BUSINESS ANALYTICS

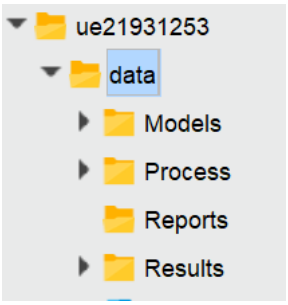
Módulo III: EDA + ETL + MODELING:

PROFESOR : CHRISTIAN VLADIMIR SUCUZHANAY

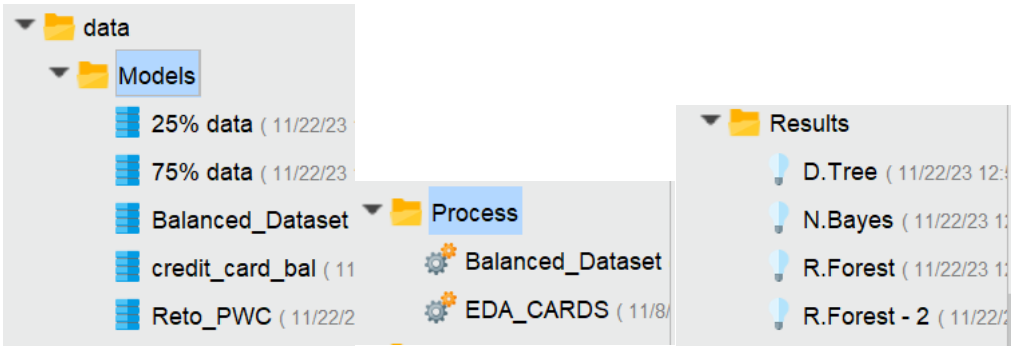
ALUMNO: MIGUEL ANGEL ALCON GALAN

ÍNDICE:

En primer lugar, se llevó a cabo la creación de las nuevas carpetas requeridas para la consecución de la actividad:



Tras la consecución de la actividad las carpetas quedaron dispuestas de la siguiente manera:

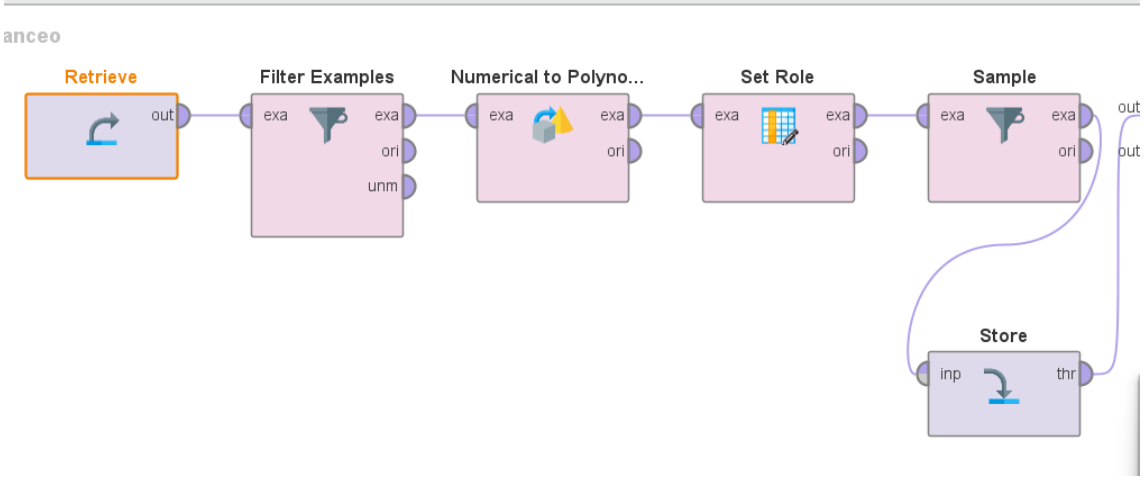


Una vez expuestas las carpetas con la información recogida en cada una de ellas, se dispone a explicar paso por paso como se llevó a cabo la consecución de la tarea:

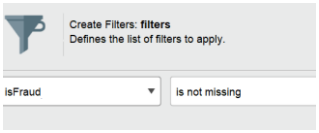
En primer lugar se presentó el problema derivado de la incapacidad del dispositivo de funcionar manejando tal cantidad de datos. En adición, los datos se encontraban tremendamente

desbalanceados, habiendo una desproporción brutal entre la cantidad de información disponible para el caso de ¨no fraude¨, respecto a los de fraude. Derivando en problemas de fiabilidad en la toma de la decisión, ya que los resultados arrojados siempre serían de ¨no fraude¨ debido a esa descompensación de los datos.

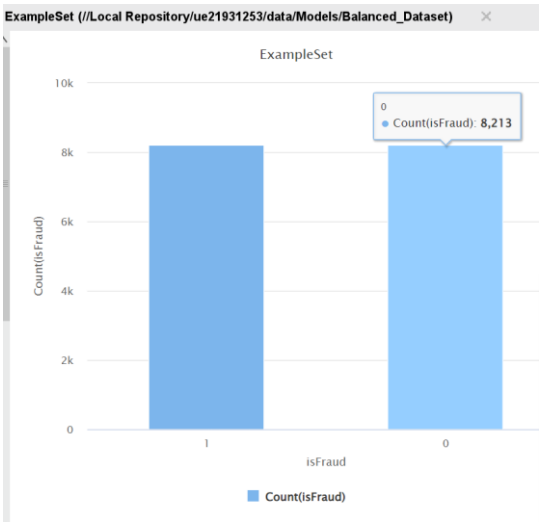
A consecuencia de ello, se optó por llevar a cabo un balanceo de los datos mediante undersampling.



En primer lugar se comenzó utilizando retrieve, tras haber importado previamente la base de datos original, con el fin de dar a entender a RapidMiner la base de datos con la que se procederá a trabajar. A continuación se recurrió al Filter Examples con el fin de eliminar posibles datos en blanco en el caso de que los hubiera.

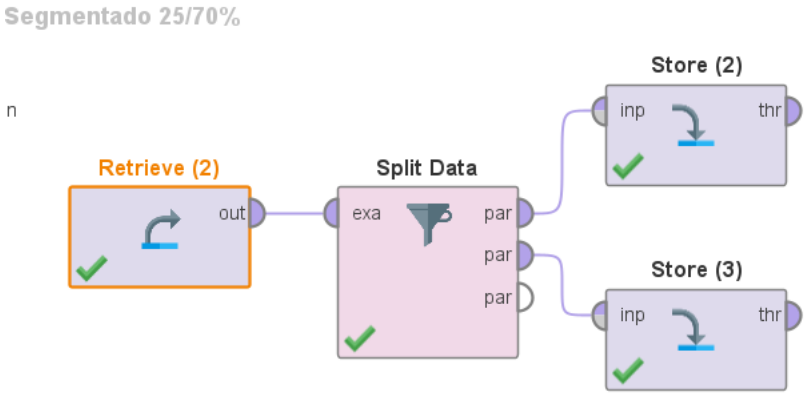


Se cambió el tipo de datos de numerical a polynomical, y se etiquetó el isfraud como la información a balancear, gracias al set role. Finalmente con la función sample terminamos de balancear nuestros datos de tal forma que obtuviésemos de forma uniforme tanto información de fraude como de no fraude, a su vez este sampleo nos permitió poder manejar los datos sin que nuestro ordenador acusase tantos problemas.

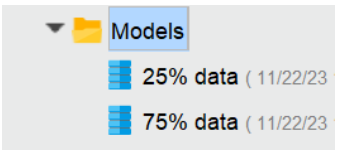


El resultado fue exitoso como se puede comprobar en la gráfica obtenida en resultados, visualizations.

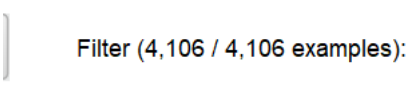
Sin embargo, previo paso a llevar a cabo la consecución de los modelos, se hace necesario dividir los datos en dos partes diferenciadas, una muestra de gran tamaño con la que entrenar al algoritmo, y otra muestra de menor tamaño, y desconocida por el mismo, con la que comprobar hasta que punto nuestro modelo es funcional.



En primer lugar se comenzó utilizando la función retrieve, donde seleccionamos la base de datos balanceada. Para ello se recurrió al Split data para dividir de forma homogénea nuestros datos en 75% y 25% respectivamente. El 75% será usado para entrenarlo, y el 25% para comprobar que ha aprendido realmente. Una vez finalizado dispondremos de dos bases de datos con las que llevar a cabo, ahora sí, la consecución de los 3 modelos



.

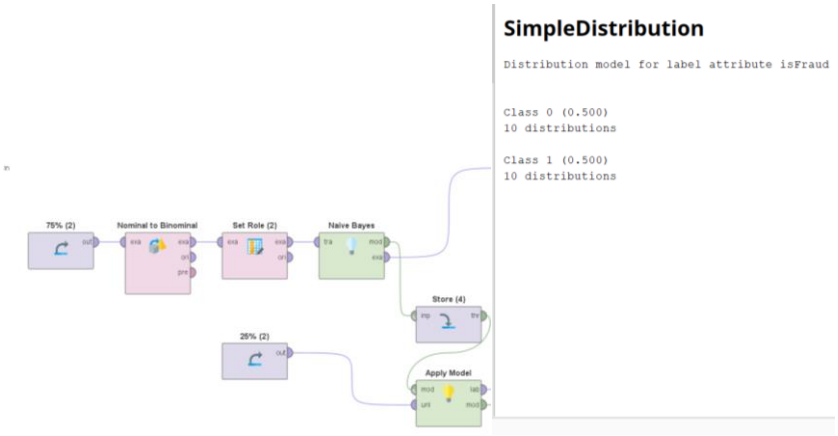


25%

Filter (12,320 / 12,320 examples):
75%

Una vez llevada a cabo la partición de los datos de la base de datos sampleada, se procedió a la elaboración de los diferentes modelos:

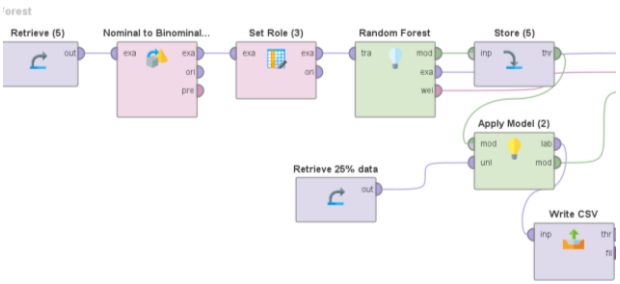
Modelo 1: Naive Bayes



Ventajas: rápido y eficiente con grandes conjuntos de datos.

Desventajas: no es el modelo más fiable como hemos podido comprobar.

Modelo 2: Random Forest: el profesor lo recomendó como una opción muy buena para llevar a cabo este tipo de tareas.



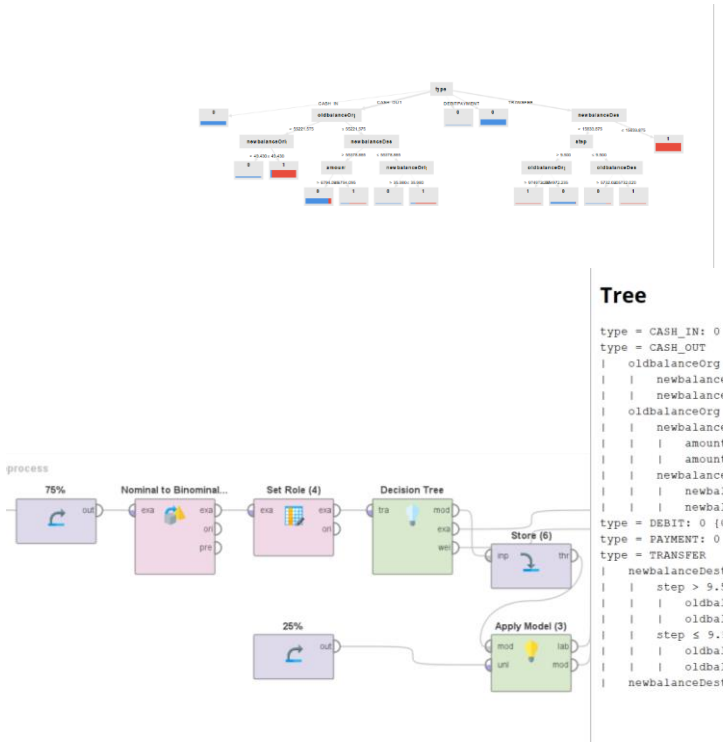
Ventajas: Debido a su capacidad para manejar conjuntos de datos complejos, se trata de la opción ideal de cara a detectar fraude.

Desventajas: requiere de una gran cantidad de recursos, a consecuencia de ello el ordenador fue incapaz de ofrecer los resultados debido a que crusheaba.

Modelo 3: Árbol de decisión: se tomó la decisión de utilizarlo debido a que en clase se había comentado con anterioridad su utilizad. La cantidad de ramas del árbol utilizadas se redujo a 5 debido a problemas de rendimiento.

Ventajas: fácilmente interpretables, se puede apreciar con claridad como va tomando decisiones en función de diferentes situaciones.

Desventajas: no son tan fiables como los random forest.



Tree

```
type = CASH_IN: 0 {0=1347, 1=0}
type = CASH_OUT
|   oldbalanceOrg > 55221.575
|   |   newbalanceOrig > 49.430: 0 {0=219, 1=1}
|   |   newbalanceOrig ≤ 49.430: 1 {0=163, 1=2639}
|   oldbalanceOrg ≤ 55221.575
|   |   newbalanceDest > 55378.865
|   |   |   amount > 6794.095: 0 {0=1751, 1=209}
|   |   |   amount ≤ 6794.095: 1 {0=27, 1=52}
|   |   newbalanceDest ≤ 55378.865
|   |   |   newbalanceOrig > 35.980: 0 {0=21, 1=0}
|   |   |   newbalanceOrig ≤ 35.980: 1 {0=41, 1=166}
type = DEBIT: 0 {0=34, 1=0}
type = PAYMENT: 0 {0=2038, 1=0}
type = TRANSFER
|   newbalanceDest > 15833.875
|   |   step > 9.500
|   |   |   oldbalanceOrg > 974972.235: 1 {0=0, 1=3}
|   |   |   oldbalanceOrg ≤ 974972.235: 0 {0=514, 1=11}
|   |   step ≤ 9.500
|   |   |   oldbalanceDest > 5732.020: 0 {0=4, 1=1}
|   |   |   oldbalanceDest ≤ 5732.020: 1 {0=0, 1=7}
|   newbalanceDest ≤ 15833.875: 1 {0=1, 1=3071}
```

Python:

Python_crash_cou | Python_crash_cou | Untitled1.ipynb | Launcher | Reto_PWC.ipynb | original sampleado

Code | Python 3 (ipykernel)

```
[ ]: import pandas as pd

[38]: pwc = pd.read_csv('original sampleado.csv')

[ ]: pwc

[ ]: pwc.describe()

[ ]: pwc.info () #el parentesis significa que es una funcion

[42]: pwc.columns #te da del tirón todas las columnas de datos para saber que clase de info se ofrece en la tabla.

[42]: Index(['step', 'type', 'amount', 'nameOrig', 'oldbalanceOrg', 'newbalanceOrig',
       'nameDest', 'oldbalanceDest', 'newbalanceDest', 'isFraud',
       'isFlaggedFraud'],
      dtype='object')

[ ]: pwc.corr #correlación: tiene 3 resultados, 1,-1, 0. Mide la relación existente entre dos variables. En que medida
     # una influye en la otra. En un contexto determinado. Smote upsampling es de los mejores para hacer sampling.
```

Useful commands

pwd = current directory ls = list files cd = change directory mkdir = make a new directory

Se exportó la base de datos creada en Rapid Miner mediante el uso de ¨Write CSV¨, una vez exportado se importó a Pyhton y mediante click derecho, copy path, se insertó el nombre del archivo en el código, read_csv. El significado de cada una de las partes se encuentra dentro del propio código precedido de #.