Curso: Técnicas de Aprendizaje Automático – Machine Learning

Tema 4: Arboles y Bosques Aleatorios

Sonia I. Mariño, Rafael Perez, Leonardo Gomez Chavez

FaCENA - UNNE - 2023

Tipología de Arboles de Decisión

Algoritmos basados en árboles para el aprendizaje automático

- Árboles de Decisiones (Decision Trees)
- Bosques Aleatorios (Random Forest)
- Aumento de Gradiente (Gradient Boosting)
- Bagging (Bootstrap Aggregation)

Arboles de decisión (AD)

- AD en ML e IA, introducción
- Técnica de AA / ML supervisado
- Aplicable en problemas de:
  - Clasificación
    - Variable dependiente es categórica
    - El valor en el nodo terminal, asume la *moda* de las observaciones del conjunto de entrenamiento que corresponde en esa región
  - Regresión
    - Variable dependiente es continua
    - El valor en el nodo terminal, asume la *media* de las observaciones en esa región

Definiciones

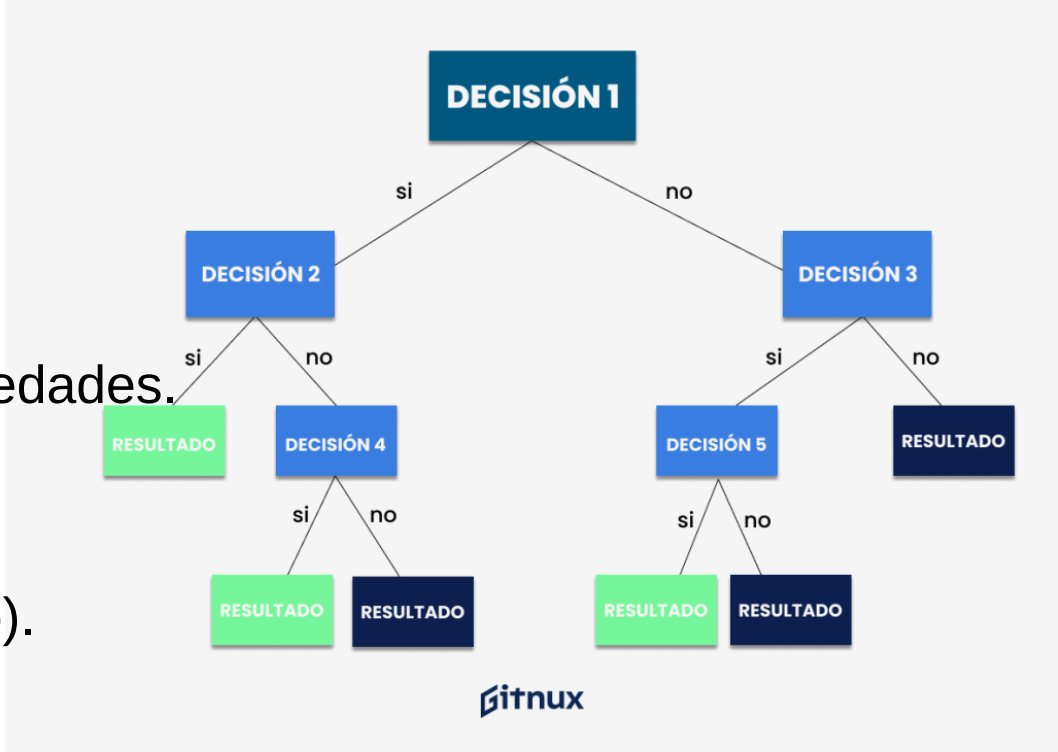
Árbol de Decisión (AD) - Algoritmo cuya finalidad es reconocer la existencia de relaciones en un determinado conjunto de datos por medio de procesos que imitan el funcionamiento del cerebro humano [1].

Random Forest (Breiman, 2001). Técnica de aprendizaje supervisado que genera múltiples árboles de decisión sobre un conjunto de datos de entrenamiento. Los resultados obtenidos se combinan a fin de obtener un modelo único más robusto en comparación con los resultados de cada árbol por separado (Lizares, 2017). Cada árbol se obtiene mediante un proceso de dos etapas [2]

[1] mencionado en Angélica Villón L., 2021 Aplicación de técnicas de minería de datos para predecir el desempeño académico de los estudiantes de la escuela, https://incyt.upse.edu.ec/ciencia/revistas/index.php/rctu/article/view/637/530

[2] mencionado en Espinosa-Zúñiga, Javier Jesús. Aplicación de algoritmos Random Forest y XGBoost en una base de solicitudes de tarjetas de crédito. Ingeniería, investigación y tecnología, 21(3), 00002. Epub 02 de diciembre de 2020.https://doi.org/10.22201/fi.25940732e.2020.21.3.022

Arboles de decisión (AD)



Elementos de un AD

- Entrada: Objetos caracterizables mediante propiedades.
- Salida:
  - En árboles de decisión: una decisión (sí o no).
  - En árboles de clasificación: una clase.
- Conjunto de reglas.

```
|--- petal width (cm) <= 0.80
|   |--- class: 0
|--- petal width (cm) >  0.80
|   |--- petal width (cm) <= 1.75
|   |   |--- class: 1
|   |--- petal width (cm) >  1.75
|   |   |--- class: 2
```

Nodos:

- intermedios o ramas, representan decisiones
- finales u hojas, brindan una predicción que conduce al objetivo

Arboles de decisión (AD). Ventajas

- **Método no paramétrico.**
  - AD carecen de suposiciones sobre la distribución del espacio y la estructura del clasificador.
- **Restricciones en tipo de datos.**
  - Puede aplicarse en variables numéricas y categóricas.
- **Limpieza de datos**
  - Requiere menos limpieza de datos en comparación con otras técnicas.
  - Carece de influencia de valores atípicos y faltantes de datos.
- .**Útil en la exploración de datos.**
  - Permite identificar las variables más significativas y relación entre dos o más.
  - Facilita crear nuevas variables o características para predecir la variable objetivo.

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

Arboles de decisión (AD). Ventajas

Alta legibilidad y comprensión de la representación / resultados

- Proporciona un resultado / salida de fácil comprensión / interpretación

Tiempo de cómputo off-line: rápido.

- algoritmos son simples

Tiempo de cómputo on-line: muy rápido.

- Clasificar un nuevo ejemplo es recorrer el árbol hasta alcanzar un nodo hoja.

Robustez

- Comportamiento robusto con ejemplos de entrenamiento con ruido

Arboles de decisión (AD). Limitaciones

Sobreajuste o Sobreentrenamiento

- Dificultad muy frecuente.
    - Solución: incluir restricciones en los parámetros del modelo y eliminar ramas en el análisis.
- Control con poda.
    - Por defecto, el valor nivel de confianza en 25%, podría brindar buenos resultados

Arboles de decisión (AD). Limitaciones

No aplicable en variables continuas.

- el AD pierde información cuando categoriza variables en diferentes categorías.

No aplicable con características muy dispersas

- **no recomendable si** datos de entrada son dispersos (Ej. características categóricas con una gran dimensión),
- Solución: preprocesar las características dispersas para generar estadísticas numéricas, o cambiar a un modelo lineal, más adecuado.

Técnicas de regularización, se deben aplicar para mejorar la generalización del modelo.

Algoritmos en AD

- ID3 (Iterative Dichotomiser 3)
- C4.5
- C5.0
- CART (Classification and Regression Trees) similar a C4.5

Algoritmos en AD

ID3 (Iterative Dichotomiser 3)

- Ross Quinlan, 1986
- Utiliza técnicas matemáticas y probabilísticas
- Introduce el concepto de ***entropía***, como medida de incertidumbre o de desorden
- Apoya toma de decisiones para determinar qué atributo debe seleccionarse.

Algoritmos en AD

C4.5

- Algoritmo sucesor de ID3
- Carece de restricciones sobre las características deben ser categóricas, definiendo dinámicamente un atributo discreto (basado en variables numéricas) que divide el valor del atributo continuo en un conjunto discreto de intervalos.
- C4.5 –a partir de los AD entrenados con ID3- brinda un conjunto de reglas *si-entonces*. Y, evalúa la precisión de cada regla para determinar el orden en el que deben aplicarse.
- La poda se realiza eliminando la condición previa de una regla si la precisión de la regla mejora sin ella.
- Aplican como criterio de división: ***information gain***

Algoritmos en AD

C5.0

- Ultima versión, licencia propietaria
- Mejora en precisión, utiliza menos memoria y construye menores conjunto de reglas que

  C4.5.0
- Aplican como criterio de división: ***information gain***

Algoritmos en AD

CART (Classification and Regression Trees)

- Diferencia de C4.5, admite variables objetivo numéricas (regresión) y no calcula conjuntos de reglas.
- Genera automáticamente las particiones, cada una con las agrupaciones más homogéneas posible.
- Los AD binarios se construyen utilizando la característica y el umbral que producen la mayor ganancia de información en cada nodo.
- Desventaja: CART es un algoritmo codicioso: para calcular el umbral óptimo en cada partición evalúa todas las posibles opciones. Es decir, a mayor número de características y más datos, se requiere más tiempo de entrenamiento.
- *scikit-learn* utiliza una versión optimizada del algoritmo CART. no acepta variables categóricas

AD, cómo se define / decide la ramificación

- En AD, se debe decidir donde dividir estratégicamente las ramas, influye en la precisión de la predicción.
- El objetivo es encontrar nodos más puros/homogéneos posible.
  - Crear subnodos mejora la homogeneidad/pureza de los subnodos resultantes, en referencia **a la variable objetivo.**
  - En general, los algoritmos prueban la división con todas las variables, y elige aquella que produce subnodos más homogéneos.

AD, cómo se define / decide la ramificación

Se disponen de distintas medidas como criterio de ramificación o selección de las divisiones en un AD.

- árboles de clasificación
- árboles de regresión

Algunos algoritmos usualmente aplicados para decidir la ramificación son:

- Indice Gini,
- Ganancia de la información, information gain: cross entropy,
- Chi Cuadrado, [los AD se denominan CHAID - chi-square automatic interaction detector]
- Classification error rate

AD, cómo se define la ramificación

Indice Gini

- Cuantifica la varianza total en el conjunto de las K clases del nodo m, es decir, mide la pureza del nodo.

$$G_m = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

Medida de homogeneidad

- Mide el grado de “impureza” de un nodo
- Si índice Gini = 0, nodos puros (con datos que pertenecen a una sola categoría) $\hat{p}_{mk}$
- Si índice Gini > 0 y <=1, nodos impuros (con datos que pertenecen a más de una categoría)

Algoritmo CART (Classification and Regression Trees) aplica como criterio de división.

AD, cómo se define la ramificación

Information Gain: Cross Entropy

- Entropía, medida que cuantifica el desorden [aleatoriedad] de un sistema.
- En el caso de los nodos, el desorden se corresponde con la impureza.
    - Nodo puro, presenta observaciones de una clase, y ***entropía*** = 0

$$D = -\sum_{k=1}^{K} \hat{p}_{mk}\, log(\hat{p}_{mk})$$

- Los algoritmos C4.5 y C5.0 emplean *information gain* como criterio de división.

AD, cómo se define la ramificación

CHI-SQUARE

Se identifica si existe una diferencia significativa entre los nodos hijos y el nodo parental. Es decir, si hay evidencias de que la división consigue una mejora.

Se aplica un test estadístico chi-square goodness of fit empleando como distribución esperada H0 la frecuencia de cada clase en el nodo parental.

A mayor valor del estadístico $\chi 2$, mayor la evidencia estadística de que existe una diferencia.

$$\chi^2 = \sum_k \frac{(\text{observado}_k - \text{esperado}_k)^2}{\text{esperado}_k}$$

AD construidos con este criterio de división, se denominan CHAID (Chi-square Automatic Interaction Detector).

AD, cómo se define la ramificación. Proceso

Se aplica el mismo proceso de construcción, varia la medida utilizada como criterio de selección de las divisiones:

- Para cada posible división se calcula el valor de la medida en cada uno de los 2 nodos resultantes.
- Se suman los 2 valores, ponderando cada uno por la fracción de observaciones que contiene cada nodo.

$$\frac{nro\ Obs\ nodo\ izq}{nro\ total\ Obs} * pureza\ izq\ + \frac{nro\ Obs\ nodo\ der}{nro\ total\ Obs} * pureza\ der$$

- La división con menor o mayor valor (dependiendo de la medida empleada) se selecciona como punto de corte óptimo.

Learning, *Gini index* y *cross-entropy* más adecuados que *classification error rate* debido a su mayor sensibilidad a la homogeneidad de los nodos.
En proceso de *pruning*, los 3 métodos son adecuados, para lograr la máxima precisión en las predicciones, aplicar *classification error rate*.

AD, cómo se define la ramificación

Residual Sum of Squares (RSS).

Objetivo: encontrar las J regiones (R1 ,..., Rj) que minimizan el Residual Sum of Squares (RSS) total:

$$RSS = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2,$$

$\hat{y}_{R_j}$ media de la variable respuesta en la región Rj

Proceso.

1. Iniciar donde todas las observaciones pertenecen a la misma región.

2. Identificar todos los posibles puntos de cortes ***s*** para cada uno de los predictores (X1, X2,..., Xp).
   - Si predictor es cualitativo, los posibles puntos de corte son cada uno de sus niveles.
   - Si predictor es continuo, ordenar de menor a mayor sus valores, el punto intermedio entre cada par de valores se emplea como punto de corte.

3. Calcular *RSS* total, con cada posible división identificada en el paso 2.

   Elegir el predictor Xj y el punto de corte ***S*** que permite nodos más homogéneos

4. Repetir iterativamente para cada una de las regiones creadas en la iteración anterior hasta que se alcanza alguna norma de *parada:* profundidad máxima, regiones con no menos de *n* observaciones, máximo de nodos terminales, incorporación de nodos si reduce error en al menos un % mínimo.

Métricas de evaluación. Clasificación

		Predicted class: P	Predicted class: N
Actual Class	P	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

Evalúan el rendimiento de un modelo que asigna observaciones a ciertas categorías.

La Matriz de confusion permite calcular:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Accuracy: proporción de observaciones que el modelo clasifica correctamente.

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

Precisión: proporción de observaciones clasificadas como positivas que realmente son positivas.

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

Recall: proporción de observaciones positivas que el modelo clasifica correctamente.

$$\text{F1-Score} = \frac{2 \times \text{Precisión} \times \text{Recall}}{\text{Precisión} + \text{Recall}}$$

F1-Score: combina la precisión y el recall, dando más peso a los valores bajos.

AUC: Es el área bajo la curva ROC, que representa la relación entre la tasa de verdaderos positivos y la tasa de falsos positivos para diferentes umbrales de clasificación. Valor alto de AUC, indica que el modelo tiene una buena capacidad de discriminar entre las clases.

TP son los verdaderos positivos, TN son los verdaderos negativos,
FP son los falsos positivos y FN son los falsos negativos.

Métricas de evaluación. Regresión

- RMSE: Es la raíz cuadrada del MSE.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

- MAPE: Es el promedio de los errores porcentuales absolutos.

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100$$

- R cuadrado: Es el coeficiente de determinación que indica la proporción de la variación de la variable dependiente que se explica por la variable independiente.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

- MSE: Es el promedio de los errores al cuadrado.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- MAE: Es el promedio de los errores absolutos.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Donde n es el número de observaciones, y_i es el valor real u observado, $\hat{y}_i$ es el valor predicho por el modelo y $\bar{y}$ es el promedio de los valores reales.

AD en Phyton

`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.

https://scikit-learn.org/stable/modules/classes.html#module-sklearn.model_selection

Clasificación

- Técnica de aprendizaje automático considerada técnica predictiva del valor de algún atributo, llamado etiqueta, de un determinado conjunto de datos (Zhao et al., 2021).
- Técnica de aprendizaje supervisado, utiliza
  - un conjunto de entrenamiento para construir el modelo de aprendizaje.
  - un conjunto de datos de prueba para verificar la consistencia del modelo de aprendizaje desarrollado (Han et al., 2012; Sarker, 2021).

AD para soluciones de Clasificación

- Problemas de aprendizaje supervisados
- Tipología de Clasificación:
  - Clasificación binaria: 2 etiquetas
  - Clasificación multiclase; n etiquetas
  - Clasificación de etiquetas multiclase: clases estructuradas jerárquicamente, dato puede pertenecer a más de una clase
- Momentos asociados a un AD
  - Creación del AD, se elige una medida de división {Ej: indice Gini}
  - Inferencia en AD
    - El AD, agrupa las observaciones de entrenamiento en los nodos terminales.
    - Predecir una nueva observación, recorrer el árbol en función del valor de sus predictores hasta llegar a uno de los nodos terminales. Aplicar la moda de la variable respuesta como valor de predicción, que representa a la clase más frecuente del nodo. Se puede incluir el porcentaje de cada clase en el nodo terminal, lo que aporta información sobre la confianza de la predicción.

AD en Phyton

DecisionTreeClassifier, parámetros y sus posibles valores

sklearn.tree.DecisionTreeClassifier

class sklearn.tree.DecisionTreeClassifier(*, criterion ={“**gini**”, “entropy”, “log_loss”}, splitter={'**best**‘, ‘random'},

max_depth={None, }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,

class_weight=None, ccp_alpha=0.0)[source]

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

Mejora en AD

Algunos parámetros que se pueden ajustar para mejorar la salida del modelo (Scikit Learn, 2019).

1. **criterion**- La medición del uso, como las impurezas de Gini
2. **class_weight**- Ninguno, lo que significa que todos los pesos de clase son 1
3. **max_depth**- 3; ramas. Cuando "ninguno" significa que el nodo se desarrollará hasta que todas las hojas sean homogéneas
4. **max_features**- ninguno; al determinar la segmentación del nodo, considere todas las características o variables independientes
5. **max_leaf_nodes** — None;
6. **min_impurity_decrease**- 0.0; si la división asegura que la imputación se reduzca o sea igual a cero, el nodo se divide
7. **min_impurity_split** — None;
8. **min_samples_leaf**- 1; muestra mínima requerida para una hoja
9. **min_samples_split**- 2; si min_samples_leaf = 1, significa que el nodo derecho y el nodo izquierdo deben tener una muestra. Es decir, el nodo principal o el nodo raíz deberían tener al menos dos muestras
10. **splitter**- "mejor"; para la selección de segmentación en cada nodo. Asegurar que todas las características se consideren al decidir dividir

AD, método

Pipeline, entrenamiento y visualización de AD.

- Paso 1: Importar los datos
- Paso 2: Limpiar los datos
- Paso 3: Crear los conjuntos de entrenamiento y test
- Paso 4: Construir el modelo
- Paso 5: Predecir utilizando el modelo
- Paso 6: Medir el rendimiento del modelo
- Paso 7: Ajustar los hiperparámetros

Entender IRIS

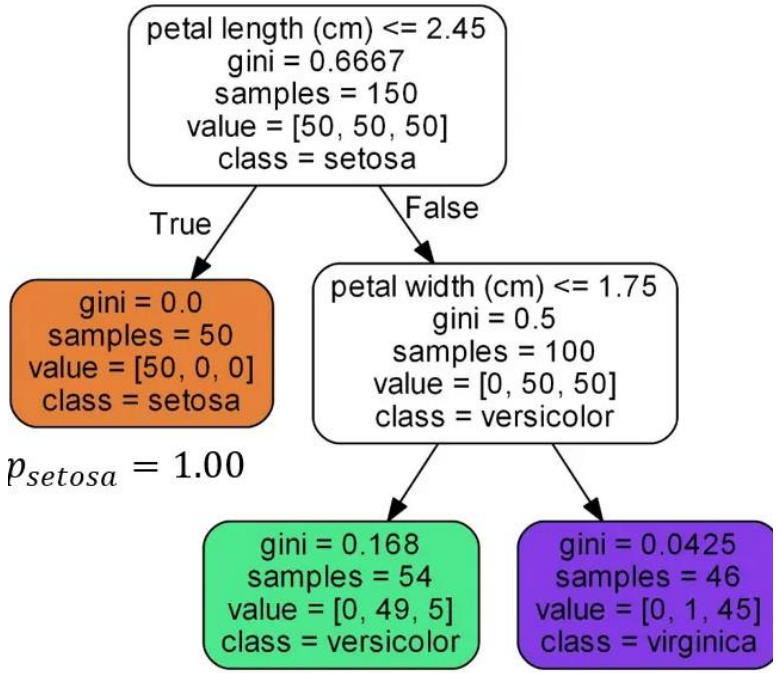
- Muestras o ejemplares: 150 flores del conjunto de datos Iris
- Clases balanceadas
- Variable objetivo: dispone de 3 valores. En el ejemplo, cada color representa a una clase.
  - Marrón para setosa, Verde para versicolor y Lila para virginica.
  - el color es más intenso según la clasificación es correcta
  - los nodos blancos, evidencia la falta de certeza

Tipos de nodo:

- Nodos de decisión: tienen una condición al principio y tienen más nodos debajo de ellos
- Nodos de predicción: no tienen ninguna condición ni nodos debajo de ellos. También se denominan «nodos hijo»

Información de cada nodo:

condición: si es un nodo donde se toma alguna decisión
gini: medida de impureza.
samples: número de muestras que satisfacen las condiciones necesarias para llegar a este nodo
value: cuántas muestras de cada clase llegan a este nodo
class: clase asignada a las muestras que llegan a este nodo



$p_{setosa} = 1.00$

Interpretación del AD

Si la longitud del pétalo es menos de 2.45 cm, flor iris pertenece a la variedad setosa.
Si *por el contrario*, la longitud del pétalo es mayor que 2.45 cm, evaluar al ancho del pétalo.
Si el ancho del pétalo es menor o igual a 1.75 cm, pertenece a la variedad versicolor con un 91% de probabilidad.
Si no, parece que sería virginica con un 98% de probabilidad.

$$gini = 1 - \sum_{k=1}^{n} p_c^2$$

Por ejemplo, para el caso del nodo donde la clasificación es *versicolor*, el cálculo sería el siguiente:

$$gini_{versicolor} = 1 - \sum_{k=1}^{n} p_c^2 = 1 - \left(\frac{0}{54}\right)^2 - \left(\frac{49}{54}\right)^2 - \left(\frac{5}{54}\right)^2 = 0.168$$

AD en Phyton . Ej Iris

DecisionTreeClassifier

```
import numpy as np
from matplotlib import pyplot as plt
from sklearn import tree
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
# Entrenar AD classifier
# Fit a DecisionTreeClassifier usando load_iris dataset.
iris = load_iris()
# información sobre del conjunto de datos iris
print(iris.DESCR)

X = iris.data
y = iris.target

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
clf = DecisionTreeClassifier(max_leaf_nodes=3, random_state=0)
clf.fit(X_train, y_train)
tree.plot_tree(clf)
```

https://colab.research.google.com/drive/1l25lOPseyFabMr1sL-DlnzQUR8S4K6FH?authuser=1#scrollTo=DDEu50oJp-Wm

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

AD en Phyton . Ej Iris

```
import numpy as np
from matplotlib import pyplot as plt
from sklearn import tree
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
# Entrenar AD classifier
# Fit a DecisionTreeClassifier usando load_iris dataset
iris = load_iris()
# información sobre del conjunto de datos iris
print(iris.DESCR)

X = iris.data
y = iris.target


X_train, X_test, y_train, y_test = train_test_split(X, y,
random_state=0)
clf = DecisionTreeClassifier(max_leaf_nodes=3, random_state=0)
clf.fit(X_train, y_train)
tree.plot_tree(clf)
```

```
:Attribute Information:
    - sepal length in cm
    - sepal width in cm
    - petal length in cm
    - petal width in cm
    - class:
            - Iris-Setosa
            - Iris-Versicolour
            - Iris-Virginica

:Summary Statistics:

============== ==== ==== ======= ===== ====================
                Min  Max   Mean    SD   Class Correlation
============== ==== ==== ======= ===== ====================
sepal length:   4.3  7.9   5.84   0.83    0.7826
sepal width:    2.0  4.4   3.05   0.43   -0.4194
petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)
============== ==== ==== ======= ===== ====================

:Missing Attribute Values: None
:Class Distribution: 33.3% for each of 3 classes.
:Creator: R.A. Fisher
:Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
:Date: July, 1988
```

https://colab.research.google.com/drive/1l25lOPseyFabMr1sL-DlnzQUR8S4K6FH?authuser=1#scrollTo=DDEu50oJp-Wm

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

AD en Phyton . Ej Iris

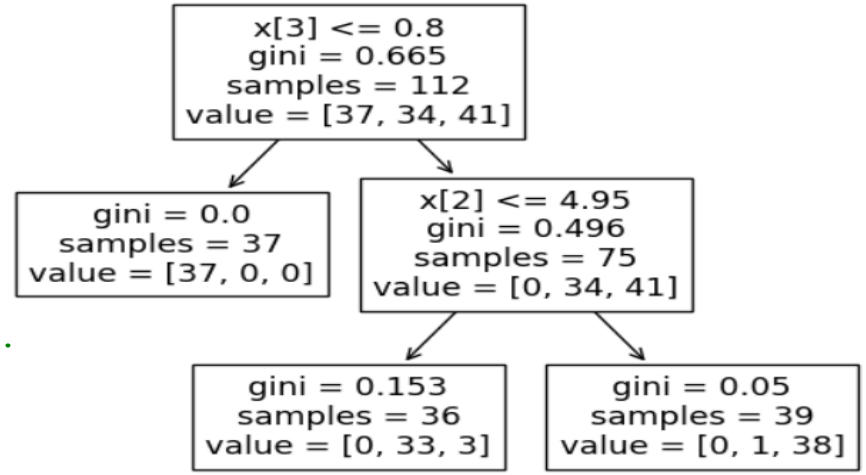
DecisionTreeClassifier

```
import numpy as np
from matplotlib import pyplot as plt
from sklearn import tree
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import export_text
# Entrenar AD classifier
# Fit a DecisionTreeClassifier usando load_iris dataset.
iris = load_iris()
# información sobre del conjunto de datos iris
print(iris.DESCR)

X = iris.data
y = iris.target


X_train, X_test, y_train, y_test = train_test_split(X, y,
random_state=0)
clf = DecisionTreeClassifier(max_leaf_nodes=3, random_state=0)
clf.fit(X_train, y_train)
tree.plot_tree(clf)
r = export_text(clf, feature_names=iris['feature_names'])
print(r)
```

```
Text(0.2, 0.5, 'gini = 0.0\nsamples = 37\nvalue = [37, 0, 0]'),
Text(0.6, 0.5, 'x[2] <= 4.95\ngini = 0.496\nsamples = 75\nvalue = [0, 34, 41]'),
Text(0.4, 0.16666666666666666, 'gini = 0.153\nsamples = 36\nvalue = [0, 33, 3]'),
Text(0.8, 0.16666666666666666, 'gini = 0.05\nsamples = 39\nvalue = [0, 1, 38]')]
```




```
|--- petal width (cm) <= 0.80
|   |--- class: 0
|--- petal width (cm) >  0.80
|   |--- petal length (cm) <= 4.95
|   |   |--- class: 1
|   |--- petal length (cm) >  4.95
|   |   |--- class: 2
```

https://colab.research.google.com/drive/1l25lOPseyFabMr1sL-DlnzQUR8S4K6FH?authuser=1#scrollTo=DDEu50oJp-Wm

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

DecisionTreeClassifier

AD en Phyton . Ej Iris

```
import numpy as np
from matplotlib import pyplot as plt
from sklearn import tree
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import export_text
from sklearn import metrics
from sklearn.metrics import accuracy_score

# Entrenar AD classifier
# Fit a DecisionTreeClassifier usando load_iris dataset.
iris = load_iris()
# información sobre del conjunto de datos iris
print(iris.DESCR)

X = iris.data
y = iris.target

X_train, X_test, y_train, y_test = train_test_split(X, y,
random_state=0)
clf = DecisionTreeClassifier(max_leaf_nodes=3, random_state=0)
clf.fit(X_train, y_train)
tree.plot_tree(clf)
r = export_text(clf, feature_names=iris['feature_names'])
print(r)
y_pred = clf.predict(X_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

https://colab.research.google.com/drive/1l25lOPseyFabMr1sL-DlnzQUR8S4K6FH?authuser=1#scrollTo=DDEu50oJp-Wm

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

AD en Phyton. Ej Iris, modifica hiperparametro `max_leaf_nodes`

DecisionTreeClassifier

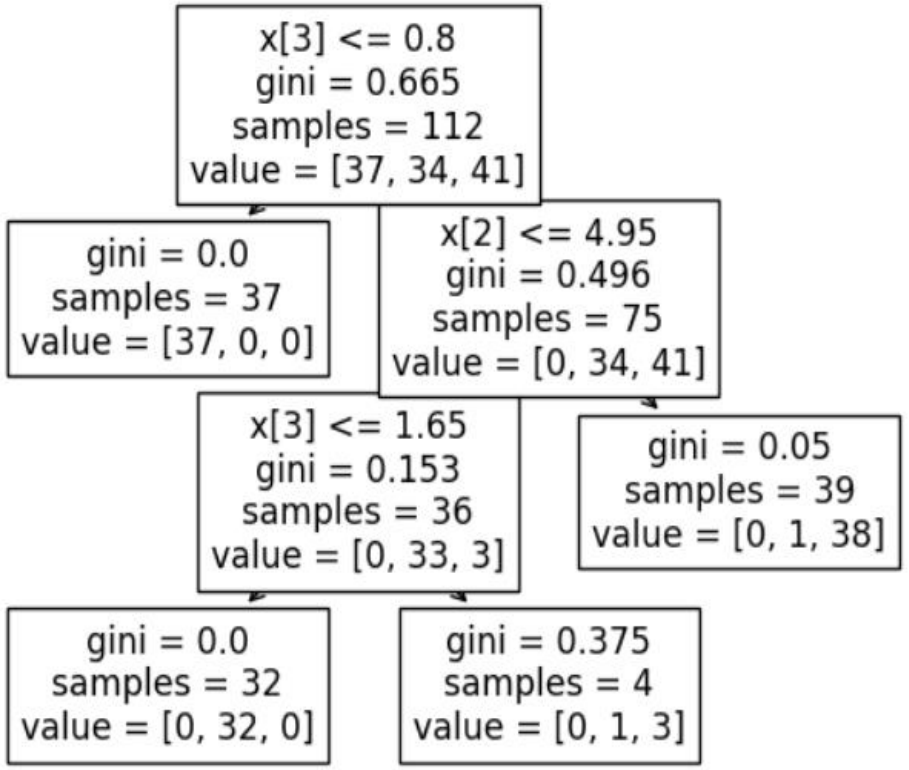
```
import numpy as np
from matplotlib import pyplot as plt
from sklearn import tree
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import export_text
# Entrenar AD classifier
# Fit a DecisionTreeClassifier usando load_iris dataset.
iris = load_iris()
# información sobre del conjunto de datos iris
print(iris.DESCR)

X = iris.data
y = iris.target


X_train, X_test, y_train, y_test = train_test_split(X, y,
random_state=0)
clf = DecisionTreeClassifier(max_leaf_nodes=4, random_state=0)
clf.fit(X_train, y_train)
tree.plot_tree(clf)
r = export_text(clf, feature_names=iris['feature_names'])
print(r)
```

https://colab.research.google.com/drive/1l25lOPseyFabMr1sL-DlnzQUR8S4K6FH?authuser=1#scrollTo=DDEu50oJp-Wm

```
|--- petal width (cm) <= 0.80
|   |--- class: 0
|--- petal width (cm) >  0.80
|   |--- petal length (cm) <= 4.95
|   |   |--- petal width (cm) <= 1.65
|   |   |   |--- class: 1
|   |   |--- petal width (cm) >  1.65
|   |   |   |--- class: 2
|   |--- petal length (cm) >  4.95
|   |   |--- class: 2
```



AD en Phyton. Ej Iris

Tarea

Un AD, como modelo de aprendizaje supervisados, presenta posibilidad de un sobreajuste. Por ello, se debe entrenar con diferentes configuraciones.

- Entre los parámetros establecidos para el modelo, están: la profundidad del árbol y el número mínimo de muestras necesarias para la división de cada nodo interno
- Realizar al menos 3 configuraciones, modificando «criterion» u otros hiperparámetros. Construir una tabla resumen.
- Analizar la mejor configuración a partir de una métrica (ej. accuracy)

AD en Regresión

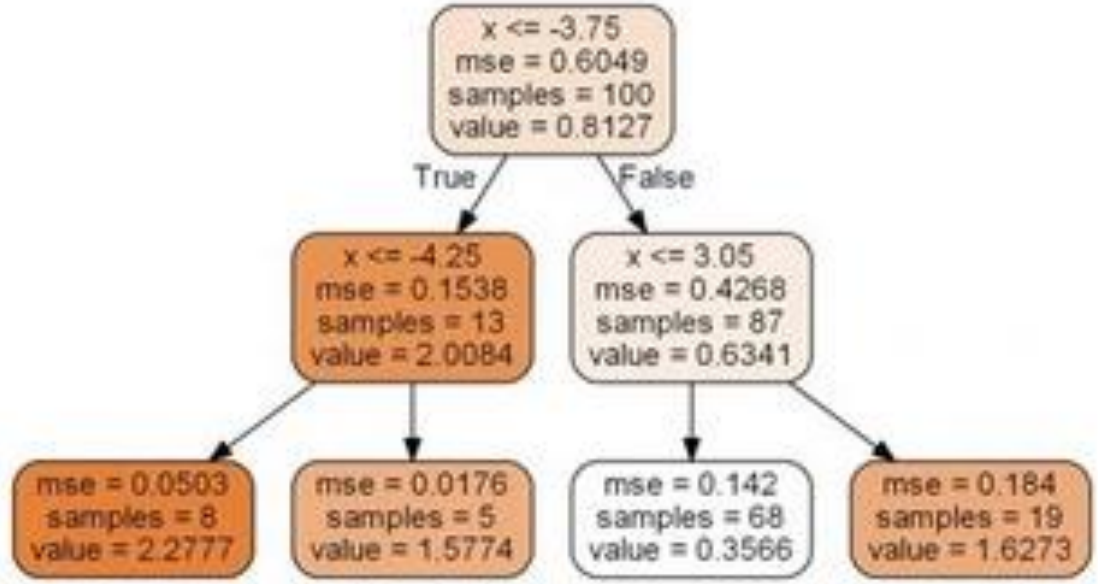
Código Python para entrenar y predecir con árboles de decisión para regresión

```
# importar las librerías necesarias
import numpy as np # NumPy para manipulación numérica
np.random.seed(42) # para hacer el código reproducible
from matplotlib import pyplot as plt
from sklearn.tree import DecisionTreeRegressor # árbol de decisión para regresión
#función y = 0.1x^2 + 0.2(Ruido Gaussiano)
def f(x):
 y = 0.1*np.square(x) + 0.2*np.random.randn(x.size)
 return y
# Generar los datos x, y con la función f(x)
x = np.arange(-5,5,0.1) # x = [-5, -4.9, -4.8, ... 4.8, 4.9, 5]
y = f(x)
# Crear y entrenar un árbol de decisión para regresión
tree = DecisionTreeRegressor(max_depth=2, random_state=42) # máxima profundidad 2
tree.fit(x.reshape(-1,1), y) # entrenar el árbol de regresión
tree.plot_tree(tree)
# Predecir qué valores de y, si x2 = [-0.7, 0.5, 2.3]
x2 = np.array([-0.7, 0.5, 2.3]).reshape(-1,1)
print( tree.predict(x2) )
# se obtiene como resultado:
```

AD en Regresión

En problemas de regresión se aplican como métricas:

- MAE
- MSE o error cuadrático medio.
- RMSE
- R2



Si se indica AD de profundidad 2,

La interpretación del AD sería: si el valor de x es menor que -4.25, predice 2.2777; si está en el intervalo (-4.25, -3.75] predice 1.5774); si está en el intervalo (-3.75, 3.05] predice 0.3566 y si es mayor que 3.05 predice 1.6273.

Mejora de AD

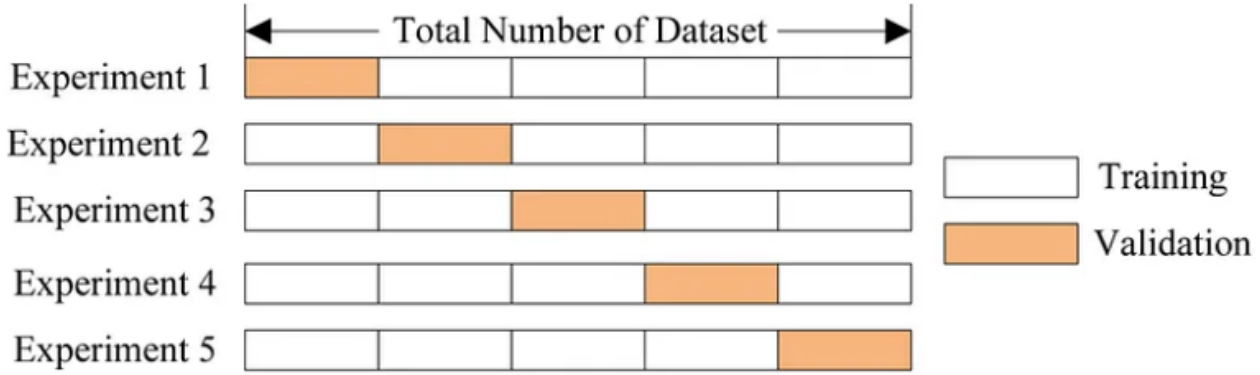
1. Error de Entrenamiento (Resubstitution)
2. Validación Cruzada (Cross Validation)
3. Control de profundidad (Leafiness)
4. Poda (Pruning)

Mejora de AD

Error de Entrenamiento

- Mide la diferencia entre la respuesta en los datos de entrenamiento y las predicciones del AD
- El error de entrenamiento es optimista con respecto a la clasificación de nuevos datos

Validación cruzada



5 Fold Cross Validation (Source)

La técnica de validación cruzada (CV) se aplica con el método más común, K-Fold CV.

En ML, se divide el conjunto de datos en subset de entrenamiento y subset de pruebas.

En K-Fold CV, se divide el subset de entrenamiento en K número de subconjuntos o pliegues.

Se ajusta iterativamente el modelo K veces, cada vez entrenando los datos en K-1 de los pliegues y evaluando en el K-ésimo pliegue (subset datos de validación).

Ej: si K = 5.

- 1era iteración, se entrenan los primeros 4 pliegues y se evalúa 5to.
- 2da iteración, se entrena: 1ra, 2da, 3ra y 5ta. Se evalúa en la 4ta.
- Se repite proceso, evaluando los restantes pliegues
- Finaliza, calculo del promedio del rendimiento en cada uno de los pliegues. Generar métricas de validación finales para el modelo.

Ej 1. Iris

Evaluar una puntuación mediante validación cruzada.

```
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.        , 0.93333333, 1.        , 0.93333333, 0.93333333,
       0.86666667, 0.93333333, 1.        , 1.        , 1.        ])
```

Matriz de puntuaciones del estimador para cada ejecución de la validación cruzada.

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html

Mejora de AD

Control de profundidad (Leafiness) y Poda (Pruning)

- Optimiza el AD
- modifica el número de hojas –reúne 2 en una- y aplica la frondosidad (leafiness)
- Sobreentrenamiento o sobreajuste: Se controla a través de una poda

Pre-poda y post-poda

El algoritmo CART

- Organizar ascendentemente los valores de la característica,
- Calcular el punto medio entre cada par de características consecutivas: “umbrales candidatos”.

1. Seleccionar el mejor de “umbrales candidatos”. Es decir, aquel que genere las particiones con la menor dispersión posible de la variable continua

2. Calcular el error cuadrático medio en cada región: [Si error cuadrático medio = 0, condición ideal, es decir, una dispersión nula. Mayor valor del error, mayor grado de dispersión.] Calcular la función de costo de cada umbral candidato y elegir aquel con el menor costo

3. Repetir los pasos 1 a 3 de forma iterativa, hasta cumplir con un criterio de parada.

Pre-poda y post-poda

Para evitar que el árbol crezca indefinidamente, criterio de parada y poda

algunas hojas presentan dispersión, si se continua la división, se podría caer en overfitting, el AD podría no responder a otros conjuntos de datos.

- pre-poda o mínimo número de datos por hoja, n = 2 y n = 4

Pre-poda y post-poda

Post-poda

- Alternativa de control de tamaño del árbol
- Consiste en entrenar el AD y aplicar post-poda, para eliminar algunas hojas sobrantes
- Algoritmo más usado: poda de complejidad de costos
- Utiliza un parámetro (alpha) que controla el nivel de poda:
- Alpha = 0 no se elimina ninguna hoja,
- Alpha, aumenta y se eliminarán más hojas, se controlará el tamaño del AD

Actividad Práctica

Tarea 2

- Un AD, como modelo de aprendizaje supervisados, presenta posibilidad de un sobreajuste. Por ello, se debe entrenar con diferentes configuraciones.
- Entre los parámetros establecidos para el modelo, están: la profundidad del árbol y el número mínimo de muestras necesarias para la división de cada nodo interno
- Realizar al menos 2 configuraciones y construir una tabla resumen.
- Analizar la mejor configuración a partir de una métrica (ej. accuracy)

Arboles de decisión. Aplicaciones

En este trabajo se presenta a los árboles de decisión como una técnica de aprendizaje automático para la clasificación de vacas como buenas productoras de leche a partir del uso de marcadores genéticos. La finalidad es realizar una selección de animales genéticamente superiores en menor tiempo y hacer más eficiente el proceso de reproducción asistida logrando con ello disminuir costos y aumentar ganancias en el sector lechero.

Los resultados de los experimentos realizados muestran hasta un 94.5% de precisión. Además, el algoritmo permitió la identificación del SNP más dominante para la clasificación, y el cromosoma que más influye en la predicción.

Palabras clave: Clasificación; árboles de decisión; producción lechera

Tabla 2. Precisión de la clasificación usando árboles de decisión.

Conjunto de datos	Precisión del conjunto de pruebas (%)	Núm. de nodos en el árbol resultante
a	93.6	67
b	91.8	117
c	91.8	107
d	90.9	93
e	90.9	95
f	93.6	93
g	90.9	85
h	94.5	85
i	94.5	97

CONCLUSIONES En este estudio se muestra el resultado de utilizar una técnica de ML para para la clasificación de vacas lecheras, específicamente se utilizan árboles de decisión. Los árboles de decisión además de identificar con muy alta precisión a las muestras dadas, identifica con éxito el SNP más importante al hacer la clasificación. Esto puede conducir a ahorros económicos pues solo se requiere genotipificar al cromosoma 14 para obtener muy buenos resultados. Aunque ya se sabía que éste es el cromosoma más relacionado con la producción lechera, no se sabía que era suficiente para determinar la clasificación. l algoritmo de árboles de decisión estudiado es capaz de gestionar de manera efectiva la información del genoma completo bovino lo que lo hace adecuado para la implementación de herramientas de predicción de rasgos económicos en la industria lechera.

Rodríguez Alcántar Edelmira. Árboles de decisión para clasificación de vacas lecheras usando información genética. Epistemus (Sonora) [revista en la Internet]. 2022 Dic [citado 2023 Sep 18] ; 16(33): 69-74. Disponible en: http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S2007-81962022000200069&lng=es. Epub 19-Mayo-2023. https://doi.org/10.36790/epistemus.v16i33.220.

Arboles de decisión. Aplicaciones

Es un algoritmo cuya finalidad es reconocer la existencia de relaciones en un determinado conjunto de datos por medio de procesos que imitan el funcionamiento del cerebro humano [20].

En este trabajo, se entrenó el modelo “árboles de decisión de regresión”.

Así, se realizó la importación de “Decision Tree Regressor” para establecer el regresor. Como paso base se determinan los parámetros para la construcción del modelo de regresión, de los cuales dependerá el rendimiento del modelo. Como todos los modelos de aprendizaje supervisados, la posibilidad de un sobreajuste existe por lo que es necesario entrenar con diferentes configuraciones.

Entre los parámetros establecidos para el modelo, están: la profundidad del árbol y el número mínimo de muestras necesarias para la división de cada nodo interno.

Tabla 1. Métricas árbol de decisión

Métrica	Valor
MAE	0.41
MSE	0.38
RMSE	0.62
R^2	0.89

Angélica Villón L., 2021 Aplicación de técnicas de minería de datos para predecir el desempeño académico de los estudiantes de la escuela

https://incyt.upse.edu.ec/ciencia/revistas/index.php/rctu/article/view/637/530