



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ACATLÁN

Diplomado en Ciencia de Datos
Generación 23

Proyecto Módulo 3

Aldo Alejandro Gallegos Ruiz



Índice

1. Introducción	4
1.1. Objetivo	4
1.2. Sobre el conjunto de datos	4
1.3. Librería del proyecto	5
2. Necesidad de Negocio	6
2.1. Necesidad de Negocio	6
2.2. Definición de Unidad Muestral	6
2.3. Target para Modelación Supervisada	6
3. Limpieza de Datos y AEX	7
3.1. Campos geolocation lat y lon	7
3.2. Pasar campos a su formato correcto	7
3.3. Detección y tratamiento de valores ausentes	7
3.4. Prueba de Kolmogórov-Smirnov	8
3.5. Compras con payment.value cero	8
3.6. Variables continuas y discretas	8
3.7. Visualización gráfica	9
4. Construcción de la TAD	14
4.1. Campos de cuatrimestre	14
5. Análisis de correlación de variables	15
5.1. Variables predictoras	15
5.2. Detección de Outliers	15
5.3. Variables predictoras variables altamente correlacionadas	16
5.4. Identificación de clusters	17
5.5. Poder predictivo con KBest	18
6. Modelación supervisada	20
6.1. Regresión Lineal	20
6.2. Gradiente Descendiente Estocástico	20
6.3. Máquina Vector Soporte	21
6.4. Árbol de decisión	21
6.5. Redes neuronales	22
6.6. Gradiente Boosting	22
6.7. Redes Elásticas	23
7. Modelo ganador de Modelación Supervisada	25
8. Monitorización del modelo	25
8.1. Visualización gráfica de la monitorización	25
9. Conclusiones de Modelación supervisada	28
10. Modelación No Supervisada	29
10.1. Objetivo de Negocio	29
10.2. Detección de Outliers y Estandarización	29
10.3. Reducción de Dimensiones	30
10.3.1. Análisis de Componentes Principales	30
10.3.2. Análisis de Componentes Independientes	31
10.3.3. Análisis Factorial	31
10.3.4. Kernel PCA	31
10.3.5. Escalamiento Multidimensional	32
10.3.6. t-SNE	32
10.3.7. IsoMap	33

10.4. Determinar Cantidad de Clusters	34
10.4.1. CODO	34
10.4.2. Silueta	34
10.4.3. Calinski Harabasz	35
10.4.4. Davies Bouldin	35
10.4.5. Clustering Jerárquico	36
10.5. Modelos de Clusterización	38
10.5.1. K-Means	38
10.5.2. DBSCAN	38
10.5.3. Gaussian Mixture Model	39
10.6. Modelo Ganador de Modelación No Supervisada	39
10.7. Análisis de Clusterización	40
11. Perfilamiento	42
12. Estabilidad	43
13. Análisis de la Canasta	44
14. Conclusiones de Modelación No Supervisada	45

1. Introducción

1.1. Objetivo

El motivo de este documento es el poder detallar y explicar los procedimientos realizados para la elaboración de este proyecto de modelación supervisada.

En él, se realizan técnicas y procedimientos de ingeniería de datos y análisis exploratorio vistos a lo largos del primer módulo de este diplomado. Posteriormente, se pondrán a prueba los diversos métodos de modelación vistos en este módulo II.

1.2. Sobre el conjunto de datos

Este conjunto de datos fue proporcionado por Olist, una empresa de almacenes de los mercados brasileños. Olist conecta pequeñas empresas de todo Brasil con canales sin complicaciones y con un único contrato. Esos comerciantes pueden vender sus productos a través de la tienda Olist y enviarlos directamente a los clientes utilizando los socios logísticos de Olist.

Después de que un cliente compra el producto en Olist Store, se notifica al vendedor para que complete ese pedido. Una vez que el cliente recibe el producto, o vence la fecha estimada de entrega, el cliente recibe una encuesta de satisfacción por correo electrónico donde puede dar una nota de la experiencia de compra y anotar algunos comentarios.

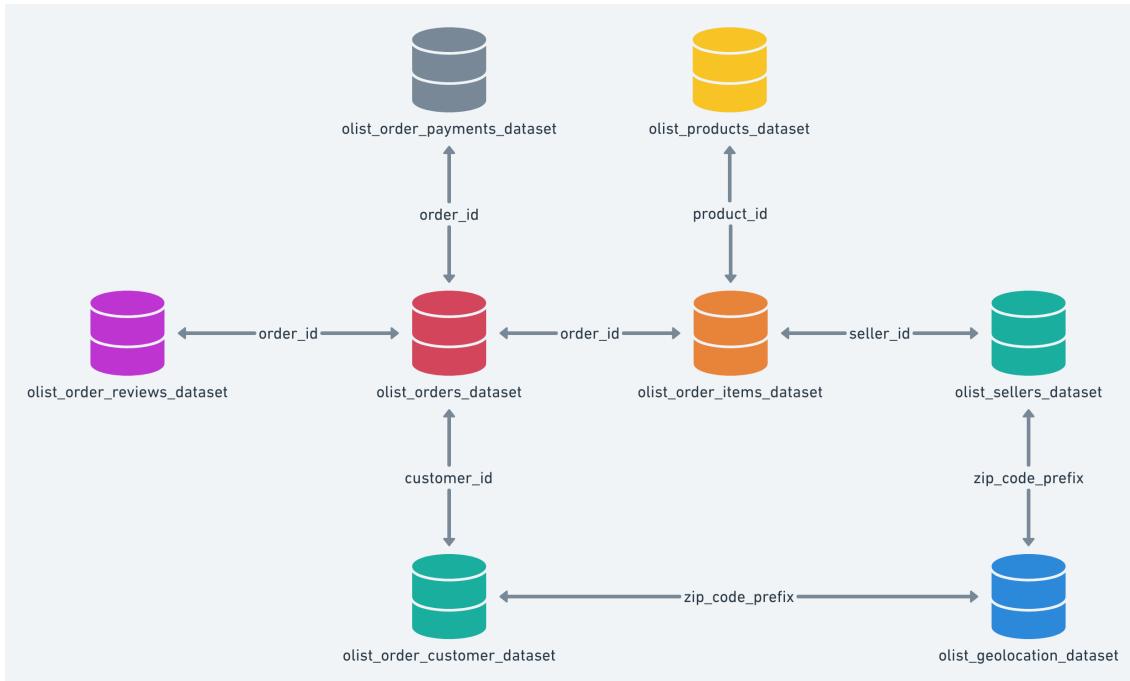
Contamos en total con 9 distintas bases de datos las cuales se interrelacionan entre sí por medio de llaves primarias (o id's). En ellas detalla la información de la compra del cliente como la fecha de realización de la compra, el monto de la compra, tipo de pago, en cuántos pagos se realizó la compra y el estatus de la compra (si ya fue entregada o no). En cuanto a información del cliente, contamos con su estado y ciudad de residencia. Para los vendedores y el producto, tenemos la ciudad y estado donde se ubicaba originalmente el producto comprado y el vendedor, entre otros. Finalmente, tenemos el score que dio el cliente a su compra, con lo cual podemos tener retroalimentación de la calidad de servicio con el que contó.

Con todo lo mencionado, contamos con 44 campos en total y 3,619,282 registros, a los cuales a continuación aplicaremos técnicas de limpieza de datos, para posteriormente generar ingeniería de características para así escoger las variables que serán predictoras en nuestro modelo.

Es importante destacar que esta base de datos de una empresa brasileña, por lo que las ciudades, estados y opiniones, se encuentran en portugués.

Nombre de las bases que conformarán nuestro DataFrame:

- olist_customers_dataset (84525, 5)
- olist_geolocation_dataset (850139, 5)
- olist_order_items_dataset (95752, 7)
- olist_order_payments_dataset (88303, 5)
- olist_order_reviews_dataset (84340, 7)
- olist_orders_dataset (84525, 8)
- olist_products_dataset (28008, 9)
- olist_sellers_dataset (2631, 4)



1.3. Librería del proyecto

Además del código donde se realizará todo el proyecto de modelado, se cuenta con un archivo .py llamado modulo3_proyecto_Libreria, en el cual se encuentran todas las funciones que se utilizarán a lo largo del proyecto, para ello importamos este archivo como librería, cuya clase donde se encuentran las funciones se llama "projeto_ecommerce", el cual será guardado como un objeto llamado "pec".

2. Necesidad de Negocio

2.1. Necesidad de Negocio

Con la información que contamos, la empresa Olist Store requirió de nuestro servicio para conocer y poder anticipar los montos de compra que estarán realizando sus clientes. Esto para saber qué tanto consumen sus clientes para con ello saber qué tipo de productos se le puede recomendar a cada cliente dependiendo del costo de entrega. Es por ello que confiaron en que nosotros les ofrecemos una predicción con base en nuestro modelo y conocimientos en Ciencia de Datos para así dar respuesta a su necesidad.

2.2. Definición de Unidad Muestral

Como mencionamos anteriormente, nuestra base de datos recopila información de las compras que los clientes realizan en Olist Store, por lo que es esta la unidad muestral, la compra, pues cada cliente realiza compras electrónicas y esa unidad muestral la podemos definir como cada registro del DataFrame.

2.3. Target para Modelación Supervisada

Dado que queremos conocer el monto de la compra que realizará el cliente, nuestra variable objetivo será el payment_value, es decir, el monto total que pagó el cliente por su pedido. Este monto contempla el costo del producto y su envío, los cuales son campos con los que también contamos, los cuales son price y freight_value (el costo de envío no siempre es el mismo ni representa el mismo porcentaje de la compra).

3. Limpieza de Datos y AEX

A partir de aquí hacemos uso de todo el conjunto de datos en un solo DataFrame, es decir de las 9 tablas, hacemos un merge (con cruce tipo inner) con lo cual nos resultará un DataFrame único, el cual llamaremos merged_df, y como ya mencionamos, se compone de 44 campos en total.

3.1. Campos geolocation lat y lon

Estos son campos generan muchos registros duplicados debido a que registran la ubicación exacta de cada unidad de producto. Por lo que decidimos presindir de estos para nuestro modelo.

Con lo cual nuestra base se reduce a solo 57,968 registros.

3.2. Pasar campos a su formato correcto

Antes de realizar un análisis más estadístico de nuestros datos, tuvimos que pasar a formato fecha los campos: order_purchase_timestamp, order_approved_at, order_delivered_carrier_date, order_delivered_customer_date, order_estimated_delivery_date, shipping_limit_date pues nos serán de ayuda a la hora de hacer ingeniería de características y posterior monitorización de nuestro modelo.

3.3. Detección y tratamiento de valores ausentes

Notamos que los siguientes campos de nuestro DataFrame cuentan con valores nulos: Campos de fechas

- order_approved_at
- order_delivered_carrier_date
- order_delivered_customer_date

Campos de los comentarios de los clientes

- review_comment_title
- review_comment_message

Campos del largo del nombre, descripción y cantidad de fotos

- product_category_name
- product_name_lenght
- product_description_lenght
- product_photos_qty

Campos de dimensiones del producto

- product_weight_g
- product_length_cm
- product_height_cm
- product_width_cm

Lo cual consideramos como “errores en la captura de datos”, pues notamos que algunos ya se encuentran en status de entregado, además de tener un precio e incluso un score del cliente, por lo cual consideramos conveniente realizar una imputación a estos, teniendo bien en cuenta que la cantidad de nulos de cada campo no sobrepasa el 5 % del total de registros.

Es por ello por lo cual en los campos relacionados a las dimensiones del producto (product_weight, product_height), realizaremos una imputación por la mediana, así como para los campos del nombre y descripción del producto.

En cuanto a los campos de fechas que tienen nulos, lo que hicimos fue imputar por la media de la diferencia en días con respecto de la fecha de compra del producto, es decir, lo que normalmente tarda en realizarse esas etapas de las compras después de que se efectuaron.

3.4. Prueba de Kolmogórov-Smirnov

Una vez realizado esos cambios, hicimos una prueba de Kolmogórov-Smirnov para validar que los cambios realizados a estos campos no afectaran la distribución Normal de los mismos.

A continuación mostramos que en efecto mantienen una distribución Normal.

		0	1
0	order_approved_at	1.0	
1	order_delivered_carrier_date	1.0	
2	order_delivered_customer_date	1.0	
3	product_weight_g	1.0	
4	product_length_cm	1.0	
5	product_height_cm	1.0	
6	product_width_cm	1.0	
7	product_name_lenght	1.0	
8	product_description_lenght	1.0	
9	product_photos_qty	1.0	

3.5. Compras con payment_value cero

Por lo que observamos en la opinión de los clientes, cuando el precio del pago está en cero es porque este producto no fue entregado y la opinión de los clientes es negativa, es decir, es un fallo en el servicio. Sin embargo, debido a la necesidad del negocio, lo que nosotros buscamos es predecir el precio del producto que comprará el cliente, no si el producto llegó al cliente, por lo que es mejor remover estos registros, pues resultan atípicos.

```
merged_df.loc[merged_df['payment_value']==0, 'review_comment_message']
✓ 0.0s
16290 Produto demorou a chegar e veio diferente do q...
16291 Produto demorou a chegar e veio diferente do q...
29701 A mercadoria não foi entregue. Entrara em cont...
29702 A mercadoria não foi entregue. Entrara em cont...
41629 NaN
54349 A mercadoria não foi entregue. Entrara em cont...
54350 A mercadoria não foi entregue. Entrara em cont...
Name: review_comment_message, dtype: object
```

3.6. Variables continuas y discretas

Teniendo los campos en su formato correcto, lo que realizamos ahora es la identificación de las variables continuas y discretas, ya que con ello podremos más adelante hacer una visualización gráfica dependiendo de si son continuas y discretas. Por lo cual, la identificación de estas variables fue de la siguiente manera, también aprovechamos para mencionar la descripción de estos campos, ya que nos serán de ayuda para entender más adelante la construcción de la TAD (Tabla Analítica de Datos) y posiblemente escoger nuestras variables predictoras:

VARIABLES CONTINUAS

- price: Precio del producto

- freight_value: Costo de envío
- product_name_length: Largo del nombre del producto
- product_description_length: Largo de la descripción del producto
- product_photos_qty: Cantidad de fotos del producto
- product_weight_g: Peso en gramos del producto
- product_length_cm: Largo en centímetros del producto
- product_height_cm: Alto en centímetros del producto
- product_width_cm: Ancho en centímetros del producto
- seller_zip_code_prefix: Prefijo del código postal del vendedor
- geolocation_zip_code_prefix: Prefijo del código postal del producto
- customer_zip_code_prefix: Prefijo del código postal del cliente

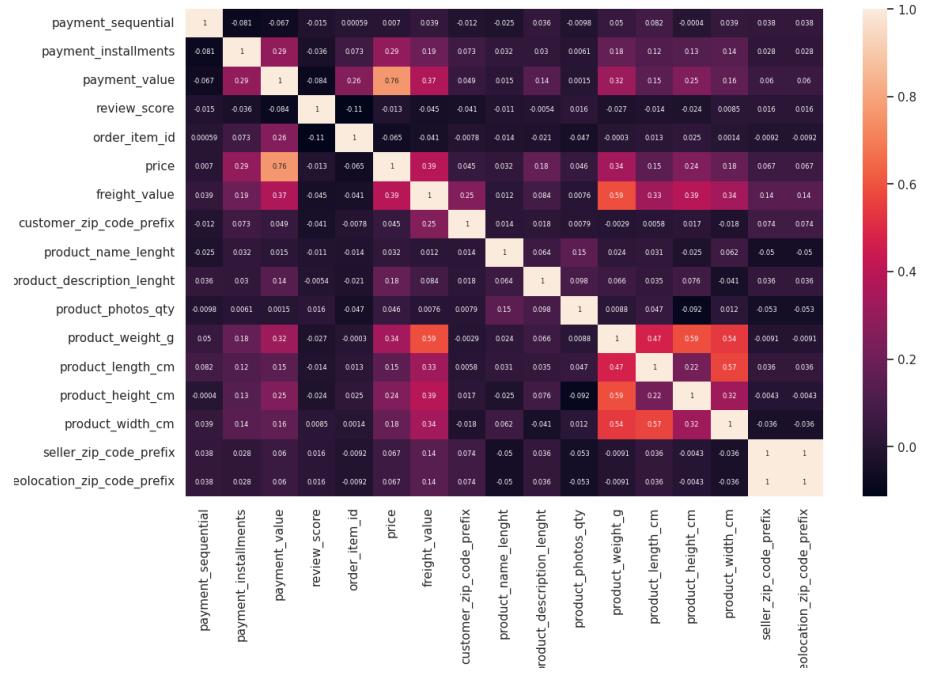
Variables discretas

- payment_sequential: Cantidad de abonos para cubrir el pago
- payment_installments: Cantidad de cuotas que se le aplican al pedido
- review_score: Puntaje de 1 a 5 que el cliente dio al servicio recibido por su pedido
- payment_type: Tipo de pago con el que el cliente hizo la compra
- order_status: Estatus del pedido
- customer_city: Ciudad de residencia del cliente
- customer_state: Estado de residencia del cliente
- product_category_name: Nombre de la categoría del producto
- seller_city: Ciudad de residencia del vendedor
- seller_state: Estado de residencia del vendedor
- geolocation_city: Ciudad de la ubicación del producto (en almacén)
- geolocation_state: Estado de la ubicación del producto (en almacén)

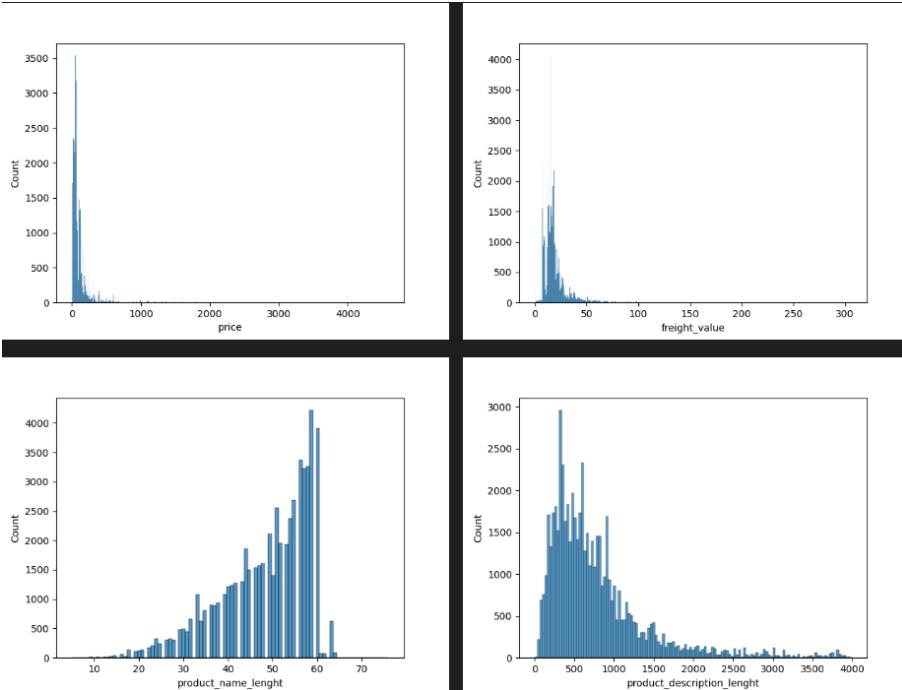
3.7. Visualización gráfica

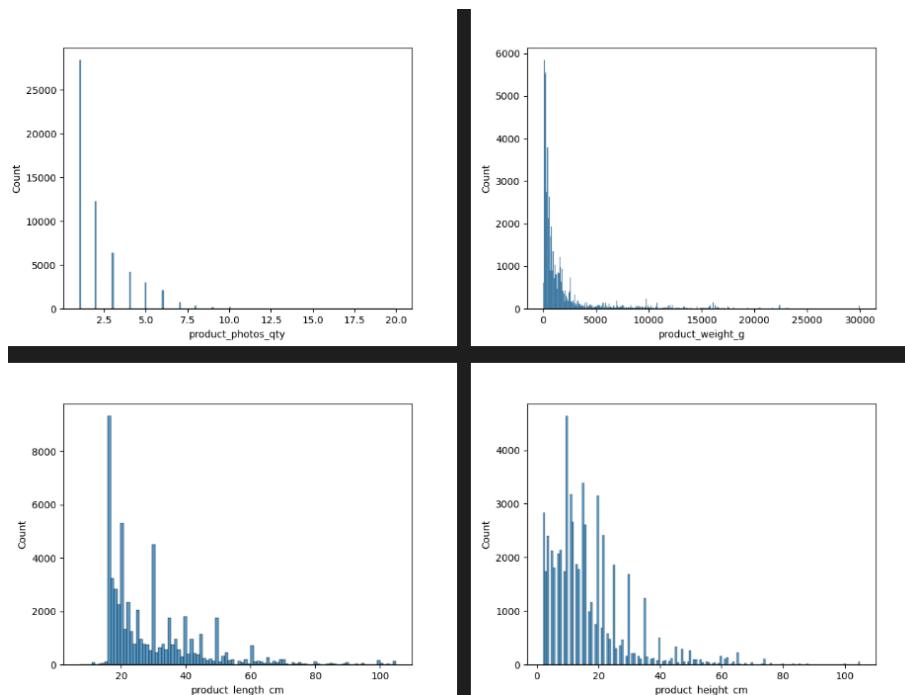
Como mencionamos anteriormente, dividiremos la visualización con el tipo de gráfica dependiendo de si los campos son continuos o discretos. Para ello utilizaremos mapas de calor, histogramas, gráfica de barras y de densidades. A continuación mostramos cómo se observa el comportamiento de nuestros campos.

Mapa de calor Podemos ver que no existe correlación absoluta de 1 entre ninguna de nuestras variables, de hecho en general las correlaciones son bajas, incluso para campos como el price y freight_value.



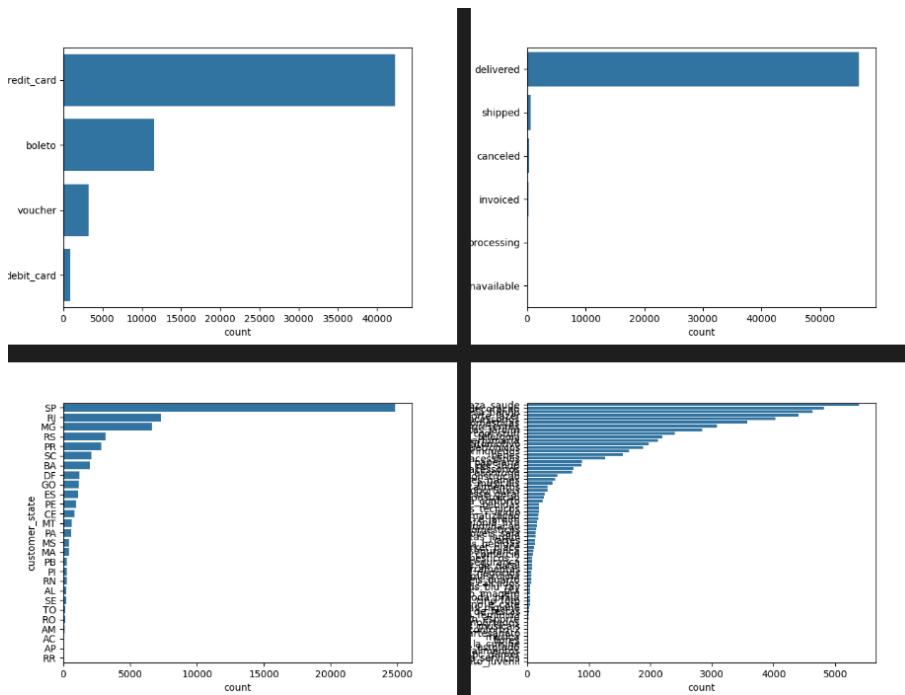
Histogramas

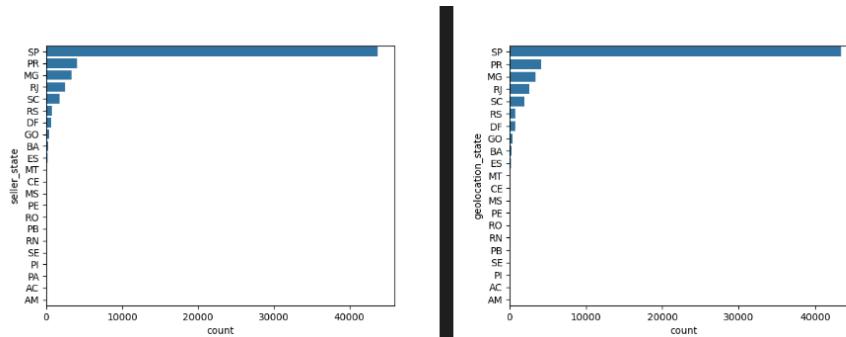




Vale destacar el campo price se concentra entre 0 y 1000 reales (moneda brasileña) y freight está entre 0 y 50, sin embargo para ambos casos existen ciertas compras que se fueron montos muy alejados de estos rangos.

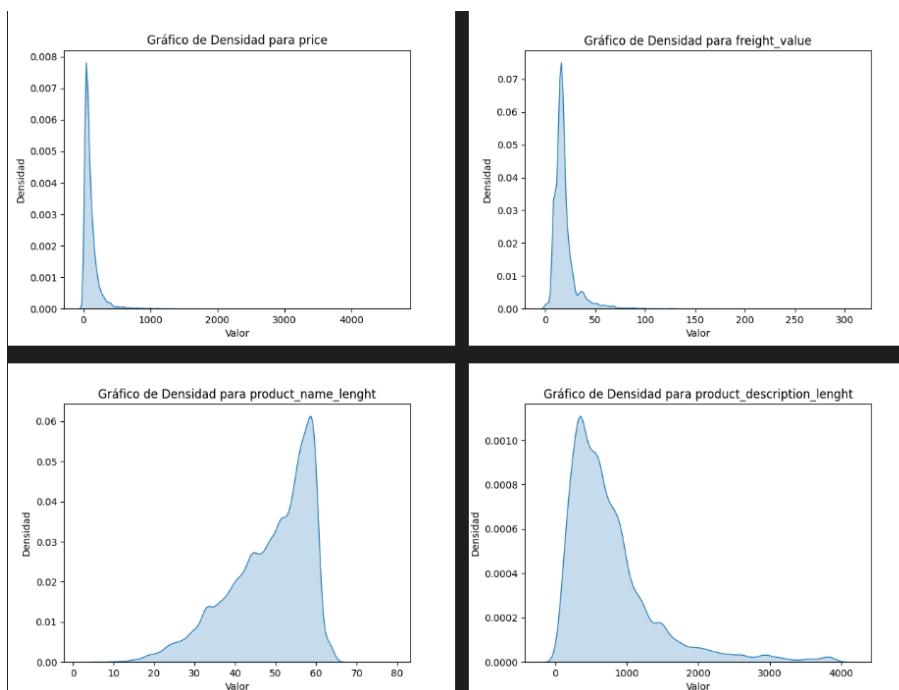
Gráfica de barras

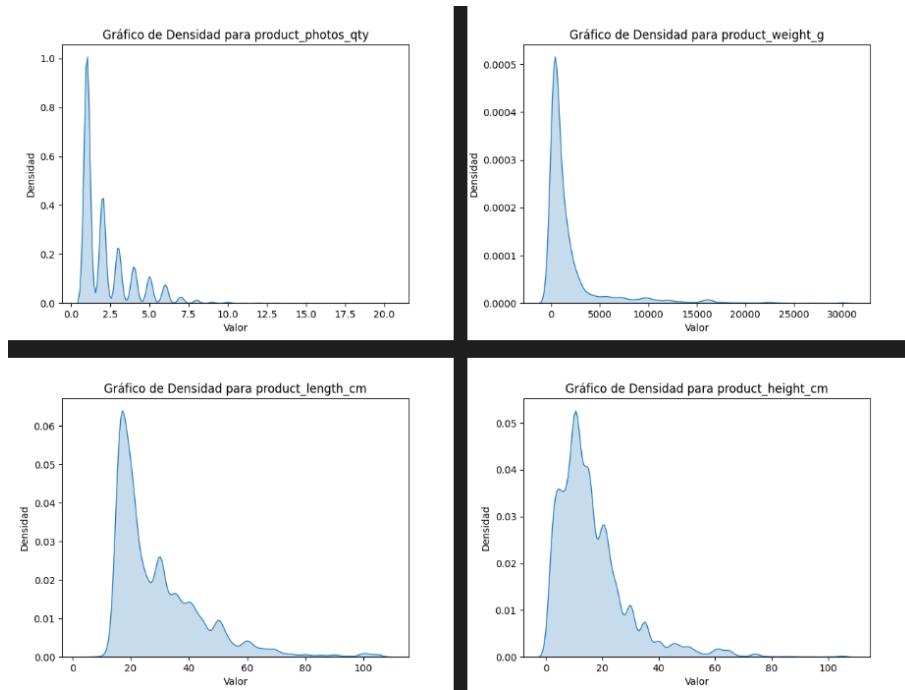




Podemos observar que la gran mayoría de las compras ya están entregados, mientras que el método de pago que hacen los clientes es con tarjeta de crédito. Ahora bien, la ciudad donde se ubica el vendedor y el producto se concentran más en los estados grandes de Brasil (Sao Paola, Rio de Janeiro, Minas Gerais), mientras que los clientes no necesariamente están tan concertados en esos estados, destacando estados como Bahía y el Distrito Federal.

Gráfica de Densidades





Esta es otra forma de visualizar los datos similar a lo que vimos con los histogramas.

4. Construcción de la TAD

Fueron varias las ideas para crear variables basados en los datos con los que contábamos, a continuación enlistamos las variables que creamos :

Variables de compras

- Obtener el año de la compra, ya que por temas inflacionarios, puede que los precios aumenten
- Cantidad de compras que hacen los clientes por vendedor y por producto
- Cantidad de compras que se hace en una orden por vendedor y por producto

Variables de costo de envío

- Queremos saber la mediana, el mínimo y máximo de lo que pagaron los clientes por envío
- Queremos saber la mediana, el mínimo y máximo de lo que costó envío por producto

Secuencias de pago

- Saber mínimo, máximo y mediana de los pagos que realizaron los clientes por su producto

Tamaño y peso del producto

- Queremos conocer qué tan grandes y qué tan pesados son los productos que compra el cliente, por lo que obtuvimos su mediana, su mínimo y su máximo de cada producto que compró el cliente

Productos comprados con tarjeta de crédito

- Queremos ver por cliente y por producto de cuántas compras fueron con tarjeta de crédito, ya que puede que si fue por esta forma el precio es porque el monto del pago fue más alto

Cuotas de pago

- Suponemos que mientras más cuotas de pago se realice al producto, más caro es, por lo que queremos saber el máximo y mínimo de las cuotas que se le ha hecho al cliente.

Cantidad de pedidos que son de la misma ciudad

- Suponemos que si la ubicación del producto es distinta a la ubicación del cliente, el pago va a ser mayor, por lo que creamos una bandera que nos da esa información.

Por último, buscamos conocer cuánto tiempo pasa entre el momento en el que el cliente hace su compra y se entrega, así como la fecha estimada de entrega, la fecha en la que se embarcó para su envío y su fecha límite de embarque.

- Diferencia en días entre fecha de pedido y fecha de `order_delivered_carrier_date`
- Diferencia en días entre fecha de pedido y fecha de `order_delivered_customer_date`
- Diferencia en días entre `order_delivered_customer_date` y `order_estimated_delivery_date`
- Diferencia en días entre `order_delivered_carrier_date` y `shipping_limit_date`

4.1. Campos de cuatrimestre

Adicionalmente, creamos un campo con el cuatrimestre y el año en que se realizó la compra, esto nos servirá posteriormente para la realización del monitoreo de nuestro modelo. Sabemos que los registros de las compras van del último Q del 2016 al tercero del 2018.

5. Análisis de correlación de variables

Una vez obtenidas las variables que generamos en la ingeniería de características, nos aseguraremos de que estas no tengan valores infinitos, así como que no tengan campos unarios.

Antes, necesitamos convertir a variables numéricas los campos tipo Object, los cuales son los siguientes:

- payment_type
- order_status
- customer_city
- product_category_name
- customer_state
- seller_city
- seller_state
- geolocation_city
- geolocation_state

Para ello, utilizamos el método de LabelEncoder() y así obtuvimos los campos como tipo Integer.

```
label_encoder = LabelEncoder()

for g in granulares:
    merged_df[g] = label_encoder.fit_transform(merged_df[g])

✓ 4.5s
```

5.1. Variables predictoras

Una vez teniendo todas nuestras variables formato numérico, creamos una lista llamada predictors, la cual contiene básicamente todas las variables que creamos y transformamos las cuales son tipo numéricas.

5.2. Detección de Outliers

Una vez asegurado esto, lo que hicimos fue poner en práctica la aplicación de modelos de detección de outliers, para que estos nos digan si existen en nuestro DataFrame.

Los modelos que usamos son IQR, Z-Score e Isolation Forest, obteniendo los siguientes resultados:

- IQR fue el método que más detectó outliers, pasando de 57961 registros a solo sec quedó con 13768 registros
- Z-Score no detectó ningún outlier, pues se mantuvo con los 57961 registros originales
- Iso-Forest fue el intermedio entre los dos pues de 57961 se quedó con 53661

```

pec.iqr(merged_df)

Tamaño original 57961 Tras aplicar IQR: 13768 registros

pec.z_score(merged_df)

Tamaño original: 57961 Al aplicar el método Z-Score: 57961

pec.iso_forest(merged_df,predictors)

Tamaño original: 57961 Al aplicar el método Iso Forest: 53661

```

5.3. Variables predictoras variables altamente correlacionadas

Consideramos como variables predictoras a todas aquellas variables numéricas a las que hicimos un tratamiento de dato y las que creamos en la TAD, por ello para quedarnos solo con las que se relacionan más con la variable objetivo y serán más importantes para el modelo, hicimos una correlación de las variables, encontrando que las variables relacionadas con el precio y el costo de envío son las más relacionadas con el payment_value.

Sin embargo, vale destacar que el peso y el tamaño del producto son variables que están muy relacionadas con el pago que realiza el cliente. Ahora bien, las variables que menos se relacionan con el target con las del nombre de la categoría del producto y las ciudades del vendedor, el cliente y del producto.

Estas fueron las variables que más se correlaciona con el target:

	target_abs
payment_value	1.000000
price	0.761564
median_freight_prod	0.392176
max_freight_cust	0.381767
suma_freight_prod	0.378762
freight_value	0.367643
median_freight_cust	0.367303
suma_freight_cust	0.355992
max_peso_prod	0.324816
product_weight_g	0.318918
med_peso_prod	0.318377
min_peso_prod	0.313613
max_volumen_prod	0.297557
max_freight_prod	0.297184
payment_installments	0.294606
pay_install_prom	0.293805
volumen_producto	0.291713
med_volumen_prod	0.290796
pay_install_max	0.288847
min_volumen_prod	0.286734
product_height_cm	0.245604
product_width_cm	0.163476
product_length_cm	0.148314

Mientras que estas fueron las que menos:

payment_sequential	0.066630
geolocation_zip_code_prefix	0.060295
seller_zip_code_prefix	0.060295
cantidad_vendedores_por_compra	0.059135
cantidad_compras	0.059135
cantidad_productos_por_compra	0.059135
cantidad_vendedores_por_cliente	0.059135
cantidad_productos_por_cliente	0.059135
customer_state	0.056427
dif_purchase_delivered_customer	0.055626
customer_zip_code_prefix	0.049052
geolocation_state	0.045133
seller_state	0.043577
pedidos_misma_ciudad	0.032326
geolocation_city	0.030932
seller_city	0.029285
suma_payment_sequential	0.027303
dif_delivered_customer_estimated	0.020128
product_name_length	0.015263
customer_city	0.014228
order_status	0.013967
product_category_name	0.004206
product_photos_qty	0.001493

Sin embargo, la correlación de las variables con el target en general es bajo aún con la ingeniería de características, por lo que esto ya nos da una idea del desempeño que podría tener nuestro modelo.

Es por ello, por lo que consideramos conveniente remover las variables que tuvieran una correlación menor 0.01, las cuales son: purchase_year, dif_delivered_customer_estimated, product_category_name. Con ello existirán registros duplicados, es decir, que quitando estos campos, es como si fuese la misma compra, por lo que es redundante considerar el mismo registro. Es por ello que para nuestra tabla merge_df aplicaremos un drop_duplicates()

5.4. Identificación de clusters

Realizamos este método para poder visualizar qué tanto se relacionan o se parecen las variables del modelo, para que con ello, las variables con una alta correlación se irán agrupando por clusters. Con ello, utilizamos VarClusHi() para este análisis y con ello identificamos que existen agrupaciones dentro de nuestras variables predictoras, es decir, mediante el RS.Ratio, la variabilidad en una variable se puede explicar mediante las demás variables en el mismo cluster. En total se identificaron 13 clusters, en la imagen mostramos los cuatro con mayor cantidad de variables.

Cluster		Variable	RS_Own	RS_NC	RS_Ratio
0	0	product_weight_g	0.860433	0.408757	0.236058
1	0	product_length_cm	0.355638	0.124402	0.735911
2	0	product_height_cm	0.514879	0.181848	0.592947
3	0	product_width_cm	0.469678	0.134676	0.612860
4	0	volumen_producto	0.920880	0.378699	0.127346
5	0	med_volumen_prod	0.921832	0.379232	0.125922
6	0	max_volumen_prod	0.915819	0.374551	0.134593
7	0	min_volumen_prod	0.916668	0.379750	0.134353
8	0	med_peso_prod	0.860843	0.409092	0.235496
9	0	max_peso_prod	0.855590	0.404649	0.242562
10	0	min_peso_prod	0.858556	0.407074	0.238552
11	1	cantidad_vendedores_por_cliente	1.000000	0.562334	0.000000
12	1	cantidad_productos_por_cliente	1.000000	0.562334	0.000000
13	1	cantidad_compras	1.000000	0.562334	0.000000
14	1	cantidad_vendedores_por_compra	1.000000	0.562334	0.000000
15	1	cantidad_productos_por_compra	1.000000	0.562334	0.000000
16	2	dif_purchase_delivered_carrier	1.000000	0.122352	0.000000
17	2	med_dif_purchase_delivered_carrier	1.000000	0.122352	0.000000
18	2	min_dif_purchase_delivered_carrier	1.000000	0.122352	0.000000
19	2	max_dif_purchase_delivered_carrier	1.000000	0.122352	0.000000
20	3	seller_zip_code_prefix	0.796100	0.191172	0.252093
21	3	seller_state	0.779342	0.150850	0.259857
22	3	geolocation_zip_code_prefix	0.796100	0.191172	0.252093
23	3	geolocation_state	0.799568	0.156935	0.237742

5.5. Poder predictivo con KBest

Otra forma de ver cómo se puede reducir dimensiones en el DataFrame, es seleccionar las mejores variables predictoras mediante SelectKBest(), lo que se hace es utilizar una función de puntuación (como el estadístico de chi-cuadrado) para evaluar la relevancia de cada característica en relación con la variable objetivo.

En este caso visualizamos que las mejores 10 variables son:

- price
- freight_value
- cantidad_vendedores_por_compra
- cantidad_productos_por_compra
- rel_freight_payment
- median_freight_prod
- suma_freight_prod
- median_freight_cust

- max_freight_cust
- suma_freight_cust

```
mejores = pec.seleccionar_kbest(merged_df[predictors+[tar]],"payment_value",num_variables_deseadas=10)
mejores

['price',
'cantidad_vendedores_por_cliente',
'cantidad_productos_por_cliente',
'cantidad_compras',
'cantidad_vendedores_por_compra',
'cantidad_productos_por_compra',
'rel_freight_payment',
'median_freight_cust',
'max_freight_cust',
'suma_freight_cust']
```

6. Modelación supervisada

Comenzamos ya la parte de la aplicación de modelos para poder predecir el pago que realizará el cliente es su pedido.

Para empezar, realizaremos un split para entrenar y validar nuestros resultados, los registros de DataFrame de validación será del 20 % del DataFrame completo. es decir de los 52692 registros, para test tomaremos 10539 registros y 53 campos.

Lo que se hará a continuación será poner a prueba una serie de algunos de los modelos de predicción vistos en el curso, analizar los resultados a nivel técnico de distintas métricas de desempeño que comentaremos a continuación y basados en ello decidiremos qué modelo será el idóneo para utilizar y dar respuesta a la necesidad de nuestro cliente.

6.1. Regresión Lineal

Iniciaremos nuestra pruebas con una Regresión Lineal, el cual es un modelo sencillo pero pero que nos ayuda mucho en darnos una idea de qué tan buenas son nuestras variables para predecir nuestro objetivo. El parámetro que indicamos que se tulice fue fit_intercept=True lo que indica que el modelo calculará el intercepto.

Con ello, vemos un Mean Absolute Error de 30.59 en entrenamiento y 31.75 en validación. lo cual es muy bueno dado que la desviación estándar del payment_value por sí solo es de 232.98, cual nos una idea de que nuestro modelo funciona bien y nos podemos pasar a otras metodologías más complejas. Mientras que el Mean Squared Error es de 7606.75 en entrenamiento y 9934.16 en validación, significa que si el modelo discrepa en promedio por 87 reales (moneda brasileña), esta métrica resalta este error elevándolo al cuadrado.

```
y_p=lr.predict(xt)
print('Train:')
print(f'MAE: {mean_absolute_error(yt, y_p)}')
print(f'MSE: {mean_squared_error(yt, y_p)}')
print(f'R2: {r2_score(yt , y_p)}')
✓ 0.0s

Train:
MAE: 30.591455608321958
MSE: 7606.757668940821
R2: 0.8163207786952629

y_p=lr.predict(xv)
print('Validation:')
print(f'MAE: {mean_absolute_error(yv, y_p)}')
print(f'MSE: {mean_squared_error(yv, y_p)}')
print(f'R2: {r2_score(yv , y_p)}')
✓ 0.0s

Validation:
MAE: 31.751982538743956
MSE: 9934.160956922884
R2: 0.788511708898153
```

6.2. Gradiente Descendiente Estocástico

Podemos a prueba el SGDRegressor, ya que, parecido al modelo anterior, este resulta ser eficiente y simple para poder interpretar nuestros resultados. En este caso indicamos que solo realice 100 iteraciones hará sobre el conjunto de datos hasta llegar a un error menor al tolerado, el cual por defecto es de 1e-3.. En este modelo de regresión caso aplicamos el MAE y observamos resultados similares, 32.02 en entrenamiento y 32.98 en validación, en el caso de MSE vemos 7736.47 en entrenamiento y 9944.33 en validación,

finalmente para el R2 vemos 0.81 en entrenamiento y 0.78 en validación.

```
El MAE del entrenamiento es: 32.028587158487426
El MAE de la validación es: 32.98555732660944
El MSE del entrenamiento es: 7736.477081664498
El MSE de la validación es: 9944.336901489265
El R2 del entrenamiento es: 0.8131884637518699
El RS de la validación es: 0.7882950732772865
```

6.3. Máquina Vector Soporte

El SVM nos sirve para ver relaciones no lineales entre las características y el target mediante el uso del kernel que en este caso usamos, el RBF (Radial Basis Function) y que nos ayuda a transformar los datos en un mayor espacio donde es más probable que las características sean linealmente separables. Teniendo esto en mente vemos que sin embargo, no se desempeña tan bien como los modelos anteriores. Para el MAE obtenemos 48.35 en entrenamiento y 49.90 en validación, para MSE vemos 26539.32 en entrenamiento y 31954.68 en validación, y para el R2 vemos 0.81 en entrenamiento y 0.78 en validación.

```
El MAE del entrenamiento es: 48.357625958990965
El MAE de la validación es: 49.90757197066805
El MSE del entrenamiento es: 26539.32242822073
El MSE de la validación es: 31954.682898868927
El R2 del entrenamiento es: 0.8131884637518699
El R2 de la validación es: 0.7882950732772865
```

6.4. Árbol de decisión

Utilizaremos un Árbol de decisión ahora pues además de ser uno de los modelos más populares debido a su fácil interpretabilidad, nos ayuda a que tomará las variables más relevantes del conjunto de datos para ir creando ramas. Como lo dice su nombre, cada nivel de división de los componentes se compra como un rama y cada división o bifurcación se comporta como una hoja. En nuestro caso pedimos que tomara una profundidad de 5 (como si fueran 5 ramas) y que la hoja de cada rama que se pudiera dividir solo si la muestra tuviera al menos 30 registros y 12 en alguna de las dos hojas. Cada rama se dividirá por la metodología del error absoluto.

Con ello podemos ver que se obtiene el mejor comportamiento del MAE, siendo de 28.95 en entrenamiento y 29.91 en validación, mientras que el MSE en entrenamiento es de 8616.03 y validación 11105.23. Cabe destacar el R2, ya que en entrenamiento obtenemos 0.79 y 0.76 lo cual es bueno pues el modelo no sobreajusta tanto como en la Regresión Lineal.

```
regressions = tree_reg.predict(Xt)
print('Train:')
print(f'MAE: {mean_absolute_error(yt, regressions)}')
print(f'MSE: {mean_squared_error(yt, regressions)}')
print(f'R2: {r2_score(yt, regressions)}')
✓ 0.1s

Train:
MAE: 28.952289991222454
MSE: 8616.039785356912
R2: 0.7919497968277786
```

```

regressions = tree_reg.predict(xv)
print('Val:')
print(f'MAE: {mean_absolute_error(yv, regressions)}')
print(f'MSE: {mean_squared_error(yv, regressions)}')
print(f'R2: {r2_score(yv, regressions)}')
✓ 0.0s

Val:
MAE: 29.913756997817625
MSE: 11105.237168505075
R2: 0.7635806746808196

```

6.5. Redes neuronales

Las Redes Neuronales son importantes ofrecen varias ventajas en el aprendizaje automático y el procesamiento de datos, además de que más adelante en el curso utilizaremos a mayor profundidad este tipo de modelos.

Lo que hicimos en este modelo fue tomar tres capas ocultas pero no fue necesario usar muchas neuronas, para evitar un sobreajuste, usamos 5 neuronas por capa, además de un tamaño de lote de 100. Finalmente lo que más nos ayudó fue usar 'adam' (el que es por defecto) como técnica de optimización pues este adapta la tasa de aprendizaje para cada parámetro, lo que permite que el algoritmo ajuste de manera efectiva la tasa de aprendizaje a lo largo del proceso de entrenamiento. Además establecimos una tasa de aprendizaje constante mientras que la pérdida disminuye, luego lo reduce si la pérdida se estanca.

Los resultados fueron los siguientes: Para entrenamiento obtuvimos un MAE de 38.83 y R2 de 0.75, mientras que para validación fue un MAE de 39.21 y R2 de 0.73

```

regressions = tree_reg.predict(xt)
print('Train:')
print(f'MAE: {mean_absolute_error(yt, regressions)}')
print(f'MSE: {mean_squared_error(yt, regressions)}')
print(f'R2: {r2_score(yt, regressions)}')
✓ 0.0s

Train:
MAE: 32.54311004897333
MSE: 8548.177003645342
R2: 0.7935884691035171

regressions = tree_reg.predict(xv)
print('Val:')
print(f'MAE: {mean_absolute_error(yv, regressions)}')
print(f'MSE: {mean_squared_error(yv, regressions)}')
print(f'R2: {r2_score(yv, regressions)}')
✓ 0.0s

Val:
MAE: 34.050844932218276
MSE: 11015.213017285609
R2: 0.7654971982787215

```

6.6. Gradiente Boosting

Este modelo es usado debido a su alto rendimiento y flexibilidad en tareas de aprendizaje automático, pero no siempre resulta conveniente en muchos casos usarlo pues se corre el riesgo de sobreajuste debido a que el modelo aprende mucho en entrenamiento, como si estuviera memorizando, entonces en validación, si se encuentra con información distinta puede fallar. Establecimos que se usan todos los nucleos del CPU para su ejecución lo que conlleva acelerar significativamente el entrenamiento del modelo, además establecimos que se creen 50 árboles de decisión para construir el modelo.

Los resultados fueron los siguientes:
Para entrenamiento obtuvimos un MAE de 5.08 y R2 de 0.99, mientras que para validación fue un MAE de 7.43 y R2 de 0.96.

Vemos que los resultados son casi perfectos, sin embargo no es recomendable su uso a la hora de quiere enviarlo a producción por lo mencionado anteriormente, es decir, si se tiene información muy distinta a con lo que entrenó el modelo, este seguramente va a fallar...

```

regressions = tree_reg.predict(xt)
print('Train:')
print(f'MAE: {mean_absolute_error(yt, regressions)}')
print(f'R2: {r2_score(yt, regressions)}')
✓ 0.0s

Train:
MAE: 5.081710928539034
R2: 0.9979236143455456

regressions = tree_reg.predict(xv)
print('Val:')
print(f'MAE: {mean_absolute_error(yv, regressions)}')
print(f'R2: {r2_score(yv, regressions)}')
✓ 0.0s

Val:
MAE: 7.435332649569152
R2: 0.9651049464485886

```

6.7. Redes Elásticas

Finalmente pondremos a prueba modelo un modelo más, el ElasticNet, el cual es un modelo de Regresión Lineal que combina las propiedades de Lasso y RidgeRegression, es decir, combina las funciones de regularización L1 y L2 de sus respectivos modelos, las cuales también observaremos a continuación. El fin de utilizar también estos métodos para asegurar que nuestros datos sí son funcionales para lograr una precisión en nuestras predicciones.

Podemos observar entonces que los resultados obtenido van acorde a lo obtenido en los demás modelos.

```

net
Train:
MAE: 29.738913410262434
R2: 0.8100079708486162
Validate:
MAE: 31.021996668812783
R2: 0.7817395242209058

lasso
Train:
MAE: 29.917594520105666
R2: 0.8139995128627826
Validate:
MAE: 31.13974954207835
R2: 0.786048076894323

```

```
bayes
Train:
MAE: 30.582171433570913
R2: 0.8163165240470222
Validate:
MAE: 31.740856414239765
R2: 0.7885399787297663
```

7. Modelo ganador de Modelación Supervisada

Con base en los resultados de las métricas de los modelos que pusimos a prueba y que mostramos en la sección anterior, vemos conveniente que el modelo que usaremos para dar respuesta a la necesidad del negocio sea el Árbol de Decisión, ya que tanto el MAE, MSE y R2 arrojan buenos resultados en entrenamiento, pero aún más importante, es un modelo más robusto y que podría mantenerse mejor en el tiempo, cosa que con un modelo de Regresión Lineal es más difícil que suceda.

Teniendo esto definido, ahora sí realizamos la monitorización de nuestro modelo para mostrar a nuestro cliente cómo se comporta el modelo a lo largo de cada Q donde tenemos información.

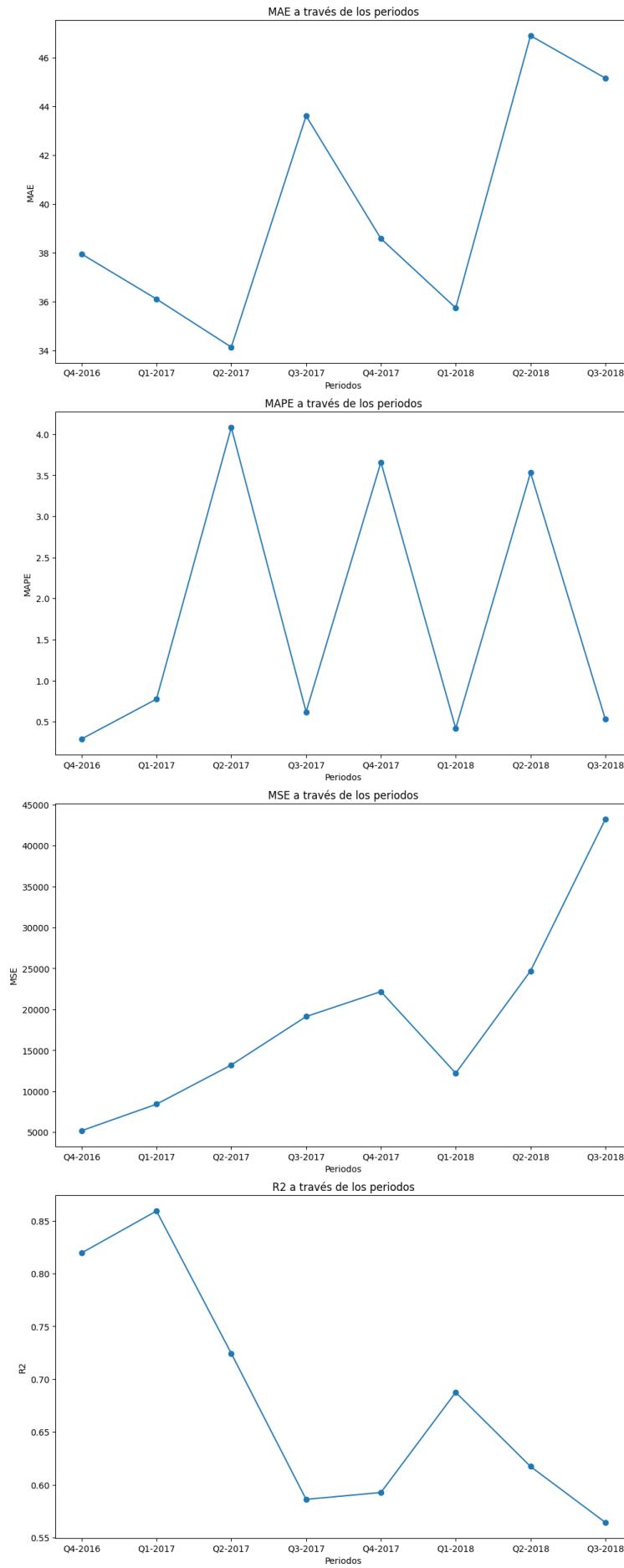
8. Monitorización del modelo

Buscamos conocer cómo se comporta nuestro modelo a través de los Q's en donde se realizaron compras (recordemos que contamos con datos de Q4-2016 al Q3-2018) con el fin de ver si existe estacionalidad que pueda impactar de manera positiva o negativa en la que nuestro modelo se desempeñe.

```
list(periods.keys())
✓ 0.0s
['Q4-2016',
 'Q1-2017',
 'Q2-2017',
 'Q3-2017',
 'Q4-2017',
 'Q1-2018',
 'Q2-2018',
 'Q3-2018']
```

8.1. Visualización gráfica de la monitorización

A continuación mostramos como se comporta nuestro modelo para las siguientes métricas en cada uno de los Q's:



Podemos ver caídas y subidas muy pronunciadas en el caso del MAE, y del MSE, en este último el puntaje se mantiene casi igual del primer al último Q a pesar de lo volátil que es.

En el caso del Mean Absolute Percentage Error, vemos que no es muy grande el incremento en los primeros 4 Q's, pero en los últimos 3, crece muy pronunciadamente.

Finalmente, en el R2 es casi constante la caída a través de los Q's a excepción del segundo y sexto, siendo del tercero al cuarto donde más caída se percibe.

9. Conclusiones de Modelación supervisada

Gracias al análisis y exploración de los datos con los que contamos, pudimos obtener una TAD la cual fuese de gran ayuda para que los modelos que pusimos a prueba arrojaran buenos resultados, pues sin estas transformaciones, los resultados de las métricas eran considerablemente inferiores, esto debido a ciertos defectos en la captura de los datos o que temas en el servicio de la empresa afectaran en los registros con los que contábamos en nuestra variable objetivo. Como expusimos anteriormente, todos los modelos puestos aprueba aseguran muchos mejores resultados que el azar, lo que muestra la calidad de la TAD.

Es por ello, que consideramos que nuestro modelo puede resolver de buena manera la necesidad de nuestro cliente en cuanto a que pueda conocer cuál es el monto de compras de sus clientes para así ellos tener una idea de qué productos pudieran recomendar con base en el precio, trayendo así generación de valor para la empresa Olist.

10. Modelación No Supervisada

Para esta sección aplicaremos los más de 10 métodos de Modelación No Supervisada vistos a lo largo del Módulo 3, para finalmente llegar al Perfilamiento de los clusters finales que obtendremos.

En secciones anteriores aplicamos un análisis de detección de Outliers, KBest e Identificación de clusters, esto nos servirá como base para la aplicación de los métodos de Modelación No Supervisada.

La ruta a seguir es poner en práctica cada uno de los modelos vistos en el curso, haremos un análisis de los resultados obtenidos y llegaremos a la conclusión del método más idóneo para nuestro DataFrame.

10.1. Objetivo de Negocio

Conocer cuáles son los segmentos de clientes que compran en Olist, con el fin de saber qué tipos de productos compran y con ello ser capaces de saber qué tipos de productos recomendar. No solo basándonos meramente en la característica del monto de sus transacciones, sino características adicionales de los clientes, las cuales descubriremos en con estas técnicas de modelación, siendo esto algo añadido a la Necesidad de Negocio que se nos había planteado originalmente.

10.2. Detección de Outliers y Estandarización

Como mencionamos en el método que mejor detectó outliers fue el IsoForest ya que, a diferencia de los otros modelos, este sí elimina valores atípicos pero mantiene suficiente información del DataFrame original. Es por ello que a partir de ahora utilizaremos el DataFrame que nos arroja este modelo, el cual ya con las variables predictoras finales, contará con 48,348 registros.

Una vez teniendo nuestro DataFrame sin outliers, realizaremos la estandarización por medio de StandardScaler de nuestro conjunto de datos para asegurar que todas los campos estén bajo la misma escala. Recordemos que los campos tipo bandera no son considerados en esta transformación pues ya se encuentran entre valores de 0 y 1, en nuestro caso solo la variable pedidos_misma_ciudad tiene este caso.

Creamos además una copia del DataFrame antes de estandarizar, llamada df_orig, el cual nos servirá más adelante en la exploración de los resultados arrojados por el modelo ganador.

10.3. Reducción de Dimensiones

Iniciamos el proceso propiamente de Modelación No Supervisada buscando definir los componentes principales que definirán el DataFrame estandarizado, es decir, reducir la dimensionalidad de este con el fin de aligerar los procesos subsecuentes de visualización gráfica del conjunto de datos y posterior determinación de los clusters que compondrán el modelo.

Métodos basados en componentes

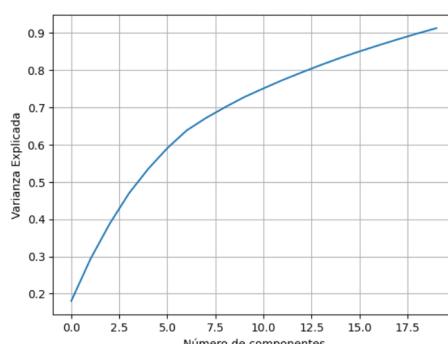
10.3.1. Análisis de Componentes Principales

El método de PCA es, como sabemos el más común cuando se busca reducir la dimensionalidad de un conjunto de datos y hasta los más posible, su eliminación de ruido. Esto nos ayudará a simplificar la información del DataFrame manteniendo la mayor variaiblidad posible.

Lo que buscamos en primera instancia es determinar con cuántos componentes se mantiene la varianza explicada con al menos el 0.9, es decir, que con la cantidad de dimensiones reducidas expliquen al menos el 90 % de la variabilidad del conjunto de datos original.

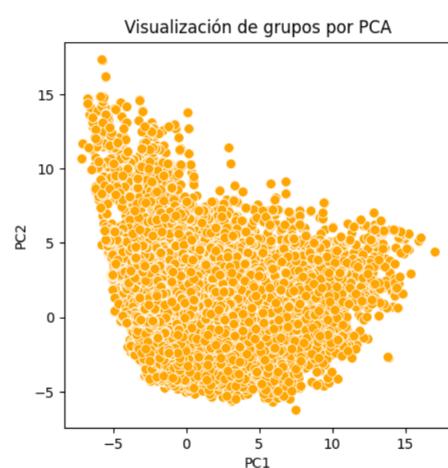
Determinar la cantidad de componentes

Realizamos la siguiente gráfica para ver con cuántos componentes se cumple dicha regla de dedo” que se suele definir para esto proceso. Lo que arroja esta gráfica, como lo podemos ver, es que con 20 componentes se conserva el 90 % de la variabilidad del conjunto original, es decir, de 53 componentes, podremos pasar a solo 20, la cual es una reducción considerable de campos.



Visualización en R2

Con el fin de ver cómo se comporta gráficamente el conjunto de datos bajo este método, haremos el proceso pero solo para 2 componentes, para que con ello podamos ver en una gráfica a nuestro conjunto de datos. Los resultados fueron los siguientes:

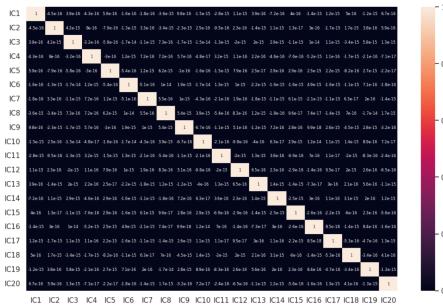


Vemos que, si bien hay algunos puntos separados del resto del conjunto, el retirar outliers evitó que este comportamiento fuera más pronunciado, y por lo tanto, obtenemos en conjunto más compacto.

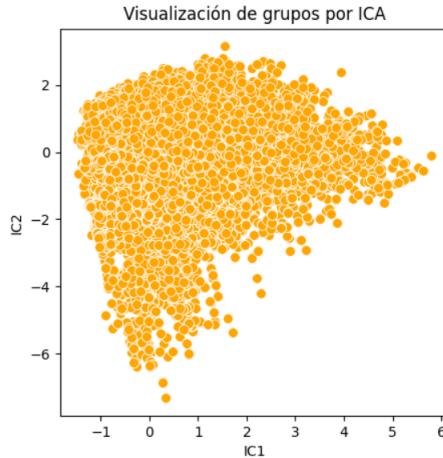
10.3.2. Análisis de Componentes Independientes

Este método se podría ver como lo contrario al PCA, pues en el modelo anterior se buscaba reducir las dimensiones manteniendo la mayor variabilidad posible de los datos, en este proceso se busca también reducir la dimensionalidad, pero bajo la premisa de que las características resultantes sean totalmente independientes entre sí.

Para este caso no tenemos una manera de determinar la cantidad de componentes idónea para reducir el DataFrame, por lo que continuaremos utilizando 20 componentes, con lo cual, al realizar una gráfica de calor sobre el DataFrame resultante vemos que en efecto se logra una independencia casi total entre los campos, pues los coeficientes de correlación para todos los casos son cercanos a cero.



Ahora bien, vemos que reduciendo el conjunto a solo dos componentes se observa una buena agrupación de los puntos de datos, similar a lo obtenido en el método anterior:



10.3.3. Análisis Factorial

Para poder aplicar este procedimiento el cual consiste en reducir la dimensionalidad del conjunto de datos de tal manera que cada componente resultante represente a ciertas variables del conjunto original, es decir, que cada componente resultante tenga rasgos que puedan ser explicables dentro del nuevo conjunto. Este proceso, sin embargo, requiere como condición que, primero, se cumpla la prueba de especificidad de Bartlett y segundo, la prueba de KMO.

Sin embargo estas condiciones no se cumplen pues, en el primer caso, el valor de chi y el p-value no cumplen con la cantidad deseada debe de ser menor a 0.05, mientras que la prueba de KMO no cumple que este sea mayor a 0.6. Por lo tanto no es viable el aplicar este método al conjunto de datos, sin embargo es conveniente comentar en este documento dicho proceso.

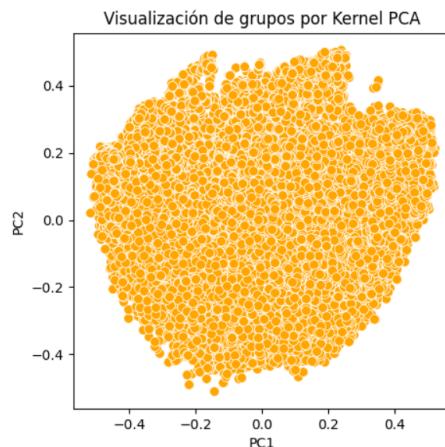
10.3.4. Kernel PCA

A diferencia del PCA, este método nos permite aplicar el PCA pero a un espacio de mucho más alta dimensionalidad. Este método utiliza una función de kernel para proyectar un conjunto de datos en un espacio

de características de mayor dimensión, donde puede ser linealmente separable, ya que, como sabemos el PCA es un método lineal, es decir, se limita a conjuntos linealmente separables.

Para este procedimiento, será aplicada solo una fracción del conjunto estandarizado, esto debido a lo complejo que es el Kernel PCA.

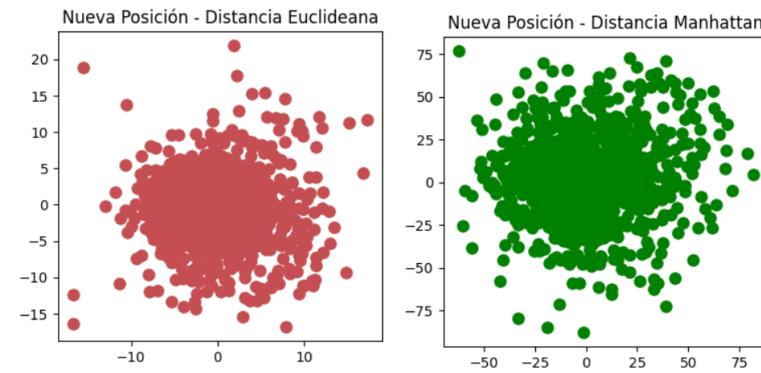
Viendo gráficamente en R2, esta técnica pareciera hacer que los puntos del conjunto estén más cercanos entre sí en relación a lo obtenido en el PCA, por lo que podríamos decir que realiza un buen trabajo en la reducción de dimensionalidad.



Métodos basados en proyección

10.3.5. Escalamiento Multidimensional

El método de MDS, a diferencia de los métodos por componentes, busca ver la relación entre las distancias de puntos del conjunto original, buscando preservar estas similitudes en el conjunto reducido. Si se busca preservar las distancias entre puntos y facilitar la visualización de las relaciones globales de los puntos, este es un método bastante recomendable de utilizar.



La métrica para guiarnos sobre el desempeño del modelo fue la medida de stress, los cuales se obtuvieron los siguientes resultados:

- MDS 1: 3841181.72
- MDS 2: 83080863.75

Por lo tanto la primer forma de MDS, según esta métrica se desempeña mejor.

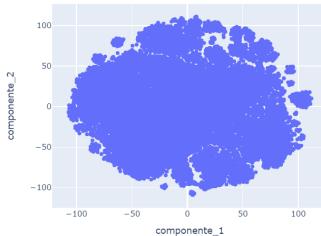
10.3.6. t-SNE

El t-SNE una técnica no lineal de reducción de dimensiones que busca preservar las similitudes entre pares de puntos los cual estén cerca entre sí en el espacio reducido, es decir se centra en las relaciones locales

de puntos, contrario al PCA donde se centra en mantener las grandes distancias entre pares para maximizar la varianza.

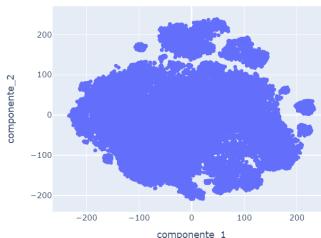
Similar a lo que hicimos en PCA y Kernel PCA, visualizaremos en R2 el conjunto de datos bajo este algoritmo.

Para este caso aplicamos el algoritmo de la forma más básica, es decir, solo indicando que se apliquen 2 componentes, teniendo así el siguiente resultado:



Ahora bien, como sabemos, en t-SNE contamos con los conceptos de perplexidad, el cual se puede explicar como el número de vecinos que considera el algoritmo, es decir a menor perplexidad más enfoque se dará en relaciones locales en la vecindad.

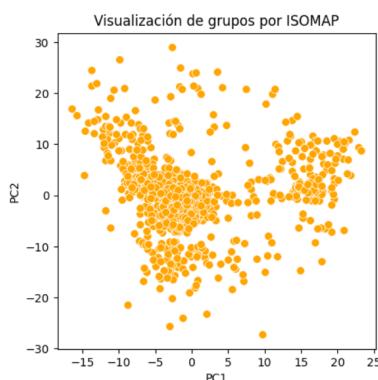
Aplicando este concepto junto con un máximo de iteraciones (perplexity = 50, n_iter = 5000). Con ello, como lo podemos observar, genera más segmentación y se observa más agrupación dentro del universo de datos.



10.3.7. IsoMap

Por último, aplicamos este método de IsoMap el cual es una extensión al MDS en cuanto a que en vez de ver relaciones entre puntos por medio de su distancia euclídea, este método aplica las relaciones de su distancia geodésica, es decir, ver las relaciones en superficies no planas, al igual que el MDS se busca preservar las relaciones globales de los puntos.

Para este caso aplicamos la medida de stress para ver el desempeño del modelo, el cual arrojó el siguiente resultado: 3255891.41



Observando el conjunto de datos en R2 vemos que este método no genera una agrupación tan bien lograda como los otros métodos, pues se observan muchos puntos dispersos y pocas zonas bien agrupadas.

10.4. Determinar Cantidad de Clusters

Con lo obtenido en el análisis de reducción de dimensiones ya sabemos que el conjunto con el que trabajemos a partir de ahora será el que obtuvimos del PCA el cual se compondrá de 20 campos reducidos, esto nos ayudará al procesamiento de los algoritmos siguientes, pero también a que solamente la información más sustanciosa del conjunto de datos se mantenga de ahora en adelante.

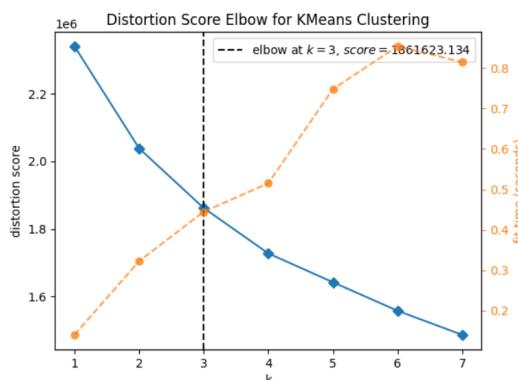
Métodos gráficos para determinar la cantidad de clusters

Los siguientes métodos nos ayudarán en determinar cuántos clusters tendrá el conjunto de datos, es decir, cuántos grupos o segmentos de la población de clientes de Olist clasificará el modelo que apliquemos, dependiendo de las características que comparten los clientes dentro del cluster.

10.4.1. CODO

Este método gráfico nos ayudará a conocer la distorsión de los puntos del conjunto de datos dentro de un mismo cluster. Buscamos el punto donde la cantidad de clusters no represente una disminución significativa del score de distorsión, es decir, que al añadir más clusters, la disminución en la dispersión de los puntos dentro de este ya no sea tan significativa y ya no sea necesario considerar más clusters. Recordemos que buscamos lo menos de clusters posible ya que de haber más, la diferencia de las características del mismo será irrelevante, con lo cual buscamos que los clusters en cuanto a sus características representen a un segmento de los clientes que han comprando en Olist sean lo más distintos posible.

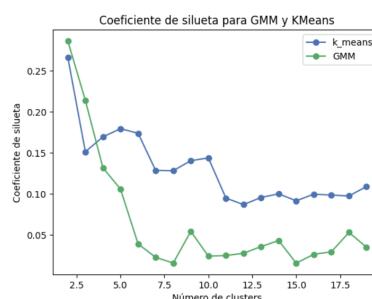
Teniendo esto en cuenta, podemos ver que este método nos sugiere que la cantidad de clusters sea de 3, esta cifra es tan clara que nos lo marca en una linea vertical como lo podemos observar:



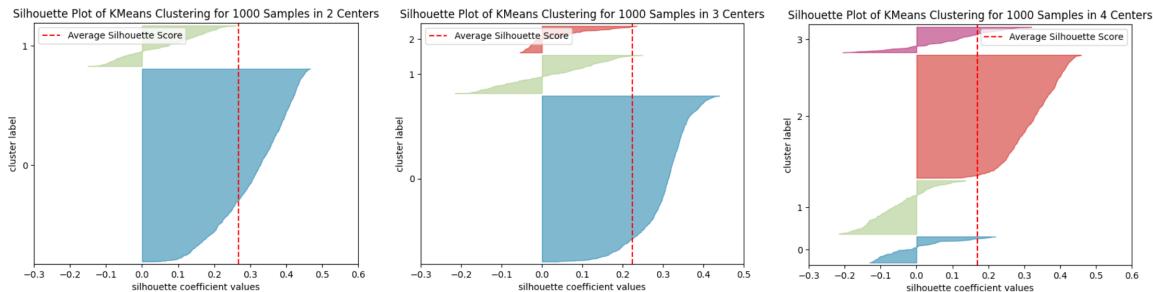
10.4.2. Silueta

Usaremos también el método de la Silueta el cual también muy utilizado en comparación con el método anterior. Consiste en medir qué tan similar es un punto dentro del cluster en relación a los demás clusters, la escala va de +1 a -1, por lo que buscamos que este coeficiente sea lo más positivo posible.

Como podemos ver en la gráfica donde comparamos el modelo de K-Means y GMM vemos que un punto óptimo para la cantidad de cluster es que sea entre 2 y hasta 5 clusters, pero para tener una mejor idea aplicaremos la gráfica de silueta para estos casos.



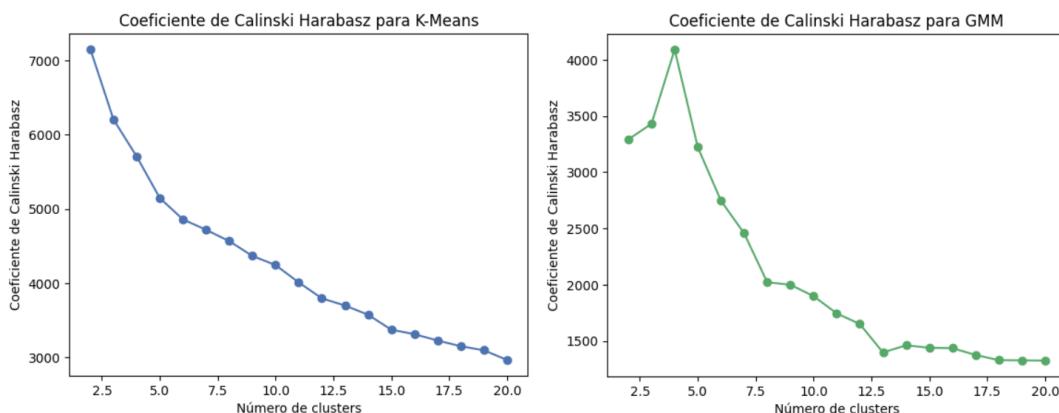
Vemos que existe una preponderancia en la cantidad de puntos dentro de un solo cluster, comportamiento que se mantiene conforme avanza la cantidad de clusters, vemos también que al aumentar los clusters hay un grupo que se mantiene con menos del 10 % de los registros, además de que el score de silueta baja aún más, por lo que descartaremos estos casos, con lo cual vemos más viable el caso donde tengamos 3 clusters, según este método.



10.4.3. Calinski Harabasz

En esta métrica de determinación de clusters se busca obtener la relación entre la dispersión dentro de un grupo y la dispersión entre grupos, es decir, mientras más alto del índice, más densidad y separación entre grupos.

Probaremos este método bajo dos modelos, el de KMeans y GMM, las cuales graficamos:



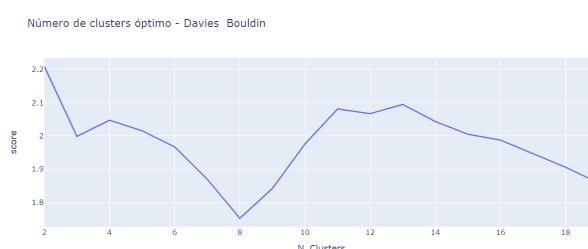
Vemos que la sugerencia de Kmeans es que se apliquen 2 clusters y en GMM sea de 3, que es donde mejore score se obtienen respectivamente.

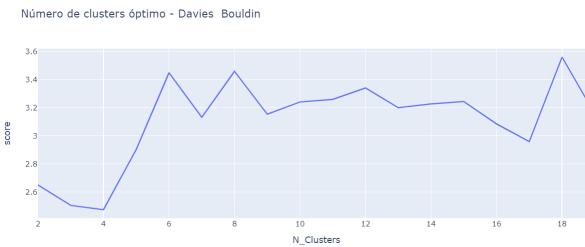
10.4.4. Davies Bouldin

Este método también sirve como una medida de similitud promedio de cada grupo de puntos con su grupo más similar, es decir, la relación entre las distancias dentro de un grupo y las distancias entre grupos. De esta forma, los clusters que estén más alejados y menos dispersos darán una mejor puntuación.

Este método junto con el Calinski Harabasz los aplicaremos posteriormente para medir el desempeño de los modelos que vayamos a aplicar.

También para este método aplicaremos el supuesto del modelo KMeans y GMM, cuyos resultados graficamos y presentamos a continuación:





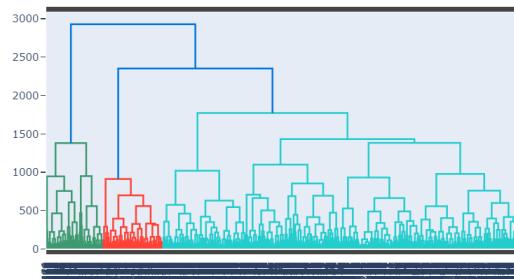
Para este método se busca el score más cercano a cero posible. Vemos que bajo KMeans donde se observa un *codo*.^{es} con 3 clusters, aunque con 8 clusters es donde mayor caída del score se obtiene.

Ahora bien, para GMM, método sugiere que se apliquen 4 clusters, dado que es donde mejor score se obtiene.

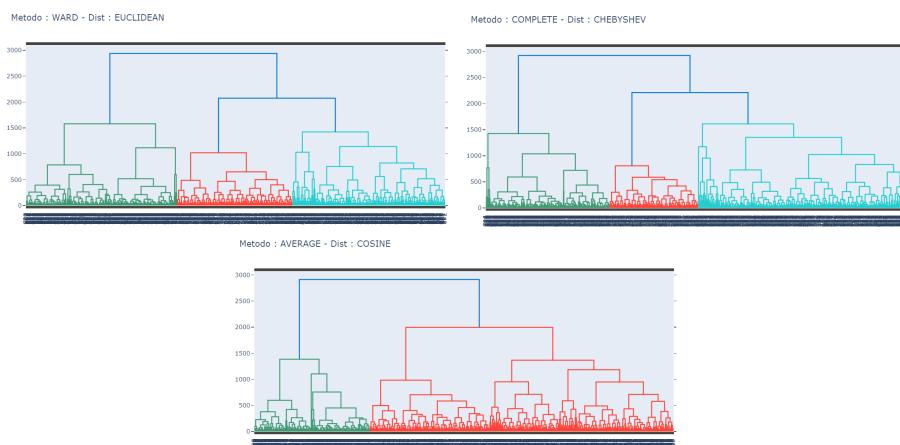
10.4.5. Clustering Jerárquico

Para esta técnica se busca ir agrupando puntos dentro del conjunto de datos hasta llegar clusters que sean similares entre sí. Utilizan técnicas de agrupación midiendo las distancias euclidianas, tipo Chebishev y tipo Cofénica, además de técnicas de agrupamiento y de métodos de vinculación como lo son el simple, completo, ward y por promedio. Cada punto es inicialmente un cluster y conforme se van encontrando similitudes en distancias con los demás puntos se van agrupando hasta llegar a la cantidad óptima de clusters, creando gráficamente una forma de árbol invertido.

A continuación vemos como se ven los resultados si aplicamos distancias euclidianas y método de vinculación simple:



Ahora bien ,como mencionamos tenemos tres tipos de medidas de distancias y cuatro métodos de vinculación, por lo cual aplicaremos las combinaciones de estos, logrando los siguientes resaltados:



Estos son los resultados más destacables, vemos que el método ward con distancia euclíadiana y el método completo con distancia Chebyshev lograron buenas clusterizaciones, la mayoría de las combinaciones aplicadas llegaron a tres clusters, como lo podemos ver en el código, aunque también el caso de método por promedio y distancia cofénica sugirió dos clusters.

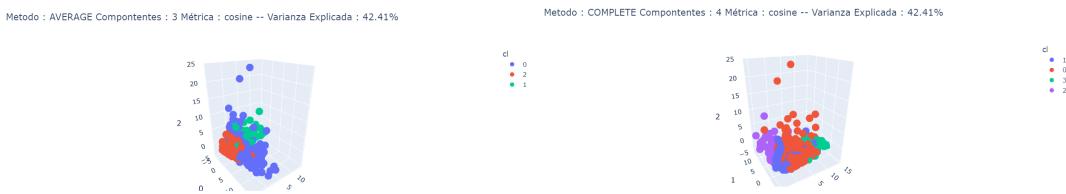
Correlación Cofénica

También dentro del Clustering Jerárquico tenemos una métrica de desempeño que es la Correlación Cofénica. Al aplicar esta métrica para cada caso vemos que estos casos son los que mejor desempeño tienen:

- AVERAGE - EUCLIDEAN: 0.8885869775952888
- SINGLE - EUCLIDEAN: 0.8482951733205424
- AVERAGE - CHEBYSHEV: 0.8356394227716557

Visualización en R3

Finalmente aplicaremos una visualizaciones de estas clusterizaciones en nuestro conjunto de datos bajo las combinaciones que mencionamos, tenemos estos resultados en R3. Mostramos a continuación las segmentaciones más destacadas:



10.5. Modelos de Clusterización

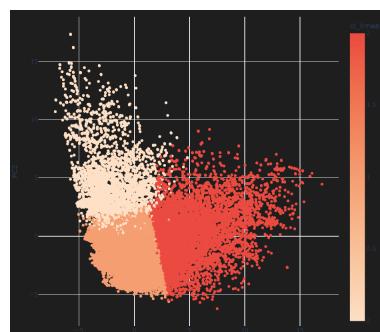
Una vez que aplicamos los métodos de reducción de dimensiones y llegamos a que son 3 los clusters óptimos para nuestro conjunto de datos (debido a que todos los métodos para determinar la cantidad de clusters coinciden en esta cifra). Lo que sigue ahora es aplicar los modelos más utilizados en la industria para el modelado no supervisado, aplicaremos tres, los cuales describiremos y comentaremos los resultados obtenidos.

10.5.1. K-Means

Este método de clusterización funciona mediante la selección de puntos aleatorios dentro del conjunto de datos los cuales son llamados centroides, a estos puntos agruparán en resto de los puntos del conjunto de datos según los que estén más cerca (en valor medio) de los centroides. Este algoritmo requiere que nosotros especifiquemos la cantidad de centroides, pero este dato ya sabemos que debe de ser 3 por el análisis anteriormente realizado.

Una extensión de este modelo es K-Means++ el cual escoge la ubicación de los centroides no de forma aleatoria, sino que el primer centroide se escoge aleatoriamente y los demás se escogen en función de que estén más espaciados entre sí. Esto forma puede ayudar a determinar los centroides óptimos de una forma más rápida.

Para este caso aplicaremos el método más común, el cual arroja este resultado de agrupación:



Existe una buena segmentación entre los clusters, de hecho entre el cluster 1 y 2 se aprecia una división un tanto lineal. Entre el cluster 0 y el 1 vemos un poco de combinación entre los puntos, pero no están marcadas.

Métricas de desempeño Para medir qué también se comportó el modelo para este conjunto de datos aplicaremos dos medidas de desempeño, Davies Bouldin Score y Calinski Harabasz Score, los resultados son los siguientes:

- Score Davies Bouldin : 2.0
- Score Calinski Harabasz : 6203.09

Sabemos que Davies Bouldin debe ser cercano a cero, mientras que Calinski Harabasz debiera ser lo más grande posible, pareciera que el segundo lo cumple, aunque el primero si bien es muy bajo, sí que es mayor a cero.

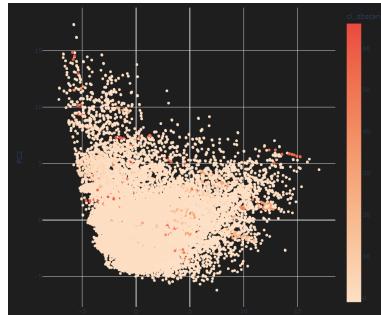
10.5.2. DBSCAN

Este método se basa en agrupar basado en la densidad del conjunto de datos, es decir los grupos que encuentre más segmentados del resto los juntará en un cluster, con ello no es necesario especificar la cantidad de clusters que queremos. Sin embargo, si que se requiere definir qué es “concurrido”, para ello se requiere definir que dentro de una vecindad, cuántos puntos deben de haber y bajo qué radio (ϵ).

Para ello aplicamos NearestNeighbors al conjunto de datos reducido el cual nos dirá cuantos puntos deben de haber dentro de la vecindad, el método de selección es como de un “codo al revés”. Para el caso se la definición del ϵ , teniendo definido la cantidad de vecinos, se grafica las distancias que hay entre las vecindades de estos, y se ve cuál es el punto de infección donde las vecindades sí son de 7 vecinos pero son bajo distancias ya muy grandes, como se ve en la imagen.

Por ello definimos 7 vecinos en un épsilon de 3 como nuestra definición de “concurrido”.

Ahora bien, si recordamos la visualización en R2 que hicimos en reducción de dimensiones y el modelo anterior, podremos ver que nuestro conjunto está bastante compactado, no hay grupos de puntos que se separen mucho del resto, por lo que aún definiendo lo anterior no es de esperarse muy buenos resultados en el caso de nuestro conjunto de datos, como lo vemos en los siguientes resultados:

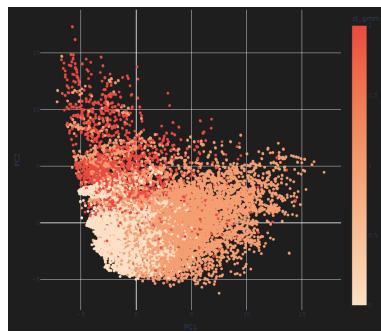


Vemos que casi todos los puntos están en un solo cluster pues están muy cerca entre sí. Es por ello que este modelo no es el conveniente para nuestro conjunto, incluso podemos ver como las métricas no muestra un buen desempeño:

- Score Davies Bouldin : 1.51
- Score Calinski Harabasz : 121.54

10.5.3. Gaussian Mixture Model

El tercer modelo que aplicaremos es el GMM, el cual es un modelo similar a K-Means en cuanto a procedimiento pero en este se aplican funciones Gaussianas para determinar similitudes entre puntos, es decir, este modelo verá qué tan **probable** es que dos puntos se relacionen entre sí, de acuerdo a sus características. Debido a que aplica una función de probabilidad para segmentar cluster es claro que este proceso es más complejo en procesamiento, además que conjunto de puntos que están muy juntos entre sí puede que sea complicado determinar a qué cluster se ubicarán, ya que todos tendrán alguna probabilidad de pertenencia.



Vemos que el modelo se comporta relativamente bien, aunque entre el cluster 1 y 2 hay bastante distorsión, el cluster 0 y 1 también se ve un poco distorsionada.

Las métricas de desempeño se comportan también relativamente bien, aunque no como en el primer modelo

- Score Davies Bouldin : 2.84
- Score Calinski Harabasz : 3430.26

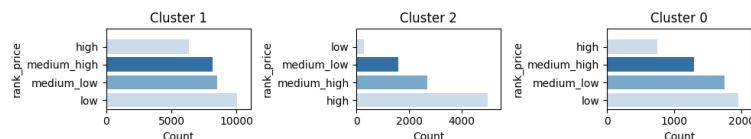
10.6. Modelo Ganador de Modelación No Supervisada

Tomando en cuenta la visualización en R2 de los modelos y los scores de desempeño de los mismos, además de la complejidad de procesamiento, vemos que es claro que el modelo que será utilizado es el K-Means, con 3 clusters y una inicialización aleatoria de centriodess.

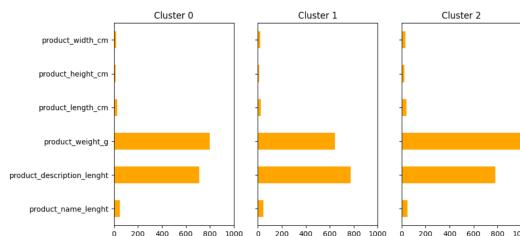
10.7. Análisis de Clusterización

Una vez teniendo definido el modelo y sabiendo qué puntos del conjunto de datos pertenecen a los clusters, lo que hacemos es etiquetar el DataFrame original (el que no está estandarizado), esto para que veamos las características de cada cluster y realizar un Perfilamiento de los mismos.

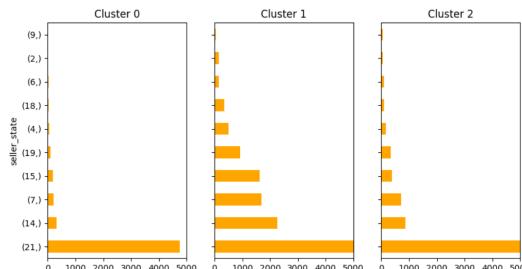
Nos apoyaremos del campo de rango de precio del producto que compró cada cliente, este campo se llama rank_price, donde se etiqueta el rango de precio del producto. A continuación demos que el cluster 2 es donde vemos una gran mayoría de productos con precios altos, contrario a los clusters 0 y 1 donde hay una ligera ventaja de productos con precios bajos.



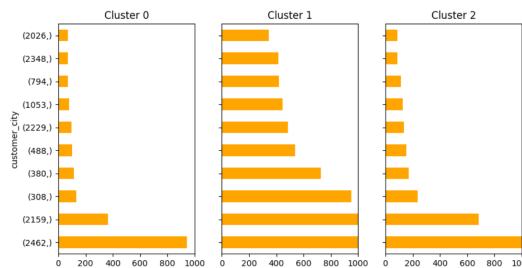
En cuanto a las dimensiones y peso de los productos, en los tres clusters es muy similar, salvo en el cluster 2 donde el peso del producto es en promedio ligeramente mayor al resto de los clusters.



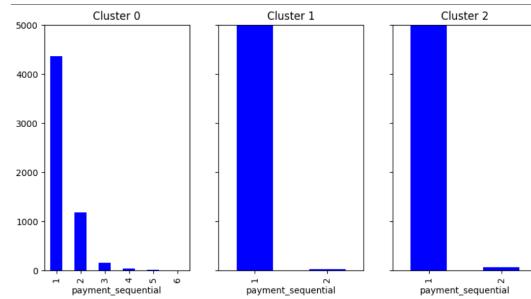
Como mencionábamos en el AEX, la ciudad donde se ubican los vencedores es en su mayoría en el estado de São Paulo, en embargo en el cluster 1 hay un poco de más presencia de ciudades distintas a esta.



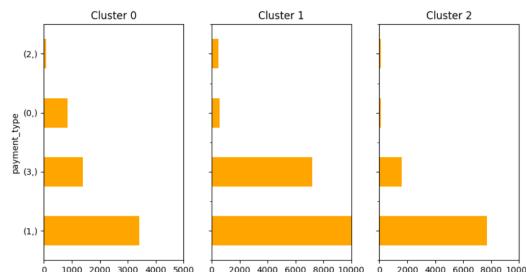
En cuanto a la ciudad donde vienen los clientes se obtienen resultados similares a lo que vimos en el AEX, los clientes viven principalmente en la ciudad de São Paulo:



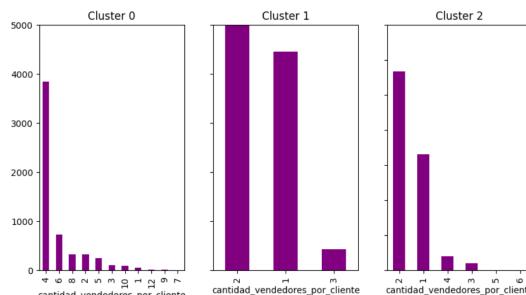
Ahora bien, en cuanto a los diferentes de pagos, el cluster 0 es donde se hacen pagos más allá de una o dos exhibiciones, en los otros dos casi siempre es en una exhibición.



Como sabemos, el cluster 1 es el que conserva más población de los tres, sin embargo en cuanto a los tipos de pago, lo clientes en el cluster 1 se hacen mucho más en efectivo (además de ser con tarjeta de crédito):



En el cluster 1 es donde más variedad de vendedores por cliente hay, es decir, en el clister 0 es donde más variedad de compras se realizan, por lo contrario, el cluster 1 es donde menos variedad de vendedores hay



Los clientes del cluster 0 es donde en promedio se hacen reviews con scores más bajos, aunque en promedio los clientes en ese cluster hacen más compras:

Review score

- Cluster 0: 3.8828451882845187
 - Cluster 1: 4.128685596657989
 - Cluster 2: 4.114533305491752

Cantidad_compras

- Cluster 0: 4.484135285913529
 - Cluster 1: 1.756917115698977
 - Cluster 2: 1.760388390060555

Finalmente, en el cluster 2 es donde más se paga por envío, es decir, en el cluster 2 es donde se hacen compras donde el producto es más costoso de enviar:

- Cluster 0: 14.967275104602512
 - Cluster 1: 15.110700187685413
 - Cluster 2: 25.04575694299436

11. Perfilamiento

Una vez hecho el análisis de los clusters según la información de los campos del DataFrame, lo que hacemos que definir qué sector de los clientes se representarán dentro de cada cluster, en términos simples.

■ Cluster 0 - Clientes de pagos diferidos

Este es el grupo más pequeño, pues aquí los clientes buscan realizar pagos diferidos (a mensualidades), por lo que de haber promociones de este tipo, por ejemplo a meses sin interés, podría ser este grupo el ideal para que compren productos.

El rango de precio de los productos que estos clientes compran, si bien lo que más compran son a precios bajos, está bastante equitativo en este aspecto, por lo que a estos clientes se le podría ofrecer cualquier tipo de productos en cuanto al precio, siempre que sea en diferimiento de pagos. En este cluster no solo hay pagos en efectivo y con tarjeta, sino también por vales.

En resumen, este grupo de clientes es ideal para promociones a mensualidades a cualquier rango de precio pues en promedio se realizan muchas más compras y a cualquier precio.

■ Cluster 1 - Clientes de precios bajos

Este es el grupo más grande y esperamos que la población en este cluster se mantenga así, pues en este cluster se encuentran los clientes que compran a precios bajos, lo que nos indica que sería bueno aplicar promociones a ciertos productos para incentivar las compras de estos clientes. Estos clientes realizan en su gran mayoría compras con tarjeta y en efectivo.

Aquí hay poca variedad de compras de distintos vendedores, otra razón para incentivar la idea de aplicar promociones a productos distintos de los que suelen comprar estos clientes, pues vemos que la mayoría de los clientes buscan comprar barato, y con ello compran con pocos vendedores.

■ Cluster 2 - Clientes de pagos con tarjeta y productos grandes

Este grupo de clientes pueden pagar por productos que no necesariamente se espere que sean entregados de inmediato, pues aquí los costos de envío son altos y compran productos procedentes de la ciudad de São Paulo, los cuales en su mayoría son productos grandes.

Se espera que en este cluster se compran productos como muebles o de tipo industrial, por lo que este tipo de productos, que sean a pagos con tarjeta sería ideal que se apliquen promociones para incentivar la actividad de compras de estos clientes.

12. Estabilidad

Suponiendo que este modelo entra en producción y pasa el tiempo, es necesario tratar de anticiparnos para poder anticiparnos a saber cómo será el comportamiento del mismo en el futuro y así poder saber cuándo será necesario un ajuste al modelo.

Por lo anterior, lo que haremos es partir nuestro DataFrame que aplicamos al modelo suponiendo que estos forman parte de una etapa del tiempo distinta (en este caso haremos cuatro DataFrames suponiendo que es uno por cada Q del año), luego aplicaremos el modelo a estos para ver el resultado de sus métricas y determinar si hay una disminución del rendimiento en el tiempo.

Seguimiento de Métricas

Vemos una relativa consistencia en la mátricas (Davies-Bouldin y Calinski-Harabasz), aunque hay un decaimento en el tiempo 2 y 3:

	DataFrame	Davies-Bouldin Score	Calinski-Harabasz Score
0	df_t1	2.525091	1310.227958
1	df_t2	2.518728	1288.124352
2	df_t3	2.552038	1248.534348
3	df_t4	2.169355	1371.555328

Sin embargo se espera que el porcentaje de diferencia entre un tiempo y otro sea menor a más/menos 5, lo cual entre el tiempo 3 y 4 no se cumple, por lo que ya desde aquí empezamos a ver que sería bueno plantear una recalibraron al modelo anualmente.

Cambio de Centriodes

Otra forma de ver cómo se está comportando el modelo al paso del tiempo es viendo la ubicación de los centriodes, pues si el modelo es consistente se esperaría que la ubicación de estos sea el mismo a lo largo del tiempo o no hayan muchas diferencias. Sin embargo, similar al caso anterior, aquí también se observa un cambio relativamente importante entre las últimas temporalidades

```
Distancia de Cambio en Centroides en el tiempo 2 y 1: [0.53972844 0.34793725 0.25731479]
Distancia de Cambio en Centroides en el tiempo 3 y 2: [7.10505556 0.15543311 6.32538754]
Distancia de Cambio en Centroides en el tiempo 4 y 3: [7.36971423 6.59741182 6.85022535]
```

Cambio de Composición

Por último revisamos si existe un campo en qué tanta población de clientes compone cada cluster a lo largo de los 4 DataFrames y no vemos que exista el caso pues, como lo vimos en secciones anteriores, en general, el porcentaje de población en los tres clusters es de 60 %, 25 % y 15 % respectivamente.

Cluster	T1	T2	T3	T4	cambio_T1_T2	cambio_T2_T3	cambio_T3_T4
0	0.255333	0.284667	0.129167	0.681811	0.029333	-0.155500	0.552644
1	0.586083	0.561833	0.567167	0.200275	-0.024250	0.005333	-0.366891
2	0.158583	0.153500	0.303667	0.117914	-0.005083	0.150167	-0.185753

Análisis Demográfico

Un último análisis que realizamos es ver campo por campo del DataFrame escalado de cada tiempo, cuál es la media de los datos, compararlos entre ellos y ver si hay un cambio significativo.

Si observamos un cambio en la media en los campos si los agrupamos por clusters, por lo tanto, a lo largo del tiempo sí existe cierto campo demográfico al pasar un año de que se ponga en producción este modelo.

En conclusión, vimos que todas las pruebas de estabilidad que hicimos coinciden en que hay cambios en la estructura de los datos y desempeño del modelo al pasar **aproximadamente un año** de entrado en producción.

13. Análisis de la Canasta

Añadimos esta sección para dar una idea a Olist de qué tipo de productos sería probable que el cliente compre dado que este compro alguno de los productos del DataFrame.

Esto será de utilizad a Olist para poder determinar alguna compañía de promociones a ciertos productos dado que se sabe que si el cliente ya compro algún otro producto, entonces quiera también comprar un producto más en oferta.

Lo datos que requerimos para llevar acabo este análisis es el id del cliente, del producto y la fecha de compra de este (recordemos que no contamos con el nombre del producto, solo su id y el tipo que es), a esta tabla la llamaremos market_df. Vemos que estos son los 10 productos más vendidos en Olist, lo cuales en su mayoría son herramientas de jardinería, productos de salud y belleza y muebles para el hogar.

	product_id	product_category_name
1202	53759a2ecddad2bb87a079a1f1519f73	ferramentas_jardim
2094	389d119b48cf043d311335e499d9c6b	ferramentas_jardim
2564	2b4609f8948be18874494203496c318	beleza_saude
3511	b532349fe46b38fb7bb3914c1bdae07	moveis_decoracao
6056	7c1bd920dbdf22470b68bde975dd3ccf	beleza_saude
8877	154e7e31ebfa092203795c972e5804a6	beleza_saude
10079	e53e557d5a159f5aa2c5e995dff244b	informatica_acessorios
12764	368tc6c730842d78016ad823897a372db	ferramentas_jardim
19753	422879e10f46682990de24d770e7f83d	ferramentas_jardim
20028	aca2eb7d00ea1a7b8ebd4e68314663af	moveis_decoracao

Lo que sigue es realizar una tabla pivoté que nos indique el producto que se vendió por cada compra (la compra que realizó el cliente en cierto día), lo cual nos va a permitir obtener el primer resultado importante de este análisis.

Support

Este resultado se obtiene a partir de la tabla anterior, obtenemos la popularidad de un producto determinado por las compras del producto entre el total de compras realizadas, obteniendo que estos productos son los más "populares":

	support	itemsets
0	0.003836	(368c6c730842d78016ad823897a372db)
1	0.000126	(ebee6703e867082f455a058a8bd04092)
2	0.000252	(429e7401fafb76436f15e86498bd7364)
3	0.000126	(4231002e80d2a25aed31d65b4b91f479)
4	0.000157	(685919d3edec26c0ec078b32241025a)

Confidence y Lift

Finalmente aplicamos association_rules para conocer por cada par de productos.

Obtenemos el conficience, el cual es la probabilidad de estas combinaciones de compra.

También obtenemos el lift, que es al aumento del ratio de la venta del producto 2 cuando se vende el producto 1, es decir que si tenemos un lift de 5.59, quiere decir que la probabilidad de comprar el producto de consequents.^aumenta 5.59 veces dado que compramos en el producto de .^antecedents".

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage
0	(389d119b48c3043d311335e499d9c6b)	(422879e10f46682990de24d770e7f83d)	0.004496	0.004999	0.000126	0.027972	5.595813	0.000103
1	(422879e10f46682990de24d770e7f83d)	(389d119b48c3043d311335e499d9c6b)	0.004999	0.004496	0.000126	0.025157	5.595813	0.000103
2	(0d85c435fd60b27ff9e9b0f88927a)	(ee57070a3b24a06dd0e02ef2d757d)	0.000786	0.000503	0.000126	0.160000	318.080000	0.000125
3	(ee57070a3b24a06dd0e02ef2d757d)	(0d85c435fd60b27ff9e9b0f88927a)	0.000503	0.000786	0.000126	0.250000	318.080000	0.000125
4	(f4f67ccaece962d013a4e1d7dc3a61f7)	(4fc3d9a5f4871e8362dfedb02b064)	0.000629	0.001258	0.000220	0.350000	278.320000	0.000219
5	(4fc3d9a5f4871e8362dfedb02b064)	(f4f67ccaece962d013a4e1d7dc3a61f7)	0.001258	0.000629	0.000220	0.175000	278.320000	0.000219
6	(36f60d45225e60c7da4558b070ce4b60)	(e53e557d5a159f5aa2c5e995dff244b)	0.001226	0.002452	0.000472	0.384615	156.844181	0.000469
7	(e53e557d5a159f5aa2c5e995dff244b)	(36f60d45225e60c7da4558b070ce4b60)	0.002452	0.001226	0.000472	0.192308	156.844181	0.000469
8	(36f60d45225e60c7da4558b070ce4b60)	(3f14d740544f37ece8a9e7bc8349797e)	0.001226	0.001352	0.000157	0.128205	94.836017	0.000156
9	(3f14d740544f37ece8a9e7bc8349797e)	(36f60d45225e60c7da4558b070ce4b60)	0.001352	0.001226	0.000157	0.116279	94.836017	0.000156
10	(53759a2ecddad2bb87a079a1f1519f73)	(389d119b48c3043d311335e499d9c6b)	0.004087	0.004496	0.000126	0.030769	6.844110	0.000107
11	(389d119b48c3043d311335e499d9c6b)	(53759a2ecddad2bb87a079a1f1519f73)	0.004496	0.004087	0.000126	0.027972	6.844110	0.000107
12	(060cb19345d90064d1015407193c233d)	(98d61056e0568ba048e5d78038790e77)	0.000283	0.000440	0.000157	0.555556	126.222222	0.000157
13	(98d61056e0568ba048e5d78038790e77)	(060cb19345d90064d1015407193c233d)	0.000440	0.000283	0.000157	0.357143	126.222222	0.000157

Con esto Olist tendrá una forma de saber qué promociones realizar a sus productos y enfocarlos a cierto segmento de la población, explotando la información que obtuvimos de la Modelación No Supervisada, sabiendo que existen 3 tipos de clientes que describimos en el Perfilamiento.

14. Conclusiones de Modelación No Supervisada

El conjunto de datos con el que trabajamos funcionó muy bien bajo la aplicación de todas las etapas del proceso de Modelación No Supervisada, lo que trajo consigo una buena segmentación de la población al aplicar el modelo de KMeans con 3 clusters.

Teniendo estos resultados, la empresa Olist será capaz de explotar esta información para poderlo traducir en generación de valor y mejorar la experiencia de los clientes.

Cabe recalcar que, dado el análisis de Estabilidad que comentamos anteriormente, es necesario que se considere una recalibración periódica de este modelo, idealmente cada año.