Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Máster

Máster en Ciberseguridad e Inteligencia de Datos

Modelos de clusterización de clientes en el sector de transporte

Customer clustering models in the transportation sector

Carmen Castro González

La Laguna, 4 de Junio de 2021

D. **Héctor Javier Reboso Morales** , con N.I.F. 43.793.371-Z, Profesor Asociado de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como cotutor

D. **Pedro Antonio Toledo Delgado**, con N.I.F. 45.725.874-B Profesor Contratado Doctor de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como cotutor

C E R T I F I C A N

Que la presente memoria titulada:

"Modelos de clusterización de clientes en el sector de transporte"

ha sido realizada bajo su dirección por **Carmen Castro González**,

con N.I.F. 43.836.129-S.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 3 de junio de 2021

Agradecimientos

Gracias a mis padres, por apoyarme diariamente y ser un ejemplo a seguir de esfuerzo y constancia.

A mis hermanos, por creer en mí más que nadie y apoyarme en todas mis decisiones.

A la empresa en la que realicé el proyecto, por la calidad humana recibida durante todo el tiempo que estuve con ellos, la oportunidad de desarrollar un trabajo en el campo que me apasiona, y por todos los conocimientos y crecimiento personal que he adquirido.

No he podido tener más suerte con las personas que me han rodeado este último año tan duro, por ello, muchas gracias a todos.

Licencia

Resumen

Hasta hace no mucho tiempo, las decisiones de negocio eran tomadas por humanos, basándose en experiencias y estadísticas. Sin embargo, con el crecimiento exponencial de la Inteligencia Artificial, hoy en día es posible delegar muchas decisiones de negocio a las máquinas, que a través de análisis y algoritmos, son capaces de tomar, incluso, mejores decisiones que las propias personas.

De este modo, lo que se pretende con este proyecto es lo anteriormente mencionado, consiguiendo segmentar a los clientes de una empresa de transporte en función a unas variables determinadas, mediante la utilización de algoritmos automáticos no supervisados, en concreto, la clusterización.

Palabras clave: Inteligencia Artificial, Aprendizaje Automático, Aprendizaje no supervisado, Clusterización.

Abstract

Until not long ago, business decisions were made by humans, based on experiences and statistics. However, with the exponential growth of Artificial Intelligence, it is possible today to delegate many business decisions to machines, which through analysis and algorithms, are able to make even better decisions than people themselves.

Thus, the aim of this project is the aforementioned, segmenting the customers of a transport company according to certain variables, through the use of unsupervised automatic algorithms, in particular, clustering.

Keywords: Artificial Intelligence, Machine Learning, Unsupervised Learning, Clustering.

Índice

Índice de figuras

Índice de tablas

Capítulo 1: Introducción

Motivación

Con el incremento exponencial de la recolección de datos en todos los ámbitos, se genera una nueva oportunidad para la automatización de tareas tan importantes en el sector empresarial como la segmentación de clientes.

Esta nueva vertiente, brinda la oportunidad de diferenciar los distintos perfiles de clientes para, por ejemplo, recomendar productos específicos a cada perfil, y así alcanzar su correspondiente objetivo, incrementar las ganancias.

En el caso que atañe a este proyecto, se trabajará precisamente para esto, la segmentación de clientes en el sector de transportes. Dado que se tratan con datos privados, no se dará información acerca de la empresa o los datos utilizados durante el proyecto, pero sí sobre las tecnologías y procedimientos llevados a cabo.

Objetivos

Los objetivos que se buscan conseguir durante la realización del proyecto se dividen en:

Objetivo General

- La clusterización de clientes, agrupando los distintos perfiles existentes a través de técnicas de aprendizaje no supervisado.

Objetivos Específicos

- Estudiar algoritmos de aprendizaje no supervisado, en concreto la clusterización.
- Familiarizarse con los datos y códigos ya existentes.
- Crear nuevas variables.
- Modificar el código con nuevas propuestas.
- Realizar la clusterización.

Metodología

Para alcanzar los objetivos anteriores se van a definir varias etapas, por lo que se puede comprender el desarrollo del proyecto en tres etapas:

En la primera etapa se lleva a cabo el preprocesamiento de datos, conocida como **Fase SQL**. Esta es la etapa en la cual se estudia y comprende el flujo de datos de la empresa, las tablas almacenadas, los procedimientos creados y la generación del dataset final que se lleva a la próxima etapa. Además, es necesario modificar dos de las variables del dataset final original, creando un procedimiento que recoja datos y consultas de otras tablas. Una vez actualizado el valor de estas variables, se obtiene un dataset final que es el que se va a llevar a la siguiente etapa.

En la segunda etapa, conocida como **Fase Python**, se recoge el dataset generado en la fase anterior, para proceder al preprocesado de datos en Python. Durante esta, se tratan diversas cuestiones como la eliminación o imputación de valores nulos, la categorización de variables, la eliminación de outliers, así como diversos análisis y transformaciones en los datos para intentar obtener el mejor rendimiento a la hora de realizar la clusterización, como es la partición de los datos en dos datasets, uno para clientes recurrentes y otro para esporádicos, el escalado de datos, un análisis de correlación, un análisis de componentes principales para la reducción de la dimensionalidad del dataset, etc.. El resultado de esta etapa es una dataset con los datos tratados, a partir de los cuales, se realiza la clusterización.

Por último, tiene lugar la **Fase Clusterización**, donde se procede a la implementación de diversas métricas para escoger cuestiones no triviales como la elección del número de clústeres. Como resultado final, se va a obtener un dataset que asocie cada cliente a su clúster.

Tecnologías utilizadas

Durante el proyecto se han utilizado diversas herramientas para el desarrollo. Las más relevantes se nombran a continuación:

SQL Developer

SQL Developer[1] es un entorno de desarrollo integrado para trabajar con SQL en bases de datos Oracle. Se ha utilizado en la primera etapa del proyecto, para la conexión a la base de datos, la consulta de datos, creación de procedimientos y generación de la tabla que se lleva finalmente a clusterización.

Anaconda

Anaconda[2] es la plataforma de distribución de Python más popular del mundo, ampliamente utilizada en la ciencia de datos y aprendizaje automático gracias a los miles de paquetes que se pueden descargar en este ámbito. Se utiliza en la segunda y tercera etapa para el uso de Python, Jupyter Notebook y la instalación de las librerías de Python que se iban necesitando.

Jupyter Notebook

Jupyter Notebook[3] es un entorno de trabajo interactivo que permite desarrollar código en Python de manera dinámica, a la vez que integrar en un mismo documento diversos bloques de código, para ir ejecutándose por partes. Es un SaaS utilizado ampliamente en análisis numérico, estadística y machine learning, entre otros campos de la informática y las matemáticas. Se usa en la segunda y tercera etapa para todo el desarrollo del preprocesado y la clusterización con Python.

Estructura del documento

La estructura del documento se divide en diversos capítulos donde se expone el trabajo de investigación y desarrollo realizado, para llegar a un modelo de clusterización.

En el Capítulo 1 se realiza una breve introducción del proyecto, donde se exponen también los objetivos, metodología y tecnologías utilizadas.

En el capítulo 2 se introduce el tema del proyecto, presentando la rama de la ciencia llamada aprendizaje automático, a partir de la cual surgen los algoritmos de clusterización que se estudian para llevar a cabo el trabajo.

En el capítulo 3 se explica como se ha llevado a cabo el desarrollo del proyecto, etapa a etapa, dando la explicación a las decisiones tomadas y los procedimientos realizados en cada una.

En el capítulo 4 se exponen las conclusiones a las que se ha llegado finalmente y posibles líneas futuras.

Capítulo 2: Aprendizaje automático y la clusterización

Aprendizaje Automático

El Aprendizaje Automático (conocido también por su nombre en inglés, Machine Learning)[4][5] es una rama de la inteligencia artificial y de la informática que se centra en el uso de datos y algoritmos para imitar la forma en que aprenden los humanos, mejorando gradualmente su precisión.

Se trata de encontrar patrones en los datos y tomar decisiones de futuro basadas en estos. El objetivo principal es que las máquinas aprendan automáticamente sin intervención o ayuda humana y ajusten las acciones que realizan en consecuencia.

Los algoritmos de Machine Learning se suelen categorizar como supervisados o no supervisados.

Aprendizaje supervisado

En el aprendizaje supervisado, se trabaja con datos “etiquetados”, es decir, datos de los que ya se conoce lo que se intentará predecir. Se intenta encontrar un mecanismo que, dados los valores de las variables de entrada, les asigne la etiqueta o clase adecuada. Los algoritmos se entrenan con un histórico de datos y así aprenden a asignar la etiqueta de salida adecuada a nuevos valores, es decir, predice el valor de salida.

El aprendizaje supervisado comprende problemas de clasificación y problemas de regresión, donde la diferencia entre ellas reside en el tipo de la variable a predecir. En clasificación es de tipo categórico, mientras que, en los casos de regresión, es de tipo numérico.

Los algoritmos más habituales que se aplican para el aprendizaje supervisado son: Árboles de Decisión (Decision Trees o Random Forest), Algoritmos de los vecinos más cercanos (k-NN; k-Nearest Neighbours), Clasificadores Bayesianos (Naïve Bayes), Redes Neuronales, Regresión Logística y Máquinas de Vectores Soporte (Support Vector Machines; SVM).

Durante el proyecto, se usarán este tipo de aprendizaje para la imputación de valores ausentes, con dos de los algoritmos mencionados en el párrafo anterior, el Random Forest Regressor y la Regresión Logística, ya que se van a predecir variables numéricas, resultando en un problema de regresión.

Aprendizaje no supervisado

La clasificación no supervisada tiene lugar cuando no se dispone de datos "etiquetados" para el entrenamiento. Sólo se conocen los datos de entrada, pero no existen datos de salida que correspondan a un determinado grupo. Por tanto, sólo se puede describir la estructura de los datos, para intentar encontrar algún tipo de organización que simplifique el análisis.

El aprendizaje no supervisado está más estrechamente ligado con la inteligencia artificial, ya que la máquina puede aprender a identificar procesos y patrones complejos sin un humano para proporcionar orientación a lo largo del camino.

Los problemas de aprendizaje no supervisados se agrupan en problemas de clusterización y asociación.

En concreto, se va a trabajar con este tipo de aprendizaje en la etapa de Clusterización, como su propio nombre indica. Este tipo de problemas se explica con más detalle a continuación.

Clusterización

La clusterización es un concepto importante cuando se trata de Aprendizaje no Supervisado. Su función principalmente es la de encontrar una estructura o patrón en una colección de datos no categorizados.

Los algoritmos de clusterización procesan los datos y encuentran grupos o clústeres naturales si existen en los datos. También se puede modificar cuántos grupos deben identificar sus algoritmos.

El objetivo general es conseguir clústeres formados por elementos muy parecidos entre sí, pero diferentes de los elementos de otros clústeres. Existen distintos tipos de clusterización que se puede aplicar. Los algoritmos de agrupamiento más comunes incluyen la clusterización basada en prototipos, la clusterización jerárquica y la clusterización basada en densidad.

Para el caso que atañe a este proyecto, se procede a explicar uno de los algoritmos de clusterización basados en prototipos, el K-means, que es además, el método que se ha aplicado en el proyecto.

K-Means

K-Means[6] es un algoritmo no supervisado de clusterización cuyo objetivo es el de encontrar "K" grupos (clústeres) entre los datos.

El algoritmo asigna iterativamente cada observación (filas del conjunto de entrada) a uno de los clústeres en función de sus características. Son agrupados en base a la similitud de sus variables (las columnas). Como resultado de ejecutar el algoritmo se obtiene:

- Los centroides de cada grupo, que serán unas “coordenadas” de cada uno de los K grupos que se utilizarán para poder etiquetar nuevas muestras.

- Etiquetas para el conjunto de datos de entrenamiento. Cada etiqueta pertenece a uno de los K grupos formados.

Las variables que se llevan a la clusterización deben ser valores numéricos continuos. Por ello, se deben mapear si existen, las variables categóricas a numéricas. Además, es recomendable también que los valores estén normalizados, manteniendo una misma escala y no conviene usar variables correlacionadas.

En cuanto al algoritmo, utiliza un proceso iterativo en el que los grupos se ajustan para producir el resultado final. Para ejecutar el algoritmo, se debe pasar un conjunto de datos y un valor K como entrada, y el conjunto de datos son las variables de cada punto. Las posiciones iniciales de los centroides K se asignan aleatoriamente desde cualquier punto del conjunto de datos de entrada. Luego se itera en dos pasos:

1. Asignación de datos: En este paso, cada “fila” del conjunto de datos se asigna al centroide más cercano basado en la distancia entre puntos. De esta forma, los puntos asignados a un centroide forman un cluster.
2. Recalcular centroide: En este paso, se volverá a calcular el centroide de cada grupo. Esto se hace tomando el promedio de todos los puntos asignados en el paso anterior.

Se repiten estos pasos hasta que se cumple un criterio de detención, como que los datos no cambian más de cluster o se alcanza un número máximo de iteraciones.

Hay que mencionar que el valor de K que se escoge como entrada no es una elección trivial. Es más, esta es una de las principales desventajas de la clusterización, pues en función del valor de K, los resultados pueden ser totalmente distintos y además, no se conoce a qué cluster debe ser asignada cada observación.

Para seleccionar un valor de K adecuado, no existe un método exacto que dé solución a que valor debe tener K, pero se puede estimar con distintos métodos, como el método del codo o el análisis de la silueta, ambos aplicados en la Fase de Python.

También es posible medir la calidad de la clusterización con diversas métricas de evaluación, como se explica en la siguiente sección.

Métricas de Validación

Una vez realizada la clusterización, se pueden aplicar diversas técnicas e índices[7] para la validación del agrupamiento.

En primer lugar, hay que mencionar que existen dos tipos de validación, externa e interna. La principal diferencia es que uno usa información externa para la validación, es decir, información que no es producto de la técnica de clusterización utilizada, por ejemplo, un dataset de entrenamiento en el que ya se tienen las observaciones clasificadas en sus correspondientes clústeres, para contrastar este con los resultados de la clusterización realizada.

Por otro lado, las técnicas de validación interna miden el rendimiento de la clusterización únicamente basándose en la información que proporcionan los datos, evaluando qué tan buena es la estructura del clustering.

Las métricas que se usan en este proyecto, son las técnicas de validación interna. Estas se utilizan para escoger el mejor modelado y número óptimo de clústeres.

Las métricas de validación interna están basadas normalmente en dos criterios, la cohesión y la separación.

- Cohesión: Define que cada observación dentro de su cluster sea lo más cercano posible al resto de observaciones del clúster.
- Separación: Establece que los cluster deben estar suficientemente separados entre ellos. Existen distintas formas de medir esta separación, como distancia entre el miembro más cercano de cada clúster, distancia entre los miembros más lejanos o la distancia de los centroides.

A continuación, se exponen las técnicas utilizadas en el desarrollo para determinar la calidad de la clusterización.

Método del codo

La idea básica de los algoritmos de clusterización es la minimización de la varianza intra-cluster y la maximización de la varianza inter-cluster. Es decir, que cada observación se encuentre muy cerca a las de su mismo cluster y los clústeres lo más lejos posible entre ellos.

Este es el método más utilizado para obtener el número óptimo de clústeres. Utiliza la distancia media de las observaciones a su centroide. Es decir, se fija en las distancias intra-cluster. Cuanto más grande es el número de clústeres, la varianza intra-cluster tiende a disminuir. Cuanto menor es la distancia intra-cluster mejor, ya que significa que los clústeres son más compactos.

Análisis de la Silueta

El análisis de la silueta mide la distancia de separación inter-cluster. Indica cómo de cerca está cada punto de un cluster a puntos de los clústeres vecinos.

Los valores que puede tomar van desde -1 a +1, donde un valor más cercano 1 indica que la observación está bien clasificado en su cluster y alejado del resto. Un valor cercano a 0 indica que la observación está en duda entre más de un cluster, mientras que los valores negativos indican que se ha asignado al cluster erróneo.

Índice Calinski y Harabasz

También conocido como Criterio de Relación de Varianza. Este es un índice basado en la “suma de cuadrados”, que mide la dispersión de los puntos a nivel inter-cluster e intra-cluster.

Un valor más alto de esta métrica significa que los clústeres son densos y están bien separados, aunque no hay un valor de corte "aceptable".

Índice Davies-Bouldin

El criterio de Davies-Bouldin evalúa la similitud intra-cluster e inter-cluster..

Valores pequeños para el índice Davies-Bouldin indica clústeres compactos, y cuyos centros están bien separados los unos de los otros. El valor mínimo es 0, por lo que lo más cerca que se encuentre de este valor, mejor.

Capítulo 3: Desarrollo

Como se ha señalado en el Capítulo 2, el desarrollo del trabajo se ha llevado a cabo en diferentes etapas. En este capítulo se pretende dar una descripción en profundidad del desarrollo llevado a cabo en cada una de ellas.

Fase SQL

Durante esta primera etapa se ha tenido una toma de contacto inicial con los datos de la empresa en cuestión. Dado que se requería de un entendimiento de negocio elevado para comprender la información almacenada en la base de datos, llevó su tiempo inferir el significado de estos.

Los procesos que existen, consultan diversas tablas dentro de la base de datos para formar la tabla final (dataset), que se va a llevar a la próxima etapa para tratarlos en Python.

Como se comenta en el Capítulo 1, este proyecto viene a proponer un desarrollo distinto al original, por lo que se redefinen dos de las variables que se consideran más importantes dentro de la tabla final, bajo unas condiciones que se establecen en el equipo correspondiente de la empresa.

Para modificar el valor de estas columnas, en SQL Developer se ha llevado a cabo la creación de dos procedimientos almacenados, que consultan varias tablas de la base de datos. Una vez que se consigue actualizar el valor de estas dos variables de mayor importancia dentro de la tabla, se continúa a la siguiente etapa con la tabla resultante, que consta de más de 170 columnas.

Fase Python

Durante la etapa de preprocesado en Python, se han visto diversos aspectos a tratar, cuya resolución se nombra a continuación.

Valores ausentes

Los valores nulos dentro de un dataframe representan una serie de inconvenientes, como la imposibilidad de usar algoritmos de Machine Learning sobre el dataset, que son necesarios tratar.

Dado que había una cantidad suficiente de variables con valores ausentes, se decidió eliminar aquellas variables con un número muy elevado de valores ausentes, en este caso, con más del 90%, dado que al tener tantos nulos, estas variables realmente no van a proporcionar información de valor.

Tras eliminarlas, se disminuye casi en la mitad el número de variables con valores ausentes, quedando entre estas sólo dos de tipo categórico, y el resto numéricas.

A estas dos variables de tipo categórico, se decide imputar un valor por defecto, ya que el porcentaje de ausentes en estas no es elevado, y la categoría imputada tiene sentido teniendo en cuenta el resto de valores dentro de las variables.

Se crea también un método para comprobar no sólo por columnas, sino por filas, que no hay ningún registro con un exagerado número de valores ausentes.

```
df['MISSING_PCT'] = df.isnull().sum(axis=1)/len(df.columns)

# La fila con mayor numero de valores ausentes, tiene sólo un 15% de datos faltantes.
df['MISSING_PCT'].max()
```

Figura 1: Registro con mayor porcentaje de valores ausentes

Se llega a la conclusión, que el cliente que posee un porcentaje mayor de valores ausentes, es solo de un 15%, lo que no supone un porcentaje elevado, y hace que no sea necesaria la eliminación de ningún registro.

En el desarrollo original, se imputan los valores nulos de las variables numéricas por la mediana, pero en este proyecto se va a realizar una predicción con algoritmos de regresión para imputar por el valor predicho.

El primer paso para aplicar los algoritmos e imputar es convertir las variables categóricas en numéricas, ya que la mayoría de algoritmos sólo procesan variables numéricas.

Manejo de variables categóricas

En el manejo de variables categóricas se van a transformar estas a su equivalente numérico. Para ello se deben tener en cuenta dos escenarios. En el primero, existen variables categóricas ordinales, es decir, que se pueden representar con un orden numérico. Un ejemplo sería mapear poco, medio y mucho a 1, 2 y 3 respectivamente. En el segundo escenario, no existe una relación ordinal en los valores de las variables. Por ello, se usan dos tipos de transformaciones[8], uno para cada escenario:

- Integer Encoding: Este tipo de transformación se lleva a cabo en las variables categóricas ordinales. Como se nombró en el ejemplo, se mapea cada valor categórico de la variable en uno numérico.
- One-Hot Encoder: Este se aplica a las variables no ordinales. La estrategia que implementa es crear una columna para cada valor distinto que exista en la variable que se quiere codificar y, para cada registro, marcar con un 1 la columna a la que pertenezca dicho registro y dejar las demás con 0.

Según la naturaleza de las variables, se aplica uno u otro. Para el caso de Integer Encoding se utiliza simplemente el método map() de Python, y para el One-Hot Encoder, el método get_dummies() de la librería pandas de Python.

Imputación de valores ausentes

A continuación de la transformación de las variables categóricas, se procede a la imputación de valores ausentes. La proporción de valores perdidos es:

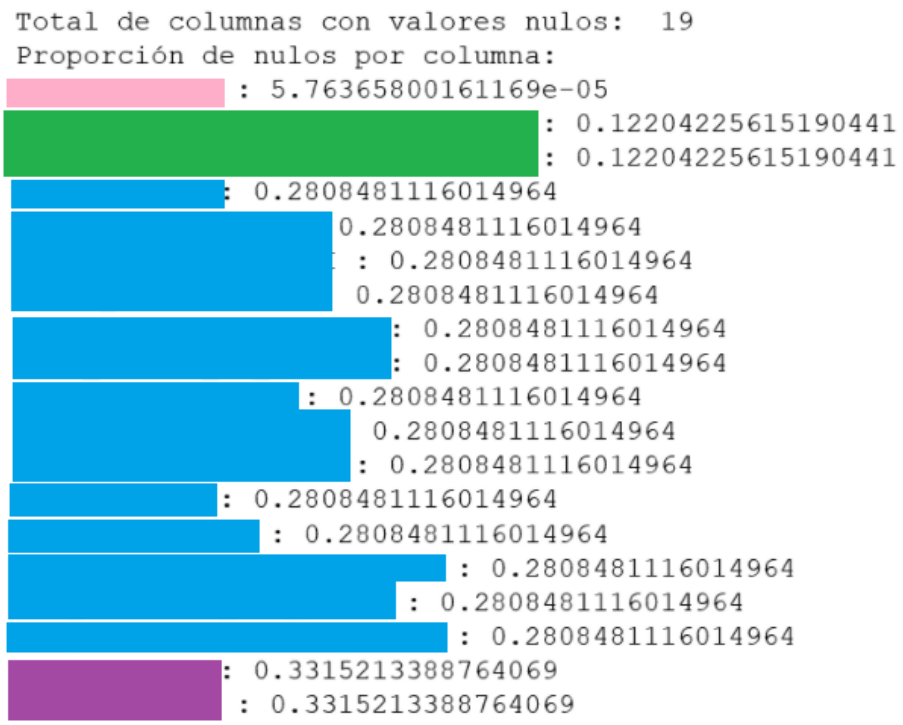

Figura 2: Proporción de valores perdidos por variable

En la figura 2, cada fila corresponde a una columna con valores ausentes en el dataset.

Se aprecia que el grupo rosa tiene un porcentaje minúsculo de valores perdidos, por lo que, para esta variable se imputa por la mediana.

Para el resto de variables, se puede observar que se pueden dividir en 3 grupos, en función de la proporción de valores perdidos:

- Grupo violeta, con un 33% de valores perdidos.
- Grupo verde con algo más del 12% de valores perdidos.
- Grupo azul, con un 28% de valores perdidos.

Teniendo esto en cuenta, se puede proceder a la imputación mediante algoritmos de regresión por cada grupo.

Grupo violeta

Primero, se crea una copia del dataset. A partir de este se crean dos listas, una lista contiene las variables que no tienen valores ausentes, llamada icols, que son las variables independientes utilizadas para realizar la predicción, y la otra lista, las variables ausentes comentadas en la figura 2, llamada jcols, que son las variables dependientes a predecir.

En resumen, la idea es predecir los valores ausentes de las variables en jcols, a partir de los datos de las variables en icols, como si de un problema de regresión normal se tratase.

Se va a realizar subdividiendo las columnas de jcols, creando 3 grupos:

1. Variables del grupo violeta.
2. Variables del grupo verde.
3. Variables del grupo azul.

Se comienza por la imputación de las variables del grupo violeta.

Para llevarlo a cabo, se crea una dataset con todas las filas en las que no hay valores perdidos, que actuará como dataset de entrenamiento, llamado df_notnans, y otro con filas en las que hay al menos un valor ausente, llamado df_nans.

El conjunto de datos df_notnans, es sobre el que se aplica el algoritmo para aprender de los datos, y poder aplicar la imputación sobre las filas de df_nans correspondientes en cada caso.

Se han probado dos algoritmos de regresión, la Regresión Lineal y el Random Forest Regressor. Se escogen como variables a predecir las del grupo violeta.

Los resultados obtenidos con la Regresión Lineal son:

```
from sklearn import metrics
y_pred = LR.predict(X_test)
print('Mean Absolute Error (MAE):', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error (MSE):', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error (RMSE):', metrics.mean_squared_error(y_test, y_pred, squared=False))
print('R^2:', metrics.r2_score(y_test, y_pred))
```

```
Mean Absolute Error (MAE): 23.78586460933786
Mean Squared Error (MSE): 93295562.55386133
Root Mean Squared Error (RMSE): 7952.40374798249
R^2: -5397505.15165339
```

Figura 3: Resultados predictivos grupo violeta con Regresión Lineal

En la figura 3 se observa que los resultados son muy malos. Una de las cosas que llama la atención es el valor negativo en R^2, esto indica lo mal que se ajusta a los datos el modelo, y que una imputación por la media sería mejor opción.

Los resultados obtenidos con la Random Forest Regressor son:

```
from sklearn import metrics
y_pred = RF.predict(X_test)
print('Mean Absolute Error (MAE):', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error (MSE):', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error (RMSE):', metrics.mean_squared_error(y_test, y_pred, squared=False))
print('R^2:', metrics.r2_score(y_test, y_pred))
```

```
Mean Absolute Error (MAE): 0.8170700428604405
Mean Squared Error (MSE): 3.5908492889551447
Root Mean Squared Error (RMSE): 1.8949058021321745
R^2: 0.8122135356948812
```

Figura 4: Resultados predictivos grupo violeta con Random Forest Regressor

Por el contrario, los resultados obtenidos con este algoritmo son bastante satisfactorios, por lo que se procede a usar este modelo para realizar la imputación.

En las filas de df_nans donde el valor de las variables del grupo violeta es nulo, se procede a aplicar el modelo para predecir estas variables e imputar el valor de la predicción.

Así, una vez realizado, ya no quedan valores ausentes en estas columnas.

Grupo verde

A continuación, se procede a imputar los valores en las variables del grupo verde.

El desarrollo es el mismo que en el grupo anterior. A partir de df_notnans, se divide el dataset en entrenamiento y test, teniendo como variables a predecir, las anteriormente mencionadas.

De nuevo, se prueba con la Regresión Lineal y con el Random Forest Regressor, y este es el resultado:

Con Regresión Lineal:

```
from sklearn import metrics
print("Reusltados de la Regresión Linear:")
y_pred = LR.predict(X_test)
print('Mean Absolute Error (MAE):', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error (MSE):', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error (RMSE):', metrics.mean_squared_error(y_test, y_pred, squared=False))
print('R^2:', metrics.r2_score(y_test, y_pred))
```

```
Reusltados de la Regresión Linear:
Mean Absolute Error (MAE): 71.38497582244153
Mean Squared Error (MSE): 608530055.1996603
Root Mean Squared Error (RMSE): 24592.632049469477
R^2: -1895338.78581382
```

Figura 5: Resultados predictivos grupo verde con Regresión Lineal

De nuevo, los resultados obtenidos con este algoritmo son demasiado deficientes como para considerar la imputación con este modelo.

Con Random Forest Regressor:

```
from sklearn import metrics
print("Resultados del Random Forest:")
y_pred = RF.predict(X_test)
print('Mean Absolute Error (MAE):', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error (MSE):', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error (RMSE):', metrics.mean_squared_error(y_test, y_pred, squared=False))
print('R^2:', metrics.r2_score(y_test, y_pred))
```

```
Resultados del Random Forest:
Mean Absolute Error (MAE): 0.3637463205791793
Mean Squared Error (MSE): 6.313376773633735
Root Mean Squared Error (RMSE): 2.5121900921061098
R^2: 0.9815655957615201
```

Figura 6: Resultados predictivos grupo verde con Random Forest Regressor

Parece que el Random Forest es el modelo que mejores resultados obtiene una vez más, por lo que es este modelo el que se utiliza para la imputación.

Grupo azul

En esta ocasión, se vuelve a realizar los pasos mencionados anteriormente en los grupos violeta y verde, sin embargo, los resultados obtenidos con ambos algoritmos no son lo suficientemente buenos como para permitir realizar imputación por predicción, por lo que se mantiene la imputación por la mediana en esta ocasión.

Una vez hecho esto, se unifican los datasets df_notnans y df_nans, consiguiendo de nuevo el conjunto de datos entero y sin ningún valor ausente.

Outliers

En el caso de los valores anómalos, se examina mediante un diagrama de caja, aquellas observaciones que se salen de lo normal o que están causando ruido, ya que estas pueden ocasionar muchos problemas a la hora de realizar la clusterización, y se los registros correspondientes a los valores exageradamente anómalos.

Peso de variables

Hay que tener en cuenta que a las variables se les asigna un peso, este es decidido por un departamento de la empresa. Por ello, como se comentaba en el apartado de anomalías, se les asigna peso 0 al grupo de variables que están causando estas anomalías, pues el ruido en el dataset afecta enormemente el rendimiento de la clusterización.

Simplemente se estandarizan los datos, mediante la función de Scikit Learn, StandardScaler(), que estandariza las variables sustrayendo la media y escalando la varianza a uno, y se aplica el peso de las variables, multiplicando el valor por el peso asignado en cada columna.

Estudio de Correlación

Cuando las variables utilizadas en la agrupación son colineales, algunas variables obtienen un mayor peso que otras. Si dos variables están perfectamente correlacionadas, representan efectivamente el mismo concepto. Pero ese concepto estaría representado dos veces en los datos y por tanto recibe el doble de peso que las demás variables. La solución final estará probablemente sesgada en la dirección de ese concepto, lo que podría ser un problema si no se prevé. Por ello, se realiza un estudio de la correlación, para eliminar las variables fuertemente correladas.

Se ha decidido por parte del departamento de comercial, dividir el dataset en **clientes recurrentes y clientes esporádicos**, nombrando a partir de ahora el dataset de recurrentes como dataR y el de esporádicos como dataE, para el estudio independiente de ambos grupos. Por ello, el análisis de la correlación se hace de forma independiente en para los conjuntos de datos..

Se procede a ver cuántas variables poseen un coeficiente de correlación mayor al 80%, para eliminarlas, pues es importante desechar aquellas variables muy correladas antes de proceder a la clusterización.

En el caso del dataset de clientes recurrentes, se encuentran más de un tercio de variables correladas del dataset total, 45 variables, mientras que en el de esporádicos es parecido, con un dos más que en el de recurrentes. Se eliminan y se actualizan los datasets, reduciendo así las dimensiones de los mismos.

Análisis del PCA

Como se le conoce en inglés, *Principal Component Analysis* (PCA)[9] es un método estadístico que permite simplificar la complejidad de espacios muestrales con muchas dimensiones a la vez que conserva su información. Se intenta mapear el espacio de variables original, de dimensión p, de tal manera que se encuentre un número de factores menor a p, llámese z, que expliquen aproximadamente lo mismo que las p variables originales Así, donde antes se necesitaban p valores para caracterizar cada individuo, ahora bastan z valores, y cada una de estas z nuevas variables recibe el nombre de componente principal.

El PCA pertenece a la familia de técnicas de aprendizaje no supervisado, y lo que intenta es “condensar” la información aportada por múltiples variables en sólo unas pocos componentes,

El proceso de PCA identifica aquellas direcciones en las que la varianza es mayor. Como la varianza de una variable se mide en su misma escala elevada al cuadrado, si antes de calcular las componentes no se estandarizan todas las variables para que tengan media 0 y desviación estándar 1, aquellas variables cuya escala sea mayor dominarán al resto. De ahí que se haya realizado dicha estandarización previamente al PCA.

Las variables que son consideradas más importantes son aquellas que influencian más a la creación de los componentes principales, y por lo tanto, son las que explican mayor varianza en los datos.

El análisis de componentes principales tiene lugar después del análisis de correlación. Con este se pretende conocer cuáles son las variables que están aportando más información en los datos.

A continuación, se muestra una visualización, donde en el eje Y se encuentra una barra por cada variable, y en el eje X el peso de cada variable.

Para el dataset de recurrentes, una parte de la gráfica queda así:

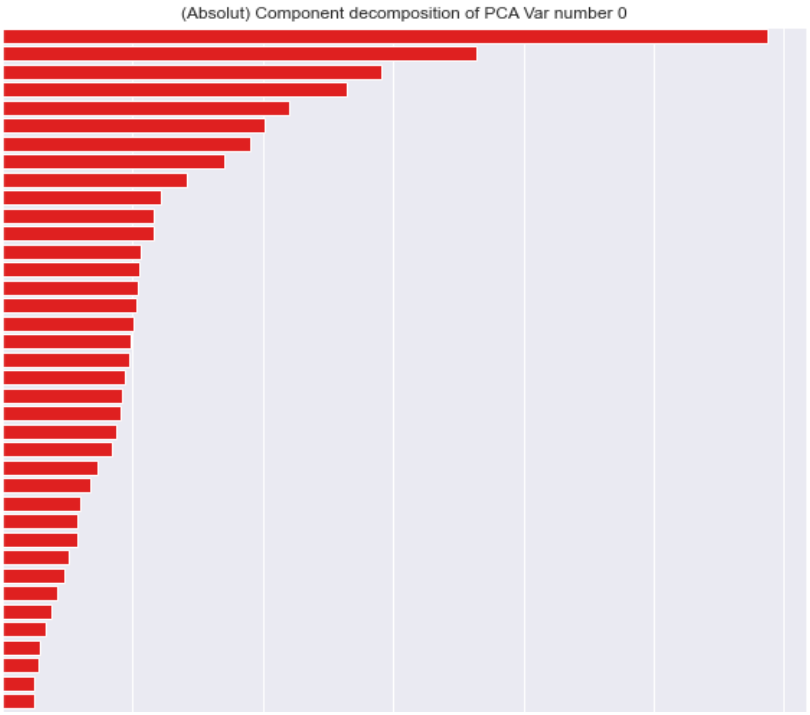

Figura 7: Gráfica del PCA del dataset de recurrentes

Para el dataset de esporádicos, una parte de la gráfica queda así:

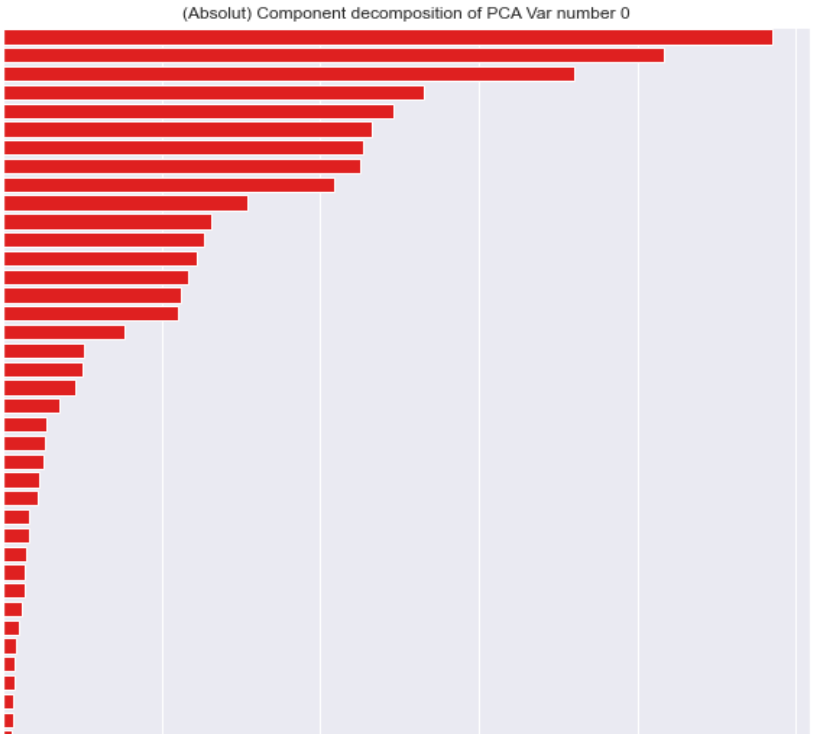

Figura 8: Gráfica del PCA del dataset de esporádicos

Dados estos resultados, se decidió hacer un estudio de cuáles serían las dimensiones óptimas del dataset que se lleva a la clusterización, por lo que se van a crear varios conjuntos de datos, filtrando por la proporción de información que aportan/varianza que explican.

Creación de diversos datasets

El filtrado tiene lugar de la siguiente manera:

```
important_featuresR0 = df_plotR[df_plotR.abs() > 0.25].index.tolist()
len(important_featuresR0) # Variables con > 25%
```

```
4
```

Figura 9: Filtrando variables a partir del PCA con una proporción de > 0.25 en recurrentes

En función del valor que se usa para filtrar, se obtienen más o menos variables, por ejemplo, filtrando por aquellas variables que aportan más del 25% en el dataset de recurrentes, sólo hay cuatro.

Para ejemplificar los datasets creados, se muestra una tabla:

Nombre	Origen del data	Proporción	Número de

	frame	usada	Variables
dataR0	Recurrentes	0.25	4
dataR1	Recurrentes	0.15	8
dataR2	Recurrentes	0.1	17
dataR3	Recurrentes	0.07	25
dataE0	Esporádicos	0.25	3
dataE1	Esporádicos	0.15	10
dataE2	Esporádicos	0.1	16
dataE3	Esporádicos	0.05	19

Tabla 1: Datasets creados a partir de los resultados del PCA

Fase Clusterización

Es esta última fase se desarrolla todo lo relacionado con la clusterización, incluyendo las técnicas y métricas para escoger el número óptimo de k, la comparación de distintos modelados, y la realización de la clusterización.

Métricas para la Clusterización

En esta sección, se han implementado diversas métricas para medir conceptos como el número óptimo de clústeres , para los diversos datasets del apartado anterior, y poder comparar entre estos.

Función del codo

La función del codo se ha realizado dos veces. Se realiza para obtener el número óptimo de clústeres. Se establece el rango para probar k de 2 a 15.

Primero se ha utilizado con la métrica de inercia y guardado los resultados en un dataset, que además se guardó en la base de datos, para su visualización y estudio de la mejor alternativa por otros miembros de la empresa. El dataset en cuestión tiene la siguiente estructura:

	inercia	num_cluster	tipo_cliente	num_dimensiones	orden
0	1.933054e+06	2	Recurrente	4	2
1	1.248638e+06	3	Recurrente	4	3
2	8.502985e+05	4	Recurrente	4	4
3	5.893682e+05	5	Recurrente	4	5
4	4.773820e+05	6	Recurrente	4	6
5	4.110429e+05	7	Recurrente	4	7

Figura 10: Dataset para comparar los distintos valores de k.

Siendo:

- inercia: Puntuación de inercia conseguido.
- num_cluster: Número de k utilizado para la clusterización.
- tipo_cliente: Dataset utilizado para la clusterización.
- num_dimensiones: Número de variables del dataset utilizado.
- orden: Valor redundante a la columna num_cluster, pero en tipo cadena, para poder realizar búsquedas en otras herramientas de forma más cómoda.

Luego, se utilizó para la visualización de distintos valores de k para los los datasets de la tabla 1, con la librería Yellowbrick[10]. Esta usa la métrica de distorsión, en vez de la inercia. La diferencia es:

- Distorsión: Se calcula como la media de las distancias al cuadrado de los centros de los clústeres respectivos. Normalmente, se utiliza la métrica de la distancia euclidiana.
- Inercia: Es la suma de las distancias al cuadrado de las muestras a su centro de cluster más cercano.

A continuación se muestra un ejemplo con la visualización de la gráfica para el data frame dataR1:

Figura 11: Método del codo para dataR1 con la librería Yellowbrick

La librería es bastante útil y marca directamente el valor de k que obtiene un mejor puntaje. En este caso, para dataR1 es 7.

Análisis de la Silueta

El análisis de la silueta también va a ayudar a la elección del número de clústeres, ya que mediante este se estudia la distancia de separación entre los clústeres resultantes.

Se utiliza la métrica *Silhouette score[11]*, de la librería scikit learn, que calcula el coeficiente de silueta medio de todas las muestras.

Un ejemplo de la salida, por ejemplo, para el data frame dataR1, es:

```
dataR 1 shape: (541109, 8)
For n_clusters = 2, silhouette score is 0.341897850940236)
For n_clusters = 3, silhouette score is 0.3923366722603221)
For n_clusters = 4, silhouette score is 0.42058278672387694)
For n_clusters = 5, silhouette score is 0.359160933343061)
For n_clusters = 6, silhouette score is 0.36820433888289944)
For n_clusters = 7, silhouette score is 0.37405591121313686)
For n_clusters = 8, silhouette score is 0.38975846990785556)
For n_clusters = 9, silhouette score is 0.39989750734561247)
For n_clusters = 10, silhouette score is 0.39960408790271257)
For n_clusters = 11, silhouette score is 0.40902835714755903)
For n_clusters = 12, silhouette score is 0.4037751565197918)
For n_clusters = 13, silhouette score is 0.4046394021924575)
For n_clusters = 14, silhouette score is 0.4078435921791648)
```

Figura 12: Análisis de la silueta para dataR1

En la imagen, se muestra que el valor considerado el número de clústeres óptimo con esta métrica es 11.

Calinski-Harabasz index

Ahora, se utiliza la función calinski_harabasz_score, de scikit learn. Se recuerda que se busca el mayor resultado posible.

La salida obtenida con dataR1 es:

```
dataR 1
Data aggregation 2 calinski_harabaz index is: 132294.319733
Data aggregation 3 calinski_harabaz index is: 162566.455035
Data aggregation 4 calinski_harabaz index is: 177532.542721
Data aggregation 5 calinski_harabaz index is: 191432.270098
Data aggregation 6 calinski_harabaz index is: 191515.884860
Data aggregation 7 calinski_harabaz index is: 193104.874871
Data aggregation 8 calinski_harabaz index is: 190876.483098
Data aggregation 9 calinski_harabaz index is: 188796.599497
Data aggregation 10 calinski_harabaz index is: 185598.747279
Data aggregation 11 calinski_harabaz index is: 184741.354774
Data aggregation 12 calinski_harabaz index is: 181491.839684
Data aggregation 13 calinski_harabaz index is: 178871.221716
Data aggregation 14 calinski_harabaz index is: 177190.521012
```

Figura 13: Calinski-Harabasz Index para dataR1

Se observa que el mejor puntaje es obtenido cuando el número de clústeres es 7.

Cabe mencionar que a mayor número de variables en el dataset, peores resultados se obtienen con todas las métricas en la ejecución del código. Es por eso que los data frames dataR0 y dataE0 obtienen los mejores resultados, seguidos de dataR1 y dataE1, y así consecutivamente.

Clusterización con Kmeans

Tras analizar las métricas obtenidas en los apartados anteriores, se pueden sopesar los distintos resultados obtenidos y escoger el número de clústeres que se considere apropiado, al tratarse de clusterización con kmeans.

Se realiza la clusterización con el dataset de recurrentes y esporádicos que obtienen los mejores resultados en los análisis realizados anteriormente.

Se ha observado, que estos son dataR1 y dataE1, que constan de 8 y 10 variables, dado que con dataR0 y dataE0 se tienen muy pocas variables, 3 y 4 variables, por lo que se prefiere llevar el primer conjunto nombrado a la clusterización.

Además, para comprobar si hay una diferencia significativa en el resultado de la clusterización con este dataset, dataR1 y dataE1 al que se referirá a partir de ahora como 1, también se hace con los dataset recurrente y esporádico resultantes del estudio de la correlación, es decir, antes de aplicar la eliminación de variables del PCA (se llaman dataR y dataE, y tienen 87 y 85 variables respectivamente), dado que se considera que se han eliminado variables importantes y se quiere ver si esto influye mucho en el resultado de la clusterización, y se quiere guardar el resultado de ambos modelados en el mismo dataset, para posteriormente, filtrar en otras herramienta y comparar ambos.

Por ello, primero se despliega la gráfica del codo para el data frame dataR y dataE.

Para dataR:

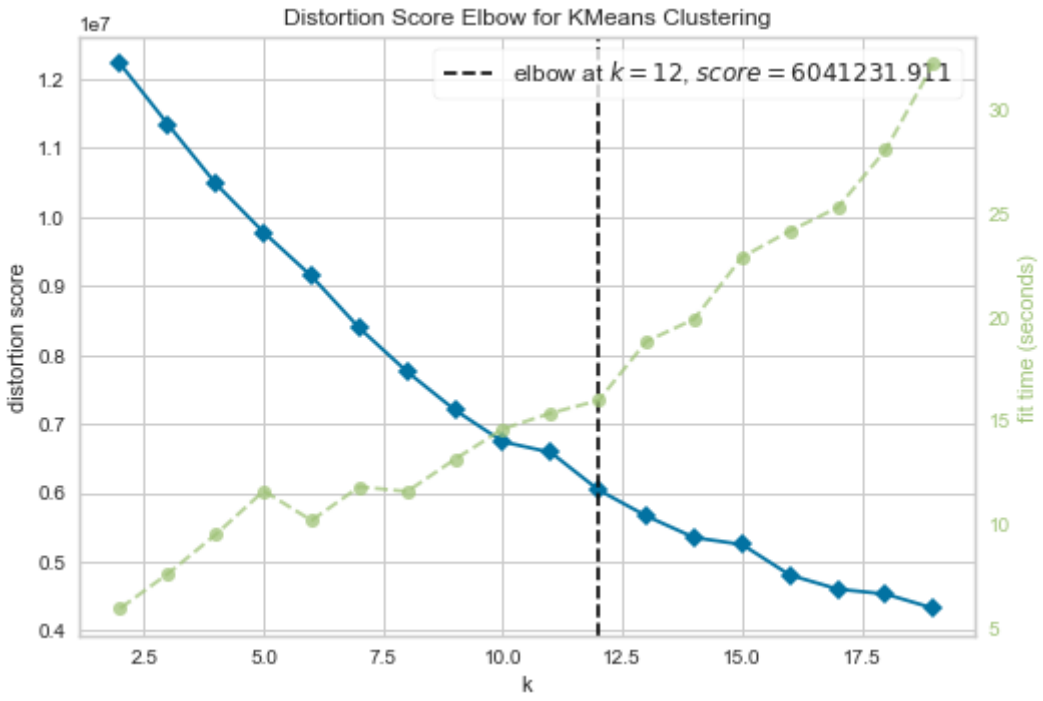

Figura 14: Gráfica del codo para dataR

Para dataE:

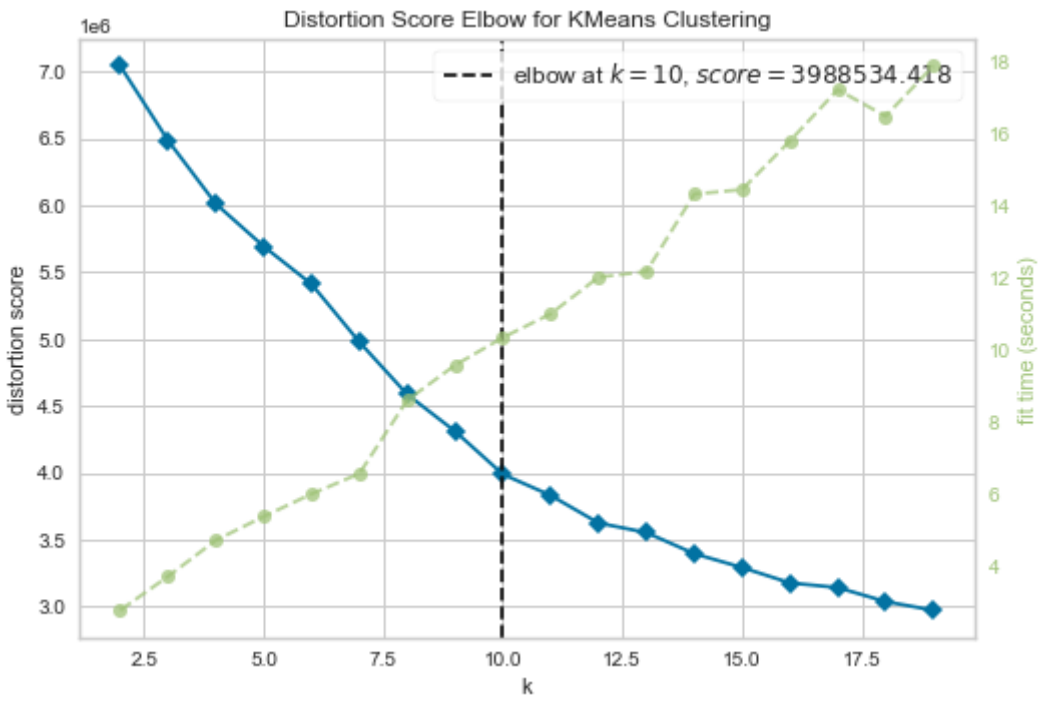

Figura 15: Gráfica del codo para dataE

Con esta información, se procede finalmente a realizar la clusterización.

Se decide utilizar k = 11 para los datasets recurrentes, ya que en el análisis de silueta obtiene los mejores resultados. Se utiliza este k tanto para dataR como para dataR1, para comprobar como dista la distribución de las observaciones en los clústeres, pero se podrían usar otros valores de k considerados para cada dataset independientemente.

Para los datasets esporádicos, se usa k = 14 dado que en el método de silueta para el dataE1, este es el valor óptimo. A su vez, por lo explicado en el párrafo anterior, se usa el mismo valor para dataE.

La función de scikit learn ejecutada para realizar la clusterización en kmeans[12] es la siguiente:

KMeans(n_clústeres =kmeans_optimal_nclus, random_state=rdm_st, n_init = ninit)

Siendo:

- n_clústeres: Número de clústeres formados así como número de centroides generados. Este valor es 11 para recurrentes y 14 para esporádicos
- random_state: Determina la generación de un número aleatorio para la inicialización de centroides. El valor en todos los casos es 42.

- n_init: Número de veces que se ejecuta el algoritmo con diferentes semillas de centroide. El resultado final será la mejor salida de n_init ejecuciones consecutivas en términos de inercia. El valor en todos los casos es 30.

Con los resultados obtenidos, se generan cuatro nuevos datasets, conformado únicamente por el identificador del cliente y el cluster al que se ha clasificado como resultado en cada modelado. Luego, se procede a realizar las siguientes visualizaciones.

Visualización

Tras aplicar el algoritmo en los cuatro datasets, se utiliza de nuevo la librería Yellowbricks, esta vez con una función llamada InterclusterDistance[13], que muestran una representación de los centros de los clústeres, mapeando el espacio de variables original en 2 dimensiones.

Cuanto más cerca estén los centros en la visualización, más cerca estarán en el espacio de características original. La dimensión de los clústeres representados son en función de la cantidad de observaciones perteneciente a cada centro. Por ello, aunque en las imágenes parezcan solapados los clústeres, no implica que se solapen también en el espacio de características original.

En dataR: En dataR1:

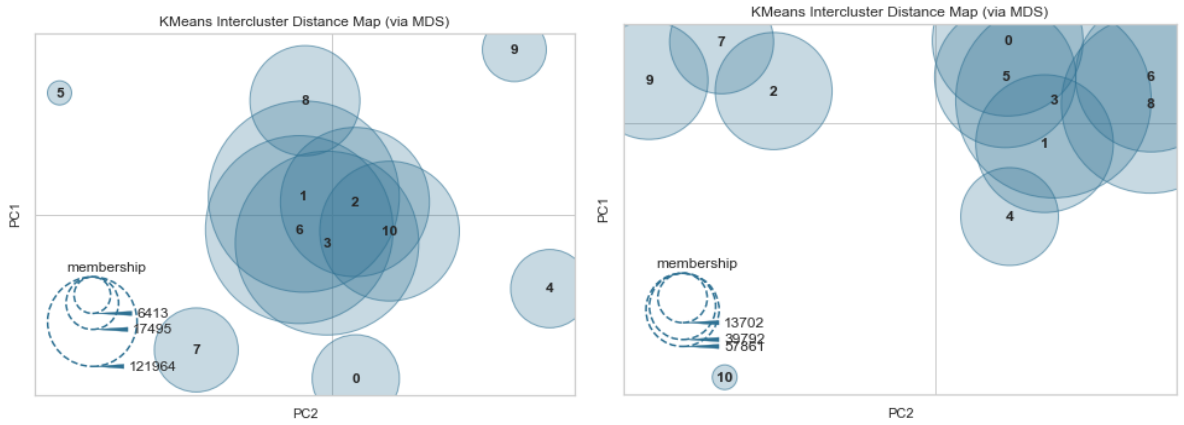

Figura 16: Comparativa de la distancia inter-cluster de los distintos modelos de recurrentes

En dataE: En dataE1:

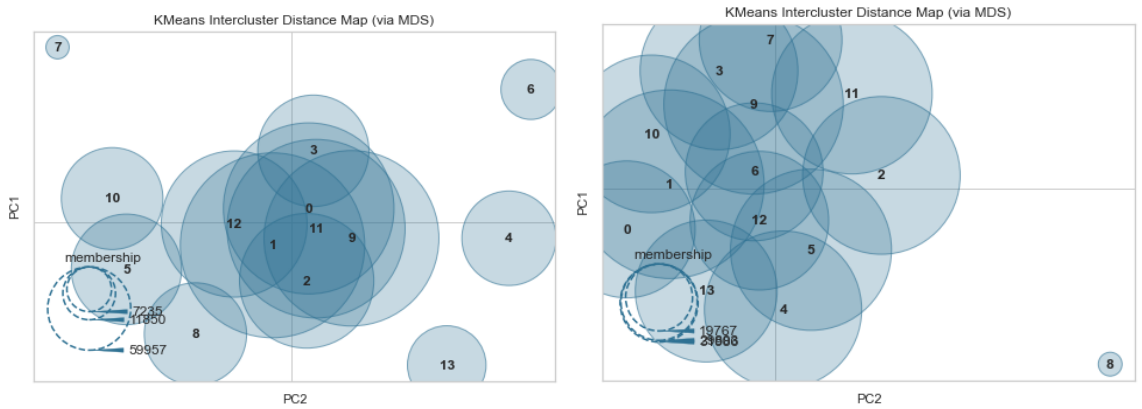

Figura 17: Comparativa de la distancia inter-cluster de los distintos modelos de esporádicos

Parece no haber mucha similitud entre los distintos modelados de recurrentes y de esporádicos.

También se compara con un diagrama de barras la cantidad de clientes clasificados en cada cluster:

En dataR: En dataR1:

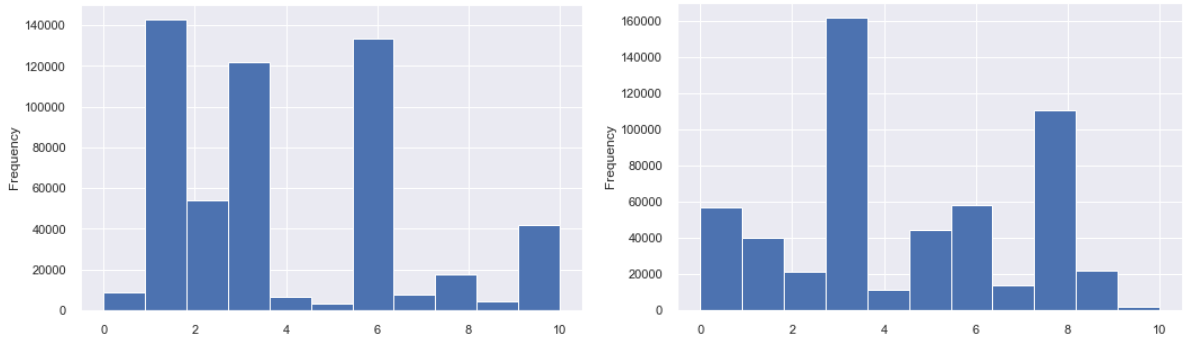

Figura 18: Comparativa de la cantidad de clientes en cada cluster para los distintos modelos de esporádicos

En dataE: En dataE1:

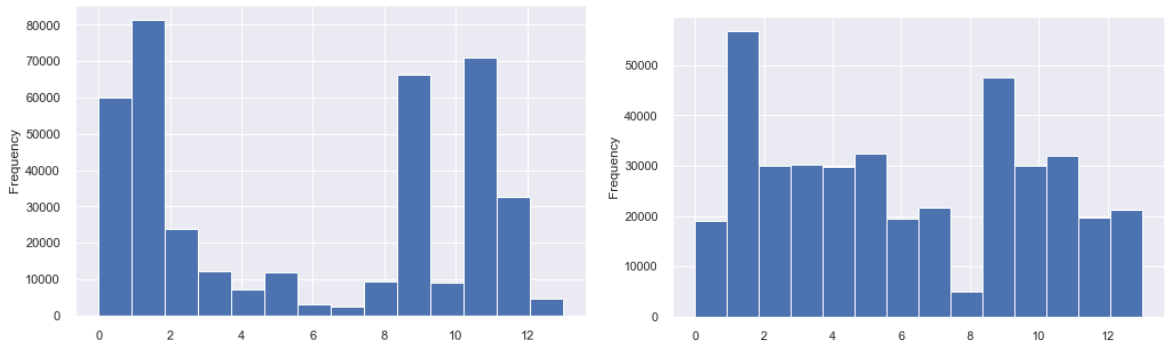

Figura 19: Comparativa de la cantidad de clientes en cada cluster para los distintos modelos de esporádicos

Se observa que la proporción de clientes en cada cluster es distinta en función del modelado.

Otra visualización interesante es la de visualizar por pares las variables. Por ejemplo, para el dataframe dataR1, se visualizan dos variables, y se obtiene:

Figura 20: Visualización de la clusterización por pares en dataR1

Se aprecia una correcta agrupación y diferenciación de las observaciones sobretodo en dirección a infinito positivo en la gráfica, sin embargo, en la parte más cercana a las coordenadas (0,0) se observa que se acumulan muchos puntos, por lo que sería interesante probar una agrupación con menor número de clústeres para el dataset de recurrentes.

Por último, se guardan los resultados de todos los modelados en un mismo dataset. En primer lugar, se van a concatenar los resultados de los modelados por pares, juntando los registros de los modelados realizados con dataR1 y dataE1, y los de dataR y dataR1, quedando dos datasets. Para ello, como se han clasificado en el caso de esporádicos en 14 clústeres, se mapean los clústeres de los recurrentes después de estos, equivaliendo el cluster 1 en el de recurrentes, al 15 en el del nuevo dataset que incluye ambos modelados. Luego, se vuelven a concatenar los dos datasets para finalmente obtener el único dataset.

		cluster	PCT_RETENIDAS	MODELO	K	NUM_VARIABLES
0		16	51.0	kmeans	11	87
1		15	51.0	kmeans	11	87
2		15	51.0	kmeans	11	87
3		15	51.0	kmeans	11	87
4		20	51.0	kmeans	11	87
...		...	...	...	...	...
1871357		11	94.0	kmeans	14	10
1871358		9	94.0	kmeans	14	10
1871359		11	94.0	kmeans	14	10
1871360		1	94.0	kmeans	14	10
1871361		3	94.0	kmeans	14	10

Figura 21: Data frame que incluye resultados de varios modelados

En este se incluye las variables:

- Identificador: Identificador de cada cliente.
- cluster: el cluster asignado a cada cliente.
- pct_retenidas:Porcentajes de variables no usadas del total de variables que se tenía en un principio.
- Modelo: algoritmo de clusterización utilizado.
- k: número de clústeres escogido para el modelado.
- num_variables: corresponde al número de variables que había en cada data frame para el modelado.

Además, se crea otra tabla para llevar a la base de datos, cuyas filas corresponden a todas las variables, tanto para el dataset de recurrentes como para el de esporádicos, para representar el peso que tienen estas en el PCA, así como otros datos importantes, como si han sido eliminadas durante el análisis de correlación.

La estructura de esta tabla es:

	index	0	tipo	variable_utilizada
0		0.587296	Recurrentes	SI
1		0.363717	Recurrentes	SI
2		0.290934	Recurrentes	SI
3		0.265030	Recurrentes	SI
4		0.220530	Recurrentes	SI
5		0.201291	Recurrentes	SI
6		0.190920	Recurrentes	SI
7		0.170928	Recurrentes	SI
8		0.141750	Recurrentes	NO
9		0.121964	Recurrentes	NO
10		0.116716	Recurrentes	NO
85		0.000054	Recurrentes	NO
86		0.000010	Recurrentes	NO
87		0.000000	Recurrentes	CORRELADA
88		0.000000	Recurrentes	CORRELADA
89		0.000000	Recurrentes	CORRELADA
90		0.000000	Recurrentes	CORRELADA

Figura 22: Data frame para informar del importancia/estado de las variables

Como se puede observar en la figura 22, aparecen todas las variables informando si se consideran importantes, si no, o si han sido eliminadas por correlación elevada, su importancia y el dataset al que corresponden. En este caso, se muestran las variables del dataset recurrentes, para el esporádicos se hace lo mismo, y se unifican para guardar la tabla en la base de datos.

Una vez hecho esto, ya se ha llevado a cabo la clusterización con dos modelados distintos, que puede pasar a ser analizada por los miembros de la empresa que corresponda, y replicada como se apunta anteriormente, con cualquier valor de k o con otras variables que se consideren oportunas.

Capítulo 4: Conclusión y líneas futuras

Finalmente, se ha podido demostrar que a través del uso de diversas técnicas de Inteligencia Artificial, como es la clusterización, es posible segmentar los clientes según sus perfiles, con el objetivo de mejorar la recomendación de servicios por parte de la empresa con la que se ha trabajado.

Se deja abierta la posibilidad de retomar este proyecto, aplicando un desarrollo alternativo a la hora de tratar el preprocesado de datos, o la clusterización, utilizando otras métricas al modelado ejecutado, pudiendo, por ejemplo, cambiar el número de clústeres para el Kmeans.

Además, se considera que sería interesante y muy aconsejable probar la clusterización con otros modelos, como el jerárquico o el basado en densidad.

Aparte de lo anteriormente mencionado, queda por ver si los resultados de este modelado aplicado en el entorno real de la empresa, obtiene buenos resultados, pero esto sería un objetivo que se apreciaría a largo plazo.

Chapter 4: Conclusion and future lines

Finally, it has been possible to demonstrate that through the use of various Artificial Intelligence techniques, such as clustering, it is possible to segment customers according to their profiles, with the aim of improving the recommendation of services by the company with which we have worked.

It is left open the possibility of retaking this project, applying an alternative development when dealing with data preprocessing, or clustering, using other metrics to the executed modeling, being able, for example, to change the number of clusters for the Kmeans.

In addition, it is considered that it would be interesting and highly advisable to test clustering with other models, such as hierarchical or density-based.

Apart from the above mentioned, it remains to be seen if the results of this modeling applied in the real environment of the company, obtains good results, but this would be an objective that would be appreciated in the long term.

Bibliografía

[1] ¿Qué es SQL Developer? (s.f.). Oracle. https://www.oracle.com/es/tools/technologies/whatis-sql-developer.html

[2] Anaconda documentation — Anaconda documentation. (s.f.). Anaconda. https://docs.anaconda.com

[3] Martín, G. (2018, 18 enero). Primeros pasos con Jupyter Notebook. Adictos al trabajo. https://www.adictosaltrabajo.com/2018/01/18/primeros-pasos-con-jupyter-notebook/

[4] Education, I. C. (2021, 21 mayo). Machine Learning. IBM. https://www.ibm.com/cloud/learn/machine-learning

[5] Team, E. (2021, 26 mayo). What is the Definition of Machine Learning? Expert.Ai. https://www.expert.ai/blog/machine-learning-definition/

[6] N. (2020, 15 julio). K-Means con Python paso a paso. Aprende Machine Learning. https://www.aprendemachinelearning.com/k-means-en-python-paso-a-paso/

[7] Universidad Nacional de Colombia Ingeniería de Sistemas y Computación, & León Guzmán, E. (s.f.). Métricas para la validación de Clustering [Diapositivas]. disi.unal.edu.co. https://disi.unal.edu.co/~eleonguz/cursos/mda/presentaciones/validacion_Clustering.pdf

[8] Brownlee, J. (2020, 30 junio). Why One-Hot Encode Data in Machine Learning? Machine Learning Mastery. https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/

[9] Amat Rodrigo, J. (2017, junio). Análisis de Componentes Principales. Ciencia de Datos. https://www.cienciadedatos.net/documentos/35_principal_component_analysis

[10]Elbow Method — Yellowbrick v1.3.post1 documentation. (s.f.). Yellowbrick. https://www.scikit-yb.org/en/latest/api/cluster/elbow.html

[11] sklearn.metrics.silhouette_score — scikit-learn 0.24.2 documentation. (s.f.). Scikit Learn. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html

[12] sklearn.cluster.KMeans — scikit-learn 0.24.2 documentation. (s.f.). Scikit Learn. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

[13] Intercluster Distance Maps — Yellowbrick v1.3.post1 documentation. (s.f.). Yellowbrick. https://www.scikit-yb.org/en/latest/api/cluster/icdm.html