



INTRODUCCIÓN A LA INTELIGENCIA ARTIFICIAL

SITUACIÓN PROFESIONAL 3

CLASE 4

Prof. Ricardo Piña

Table of Contents

- [1 Situación Profesional](#)
 - [1.1 Ejercicio 1: Marketing Bancario:](#)
- [2 Resolución de la Situación Profesional](#)
 - [2.1 Flujo de trabajo](#)
- [3 Medidas de Evaluación de problemas de Clasificación](#)
 - [3.1 Confusion Matrix](#)
 - [3.2 AUC](#)
 - [3.3 CA](#)
 - [3.4 F1](#)
 - [3.5 Precision](#)
 - [3.6 Recall](#)
 - [3.7 Evaluación de Modelos](#)
- [4 El problema del Overfitting](#)
 - [4.1 Sorpresa!](#)
 - [4.2 Fronteras de Decisión para distintas profundidades de árbol](#)
 - [4.3 Conclusión](#)
 - [4.4 Cuestionario o Autoevaluación \(parte teórica\)](#)

Ver en Video:

Título: [IA] Clase 4

link: <https://www.youtube.com/watch?v=YvbFfM3OIU8&t=1237s> (<https://www.youtube.com/watch?v=YvbFfM3OIU8&t=1237s>)

desde: inicio

hasta: fin

Ignore este comentario que está dirigido al profesor: Atención también puede ir la clase 5 completa, ya que creo que es la resolución de algunos casos prácticos.

Out[5]:

A esta altura de la carrera Ud. todavía no sabe programar en Python, así que en este archivo hemos ocultado las celdas que contienen código para facilitar su lectura. Si Ud. quiere ver el código u ocultarlo, haga [click aquí](#).

Out[6]:

Situación Profesional

Ejercicio 1: Marketing Bancario:

- Fuente de los datos: Universidad de Irvine, California.
- Origen de los datos: Datos **reales** de un banco portugués.
- archivo: *bank-additional.csv*

La siguiente es la información brindada por el banco portugués.

Descripción del Problema:

- The data is related with direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to access if the product (nota: bank term deposit = "plazo fijo") would be ('yes') or not ('no') subscribed.
- The **classification goal is to predict if the client will subscribe (yes/no) a term deposit (variable y)**.

A continuación nos proveen de la descripción de cada una de las variables:

Input variables:

bank client data:

1. age (numeric)
2. job : type of job (categorical: 'admin.', 'blue-collar', 'entrepreneur', 'housemaid', 'management', 'retired', 'self-employed', 'services', 'student', 'technician', 'unemployed', 'unknown')
3. marital : marital status (categorical: 'divorced', 'married', 'single', 'unknown'; note: 'divorced' means divorced or widowed)
4. education (categorical: 'basic.4y', 'basic.6y', 'basic.9y', 'high.school', 'illiterate', 'professional.course', 'university.degree', 'unknown')
5. default: has credit in default? (categorical: 'no', 'yes', 'unknown')
6. housing: has housing loan? (categorical: 'no', 'yes', 'unknown')
7. loan: has personal loan? (categorical: 'no', 'yes', 'unknown')

related with the last contact of the current campaign:

1. contact: contact communication type (categorical: 'cellular', 'telephone')
2. month: last contact month of year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')
3. day_of_week: last contact day of the week (categorical: 'mon', 'tue', 'wed', 'thu', 'fri')
4. duration: last contact duration, in seconds (numeric). Important note: this attribute highly affects the output target (e.g., if duration=0 then y='no'). Yet, the duration is not known before a call is performed. Also, after the end of the call y is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model.

other attributes:

1. campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
2. pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)
3. previous: number of contacts performed before this campaign and for this client (numeric)
4. poutcome: outcome of the previous marketing campaign (categorical: 'failure', 'nonexistent', 'success')

social and economic context attributes:

1. emp.var.rate: employment variation rate - quarterly indicator (numeric)
2. cons.price.idx: consumer price index - monthly indicator (numeric)
3. cons.conf.idx: consumer confidence index - monthly indicator (numeric)
4. euribor3m: euribor 3 month rate - daily indicator (numeric)
5. nr.employed: number of employees - quarterly indicator (numeric)

Output variable (desired target):

1. y - has the client subscribed a term deposit? (binary: 'yes', 'no')

Resolución de la Situación Profesional

Aplicaremos el método que presentamos con anterioridad dividiendo el conjunto de datos que poseemos en dos partes, el **Train Set** y el **Test Set**. Existen diversos métodos aplicables al Aprendizaje Supervisado; el que vamos a investigar ahora se puede utilizar prácticamente siempre, existen muchas variantes adecuadas para diversos casos, pero que mantienen una misma idea:

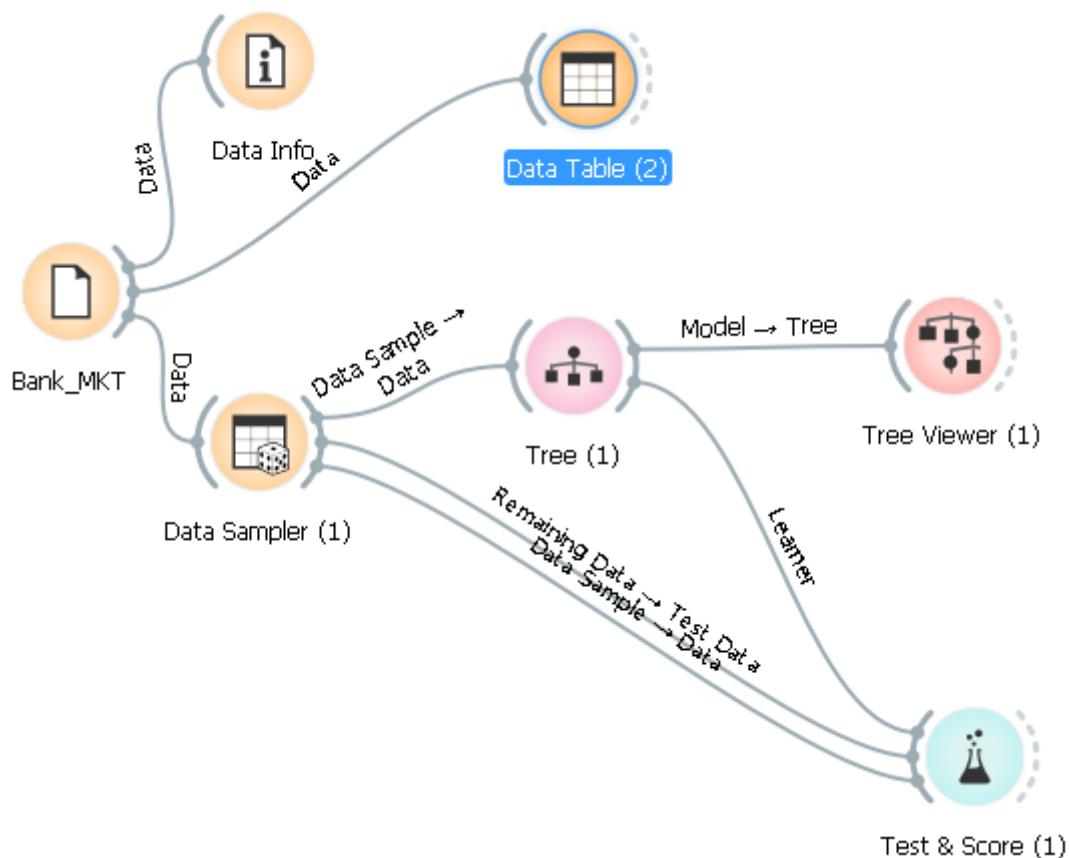
- **entrenaremos** a nuestro modelo con el "Train Set", con el cual elaboraremos nuestro modelo o hipótesis.
- **testearemos** al modelo o hipótesis con "Test Set"
- utilizaremos algún criterio **previamente establecido** para **medir** qué tan bien pronosticó nuestra hipótesis **en el test set** y si no alcanzó los valores deseados tomaremos alguna medida para cambiar el modelo o hipótesis. En nuestro caso como todavía no conocemos ningún criterio de evaluación lo dejaremos para el final.

Flujo de trabajo

Realizaremos todo el trabajo con Orange3

1. **Cargar** los datos del archivo "bank-additional.csv" en Orange3 con el widget **File**.
2. **Dividir** el Data Set en Train y Test (70% / 30%). Usar el widget **Data Sampler**
3. Efectuar el **modelado** con Árbol de Decisión de profundidad máxima.
4. **Evaluar** los resultados, parámetros Exactitud (Accuracy) y F1 Score.
5. **Repetir** el modelado y evaluación **con árboles de distinta profundidad**.
6. **Explicar** los resultados.

El **Flujo de Trabajo (Workflow)** básico en Orange puede ser como el siguiente:



Encontrará todo el Desarrollo en el video indicado, véalo hasta el minuto 43:50

En la siguiente figura le mostramos las opciones de configuración de los widgets de Orange3 y los valores seleccionados para los pasos 1 a 4 del flujo de trabajo mencionado anteriormente (hemos omitido la imagen del Tree Viewer porque era muy grande).

Dentro de las opciones de configuración más importantes que deberá tener en cuenta, preste atención a las siguientes

1. Widget **Data Sampler**:

- Utilice Fixed Proportion for Data = 70%. Este control lo usamos para crear el Train Set y el Test Set, en este ejercicio hemos decidido utilizar la proporción de 70% / 30%
- Replicable (deterministic) sample: las observaciones que irán al Train y Test Set deben elegirse al azar, si seleccionamos esta opción cada vez que corremos este archivo las observaciones que se asignarán a cada uno de estos conjuntos serán las mismas y se obtendrán los mismos resultados. Para nosotros es muy importante que esto sea así porque pretendemos que nuestros trabajos tengan validez científica lo cual, entre otras cosas, implica que si otra persona repite lo que hemos hecho nosotros debe obtener los mismos resultados.

1. Widget **Tree**. Es el que genera el Árbol de Decisión. Las opciones que trae son para "podar" el árbol, es decir para achicar su profundidad. Nosotros sólo vamos a seleccionar

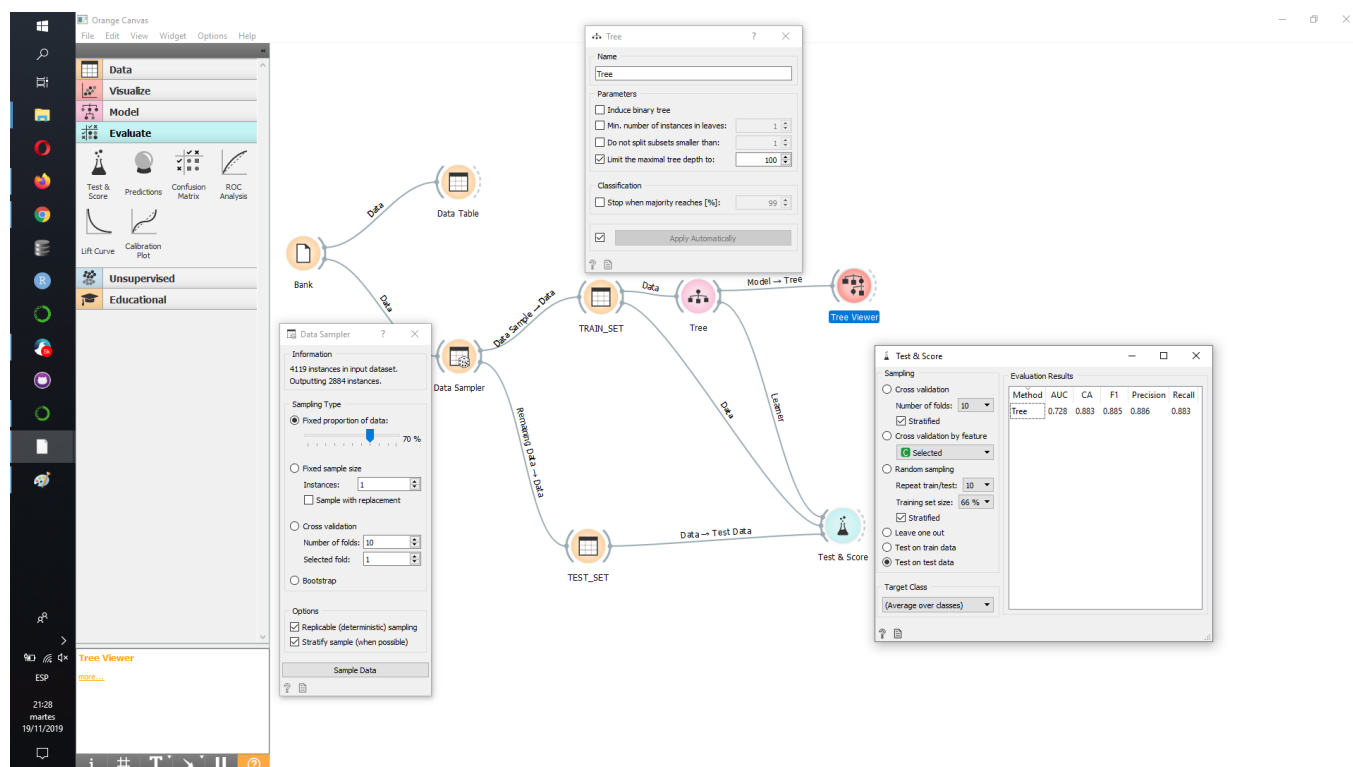
- **Limit the maximal tree depth to** y le asignaremos una enorme profundidad de 100. En nuestro caso será lo mismo que decir que no lo hemos limitado. Un poco más adelante le pondremos límites más pequeños.

2. Widget **Test&Score**: Es el que nos permitirá medir la performance de los modelos que creemos. Dado que hemos creado un Test Set específico para testear debe seleccionar la opción **Test on Test Data**.

3. Widget **Tree Viewer**: Es el encargado de mostrarnos gráficamente la composición del Árbol de Decisión. Simplemente úselo.

Hágalo Ud en su propio archivo de Orange, si tiene inconvenientes, revise el archivo:

Clase_04_bank_mkt_2.ows, **Nota**: debe abrirlo desde dentro de Orange3!



Ahora quiero detenerme en las opciones del control Test & Score que son las medidas de evaluación típicas.

Medidas de Evaluación de problemas de Clasificación

Una vez entrenado nuestro modelo con el Train Set, debemos evaluar qué tan bien se comportará el mismo frente a los casos en los que no ha sido entrenado, para ello nos hemos guardado el Test Set. Como conocemos los resultados correctos de cada una de las observaciones podemos correr nuestro modelo y ver si los resultados pronosticados por él aciertan o no.

De esta manera podremos medir qué tan bien o qué tan mal funciona nuestro modelo.

Existen diversas maneras de medir esta performance, a continuación veremos varias de ellas.

Confusion Matrix

En nuestro caso los valores a pronosticar son dos, Si o No, los valores que corresponden a la variable **y** que indicaba si la persona tomaba o no el plazo fijo. Cuando pronosticamos con nuestro modelo y lo testamos contra el Test Set pueden ocurrir 4 situaciones:

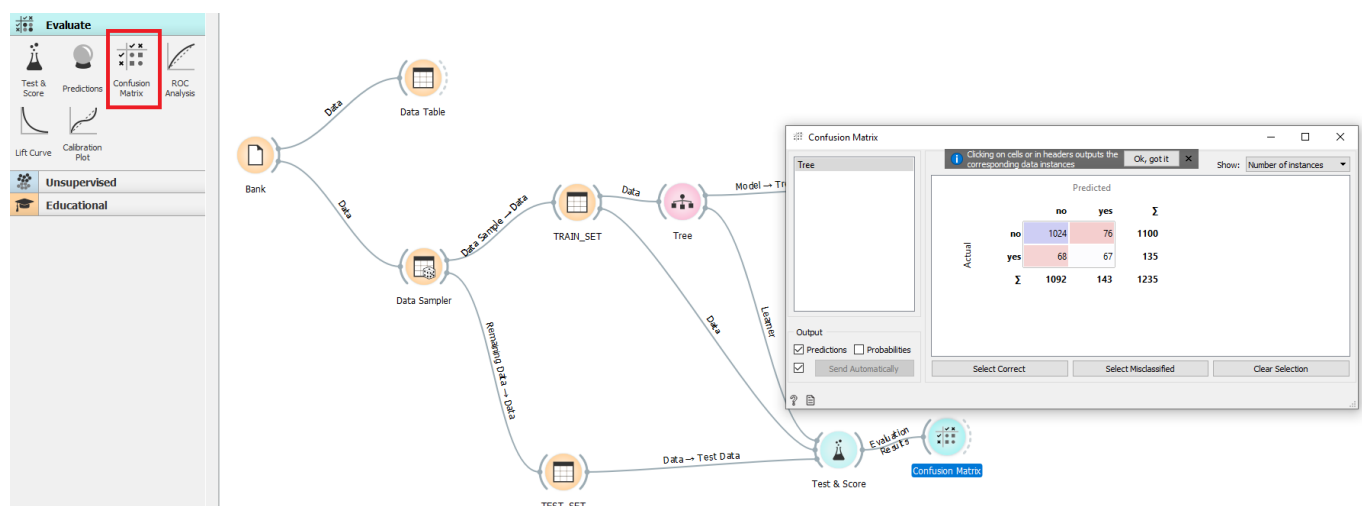
- que el valor real sea Si y nuestro modelo pronostique Si (un acierto para el modelo)
- que el valor real sea Si y nuestro modelo pronostique No (un error para el modelo)
- que el valor real sea No y nuestro modelo pronostique No (un acierto para el modelo)
- que el valor real sea No y nuestro modelo pronostique Si (un error para el modelo)

Como en el Test Set tenemos muchas observaciones, podemos contar cuántos casos de cada una de estas situaciones tenemos.

Esa información se suele resumir en forma de matriz y se denomina **"Confusion Matrix"**.

Podríamos hacerlo a mano, contando una por una las observaciones y comparando el valor pronosticado con el real del Test Set, pero Orange3 y la mayoría del software de Machine Learning, nos permiten hacerlo automáticamente.

En el caso de Orange3, arrastre el widget Confusion Matrix desde el panel Evaluate hacia la zona de trabajo y vincúlelo a la salida del widget Test & Score, como se muestra en la siguiente figura.



Para leer la matriz de confusión debemos tener en cuenta que en inglés **actual** significa **real**.

De esta manera:

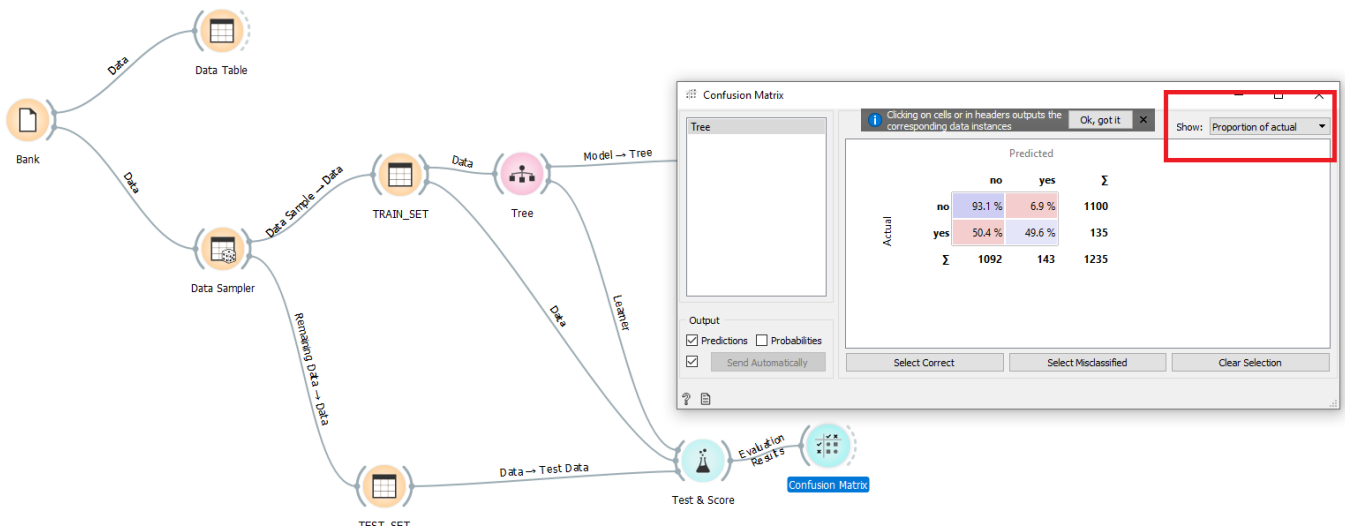
- la primer fila corresponde a las observaciones en las cuales el valor **real** es **No**
- la segunda fila corresponde a las observaciones en las cuales el valor **real** es **Si**
- la primer columna corresponde a las observaciones en las cuales el valor **pronosticado** es **No**
- la segunda columna corresponde a las observaciones en las cuales el valor **pronosticado** es **Si**

Podemos leer la matriz de la siguiente forma:

- De las 1100 observaciones que tenían valor real No, nuestro modelo pronosticó como No a 1024 de ellas (correctamente pronosticadas)
- De las 1100 observaciones que tenían valor real No, nuestro modelo pronosticó como Si a sólo 76 de ellas (mal pronosticadas)
- De las 135 observaciones que tenían valor real Si, nuestro modelo pronosticó como No a 68 de ellas (mal pronosticadas)
- De las 135 observaciones que tenían valor real Si, nuestro modelo pronosticó como Si a 67 de ellas (bien pronosticadas)

En total habían 1235 observaciones.

También es posible calcular la Matriz de Confusión como porcentaje de los datos reales:

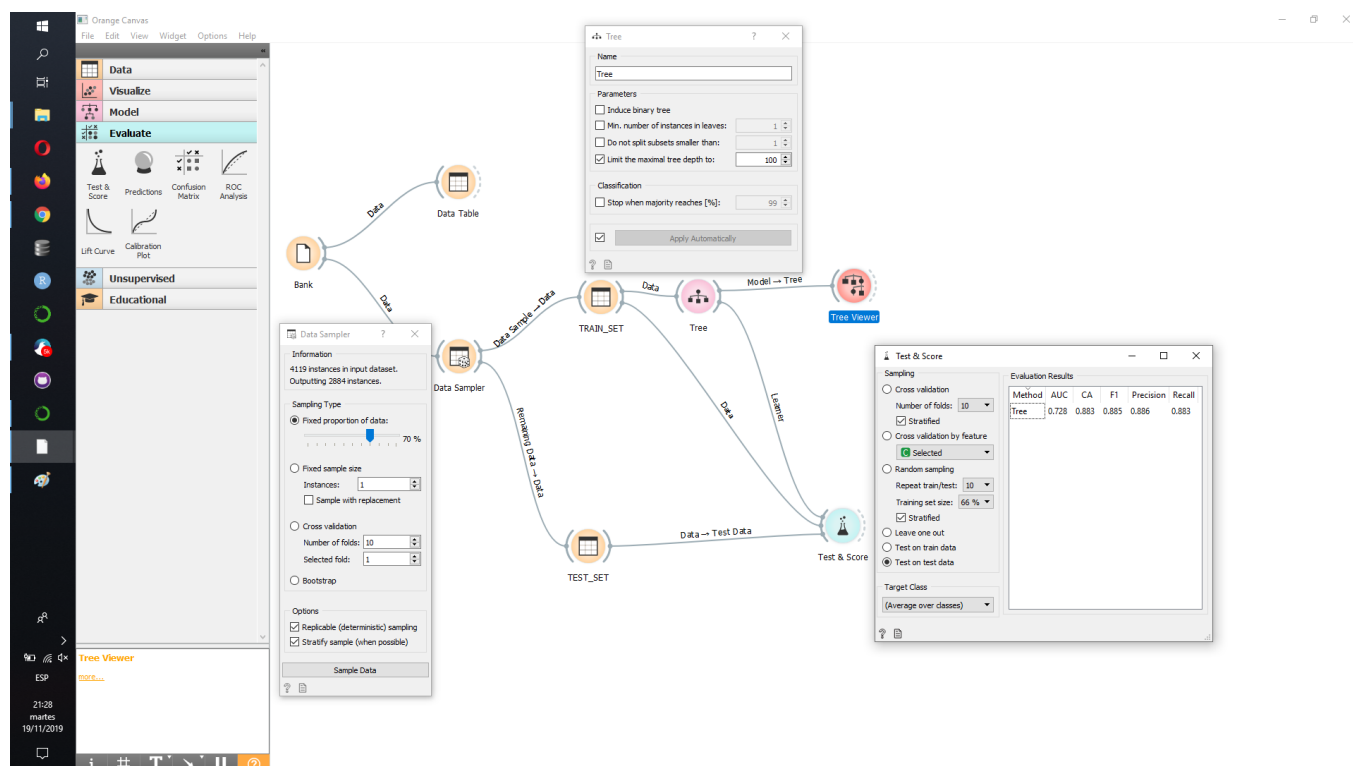


En esta materia no profundizaremos más este análisis, pero daría la impresión que los casos que son No, los clasifica muy bien, pero los casos que son Si, los clasifica bastante mal, acertando sólo la mitad de las veces.

Como decíamos, en otras materias analizaremos a qué puede deberse ésto y veremos cómo podemos mejorar la performance de nuestro modelo, pero le voy a dar una pista, es posible que en el conjunto de datos tengamos pocos casos Si y muchos casos No y que nuestro modelo esté aprendiendo de manera un poco sesgada. Sin más análisis no podemos decir nada.

Sigamos con nuestro trabajo.

El resto de las medidas de evaluación las encontraremos en el widget Test&Score, recordemos la imagen:



AUC

Area Under Curve (AUC): La veremos después en otra materia.

CA

Classification Accuracy (CA) o Exactitud:

Accuracy o Exactitud es una de las métricas más intuitivas y más usadas para evaluar modelos de clasificación y es tan simple como contar la cantidad de observaciones bien hechas y dividir las sobre el total de las observaciones:

$$Accuracy = \frac{\text{Cantidad de Aciertos}}{\text{Total de Observaciones}}$$

La Exactitud siempre será un valor comprendido entre 0 y 1 inclusive ambos extremos.

En nuestro caso a través de la Matriz de Confusión podemos calcular la **exactitud**.

Cuáles son los aciertos?

- Los que eran No y nuestro modelo clasificó como No, que son 1024 casos
- Los que eran Si y nuestro modelo clasificó como Si, que son 67

Y el total de observaciones es de 1235, así que la Exactitud o Accuracy será de:

$$Accuracy = \frac{(1024 + 67)}{1235}$$

$$Accuracy = 0,883$$

Observe que en nuestro problema la **Exactitud o Accuracy** es muy buena, 0.833 (sobre un máximo de 1) merced más que nada a los 1024 valores No bien pronosticados, sin embargo en las observaciones que tendrían que haber dado que Sí, vemos que pronosticó bien alrededor de la mitad de los casos. En materias posteriores analizaremos en profundidad estos casos y cómo mejorar esa performance

F1

El valor de **F1** es una métrica en algunos aspectos **más completa** que la exactitud para evaluar la performance de nuestro modelo, pero algo más compleja para comprender, por eso en esta materia nos conformaremos con saber que es una muy buena métrica para evaluar los modelos de clasificación.

El **valor de F1 también está comprendido entre 0 y 1 siendo mejor cuanto mayor sea su valor.**

Observe en los resultados del widget Test&Score de Orange3 y veremos que el valor de **F1 = 0,885** que es también un muy buen valor.

Precision

Este valor también lo veremos en una materia posterior

Recall

Este valor también lo veremos en una materia posterior

En definitiva, para los objetivos que tenemos en esta materia, **utilizaremos Accuracy (Exactitud) y el coeficiente F1** para estimar qué tan bien pronostica nuestro modelo.

Ambos valores están comprendidos entre 0 y 1, y cuanto más cercanos a 1, mejor.

Hasta aquí hemos resuelto los pasos 1 a 4 del flujo de trabajo propuesto. El paso 5 indica que repitamos el proceso con árboles de diversa profundidad. Recuerda que al árbol actual le habíamos puesto una profundidad de 100, que en este caso era lo mismo que decir sin límite, bueno ahora vamos a crear otros árboles, pero de menor profundidad y veremos qué pasa ...

Árbol de Profundidad 1

Al flujo de Orange le vamos a agregar un nuevo widget Tree del Panel Model, esta vez:

- seleccionamos que la profundidad sea limitada a 1, como se muestra en la figura siguiente con un recuadro rojo,
- lo vinculamos de la misma manera que al anterior al Train Set para que tome los mismos datos que antes para entrenar a este modelo más "chato".
- y le cambiamos el nombre, por el de Tree_01 para recordarnos que será el árbol de nivel 1
- también cámbiele el nombre al mismo widget por TREE_01.

Desde el Panel Visualize agregamos un control Tree Viewer para visualizar el árbol

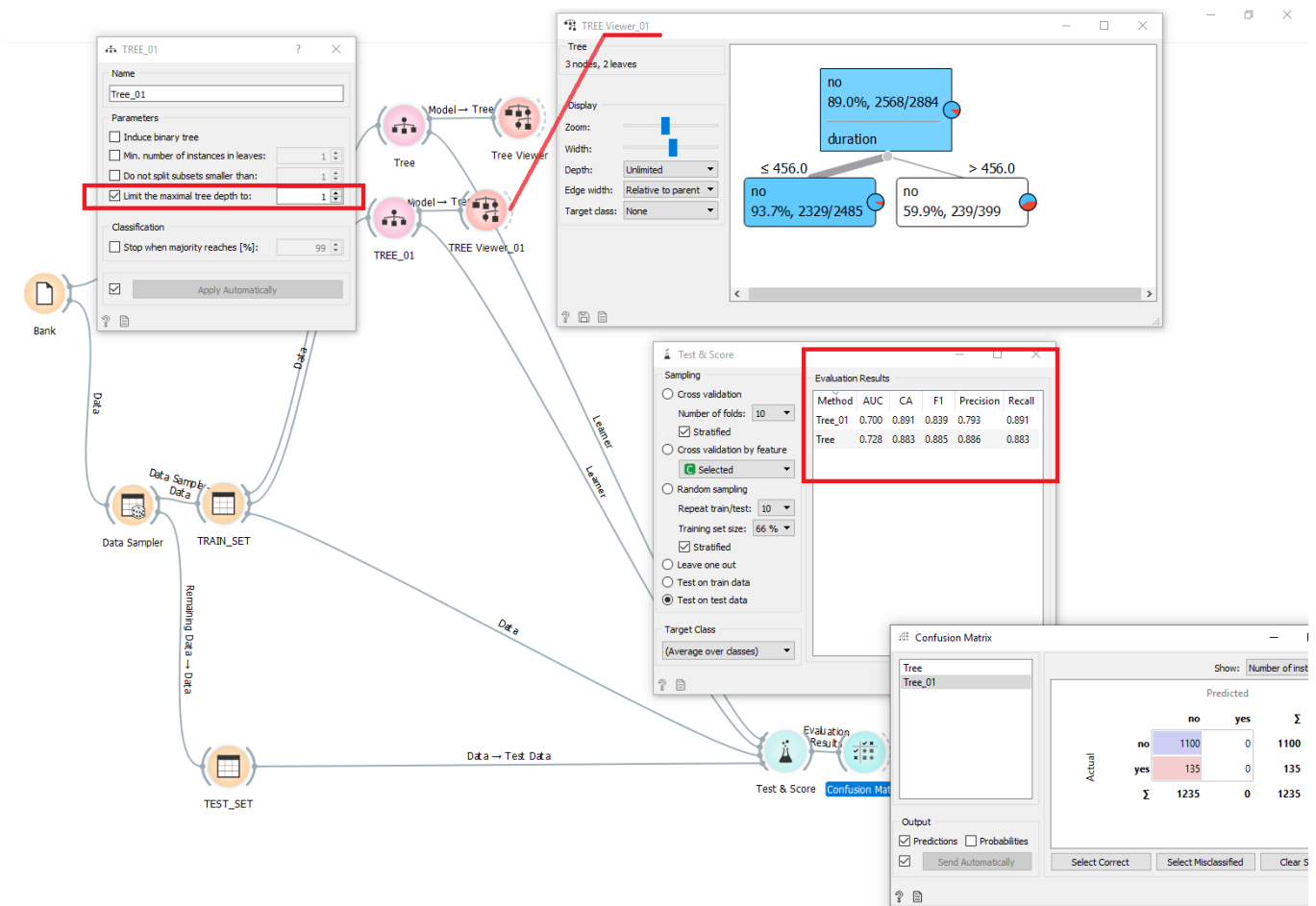
- conéctelo a la salida del TREE_01
- Cámbiele el nombre por TREE Viewer_01
- Debería ver un árbol de un solo nivel como se observa en la figura siguiente.
- Observe que ha evaluado que la mejor variable para comenzar el árbol es **duration**, esto nos indica que **esta variable es la que posee mayor cantidad de información para pronosticar**, esto es muy importante porque no todos los métodos de IA son tan claros al mostrarnos estos resultados como lo es el Árbol de Decisión.

Evaluación:

- Para evaluar qué tan bien anda este modelo de un solo nivel de profundidad, **simplemente conecte la salida del modelo de árbol TREE_01 a la entrada del control Test&Score que habíamos creado con anterioridad** y fíjese que ha agregado los resultados de Tree_01 en la zona de la derecha donde presenta los resultados. En la figura le hemos puesto un recuadro rojo.
- El valor de **Exactitud (Accuracy)** que muestra en la columna **CA (Classification Accuracy)** es inclusive superior al del árbol completo! Sin embargo el F1 es menor. Cómo puede ser que con un árbol de 1 solo nivel de profundidad tengamos una accuracy superior? La respuesta la encontrará en ...

Confusion Matrix

- En el mismo control para la matriz de Confusión que habíamos creado con anterioridad, fíjese que ahora en el panel de la izquierda nos muestra los dos árboles, el Tree y el Tree_01, seleccione el Tree_01 y verá qué aprendió de los datos este árbol de sólo un nivel.
- Si presta atención verá que aprendió ... a pronosticar cualquier observación como No!
- Como en este caso había muchas más observaciones con valor No que con valor Si, la Accuracy le da bien, pero fíjese que el F1 denota que algo ha empeorado. Pero eso le comentábamos que el Score F1 es un mejor indicador de qué tan bien o mal está nuestro modelo.

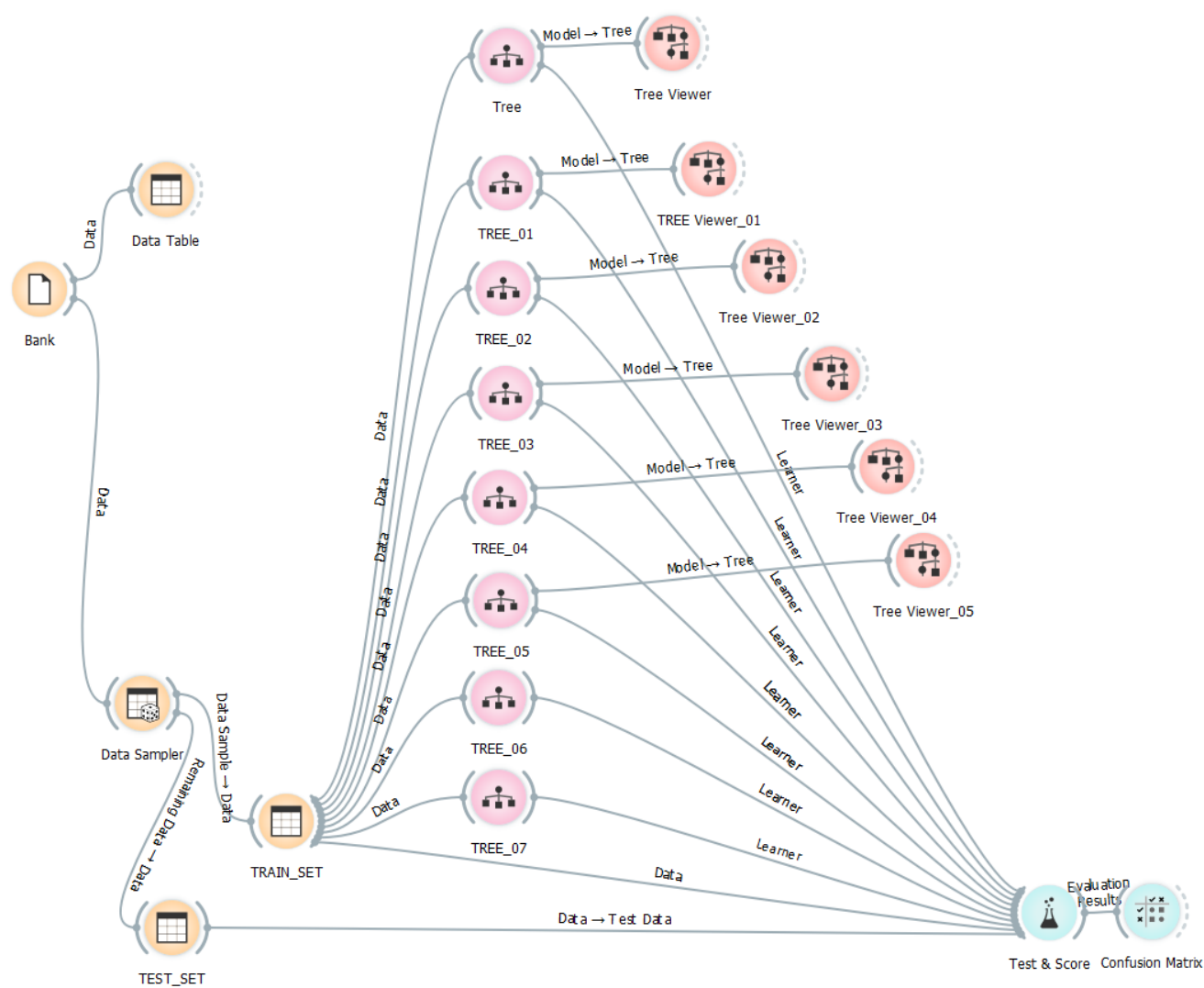


Ahora vamos a agregar más árboles de nivel 2, 3, 4 y 5 al mismo flujo.

A cada uno de ellos los llamaré con algún nombre que nos permita distinguirlos, yo voy a usar Tree_02 para nivel 2, etc y a los visores también les pondré _02, etc.

Finalmente mi flujo de trabajo en Orange3 quedó como se muestra en la siguiente figura, después de reordenar un poco algunos controles. Después de cierto nivel de profundidad ya ignoré el Tree Viewer para mejorar la visualización.

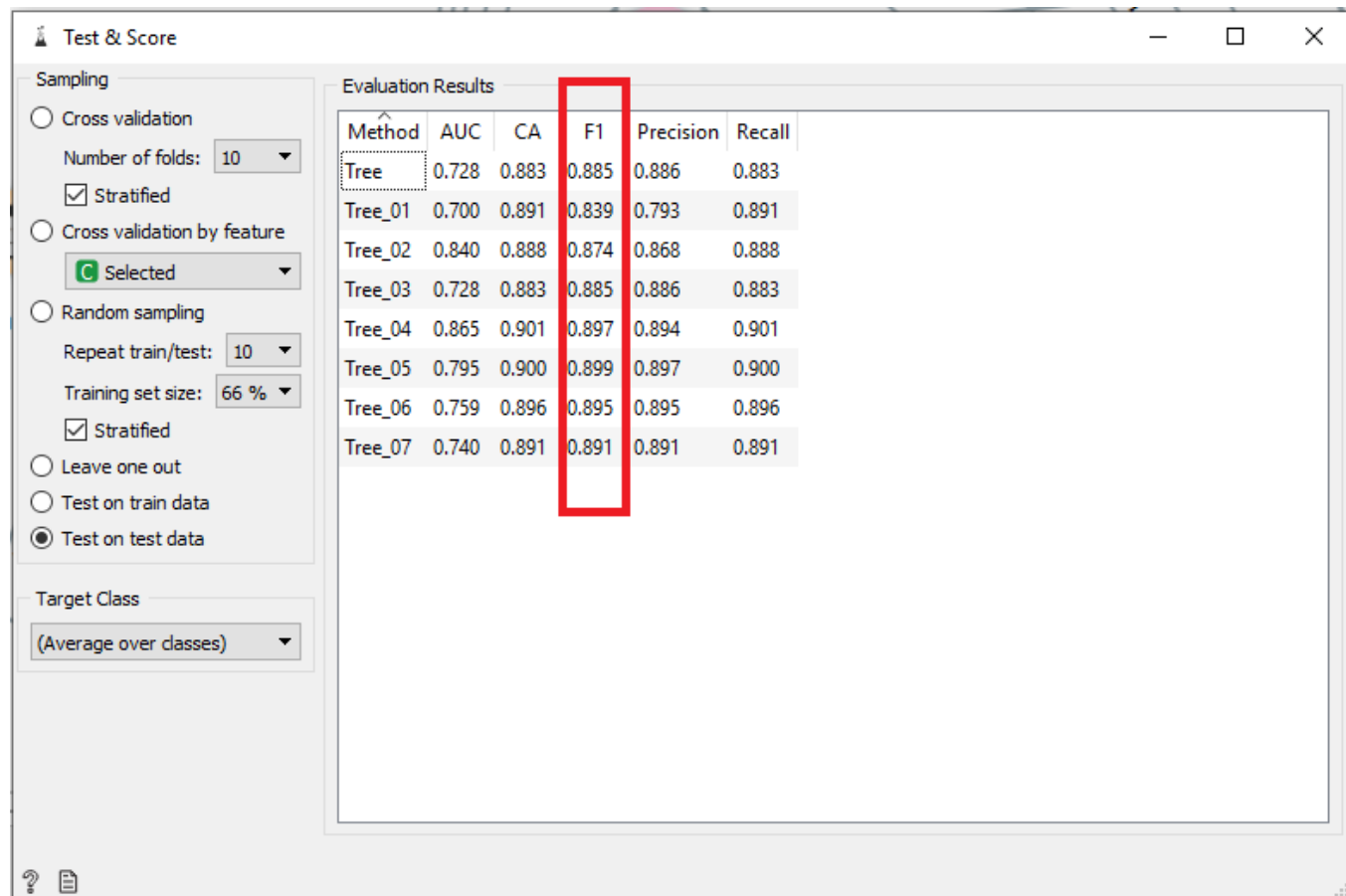
El archivo completo de Orange3 puede encontrarlo con el nombre **Clase_04_bank_mkt_4.ows**



Evaluación de Modelos

Estamos en condiciones de ver qué tan bien se han comportado los árboles de distintos niveles de profundidad, haga doble clic en el control **Test&Score** y vemos qué tenemos, prestemos atención al valor de F1 que es nuestra mejor métrica de evaluación.

En la siguiente figura hemos ordenado alfabéticamente los árboles, el primero que figura es de profundidad 100 y el resto aparecen ordenados según su profundidad.



Method	AUC	CA	F1	Precision	Recall
Tree	0.728	0.883	0.885	0.886	0.883
Tree_01	0.700	0.891	0.839	0.793	0.891
Tree_02	0.840	0.888	0.874	0.868	0.888
Tree_03	0.728	0.883	0.885	0.886	0.883
Tree_04	0.865	0.901	0.897	0.894	0.901
Tree_05	0.795	0.900	0.899	0.897	0.900
Tree_06	0.759	0.896	0.895	0.895	0.896
Tree_07	0.740	0.891	0.891	0.891	0.891

El problema del Overfitting

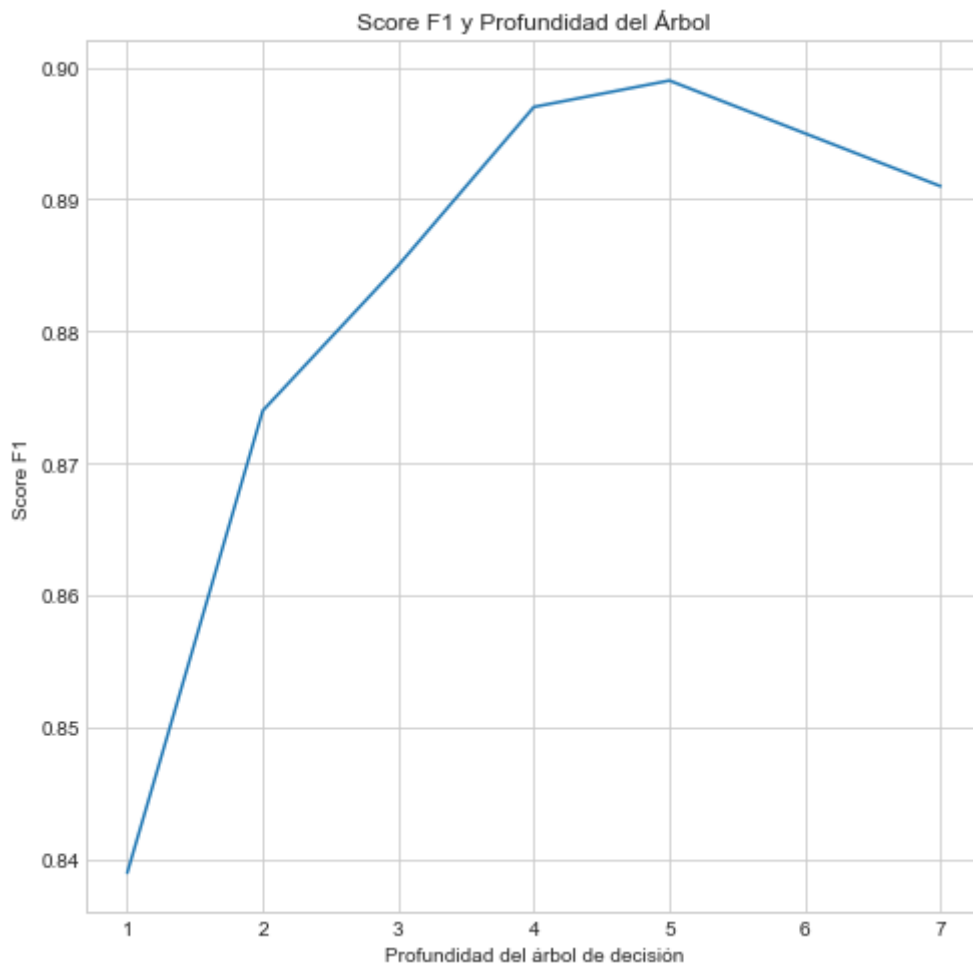
Sorpresa!

- Lo primero sorprendente es que el árbol de mayor profundidad **no es el que tiene mejor F1!**
- En realidad el mejor de todos los árboles es el de profundidad 5!

Podemos ver esta información en el siguiente gráfico, en el eje horizontal hemos ubicado la profundidad de nuestro árbol y en el vertical el valor de F1:

Out[8]:

[<matplotlib.lines.Line2D at 0x1f2b8e9e0b8>]



En la figura anterior queda claro que **el valor de F1 va creciendo desde el nivel de profundidad 1 hasta el 5 valor en el cual alcanza su máximo valor y luego comienza a decrecer.**

- Su primer pregunta debe ser, este es un comportamiento real, habitual, o es un truco en los datos que nos dio el profesor?.
- Respuesta: le recuerdo que este conjunto de datos es **real!** Este comportamiento es habitual, es más, es lo **que debemos esperar que suceda!**

Por qué?

Razonemos lo siguiente:

- Comenzamos con un conjunto de datos, que son **sólo una fracción de todos los casos que pueden darse en la realidad** (y en esta profesión "nunca" contaremos con todos los casos de la realidad!).
- Hemos entrenado a nuestro modelo con una parte de estos datos: el Train Set. Es decir que nuestro modelo ha aprendido sólo de la información que hay en el Train Set, que por supuesto es sólo una fracción de todas las observaciones que *podrían* ocurrir.
- Como nosotros estamos interesados en que nuestro modelo **pronostique lo mejor posible justamente en las observaciones que no tenemos**, evaluamos el modelo frente al Test Set, que es nuestra forma de simular las miles y miles de observaciones que no tenemos porque justamente no formó parte del entrenamiento.
- A medida que aumentamos la profundidad del Árbol de Decisiones, **nuestro modelo aprende más y más y más del limitado conjunto de observaciones del Train Set**, pero el Train Set representa sólo una fracción de los comportamientos posibles en el mundo exterior a nuestras observaciones.
- Es por esto que a medida que vamos aumentando el nivel de profundidad del árbol, **el modelo se va pareciendo más al Train Set** y al principio también va pronosticando mejor en el "resto del mundo", pero como el Train Set no es idéntico al "resto del mundo" en algún momento comienzan a aparecer las diferencias y nuestro modelo comienza a adaptarse muy bien al Train Set pero se comienza a apartar del "resto de mundo".

Este fenómeno, que se produce cuando nuestro modelo se parece más al Train Set que lo que debería para mejorar su poder de pronóstico sobre el resto del mundo es decir la observaciones desconocidas; se denomina **OVERFITTING** lo que podemos traducir como **SOBREAJUSTE**.

Fronteras de Decisión para distintas profundidades de árbol

Para poder visualizar la forma en la cual los diferentes árboles de decisión dividen el plano de las features o variables tendremos que tomar un ejemplo con sólo 2 variables o features y la variable target y que deseamos clasificar.

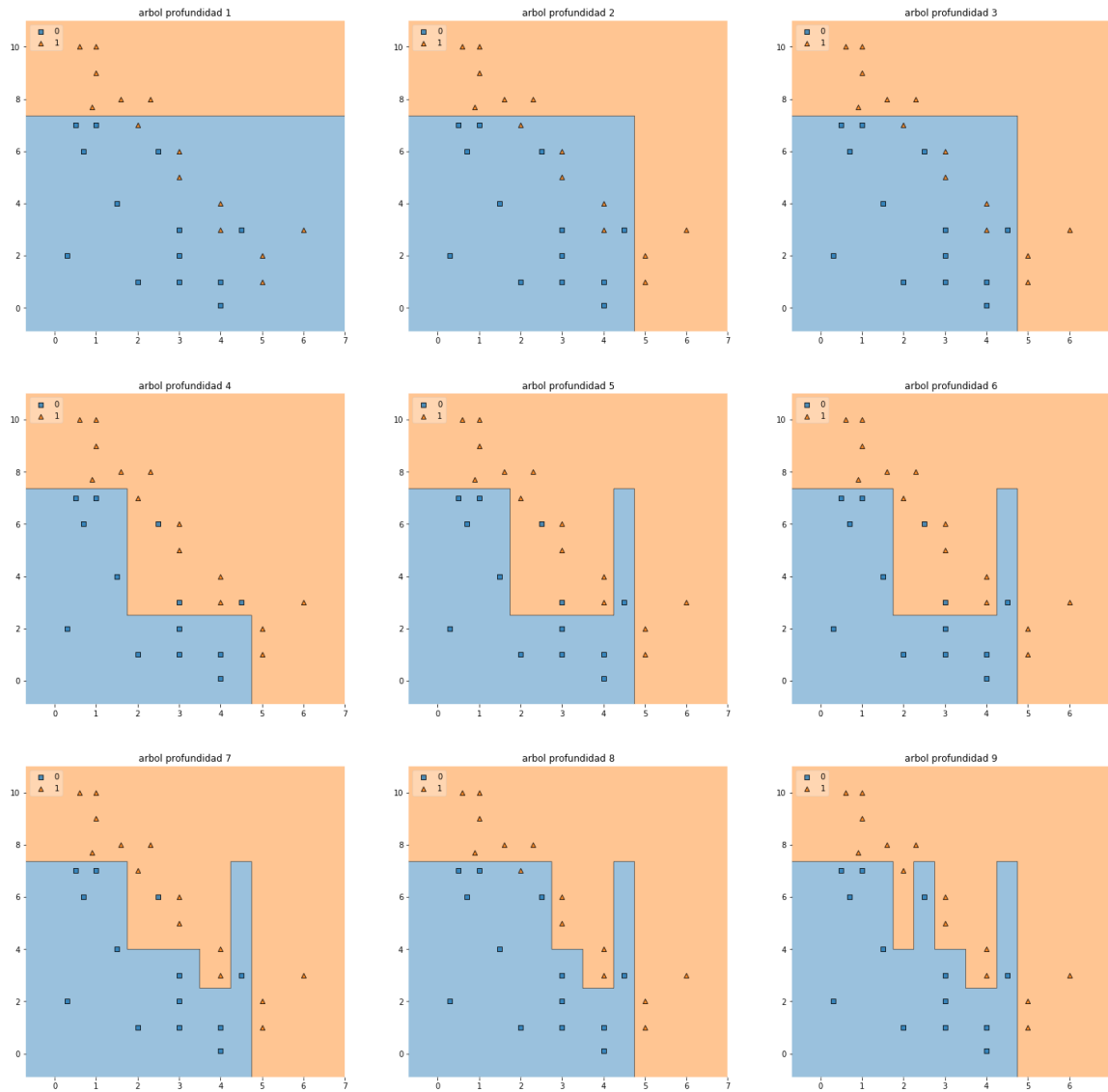
Veremos cómo árboles de diversas profundidades resuelven el problema en el Train Set para formarnos una idea gráfica del underfitting y del overfitting.

Supongamos un caso con dos variables, x_1 y x_2 y un target y que asume las etiquetas 0 y 1:

Out[90]:

	x1	x2	y
0	0.5	7.0	0
1	4.0	0.1	0
2	0.3	2.0	0
3	0.7	6.0	0
4	1.0	7.0	0
5	1.0	9.0	1
6	0.6	10.0	1
7	1.0	10.0	1
8	1.6	8.0	1
9	1.5	4.0	0
10	2.0	1.0	0
11	2.0	7.0	1
12	2.3	8.0	1
13	3.0	1.0	0
14	3.0	2.0	0
15	3.0	3.0	0
16	3.0	5.0	1
17	3.0	6.0	1
18	4.0	1.0	0
19	4.0	3.0	1
20	4.0	4.0	1
21	5.0	1.0	1
22	5.0	2.0	1
23	6.0	3.0	1
24	2.5	6.0	0
25	4.5	3.0	0
26	0.9	7.7	1

Vemos cómo serían las Fronteras de Decisión para árboles de distinta profundidad, en este caso desde profundidad 1 a profundidad 9 y podrá observar visualmente cómo para las menores profundidades se puede observar underfitting y para las mayores profundidades un overfitting:



La impresión visual es que hasta la profundidad 3 parecería ser que los árboles están cometiendo **underfitting**, para las profundidades de 5 en adelante ya se ajusta tanto a los datos del Train (hasta parece un poco **forzado**) que podríamos pensar que está cometiendo **overfitting** (fíjese que para profundidad 9 el ajuste a los datos del train es perfecto); así que por la impresión visual pensaríamos que **un buen árbol que generalice en los casos desconocidos podría tener profundidad 4**, para poder decir más deberíamos compararlos numéricamente por ejemplo comparando la Accuracy o el F1 de todos ellos.

Conclusión

Como nuestro interés es que el modelo pronostique lo mejor posible en las observaciones desconocidas (se suele decir que buscamos que **generalice** lo mejor posible), ahora que conocemos el fenómeno del **overfitting** sabemos que no será conveniente generar un árbol de profundidad infinita, sino que deberemos entrenar con el Test Set árboles de profundidad creciente hasta encontrar aquel que nos provea el mejor valor de F1 testeando en el Test Set.

Como cuanto más profundo es el árbol, más se parecerá al limitado Train Set si, por ejemplo, encontramos que el F1 aumenta hasta un nivel de profundidad de digamos 4, luego baja y luego sube nuevamente hasta alcanzar un valor máximo similar al del nivel 4, digamos en el nivel 15; preferiremos adoptar el árbol de nivel 4, porque seguramente generalizará mejor en la realidad.

Ignorar este comentario, es para el profesor: Podríamos agregar como ejercicio el de Iris para que lo entreguen y que lo comencé a desarrollar en el video de la Clase 3 a partir de 1:04:22 pero se perdió parte de la grabación, así que no está terminado.

Cuestionario o Autoevaluación (parte teórica)

Ver archivo SP03_Clase_04_Autoeval o según indique el profesor.