---

title: "My Title"

description: "Awesome description"

layout: post

toc: false

comments: true

image: images/some\_folder/your\_image.png

hide: false

search\_exclude: true

categories: [fastpages, jupyter]

metadata\_key1: metadata\_value1

metadata\_key2: metadata\_value2

---

En múltiples ocasiones, los equipos de marketing deberán estimar la mejor estrategia para vender un nuevo producto. Especialmente cuando se trata de una marca establecida, direccionar la estrategia para los clientes potenciales determina el éxito del lanzamiento del producto. Estos casos son especiales, ya que es muy alta la probabilidad de que se disponga de datos sobre los productos anteriormente lanzados.

En base a datos históricos, es posible entrenar un modelo que permita determinar si un cliente comprará el nuevo producto cuando este sea anunciado. En este caso, se estudiará un Dataset con información de los patrones de comportamiento de los clientes en sus actividades en el sitio web de una empresa de E-Readers.

## Contexto

Se dispone de dos Dataset con información de los clientes. Uno de ellos está completamente etiquedado de acuerdo con las compras realizadas de los clientes. La variable objetivo a predecir es del tipo nominal, por lo que tiene múltiples y determinados valores posibles.

Por su parte, el segundo Dataset no contiene esta variable, el mismo corresponde a información actual de los perfiles de los clientes. A partir de éste, se deberán predecir las clases en la etapa de testeo del modelo.

Naturalmente este tipo de problema se tratará con un modelo supervisado de clasificación, ya que es de nuestro interés predecir la clase de clientes sin clasificar.

Para este ejercicio, se utilizará un modelo de Árbol de Decisión en Rapidminer. El objetivo será examinar y comparar el funcionamiento de este tipo de algoritmo no lineal para un ejercicio de clasificación.

## Datos

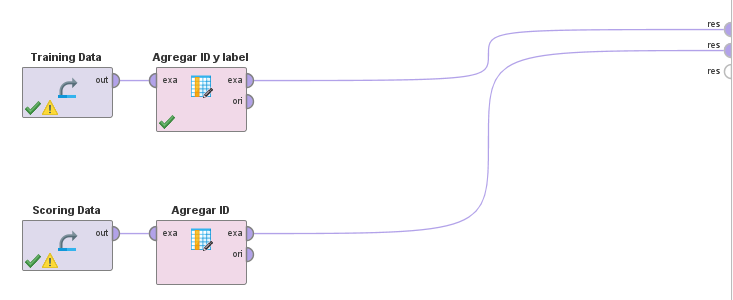
Los árboles de decisión son modelos de Machine Learning que requieren poca preparación de los datos, ya que son bastantes permisivos con valores faltantes, distribuciones sesgadas y distintos tipos de datos.

Con respecto a este último punto, la siguiente tabla muestra información sobre los predictores disponibles:

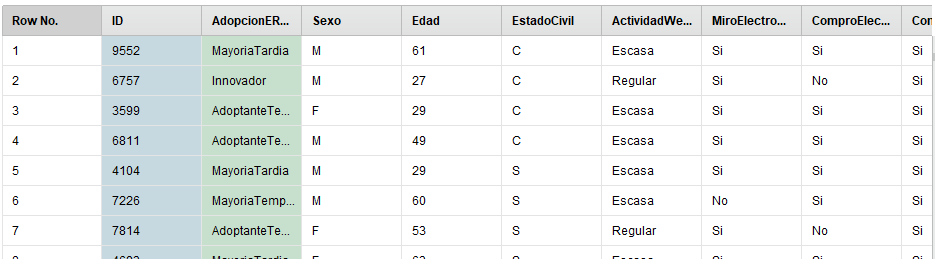
|  |  |  |
| --- | --- | --- |
| Predictor | Rango | Comentario |
| ID | - | Identificador del cliente |
| Edad | - | Variable numérica |
| EstadoCivil | C - S | Variable binomial |
| Sexo | F – M | Variable binomial |
| ActividadWebsite | Escasa – Regular - Frecuente | Variable nominal |
| ComproElectronicos12 | SI – NO | Variable binomial |
| MiroElectronicos12 | SI – NO | Variable binomial |
| ComproMedios18 | SI – NO | Variable binomial |
| ComproLibrosDigitales | SI - NO | Variable binomial |
| MetodoPago | Transferencia Bancaria – Cuenta Website – Tarjeta Credito – Debito Mensual | Variable nominal |

En este caso, la mayoría de variables son categóricas. Considero que todos los datos proporcionados son relevantes para el ejercicio, a excepción del identificador del cliente. Este dato no es de utilidad para la predicción, ya que se trata de una referencia a un registro en particular.

Este tipo de datos puede ser excluido del Dataset desde el inicio. Sin embargo, en este caso usaré una ventaja que aporta Rapidminer, cambiaré el tipo de dato a “Identificador”. De esta manera, se mantendrá a lo largo del flujo, pero no influirá en el entrenamiento del modelo.



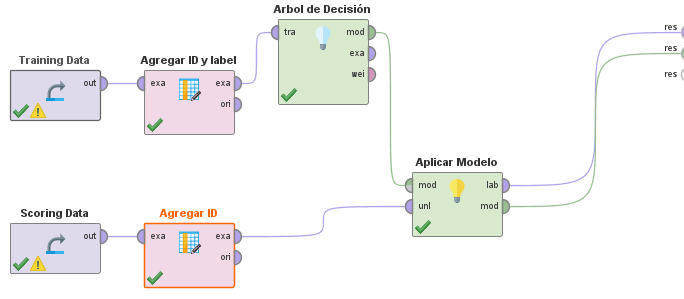
Por medio del operador “Set Role” es posible modificar el papel de ese atributo en Rapidminer. En la siguiente imagen se puede ver como son diferenciados de los demás.



## Modelo

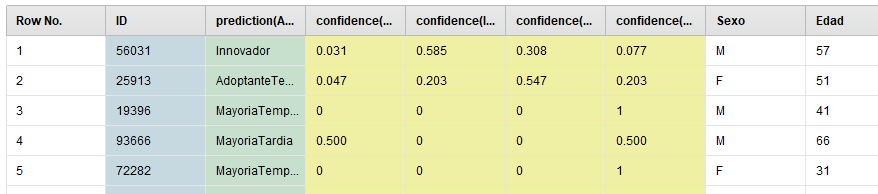
Agregamos un operador de Árbol de Decisión en el flujo de Rapidminer. El mismo usará el dataset de entrenamiento como entrada. El modelo resultante será probado mediante el operador “Apply Model”, usando el Dataset de Test.

El flujo resultante se ve de la siguiente manera:



Al ejecutar el modelo con los datos de prueba, vemos que Rapidminer agrega 4 atributos de confianza y un atributo adicional para la predicción de la clase objetivo.

Cada una de estas confianzas indican que tan probable es que el modelo clasifique el registro en cada una de las 4 clases. Por tanto, la suma de todas las confianzas da como resultado 1.



La probabilidad obtenida para cada registro corresponde a la proporción de ejemplos de clases distintas que durante el entrenamiento se generaron en el mismo nodo hoja.

Queda claro que aquellos registros con confianzas más equilibrados tienen menor seguridad de predicción frente a aquellos donde las diferencias son marcadas.

A continuación, voy a seleccionar 3 registros en particular y aplicar modificaciones en los hyperparámetros del árbol de decisión. Esta prueba debería arrojar resultados distintos de clase objetivo, especialmente si los registros elegidos tienen confianzas equilibradas.

Se ejecutarán 4 configuraciones distintas con los siguientes valores de parámetros

1. Criterio: Gain\_Ratio  
   Maximal\_Depth: 10  
   Minimal\_Gain: 0.01
2. Criterio: Gini\_Index  
   Maximal\_Depth: 10  
   Minimal\_Gain: 0.01
3. Criterio: Gain\_Ratio  
   Maximal\_Depth: 5  
   Minimal\_Gain: 0.05
4. Criterio: Gain\_Ratio  
   Maximal\_Depth: 20  
   Minimal\_Gain: 0.05

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID Cliente | Predicción Test 1 | Predicción Test 2 | Predicción Test 3 | Predicción Test 4 |
| 98200 | Adoptante Temprano 54% | Adoptante Temprano 100% | Adoptante Temprano 41% | MayoriaTardia 84% |
| 76655 | MayoriaTardia 84% | Adoptante Temprano 50% | MayoriaTardia 75% | Adoptante Temprano 100% |
| 63570 | Adoptante Temprano 100% | Adoptante Temprano 75% | Adoptante Temprano 100% | Adoptante Temprano 100% |

Como conlusión, los parámetros que establecen las distintas operaciones a efectuar sobre el árbol de decisión generan resultados muy distintos entre sí, cambiando la confianza de predicción de algunos registros, incluso la clase predicha en sí.

Este tipo de algoritmo de Machine Learning es fácil de entender, ya que su salida es una representación que podemos entender. El modelo genera un árbol donde en cada nodo se deberá tomar una decisión con respecto a un predictor dado y en sus hojas estarán las clases objetivo a predecir. Por tanto, predecir un nuevo valor implica simplemente recorrer el árbol generado.