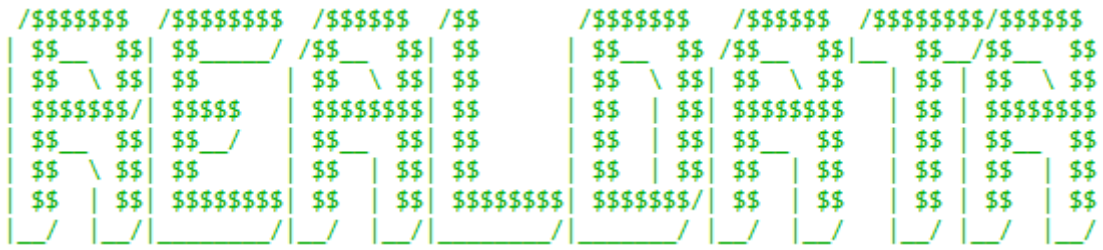
UNIVERSITYHACK 2018®
DATATHON

Reto Salesforce Predictive Modelling



Universidad Europea de Madrid
LAUREATE INTERNATIONAL UNIVERSITIES

Autores:
Marcelo José Laprea
Enrique Mora Alonso
Carlos Moral Rubio

INDICE

páginas

1. INTRODUCCIÓN

El equipo de RealData, representante por la Universidad Europea de Madrid para el reto de Salesforce Predictive Modelling, ha diseñado un algoritmo de análisis de datos que permite predecir el poder adquisitivo de un usuario común del banco Cajamar, a través de algunos de los datos que se almacenan respecto de los mismos.

Para la realización de este algoritmo se planteó la utilización de dos herramientas distintas: RapidMiner y Python. En un primer lugar, y debido a su elevada curva de aprendizaje, se optó por utilizar RapidMiner dónde conseguimos realizar una primera fase de análisis de las variables que se nos dan y cosechamos algunos resultados no muy esperanzadores. Para toda esta parte se utilizó todas las funciones que de por sí ya implementa RapidMiner por lo que no se hizo uso de ninguna librería adicional.

Tras un tiempo usando RapidMiner, detectamos unas series de carencias (tales como falta de configuración, falta de entendimiento de modelos, poca optimización de los mismos y una lentitud general en el uso del software) que necesitábamos cubrir y vimos en Python (y sus numerosas librerías) el mejor aliado.

En Python se realizó la última parte de la primera fase (fase clasificatoria) y toda la segunda fase del reto, donde se han ejecutado numerosas pruebas usando distintos modelos y modificaciones sobre los datos que se intentarán reflejar a lo largo de este documento. El conjunto de librerías usado durante todas estas fases se resume aquí:

- **Panda**
- **Numpy**
- **Xgboost**
- **Matplotlib**
- **Sklearn**
- **Math**
- **Mpl_toolkits**

2. ANÁLISIS INICIAL DE LOS DATOS

Una vez cargados los datos, se realizó un análisis inicial sobre todo el conjunto de variables que se disponen en el Dataset y que consistían en realizar distintas comprobaciones sobre todos los datos para asegurarnos que no encontrábamos ningún valor nulo o fuera de lugar. El resultado de esta comprobación fue completamente positivo y no se tuvo que realizar ninguna acción adicional.

Tras este primer paso se dividieron todo el conjunto de variables en dos: variables categóricas y variables numéricas. Esta primera subdivisión era necesaria puesto que a continuación se pasan a hacer distintas transformaciones atendiendo a que grupo pertenece las variables en cuestión.

Como variables categóricas se han considerado aquellas que se indicaban en uno de los correos y que se corresponden a las siguientes:

- 'Socio_Demo_01'
- 'Socio_Demo_02'
- 'Ind_Prod_01'...'Ind_Prod_24'

Para cada una de ellas se hizo un estudio de cómo su valor influía sobre nuestro target, el poder adquisitivo, y en función de ello se adoptan o se realizan unas u otras acciones

Como variables numéricas se tomaron todas las variables restantes y lo único a destacar es que tras una primera revisión de los datos que contenían la variable 'Socio_Demo_04' se observó que contenía un número muy bajo de valores distintos, lo que nos indica que podría ser tratada como una variable categórica ya que además tiene más sentido hacerlo como tal puesto que es una variable que indica alguna característica de los usuarios del banco y por tanto, los agrupa en distintos grupos.

3. ANÁLISIS DE VARIABLES NUMÉRICAS

Este es el conjunto total de variables numéricas:

- 'Imp_Cons_01'…'Imp_Cons_17'
- 'Imp_Sal_01'…'Imp_Sal_21'
- 'Num_Oper_01'…'Num_Oper_20'
- 'Socio_Demo_03'
- 'Socio_Demo_05'

En una primera instancia, se realizó una suma sobre todas las variables de Consumo, Saldos y Operaciones de forma que se obtenían 3 nuevas variables: 'Imp_Cons_Tot', 'Imp_Sal_Tot' y 'Num_Oper_Tot' y se eliminaban las 58 variables que las formaban. Esto tenía mucho sentido puesto que cada uno de los conjuntos representa el saldo, consumo u operaciones respectivamente para un determinado producto y además se reducía considerablemente la dimensionalidad del dataset, pero más tarde se realizó una prueba sobre el mismo para determinar si esta suposición era cierta. La prueba consistía en la aplicación de una regresión lineal sobre dos datasets diferentes: uno con las variables sumadas y otro sin dicha modificación, el resultado que se obtuvo fue una mejora de más de 1000 en el RMSE del modelo de regresión lineal para el dataset que no contenía las variables sumadas, por lo que finalmente se acabó desechando la hipótesis inicial y se decidió no calcular las variables totales pese al aumento de la dimensionalidad en el dataset.

La siguiente y última acción que se ejecutó para el conjunto de variables numéricas fue la estandarización de todos los datos siguiendo la siguiente fórmula:

$$X\_stand = (X - X.mean)/(X.stddev)$$

El motivo por el cual se ejecuta la estandarización es porque muchos de los algoritmos de modelado matemáticos están optimizados para los casos en los que las variables de entrada están acotadas y estandarizadas.

Otro tema importante que comentar es el motivo de la elección de la estandarización frente a la normalización. Para nuestro caso fue determinante el hecho de que la normalización puede ampliar el "ruido" que contengan nuestros datos y fue este el hecho que nos impulsó a escogerla en lugar de optar por la normalización.

4. ANÁLISIS DE VARIABLES CATEGÓRICAS

Este es el conjunto de variables categóricas:

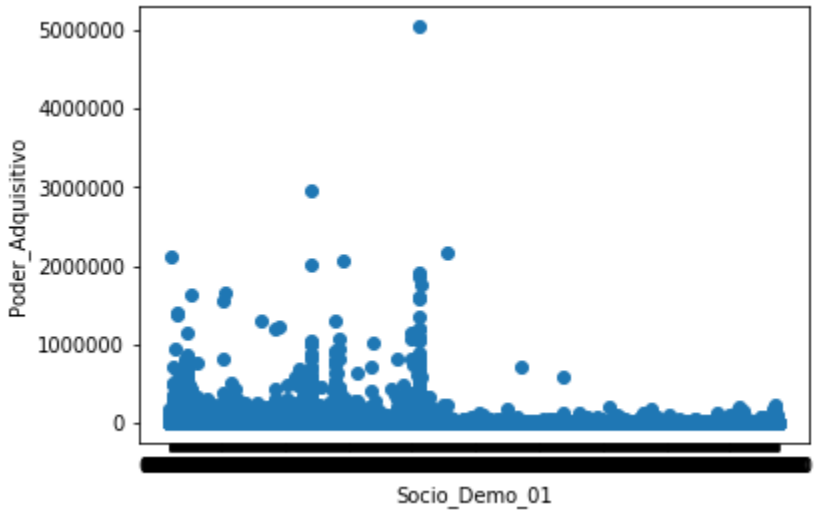
- 'Ind_Prod_01'...'Ind_Prod_24'
- 'Socio_Demo_01'
- 'Socio_Demo_02'
- 'Socio_Demo_04'

Para este conjunto de variables se hizo un estudio más detallado que con respecto a las variables numéricas, ya que son el conjunto más importante de variables del dataset. Para

poner en contexto, las variables categóricas nos permiten crear grupos de entre todos los usuarios del banco (que están representados mediante registros en nuestro dataset), agrupando para ello a todos aquellos que tienen algunas características comunes y que por lo tanto también son más propensos a tener un poder adquisitivo similar. Es por ello que es muy importante detectar esos grupos y poder definirlos correctamente.

Para conseguir lo mencionado anteriormente, lo primero que hicimos fue representar las tres variables demográficas frente al Poder Adquisitivo para ver si nos ofrecía una distribución que fuese representativa o que nos permitiese sacar algo más de partido a los datos:

SOCIO_DEMO_01

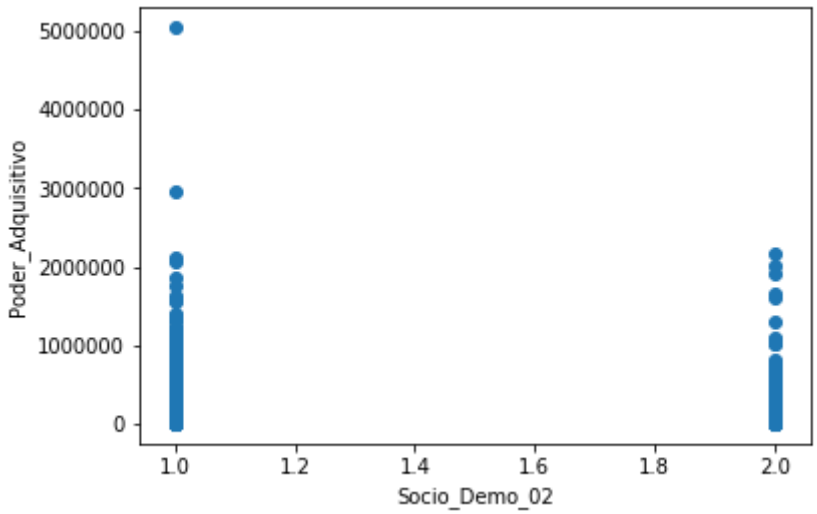


Si observamos esta distribución podremos observar que no hay ningún patrón claro al respecto, excepto que si el usuario queda representado en la mitad izquierda del gráfico tendría a priori más probabilidad de tener un Poder Adquisitivo Superior a 100.000 euros. Aun así, se trató de poner en práctica la observación que se acaba de comentar, pero resultó un proceso muy complejo puesto que el valor que toman los datos de la variable 'Socio_Demo_01' son de tipo alfanuméricos, haciendo prácticamente imposible la separación de los datos tal y como se observa en el gráfico.

Además, para esta variable podíamos tener más de 950 valores distintos, lo que traducido a una realización del one-hot encoding (algoritmo que expondremos más adelante y que se aplica a todas las variables categóricas) se traduce en la creación de más de 950 nuevas variables que se habrían de

añadir al dataset, algo insostenible para una ejecución eficiente del modelo.

SOCIO_DEMO_02

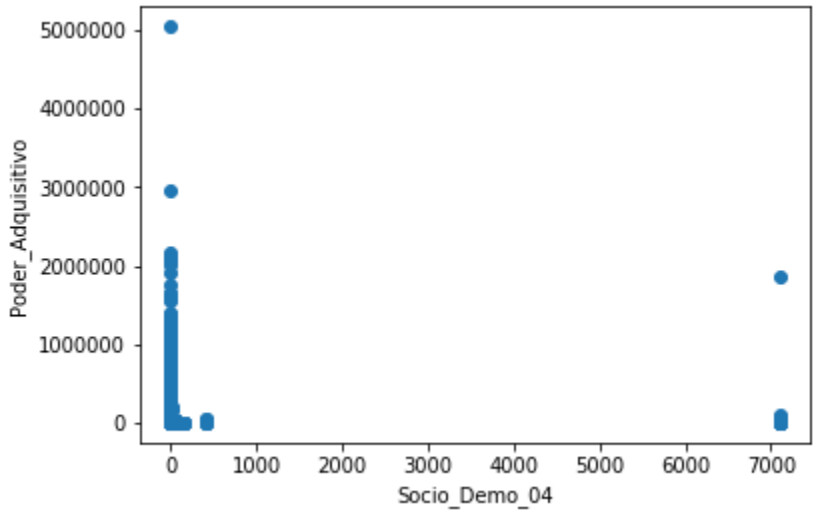


Para la variable 'Socio_Demo_02' sí que se puede observar que uno de los dos grupos está mucho mejor definido que el otro, ya que su rango está más acotado y es más lineal. Esto nos llevó a la decisión de realizar clusterización sobre la variable de modo que separamos el dataset en dos partes, una para cada valor de 'Socio_Demo_02' y se siguió con el procedimiento.

El motivo principal por el que se pretendía separar el dataset era que cada uno de los dos nuevos dataset resultantes de la división de la variable 'Socio_Demo_02', se atacara de forma diferente (modelos distintos), ya que se puede observar que cada uno tiene una distribución diferente, pero al final no encontramos un modelo que fuese mucho más apropiado para el grupo 1 que para el grupo 2 por lo que se ejecutó el mismo modelo para ambos.

Dicho esto, creemos que hay una forma de representar el grupo 1 de una forma distinta a la del grupo 2 y que haría más eficiente la predicción del Poder Adquisitivo para aquellos usuarios que se correspondiesen a la variable 1 de 'Socio_Demo_02'.

SOCIO_DEMO_04



Con respecto a la variable 'Socio_Demo_04' y su representación se puede observar claramente que pese a que contiene una variedad grande de números está dividida en dos grupos: aquellos inferiores al valor 5000 (o cualquier valor que va desde 1000 a 6000) y los inferiores al mismo. Por lo tanto, lo que se ha hecho para esta variable es una clusterización donde se ha asignado el valor 0 a la variable 'Socio_Demo_04' para aquellos cuyo valor anterior era inferior a 5000 y un valor de 1 para aquellos cuyo valor anterior era superior a 5000, de este modo no sólo hemos conseguido clusterizar la variable, sino que además ya la tenemos con un valor binario (y por ello ya no sería necesario aplicar el one-hot encoding a esta variable).

Anteriormente hemos comentado el método de one-hot encoding el cual consiste en redimensionar el conjunto de datos creando tantas variables nuevas como distintos valores pueda tomar la variable sobre la que se aplica e incluyendo el valor de 1 o 0 sobre las nuevas variables en función de que se tome o no el valor que la contiene. Este método es muy importante ya que necesitamos que todas las variables categóricas tengan valor binario (0/1) para conseguir (al igual que ocurría con las variables numéricas) una mejor optimización de los procesos de modelado matemático.

El método se aplica sobre todas las variables excepto por la 'Socio_Demo_04' que ya ha sido modificada manualmente. Una vez aplicado este método nos dimos cuenta de que en algunas

variables no estaban contempladas todas las casuísticas que se podían dar, por lo que finalmente se decidió por hacer el método de one-hot encoding en dos fases:

1. Aplicación del método sobre las variables 'Ind_Prod_01'...'Ind_Prod_24'. Puesto que estas variables pueden tomar tres posibles valores (0, 1 y 2) y podría ocurrir el caso en que uno de ellos no aparezca representado en todo el conjunto de datos (ya sea en el dataset de TEST o TRAIN) se añadieron 3 filas al final del dataset con valores constantes de 0, 1 y 2. De esta forma se garantiza la existencia de todas las posibles casuísticas que podrían darse.
   *Tras la aplicación del método one-hot encoding sobre estas variables se eliminan las 3 filas añadidas anteriormente para que no perjudique la calidad de los datos.

2. Aplicación del método one-hot encoding sobre la variable 'Socio_Demo_02'.

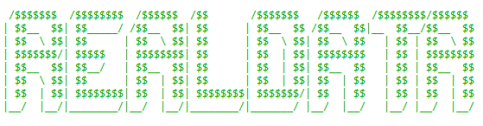
5. EJECUCIÓN DE DISTINTOS MODELOS

Una vez realizado todo el preprocesamiento de los datos, queda aplicar los distintos modelos matemáticos que nos permitirán predecir el valor del Poder Adquisitivo del cliente.
Para la elección del modelo matemático de predicción se va a usar el método de cross-validation con 10 folds (ya que es el que nos aporta una mayor seguridad al realizar más pruebas con más conjuntos de test) y las métricas de MSE (Mean Square Error) y RMSE (Root Mean Square Error).

En primer lugar, se utilizó una regresión lineal para una primera aproximación obteniendo los siguientes resultados:

MSE: 426.305.139,22
RMSE: 20.647,16

Seguidamente se planteó la ejecución de otros modelos matemáticos para determinar la posible mejora de las métricas y permitir así una mayor aproximación al valor esperado. Debido a su elevada capacidad de adaptación frente a los datos se decidió utilizar un modelo de redes neuronales (Multi-Layer Perceptron) para el cual realizamos numerosas pruebas, donde la mejor de ella fue la obtenida para una parametrización de 34 perceptrones en una sola capa y un límite de iteración de 750:



MSE: 17468.34
RMSE: 305143037.31

Aunque nos ofrecía un resultado muy mejorado con respecto a la regresión lineal seguimos intentando otros modelos en pos de conseguir el mejor resultado que nos fuese posible.

Para ello se aplicó el modelo matemático RandomForest con la parametrización por defecto de la librería scikit-learn, con la obtención de los siguientes resultados:

MSE: 398.168.237,57
RMSE: 19.954,15

De igual manera se aplicó GradientBoosting obteniendo las siguientes métricas de salida:

MSE: 403.308.364,89
RMSE: 20.082,54

Como mejora de la aplicación del modelo matemático GradientBoosting se realizó una prueba para el modelo XGBoost, que implementa una optimización del algoritmo anterior. La ejecución de XGBoost proporcionó las siguientes métricas:

MSE: 389.721.319,56
RMSE: 19.741,36

Siguiendo la búsqueda de encontrar el mínimo error posible en la predicción se intentó mejorar la ejecución de los modelos matemáticos anteriores mediante el uso de ensembles sobre los mismos, pero no obtuvimos ninguna mejora, ni siquiera con el uso de Adaboost y lo que sí que tuvimos fue un notable aumento del tiempo de computación por lo que, de nuevo se desechó esta idea.

6. ELECCIÓN Y PUESTA EN PRÁCTICA DEL MEJOR MODELO

Tras las numerosas pruebas realizadas y comentadas en el apartado anterior (no sólo por modelos diferentes, sino por distintas parametrizaciones y demás configuraciones…) se escogió el modelo MLP, MultiLayer Perceptron, con la siguiente configuración: 34 perceptrones distribuidos en una sola hidden_layer y un número máximo de iteraciones de 750, que permitía obtener convergencia en los datos.

Una vez escogido, se realizó de nuevo toda la fase de entrenamiento ahora con todos los datos de TRAIN y se aplicó el modelo obtenido al dataset de TEST obteniendo el Test_Mission que se adjunta con este documento.