**Ventajas**

Claridad de los datos “Puedes saber qué está haciendo dado el camino”

Tolerantes al ruido y valores faltantes

Las reglas extraídas permiten hacer predicciones

**Desventajas**

Criterio de división deficiente”, ya que, si no se divide bien los datos, se puede desviar la pred.”

Sobreajuste

Ramas poco significativas

**¿Cómo se divide un árbol de decisión?**

Ganancia de información

Dividir en pequéños árboles

**Optimización de nuestro modelo**

Evitar sobreajuste

Selección de atributos “Evaluar muy bien los atributos que dibidirán los datos”

Campos nulos “No tener campos nulos, esto provoca ruido en nuestro arbol”

**Primero**: Importamos las linrerías que vamos a usar

import pandas as pd

from sklearn import tree

from sklearn.model\_selection import train\_test\_split

import matplotlib.pyplot as plt

import seaborn as sns

import re

import numpy as np

%matplotlib inline

sns.set()

**Segundo:**

Importamos nuestros datos

**Tercero:**

Separamos nuestros datos para entrenamiento y testeo

test\_df = pd.read\_csv('titanic-test.csv')

train\_df = pd.read\_csv('titanic-train.csv')

Nota1: train\_df.info()

Con esta isntrucción obtenemos la información de los datos de este DataFrame

Nota2: train\_df.Sex.value\_counts().plot(kind = 'bar', color = ['b', 'r'])

Con esta instrucción elaboramos un gráfico de barras con los colores rojo y azul

**Cuarto:**

Debemos de procesar nuestros datos para poder trabajar con ellos

**De esta manera estamos modificando los valores de género a numéricos**

from **sklearn** import **preprocessing**

label\_encoder = **preprocessing.LabelEncoder()**

encoder\_sex = **label\_encoder**.fit\_transform(train\_df['Sex'])

**De esta manera llenamos los datos nulos.**

train\_df['Age'] = train\_df['Age']**.fillna**(train\_df['Age'].median())

train\_df['Embarked'] = train\_df['Embarked']**.fillna**('S')

#Es recomendable intercambiarlo por un valor promedio para que no afecte al calc. la media, Desv. Est., etc.

**De esta manera estamos quitando las columnas que son innecesarias para hacer nuestros cálculos de predicción.**

train\_predictors = **train\_df.drop**('PassengerId', 'Survived', 'Name', 'Ticket', 'cabin', axis= 1)

**Por otro lado podemos hacer los siguiente**

**Talking abot the conditional " train\_predictors[cname].nunique() < 10 "**

Lo que sucede es que puedes tener una variable categórica que tenga “muchos” valores distintos, entonces lo que hace esa porción de código es que toma las variables cuya cantidad de valores distintos no sea mayor a 10.

**categorical\_cols = [cname for cname in train\_predictors.columns if**

**train\_predictors[cname].nunique() < 10 and**

**train\_predictors[cname].dtype == 'object' ]**

**Asimismo, podemos hacer esto**

* Tomaremos las variables numéricas
* Recorremos también con "cname"

La condición aquí es que las columnas deben de ser de "tipo numérico".En este caso deben ser de tipo entero o de tipo flotante de 64 bits

**numerical\_cols = [cname for cname in train\_predictors.columns if**

**train\_predictors[cname].dtype in ['int64', 'float64']**

**]**

**Ahora los vamos a unir los 2 últimos para tenerla en una sola “variable”**

**my\_cols = categorical\_cols + numerical\_cols**

**#Ahora a train predictors le estamos dando las variables que queremos para el entrenamiento**

**train\_predictors = train\_predictors[my\_cols]**