Paso 4 Modelado y Evaluación

Ver Sección 7.5 del document adjunto, hasta 7.5.3

Desbalance de clase

Observar que la distribución del atributo o clase objetivo, ***is_buyer***, está desbalanceada, lo cual es bastante común: hay muchos más no-compradores que compradores (sólo cerca del 7% son compradores). Esto puede afectar significativamente a los algoritmos de ML, ya que se puede lograr una alta exactitud simplemente prediciendo que todos los clientes son no-compradores.

Hay al menos dos posibles métodos para tratar este problema: ponderación y balanceo.

1. ***Ponderación*** introduce un atributo especial en los datos, que RM reconoce por su rol. Es un atributo numérico que asocia cada ejemplo (en este caso el cliente) con un factor de peso, permitiendo así asignar mayor peso a la clase minoritaria (en este caso, los compradores).

   El operador de RM ***Generate Weight (Stratification)*** puede ser usado para la creación automática de un atributo de ponderación. Importa y estudia el proceso adjunto "*03 XValidateWeighted".*

   El operador distribuye una cantidad total de peso entre los ejemplos de la entrada, de forma tal que para cada clase la suma de los pesos de todos los ejemplos es igual.

2. ***Re-balanceo.*** Se crea un subconjunto especial del dataset en el que ambas clases de la variable objetivo aparecen aproximadamente con la misma frecuencia (compradores y no compradores). Este subconjunto se utiliza solamente para entrenar o crear el modelo, pero no para probarlo ni desplegarlo. Este enfoque se ilustra en el proceso de ejemplo adjunto "*04 XValidateBalanced*". El único cambio con respecto al anterior es que el operador *Sample* reemplaza al *Generate Weight (Stratification)* en el subproceso de entrenamiento. Lo utilizamos para tomar aleatoriamente 10% de los no-compradores en los datos de entrenamiento, pero todos los compradores. Esto resulta en una distribución aproximadamente equilibrada de compradores y no compradores.

Evaluación Sencilla de los modelos

- Observa que en ambos modelos se ha utilizado un operador Cross *Validation*, y tomar nota de los parámetros del mismo. En ambos modelos usamos un operador *Performance (Binomial Clasification)* que nos permite emitir las distintas métricas propias de la clasificación binaria.

- Compara las matrices de confusión creadas por los dos procesos de ejemplo. Cada una es el resultado de sumar las matrices creadas por el operador *X-Validation* en cada proceso. Otras métricas de rendimiento emitidas por el operador *X-Validation* son los promedios de sus corridas individuales. Se puede inspeccionar estas corridas individuales utilizando un operador Log, que se ejecuta en cada subproceso de test, agregando los valores listados en el parámetro *log* al final del archivo especificado.
- Dos métricas importantes a tener en cuenta en aplicaciones como ésta son "*recall*" y "*precission*" de la predicción de compradores (o sea, la clase positiva). ¿Cuál es más importante?. Depende de la aplicación… En ésta, el objetivo es encontrar los no-compradores que son similares a los compradores. Entonces el conjunto de compradores predichos como compradores debería incluir una cantidad de compradores reales, y no debe ser muy pequeño, que es lo que probablemente suceda al buscar alta precisión sobre los datos reales. Entonces deberíamos poner más énfasis en alcanzar valores de *recall*, a la vez que el valor de *precisión* debería mantenerse significativamente mayor que si la predicción fuera aleatoria. Esto significa que debe mantenerse mayor que la relación general de ejemplos positivos (7% en nuestros datos).
- Con datos reales puede suceder que no podamos alcanzar valores de precisión mayores que dos o tres veces esta relación, dependiendo de la aplicación. En cualquier caso, esto aún significaría que las predicciones son al menos dos o tres veces mejores que si fueran aleatorias, y no debería impedir el despliegue del modelo.