CURSO ESPECIALIZACIÓN IA Y BIG DATA

MÓDULO: SISTEMAS DE APRENDIZAJE AUTOMÁTICO

U3: APRENDIZAJE SUPERVISADO

L'FSE inverteix en el teu futur

Fons Social Europeu - FSE

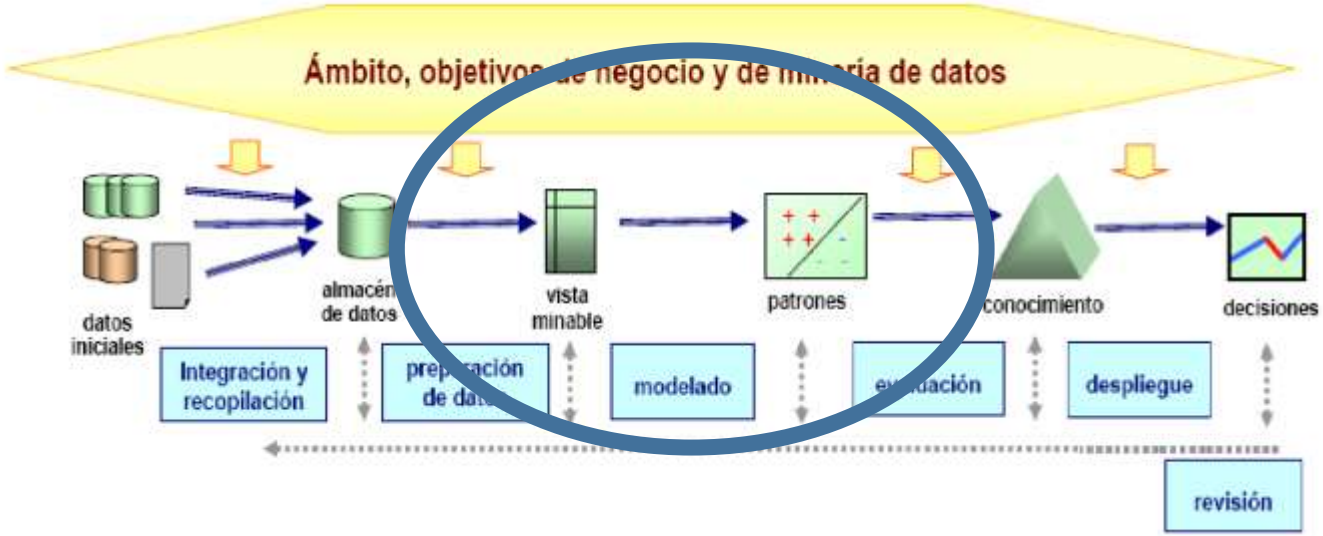
UNIO EUROPEA

GENERALITAT VALENCIANA

Conselleria d'Educació, Cultura i Esport

Índice

1.- Introducción



1.- Aprendizaje supervisado

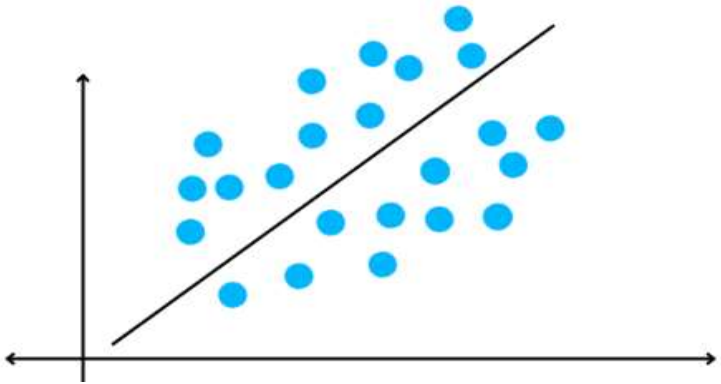
Utiliza un conjunto de datos conocidos, de los que se conoce el resultado, (datos de entrenamiento) para entrenar un algoritmo y realizar predicciones. El conjunto de datos de entrenamiento incluye datos de entrada etiquetados que se emparejan con los valores de salida o de respuesta deseados. A partir de él, el algoritmo de aprendizaje supervisado intenta crear un modelo estableciendo relaciones entre las características y los datos de salida para realizar predicciones acerca de los valores de respuesta para un nuevo conjunto de datos.

Una vez entrenado el algoritmo, se suele utilizar un conjunto de datos de prueba, que no se haya utilizado en el entrenamiento, para predecir el rendimiento y validar el algoritmo.

1.- Aprendizaje supervisado

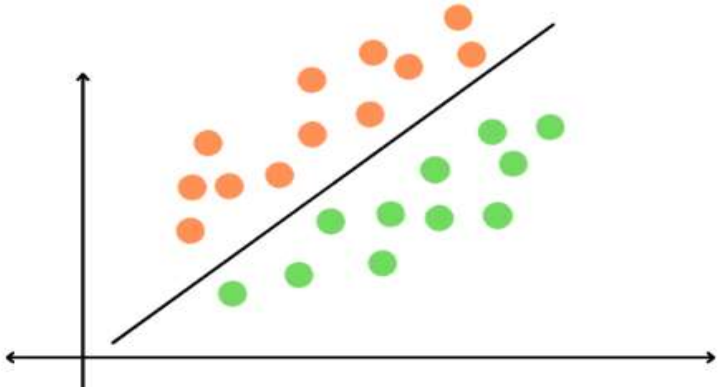
Regresión

¿Qué **valor** tendré a la salida para esta muestra?



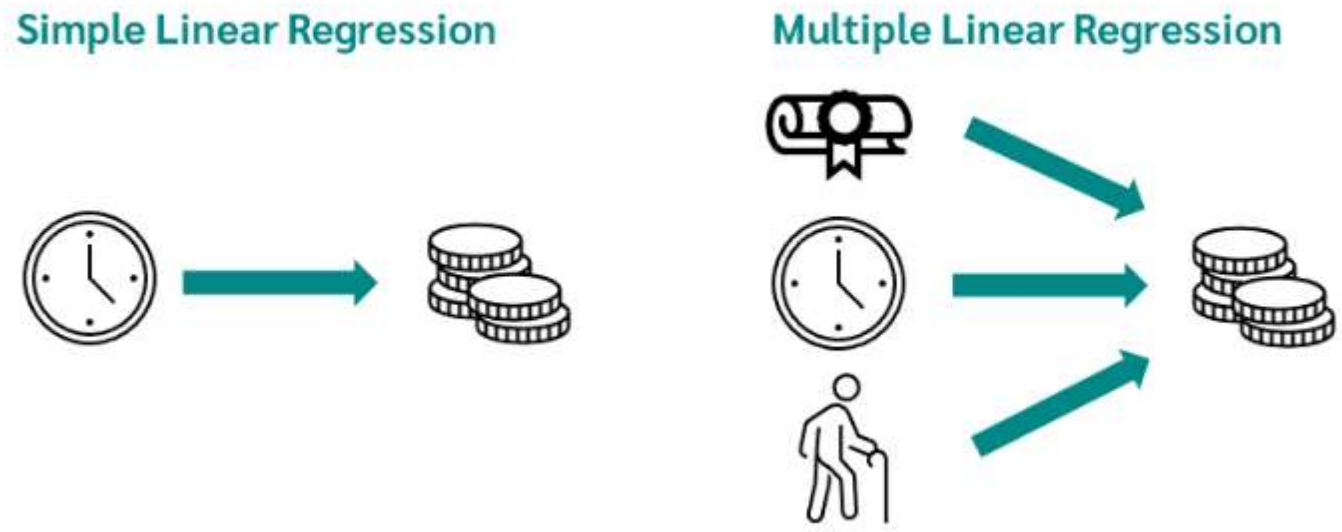
Clasificación

¿A qué **clase** pertenece esta muestra?



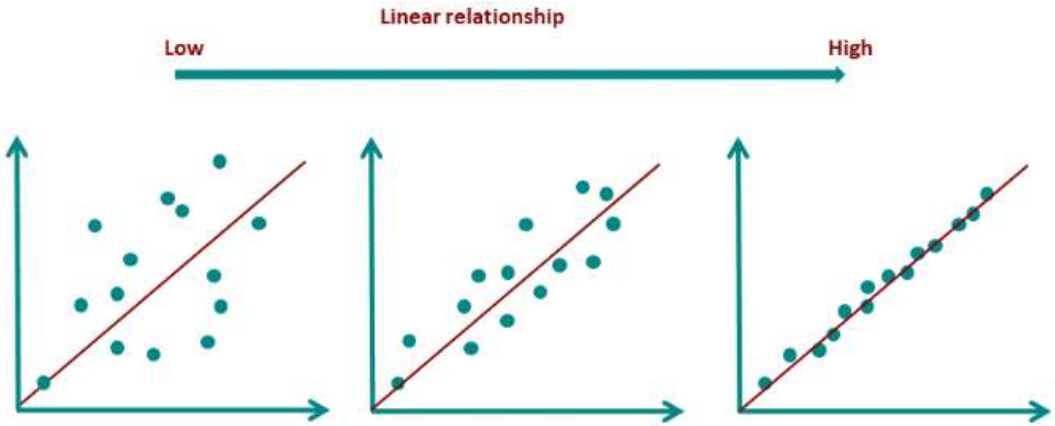
2.- Regresión lineal

Modelo estadístico que estudia las relaciones entre un conjunto de variables independientes Xi y una variable dependiente Y. Es un método de **regresión**.



2.- Regresión lineal simple

Sólo tenemos una variable independiente.



$$\hat{y} = b \cdot x + a$$

Estimated dependent variable

Slope

Independent variable

y intercept

2.- Regresión lineal simple

Ejemplo:

```
import numpy as np
from sklearn.linear_model import LinearRegression

X = np.array([[1, 1], [1, 2], [2, 2], [2, 3]])
y = np.dot(X, np.array([1, 2])) + 3                    # y = 1 * x_0 + 2 * x_1 + 3
reg = LinearRegression().fit(X, y)
reg.score(X, y)
  1.0
reg.coef_
  array([1., 2.])
reg.intercept_
  3.0...
reg.predict(np.array([[3, 5]]))
  array([16.])
```

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

2.- Regresión lineal múltiple

Tenemos dos o más variables independientes

Simple Linear Regression

$$\hat{y} = b \cdot x + a$$

Multiple Linear Regression

$$\hat{y} = b_1 \cdot x_1 + b_2 \cdot x_2 + \ldots + b_k \cdot x_k + a$$

Para evaluar el modelo tenemos:

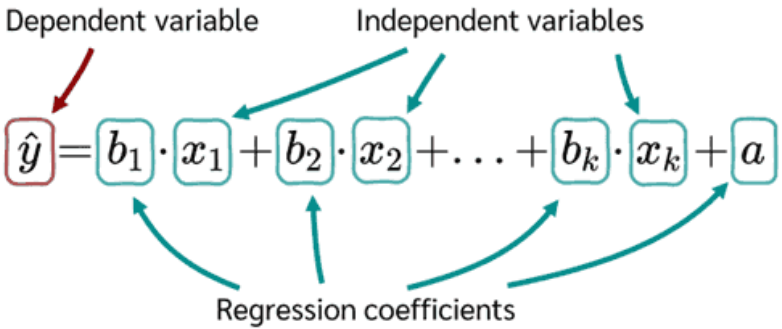
- Coeficiente de determinación R2 o explicación de la varianza, indica cómo de grande es la varianza que

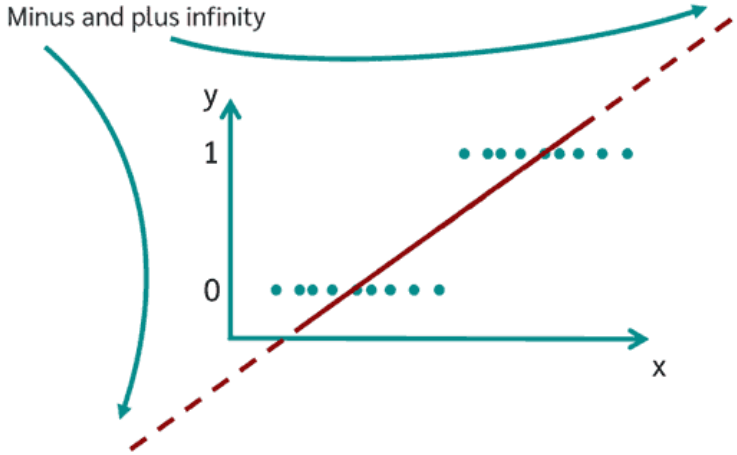
Error estándar de estimación

3.- Regresión logística

Modelo estadístico que estudia las relaciones entre un conjunto de variables Xi y una variable cualitativa Y. Es un método de **clasificación**.

Para construir el modelo se parte de la ecuación de regresión lineal:



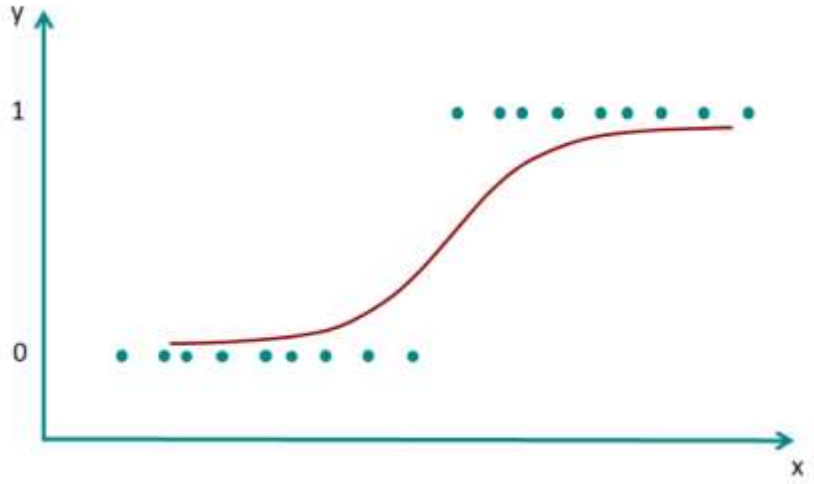


3.- Regresión logística

El objetivo de la regresión logística es la estimación de probabilidades, por lo que se usa una función logística que hace que los valores de salida estén entre 0 y 1.

$$\hat{y} = b_1 \cdot x_1 + b_2 \cdot x_2 + \ldots + b_k \cdot x_k + a$$

$$f(z) = \frac{1}{1+e^{-z}} = \frac{1}{1+e^{-(b_1 \cdot x_1 + \ldots + b_k \cdot x_k + a)}}$$



3.- Regresión logística

Ejemplo:

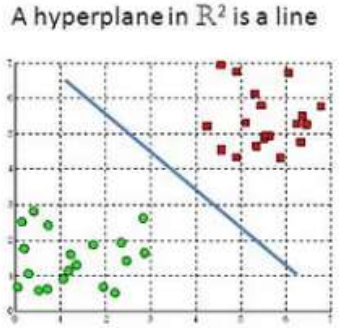
```
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression

X, y = load_iris(return_X_y=True)
clf = LogisticRegression(random_state=0).fit(X, y)
clf.predict(X[:2, :])
clf.predict_proba(X[:2, :])
    array([[9.8...e-01, 1.8...e-02, 1.4...e-08], [9.7...e-01, 2.8...e-02, ...e-08]])
clf.score(X, y)
   0.97...
```

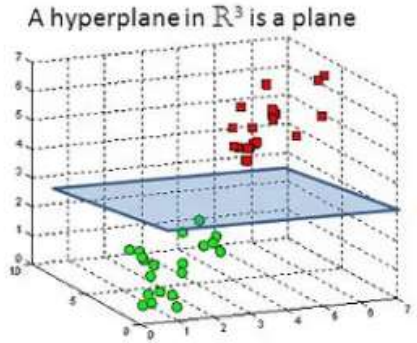
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

4.- SVM (Máquinas de vectores de soporte)

Las SVM inicialmente fueron concebidas para solucionar problemas de clasificación y, posteriormente, fueron extendidas a regresión:

- SVC: Support Vector Classification.
- SVR: Support Vector Regression.



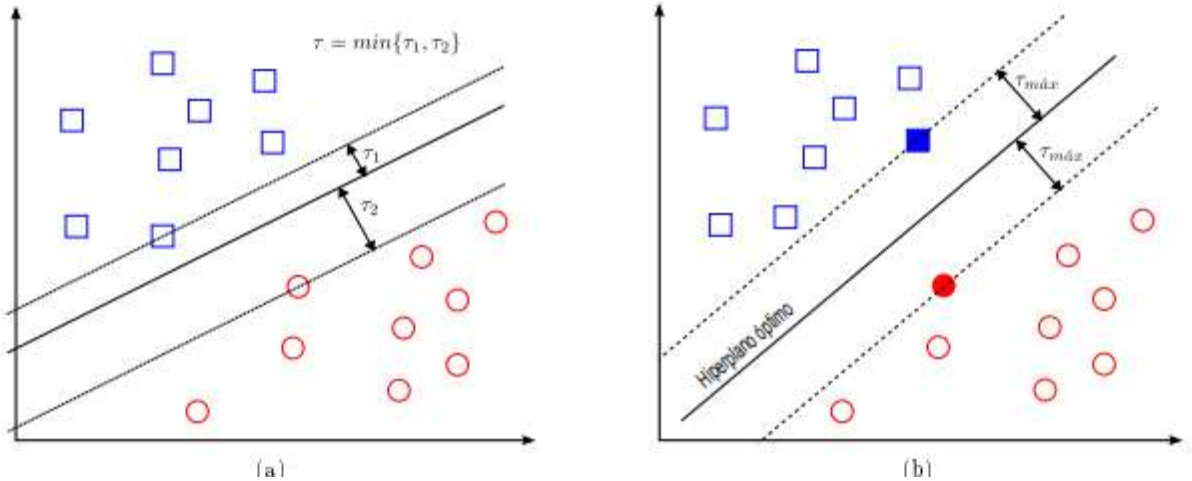


Se basa en el concepto de hiperplano, en un espacio p-dimensional, un hiperplano se define como un subespacio de dimensiones p – 1.

En un espacio de dos dimensiones el hiperplano tiene una dimensión, si el espacio es de tres dimensiones el hiperplano es un plano.

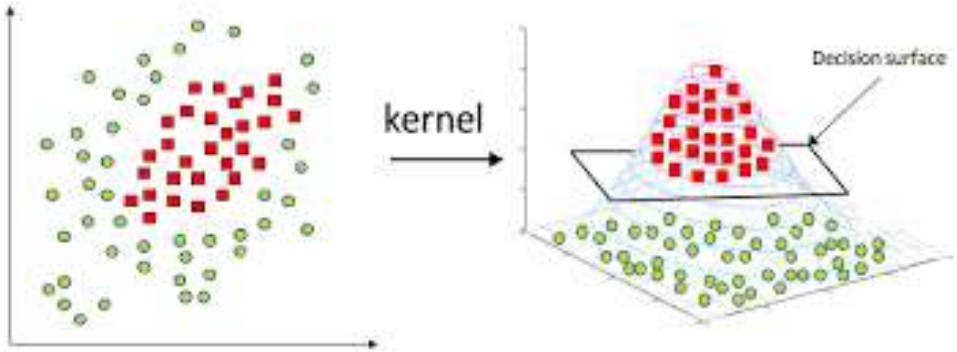
4.- SVM (Máquinas de vectores de soporte)

Para un espacio de dos dimensiones el mejor hiperplano es aquel que maximiza los márgenes, esto es, cuya distancia al elemento más cercano es la más grande.



4.- SVM (Máquinas de vectores de soporte)

Normalmente los datos no se pueden separar linealmente por lo que no hay un hiperplano de separación. En estos casos se aplica una función a los datos que incrementa su espacio dimensional.



4.1.- SVC

Ejemplo:

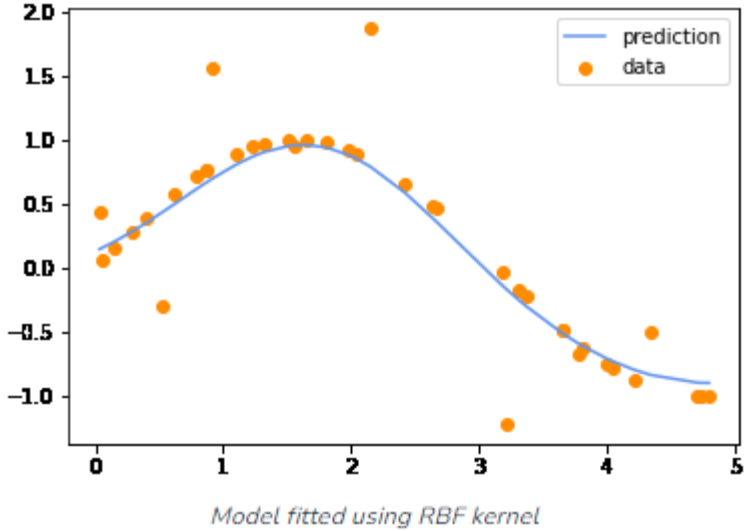
```
import numpy as np
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC

X = np.array([[-1, -1], [-2, -1], [1, 1], [2, 1]])
y = np.array([1, 1, 2, 2])
clf = make_pipeline(StandardScaler(), SVC(gamma='auto’))
clf.fit(X, y)
  Pipeline(steps=[('standardscaler', StandardScaler()), ('svc', SVC(gamma='auto'))])
print(clf.predict([[-0.8, -1]]))
  [1]
```

https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

4.2.- SVR

Ejemplo (*):

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.svm import SVR

X = np.sort(5 * np.random.rand(40, 1),  axis=0)
y = np.sin(X).ravel()
y[::5] += 3 * (0.5 - np.random.rand(8))

svr = SVR(kernel='rbf')
svr.fit(X, y)
y_pred = svr.predict(X)

plt.scatter(X, y, color='darkorange',  label='data')
plt.plot(X, y_pred, color='cornflowerblue',  label='prediction')
plt.legend()
plt.show()
```



Model fitted using RBF kernel

(*) https://www.geeksforgeeks.org/support-vector-regression-svr-using-linear-and-non-linear-kernels-in-scikit-learn/

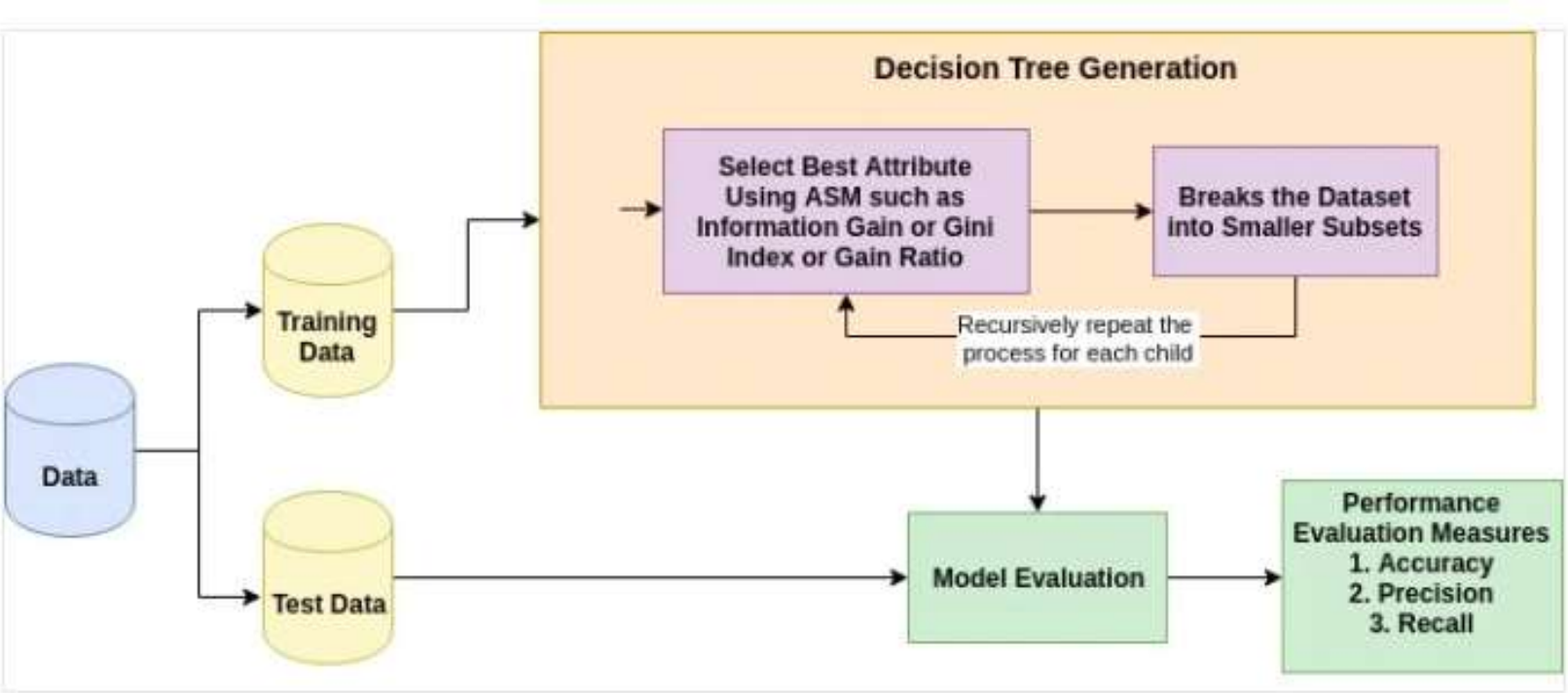
5. Decision trees

Un árbol de decisión es una estructura de datos compuesta por nodos que representan decisiones o acciones, y ramas que representan las posibles consecuencias de esas decisiones.

Los árboles de decisión se construyen mediante una serie de cortes o divisiones del conjunto de datos en subconjuntos más pequeños. Estas divisiones se realizan de manera iterativa utilizando un atributo específico y un criterio de división.

En resumen, los árboles de decisión son algoritmos versátiles que se utilizan tanto para la clasificación como para la regresión en machine learning.

5. Decision trees



5.1.- DecisionTreeClassifier

Ejemplo:

```
from sklearn.datasets import load_iris
from sklearn import tree

iris = load_iris()
X, y = iris.data, iris.target
clf = tree.DecisionTreeClassifier()
clf = clf.fit(X, y)
```

5.2.- DecisionTreeRegresor (*)

```
import numpy as np                                  # for array operations
import pandas as pd                                 # for working with DataFrames
import requests, io                                 # for HTTP requests and I/O commands
import matplotlib.pyplot as plt                     # for data visualization
%matplotlib inline

# scikit-learn modules
from sklearn.model_selection import train_test_split      # for splitting the data
from sklearn.metrics import mean_squared_error            # for calculating the cost function
from sklearn.tree import DecisionTreeRegressor            # for building the model

# Importing the dataset from the url of the data set
url = "https://drive.google.com/u/0/uc?id=1mVmGNx6cbfvRHC_DvF12ZL3wGLSHD9f_&export=download"
data = requests.get(url).content

# Reading the data
dataset = pd.read_csv(io.StringIO(data.decode('utf-8')))
dataset.head()

x = dataset.drop('Petrol_Consumption', axis = 1)          # Features
y = dataset['Petrol_Consumption']                         # Target
```

5.2.- DecisionTreeRegresor (*)

```
# Splitting the dataset into training and testing set (80/20)
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 28)

model = DecisionTreeRegressor(random_state = 0)                    # Initializing the Decision Tree Regression model

model.fit(x_train, y_train)                                        # Fitting the Decision Tree Regression model to the data

y_pred = model.predict(x_test)                                     # Predicting the target values of the test set

# RMSE (Root Mean Square Error)
rmse = float(format(np.sqrt(mean_squared_error(y_test, y_pred)),'.3f'))
print("\nRMSE:",rmse)

from sklearn.tree import export_graphviz                           # Visualizing the decision tree structure

# export the decision tree model to a tree_structure.dot file
# paste the contents of the file to webgraphviz.com
export_graphviz(model, out_file ='tree_structure.dot',
                feature_names = ['Petrol_tax', 'Average_income', 'Paved_Highways', 'Population_Driver_licence(%)'])
```

(*) https://medium.com/@theclickreader/decision-tree-regression-explained-with-implementation-in-python-1e6e48aa7a47

Gracias