MEMORIA TÉCNICA

capstone-project-1123bcn-fintech-1

Alberto Valero - Gonzalo Repetto - David Gónzalez

Entrega: 24 de mayo de 2024 - **Defensa**: 14 de mayo de 2024

Programa: DSC1123BCN

Tutor: Raquel Revilla

NUCLIO
DIGITAL SCHOOL

ÍNDICE

1 - Introducción

En esta memoria técnica, se detalla el desarrollo del Capstone Project del Máster (TFM) en Data Science, centrado en el caso práctico de easyMoney. Los miembros de este grupo asumiremos el rol de Data Scientist, dentro de una empresa dedicada a la comercialización de productos financieros.

Este proyecto tiene como objetivo conseguir aumentar la penetración en la cartera actual de clientes, consiguiendo aumentar el número de ventas de productos gracias a un recomendador personalizado, además de simular un entorno laboral real, donde los requerimientos están escasamente definidos o no lo están en absoluto.

Para alcanzar dicho objetivo, primero hemos analizado los datos que teníamos, hemos aplicado técnicas de limpieza, hemos agrupado a los clientes en "clusters" y finalmente hemos hecho un recomendador personalizado para cada uno de estos grupos, consiguiendo una buena previsión en el aumento de ventas del próximo mes.

Además de la experiencia de trabajar como data scientist, este ejercicio práctico nos brinda la oportunidad de trabajar en equipo, permitiéndonos familiarizarnos con una faceta crucial de la profesión: la colaboración y gestión de código desarrollado por otros.

2 - Justificación y Contextualización

En un entorno cada vez más competitivo, las empresas comercializadoras de productos financieros, como easyMoney, deben optimizar sus estrategias de marketing y desarrollo de negocio para atraer y retener clientes. La analítica avanzada ofrece herramientas poderosas para comprender mejor el comportamiento del cliente, predecir tendencias del mercado y mejorar la eficiencia operativa.

La importancia del tema radica en su potencial para transformar datos en información valiosa que guíe decisiones informadas. En particular, la aplicación de técnicas de ciencia de datos puede proporcionar insights profundos sobre patrones de consumo, segmentación de clientes y efectividad de campañas de marketing. Al abordar un escenario realista con requerimientos poco definidos, esta investigación también enfatiza la importancia de la creatividad y la capacidad de los Data Scientists para generar soluciones innovadoras en situaciones de incertidumbre.

Como estudiantes del Máster en Data Science, esta investigación representa una oportunidad valiosa para aplicar conocimientos teóricos en un escenario práctico, lo cual es fundamental para el desarrollo profesional en este campo. La experiencia de trabajar en equipo y gestionar código de otros también es crucial, ya que refleja la realidad del trabajo en la industria, donde la colaboración y la comunicación son esenciales para el éxito.

3 - Objetivos

El principal objetivo del proyecto es aplicar técnicas avanzadas de data science para mejorar las estrategias de marketing y desarrollo de negocio de easyMoney,. En particular, se pretende:

1. **Analizar y segmentar la base de clientes**: Identificar distintos segmentos de clientes basados en su comportamiento y características, lo cual permitirá personalizar estrategias de marketing.
2. **Optimizar campañas de marketing**: Evaluar la efectividad de las campañas actuales y desarrollar modelos predictivos para mejorar la planificación y ejecución de futuras campañas.
3. **Predecir tendencias de mercad**o: Utilizar técnicas de análisis predictivo para identificar tendencias emergentes en el mercado financiero y adaptar las estrategias de la empresa en consecuencia.
4. **Mejorar la penetración en la cartera de clientes**: Desarrollar un recomendador que aumente el número de ventas en la base de clientes actual

4 - Metodología empleada

4.1 - EDA + Data Cleaning

Nuestro proyecto comienza como 3 data sets distintos; uno para los productos, otro para la información sociodemográfica y otro para la actividad comercial.

- *products_df.csv*
- *sociodemographic_df.csv*
- *comercial_activity_df.csv*

Nuestra primera aproximación fue inspeccionarlos brevemente por separado, para tener claro como unirlos.

Para unirlos, usamos el método "***merge***", de la librería pandas, que nos permite unir los data sets a pares, usando las columnas "pk_cid" y "pk_partition" de nuestros data sets, ya que esta combinación de columnas nos da un identificador único para cada cliente, a lo largo de las distintas fechas que abarcan los data sets.

Con los data sets unidos en uno solo ("**df_cleandata.ipynb**"), comienza nuestra limpieza del mismo.

Asignamos un nuevo nombre "Index" a nuestra columna de índices y eliminamos la columna "Unnamed: 0", ya que esta era simplemente un índice asignado de manera aleatoria.

Convertimos las columnas "pk_partition" y "entry_date" al formato adecuado, ya que estas contienen fechas.

4.1.1 Imputación de nulos

A continuación, buscamos los nulos en todo el data set con "***isnull***()".

Obtenemos nulos en las columnas: "entry_date", "entry_channel", "segment", "payroll", "pension_plan", "region_code", "gender" y "salary".

Debido al bajo número de nulos en las columnas "entry_date", "payroll", "pension_plan", "region_code", "gender", decidimos eliminar los registros nulos con "***dropna***"

Para la columna "salary", primero calculamos la media del salario para cada región con "***groupby"***, e imputamos dicha media a los nulos de cada región, con su media correspondiente usando el método "***map***", aplicando un "***fillna***" y filtrando por "region_code"

Descartamos que haya duplicados.

Para la columna "segment", al haber solo 3 posibles valores, decidimos imputar la moda, ya que es el valor mayoritario.

Para la columna "entry_channel", al ser una columna categórica, antes de imputar nulos, echamos un vistazo a la distribución de valores con el método ***.value_counts()*** y vemos que la mayoría de valores se concentran en los mismos registros.

Por tanto, definimos una función, "obten_lista_eliminar" que lo que hará es filtrar los grupos de valores minoritarios de esta columna. Pasándole como inputs la columna y el umbral de valores, nos devuelve una lista con los elementos a eliminar.

Definiendo una segunda función, "eliminar_lista_df", que lo que hará es eliminar el grupo previamente filtrado de valores minoritarios.

Con esto, conseguimos que la variedad de valores en la columna "entry_channel" sea menor. De esta manera es mas sencillo imputar nulos con más sentido.

Como en el caso de "salary", lo que haremos es agrupar los nulos de "entry_channel" por "segment" y aplicarle a la moda de su segmento.

Con esto, el dataset quedaría completamente limpio de nulos.

4.1.2 Exploración de datos

Por último, hacemos una exploración de datos para poder sacar insights y posibles KPIs, inspeccionando las columnas con "***value_counts()***", "***nunique()***".

También pintamos algunas gráficas para ver las distintas distribuciones de los valores con la librería "***plt"***.

4.1.3 Tratamiento de categóricas

En esta sección, definimos una nueva función “obtener_lista_variables”, que separa los nombres de las columnas en función de si los valores que contienen son numéricos, categóricos o booleanos.

Para las columnas categóricas, decidimos convertirlas a valores numéricos usando encoders.

La columna “gender” abarca dos posibles valores, por tanto, aplicamos un enconder booleano, usando el método “***map()***” que convierte los valores categóricos “H” y “V” en “0” y “1” respectivamente.

Aplicamos lo mismo para la columna “deceased”

Para “country_id”, aplicamos un ordinal enconder, que lo que hace es asignar valores numéricos consecutivos a las distintas categorías de la columna, convirtiendo las etiquetas de cada país, en un número.

Aplicamos la misma idea a la columna “segment”, pero en este caso con sus 3 unicos valores convertidos a 1,2 y 3 respectivamente.

Con nuestro data set limpio y encodeado, los guardamos en el archivo “**clean_data.csv**” para continuar con el feature engineering.

4.1.4 Feature Engineering

Comenzamos importando el data set limpio de la sección anterior.

En esta sección el objetivo es agrupar por “familias de productos” todos los productos que tenemos en cartera, con el fin de facilitar la tarea del recomendador en el paso siguiente.

Las tres familias de productos serán “Financiación”, “Ahorro” y “Cuentas”.

Productos de Financiación: 'credit_card', 'debit_card', 'loans', 'mortgage', 'securities'

Productos de Ahorro: 'funds', 'long_term_deposit', 'pension_plan', 'short_term_deposit'

Productos de Cuentas: 'em_account_p', 'em_account_pp', 'em_acount', 'payroll','payroll_account','emc_account'

Creamos 3 columnas nuevas, una para cada familiar de productos, cuyo valor contendrá la **suma** de todos los productos de ficha familiar que tenga el cliente en dicho registro.

Tras ello, usando el método “***value_counts ()***”, podemos ver cuantos registros contienen qué cantidad de productos.

Guardamos el dataset con las nuevas columnas en el archivo “**feature_eng_data.csv**” para su posterior uso.

4.2 - Análisis Power BI

Para llevar a cabo el análisis de los datos, utilizamos la herramienta Power BI.
En primer lugar, abrimos la herramienta, y creamos un lienzo nuevo en blanco. Importamos los datos como documento csv. El archivo a utilizar es “clean_data.csv” ya que es el data set que tiene los 3 archivos juntos, limpios y encodeados.

La primera página del archivo la usaremos como portada, usando el logo de easy money, insertando un archivo tipo imagen.

La segunda página será para analizar información demográfica. Un primer gráfico será una tarjeta de resultados, que mostrará el número de usuarios totales diferentes. La métrica a utilizar será “pk_cid” y se mostrará como recuento distintivo, para que así solo nos cuente los valores únicos.

Usamos un gráfico circular para ver el desglose de usuarios que tienen descargada la app, así como la distribución del género de nuestros usuarios. En ambos casos, los valores serán “recuento distintivo de pk_cid”, y en la leyenda usaremos “active_customer” ó “gender” respectivamente. Esta página la completamos con un gráfico de columna vertical, para poder ver el reparto de la edad de los usuarios, pudiendo ver fácilmente las edades predominantes, y, con un gráfico de treemap, para ver la distribución de las regiones de los usuarios.

La siguiente página la utilizaremos para analizar información comercial de los usuarios. Utilizaremos un gráfico de líneas para ver el evolutivo de captación de usuarios, si bien, desglosamos el eje X en trimestres, para facilitar la visualización de los datos.

Además, completamos esta info con un gráfico de columnas, que muestre la misma información pero por años, para así poder ver en qué año se hicieron más registros.

En ambos gráficos utilizaremos en el eje X la dimensión “entry_date” y en el eje Y la métrica “pk_cid”

Esta página la completamos con dos gráficos, un gráfico de barras horizontal para ver el canal de entrada de los usuarios, “entry_channel”, y un gráfico circular para ver los diferentes segmentos a los que pertenecen los usuarios, lo cual lo indica la dimensión “segment”

Por último, en la última página, mostraremos una tabla para cada productos, para ver cuántos usuarios tiene contratado cada uno de los productos.

Cada tabla la formaremos añadiendo el nombre del producto y la métrica “pk_cid” mostrada como recuento distintivo. Para facilitar las visualizaciones de los datos, todas las leyendas y títulos de los ejes han sido editados para que se pudiesen entender mejor.

4.3 - Clustering

4.3.1 - Introducción

El objetivo es segmentar nuestros clientes en grupos mediante técnicas de clustering, lo que permitirá identificar patrones ocultos y optimizar estrategias comerciales. El dataset utilizado proviene del CSV 'clean_data.csv' que tiene los 3 archivos juntos, limpios y encodeado con información relevante sobre clientes, incluyendo datos demográficos, económicos y de productos financieros adquiridos.

4.3.2 - Carga y Exploratory Data Analysis (EDA)

Se utilizaron librerías de pandas, numpy, matplotlib, seaborn y sklearn.

Tenemos un total de 453636 clientes únicos, para un total de 5936633 registros. Como el dataset ya estaba previamente gestionado, tenemos un total de 0 nulos.

Tomamos la decisión de gestionar las variables de “salary”, “age”, “region_code” y los productos de venta. Como punto de partida para crear nuevas features.

Sobre “salary”, analizamos los rangos máximos, mínimos y la media, donde detectamos que hay algunos rangos de de salarios que generaban outliers en el clustering, por lo que dropeamos los cliente que tenían un percentil por encima del 95%.

Para “age” analizamos nuestros usuarios y definimos grupos etarios. Nuestra media estaba en los 29 años de edad y definimos los rangos según, **Menores de Edad** legalmente permitidos para tener una cuenta bancaria, los **Jóvenes Adultos**, hasta los 30 años donde se concentra nuestro mayor segmento de clientes, los **Trabajadores** hasta los 65 años y **Jubilados** más de 65 años

```python
# Funcion para definir los rangos de edad
def categorizar_edad(edad):
    if edad <= 17:
        return 'Menores de Edad'
    elif edad <= 30:
        return 'Jovenes Adultos'
    elif edad <= 65:
        return 'Trabajadores'
    else:
        return 'Jubilados'
```

Al analizar los resultados, vimos que había usuarios con 2 edades, esto se explica al rango de fechas del DataSet con el que trabajamos que es de 17 meses.

En relación al “region_code”, analizamos los códigos de cada usuario y contrastamos esta información con los códigos de Provincia provistos por el INE (Instituto Nacional de Estadística). Nos ayudó a validar que las Provincias con mayor población, son las que más usuarios tienen. Para generar categorías de región: 'Pequeña', 'Mediana', 'Grande' utilizamos el percentil 0.33 y 0.66 según la cantidad de usuarios en cada región.

Por último, agrupamos todos los productos que tenemos en cartera dependiendo el revenue que deja cada uno:

['**Financiacion'**] =['credit_card', 'debit_card', 'loans', 'mortgage', 'securities']

['**Ahorro'**] = ['funds', 'long_term_deposit', 'pension_plan', 'short_term_deposit']

['**Cuentas'**] = ['em_account_p', 'em_account_pp', 'em_acount',
'payroll','payroll_account','emc_account']

4.3.3 - Creación de variables

Con el análisis realizado, creamos nuestras nuevas variables para luego juntarlas con nuestro Dataset. Esta práctica genera una reducción de la dimensionalidad que nos permite tratar con mayor efectividad las features que nos interesan en el clustering.

Hacemos un groupby por “pk_cid” y se crean nuevas características agregadas. Se asignan valores binarios (0 o 1) para diferentes grupos etarios (Jubilados, JovenAdulto, Menores, Trabajadores) y tamaños de región (Grande, Mediana, Pequeña), indicando si cada pk_cid pertenece a estos grupos o no.

4.3.4 - Join con variables finales

Una vez generadas nuestras variables, preparamos el dataset final con el que realizaremos el cluster. Para eso preparamos una lista de los features a utilizar definido df3. Las features son: "salary", "age_group_Jubilados", "age_group_JovenAdulto", "age_group_Menores", "age_group_Trabajadores", "age", "Grande_region_size", "Mediana_region_size", "Pequeña_region_size", "Financiacion", "Ahorro" y "Cuentas"

4.3.5 - Creación de Transformers

Definimos nuestros propios transformers utilizando el FunctionTransformer, primero definimos las funciones con las “transformaciones” que queremos. Con las funciones definidas, paso la función a FunctionTransformer.

Luego definimos nuestro pipeline, con los siguientes pasos:

1 - Usar el KNNImputer, basado en la noción de métricas de distancia para imputar los valores nulos en función de los clientes más similares.

2- Creamos variables a nivel de cliente con ClientIdFeatureGenerator. Nuestro DataFrame de salida será más pequeño.

3- Estandarizamos los valores, usando RobustScaler para eliminar la influencia de los outliers.

4- Por último hacemos un fit con KMeans para calcular la dispersión de los datos al centroide.

4.3.6 - Elbow Curve

Aplicamos la técnica del Elbow Curve para analizar la variación de la dispersión de los clústers en función de la k. Y definimos que 7 clusters es un “sweet spot” para la cantidad de clusters, este valor mostró una buena balance entre inercia y simplicidad del modelo.

4.3.7 - Segmentación de los clientes con la "k adecuada"

Ahora que hemos determinado el número de centroides correcto avanzamos a fittear nuestro pipeline con la k adecuada. Añadimos al pipeline el paso del Clustering, se especifica random_state=175 para asegurar la reproducibilidad de los resultados.

pipe[:2] transforma utilizando solo las dos primeras etapas del pipeline (imputación y generación de características agregadas). Esto produce un conjunto de datos procesado sin escalar y sin clústers asignados, pero con características agregadas.

Luego pipe.predict aplica todo el pipeline a df3, incluyendo la etapa de clustering, para predecir las etiquetas de los clústers para cada registro.

X_processed["cluster"] = labels: Añade una nueva columna "cluster" al DataFrame procesado X_processed, asignando a cada cliente su correspondiente etiqueta de clúster. Esta asignación se realiza después de la transformación inicial, pero antes del escalado, para facilitar la interpretación.

4.3.8 - Ficha de los Clientes

Con la segmentación completa, creamos una ficha resumen de cada grupo con las principales variables de negocio necesarias para poder explicar los resultados al departamento de marketing.

Para cada columna, se agrupan los datos por clúster y se calculan las estadísticas descriptivas. Se seleccionan tres estadísticas: la media, el mínimo y el máximo.

Generamos un MultiIndex para el DataFrame de resumen, se crean listas para el out_index, inner_index y estadísticos. Calculamos el tamaño de los Clústeres y se presenta el dataframe final con estilo visual que facilita la visualización de las estadísticas, ayudando a destacar las diferencias entre los clústeres.

4.3.9 - Conclusión

El clustering realizado permitió identificar siete grupos distintos de clientes, cada uno con características únicas en términos de ingresos, edad, región y productos financieros adquiridos. Las decisiones tomadas durante el proceso, como la imputación de valores nulos y la normalización de datos, fueron fundamentales para asegurar la calidad del modelo.

4.4 - Recomendador

Ahora que tenemos toda la información necesaria, y nuestro data set limpio, podemos pasar al recomendador.

Decidimos hacer un recomendador para cada familiar de productos, pero aplicando la misma metodología a las 3 familias de productos, por tanto, explicaremos la metodología de una solamente.

Comenzamos importando el archivo .csv anterior "**feature_eng_data.csv**".

El primer paso es detectar, en todos los registros, cuando se realiza una compra.

Entendemos por una compra, cuando la suma total de productos (de una misma familia), para un mismo cliente, aumenta en al menos una unidad, entre 2 meses consecutivos.

Creamos una nueva columna, llamada " diff_productos", en la que agruparemos a los clientes por su id "pk_cid" y por el número de productos que tenga de una familia en concreto (en este caso, "Ahorro".

Una vez agrupado, aplicamos el método diff, que lo que hace es calcular la diferencia entre dos meses seguidos.

A continuación, creamos una nueva columna, llamada "compra", que tendrá un valor de 1 (éxito) en caso de que la columna anterior, "diff_productos" tenga un valor igual o superior a 1, y un valor de 0 (no éxito) en caso contrario.

Así, obtenemos en una sola columna las compras totales realizadas, que podemos visualizar con el método "***value_counts()***".

El siguiente paso es imputar los nuevos nulos de la columna "diff_productos" que se han generado, ya que el primer mes del data set no tiene con qué compararse del mes anterior.

Creamos también una nueva columna, que consideramos interesante para el recomendador, llamada "Antigüedad", que es un valor numérico que muestra, en número de meses, cuanto tiempo lleva el cliente dado de alta desde que realizó la primera compra.

Tras esto, vamos a tratar de predecir primero, de manera sencilla, las ventas estimadas del mes próximo, en función del histórico de ventas. Esto nos permitirá comparar entre las ventas que habría sin aplicar el recomendador, y las ventas aumentadas aplicando el recomendador.

Hacemos primero una serie temporal univariante, con las ventas distribuidas por mes. Preparamos la serie temporal para poder pasarla al modelo y separamos nuestro target "compras".

Aplicando el modelo "LinearRegression" de SckitLearn, podemos hacer un predict para el mes próximo, que nos devuelve unas ventas estimadas de 2862 productos.

Multiplicando las ventas por el beneficio que reporta cada producto, obtenemos unas ganancias estimadas de 114.485€

Por último, antes de realizar el recomendador, generaremos un modelo capaz de predecir, en función de datos nuevos de entrada de los clientes, si dicho cliente realizará una compra o no.

Para ello dividimos el data set entre el target (“compra”) y el resto de columnas. Hacemos un split entre el entrenamiento (80%) y el test (20%) y evaluamos el modelo elegido “XGBoost”, el cual consigue un acierto del 98.8%.

Para el recomendador, además de nuestro dataset actual, también recuperamos la columna “cluster” del apartado de ***clustering,*** ya que nos servirá para extraer los clientes pertenecientes a un cluster en concreto, que coincide con nuestro objetivo de recomendar una familia de productos específica a un cluster específico.

Con nuestra capacidad de cálculo computacional, nos vemos obligados a extraer solo una fracción del dataset total, fracción que hemos adaptado en función de la necesidad de número de usuarios.

Elegimos además un usuario, dentro del cluster, que sabemos con certeza que ha comprado uno de los productos de la familia de productos que le estamos ofreciendo a su cluster.

Calculamos la similaridad del coseno, teniendo en cuenta las siguientes columnas:

"salary","age","segment","gender","region_code",
"Financiacion","Ahorro","Cuentas","Antiguedad","compra"

Con dicha similaridad, ya tenemos todo lo necesario para buscar clientes similares.

Definimos una función que compare al “usuario estrella” (usuario elegido dentro del cluster que sabemos que ha realizado una compra”) con los 8600 (este valor varía en función de la familia de productos, ya que depende del número de mails enviados) más parecidos, y que nos devuelve el ID de dichos clientes y su valor de similitud.

Calculando la media en el valor de similitud entre estos 8600 clientes, obtenemos el % de probabilidad de compra medio, que en este caso es del 76%.

Todo este proceso, se repite de la misma manera para las otras dos familias de productos (Cuentas y Financiación), pero aplicando en cada caso el número de cluster al que pertenecen y el número de clientes similares necesarios.

5 - Resultados y conclusiones

5.1 - Conclusiones

El desarrollo de este proyecto nos ha permitido aplicar de manera efectiva técnicas avanzadas de data science para abordar problemas reales en el ámbito de la comercialización de productos financieros. A lo largo del proyecto, hemos logrado:

1. **Segmentación y Personalización:** La segmentación de la base de clientes en clusters distintos ha facilitado la personalización de estrategias de marketing, permitiéndonos dirigir dichas estrategias de manera más precisa y efectiva.
2. **Optimización de Campañas de Marketing**: Mediante el análisis predictivo, hemos evaluado y mejorado las campañas de marketing existentes, incrementando la eficiencia y efectividad de estas iniciativas.
3. **Predicción de Tendencias**: Las técnicas de análisis predictivo aplicadas nos han permitido identificar tendencias emergentes en el mercado financiero, proporcionando a easyMoney una ventaja competitiva al anticiparse a cambios y ajustar sus estrategias en consecuencia.
4. **Aumento de Ventas**: El desarrollo de un recomendador personalizado ha mostrado un aumento potencial en las ventas de productos financieros, basado en la capacidad del modelo para sugerir productos relevantes a los clientes adecuados.

5.2 - Resultados

5.2.1 - EDA y Limpieza de Datos

- La limpieza de los datos y la imputación de valores nulos nos permitió trabajar con un dataset completo y sin inconsistencias. La eliminación de duplicados y la imputación de valores faltantes aseguraron la integridad de los datos.
- El tratamiento de variables categóricas y numéricas, así como el “feature engineering”, proporcionó un conjunto de datos robusto para el análisis posterior.

5.2.2 - Análisis en Power BI

- La visualización de datos en Power BI permitió obtener insights claros sobre la demografía de los clientes, su actividad comercial y la distribución de productos financieros. Estas visualizaciones son esenciales para que podamos entender el comportamiento del cliente y ajustar las estrategias de negocio.

5.2.3 - Clustering

- La segmentación en siete clusters distintos nos reveló patrones en la base de clientes, proporcionando una comprensión más profunda de los diferentes segmentos y sus necesidades y facilitando el proceso de recomendación de productos.

5.2.4 - Recomendador Personalizado

- La implementación del recomendador para las tres familias de productos (Financiación, Ahorro y Cuentas) mostró un incremento potencial en las ventas mensuales del próximo mes en 7725 productos en total, traducido en ganancias estimadas de 304.400€ euros.

En resumen, el proyecto ha demostrado que la aplicación de técnicas de data science puede generar un valor significativo para empresas del sector financiero, como easyMoney. La combinación de análisis de datos, segmentación de clientes, optimización de marketing y modelos predictivos ha proporcionado una base sólida para futuras iniciativas y estrategias de negocio.

Alberto Valero - Gonzalo Repetto - David Gónzalez