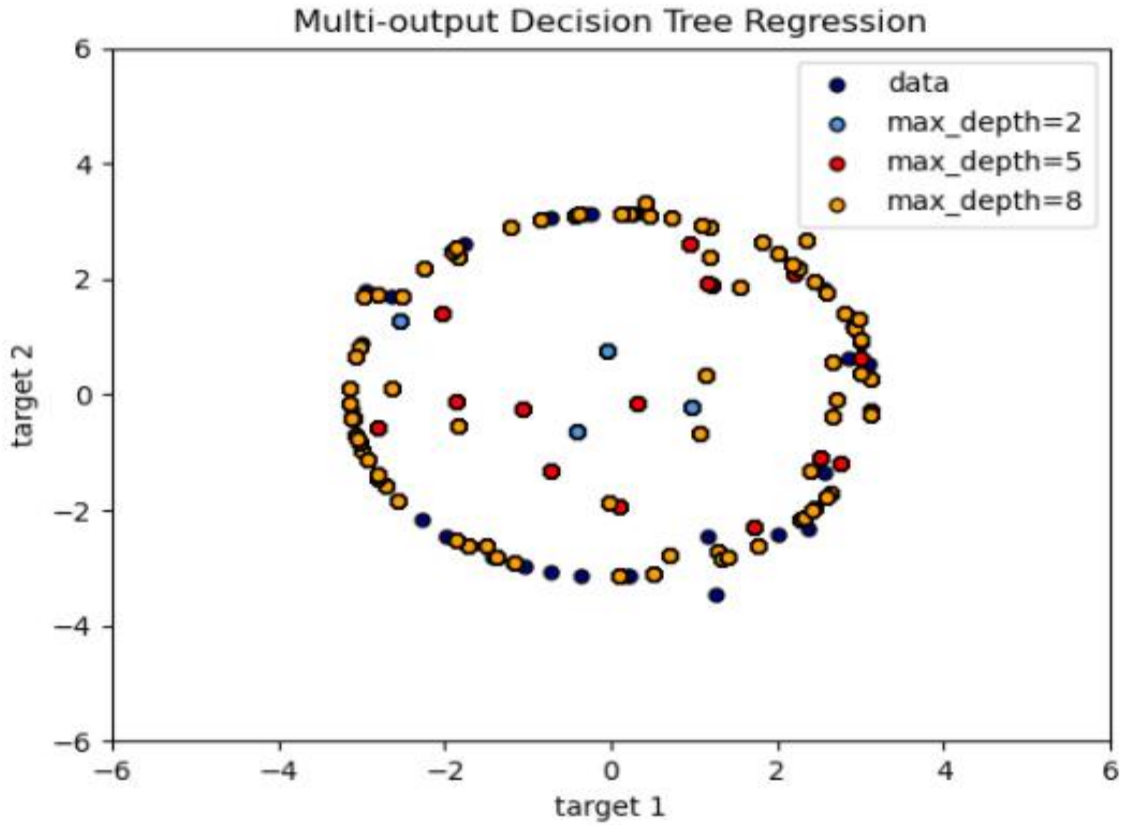
DecisionTreeRegressor class

```
>>> from sklearn import tree
>>> X = [[0, 0], [2, 2]]
>>> y = [0.5, 2.5]
>>> clf = tree.DecisionTreeRegressor()
>>> clf = clf.fit(X, y)
>>> clf.predict([[1, 1]])
array([0.5])
```

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

1.10.3. Multi-output problems

`(n_samples, n_outputs)`



1.10.4. Complexity

1.10.5. Tips on practical use

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

¡MALAS NOTICIAS! IMPLEMENTA SÓLO AL MODELO CART

ID3 (Iterative Dichotomiser 3) was developed in 1986 by Ross Quinlan. The algorithm creates a multiway tree, finding for each node (i.e. in a greedy manner) the categorical feature that will yield the largest information gain for categorical targets. Trees are grown to their maximum size and then a pruning step is usually applied to improve the ability of the tree to generalise to unseen data.

C4.5 is the successor to ID3 and removed the restriction that features must be categorical by dynamically defining a discrete attribute (based on numerical variables) that partitions the continuous attribute value into a discrete set of intervals. C4.5 converts the trained trees (i.e. the output of the ID3 algorithm) into sets of if-then rules. These accuracy of each rule is then evaluated to determine the order in which they should be applied. Pruning is done by removing a rule's precondition if the accuracy of the rule improves without it.

C5.0 is Quinlan's latest version release under a proprietary license. It uses less memory and builds smaller rulesets than C4.5 while being more accurate.

CART (Classification and Regression Trees) is very similar to C4.5, but it differs in that it supports numerical target variables (regression) and does not compute rule sets. CART constructs binary trees using the feature and threshold that yield the largest information gain at each node.

scikit-learn uses an optimised version of the CART algorithm; however, scikit-learn implementation does not support categorical variables for now.

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

MODELOS TDIDT: CART

1.10.7. Mathematical formulation

1.10.7.1. Classification criteria

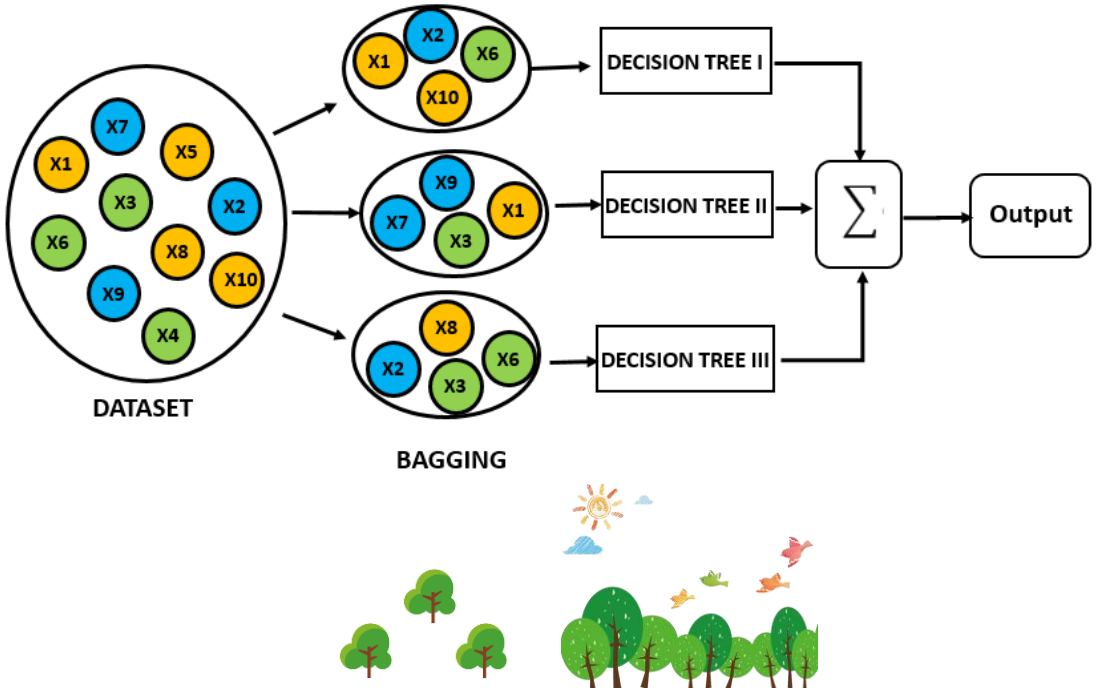
- Gini
- Entropy

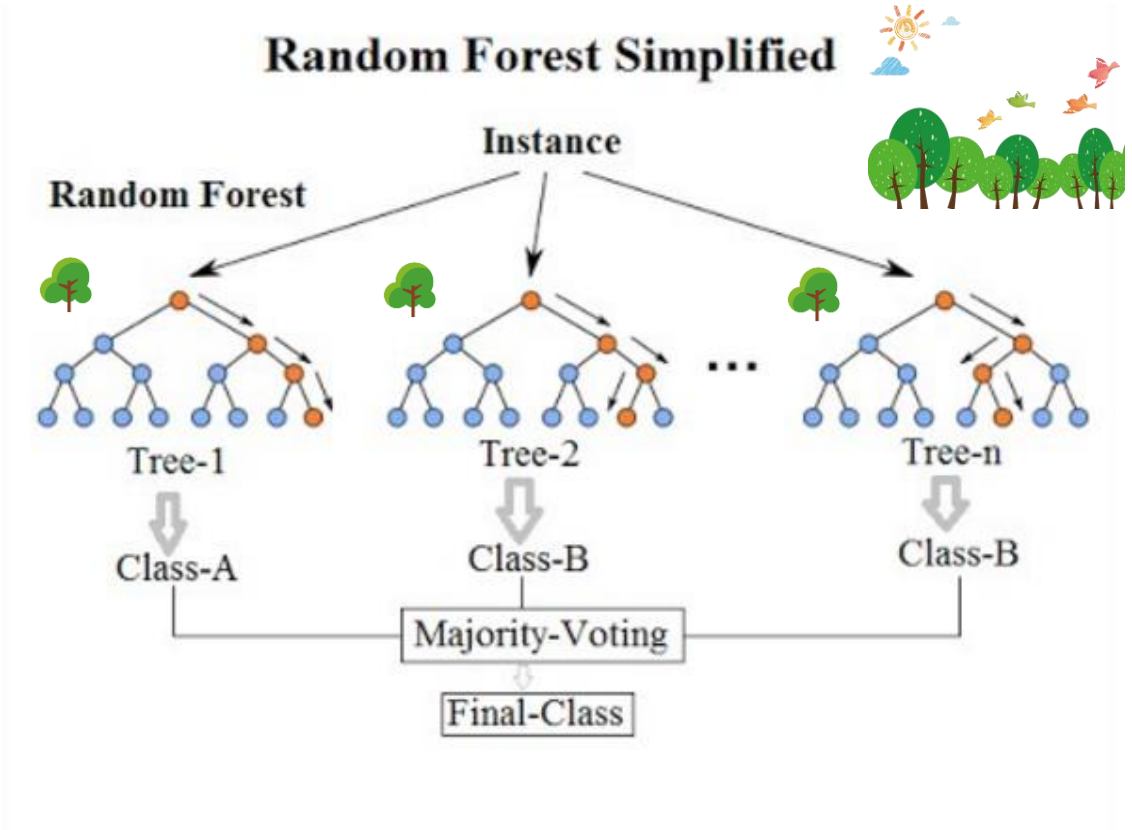
1.10.7.2. Regression criteria

- Mean Squared Error:
- Half Poisson deviance

Ajuste más lento

MODELOS RANDOM FOREST



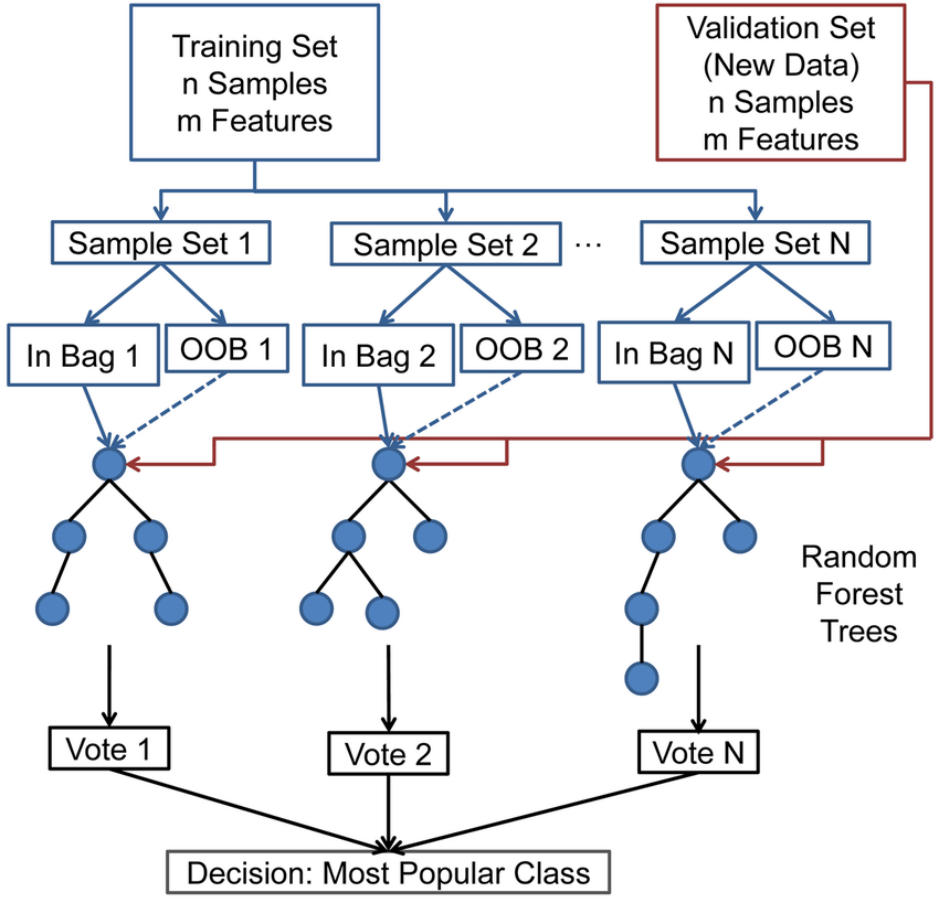


Fuente: https://bookdown.org/content/2031/ensambladores-random-forest-parte-i.html y Wikiedia

RESUMIENDO…

Random

- **Bagging particiona en N subconjuntos**
- **Escoge m atributos, m < M**



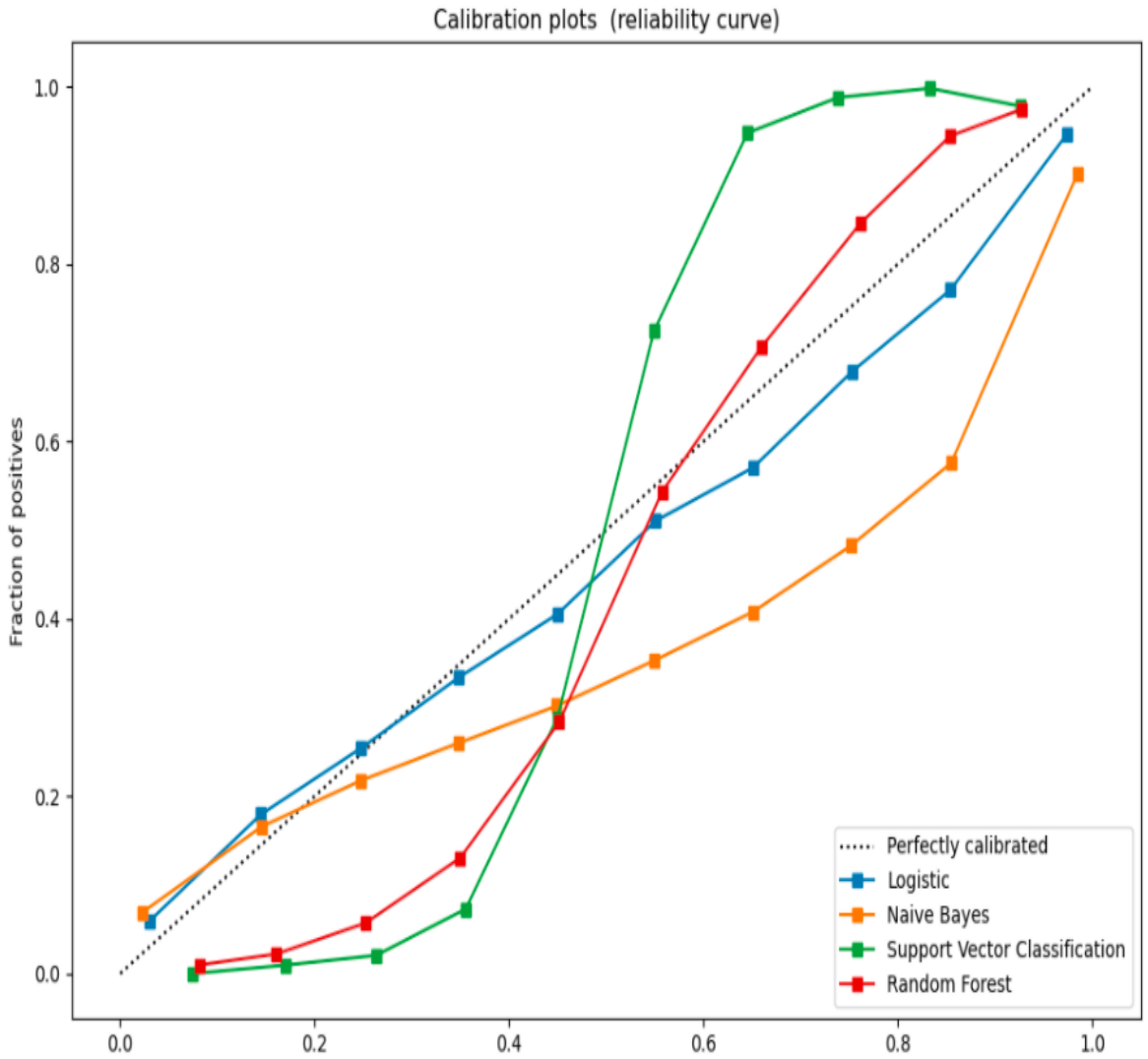
Fuente: https://bookdown.org/content/2031/ensambladores-random-forest-parte-i.html

- sklearn.ensemble.RandomForestClassifier
- sklearn.ensemble.RandomForestRegressor
- Feature importances with a forest of trees

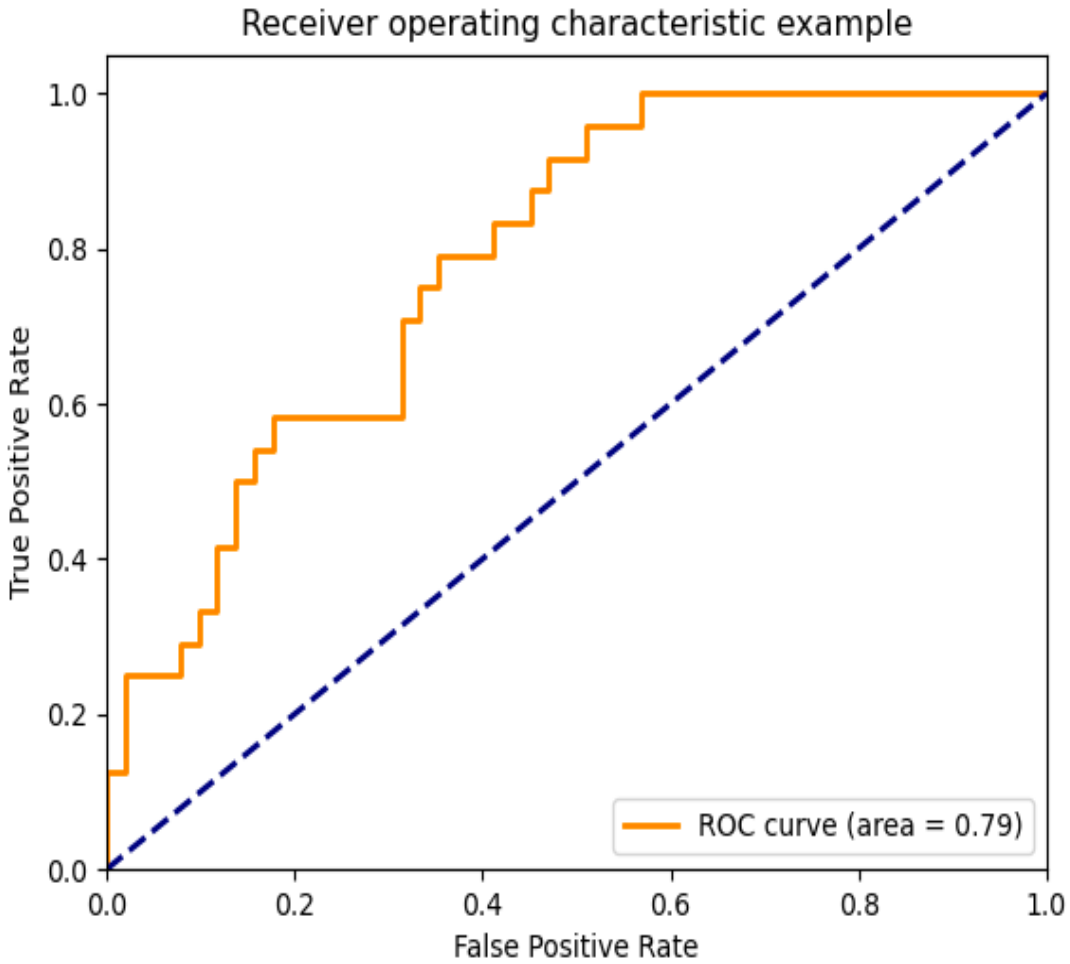
- sklearn.ensemble.RandomForestClassifier.apply
- sklearn.ensemble.RandomForestClassifier.decision_path
- sklearn.ensemble.RandomForestClassifier.feature_importances_
- sklearn.ensemble.RandomForestClassifier.fit
- sklearn.ensemble.RandomForestClassifier.get_params
- sklearn.ensemble.RandomForestClassifier.predict
- sklearn.ensemble.RandomForestClassifier.predict_log_proba
- sklearn.ensemble.RandomForestClassifier.predict_proba
- sklearn.ensemble.RandomForestClassifier.score
- sklearn.ensemble.RandomForestClassifier.set_params
- sklearn.ensemble.RandomForestRegressor.apply
- sklearn.ensemble.RandomForestRegressor.decision_path
- sklearn.ensemble.RandomForestRegressor.feature_importances_
- sklearn.ensemble.RandomForestRegressor.fit
- sklearn.ensemble.RandomForestRegressor.get_params
- sklearn.ensemble.RandomForestRegressor.predict
- sklearn.ensemble.RandomForestRegressor.score
- sklearn.ensemble.RandomForestRegressor.set_params

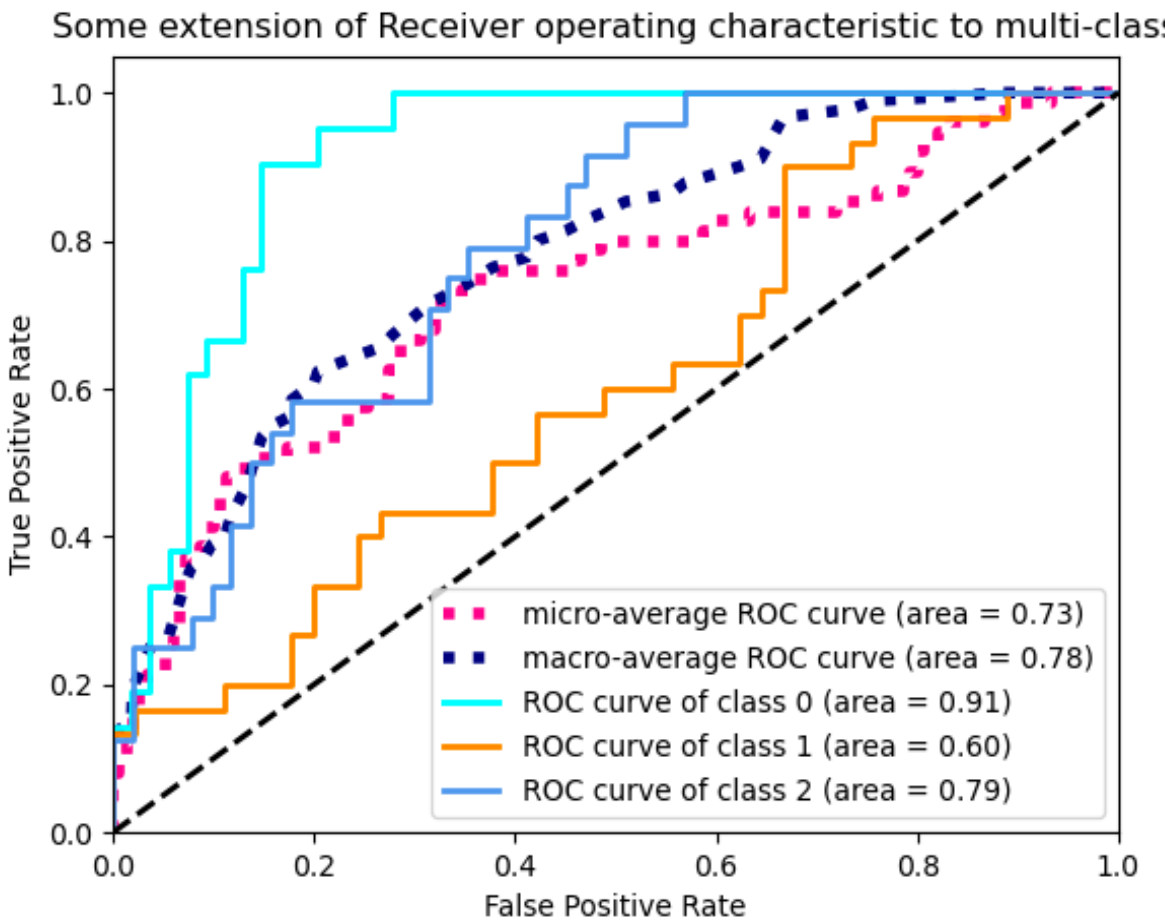
https://scikit-learn.org/stable/modules/ensemble.html#forest

https://scikit-learn.org/stable/search.html?q=random+forest



https://scikit-learn.org/stable/auto_examples/calibration/plot_compare_calibration.html#sphx-glr-auto-examples-calibration-plot-compare-calibration-py





https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classificatio

sklearn.tree: Decision Trees

The sklearn.tree module includes decision tree-based models for classification and regression.

User guide: See the Decision Trees section for further details.

tree.DecisionTreeClassifier(* [, criterion, ...])	A decision tree classifier.
tree.DecisionTreeRegressor(* [, criterion, ...])	A decision tree regressor.
tree.ExtraTreeClassifier(* [, criterion, ...])	An extremely randomized tree classifier.
tree.ExtraTreeRegressor(* [, criterion, ...])	An extremely randomized tree regressor.

tree.export_graphviz(decision_tree[, ...])	Export a decision tree in DOT format.
tree.export_text(decision_tree, *[, ...])	Build a text report showing the rules of a decision tree.

Plotting

tree.plot_tree(decision_tree, * [, ...])	Plot a decision tree.

sklearn.utils: Utilities

https://scikit-learn.org/dev/modules/classes.html#module-sklearn.tree

sklearn.tree.DecisionTreeClassifier

clase `sklearn.tree.` **`DecisionTreeClassifier`**(**, criterio = 'gini' , divisor = 'mejor' , max_depth = Ninguno , min_samples_split = 2 , min_samples_leaf = 1 , min_weight_fraction_leaf = 0,0 , max_features = Ninguno , random_state = Ninguno , max_leaf_nodes = Ninguno , min_impurity_decrease = 0,0 , min_impurity_split = Ninguno, class_weight = Ninguno , ccp_alpha = 0.0*) [fuente]

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#

Parámetros:

criterio : *{"gini", "entropía"}, predeterminado = "gini"*

La función para medir la calidad de una división. Los criterios admitidos son "gini" para la impureza de Gini y "entropía" para la ganancia de información.

divisor : *{"mejor", "aleatorio"}, predeterminado = "mejor"*

La estrategia utilizada para elegir la división en cada nodo. Las estrategias admitidas son "mejores" para elegir la mejor división y "aleatorias" para elegir la mejor división aleatoria.

max_depth : *int, predeterminado = Ninguno*

La profundidad máxima del árbol. Si es None, los nodos se expanden hasta que todas las hojas sean puras o hasta que todas las hojas contengan menos de min_samples_split muestras.

min_samples_split : *int o float, predeterminado = 2*

El número mínimo de muestras necesarias para dividir un nodo interno:

- Si es int, entonces considérelo `min_samples_split` como el número mínimo.
- Si es flotante, entonces `min_samples_split` es una fracción y es el número mínimo de muestras para cada división. `ceil(min_samples_split * n_samples)`

Modificado en la versión 0.18: Se agregaron valores flotantes para fracciones.

https://scikit-

min_weight_fraction_leaf : ***float, predeterminado = 0.0***

La fracción ponderada mínima de la suma total de pesos (de todas las muestras de entrada) que se requiere para estar en un nodo hoja. Las muestras tienen el mismo peso cuando no se proporciona sample_weight.

max_features : ***int, float o {"auto", "sqrt", "log2"}, predeterminado = Ninguno***

La cantidad de características a considerar al buscar la mejor división:

- Si es int, entonces considere las `max_features` características en cada división.
- Si es flotante, entonces `max_features` es una fracción y las características se consideran en cada división. `int(max_features * n_features)`
- Si es "automático", entonces `max_features=sqrt(n_features)`.
- Si es "sqrt", entonces `max_features=sqrt(n_features)`.
- Si es "log2", entonces `max_features=log2(n_features)`.
- Si es Ninguno, entonces `max_features=n_features`.

Nota: la búsqueda de una división no se detiene hasta que se encuentra al menos una partición válida de las muestras de nodo, incluso si requiere inspeccionar de manera efectiva más de `max_features` características.

random_state : ***int, instancia de RandomState o Ninguno, predeterminado = Ninguno***

Controla la aleatoriedad del estimador. Las características siempre se permutan aleatoriamente en cada división, incluso si `splitter` se establece en `"best"`. Cuándo , el algoritmo seleccionará al azar en cada división antes de encontrar la mejor división entre ellas. Pero la mejor división encontrada puede variar entre diferentes ejecuciones, incluso si . Ese es el caso, si la mejora del criterio es idéntica para varias divisiones y una división debe seleccionarse al azar. Para obtener un comportamiento determinista durante el ajuste, debe fijarse a un número entero. Consulte el glosario para obtener más detalles. `max_features <`

https://scikit-

número entero. Consulte el glosario para obtener más detalles.`max_features <` `n_features` `max_features` `max_features=n_features` `random_state`

max_leaf_nodes : *int, predeterminado = Ninguno*

Cultiva un árbol `max_leaf_nodes` de la mejor manera primero. Los mejores nodos se definen como una reducción relativa de la impureza. Si es Ninguno, entonces un número ilimitado de nodos hoja.

min_impurity_decrease : *float, predeterminado = 0.0*

Un nodo se dividirá si esta división induce una disminución de la impureza mayor o igual a este valor.

La ecuación ponderada de disminución de impurezas es la siguiente:

```
N_t / N * (impurity - N_t_R / N_t * right_impurity
                    - N_t_L / N_t * left_impurity)
```

donde `N` es el número total de muestras, `N_t` es el número de muestras en el nodo actual, `N_t_L` es el número de muestras en el hijo izquierdo y `N_t_R` es el número de muestras en el hijo derecho.

`N`, `N_t`, `N_t_R` Y `N_t_L` todos se refieren a la suma ponderada, si `sample_weight` se pasa.

Nuevo en la versión 0.19.

https://scikit-learn.org/dev/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier

min_impurity_split : *float, predeterminado = 0*

Umbral de parada temprana en el crecimiento de los árboles. Un nodo se dividirá si su impureza está por encima del umbral; de lo contrario, es una hoja.

> *En* `min_impurity_split` desuso *desde la versión 0.19:* ha quedado en desuso a favor de `min_impurity_decrease` 0.19. El valor predeterminado de `min_impurity_split` ha cambiado de 1e-7 a 0 en 0.23 y se eliminará en 1.0 (cambio de nombre de 0.25). Úselo en su `min_impurity_decrease` lugar.

class_weight : *dict, lista de dict o "equilibrado", predeterminado = Ninguno*

Pesos asociados a clases en el formulario . Si es Ninguno, se supone que todas las clases tienen el peso uno. Para problemas de múltiples salidas, se puede proporcionar una lista de dictados en el mismo orden que las columnas de y. `{class_label: weight}`

Tenga en cuenta que para múltiples salidas (incluidas múltiples etiquetas), los pesos deben definirse para cada clase de cada columna en su propio dictado. Por ejemplo, para las clasificaciones de etiquetas múltiples de cuatro clases, las ponderaciones deben ser [{0: 1, 1: 1}, {0: 1, 1: 5}, {0: 1, 1: 1}, {0: 1, 1: 1}] en lugar de [{1: 1}, {2: 5}, {3: 1}, {4: 1}].

El modo "balanceado" utiliza los valores de y para ajustar automáticamente los pesos de forma inversamente proporcional a las frecuencias de clase en los datos de entrada como `n_samples / (n_classes * np.bincount(y))`

Para múltiples salidas, se multiplicarán los pesos de cada columna de y.

Tenga en cuenta que estos pesos se multiplicarán por sample_weight (pasado a través del método de ajuste) si se especifica sample_weight.

https://scikit-learn.org/dev/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeCla

ccp_alpha flotante : *no negativo, predeterminado = 0.0*

Parámetro de complejidad utilizado para la poda de costo mínimo-complejidad. Se elegirá el subárbol con la complejidad de costo más grande que sea más pequeño `ccp_alpha`. De forma predeterminada, no se realiza ninguna poda. Consulte Poda de costo mínimo y complejidad para obtener más detalles.

Nuevo en la versión 0.22.

Atributos:

classes_ : *ndarray de forma (n_classes,) o lista de ndarray*

Las etiquetas de clases (problema de salida única) o una lista de matrices de etiquetas de clase (problema de salida múltiple).

`feature_importances_` : *ndarray de forma (n_features,)*

Devuelve la importancia de la función.

max_features_ : *int*

El valor inferido de max_features.

n_classes_ : *int o lista de int*

El número de clases (para problemas de salida única) o una lista que contiene el número de clases para cada salida (para problemas de salida múltiple).

n_features_ : *int*

El número de funciones cuando `fit` se realiza.

n_outputs_ : *int*

El número de salidas cuando `fit` se realiza.

tree_ : *Instancia de árbol*

El objeto Tree subyacente. Consulte los `help(sklearn.tree._tree.Tree)` atributos del objeto Árbol y Comprensión de la estructura del árbol de decisiones para conocer el uso básico de estos atributos.

https://scikit-

Ejemplos de

```
>>> 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.      ])
```

Métodos

apply(X [, check_input])	Devuelve el índice de la hoja con el que se predice cada muestra.
cost_complexity_pruning_path(X, y [,...])	Calcule la ruta de poda durante la poda de costo mínimo y complejidad.
decision_path(X [, check_input])	Devuelve la ruta de decisión en el árbol.
fit(X, y [, sample_weight, check_input,...])	Construya un clasificador de árbol de decisión a partir del conjunto de entrenamiento (X, y).
get_depth()	Devuelve la profundidad del árbol de decisiones.
get_n_leaves()	Devuelve el número de hojas del árbol de decisiones.
get_params([profundo])	Obtenga parámetros para este estimador.
predict(X [, check_input])	Predecir la clase o el valor de regresión para X.
predict_log_proba(X)	Predecir las probabilidades logarítmicas de clase de las muestras de entrada X.
predict_proba(X [, check_input])	Predecir las probabilidades de clase de las muestras de entrada X.

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#

Y más!!!!

sklearn.ensemble.RandomForestRegressor

class `sklearn.ensemble.`**RandomForestRegressor**(*n_estimators=100, *, criterion='mse', 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, min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, ccp_alpha=0.0, max_samples=None*) [source]

A random forest regressor.

A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the `max_samples` parameter if `bootstrap=True` (default), otherwise the whole dataset is used to build each tree.

Read more in the User Guide.

Parameters:

n_estimators : *int, default=100*

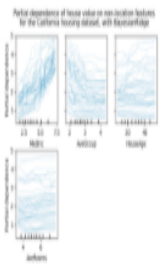
The number of trees in the forest.

Changed in version 0.22: The default value of `n_estimators` changed from 10 to 100 in 0.22.

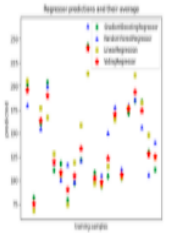
criterion : *{"mse", "mae"}, default="mse"*

The function to measure the quality of a split. Supported criteria are "mse" for the mean squared error, which is equal to variance reduction as feature selection criterion, and "mae" for the mean absolute error

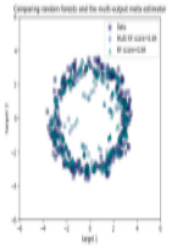
Examples using `sklearn.ensemble.RandomForestRegressor`



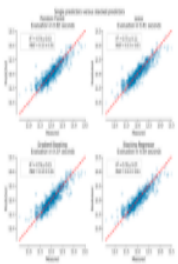
Release Highlights for scikit-learn 0.24



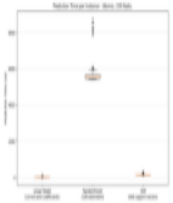
Plot individual and voting regression predictions



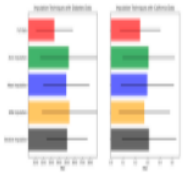
Comparing random forests and the multi-output meta estimator



Combine predictors using stacking



Prediction Latency



https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html?highlight=random%20forest#sklearn.ensemble.RandomForestRegressor.predict

sklearn.ensemble.RandomForestClassifier

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, min_impurity_split=None, 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*) [source]

A random forest classifier.

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the `max_samples` parameter if `bootstrap=True` (default), otherwise the whole dataset is used to build each tree.

Read more in the User Guide.

Parameters:

n_estimators : *int, default=100*

The number of trees in the forest.

Changed in version 0.22: The default value of `n_estimators` changed from 10 to 100 in 0.22.

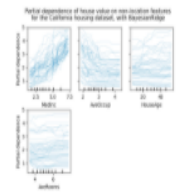
criterion : *{"gini", "entropy"}, default="gini"*

The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" fo the information gain. Note: this parameter is tree-specific.

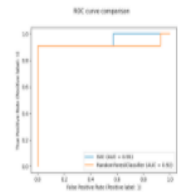
max_depth : *int, default=None*

The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves

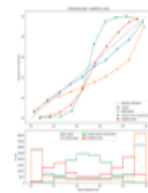
Examples using `sklearn.ensemble.RandomForestClassifier`



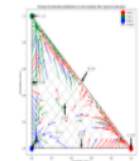
Release Highlights for scikit-learn 0.24



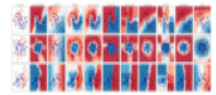
Release Highlights for scikit-learn 0.22



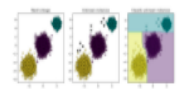
Comparison of Calibration of Classifiers



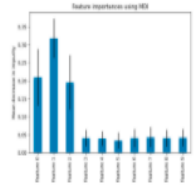
Probability Calibration for 3-class classification



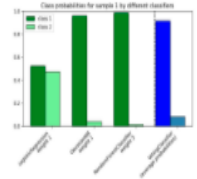
Classifier comparison



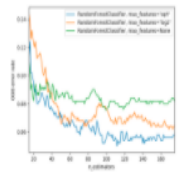
Inductive Clustering



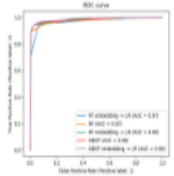
Feature importances with a forest of trees



Plot class probabilities calculated by the VotingClassifier



OOB Errors for Random Forests



Feature transformations with ensembles of trees

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html?highlight=random%20forest#sklearn.ense

¿PREGUNTAS?

¡GRACIAS!