MEMORIA

ANÁLISIS “CUSTOMER JOURNEY” PARA CLIENTE DE BANCA

Máster Data Science 12ª Edición Óscar Martínez Aceña

Introducción

El objetivo de mi TFM consiste en iniciar, desde el equipo de Data Science, una colaboración con el Área de Negocio del Banco para analizar si a partir de los datos de los que disponemos y que se pueden encontrar en la siguiente ruta:

https://data.world/lpetrocelli/czech-financial-dataset-real-anonymized-transactions

Se puede definir el "customer journey" o incrementar el conocimiento de los clientes para alguno o varios productos. Es decir, si se puede conocer qué operativa realizada por un cliente puede indicar la contratación de determinados productos bancarios.

A nivel personal me interesan los problemas que puedan tener un reflejo claro en la Cuenta de Resultados y considero que abordar el tema de este TFM puede ser un punto de partida. Además, al tratarse de un problema en una entidad financiera abordar el problema desde el punto de vista de conocer la operativa de cliente, para poder realizar mejores campañas comerciales, no me limita a tener que utilizar modelos "auto explicativos" como Regresión y Decision Trees (sí que podría pasar en el caso de modelos para concesión de productos: préstamos, seguros,..).

Los ficheros contenidos en el REPO son:

- Memoria TFM (este documento)

- TFM Data Set análisis productos.R: Utilizando R me conecto a la web anterior y descargo todos los ficheros disponibles (contienen información sobre productos, clientes, transacciones, datos socio económicos, …). Combinando la información recogida en estos ficheros se genera "DFTenenciaProductos.csv" y "DFTenenciaProductos_wo_LoanPayment.csv", que son los ficheros con los que vamos a analizar los datos.

- DFTenenciaProductos.csv: Se incluye por si se desea descargar los datos desde GitHub

- DFTenenciaProductos_wo_LoanPayment.csv: Se incluye por si se desea descargar los datos desde GitHub. Este fichero se deriva a partir del anterior eliminando el efecto de las transacciones con categoría "Sym_LoanPayment". Ampliamos información más adelante y de momento nos centramos en "DFTenenciaProductos.csv"

- Tenencia del producto préstamo Banco Checo_wo_LoanPayment.ipynb: A partir del fichero "DFTenenciaProductos_wo_LoanPayment.csv" se analiza con Python en un Jupyter Notebook, la tenencia del producto préstamo y se presentan las conclusiones del trabajo.

- Tenencia del producto préstamo Banco Checo.ipynb: A partir del fichero "DFTenenciaProductos.csv" se analiza con Python en un Jupyter Notebook, la tenencia del producto préstamo y se presentan las conclusiones del trabajo.

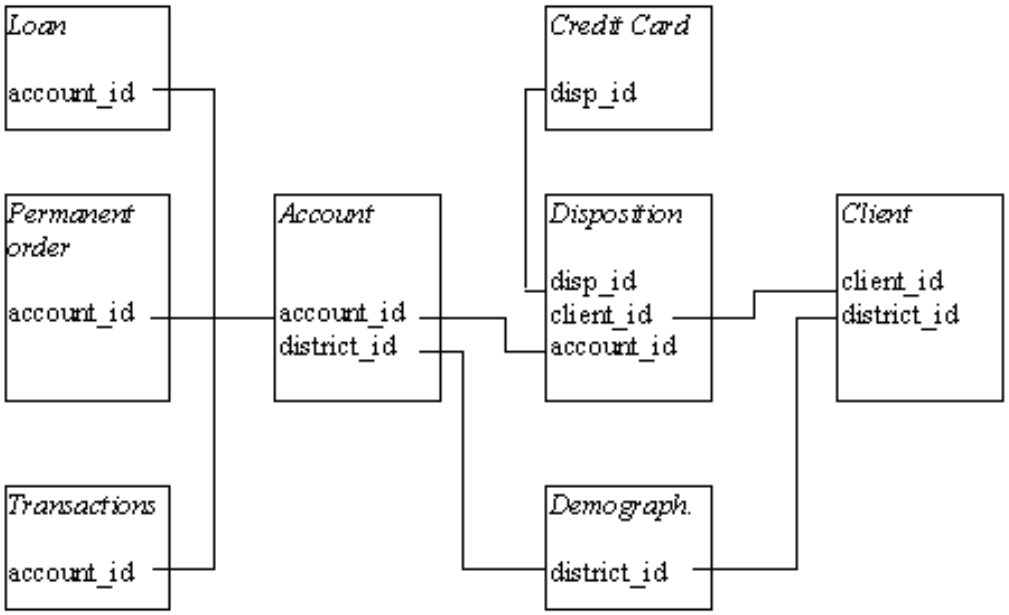
Descripción inicial de los datos

Los ficheros contienen datos reales de clientes de un banco Checo entre 1993 y 1998. Los campos de texto de los ficheros, por lo general, no están en inglés, por lo que a menudo va a ser necesario traducir dichos campos para poder interpretar su significado (las traducciones se encuentran en la web indicada anteriormente, en la descripción de cada fichero).

Por ejemplo la información del fichero account la podemos encontrar en:

https://data.world/lpetrocelli/czech-financial-dataset-real-anonymized-transactions/workspace/file?filename=account.csv

Una descripción visual de los ficheros disponibles es la siguiente:



Como podemos ver hay distintos tipos de datos:

- *Específicos de productos*: Accounts (cuentas corrientes), Credit Card (tarjetas de crédito), Permanent order (domiciliaciones / pagos domiciliados) y Loans (préstamos)

- *Específicos de clientes*: Client (clientes) y Disposition (nos va a permitir relacionar al cliente y los productos)

- *Transaccionalidad realizada por las cuentas* (Transactions)

- *Datos socioeconómicos*: Demograph (datos de 77 distritos geográficos)

La estrategia seguida para aglutinar toda la información consiste en crear un DataFrame, que finalmente llamaremos “DFTenenciaProductos”. Este DataFrame parte de la información de las cuentas corrientes (Accounts) y la va enriqueciendo con la información que consideramos relevante del resto de ficheros.

Generación dataset “DFTenenciaProductos.csv”

Los paquetes con los que vamos a trabajar son: “tidyverse”, "DataExplorer", "lubridate", "hashmap" y "dplyr". Están enumerados al inicio del código

Los ficheros que vamos a descargar de la web anterior están en formato csv y son por lo general bastante ligeros, a excepción del fichero de transacciones que ocupa unos 68MB, que contiene más de 1 millón de transacciones y que tarda alrededor de 5 minutos en descargarse.

Para cada fichero que descarguemos la metodología de trabajo que vamos a seguir consiste en:

- Analizar la estructura del fichero (observaciones y variables)
- Buscar missings y duplicates: No hay duplicates en los ficheros y sólo hay missings en el fichero de datos socioeconómicos *(district)*
- Incorporamos al DataFrame generado a partir del fichero *account* (que nos sirve como base), los datos que consideramos relevantes del fichero que estamos tratando.

Iniciamos el proceso con el fichero *account* (cuentas corrientes):

Lo que se cuenta a partir de aquí puede seguirse en paralelo con el código del fichero “TFM Data Set análisis productos.R”

Este fichero aporta número de cuenta, fecha de apertura de la cuenta, distrito (más tarde lo ligaremos con el fichero *district*) en el que estaba la sucursal en la que se abrió la cuenta y frecuencia con la que se generan extractos de la cuenta (este campo no está en inglés y se traduce en el código. Por ejemplo: "POPLATEK MESICNE" significa periodicidad mensual).

De este fichero incorporamos todas sus variables al dataset final. Hay 4.500 cuentas en total.

Fichero *client* (datos personales de los clientes):

Este fichero contiene las variables de número de cliente, distrito y birth_number (la explicamos a continuación), pero no contiene variables para poder cruzarlo con el DataFrame *account*, necesitaremos del fichero *disp* (de disposiciones / claves de propiedad) para poder hacerlo.

La variable birth_number recoge la fecha de nacimiento y el sexo de cada cliente, ya que para los hombres está en formato YY/MM/DD y para las mujeres en formato YY/MM+50/DD. Y por tanto a partir de las posiciones que indican el mes de nacimiento se deduce el sexo del cliente.

Fichero *disp* (claves de propiedad /disposición):

Contiene las variables disp_id (clave de disponibilidad, es un identificador), client_id (el número de cliente que hemos visto en el fichero anterior), account_id (el número de cuenta) y type que indica el tipo de disposición del cliente sobre la cuenta (si es OWNER puede realizar todo tipo de operativas y si es DISPONENT (autorizado) no puede

realizar todo tipo de operativas (por ejemplo dar de alta una domiciliación o solicitar un préstamo). Por tanto, va a haber cuentas con 2 tipos de usuarios: propietarios y autorizados y cuentas que únicamente tienen propietarios.

A partir de este fichero y utilizando 2 diccionarios, incorporamos al DataFrame *account* el ID del propietario de la cuenta y el ID del autorizado (disponent) si lo tiene (estos datos estaban en el fichero *disp)*. Del fichero *client* incorporamos el sexo y la fecha de nacimiento de propietario y autorizado (si lo hay). Sólo 869 cuentas tienen autorizado y por tanto esta información se añadirá únicamente a una parte de las cuentas. En el resto estas variables aparecen como N/A.

Fichero *loan* (prestamos contratados Enel periodo por el perímetro de clientes / cuentas con el trabajamos. Los préstamos pueden estar en vigor o ya finalizados a la fecha de extracción de la información):

Contiene las variables loan_id, account_id (nos servirá para realizar un merge), date (fecha de concesión del préstamo), amount (importe concedido inicialmente), duration (plazo inicial del préstamo), payments (cuota mensual del préstamo) y status (es una variable categórica que indica si el préstamo está o no al corriente de pago y si el préstamo está vivo o se ha amortizado completamente).

Los 4 niveles de status son:

- A: para préstamo finalizado y al corriente de pago. Para entenderlo mejor lo codificamos como “OK”
- B: para préstamo finalizado pero moroso. Lo codificamos como “NPL” (de Non Performing Loan)
- C: para préstamo en vigor y al corriente de pago. Lo codificamos como “OK_not_fin” (not_fin de not finished)
- D: para préstamo en vigor pero en situación de mora. Lo codificamos como “NPL_not_fin”

Adicionalmente, de la variable status generamos una variable binaria “status_loan_bin” que contenga si el préstamo ha incurrido en situación de mora o no.

Se incorporan todas las variables del fichero *loan* + “status_loan_bin” al DataFrame *account* (que nos sirve como base sobre la que agregar variables). No todas las cuentas han contratado algún préstamo, de hecho sólo 682 lo han hecho y por lo tanto estas variables se añadirán únicamente a una parte de las cuentas. En el resto de observaciones estas variables aparecen como N/A.

Fichero *order* (transferencias / domiciliaciones automáticas).

Contiene las variables:

- order_id: es el identificador de la domiciliación. No se incluye en *account* porque considero que no aporta información relevante.
- account_id: nos va a servir para incluir la información en el DF *account*
- bank_to: es el banco receptor de la transferencia / domiciliación. No se incluye en *account*
- account_to: es la cuenta receptora de la transferencia / domiciliación. No se incluye en *account*

- k_symbol es la finalidad de la domiciliación. La finalidad no está en inglés, así que la traducimos. Además, para cada cuenta agrupamos en *account* cuantas domiciliaciones tiene de cada tipología/finalidad.
- amount: es el importe de la domiciliación. Sumamos el importe de cada finalidad de las domiciliaciones y lo incluimos en *account*.

Por tanto, a partir de la información contenida en *order*, generamos 2 nuevas variables por cada tipología de domiciliación (seguros, pagos del hogar, préstamos, leasings, ...) que nos indica cuantas domiciliaciones de esa finalidad ha tenido la cuenta en el periodo de recogida de los datos y qué importes se han destinado a domiciliaciones de dicha finalidad en el periodo observado.

Fichero *card* (tarjetas de crédito emitidas en el período observado). Contiene las variables:

- card_id: es el identificador de la domiciliación. No se incluye en *account* porque considero que no aporta información relevante.
- disp_id: es la clave de disposición de la tarjeta. Mediante esta variable vamos a incluir la información de este fichero en nuestro DF *account.*
- type: es la tipología de tarjeta “classic”, “gold”, “junior”. Esta información la vamos a incluir en nuestro DF *account.*
- issued: fecha de emisión de la tarjeta. Esta información la vamos a incluir en nuestro DF *account.*

Dado que tenemos la disp_id podemos saber la cuenta a la que corresponde cada tarjeta y además si el titular de la tarjeta es propietario de cuenta o autorizado (vemos que todos los titulares son propietarios de cuenta) y añadimos para cada cuenta si tiene tarjeta o no y de qué tipo (variable "owner_card_type") y en caso de contar con tarjeta, la fecha de emisión de la tarjeta ("owner_card_date"). Sólo hay 892 cuentas con tarjeta.

Fichero *district* (características socioeconómicas de las zonas recogidas en el perímetro de datos con el que trabajamos).

Este fichero contiene 16 variables socioeconómicas para 77 distritos distintos. Se incluyen todas las variables y se realiza un merge con el DF *account* por la variable “district_id”. Las variables tienen, en el fichero nombres codificados, pero en la web del dataset (https://data.world/lpetrocelli/czech-financial-dataset-real-anonymized-transactions/workspace/file?filename=district.csv) nos dan la codificación.

Este fichero tiene 2 particularidades:

- *Tiene 2 missings*: la observación/distrito 69 tiene missings en “unemployment rate'95” y “no. of committed crimes '95”. Para el caso de “unemployment rate'95” hemos calculado el dato que podría ser más aproximado que es la media de la región a la que pertenece el distrito 69 (la región es la zona “más cercana” para la que tenemos información). Para “no. of committed crimes '95” (cifra en valor absoluto) no hacemos directamente la media, ya que entonces el dato puede estar muy influenciado por el número de habitantes de otros distritos de la región, sino que calculamos un nueva variable “crimes_95_ratio” como el cociente entre “no. of committed crimes '95” y “num_inhabitants”. A partir de esta nueva variable calculamos la media de “crimes_95_ratio” de la región y multiplicando por el número de habitantes del distrito 69 obtenemos la estimación para “no. of committed crimes '95”.

- Hay algunos datos que se muestran en valor absoluto y pueden ser más comparables si los mostrarmos como ratio, dividiendo por el número total de habitantes. Entonces añadimos en el DF account los ratios “crimes_96_ratio” y “entrepreneurs_ratio”

Fichero *trans* (transacciones realizadas en el perímetro de datos que nos han facilitado).

Hasta este punto del fichero la importación de datos y el tratamiento de los datos se ha producido de forma prácticamente instantánea, pero la ejecución del código en este caso va a tardar bastante (25 min - 30 min).

Este fichero tiene 1.056.320 observaciones de 10 variables:

- trans_id: es el identificador de la transacción. No se incluye en *account* porque considero que no aporta información relevante.
- account_id: nos va a servir para incluir la información en el DF *account.*
- bank: Banco con el que se realiza la transacción (parte tercera). No se incluye en DF *account.*
- account: Cuenta con la que se interactúa en la transacción (parte tercera). La mayoría de las observaciones no están informadas. No se incluye en el DF *account*.
- date: es la fecha de realización de la transacción. No se incluye en el DF *account*.
- amount: es el importe de cada transacción. No se incluye en el DF *account*.

Las variables de las que sí vamos a incluir información en el DF *account* son:

- type: Indica si la transacción es de cargo o de abono. Hay una categoría para los abonos y dos para los cargos, ya que hay un tipo especial de cargos que es Vyber.
- operation: Indica si la operación es un envío de dinero, un ingreso, un ingreso en efectivo, un reintegro en efectivo o un reintegro con tarjeta. Adicionalmente hay otra tipología no definida y que categorizamos como nula ya que no está definida en la información de la web.
- k_symbol: Indica si la operación es cobro de pensión, un pago de seguro, un pago/comisión por saldo negativo, un cobro por intereses de un depósito, un pago por un préstamo o un pago de facturas del hogar o statements. Adicionalmente hay otros 2 tipos de finalidades que son definimos como nulas porque existen en los datos pero no hay definición al respecto.

Para cada tipología / finalidad de las 3 variables anteriores creamos una nueva variable en el DF *account* y contamos cuantas veces se repite la operativa en cada cuenta (similar a lo que realizamos con el fichero *order).* De esta forma asignamos a cada cuenta el tipo de operativa que realiza y como de habitual es dicha operativa en el periodo analizado.

Finalmente, a partir de la variable “balance”, que se corresponde con el saldo que queda en la cuenta después de realizar cada transacción, generamos la variable "Balance_in_negative", que nos indica en cuantas ocasiones una cuenta se ha quedado

en negativo y que puede ayudarnos a analizar la solvencia del cliente asociado a una cuenta.

En definitiva, hemos generado un DF llamado *account* que nos permite comenzar a analizar qué características tienen las cuentas que:

- Han contratado préstamos, tarjetas de crédito, seguros, leasings, depósitos,…
- Han generado préstamos morosos (para el análisis de la mora también podemos considerar las características de los préstamos: plazo, importe, cuota mensual,…).

El DF *account* lo guardamos en un fichero llamado "DFTenenciaProductos.csv", que va a ser nuestro input para el análisis con Python realizado en el fichero "Tenencia del producto préstamo Banco Checo.ipynb".

IMPORTANTE:

A continuación vamos a analizar si las cuentas contratan o no contratan préstamo. Elaborando esta memoria me di cuenta que en el trabajo que había realizado, al considerar las transacciones que incluyen la característica "Sym_LoanPayment" estaba transfieriendo a las variables "Num_Op_Remittances" y "Num_Type_Withdrawal" información sobre si una cuenta ha contratado préstamo que es lo que queremos predecir.

Entonces eliminamos las transacciones que incluyen la característica "Sym_LoanPayment" y generamos un nuevo fichero llamado "DFTenenciaProductos_wo_LoanPayment.csv" con el que vamos a analizar el fichero "Tenencia del producto préstamo Banco Checo_wo_LoanPayment.ipynb"

Todo mi trabajo se había basado en los ficheros:

- **"DFTenenciaProductos.csv" y "Tenencia del producto préstamo Banco Checo.ipynb". Las conclusiones del fichero "Tenencia del producto préstamo Banco Checo.ipynb" son incorrectas, ya que sobretodo "Num_Op_Remittances" y también "Num_Type_Withdrawal" contienen información sobre la carácterística "Sym_LoanPayment" que contiene la información de la variable que queremos predecir.**

Pero la última semana al darme cuenta del error me pongo a trabajar con:

- **"DFTenenciaProductos_wo_LoanPayment.csv" y "Tenencia del producto préstamo Banco Checo_wo_LoanPayment.ipynb" y estos ficheros son los que entrego como TFM. Aunque en GitHub he dejado también los de la versión errónea por si se quiere revisar el análisis.**

Análisis con "Tenencia del producto préstamo Banco Checo.ipynb" y "Tenencia del producto préstamo Banco Checo_wo_LoanPayment.ipynb"

Instrucciones

En estos ficheros vamos a intentar analizar qué características tienen las cuentas que contratan préstamos.

En estos análisis utilizamos paquetes habituales en nuestras clases como numpy, pandas, matplotlib, seaborn y sklearn.

Y otros menos frecuentes como:

- io, IPython y pydotplus para graficar Decision Trees
- pdpbox para realizar análisis de Partial Dependence y graficarlos
- shap para calcular SHapley Additive exPlanations del resultado obtenido por una observación y para ver como influye el valor de una variable en los modelos
- yellowbrick para:
    - Realizar gráficos del tipo RadViz
    - Calcular y graficar la Learning Curve de los modelos en función de la cantidad de datos utilizados
    - Ver la evolución de las métricas de Precision, Recall y F1 Score en función del threshold de un clasificador y calcular el threshold óptimo de un clasificador

Finalmente, también utilizamos el paquete imblearn para hacer oversampling en un problema con clases desbalanceadas. Vamos a aplicar el algoritmo SMOTE, concretamente en su versión SMOTENC (para variables continuas y categóricas).

Metodología

En este análisis vamos a generar modelos que clasifiquen las cuentas u observaciones (cuentas corrientes enriquecidas con la información del resto de ficheros) en función de si contratan o no contratan un préstamo. Y para esos clasificadores vamos a seleccionar las variables con mayor relevancia y vamos a analizar su comportamiento en el modelo.

Hemos empleado, fundamentalmente, modelos basados en árboles (Decision Trees, Random Forest, Gradient Boosting Classifier y XGBoost), pero también Support Vector Machines y KNN. Para la selección de modelos hemos seguido el protocolo propuesto por scikit-learn:

https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

La métrica seleccionada inicialmente era la Precision, ya que buscamos modelos que no "nos engañen" al predecir cuales son las variables fundamentales para haber contratado un préstamo, pero rápidamente hemos pivotado hacia F1 Score ya que para mejorar "ligeramente" la Precision teníamos sacrificar mucho Recall. Este trade-off entre Precision y Recall puede visualizarse al final del Notebook en los gráficos realizados con la función "DiscriminationThreshold".

Incluso utilizando Decision Trees puede resultar complicado interpretar el modelo resultante si "max_depth" del modelo es elevado. Por eso hemos aplicado técnicas que nos ayuden a conocer el comportamiento de las variables más relevantes en los

modelos. Hemos aplicado SHAP values (en los casos que es posible) y Partial Dependence.

Finalmente, nos hemos preguntado por la capacidad de los modelos para seguir mejorando y reduciendo el overfitting si se consiguieran más observaciones (utilizando LearningCurve) y también nos hemos planteado distintas formas de presentar los resultados obtenidos en función del threshold utilizado para discriminar si una probabilidad determinada implica la predicción de haber contratado préstamo o no (utilizando DiscriminationThreshold). Vemos que un mismo modelo se puede presentar de formas muy distintas (por ejemplo con un Precision y Recall muy similares o por el contrario muy alejados).

Principales resultados y conclusión

"Tenencia del producto préstamo Banco Checo_wo_LoanPayment.ipynb"

Este es el Notebook que entrego como TFM y en el que está corregido el efecto de "Sym_LoanPayment" a las variables 'Num_Op_Remittances' y "Num_Type_Withdrawal".

Mediante la ejecución de distintos modelos hemos encontrado 5 variables que aparecen entre las más relevantes para determinar que una cuenta/cliente haya contratado un préstamo. Son:

-'Num_Type_VYBER' (número de operaciones realizadas de un tipo especial de reintegros).

-'Num_Sym_IntDep' (número de abonos en la cuenta por pago de intereses de depósitos).

-'Date_Account' (Fecha de apertura de la cuenta)

-'Num_Op_Null' (Es un resto en el tipo de operativa realizada)

-'Ord_Leasing_amount' (Importe agregado de las domiciliaciones de pago de Leasings).

Con Partial Dependence vemos que la operativa de 'Num_Type_VYBER','Num_Op_Null' y 'Num_Sym_IntDep' (en este caso sobretodo para valores elevados) incrementa la probabilidad de haber contratado un préstamo. La variable 'Date_Account' nos indica que las cuentas que se han aperturado más recientemente, tienen mayor probabilidad de haber contratado un préstamo y finalmente con 'Ord_Leasing_amount' hemos visto que tener domiciliaciones de Leasing disminuye la probabilidad de contratar préstamo.

Para las variables 'Num_Type_VYBER', 'Ord_Leasing_amount' y 'Num_Sym_IntDep' hemos utilizado SHAP values y hemos contrastado los efectos anteriores pero con algunos matices (por ejemplo en el caso de valores muy elevados de 'Num_Type_VYBER' este efecto se reduce). Para poder subir a Github el código ejecutado sólo está explícito el primer gráfico de SHAP values para la variable 'Num_Type_VYBER'. En el resto de casos se puede obtener dicho gráfico ejecutando las celdas y siguiendo las instrucciones de cada celda para obtener la variable que se quiere graficar del desplegable del gráfico.

Para seguir trabajando con el equipo de Negocio en la razonabilidad de los efectos de las 5 variables anteriores entregaría:

- El generador interactivo de gráficos de SHAP Values que permite ver como impacta el valor de cada variable en la probabilidad de contratar préstamos.

- La calculadora de SHAP values para analizar el efecto de cada variable en la predicción de una observación.

Después de presentar estos resultados a negocio recogería sus inputs para continuar conjuntamente con el proyecto.

“Tenencia del producto préstamo Banco Checo.ipynb”

Este análisis es el realizado antes de detectar la filtración de información de las transacciones de tipología "Sym_LoanPayment" a las variables 'Num_Op_Remittances' y “Num_Type_Withdrawal”. Las conclusiones del fichero “Tenencia del producto préstamo Banco Checo.ipynb” son incorrectas, ya que sobretodo “Num_Op_Remittances” y también “Num_Type_Withdrawal” contienen información sobre la carácterística "Sym_LoanPayment" que contiene la información de la variable que queremos predecir.

Mediante la ejecución de distintos modelos hemos encontrado 2 variables que recurrentemente aparecen entre las más relevantes para determinar que una cuenta/cliente haya contratado un préstamo. Son 'Num_Type_VYBER' (número de operaciones realizadas de un tipo especial de reintegros) y 'Num_Op_Remittances' (número de envíos de dinero a otros bancos).

Con Partial Dependence vemos que estas variables tienden a incrementar las probabilidades de haber contratado un préstamo y con SHAP values vemos que el efecto de esta variable puede no ser monotónico y que por tanto debemos analizar el sentido de negocio del cambio de impacto de estas variables (para la mayoría de valores contribuyen positivamente a la probabilidad de contratar préstamo y para algunos valores contribuyen negativamente).

Para poder subir a Github el código ejecutado sólo está explícito el primer gráfico de SHAP values para la variable 'Num_Type_VYBER'. El resto de gráficos se puede obtener dicho gráfico ejecutando las celdas y siguiendo las instrucciones de cada celda para obtener la variable que se quiere graficar del desplegable del gráfico.

Además, ‘Ord_Leasing’>0 indica que la cuenta no ha contratado préstamo y por tanto disminuye la probabilidad de haber contratado un préstamo. Deberíamos analizar con negocio la razón de esta relación.

Adicionalmente, cuando utilizamos Gradient Boosting Classifier, en alguna ejecución han aparecido como variables de las más relevantes ‘Ord_Household_Payment_amount’ (la existencia de esta domiciliación disminuye la probabilidad de haber contratado préstamo) y ‘Num_Sym_Null2’ (a mayor volumen de esta operativa menor probabilidad de haber contratado un préstamo).

Para seguir trabajando con el equipo de Negocio, además de los efectos de las 5 variables anteriores, entregaría:

- El generador interactivo de gráficos de SHAP Values que permite ver como impacta el valor de cada variable en la probabilidad de contratar préstamos.

- La calculadora de SHAP values para analizar el efecto de cada variable en la predicción de una observación.

- Los gráficos que muestran que los modelos podrían mejorar con una mayor cantidad de datos, por si fuera fácil conseguirlos.

Después de presentar estos resultados a negocio recogería sus inputs para continuar conjuntamente con el proyecto.

NOTA:

Para cualquier consulta o comentario puedes encontrarme en el 609-05-83-50 o en osmmac@gmail.com