Universidad Nacional Autónoma de México

Facultad de Estudios Superiores Acatlán

Diplomado en Ciencia de Datos
Generación 23

Proyecto Módulo 2

Aldo Alejandro Gallegos Ruiz

Índice

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.

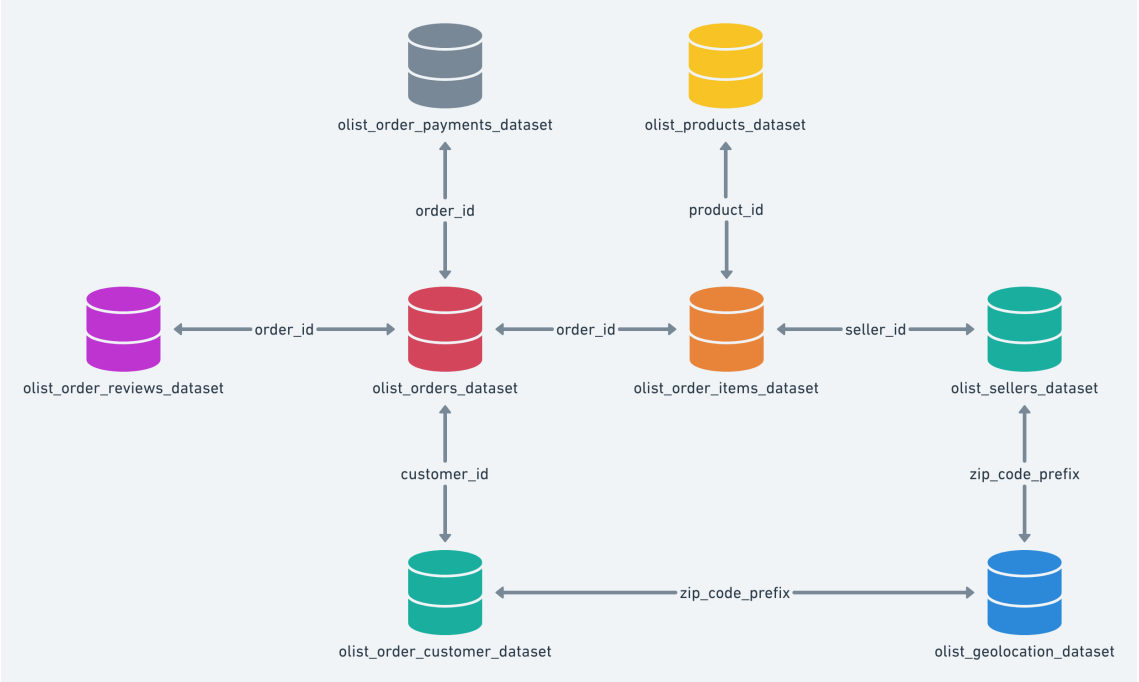
Contamos en total con 9 distintas bases de datos las cuales se interrelacionan entre sí por medio de llaves primarias (o id's). En ellas detalla la información de la compra del cliente como la fecha de realización de la compra, el monto de la compra, tipo de pago, en cuántos pagos se realizó la compra 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 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)


olist_order_payments_dataset
olist_products_dataset
order_id
product_id
olist_order_reviews_dataset
order_id
olist_orders_dataset
order_id
olist_order_items_dataset
seller_id
olist_sellers_dataset
customer_id
zip_code_prefix
olist_order_customer_dataset
zip_code_prefix
olist_geolocation_dataset

2. Primera Parte: 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 ofrezcamos 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. Definición de Target

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. Segunda Parte: 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 excta 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 hace 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 atuda para entender más adelante la contrución de la TAD (Tabla Analítica de Datos) y poseteriormente escoger nuetras variables predictoras:
Variables Continuas

- price: Precio del producto

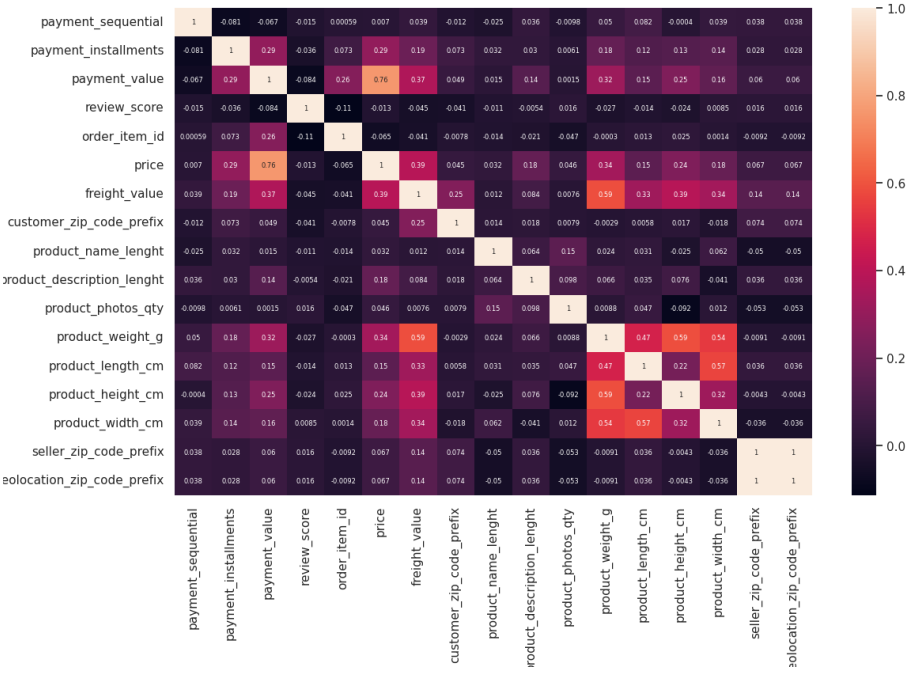
- freight_value: Costo de envío
- product_name_lenght: Largo del nombre del producto
- product_description_lenght: Largo de la descripción del producto
- product_photos_qty: Cantidad de fotos del producto
- product_weight_g: Peso en gramos del producto
- product_length_cm: Lago en centímetros del producto
- 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

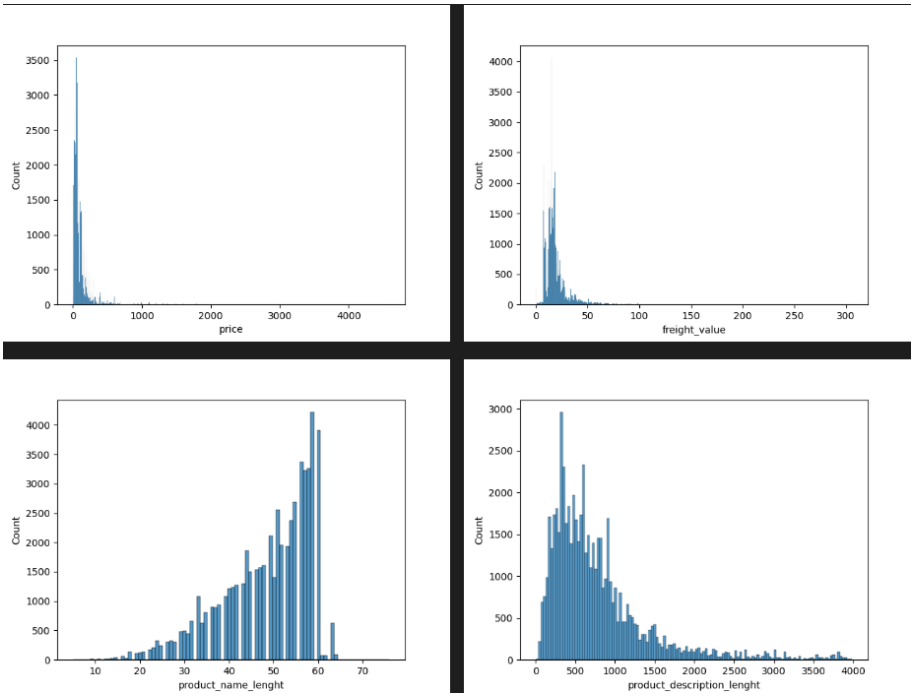
- payment_sequential: Cantidad de abonos para cubrir el pago
- payment_installments: Cantidad de cuotas que se le aplican al pedido
- review_score: Puntaje de 1 a 5 que el cliente dio al servicio recibido por su pedido
- payment_type: Tipo de pago con el que el cliente hizo la compra
- order_status: 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)

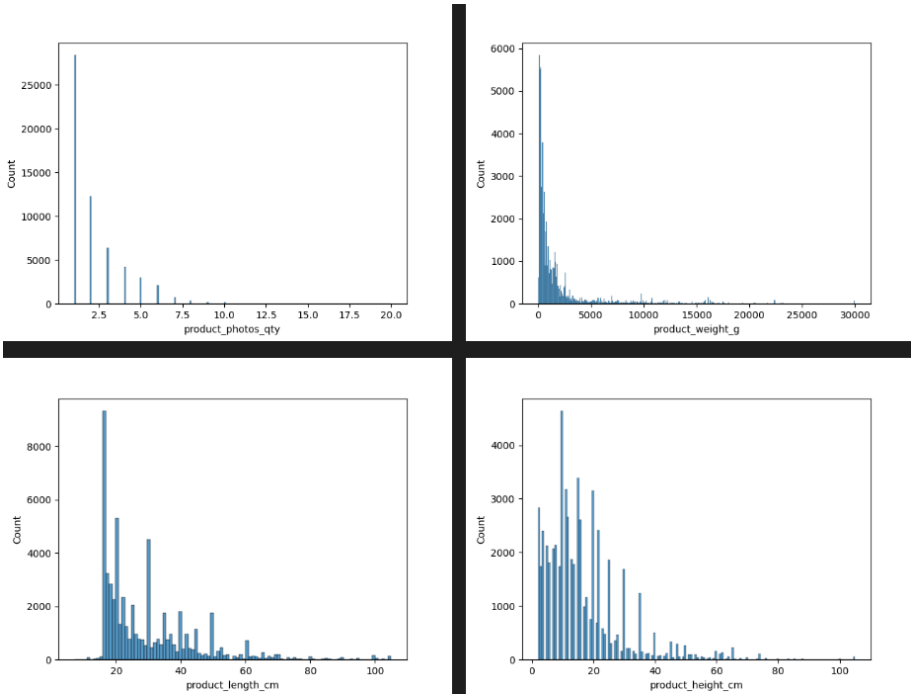
3.7. Visualización gráfica

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.



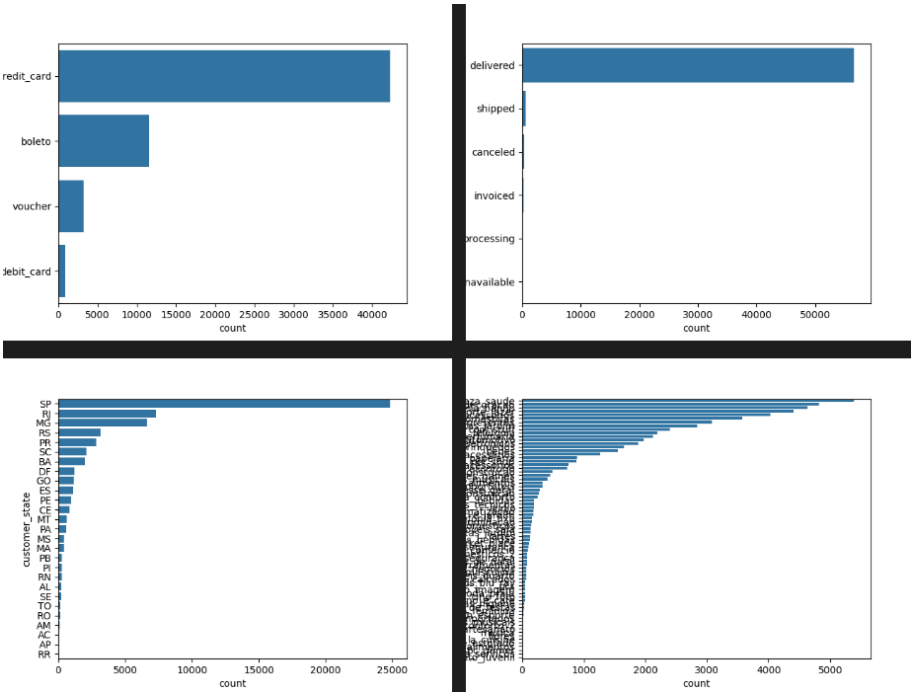
Histogramas

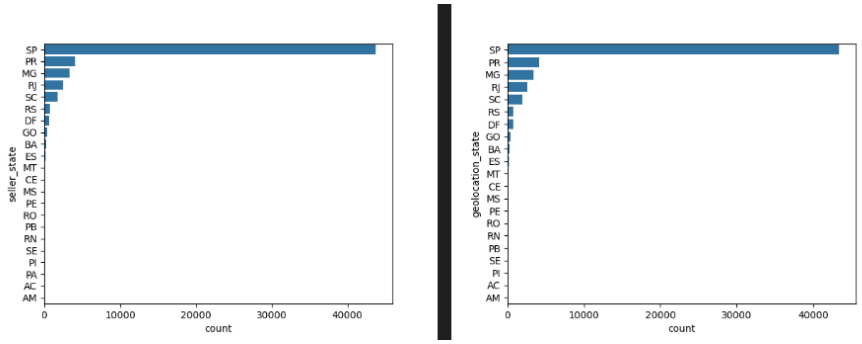




Vale destacar el campo price se concetra entre 0 y 1000 reales (moneda brasileña) y freight está entre 0 y 50, sin embargo para ambos casos existen ciertas compras que se fueron montos muy alejados de estos rangos.

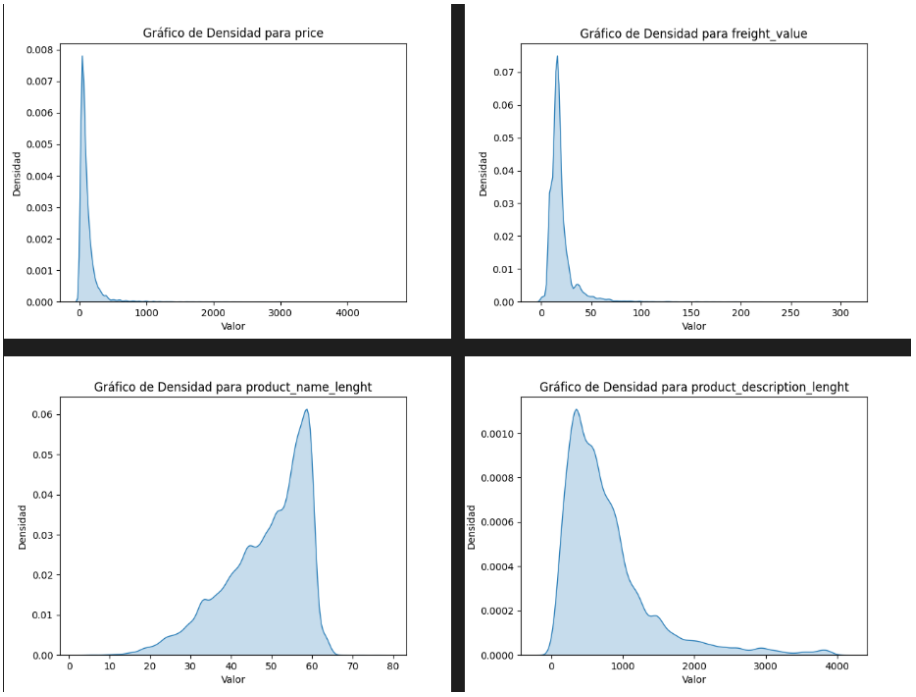
Gráfica de barras

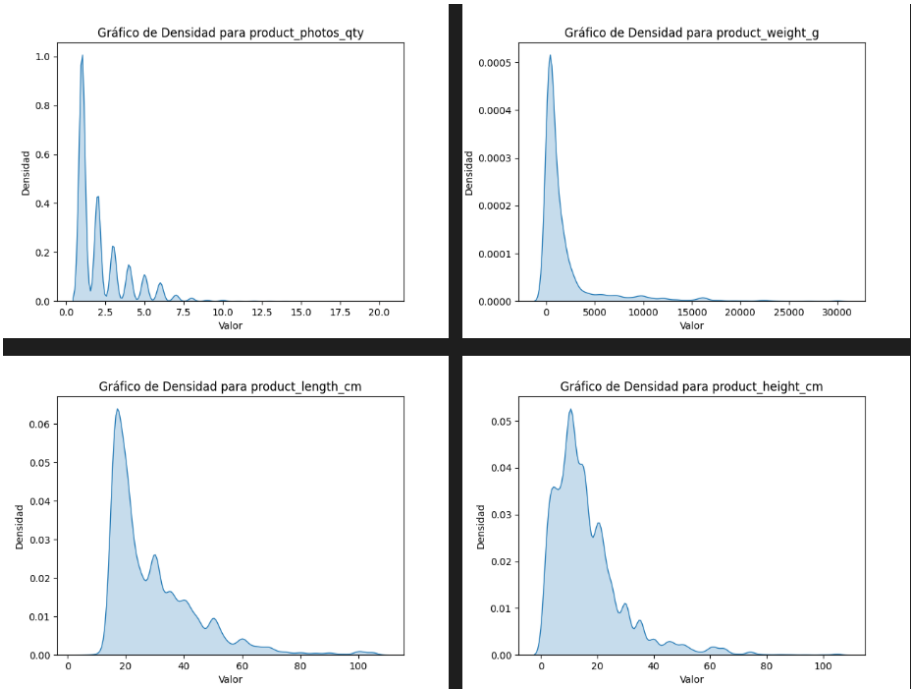




Podemos observar que la gran mayoría de las compras ya están entregados, mientras que el método de pago que hacen los clientes es con tarjeta de crédito. Ahora bien, la ciudad donde se ubica el vendedor y el producto se concentran más en los estados grandes de Brasil (Sao Paola, Rio de Janeiro, Minas 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. Tercera parte: 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 sabes la mediana, el mínimo y máximo de lo que pagaron los cliente por envío
- Queremos sabes 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 que tan grandes y qué tan pesados son los productos que compra el cliente, por lo que obtuvimos su median, 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, mas 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 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 limite 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 con las que contamos van del último Q del 2016 al tercero del 2018.

5. Cuarta parte: 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 ves 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 nof 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

```
iqr(merged_df)
✓ 0.1s
Tamaño original 57961 Tras aplicar IQR: 13768 registros

z_score(merged_df)
✓ 0.2s
Tamaño original: 57961 Al aplicar el método Z-Score: 57961

iso_forest(merged_df,predictors)
✓ 0.5s
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_lenght	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 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 le DataFrame, es seleccionar al as mejores varaibles 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 cao visualizamos que las mejores 10 varaibles 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 = seleccionar_kbest(merged_df[predictors+[tar]],"payment_value",num_variables_deseadas=10)
mejores
```

```
['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']
```

6. Quinta parte: 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 SDGRegressor, 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 sobreajute, 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 decisón 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. Quinta Parte: Modelo ganador

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 tipo, 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.

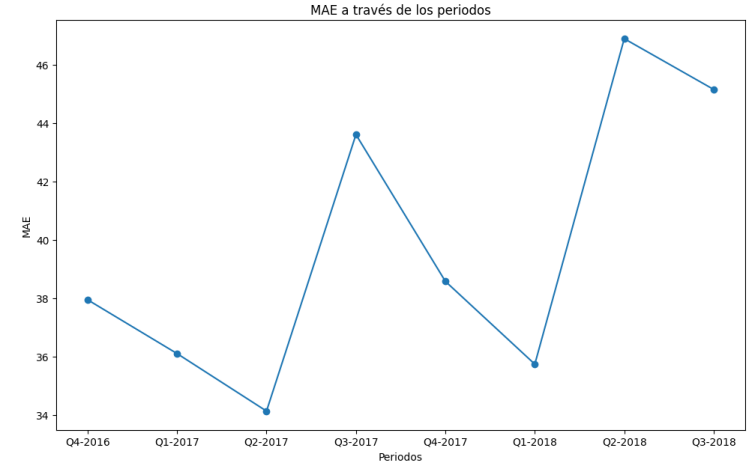
8. Monitorización del modelo

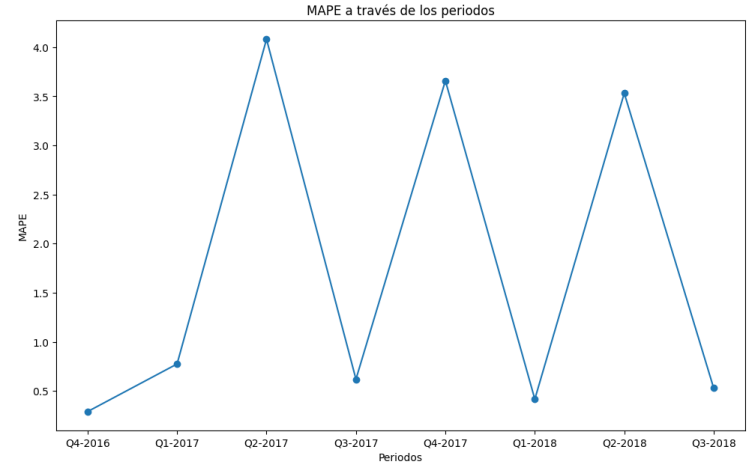
Buscamos conocer cómo se comparta nuestro modelo a través de los Q's en donde se realizaron compras (recordemos que contamos con datos de Q4-2016 al Q3-2018) con el fin de ver si existe estacionalidad que pueda impactar de manera positiva o negativa en la que nuestro modelo se desempeñe.

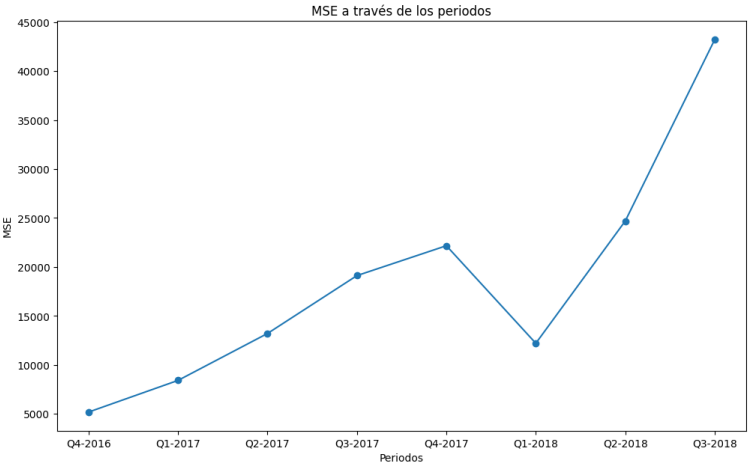
```
list(periods.keys())
✓ 0.0s
['Q4-2016',
 'Q1-2017',
 'Q2-2017',
 'Q3-2017',
 'Q4-2017',
 'Q1-2018',
 'Q2-2018',
 'Q3-2018']
```

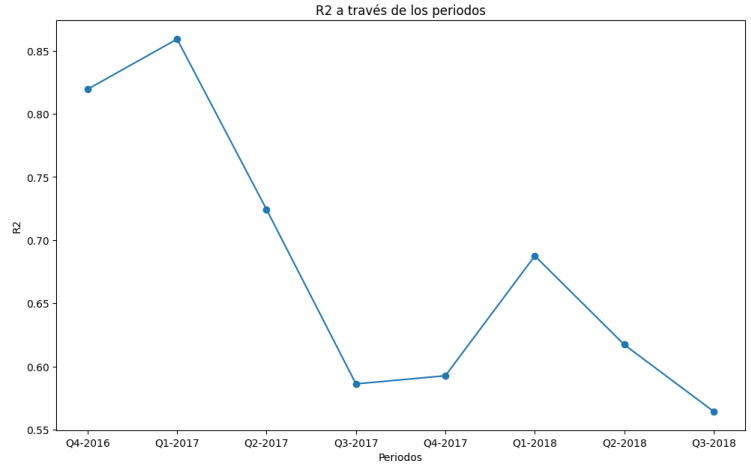
8.1. Visualización gráfica de la monitorización

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


MAE a través de los periodos
MAE
Periodos
Q4-2016
Q1-2017
Q2-2017
Q3-2017
Q4-2017
Q1-2018
Q2-2018
Q3-2018


MAPE a través de los periodos
MAPE
Periodos
Q4-2016
Q1-2017
Q2-2017
Q3-2017
Q4-2017
Q1-2018
Q2-2018
Q3-2018


MSE a través de los periodos
MSE
Periodos
Q4-2016
Q1-2017
Q2-2017
Q3-2017
Q4-2017
Q1-2018
Q2-2018
Q3-2018


R2 a través de los periodos
R2
Periodos
Q4-2016
Q1-2017
Q2-2017
Q3-2017
Q4-2017
Q1-2018
Q2-2018
Q3-2018

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

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, a pesar de que el conjunto de datos presentaran ciertos defectos en la captura de los datos o que temas en del 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.