Análisis y Ciencia de Datos en el ámbito del Retail

Enrique Rocho Simon

Inteligencia de Negocio y Big Data Analytics
-Ciencia de Datos

Tutor/a de TF
José Luis Gomez García
Profesor/a responsable de la asignatura
Atanasi Daradoumis, Josep Curto

16/01/2024

Universitat Oberta de Catalunya

Ficha del Trabajo Final

Título del trabajo:	Análisis y Ciencia de Datos en el ámbito del Retail
Nombre del autor/a:	Enrique Rocho Simon
Nombre del Tutor/a de TF:	José Luis Gomez García
Nombre del/de la PRA:	Atanasi Daradoumis, Josep Curto
Fecha de entrega:	01/2024
Titulación o programa:	Inteligencia de Negocio y Big Data Analytics
Área del Trabajo Final:	Ciencia de Datos
Idioma del trabajo:	Castellano
Palabras clave	Ciencia de datos, Supply Chain, Ventas

Resumen del Trabajo

Estudio analítico, de ciencia y minería de datos aplicado a datos obtenidos en la comercialización de productos en diferentes establecimientos (afiliados) con la finalidad de monitorizar, analizar y predecir datos de ventas o rotura de stock.

Este estudio puede permitiría optimizar las decisiones a tomar por parte de departamentos como ventas, logística, compras, entre otros, posibilitando mejorar los resultados de la compañía.

La metodología utilizada ha sido **CRISP-DM** (Cross-Industry Standard Process for Data Mining), siguiendo estos pasos: comprensión de negocio, comprensión de datos, preparación de datos, modelado, evaluación y despliegue.

Inicialmente se han cargado los datos en MS SQL Server, donde se ha realizado un primer estudio sobre los datos, después se ha exportado la información a Jupyter Notebooks para realizar indagaciones más profundas de los datos, limpieza y adecuación para después poder aplicar diferentes algoritmos de predicción.

Para la predicción sobre ventas, se ha optado por un modelo **Random Forest**

después de realizar una selección de features con más peso sobre la variable objetivo Total_Sales, obteniendo un MSE 32.88 y R2 76%. Se concluye que, a un mayor número de variables, la precisión de las predicciones decrece, pero reducir esas variables limita la aplicación en producción del modelo.

Para la predicción sobre rotura de stock se han detectado problemas por la alta frecuencia del valor 0 con respecto al resto de valores por encima de 0. Esto ha llevado al uso de estrategias de resampling para intentar compensar dicho desequilibrio. A pesar de esto, el resultado ha sido condicionado por esta tendencia, obteniendo con **Decission Tree Classifier** una precisión general de 56%. Se llega a la conclusión de que es necesario establecer otras estrategias para intentar limitar el peso del valor 0 y así poder mejorar las predicciones.

Abstract

Analytical, science and mining data study applied to information obtained in the commercialization of products in different affiliated shops, with the purpose of monitoring, analysing and predicting sales and out of stock results.

This study could optimize decision making by company divisions such as sales, logistics, operations, purchasing, allowing business results improvements.

The methodology employed has been **CRISP-DM** (Cross-Industry Standard Process for Data Mining) following these steps: business underdstanding, data underdstanding, data preparation, modelling, evaluation and deployment.

Firstly, data has been uploaded to MS SQL Server, where a first analysis has been carried out on the data. After that, the information has been exported to Jupyter Notebooks, to conduct further analysis, clean, and transform data to apply prediction algorithms.

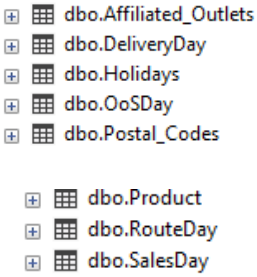
For the sales predictions, a **Random Forest** model has been chosen after a selection of the features with a higher impact on the objective feature Total_Sales, obtaining a MSE 32.88 and R2 76%. It could be observed, that

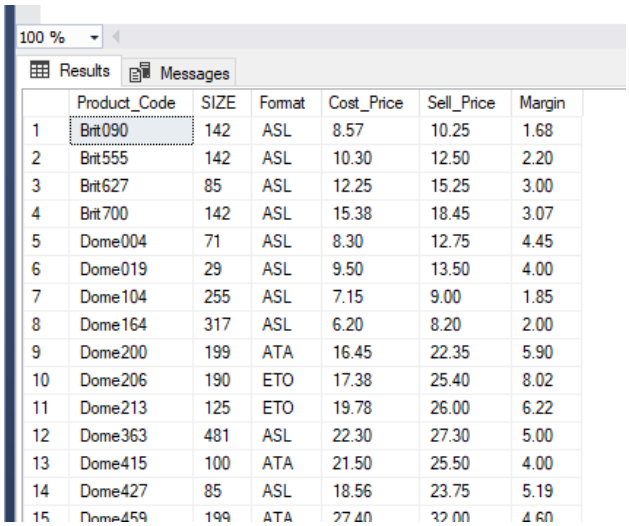
with a higher feature selection accuracy would fall, however reducing these variables limits the production application of the model.

For the out of stock predictions, there were issues detected in regards of the high frequency for value 0 compared to the values above 0. This led to the use of resampling strategies to compensate for the imbalance. Despite this, the result was conditioned by this trend, obtaining a general precision of 56% for a **Decission Tree Classifier** model. It is concluded that further strategies are needed to limit the value 0 prevalence when training the models so its accuracy can be improved.

Index

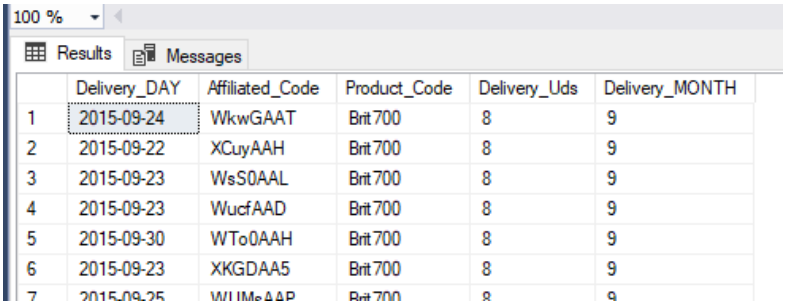
Lista de Figuras





100 % — Results — Messages

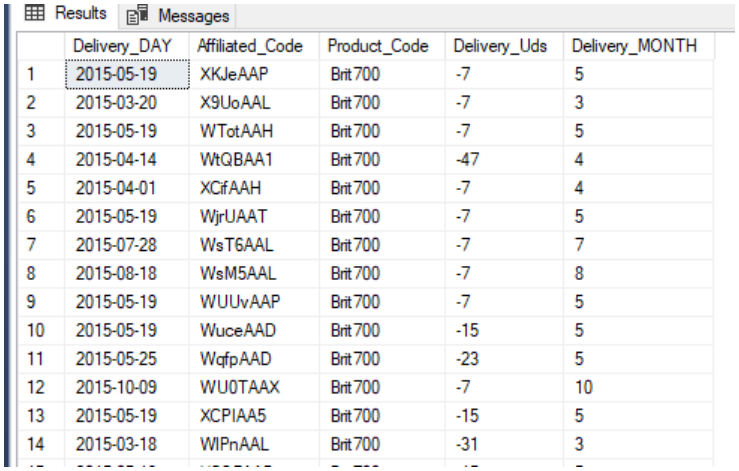
	Product_Code	SIZE	Format	Cost_Price	Sell_Price	Margin
1	Brit090	142	ASL	8.57	10.25	1.68
2	Brit555	142	ASL	10.30	12.50	2.20
3	Brit627	85	ASL	12.25	15.25	3.00
4	Brit700	142	ASL	15.38	18.45	3.07
5	Dome004	71	ASL	8.30	12.75	4.45
6	Dome019	29	ASL	9.50	13.50	4.00
7	Dome104	255	ASL	7.15	9.00	1.85
8	Dome164	317	ASL	6.20	8.20	2.00
9	Dome200	199	ATA	16.45	22.35	5.90
10	Dome206	190	ETO	17.38	25.40	8.02
11	Dome213	125	ETO	19.78	26.00	6.22
12	Dome363	481	ASL	22.30	27.30	5.00
13	Dome415	100	ATA	21.50	25.50	4.00
14	Dome427	85	ASL	18.56	23.75	5.19
15	Dome459	199	ATA	27.40	32.00	4.60



100 % — Results — Messages

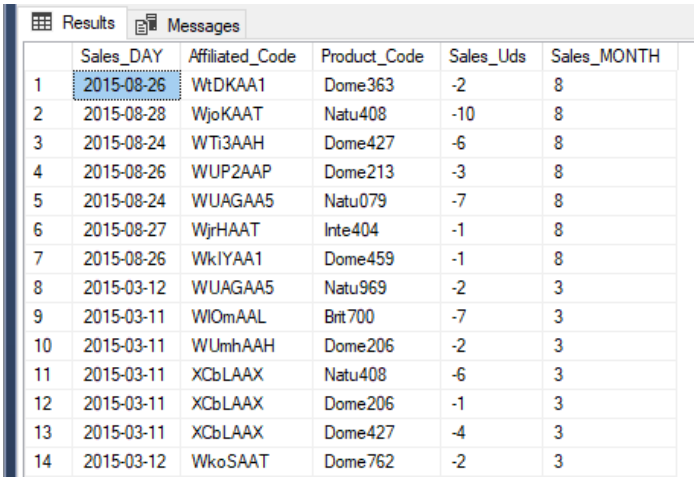
	Delivery_DAY	Affiliated_Code	Product_Code	Delivery_Uds	Delivery_MONTH
1	2015-09-24	WkwGAAT	Brit700	8	9
2	2015-09-22	XCuyAAH	Brit700	8	9
3	2015-09-23	WsS0AAL	Brit700	8	9
4	2015-09-23	WucfAAD	Brit700	8	9
5	2015-09-30	WTo0AAH	Brit700	8	9
6	2015-09-23	XKGDAA5	Brit700	8	9
7	2015-09-25	WUMeAAP	Brit700	8	9

Figuras 4 y 5- Unidades negativas



Results Messages

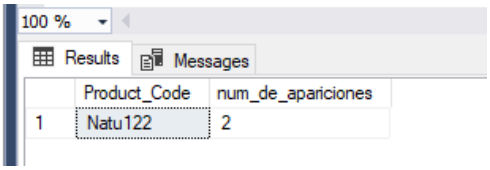
	Delivery_DAY	Affiliated_Code	Product_Code	Delivery_Uds	Delivery_MONTH
1	2015-05-19	XKJeAAP	Brit700	-7	5
2	2015-03-20	X9UoAAL	Brit700	-7	3
3	2015-05-19	WTotAAH	Brit700	-7	5
4	2015-04-14	WtQBAA1	Brit700	-47	4
5	2015-04-01	XCifAAH	Brit700	-7	4
6	2015-05-19	WjrUAAT	Brit700	-7	5
7	2015-07-28	WsT6AAL	Brit700	-7	7
8	2015-08-18	WsM5AAL	Brit700	-7	8
9	2015-05-19	WUUvAAP	Brit700	-7	5
10	2015-05-19	WuceAAD	Brit700	-15	5
11	2015-05-25	WqfpAAD	Brit700	-23	5
12	2015-10-09	WU0TAAX	Brit700	-7	10
13	2015-05-19	XCPIAA5	Brit700	-15	5
14	2015-03-18	WlPnAAL	Brit700	-31	3



Results Messages

	Sales_DAY	Affiliated_Code	Product_Code	Sales_Uds	Sales_MONTH
1	2015-08-26	WtDKAA1	Dome363	-2	8
2	2015-08-28	WjoKAAT	Natu408	-10	8
3	2015-08-24	WTi3AAH	Dome427	-6	8
4	2015-08-26	WUP2AAP	Dome213	-3	8
5	2015-08-24	WUAGAA5	Natu079	-7	8
6	2015-08-27	WjrHAAT	Inte404	-1	8
7	2015-08-26	WkIYAA1	Dome459	-1	8
8	2015-03-12	WUAGAA5	Natu969	-2	3
9	2015-03-11	WlOmAAL	Brit700	-7	3
10	2015-03-11	WUmhAAH	Dome206	-2	3
11	2015-03-11	XCbLAAX	Natu408	-6	3
12	2015-03-11	XCbLAAX	Dome206	-1	3
13	2015-03-11	XCbLAAX	Dome427	-4	3
14	2015-03-12	WkoSAAT	Dome762	-2	3

Figura 6- Referencia duplicada Product

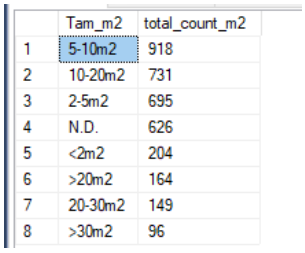


100 %

Results Messages

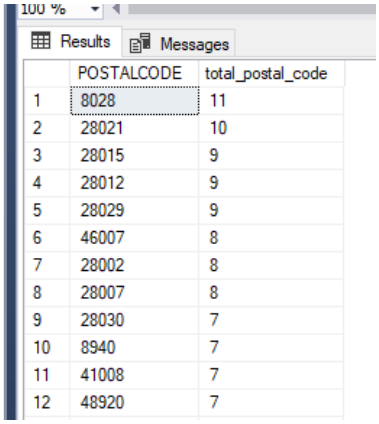
	Product_Code	num_de_apariciones
1	Natu122	2

Figura 7- Count Tam_m2 de Affiliated Outlets



	Tam_m2	total_count_m2
1	5-10m2	918
2	10-20m2	731
3	2-5m2	695
4	N.D.	626
5	<2m2	204
6	>20m2	164
7	20-30m2	149
8	>30m2	96

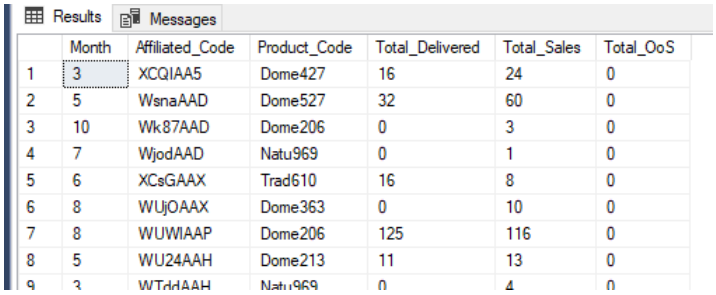
Figura 8- Count Postal Code



100 %
Results Messages

	POSTALCODE	total_postal_code
1	8028	11
2	28021	10
3	28015	9
4	28012	9
5	28029	9
6	46007	8
7	28002	8
8	28007	8
9	28030	7
10	8940	7
11	41008	7
12	48920	7

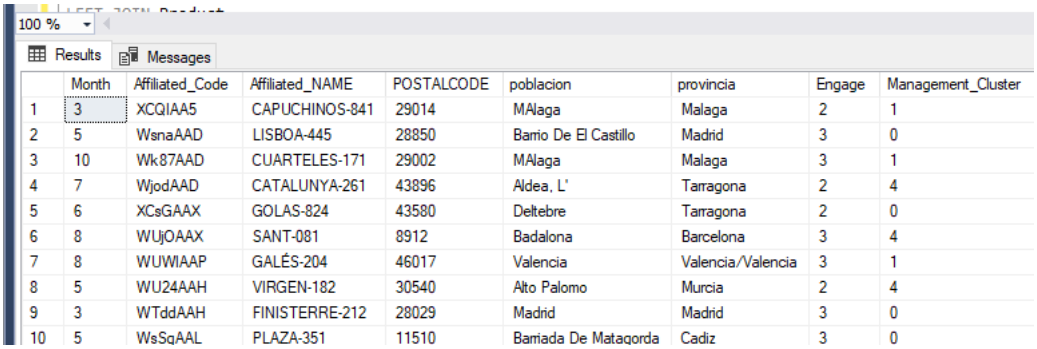
Figura 9 - Vista fact_tables



Results Messages

	Month	Affiliated_Code	Product_Code	Total_Delivered	Total_Sales	Total_OoS
1	3	XCQIAA5	Dome427	16	24	0
2	5	WsnaAAD	Dome527	32	60	0
3	10	Wk87AAD	Dome206	0	3	0
4	7	WjodAAD	Natu969	0	1	0
5	6	XCsGAAX	Trad610	16	8	0
6	8	WUjOAAX	Dome363	0	10	0
7	8	WUWIAAP	Dome206	125	116	0
8	5	WU24AAH	Dome213	11	13	0
9	3	WTddAAH	Natu969	0	4	0

Figura 10- Vista all_tables:



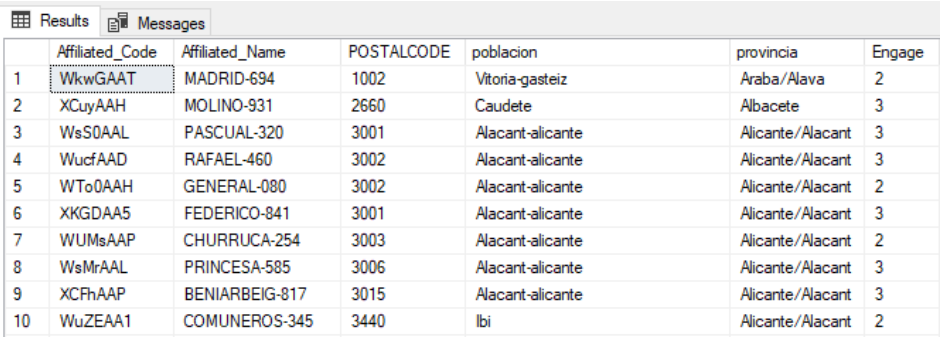
100 %

Results | Messages

	Month	Affiliated_Code	Affiliated_NAME	POSTALCODE	poblacion	provincia	Engage	Management_Cluster
1	3	XCQIAA5	CAPUCHINOS-841	29014	MAlaga	Malaga	2	1
2	5	WsnaAAD	LISBOA-445	28850	Barrio De El Castillo	Madrid	3	0
3	10	Wk87AAD	CUARTELES-171	29002	MAlaga	Malaga	3	1
4	7	WjodAAD	CATALUNYA-261	43896	Aldea, L'	Tarragona	2	4
5	6	XCsGAAX	GOLAS-824	43580	Deltebre	Tarragona	2	0
6	8	WUjOAAX	SANT-081	8912	Badalona	Barcelona	3	4
7	8	WUWIAAP	GALÉS-204	46017	Valencia	Valencia/Valencia	3	1
8	5	WU24AAH	VIRGEN-182	30540	Alto Palomo	Murcia	2	4
9	3	WTddAAH	FINISTERRE-212	28029	Madrid	Madrid	3	0
10	5	WsSqAAL	PLAZA-351	11510	Barriada De Matagorda	Cadiz	3	0

Location	Tam_m2	Product_Code	SIZE	Format	Cost_price	Sell_price	Margin	Total_Delivered	Total_Sales	Total_OoS
ANY	2-5m2	Dome427	85	ASL	18.56	23.75	5.19	16	24	0
VILLAGE	2-5m2	Dome527	85	ASL	40.57	46.90	6.33	32	60	0
CITY	2-5m2	Dome206	190	ETO	17.38	25.40	8.02	0	3	0
VILLAGE	10-20m2	Natu969	85	ASL	62.30	75.00	12.70	0	1	0
VILLAGE	>20m2	Trad610	142	ASL	5.90	7.00	1.10	16	8	0
BORDER	2-5m2	Dome363	481	ASL	22.30	27.30	5.00	0	10	0
ANY	5-10m2	Dome206	190	ETO	17.38	25.40	8.02	125	116	0
ESCAPE	>20m2	Dome213	125	ETO	19.78	26.00	6.22	11	13	0
CITY	5-10m2	Natu969	85	ASL	62.30	75.00	12.70	0	4	0

Figura 11- Vista Delivery_Route:



Results | Messages

	Affiliated_Code	Affiliated_Name	POSTALCODE	poblacion	provincia	Engage
1	WkwGAAT	MADRID-694	1002	Vitoria-gasteiz	Araba/Alava	2
2	XCuyAAH	MOLINO-931	2660	Caudete	Albacete	3
3	WsS0AAL	PASCUAL-320	3001	Alacant-alicante	Alicante/Alacant	3
4	WucfAAD	RAFAEL-460	3002	Alacant-alicante	Alicante/Alacant	3
5	WTo0AAH	GENERAL-080	3002	Alacant-alicante	Alicante/Alacant	2
6	XKGDAA5	FEDERICO-841	3001	Alacant-alicante	Alicante/Alacant	3
7	WUMsAAP	CHURRUCA-254	3003	Alacant-alicante	Alicante/Alacant	2
8	WsMrAAL	PRINCESA-585	3006	Alacant-alicante	Alicante/Alacant	3
9	XCFhAAP	BENIARBEIG-817	3015	Alacant-alicante	Alicante/Alacant	3
10	WuZEAA1	COMUNEROS-345	3440	Ibi	Alicante/Alacant	2

Management_Cluster	Location	Tam_m2	Delivery_DAY	Route_DAY	Delivery_out_of_Route	Festivo
0	BORDER	5-10m2	2015-09-24	2015-09-24	No	Laborable
3	VILLAGE	5-10m2	2015-09-22	2015-09-22	No	Laborable
4	CITY	5-10m2	2015-09-23	2015-09-23	No	Laborable
3	CITY	5-10m2	2015-09-23	2015-09-23	No	Laborable
3	LEISURE	10-20m2	2015-09-30	2015-09-30	No	Laborable
3	CITY	20-30m2	2015-09-23	2015-09-23	No	Laborable
1	CITY	5-10m2	2015-09-25	2015-09-25	No	Laborable
1	BORDER	20-30m2	2015-09-25	2015-09-25	No	Laborable
1	ANY	>30m2	2015-09-30	2015-09-30	No	Laborable
4	ANY	>20m2	2015-09-25	2015-09-25	No	Laborable

Figura 12- Vista Sales_Rotura:

	Sales_DAY	Affiliated_Code	POSTALCODE	poblacion	provincia
1	2015-08-26	WjxnAAD	14011	Caballera, De La (santa Maria De Trassierra)	Cordoba
2	2015-08-26	WjkSAAT	24010	Leon	Leon
3	2015-08-28	WU41AAH	31610	Atarrabia	Navarra
4	2015-08-27	XCvxAAH	30620	Fortuna	Murcia
5	2015-08-27	XCcjAAH	41510	Bencarron	Sevilla
6	2015-08-27	WTd8AAH	28038	Madrid	Madrid
7	2015-08-27	WsSHAA1	14011	Caballera, De La (santa Maria De Trassierra)	Cordoba
8	2015-08-24	WktIAAT	46200	Paiporta	Valencia/Valencia
9	2015-08-24	WU41AAH	31610	Atarrabia	Navarra
10	2015-08-24	XCOsAAP	30500	Molina De Segura	Murcia

Product_Code	Sales_Uds	OoS_DAY	Rotura	Festivo
Brit627	1	NULL	No	Laborable
Brit627	1	NULL	No	Laborable
Brit700	7	NULL	No	Laborable
Brit700	5	NULL	No	Laborable
Brit700	1	NULL	No	Laborable
Brit627	1	NULL	No	Laborable
Brit627	3	NULL	No	Laborable
Brit700	1	NULL	No	Laborable
Brit627	1	NULL	No	Laborable
Brit627	1	NULL	No	Laborable

Figura 13- df_all_tables información tabla Python

```
df_all_tables:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 380127 entries, 0 to 380126
Data columns (total 19 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   Month               380127 non-null  int64
 1   Affiliated_Code     380127 non-null  object
 2   Affiliated_NAME     380127 non-null  object
 3   POSTALCODE          380127 non-null  int64
 4   poblacion           379228 non-null  object
 5   provincia           379228 non-null  object
 6   Engage              380127 non-null  int64
 7   Management_Cluster  380127 non-null  int64
 8   Location            380127 non-null  object
 9   Tam_m2              380127 non-null  object
 10  Product_Code        380127 non-null  object
 11  SIZE                380127 non-null  int64
 12  Format              380127 non-null  object
 13  Cost_price          380127 non-null  float64
 14  Sell_price          380127 non-null  float64
 15  Margin              380127 non-null  float64
 16  Total_Delivered     380127 non-null  int64
 17  Total_Sales         380127 non-null  int64
 18  Total_OoS           380127 non-null  int64
dtypes: float64(3), int64(8), object(8)
```

Figura 14- df_Delivery_Route información tabla Python

```
df_Delivery_Route:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 117546 entries, 0 to 117545
Data columns (total 13 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   Affiliated_Code       117546 non-null  object
 1   Affiliated_Name       117546 non-null  object
 2   POSTALCODE            117546 non-null  int64
 3   poblacion             117294 non-null  object
 4   provincia             117294 non-null  object
 5   Engage                117546 non-null  int64
 6   Management_Cluster    117546 non-null  int64
 7   Location              117546 non-null  object
 8   Tam_m2                117546 non-null  object
 9   Delivery_DAY          117546 non-null  object
 10  Route_DAY             69888 non-null   object
 11  Delivery_out_of_Route 117546 non-null  object
 12  Festivo               117546 non-null  object
dtypes: int64(3), object(10)
```

Figura 15- Valores únicos df_all_tables

```
Affiliated_Code 3583
Affiliated_NAME 3547
poblacion 1901
provincia 49
Location 8
Tam_m2 8
Product_Code 57
Format 3
```

Figura 16- Sumario estadístico df_all_tables

```
Sumario estadístico de df_all_tables:
              Engage  Management_Cluster     Cost_price     Sell_price  \
count  380127.000000       380127.000000  380127.000000  380127.000000
mean        2.204363            2.027675      32.951447      39.629296
std         0.517617            1.681083      17.991260      20.632830
min         1.000000            0.000000       5.900000       7.000000
25%         2.000000            0.000000      17.380000      23.750000
50%         2.000000            2.000000      35.410000      43.500000
75%         3.000000            4.000000      47.580000      56.000000
max         3.000000            4.000000      66.600000      80.300000

              Margin  Total_Delivered    Total_Sales      Total_OoS
count  380127.000000    380127.000000  380127.000000  380127.000000
mean        6.677849        13.840606      14.826250       0.689975
std         3.330691        20.751180      20.883983       2.928816
min         1.100000      -195.000000      -8.000000       0.000000
25%         3.970000         0.000000       3.000000       0.000000
50%         6.220000         8.000000       8.000000       0.000000
75%         8.400000        16.000000      19.000000       0.000000
max        17.220000      1132.000000    1134.000000      68.000000
```

Figura 17- Visualización df_all_tables, Month Count

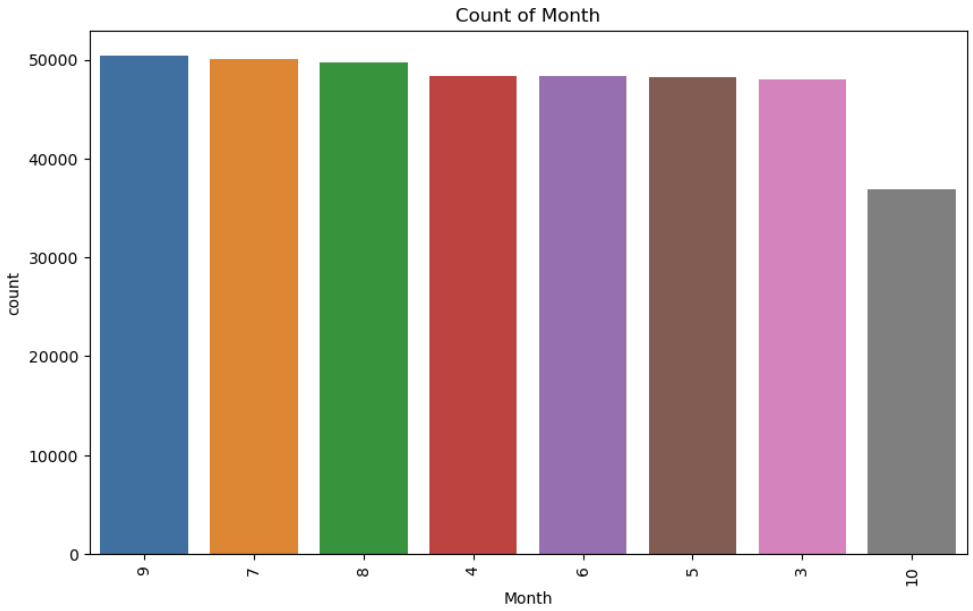


Figura 18 - Visualización df_all_tables, Provincia Count

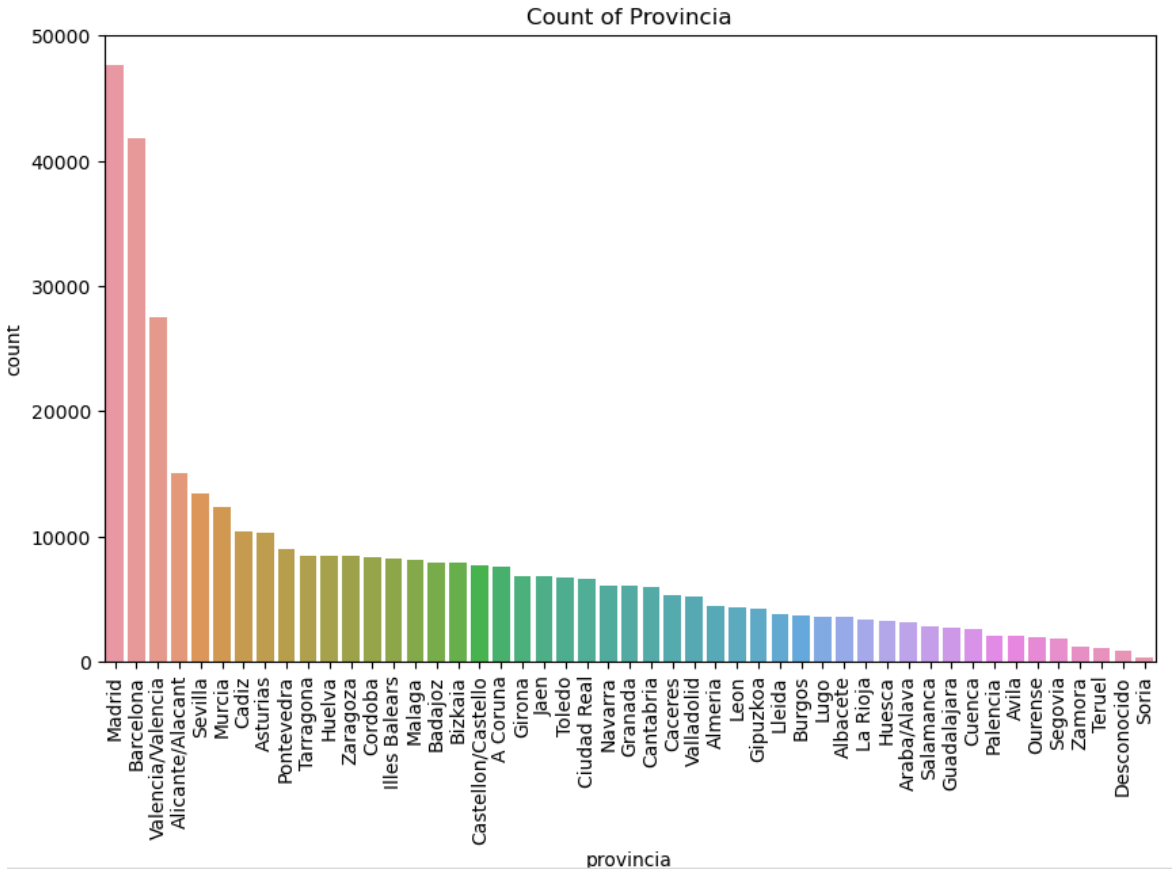


Figura 19 - Visualización df_all_tables, Engage Count

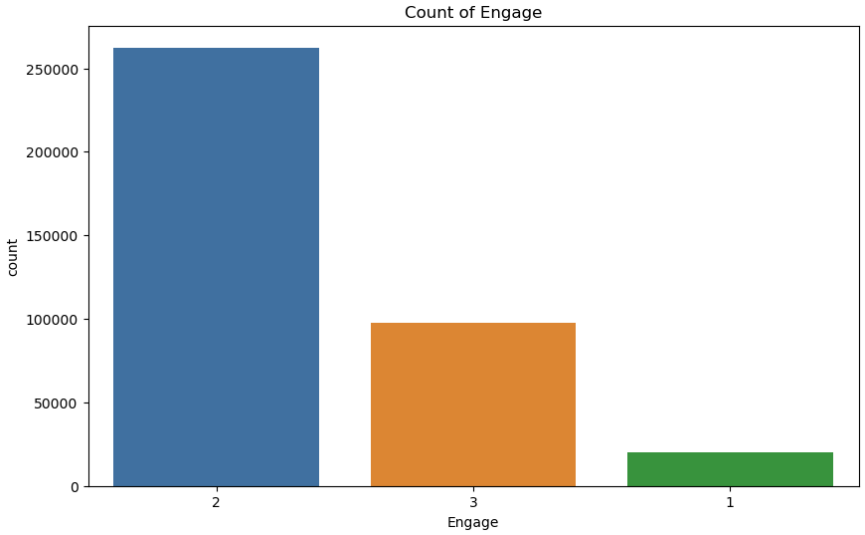


Figura 20 - Visualización df_all_tables, Management_Cluster Count

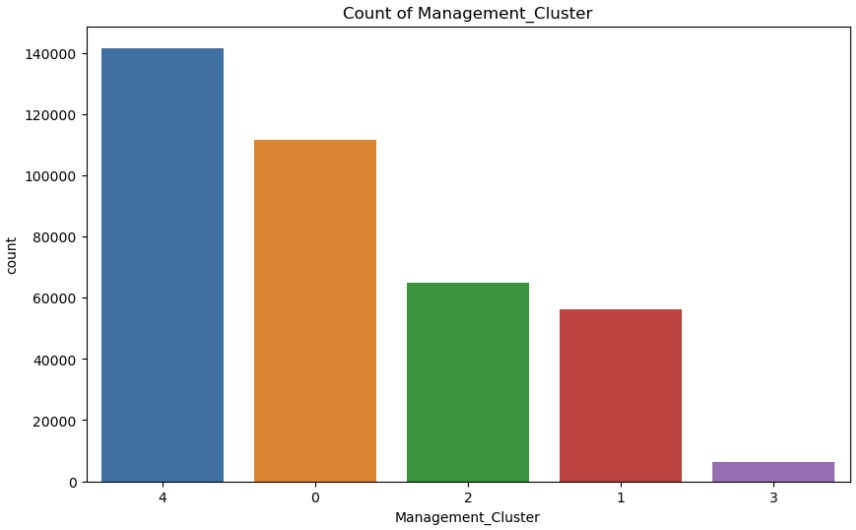


Figura 21 - Visualización df_all_tables, Location Count

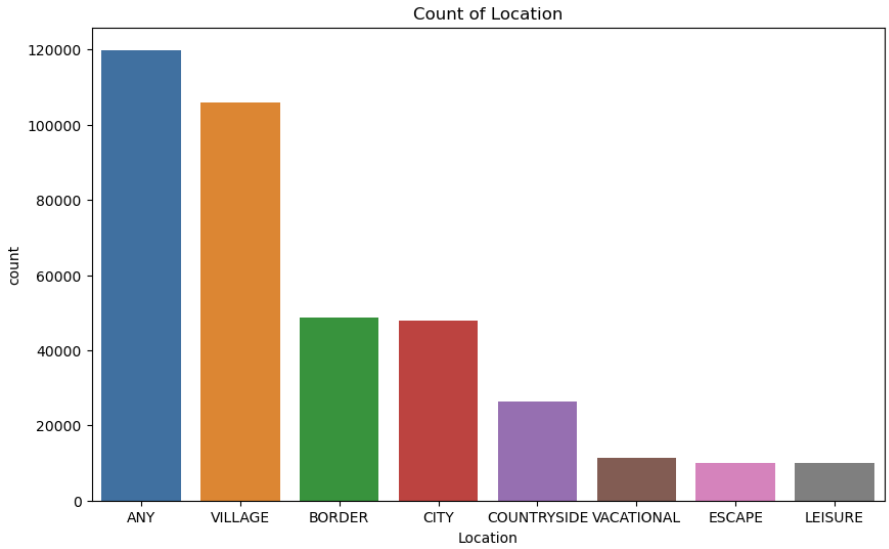


Figura 22 - Visualización df_all_tables, Tam_m2 Count

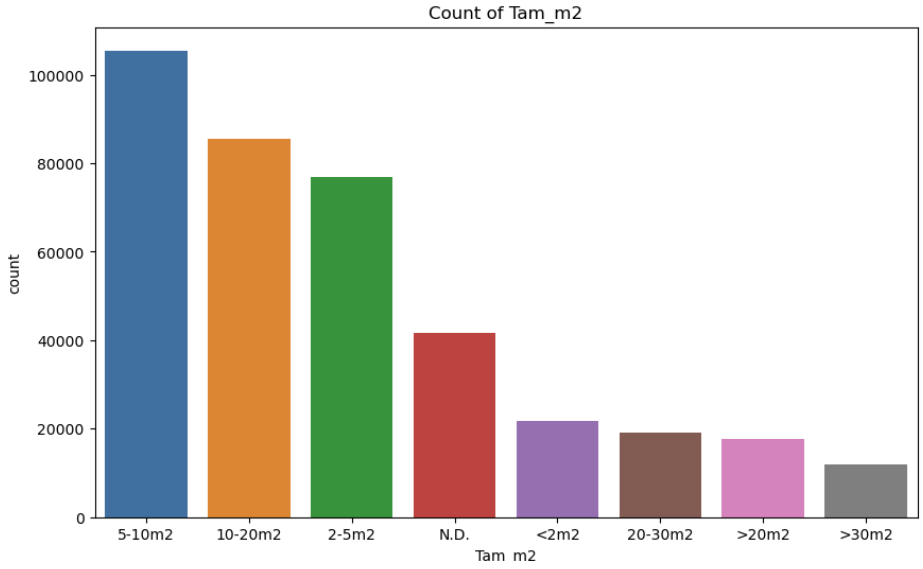


Figura 23 - Visualización df_all_tables, Product_Code Count

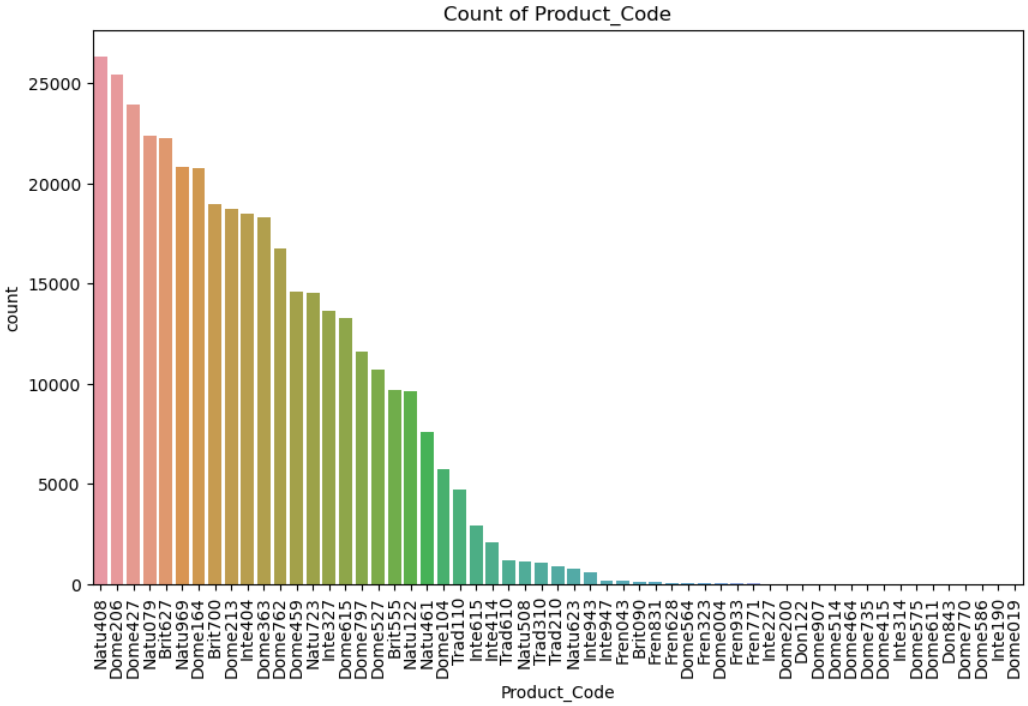


Figura 24 - Visualización df_all_tables, Size Count

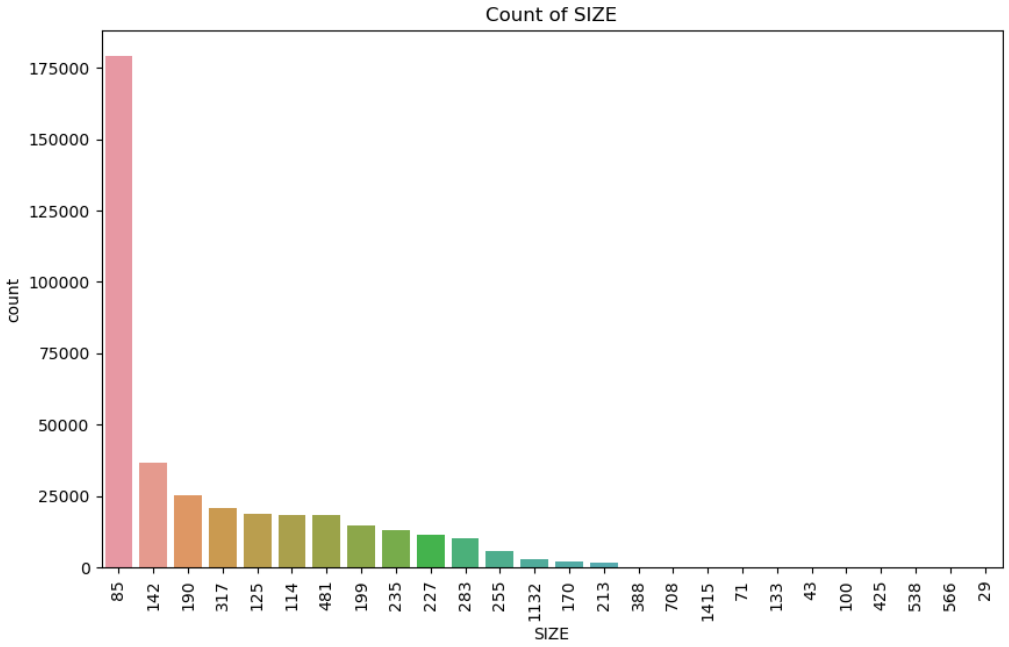


Figura 25 - Visualización df_all_tables, Format Count

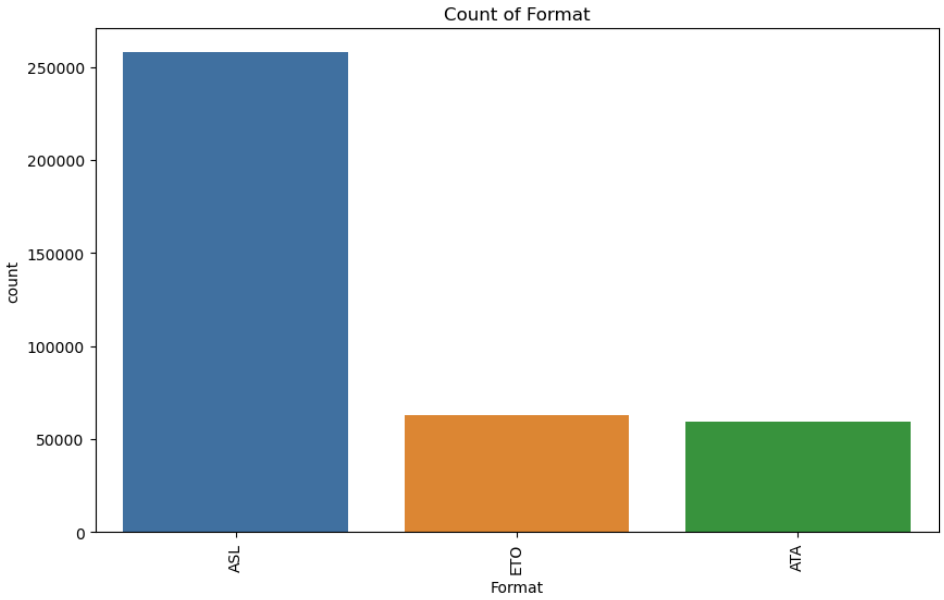


Figura 26 - Visualización df_all_tables, Total_OoS

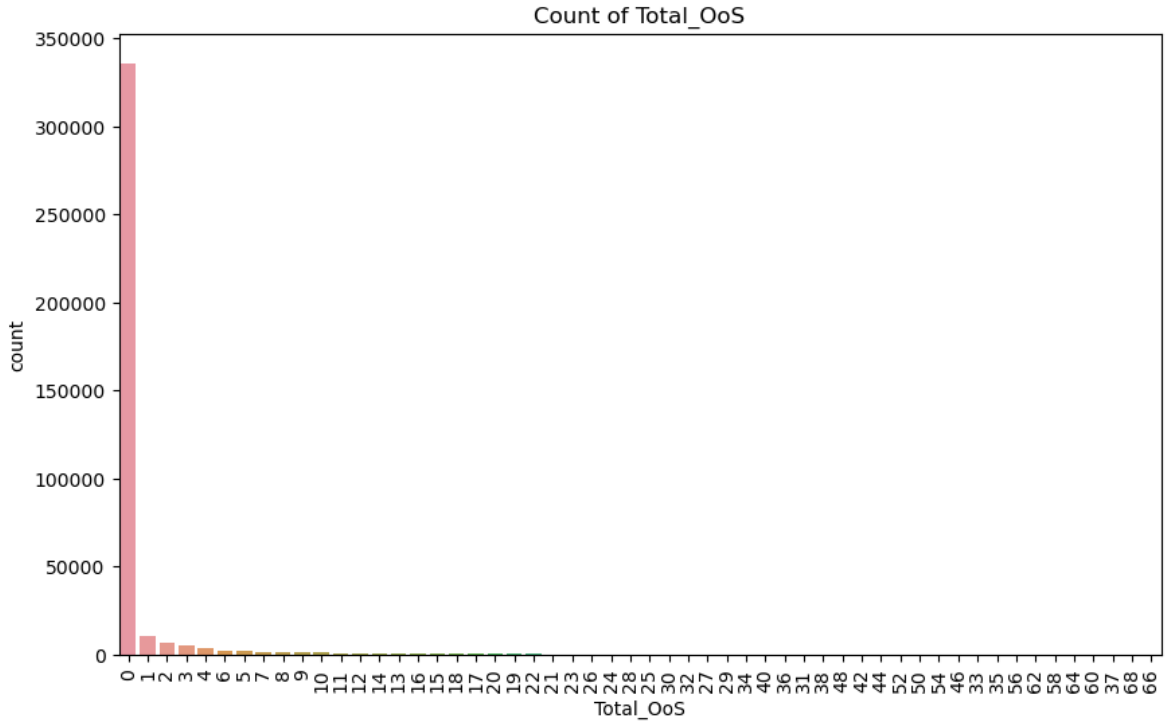
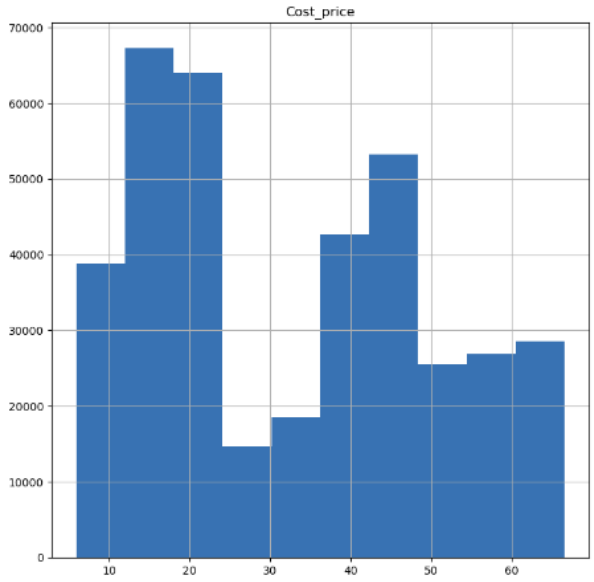
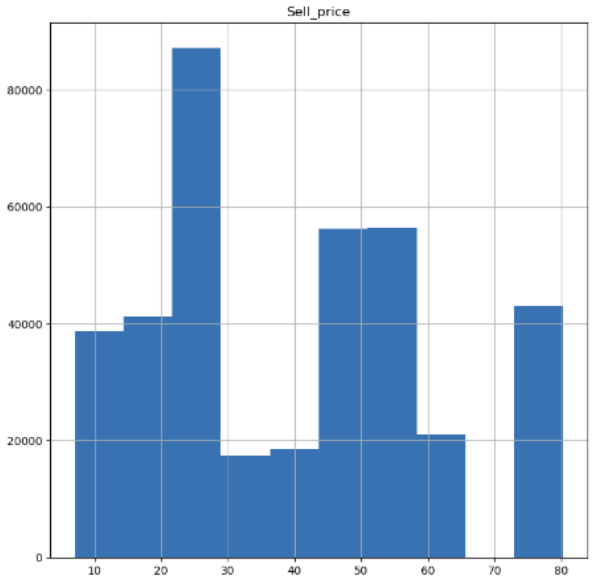
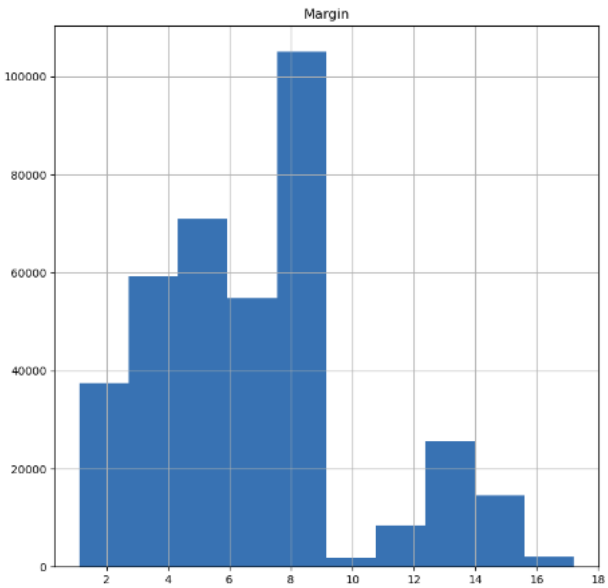
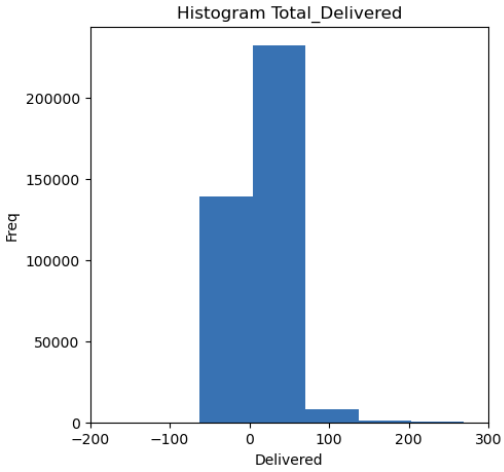


Figura 27 - Visualización univariante df_all_tables









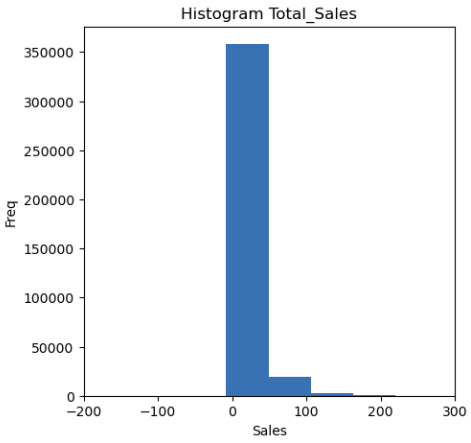


Figura 28 - Visualización de Correlación df_all_tables

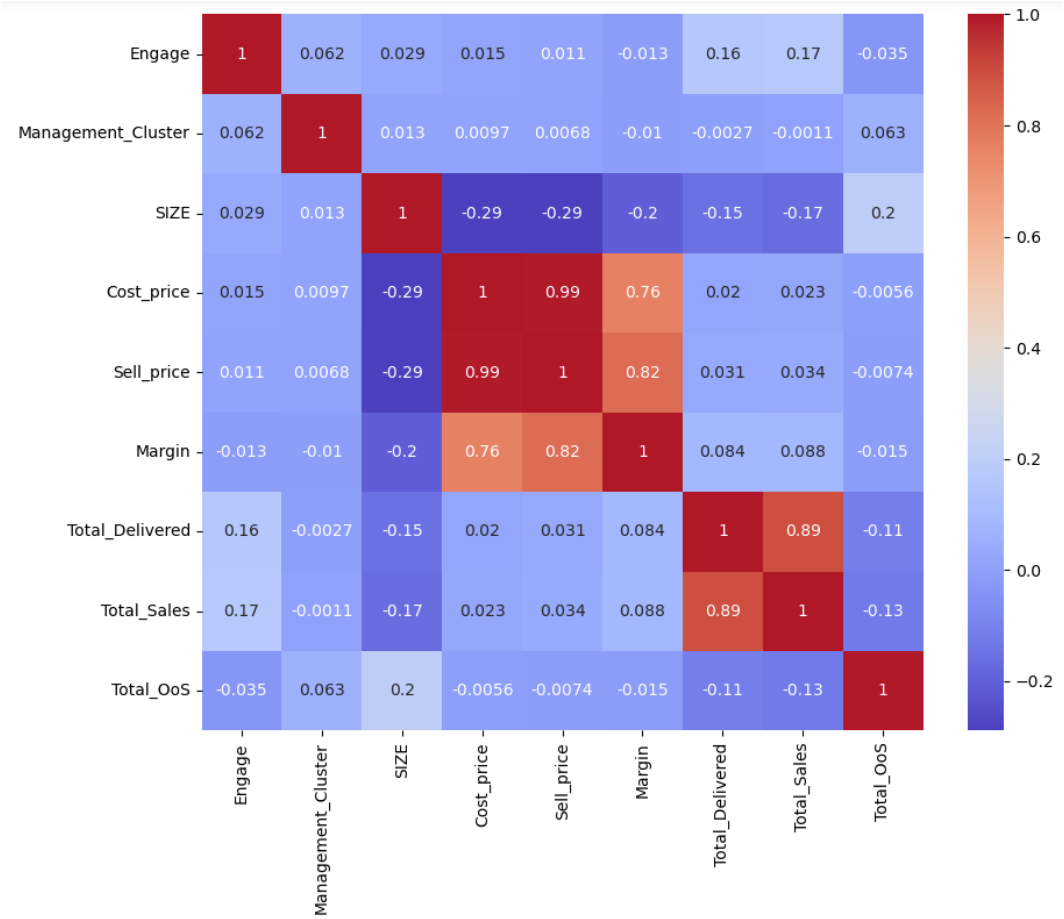
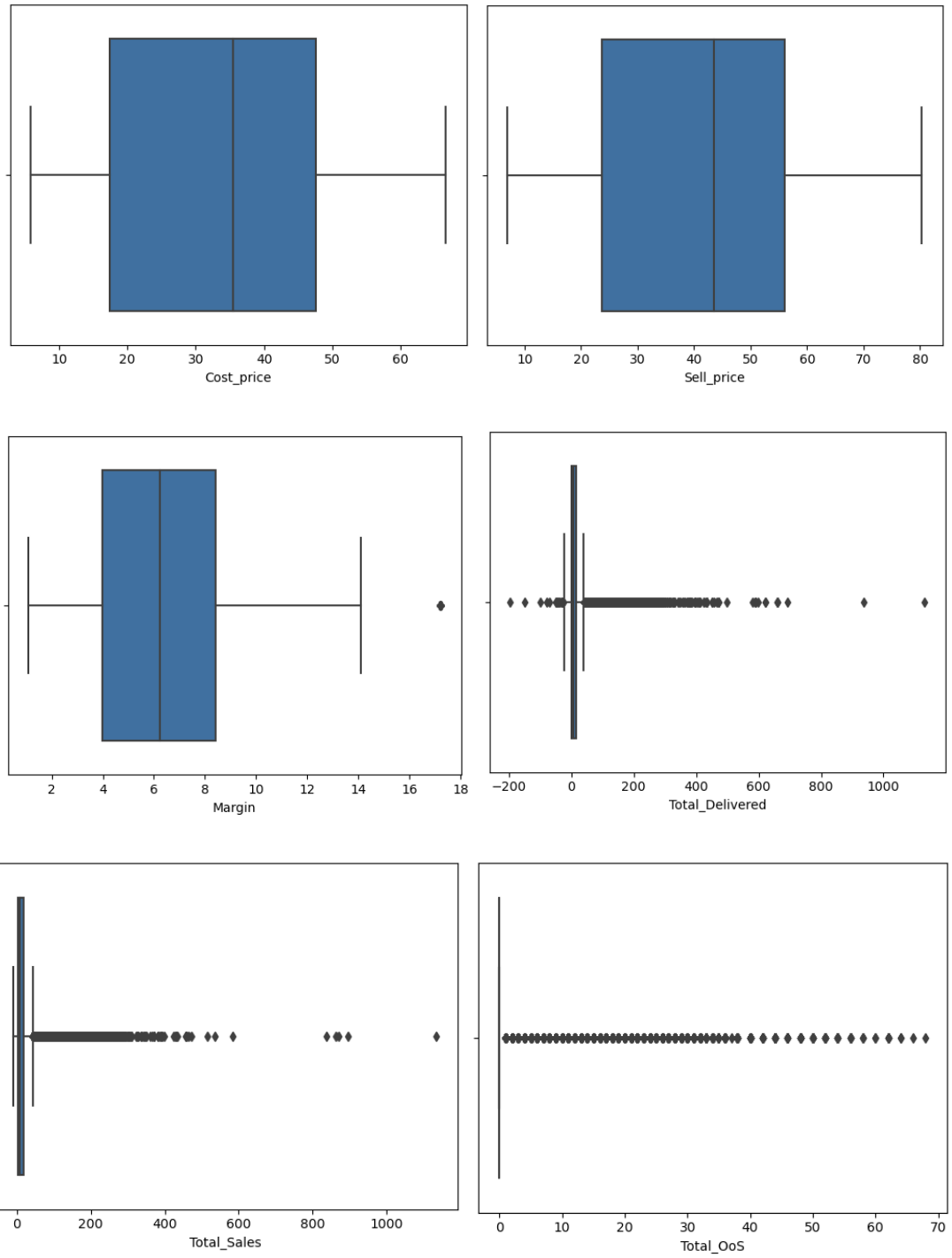


Figura 29 - Outliers df_all_tables



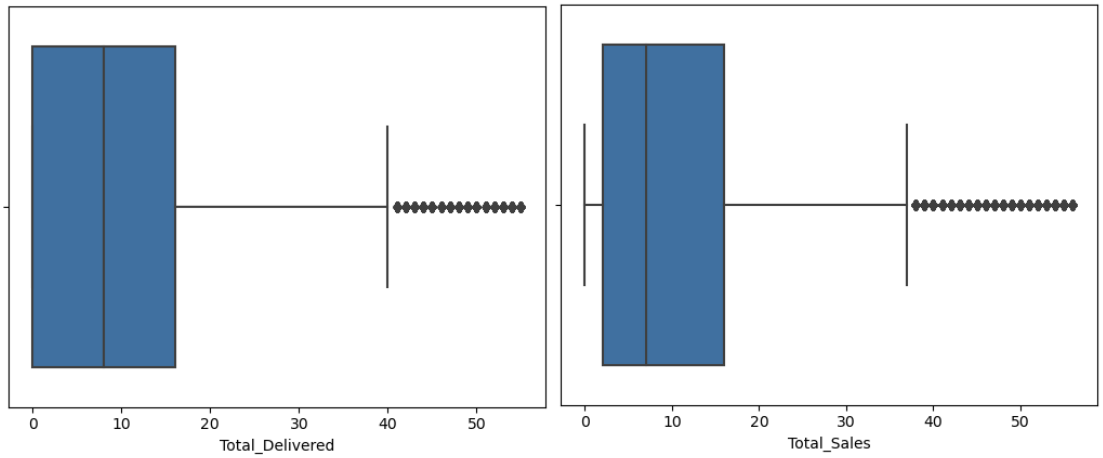


Figura 30 – Componentes transformación PCA para predicción Total_Sales

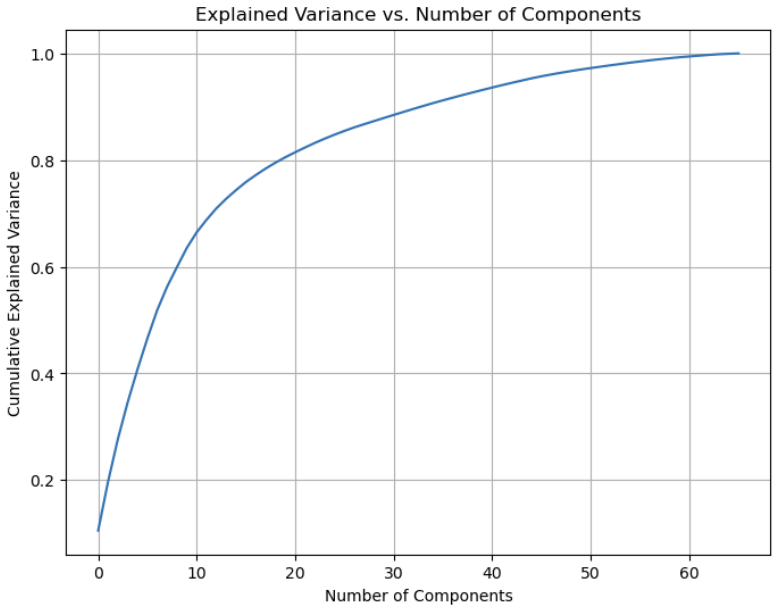


Figura 31 – MSE y R2 Regresión Lineal predicción Total_Sales PCA

```
Mean Squared Error: 135.569227250915
R-squared (R2): 0.031239138434464686
```

Figura 32 – Actual vs Predicted Total_Sales Regresión Lineal PCA

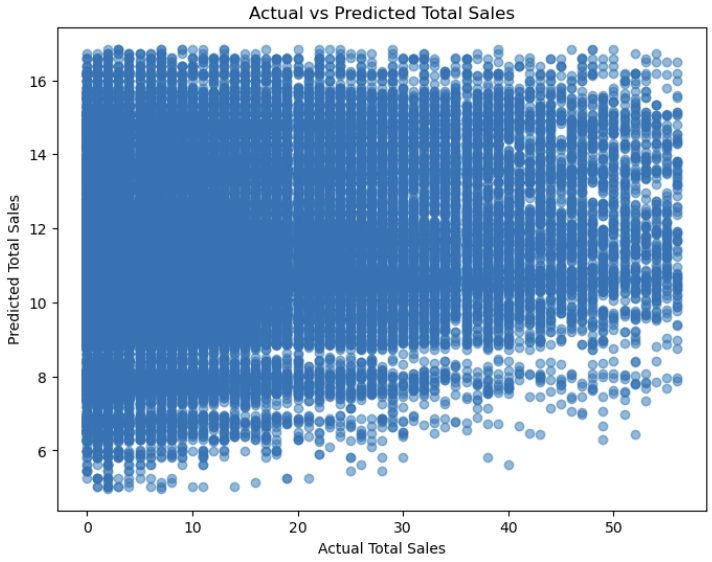


Figura 33 – Resultados Cross-Validation PCA Total_Sales

```
Cross-Validation - Mean Squared Error (MSE): 134.98373495620308 ± 0.7110005109914118
Cross-Validation - R-squared (R2): 0.0390707833132107 ± 0.0010175547399974114
```

Figura 34 – Resutados Polynomial Regression PCA Total_Sales

```
Polynomial Regression - Mean Squared Error (MSE): 134.27433135609536
Polynomial Regression - R-squared (R2): 0.04049230368546508
```

Figura 35 – Resultados Random Forest PCA Total_Sales

```
Random Forest - Mean Squared Error (MSE): 129.70666045627354
Random Forest - R-squared (R2): 0.07313231267562725
```

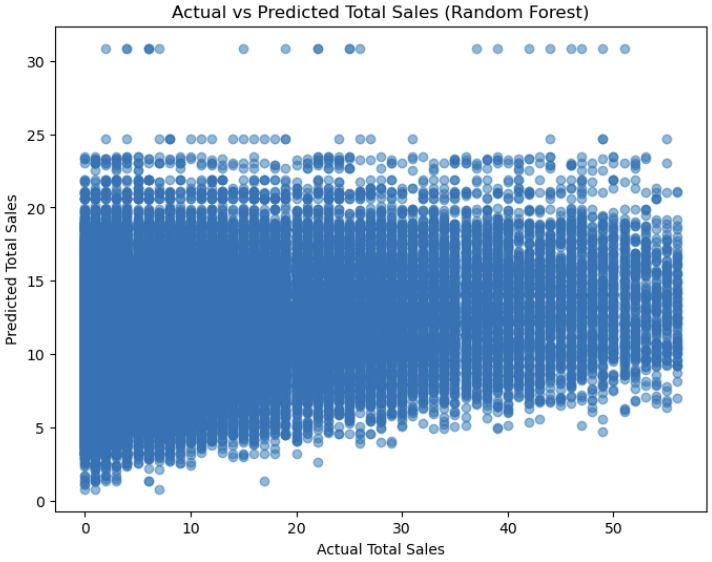


Figura 36- Revisión nuevas features Total_Sales

```
Selected Features:
   Margin  Total_Delivered  Total_OoS  Month_5  Month_10  provincia_Madrid  \
0    5.19            16.0        0.0      0.0       0.0               0.0
1    8.02             0.0        0.0      0.0       1.0               0.0
2   12.70             0.0        0.0      0.0       0.0               0.0
3    1.10            16.0        0.0      0.0       0.0               0.0
4    5.00             0.0        0.0      0.0       0.0               0.0

   Engage_1  Engage_2  Engage_3  Location_CITY
0       0.0       1.0       0.0            0.0
1       0.0       0.0       1.0            1.0
2       0.0       1.0       0.0            0.0
3       0.0       1.0       0.0            0.0
4       0.0       0.0       1.0            0.0
```

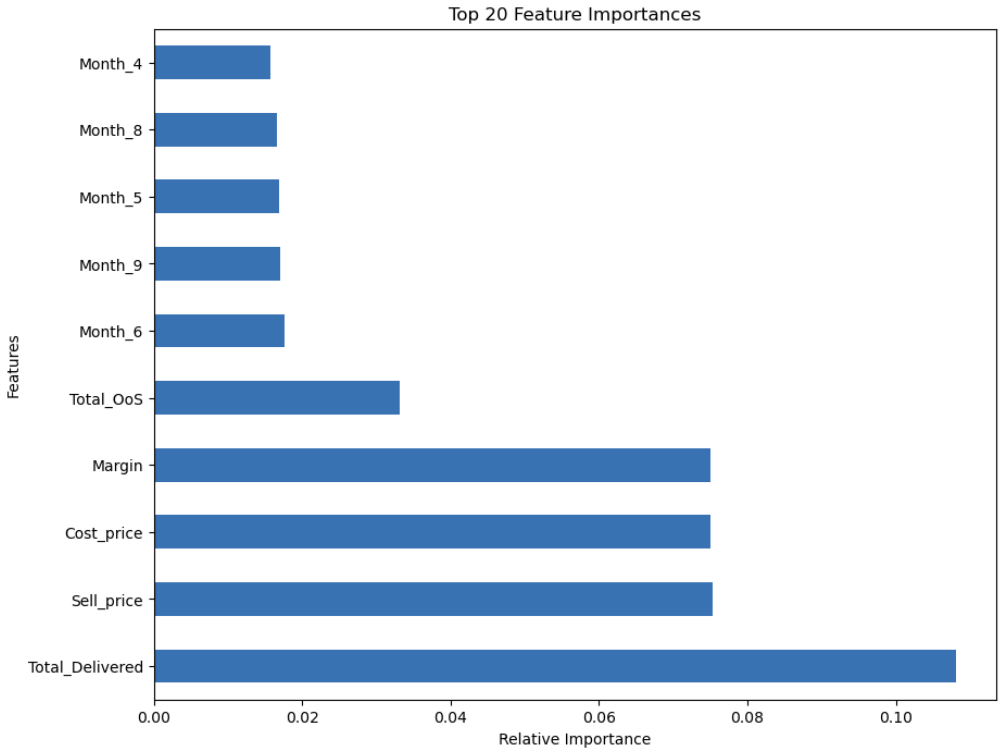


Figura 37- Resultados nueva regresión lineal Total_Sales

```
Mean Squared Error (MSE): 40.79130704613109
R-squared (R2): 0.7085103857289448
```

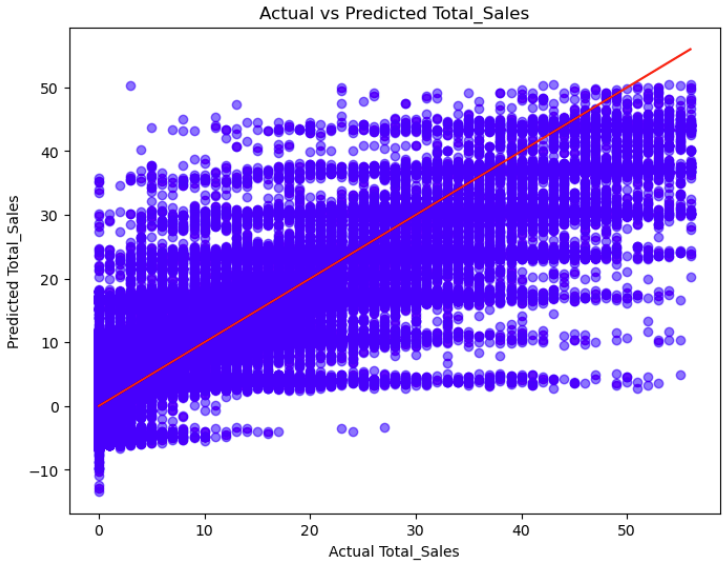


Figura 38- Resultados nueva Cross-Validation Total_Sales

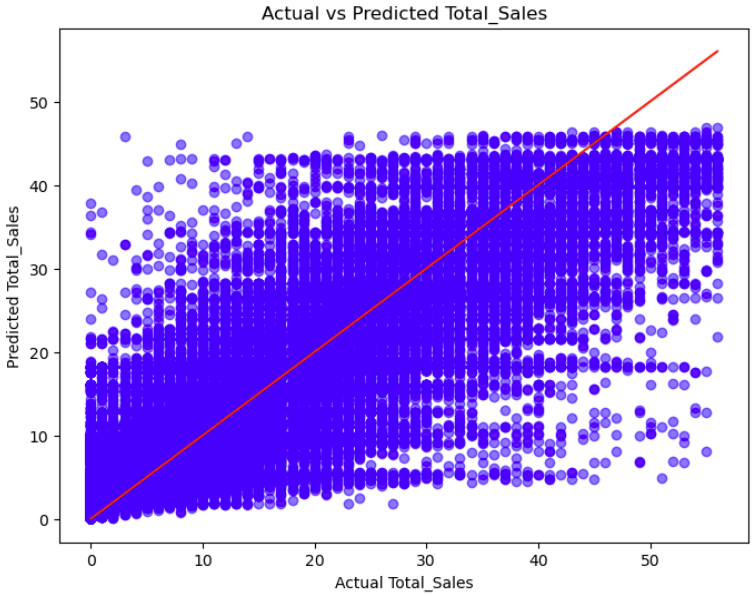
```
R-squared scores for each fold: [0.70017498 0.70444972 0.704315  ]

Mean R-squared (R2) across folds: 0.7029798995468545
Standard deviation of R-squared (R2) across folds: 0.0019841390993079223
```

Figura 39- Resultados nueva Polynomial Regression Total_Sales

```
Mean Squared Error (MSE): 35.82169933397056
R-squared (R2): 0.7440225852634661
```

Figura 40- Resultados nuevo Random Forest Total_Sales y Cross-Validation

```
Random Forest Mean Squared Error (MSE): 32.888752585901585
Random Forest R-squared (R2): 0.7649810584819225
```



```
Random Forest R-squared scores for each fold: [0.75324779 0.75786117 0.7629589 ]

Random Forest Mean R-squared (R2) across folds: 0.7580226203513999
Random Forest Standard deviation of R-squared (R2) across folds: 0.003966187344234187
```

Figura 41- Número de clusters óptimo para features Total_OoS

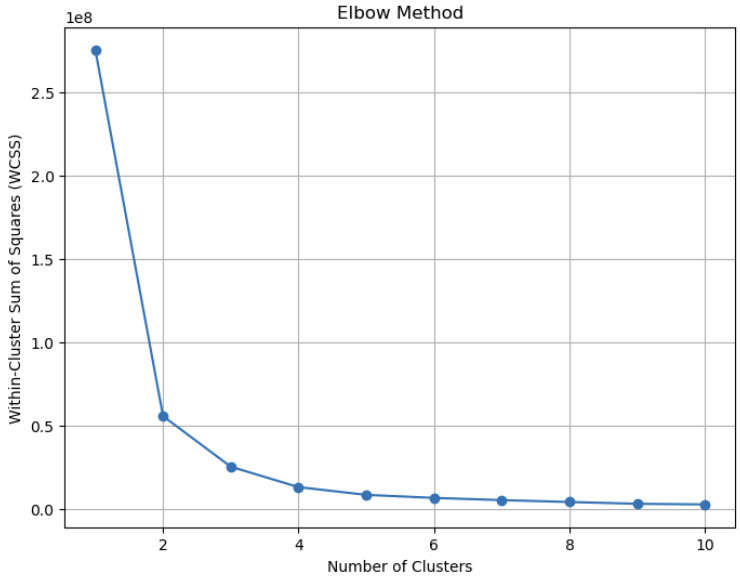


Figura 42- Revisión componentes PCA óptimos para Total_OoS

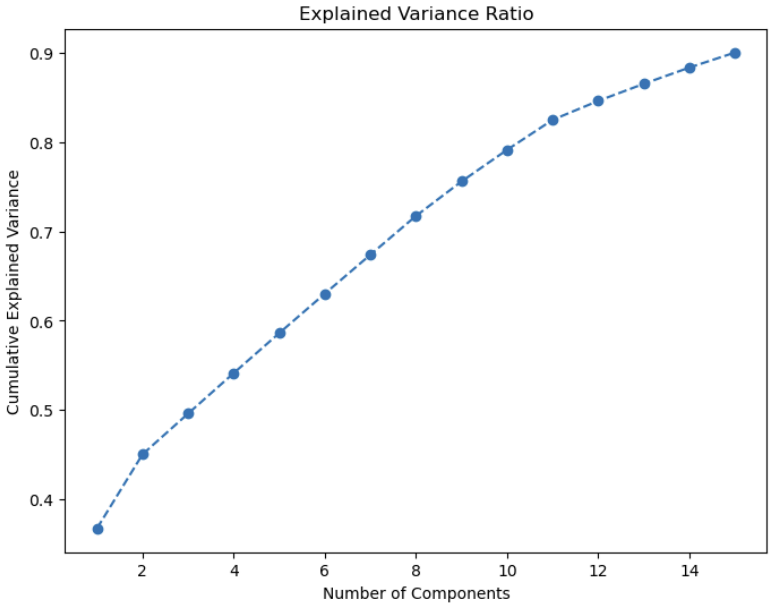


Figura 43- Resultados de Regresión Lineal tras clustering y PCA para Total_OoS

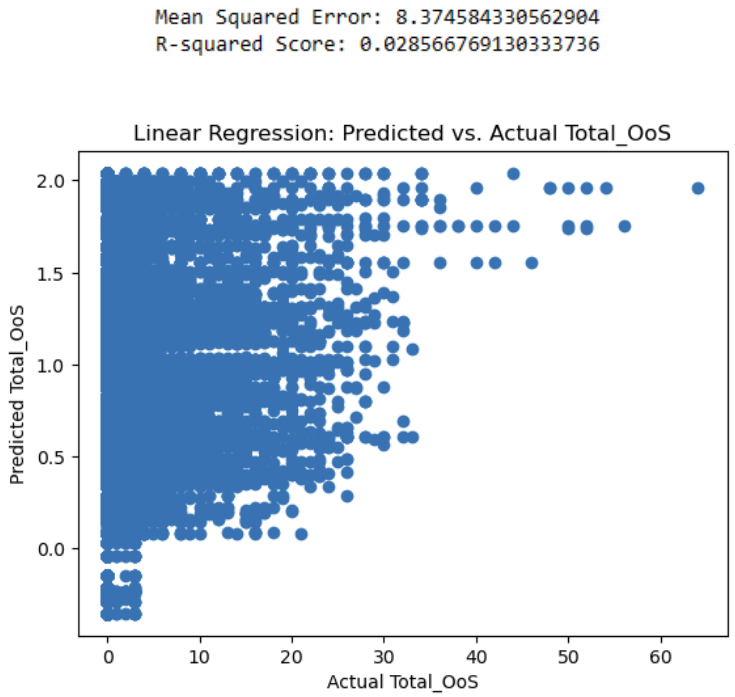


Figura 44- Visualización 3 primeros componentes PCA con asignación de clusters para Total_OoS

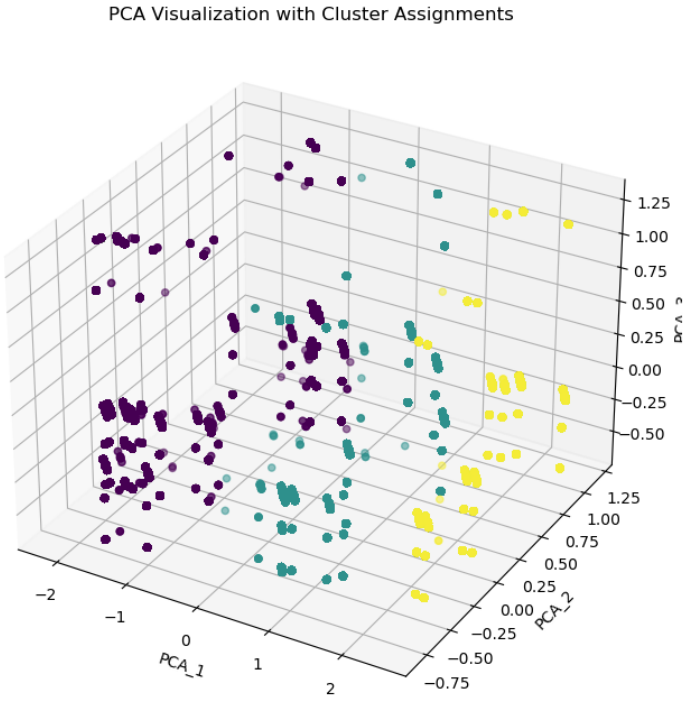


Figura 45- Resultados Random Forest sobre PCA para Total_OoS

```
Mean Squared Error with selected features: 8.375288158280846
R-squared Score with selected features: 0.028485126674179795
```

Figura 46- Resultados Gradient Boosting Regressor para Total_OoS

```
Mean Squared Error with Gradient Boosting: 7.977937779180815
R-squared Score with Gradient Boosting: 0.07457689043464866
```

Figura 47- Revisión del peso de componentes PCA componentes sobre Total_OoS

	Feature	Score	p-value
8	PCA_9	7544.566250	0.000000e+00
1	PCA_2	2177.990023	0.000000e+00
6	PCA_7	731.655330	5.676718e-161
4	PCA_5	246.828914	1.331423e-55
2	PCA_3	156.298044	7.413203e-36
3	PCA_4	63.782282	1.393635e-15
10	PCA_11	62.448274	2.743016e-15
9	PCA_10	35.520314	2.526383e-09
0	PCA_1	24.434632	7.690874e-07
7	PCA_8	23.696738	1.128212e-06
5	PCA_6	8.271974	4.026407e-03

Figura 48- Regresión Lineal sobre top5 componentes PCA

```
Mean Squared Error (MSE): 8.381405324885703
R-squared (R2) Score: 0.027775548898834157
```

Figura 49- Revisión de features con más incidencia sobre Total_OoS

```
Selected Features:
   Total_Delivered  Total_Sales  Month_10  Product_Code_Inte615  \
0              0.0          8.0       0.0                   0.0
1              0.0          2.0       0.0                   0.0
2              0.0          2.0       0.0                   0.0
3              0.0          1.0       0.0                   0.0
4              0.0          2.0       0.0                   0.0

   Product_Code_Natu122  SIZE_85  SIZE_213  SIZE_283  SIZE_1132  Format_ATA
0                   0.0      1.0       0.0       0.0        0.0         0.0
1                   0.0      1.0       0.0       0.0        0.0         0.0
2                   1.0      0.0       0.0       1.0        0.0         1.0
3                   0.0      0.0       0.0       0.0        0.0         0.0
4                   0.0      1.0       0.0       0.0        0.0         0.0
```

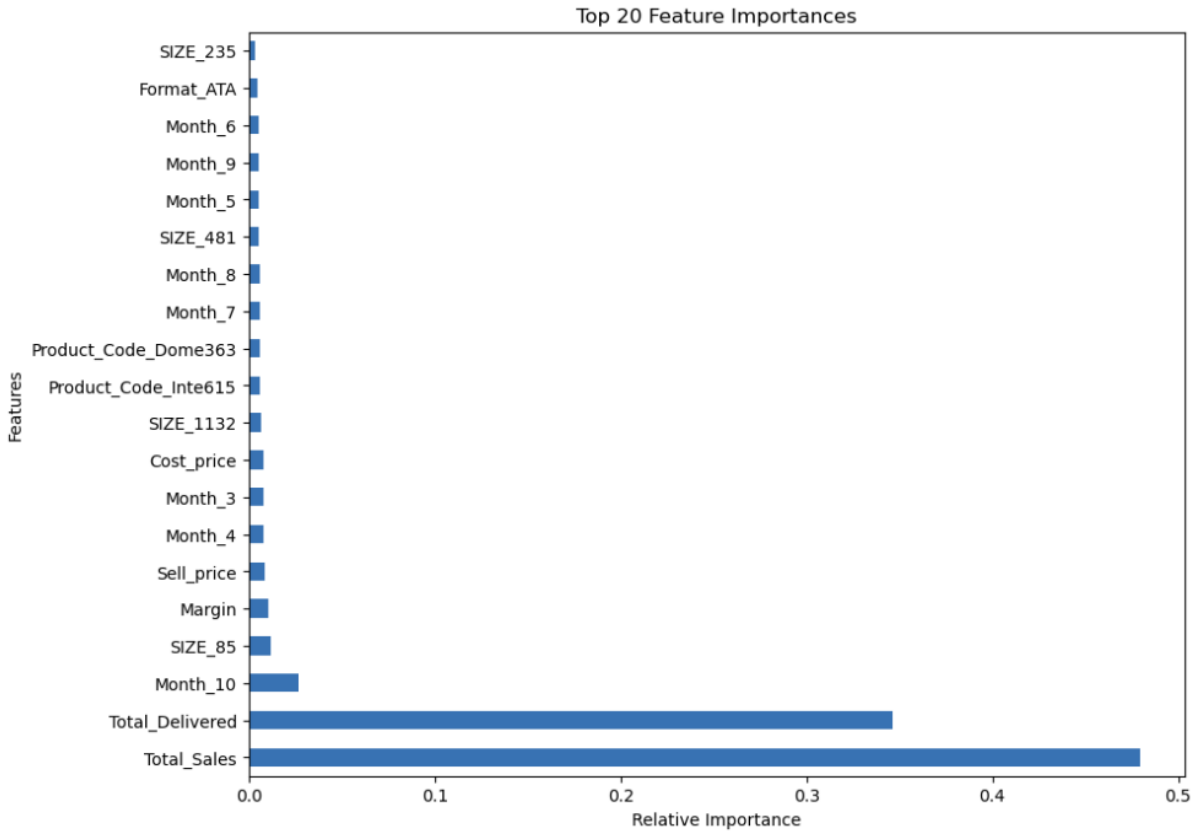


Figura 50- Resultados Decision Tree con features limitadas para Total_OoS

```
Decision Tree MSE: 5.189356056326031
Decision Tree R-squared Score 0.3980461929874646
```

Figura 51- Resultados con resampling y nuevo Decision Tree con mismas features limitadas para Total_OoS

	precision	recall	f1-score	support
0	0.96	0.62	0.76	63252
1	0.04	0.10	0.06	2102
2	0.03	0.06	0.04	1347
3	0.05	0.19	0.08	1011
4	0.02	0.05	0.02	691
5	0.01	0.04	0.02	476
6	0.02	0.07	0.03	468
7	0.01	0.04	0.02	340
8	0.01	0.05	0.02	311
9	0.01	0.06	0.02	209
10	0.01	0.05	0.02	206
11	0.01	0.03	0.01	153
12	0.00	0.02	0.01	148
13	0.01	0.03	0.01	146
14	0.01	0.04	0.02	156
15	0.03	0.16	0.05	120
16	0.00	0.00	0.00	107
17	0.01	0.03	0.01	98
18	0.01	0.06	0.02	89
19	0.01	0.05	0.01	59
20	0.00	0.01	0.00	80
21	0.01	0.06	0.01	47
22	0.00	0.02	0.01	53
23	0.02	0.15	0.03	39
24	0.00	0.00	0.00	38
25	0.01	0.12	0.02	24
26	0.00	0.00	0.00	39
27	0.01	0.22	0.02	9
28	0.01	0.03	0.02	29
29	0.00	0.00	0.00	6
30	0.03	0.11	0.05	18
31	0.00	0.00	0.00	5
32	0.00	0.00	0.00	10
33	0.00	0.00	0.00	2
34	0.00	0.00	0.00	12
35	0.00	0.00	0.00	0
36	0.00	0.00	0.00	5
37	0.00	0.00	0.00	0
38	0.00	0.00	0.00	2
40	0.00	0.00	0.00	3
42	0.00	0.00	0.00	2
44	0.00	0.00	0.00	2
46	0.07	1.00	0.12	1
48	0.00	0.00	0.00	2
50	0.00	0.00	0.00	3
52	0.00	0.00	0.00	3
54	0.00	0.00	0.00	1
56	0.00	0.00	0.00	1
58	0.00	0.00	0.00	0
60	0.00	0.00	0.00	0
62	0.00	0.00	0.00	0
64	0.00	0.00	0.00	1
accuracy			0.56	71926
macro avg	0.03	0.07	0.03	71926
weighted avg	0.85	0.56	0.67	71926

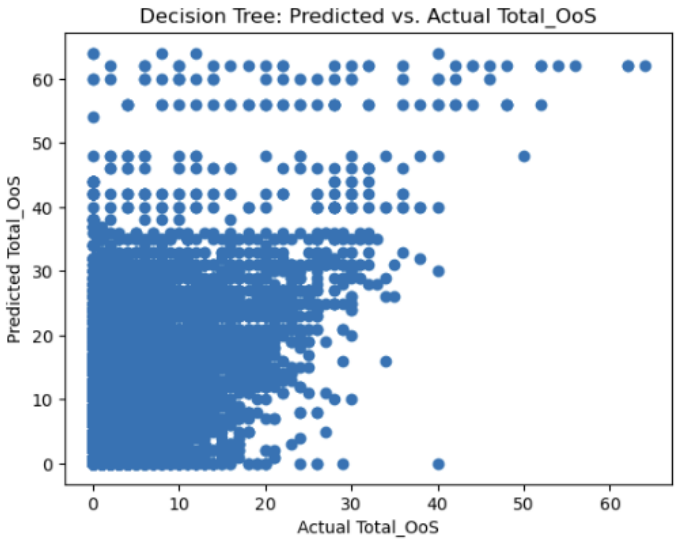


Figura 52- Resultados con resampling y GausianNB con features limitadas para Total_OoS

accuracy			0.07	71926
macro avg	0.02	0.06	0.01	71926
weighted avg	0.86	0.07	0.12	71926

Figura 53- Resultados con resampling y SVM con mismas features limitadas para Total_OoS y solo una muestra de datos de 0.03

accuracy			0.41	2158
macro avg	0.04	0.07	0.04	2158
weighted avg	0.83	0.41	0.54	2158

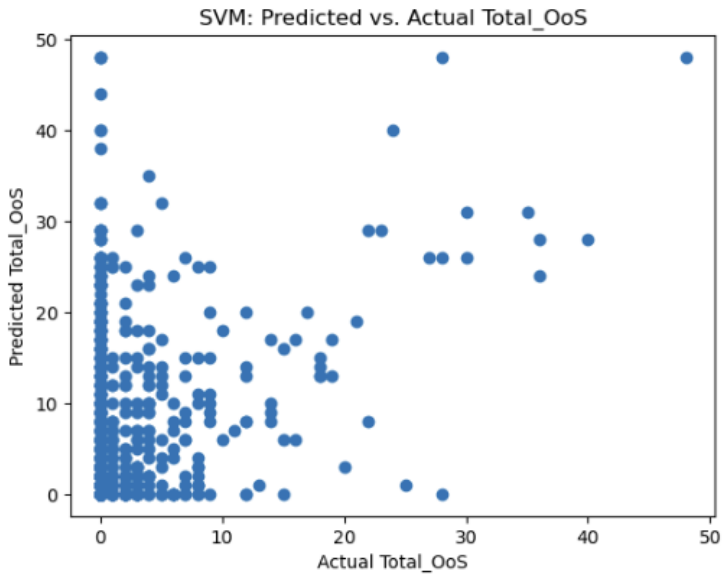


Figura 54- Presentación de predicciones Total_Sales para Month_11 en PowerBi.

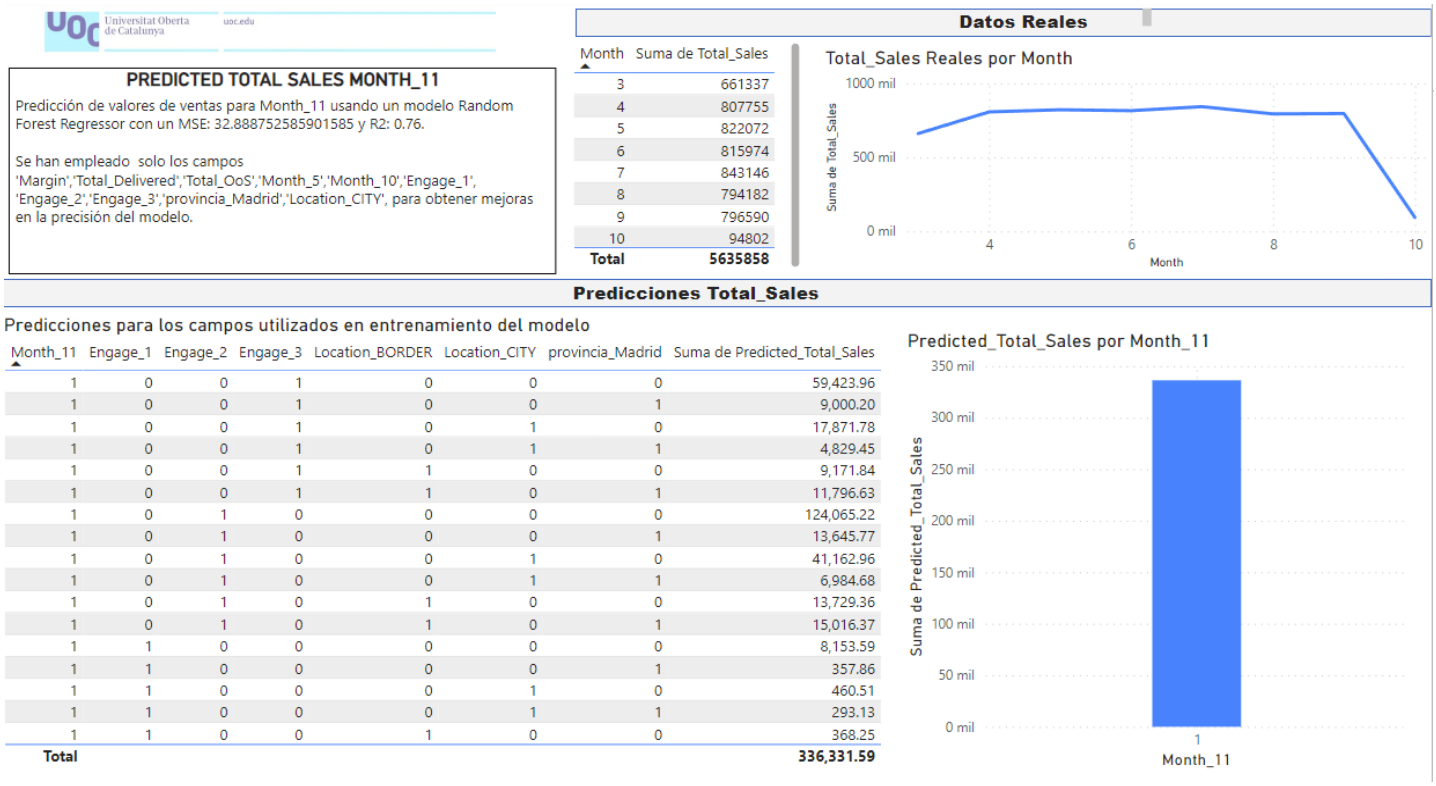
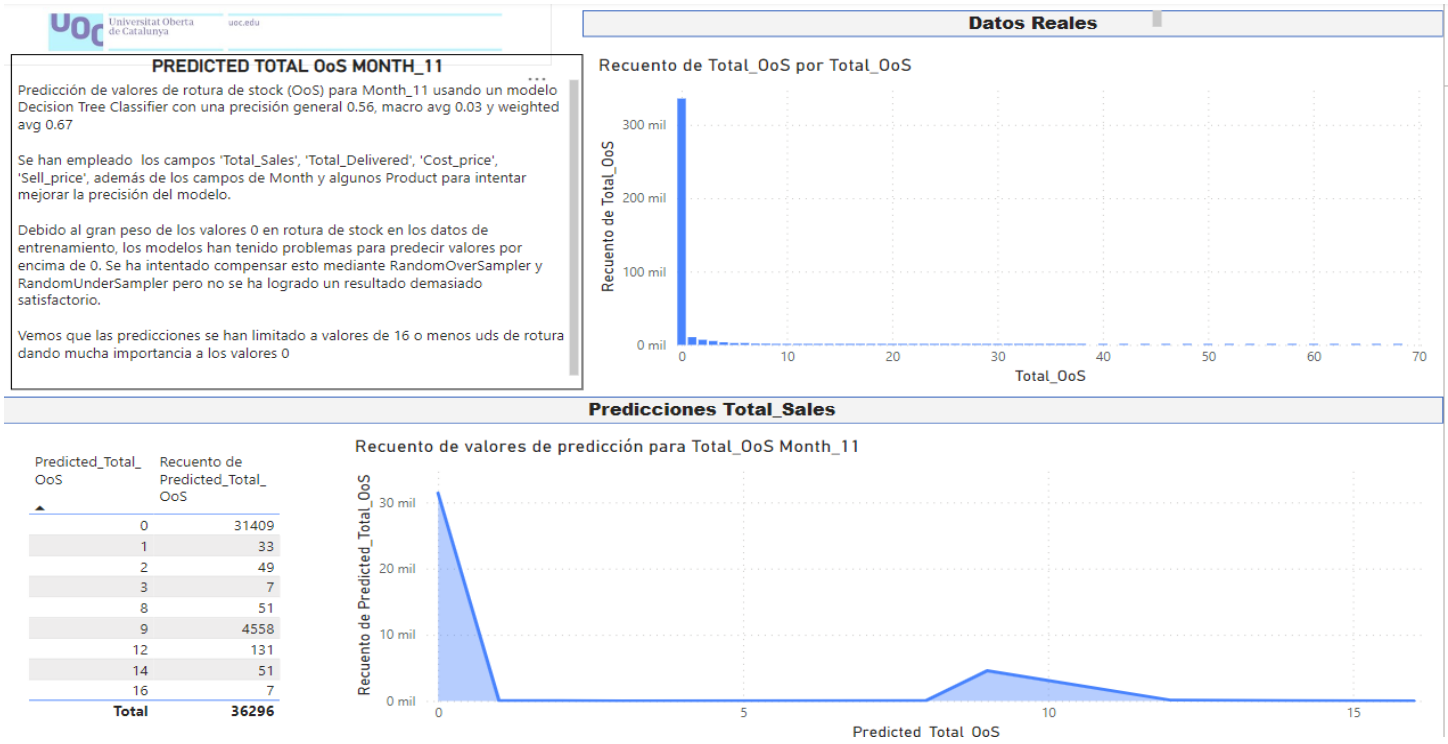


Figura 55- Presentación de predicciones Total_OoS para Month_11 en PowerBi.



1. Introducción

Este TFM se centra en los datos obtenidos de la cadena de valor de la comercialización de productos, buscando monitorizar, analizar y predecir el comportamiento de ventas, roturas de stock de productos y establecimientos y entregas.

1.1. Contexto y justificación del Trabajo

Estudiar estos datos desde un punto de vista de Análisis y Ciencia de datos puede permitir optimizar las decisiones a tomar por parte de departamentos de ventas, compras, logística, evitando sobrestock que cause tener producto caducado u obsoleto, asegurando que el consumidor siempre tenga acceso a los productos que necesita evitando roturas de stock, poder mejorar entregas o reducir el tiempo que las mercancías pasan almacenadas (reduciendo así costes de almacenamiento).

La motivación para este TFM viene dada por la experiencia profesional del estudiante durante muchos años en diferentes departamentos de compras o logística realizando pedidos y analizando variaciones de stock.

1.2. Objetivos del Trabajo

Este TFM se ha centrado finalmente en la predicción de unidades vendidas (Total_Sales) y unidades de rotura de stock (Total_OoS). Otros ámbitos que finalmente se han descartado han sido el análisis de ventas o rotura de stock según días festivos, clusterización por tiendas, código postal, provincias o productos.

La Hipótesis puede ser de causalidad, donde intentaremos ver si hay algún factor concreto que cause roturas de stock, y mayores o menores ventas de productos. También podemos tener hipótesis descriptiva para explicar el comportamiento del consumo de determinados productos.

Algunos de los objetivos parciales pueden ser:

- Con qué frecuencia, en qué periodos de tiempo y para qué productos o tiendas se suelen producir roturas de stock.
- Revisar los productos o tiendas con más ventas y periodos de tiempo relacionados.

1.3. Impacto en sostenibilidad, ético-social y de diversidad

Este TFM puede encuadrarse dentro de la dimensión de sostenibilidad impactando en el siguiente punto:

ODS 12 - Responsible consumption and production: el hecho de poder estudiar datos de stock o ventas y realizar predicciones sobre los mismos puede llevar a una gestión más eficiente del stock producido o almacenado. Esto supone por un lado evitar roturas de stock y por otro evitar sobrestocks que pudieran generar producto caducado u obsoleto que debiera ser retirado sin haber sido utilizado o consumido.

1.4. Enfoque y método seguido

La estrategia de investigación será por un lado analítica, ya que utilizaremos datos disponibles sobre ventas, roturas, intentando entender un conjunto de datos complejo, estudiando y analizando los datos. Estudiamos una relación causa-efecto, por ejemplo, entre roturas de stock, productos y fechas determinadas.

También será una estrategia predictiva, ya que intenta entender el futuro de lo que queremos estudiar y su fundación se basa en la probabilidad. Por ejemplo, prever esas roturas de stock para poder minimizar su impacto.

La metodología utilizada será CRISP-DM (Cross-Industry Standard Process for Data Mining).

Esta metodología cuenta con 6 fases:

- análisis del problema
- análisis de los datos
- preparación de los datos
- modelado
- evaluación
- explotación

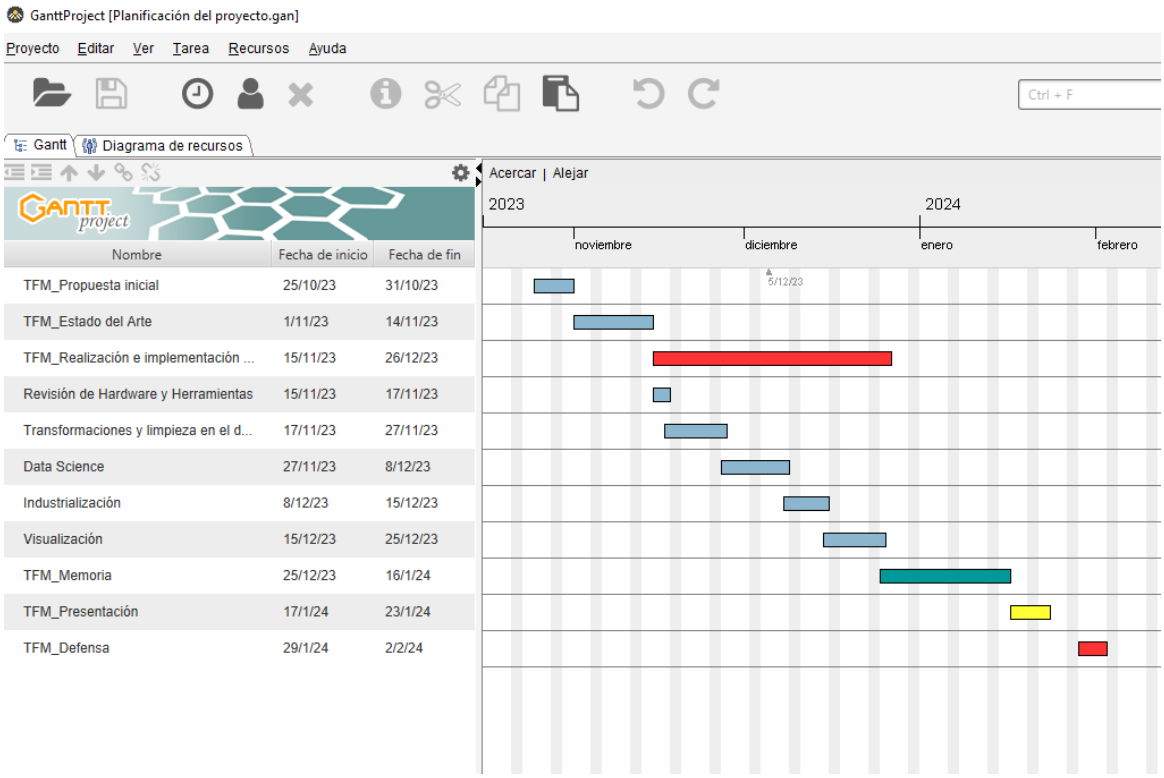
En esta metodología tiene mucha importancia el “Data Understanding”, se emplearán herramientas como Excel, SQLServer y Python para realizar este análisis.

1.5. Planificación del trabajo

Las tareas a realizar durante la realización/implementación del proyecto son las siguientes:

- Revisar el HW para el servidor: Excel, MSSQL, Python, PowerBi, Azure. 15 Nov- 17 Nov
- Transformaciones y limpieza en el dataset: descartar outliers, full outer join, disponibilidad del dato, pivot data, feature engineering, clustering y dimensiones derivadas. 18-28 Nov
- Data Science: análisis exploratorio de las variables, algoritmos. 29 Nov-8 Dic.
- Industrialización: carga en BBDD del proceso de predicción. 8 Dic – 16 Dic.
- Visualización: dashboards con PowerBi y Python. 17-23 Dic.

Además, con el uso de Gantt Project se han establecido los plazos para todas las actividades del TFM:



1.6. Estado del Arte

Tema de investigación
Este TFM se centra en los datos obtenidos de la cadena de valor de la comercialización de productos, buscando monitorizar, analizar y predecir el comportamiento de ventas y rotura de stock principalmente. Esto permitirá mejorar tiempos de entrega, asegurar que las tiendas tienen siempre el stock necesario disponible, mejorando la satisfacción del cliente y los beneficios de la organización.

Palabras clave
Las palabras clave utilizadas han sido: *Data Analysis, Supply Chain, Data Science, Logistics, Data mining, Algorithms*. Otros términos como "Stock" se han excluido ya que derivaban en documentos relacionados con acciones financieras.

Fuentes de información
Las fuentes de información consultadas para obtener papers han sido:
Springer Link: plataforma con acceso a ebooks, trabajos de referencia, revistas y protocolos en un amplio rango de disciplinas.

Science Direct: dispone de publicaciones técnicas, científicas y de salud. Es la plataforma que ha dado más resultados, aunque algunas de las publicaciones obtenidas son algo más antiguas que en las otras fuentes.

ACM Digital Library: plataforma de investigación, descubrimiento y networking con colecciones de revistas, newsletters, journals entre otras publicaciones.

Revisión bibliográfica.
La revisión bibliográfica se ha dividido en temas para facilitar su lectura y comprensión y se han ordenado los documentos cronológicamente dentro de cada apartado.

Big Data aplicado a logística y supply chain

"Big data analytics in logistics and supply chain management: Certain investigations for research and applications" Gang Wang, Angappa Gunasekaran, Eric W.T. Ngai, Thanos Papadopoulos. Version of Record 31 March 2016.

Revisa literatura sobre la aplicación de BDBA (Big Data Business Analytics) en Logística y Supply Chain Management (LSCM) y lo define como SCA (Supply Chain Analytics) basado en el tipo de análisis (descriptivo, predictivo o prescriptivo) y el foco de LSCM (estrategia y operaciones). Propone una estructura de SCA basado en cuatro niveles de capacidad: funcional, basado en proceso, colaborativo, SCA ágil y SCA sostenible.

El SCA puede ayudar a medir el desempeño de varias áreas de logística y supply chain y proporcionarles la posibilidad de establecer un punto de referencia para determinar operaciones de valor añadido.

Como limitaciones, este documento se basa en datos de artículos puramente académicos. Además, el estudio se basa en los años 2004-2014, por lo que pueden haber surgido nuevos avances en los últimos años.

"Big data analytics in supply chain management: A state-of-the-art literature review". Truong Nguyen, Li ZHOU, Virginia Spiegler, Petros Ieromonachou, Yong Lin. Version of Record 6 July 2018.

Realiza una clasificación para identificar funciones de Supply Chain con niveles de analítica, modelos y técnicas BDA (Big Data Analytics). Intenta dar una imagen global sobre como el BDA se ha aplicado a Supply Chain mapeando los modelos BDA a funciones de la cadena de suministro.

El análisis prescriptivo es el que más está creciendo en el ámbito de supply chain gestionado con BDA, seguido por análisis predictivo, mientras que el descriptivo está siendo menos considerado. En el prescriptivo, la "optimización" es una de las aproximaciones usadas en logística. La "clasificación" es una aproximación del análisis predictivo sobre todo en planificación y control. El análisis semántico se limita a planificación de demanda, y "asociación" se usa para análisis descriptivos en muchas áreas de SC.

El periodo de estudio abarca de 2011 a 2016, por lo que puede haber habido cambios significativos en los últimos años.

"Dimensions of Data Analytics in Supply Chain Management: Objectives, Indicators and Data Questions". Patrick Brandtner ,Chibuzor Udokwu, Farzaneh Darbanian, Taha Falatouri. January 2021.

Hay procesos como CRISP-DM que fallan al intentar proporcionar detalles específicos de dominio a nivel operativo, sobretodo en SCM. Este paper revisa la literatura para realizar la fase de comprensión de negocio de CRISP-DM en proyectos SCM DA. Se intentan proporcionar las preguntas DA adecuadas según los objetivos DA y las actividades SCM.

Se aplica en un proyecto DA para demostrar la aplicabilidad de la investigación, mostrando que el tiempo requerido para realizar la fase de comprensión de negocio se podría reducir significativamente. Se identifican medidas cuantificables para asegurar esa mejora y se mapea a 9 actividades clave de SCM. A partir de estas medidas, se desarrollan 69 preguntas para DA en SCM.

Sin embargo, la subjetividad de los investigadores puede haber influenciado la generación de las preguntas iniciales, y además se puede haber generalizado.

"Data Preprocessing in Supply Chain Management Analytics - A Review of Methods, the Operations They Fulfill, and the Tasks They Accomplish". Tobechi Obinwanne, Zimmermann Robert, Udokwu Chibuzor, Brandtner Patrick. January 13–15, 2023.

Se centra en el preprocesado de datos como uno de los pasos más importantes en análisis de datos, sobre todo a nivel Supply Chain. Identifica tareas de preprocesado de datos en análisis de SCM, operaciones y métodos usados para alcanzar los objetivos de las distintas operaciones.

PCA (principal component analysis) es uno de los métodos más utilizados para el preprocesado, y se subraya la necesidad de transformar los datos manipulando las características para que, al aplicar los algoritmos de datos, se puedan obtener resultados óptimos. La transformación de datos es la tarea de preprocesado más frecuente, mientras que reducción de dimensionalidad, normalización y tratamiento de datos ausentes son los más utilizados como operaciones de preprocesado. Las tareas que más se realizan en preprocesado de SCM analytics son transformación, limpieza, y reducción de datos.

Segmentación de clientes

"RFM model customer segmentation based on hierarchical approach using FCA". Chongkolnee Rungruang, Pakwan Riyapan, Arthit Intarasit, Khanchit Chuarkham, Jirapond Muangprathub. Version of Record 18 September 2023.

La mayoría de metodologías de clustering solo proporcionan grupos o segmentos, donde los clientes de cada grupo tienen características similares, pero sin relevancia en la información de cada uno de ellos.

Este documento propone un enfoque jerárquico usando un nuevo algoritmo de clustering que combina Novedad (Recency), Frecuencia (Frequency) y Monetario (Monetary) dando lugar al modelo RFM. Esta metodología usa las ventajas de FCA (formal concept analysis) por lo que contiene conocimiento explícito e implícito. El conocimiento explícito muestra información en el modelo de estructura jerárquico, mientras que el implícito está incorporado en la estructura con sus propiedades. Esto hace que la estructura de conocimiento de FCA revele relaciones entre puntos de datos de una manera comprensible.

La ventaja del uso del modelo RFM es que usan pocas variables para reducir la complejidad del modelo. Se sugiere añadir más variables para conseguir más precisión e información y usar valores no-binarios con fuzzy sets para mejorar el modelo.

Routing y última milla

"Towards Equitable Assignment: Data-Driven Delivery Zone Partition at Last-mile Logistics". Baoshen Guo,Shuai Wang, Haotian Wang, Yunhuai Liu, Fanshuo Kong, Desheng Zhang, Tian He. August 6–10, 2023.

Estudia la última milla, proponiendo la zona de entrega para asignar tareas de entrega. Cada repartidor tiene asignada una zona y recogen pedidos para ellas en estaciones para luego realizar las entregas a los clientes. Se propone la E-partition, estructura guiada por datos de partición de zonas de entrega para conseguir un equilibrio en la carga de trabajo de la logística de última milla.

Diseña un modelo de aprendizaje y predicción basado en carga de trabajo para estimar el tiempo de servicio dado un plan de partición de zonas. Después se propone un algoritmo de partición de zona que optimiza de manera iterativa la generación del núcleo del AOI (área de interés) de los repartidores y la asignación de ese AOI.

Resultados offline muestran que este método es mejor que otros en predicción de tiempo y equilibrio de carga de trabajo. Investigaciones en el mundo real proporcionan un 2,2% de mejoría en tiempo de servicio comparado con soluciones más habituales.

Conclusiones:

- Varios artículos combinan métodos o modelos existentes para alcanzar mejores resultados, aunque hay una presencia importante de PCA para el preprocesado.
- Muchos estudios usan datos no reales. Las aplicaciones en casos reales, no parecen ser tan habituales y en profundidad.
- Los estudios más recientes (a partir de 2020 aprox.) empiezan a integrar en sus modelos la sostenibilidad como un punto importante.

Inconvenientes:

- Algunos estudios de 2016 e incluso previos pueden haber perdido validez.
- El uso de métodos complejos y transversales dificulta su implementación, al incrementar su dificultad de uso e interpretación con respecto a otros métodos tradicionales.
- Limitar los estudios a datos no reales, a un tipo de producto o industria en concreto, puede limitar, distorsionar o generalizar los resultados.

Propuestas:

- Establecer un estándar en métodos Big Data, análisis de datos o data mining en el ámbito de logística y supply chain debería ser fundamental, ya que hay algo de confusión a la hora de establecer definiciones, modelos o métodos.
- Se necesitaría una mayor interacción del mundo académico con el mundo de operaciones reales o de negocio.

1.7. Breve sumario de productos obtenidos

Se ha obtenido un algoritmo de previsión de ventas (con un modelo Random Forest) y un algoritmo de previsión de rotura de stock (con un Decission Tree Classifier) con diferentes grados de éxito y precisión. Los dos algoritmos tienen limitaciones en cuanto a los campos (features) utilizados para poder realizar predicciones debido a problemas de precisión.

1.8. Breve descripción de otros capítulos de la memoria

2.Materiales y métodos:

- Metodología: descripción de las tareas realizadas dentro del método CRISP- DM.
- Revisión de Tablas: relación de las tablas de orígen de los datos, variables y tipos de datos.
- Diseño: detalle de las tareas realizadas para establecer en TFM mediante, MSSQL Server, Python y PowerBI.
- Esquema de flujo de datos: visualización de flujo de datos una vez completadas las tareas descritas en “Diseño”.
- Productos obtenidos: sumario de los modelos entrenados y sus métricas de rendimiento.

3.Resultados: descripción breve sobre la aplicación de los modelos sobre nuevos datos para realizar predicciones.

4.Conclusiones y trabajos futuros: relación de las diferentes incidencias y condicionantes que se han encontrado al realizar el TFM, causas, posibles soluciones y futuros estudios a aplicar.

5.Glosario: términos más relevantes dentro del TFM.

6.Bibliografía: material y sitios revisados para la relación del TFM y su estado del arte.

2. Materiales y métodos

METODOLOGÍA

Se ha seguido una metodología CRISP-DM con los siguientes pasos:

- **Comprensión de negocio**: se revisa la información en la descripción del proyecto para entender el funcionamiento del negocio, tipo y periodos de entregas, actores implicados, horarios de apertura, frecuencia de creación de datos, corrección de datos numéricos, etc.

- **Comprensión de datos**: se realiza un análisis de las diferentes tablas y sus campos para obtener información sobre tipos de datos o relaciones existentes. Revisión de datos negativos para ventas y entregas, creación de nuevas columnas de precio de coste, venta, margen, etc. Se añaden tablas que pueden enriquecer la información ya existente, como códigos postales y días festivos. Se modifica algún duplicado para evitar errores en Joins. Además, se realizan diversas visualizaciones para entender la distribución de los valores de los campos.

- **Preparación de datos**: creación de vistas y exportación para tratamiento en Python, aplicando binarización de columnas, tratamiento de outliers, normalización, PCA para poder aplicar los algoritmos correctamente.

- **Modelado**: para la predicción de datos de ventas se aplica un modelo de Regresión Lineal, se añaden Polynomial features, Random Forest, optimización de features, y aplicación de nuevo de Regresión Lineal y Random Forest. Para los datos de rotura inicialmente Regresión Lineal, Random Forest, GradientBoosting, después optimización de features. Finalmente se opta por usar un Decission Tree Classifier con RandomOverSampler y RandomUnderSampler para compensar frecuencia muy alta de valores 0.

- **Evaluación**: para los modelos de regresión se evalúan resultados con MSE, para medir la diferencia media cuadrada entre los valores reales y los predichos, y R2 para revisar proporción de varianza de la variable objetivo explicada por el resto de las variables del modelo. Para los modelos de clasificación se evalúan los resultados principalmente con un “Classification Report” que incluye métricas como precisión, recall, F1-score, Support, Accuracy. También se han utilizado gráficos de dispersión para revisar las diferencias y similitudes entre los valores reales y los predichos.

- **Despliegue**: se exportan las predicciones a MS SQL Server desde donde se pueden obtener datos para estudiar información en PowerBi, por ejemplo.

REVISIÓN DE TABLAS

Es necesario realizar una descripción de los ficheros utilizados, sus campos y relaciones para poder entender el funcionamiento del negocio y posibles estudios de datos que se podría aplicar sobre ellos.

Affiliated_Outlets: información relativa a los establecimientos de la cadena de tiendas.

	Affiliated_Code	**Affiliated_Name**	**Postalcode**	**Engage**
Tipo de Dato	Varchar	Varchar	Integer	Integer
Descripción	Código único para cada establecimiento. Primary Key	Nombre del establecimiento no descriptivo	Código numérico de población. Foreign Key	Valor numérico de 1 a 3, un mayor número es un buen indicativo para el establecimiento
Ejemplo	WjkMAAT	ZUMEA-607	20140	2

	Management_Cluster	**Location**	**Tam_m2**
Tipo de Dato	Integer	Text	Varchar
Descripción	Valor numérico de 0 a 4, un mayor número es un buen indicativo para el establecimiento	Descripción de la zona a la que pertenece el establecimiento	Tamaño del establecimiento
Ejemplo	0	LEISURE	2-5m2

DeliveryDAY: tabla de hechos con los días de entrega en los establecimientos y productos entregados.

	Delivery_Day	**Affiliated_Code**	**Product_Code**
Tipo de Dato	Date	Varchar	Varchar
Descripción	Fecha de entrega en los	Código de establecimiento donde se	Código de producto entregado. Foreign Key

	establecimientos. Va de 2015-03-09 a 2015-10-16	produce la entrega. Foreign Key	
Ejemplo	2015-09-24	WUJAAA5	Brit700

	Delivery_Uds	**Delivery_MONTH**
Tipo de Dato	Integer	Integer
Descripción	Unidades entregadas al establecimiento, sin restricciones	Campo obtenido a partir de Delivery_DAY para agrupar los datos por mes
Ejemplo	8	10

OoSDay: tabla de hecho con los días en los que se produce rotura de stock para un producto y tienda. El establecimiento no tiene unidades suficientes para poner a la venta, lo que supone pérdida de ingresos.

	OoS_Day	**Affiliated_Code**	**Product_Code**	**OoS_MONTH**
Tipo de Dato	Date	Varchar	Varchar	Integer
Descripción	Fecha en la que se produce rotura de stock. Va de 2015-03-09 al 2015-10-04	Código del establecimiento donde se produce la rotura. Foreign Key	Código de producto para el que se produce la rotura. Foreign Key	Campo obtenido a partir de OoS_Day para agrupar los datos por mes
Ejemplo	2015-03-26	WTjtAAH	Natu461	10

Product: información sobre los productos vendidos. Se han añadido las columnas Cost_Price, Sell_Price y Margin.

	Product_Code	**Size**	**Format**
Tipo de Dato	Varchar	Integer	Text
Descripción	Código único de Producto. Primary Key	Indicador de talla de producto	Indicador de formato de producto

Ejemplo	Brit090	142	ASL

	Cost_Price	**Sell_Price**	**Margin**
Tipo de Dato	Decimal	Decimal	Decimal
Descripción	Precio de Coste de producto. Columna añadida con datos aleatorios.	Precio de Venta de producto. Columna con números aleatorios superiores a Cost_Price	Columna creada de la diferencia de Cost_Price y Sell_Price
Ejemplo	8.57	10.25	1.68

RouteDay: día de entrega preestablecido. Fecha por defecto para la entrega de productos en los establecimientos. Si un establecimiento quiere recibir entregas fuera de RouteDay tiene que pagar un sobrecoste. Esto implica que los días de entrega no tienen por qué coincidir con los de ruta.

	Route_DAY	**Affiliated_Code**	**Route_MONTH**
Tipo de Dato	Date	Varchar	Integer
Descripción	Fecha preestablecida de entrega de productos. Va de 2015-03-09 a 2015-12-11	Establecimiento para el que está prevista la ruta. Foreign Key	Campo obtenido a partir de Route_DAY para agrupar los datos por mes
Ejemplo	2015-03-09	WjeGAAT	3

SalesDay: día en el que se produce la venta en un determinado establecimiento y producto.

	Sales_DAY	**Affiliated_Code**	**Product_Code**
Tipo de Dato	Date	Varchar	Varchar
Descripción	Fecha en la que se produce una venta. De 2015-03-09 a 2015-10-04	Establecimiento en el que se produce una venta. Foreign Key	Producto para el que se produce la venta. Foreign Key
Ejemplo	2015-10-04	XCfDAAX	Dome527

	Sales_Uds	Sales_MONTH
Tipo de Dato	Integer	Integer
Descripción	Unidades de venta. Sin restricciones	Campo obtenido a partir de Sales_DAY para agrupar los datos por mes
Ejemplo	3	8

Postal_Codes: información relativa a códigos postales, ciudades y provincias obtenida de fuentes abiertas. Permite enriquecer la información de la tabla Affiliated_Outlets.

	PostalCode	Poblacion	Provincia
Tipo de Dato	Varchar	Text	Text
Descripción	Identificador portal de una población o zona municipal. Primary Key	Nombre del municipio perteneciente al código postal	Provincia a la que pertenece la población o código postal
Ejemplo	2137	Arteaga	Albacete

Holidays: días festivos y domingos por provincia. Es posible cruzar estas fechas con las de las tablas de hechos (DeliveryDAY, OoSDay, RouteDay, SalesDay) para obtener detalles respecto a acciones que se realizan en fechas no laborables y que podrían implicar un coste extra.

	Provincia	Fecha	Festivo
Tipo de Dato	Text	Date	Text
Descripción	Provincia en la que se produce el día festivo/domingo.Foreign key	Fecha de día festivo/domingo	Indicador de día Domingo o Festivo para facilitar cruce con fechas de tablas de hechos
Ejemplo	Alicante/Alacant	2015-12-13	Domingo o Festivo

Como resumen del funcionamiento de la actividad de negocio se aprecian puntos a destacar.

La **diferencia entre RouteDay y DeliveryDay** es importante: mientras el día de ruta describe una entrega preestablecida y sin coste extra, la entrega en DeliveryDay puede indicar también entregas fuera de la fecha preestablecida, suponiendo un coste extra para el afiliado.

Es decir, si Delivery_Day = Route_DAY, entonces no se produce coste extra, sucediendo lo contrario si las fechas no coinciden.

Existencia de **valores negativos Delivery_Uds de la tabla DeliveryDay y en Sales_Uds de la tabla SalesDay**, que se derivan por compensación de entregas erróneas, por un lado, y de hurtos, deterioros, muestras por otro. En el caso de este TFM se ha optado por considerar como outliers los valores inferiores a 0 para evitar sesgos en los modelos de entrenamiento.

Las **fechas de actividad de los datos van de Enero a Octubre de 2015**.

DISEÑO

MS SQL Server: Para realizar la carga de datos inicial se opta inicialmente por implementar SQL Server mediante Azure, pero por problemas de facturación se migra a local.
Se cargan los ficheros de datos, "Affiliated_Outlets", "DeliveryDay", "OoSDay", "Product", "RouteDay", "SalesDay", "Festivos" y "Códigos_postales" desde archivos CSV.

En la revisión de datos en SQL Server para la tabla "Product" se añaden nuevas columnas de "Cost_Price", "Sell_Price" y "Margin" con importes aleatorios.

A continuación, se realizan revisiones de datos: revisión de Nulls, integridad referencial entre tablas, valores negativos, duplicados, precisión de datos.

Finalmente se crean varias vistas para poder exportar los datos necesarios a Jupyter Notebook.
Las vistas creadas y sus correspondientes columnas son:

fact_tables: su objetivo es proporcionar información mensual centrada en unidades de venta, entrega y rotura.

Month	Affiliated_Code	Product_Code	Total_Delivered
Total_Sales	Total_OoS		

all_tables: proporciona información detallada de datos de afiliado, producto, ventas, entregas, rotura, entre otros de manera mensual.

Month	Affiliated_Code	Affiliated_NAME	Postalcode
Poblacion	Provincia	Engage	Management_Cluster
Location	Tam_m2	Product_Code	Size
Format	Cost_price	Sell_price	Margin

Total_Delivered	Total_Sales	Total_OoS	

Delivery_Route: información diaria de las tiendas afiliadas, comparando la fecha de entrega con la de ruta e indica si la entrega se da en festivo o domingo.

Affiliated_Code	Affiliated_Name	Postalcode	Poblacion
Provincia	Engage	Management_Cluster	Location
Tam_m2	Delivery_DAY	Route_DAY	Delivery_out_of_Route
Festivo			

Sales_Rotura: proporciona información diaria sobre establecimientos, unidades de venta y unidades de rotura para poder apreciar si se ha dado rotura de stock y en qué momento.

Sales_DAY	Affiliated_Code	Postalcode	Poblacion
Provincia	Product_Code	Sales_Uds	OoS_DAY
Rotura	Festivo		

Jupyter Notebooks/Python: se divide el análisis, tratamiento de datos y aplicación de modelos en varias partes o ficheros:

TFM 1

- Importación de las vistas all_tables, Delivery_Route y Sales_Rotura.
- Aplicación de un análisis EDA (Exploratory Data Analysis) donde revisamos tipos de datos, nulls, datos únicos, sumario estadístico, visualizaciones de los diferentes campos (incluyendo análisis univariante y análisis de correlación)
- Detección y tratamiento de outliers.

TFM 2

- Se inicia el entrenamiento de modelos para realizar **predicciones sobre unidades de ventas (Total_Sales).**
- Se eligen las variables iniciales a utilizar, features relacionadas con los establecimientos, para predecir las ventas (Total_Sales) según diferentes características de los mismos.
- Binarización para las columnas categóricas y PCA para reducir la dimensionalidad y poder trabajar con ellas.

- Primer entrenamiento de un modelo de regresión lineal, para el que se obtienen resultados de precisión pobres, por lo que se intentan mejoras como aplicación de Polynomial Features.
- Al no obtener mejoras con este cambio ni con un modelo Random Forest, se decide revisar las features utilizadas mediante SelectKBest y un clasificador Random Forest. A partir de aquí seleccionamos solo las features 'Margin','Total_Delivered','Total_OoS','Month_5','Month_10','Engage_1', 'Engage_2','Engage_3','provincia_Madrid','Location_CITY'.
- Con una nueva regresión lineal obtenemos mejores resultados que mejoramos añadiendo de nuevo Polynomial Features.
- Finalmente nos decantamos por un nuevo modelo Random Forest aplicado a estos nuevos campos con una precisión entorno al 76%.

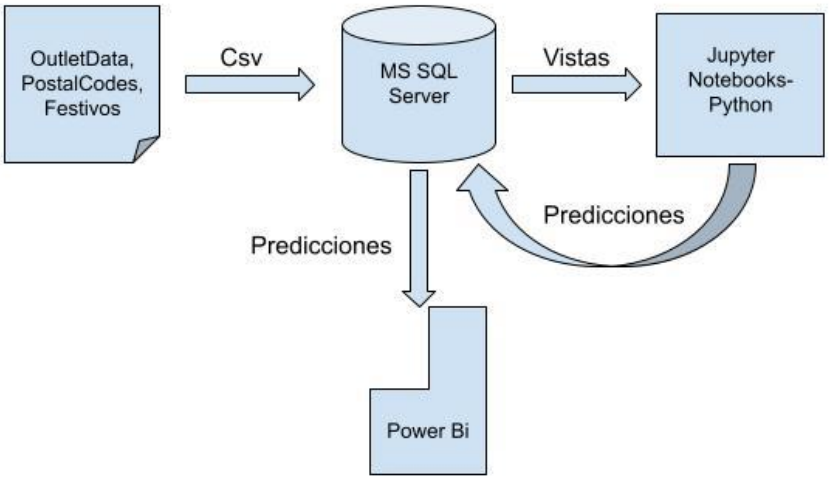
TFM 3

- Se busca **predecir las unidades de rotura de stock (Total_OoS) para los productos**.
- Se seleccionan inicialmente las columnas Cost_Price, Sell_Price y varias features que hacen referencia a los productos, talla y formato con más peso.
- Se aplica clustering y PCA, a partir de los que se inician modelos de Regresión Lineal, Random Forest, GradientBoosting en incluso selección de componentes PCA obteniendo resultados muy pobres.
- Se ajustan más las features utilizadas para los entrenamientos y se aplica un Decision Tree con unos mejores resultados, pero aun no demasiado óptimos.
- Se detecta que el problema para obtener mejoras puede estar en el gran peso del valor de rotura 0 con respecto a valores superiores, por lo que se aplica RandomOverSampler y RandomUnderSampler para intentar corregir ese desequilibrio.
- Se aplica un Decission Tree Classifier mejorando un poco la precisión (56%), pero con problemas para detectar valores altos de rotura. Se aplican varios modelos más complejos como SVM y GaussianNB pero sin poder optimizar predicciones para valores superiores de rotura de stock.

Se utilizarán los modelos obtenidos **Random Forest para predecir Total_Sales** y **Decission Tree Classifier con resampling para Total_OoS**. Se aplican sobre un set que incluye un mes adicional Month_11 que hemos obtenido con regresiones logísticas, para predecir los valores de Total_Sales y Total_OoS de ese nuevo Month_11. Estos resultados se exportan a MS SQL Server.

PowerBi: Se crean visualizaciones importando las predicciones para comparar las unidades predichas de ventas para Month_11 con la evolución de unidades para los meses anteriores.
Se crea también una visualización para comprobar la frecuencia de unidades de rotura predichas con respecto a la frecuencia de unidades de rotura reales. Se puede apreciar que en la predicción hay una proporción menor de valores 0, pero no hay predicciones más allá de 16 uds de rotura, cuando en los datos reales se llega a 68.

ESQUEMA DE FLUJO DE DATOS

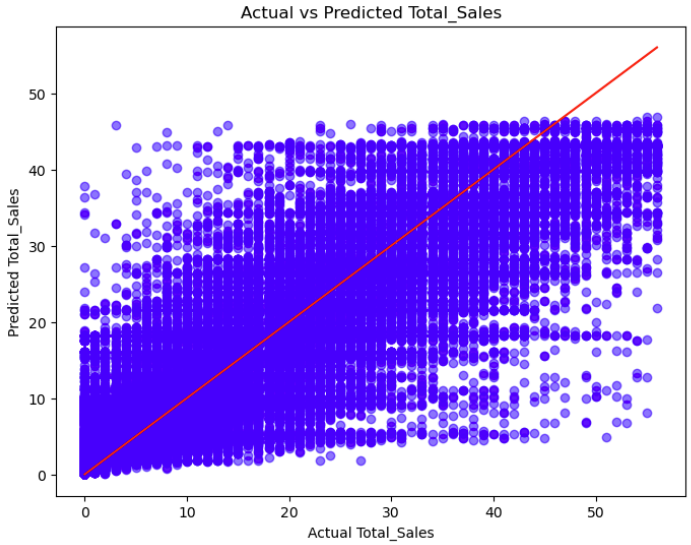


PRODUCTOS OBTENIDOS

Para **unidades de venta (Total_Sales)** se obtienen predicciones con un modelo Random Forest, tras la optimización de features, con un MSE de 32.88 y un R2 del 76%.

```
Random Forest Mean Squared Error (MSE): 32.888752585901585
Random Forest R-squared (R2): 0.7649810584819225
```

En la siguiente visualización se aprecia la distribución de valores reales contra las predicciones:

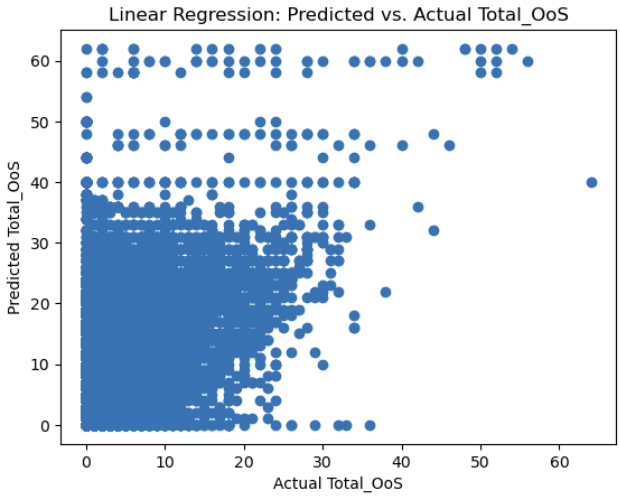


Para **unidades de rotura de stock (Total_OoS)** se obtienen predicciones con un modelo Decission Tree Classifier después de aplicar resampling, con las siguientes métricas:

	precision	recall	f1-score	support
0	0.96	0.62	0.76	63252
1	0.04	0.10	0.06	2102
2	0.03	0.06	0.04	1347
3	0.05	0.19	0.08	1011
4	0.02	0.05	0.02	691
5	0.01	0.04	0.02	476
6	0.02	0.07	0.03	468
7	0.01	0.04	0.02	340
8	0.01	0.05	0.02	311
9	0.01	0.06	0.02	209
10	0.01	0.05	0.02	206
11	0.01	0.03	0.01	153
12	0.00	0.02	0.01	148
13	0.01	0.03	0.01	146
14	0.01	0.04	0.02	156
15	0.03	0.16	0.05	120
16	0.00	0.00	0.00	107
17	0.01	0.03	0.01	98
18	0.01	0.06	0.02	89
19	0.01	0.05	0.01	59
20	0.00	0.01	0.00	80
21	0.01	0.06	0.01	47
22	0.00	0.02	0.01	53
23	0.02	0.15	0.03	39
24	0.00	0.00	0.00	38
25	0.01	0.12	0.02	24
26	0.00	0.00	0.00	39
27	0.01	0.22	0.02	9
28	0.01	0.03	0.02	29
29	0.00	0.00	0.00	6
30	0.03	0.11	0.05	18
31	0.00	0.00	0.00	5
32	0.00	0.00	0.00	10
33	0.00	0.00	0.00	2
34	0.00	0.00	0.00	12
35	0.00	0.00	0.00	0
36	0.00	0.00	0.00	5
37	0.00	0.00	0.00	0
38	0.00	0.00	0.00	2
40	0.00	0.00	0.00	3
42	0.00	0.00	0.00	2
44	0.00	0.00	0.00	2
46	0.07	1.00	0.12	1
48	0.00	0.00	0.00	2
50	0.00	0.00	0.00	3
52	0.00	0.00	0.00	3
54	0.00	0.00	0.00	1
56	0.00	0.00	0.00	1
58	0.00	0.00	0.00	0
60	0.00	0.00	0.00	0
62	0.00	0.00	0.00	0
64	0.00	0.00	0.00	1
accuracy			0.56	71926
macro avg	0.03	0.07	0.03	71926
weighted avg	0.85	0.56	0.67	71926

Como vemos, este modelo tiene una precisión alta para el valor 0, pero mucho más baja para el resto de modelos. A pesar de utilizar RandomOverSampler y RandomUnderSampler la poca prevalencia de valores superiores a 0 ha lastrado el entrenamiento del modelo.

Podemos ver la distribución de los valores reales con respecto a las previsiones, donde se aprecia que las predicciones son algo mejores cuanto más bajos son los valores.



3. Resultados

Se han entrenado modelos que permiten predecir ventas o rotura con diversa precisión.
El modelo de predicción de ventas es mucho más preciso que el de rotura de stock.
Hemos aplicado el modelo entrenado sobre una predicción de un mes ficticio (Month_11), sobre al que su vez hemos predicho ventas y rotura.

4. Conclusiones y trabajos futuros

Para la mejora en predicción de ventas ha sido más determinante la elección de las features de gran influencia sobre la variable objetivo Total_Sales, que la aplicación y mejora de diferentes modelos. Se ha obtenido una precisión aceptable, pero a costa de una reducción sensible de las features empleadas, lo que a su vez reduce la aplicabilidad del modelo en casos reales. Sería necesario llegar a un mayor equilibrio entre la cantidad de features útiles y la precisión del modelo. Sin embargo, en el caso estudiado la precisión se ha visto reducida drásticamente con mínimos cambios de features.

Para la predicción de rotura de stock, se ha detectado una gran influencia de los valores 0 sobre el entrenamiento de los modelos, dificultando la predicción de valores diferentes a 0. A pesar de utilizar técnicas de corrección, este problema solo se ha mitigado parcialmente
Tal vez disponer de una muestra más grande de valores diferentes a 0 o limitar más los valores 0 en las muestras pueda mejorar las predicciones de valores mayores a 0, a riesgo de perder algo de precisión.

Por tanto, el objetivo de obtención de unos algoritmos sólidos se ha alcanzado parcialmente, ya que los modelos empleados han tenido que ser limitados, haciendolos menos flexibles en su empleabilidad.

Los estudios se han limitado a datos obtenidos de una sola vista (all_tables) de todas las vistas obtenidas inicialmente en MSSQL Server. Esto es así por un lado por falta de tiempo para intentar aplicar otros estudios, y además porque se ha limitado el estudio a un tipo de vista que ofrecía datos mensuales, no diarios, ya que se producían problemas de cálculo para modelos con datasets muy grandes, lo que obligaría a tratar con muestras reducidas de los datos.

La planificación de tareas se ha cumplido con bastante precisión hasta el momento de aplicación, optimización y entrenamiento de modelos, donde se ha tenido que invertir un tiempo mayor al esperado para poder obtener unos resultados aceptables. Se han entrenado muchos más modelos de los apreciables en el código de Python, pero se ha resumido ese código para evitar confusión en favor de los modelos más favorables.

Además de lo mencionado anteriormente podría ser interesante poder aplicar predicciones sobre columnas nuevas referentes a las diferencias en ruta y entrega, ya que estas diferencias suponen mayores costes de entrega, menos eficiencia en transporte y más impacto medioambiental.

5. Glosario

Análisis de datos, Ciencia de datos, Supply Chain, Ventas, Stock.

6. Bibliografía

- **"Big data analytics in logistics and supply chain management: Certain investigations for research and applications"** Gang Wang, Angappa Gunasekaran, Eric W.T. Ngai, Thanos Papadopoulos. Version of Record 31 March 2016.

- **"Big data analytics in supply chain management: A state-of-the-art literature review"**. Truong Nguyen, Li ZHOU, Virginia Spiegler, Petros Ieromonachou, Yong Lin. Version of Record 6 July 2018.

- **"Dimensions of Data Analytics in Supply Chain Management: Objectives, Indicators and Data Questions".** Patrick Brandtner ,Chibuzor Udokwu, Farzaneh Darbanian, Taha Falatouri. January 2021.

- **"Data Preprocessing in Supply Chain Management Analytics - A Review of Methods, the Operations They Fulfill, and the Tasks They Accomplish"**. Tobechi Obinwanne, Zimmermann Robert, Udokwu Chibuzor, Brandtner Patrick. January 13–15, 2023.

- **"RFM model customer segmentation based on hierarchical approach using FCA"**. Chongkolnee Rungruang, Pakwan Riyapan, Arthit Intarasit, Khanchit Chuarkham, Jirapond Muangprathub. Version of Record 18 September 2023.

- **"Towards Equitable Assignment: Data-Driven Delivery Zone Partition at Last-mile Logistics"**. Baoshen Guo,Shuai Wang, Haotian Wang, Yunhuai Liu, Fanshuo Kong, Desheng Zhang, Tian He. August 6–10, 2023.

- **"A Data Analytics Mindset with CRISP-DM".** Richard O'Hara, Lisa S. Haylon, Douglas M. Boyle. February 01, 2023.

- **"How to formulate a research strategy?"** Ashni Walia, Priya Chetty. February 21, 2020

- **SpringerLink**

- **ScienceDirect**

- **ACM Digital Library**

7. Anexos

A continuación, se adjunta el código relativo al TFM.

MS SQL Server

Se realiza la carga de datos con ficheros CSV. Después se procede a realizar diferentes cambios en las tablas.

Para la tabla **Product** se añaden las **nuevas columnas Cost_Price y Sell_Price**:

```
ALTER TABLE dbo.Product
ADD Cost_Price DECIMAL(5,2),
    Sell_Price DECIMAL(5,2);
```

Se añaden valores para estas nuevas columnas:

```
UPDATE [dbo].[Product]
SET Cost_Price =
       CASE
              WHEN Product_Code = 'Brit090' THEN '8.57'
              …                              …
              WHEN Product_Code = 'Trad610' THEN '5.90'

       ELSE ''
 END,

Sell_Price =
       CASE
              WHEN Product_Code = 'Brit090' THEN '10.25'
              …
              WHEN Product_Code = 'Trad610' THEN '7'
              ELSE ''
                END;
```

Para la tabla Product se crea también la columna calculada Margin:

```
ALTER TABLE [dbo].[Product]
ADD Margin DECIMAL(5, 2);

UPDATE [dbo].[Product]
SET Margin = Sell_Price - Cost_Price;
```

Se crean **columnas de mes** para las diferentes **tablas de hechos**:
Delivery_MONTH a partir de **DeliveryDay**:

```
ALTER TABLE dbo.DeliveryDay
ADD Delivery_MONTH int;

UPDATE dbo.DeliveryDay
SET Delivery_MONTH =  MONTH([Delivery_DAY]);
```

OoS_MONTH a partir de OoSDAY, **Sales_MONTH** a partir de SalesDay, Route_MONTH a partir de RouteDay.

Se realiza una revisión de datos de las diferentes tablas.

Completeness Check: revisión de completitud de las diferentes tablas. Se aprecia que no hay NULL en ninguna de las tablas. Como ejemplo de código utilizado, se emplea el siguiente:

```
SELECT *
FROM Affiliated_Outlets
WHERE Affiliated_Code IS NULL
       OR Affiliated_NAME IS NULL
       OR POSTALCODE IS NULL
       OR Engage IS NULL
       OR Management_Cluster IS NULL
       OR Location IS NULL
       OR Tam_m2 IS NULL;
```

Revisión de integridad referencial entre tablas: se intenta identificar si hay claves foráneas que no tengan correspondencia con las claves principales y que por lo tanto impiden la creación de Joins.

Por ejemplo, se revisa si en DeliveryDay hay registros de Affiliated_Code que no están en la tabla Affiliated_Outlets.

```
SELECT *
FROM DeliveryDay d

LEFT JOIN Affiliated_Outlets a ON d.Affiliated_Code = a.Affiliated_Code
WHERE a.Affiliated_Code IS NULL;
```

Se hace la misma comprobación para correspondencias entre DeliveryDay y el resto de tablas con campos coincidentes: Product, SalesDay.

Además, se comprueban los Join del resto de tablas. Por ejemplo, en el caso de OoSDay con Affiliated_Code- Affiliated_Outlets y Product_Code- Product:

```
SELECT *
FROM OoSDay o

LEFT JOIN Affiliated_Outlets a ON o.Affiliated_Code = a.Affiliated_Code
WHERE a.Affiliated_Code IS NULL;
```

En el caso de RouteDay se comprueba Affiliated_Code- Affiliated_Outlets:

```
SELECT *
FROM RouteDay r

LEFT JOIN Affiliated_Outlets a ON r.Affiliated_Code = a.Affiliated_Code
WHERE a.Affiliated_Code IS NULL;
```

Para la tabla SalesDay también revisamos las correspondencias con Affiliated_Code- Affiliated_Outlets y Product_Code- Product.

```
SELECT *
FROM SalesDay s

LEFT JOIN Affiliated_Outlets a ON s.Affiliated_Code = a.Affiliated_Code
WHERE a.Affiliated_Code IS NULL;
```

Consistencia de datos: verificación de cumplimiento de reglas de consistencia.
Se revisa si hay unidades negativas en ventas o entregas, que como hemos visto sí sucede:

```
SELECT *
FROM DeliveryDay
WHERE Delivery_Uds < 0;
```

```
SELECT *
FROM SalesDay
WHERE Sales_Uds <0;
```

No se realizan cambios con respecto a las unidades negativas en MS SQL Server.

Valores duplicados: revisamos valores duplicados para las tablas que hacen referencia a producto y tiendas. Primera revisión con la tabla Product:

```
SELECT  Product_Code,
       COUNT(*)
 FROM dbo.Product
GROUP BY Product_Code
HAVING COUNT(*) > 1
```

Se encontró un duplicado para el código de producto Natu122. Esto impedía establecer Product_Code como Primary Key para la tabla, y estos valores deberían ser únicos para evitar errores.

Después de revisar información de este producto, se vio que **el formato para los dos códigos Natu122 es distinto**, **por lo que se interpreta que son códigos diferentes**. Por tanto, en lugar de eliminar uno de los registros**, se ha cambiado el Product_Code a Natu123 para uno de ellos**:

```
UPDATE Product
SET Product_Code = 'Natu123'
WHERE Product_Code = 'Natu122'
       AND Format = 'ASL';
```

La table Affiliated_Outlets no contiene duplicados.

Precisión de datos y validez: se revisan los dominios de los datos, para validar si se encuentran entre los rangos o dominios establecidos. Así se verifica si hay columnas con grandes outliers o si las fechas están dentro de rangos razonables.

Se ha revisado si la columna Tam_m2 de Affiliated_Outlets tiene outliers para alguna de sus categorías. No se encontraron categorías que destacaran demasiado, pero sí muchos N.D.:

```
SELECT
       Tam_m2,
       COUNT (*) AS total_count_m2
FROM dbo.Affiliated_Outlets
GROUP BY Tam_m2
ORDER BY total_count_m2 DESC
```

En los códigos postales, no se aprecian valores extremos tampoco:

```
SELECT
       POSTALCODE,
       COUNT (*) AS total_postal_code
FROM [dbo].[Affiliated_Outlets]
GROUP BY POSTALCODE
ORDER BY total_postal_code DESC
```

Consistencia de fechas: se verifica si hay alguna fecha en las tablas de hechos anteriores a 2015 o posteriores a 2023 (aunque en realidad no hay datos posteriores a 2015).

```
SELECT *
FROM DeliveryDay
WHERE Delivery_DAY < '2015-01-01'
OR Delivery_DAY > '2023-12-31'
```

Esta verificación se aplica a DeliveryDay, OoSDay, RouteDay y SalesDay, y no se encuentra ningún registro fuera del rango de fechas establecidas.

Creación de vistas: se crean vistas para exportar datos a Jupyter Notebook.

Vista **fact_tables**:

```
CREATE VIEW fact_tables AS

WITH DeliveredMonthly AS (
	SELECT
		MONTH(DeliveryDay.Delivery_DAY) AS Delivery_Month,
		DeliveryDay.Affiliated_Code,
		DeliveryDay.Product_Code,
		SUM(DeliveryDay.Delivery_Uds) AS Total_Delivered
	FROM
		dbo.DeliveryDay

	GROUP BY
		MONTH(DeliveryDay.Delivery_DAY),
		DeliveryDay.Affiliated_Code,
		DeliveryDay.Product_Code
	),

SoldMonthly AS (
	SELECT
		MONTH(SalesDay.Sales_DAY) AS Sales_Month,
		SalesDay.Affiliated_Code,
		SalesDay.Product_Code,
		SUM(SalesDay.Sales_Uds) AS Total_Sales
	FROM
		dbo.SalesDay

	GROUP BY
		MONTH(SalesDay.Sales_DAY),
		SalesDay.Affiliated_Code,
		SalesDay.Product_Code
	),

OoSMonthly AS (
	SELECT
		MONTH (OoSDay.OoS_Day) AS Oos_Month,
		OoSDay.Affiliated_Code,
		OoSDay.Product_Code,
		COUNT (*) AS OoS_Times

	FROM
		dbo.OoSDay

	GROUP BY
		Affiliated_Code,
		Product_Code,
		MONTH (OoS_Day)
	)
```

```
SELECT
	COALESCE(DeliveredMonthly.Delivery_Month, SoldMonthly.Sales_Month,
	OoSMonthly.Oos_Month) AS Month,
	COALESCE(DeliveredMonthly.Affiliated_Code, SoldMonthly.Affiliated_Code,
	OoSMonthly.Affiliated_Code) AS Affiliated_Code,
	COALESCE(DeliveredMonthly.Product_Code, SoldMonthly.Product_Code,
	OoSMonthly.Product_Code) AS Product_Code,
	COALESCE(Total_Delivered, 0) AS Total_Delivered,
	COALESCE(Total_Sales, 0) AS Total_Sales,
	COALESCE(OoS_Times,0) AS Total_OoS

FROM
	DeliveredMonthly

FULL JOIN SoldMonthly
ON DeliveredMonthly.Delivery_Month = SoldMonthly.Sales_Month
AND DeliveredMonthly.Affiliated_Code = SoldMonthly.Affiliated_Code
AND DeliveredMonthly.Product_Code = SoldMonthly.Product_Code

FULL JOIN OoSMonthly
ON DeliveredMonthly.Delivery_Month = OoSMonthly.Oos_Month
AND DeliveredMonthly.Affiliated_Code = OoSMonthly.Affiliated_Code
AND DeliveredMonthly.Product_Code = OoSMonthly.Product_Code
```

Vista **all_tables:**

```
CREATE VIEW all_tables AS

SELECT
	Month,
	fact_tables.Affiliated_Code,
	Affiliated_NAME,
	Affiliated_Outlets.POSTALCODE,
	poblacion,
	provincia,
	Engage,
	Management_Cluster,
	Location,
	Tam_m2,
	fact_tables.Product_Code,
	SIZE,
	Format,
	Cost_price,
	Sell_price,
	Margin,
	Total_Delivered,
	Total_Sales,
	Total_OoS

FROM fact_tables

LEFT JOIN Affiliated_Outlets
	ON fact_tables.Affiliated_Code= Affiliated_Outlets.Affiliated_Code
```

```
LEFT JOIN Product
       ON fact_tables.Product_Code= Product.Product_Code

LEFT JOIN Postal_Codes
       ON Affiliated_Outlets.POSTALCODE = Postal_Codes.PostalCode
```

Vista **Delivery_Route**:

```
CREATE VIEW Delivery_Route AS

SELECT
       DeliveryDay.Affiliated_Code,
       Affiliated_Name,
       Affiliated_Outlets.POSTALCODE,
       poblacion,
       Postal_Codes.provincia,
       Engage,
       Management_Cluster,
       Location,
       Tam_m2,
       Delivery_DAY,
       Route_DAY,
CASE
       WHEN DeliveryDay.Delivery_DAY <> RouteDay.Route_DAY OR
       (DeliveryDay.Delivery_DAY IS NULL AND RouteDay.Route_DAY IS NOT NULL) OR
       (DeliveryDay.Delivery_DAY IS NOT NULL AND RouteDay.Route_DAY IS NULL)
THEN 'Yes'
ELSE 'No'
END AS Delivery_out_of_Route,
ISNULL(Festivo, 'Laborable') AS Festivo


FROM dbo.DeliveryDay

LEFT JOIN dbo.RouteDay
       ON DeliveryDay.Affiliated_Code = RouteDay.Affiliated_Code
       AND DeliveryDay.Delivery_DAY = RouteDay.Route_DAY

LEFT JOIN dbo.Affiliated_Outlets
       ON DeliveryDay.Affiliated_Code = Affiliated_Outlets.Affiliated_Code

LEFT JOIN dbo.Postal_Codes
       ON Affiliated_Outlets.POSTALCODE = Postal_Codes.PostalCode

LEFT JOIN dbo.Holidays
       ON Postal_Codes.provincia = Holidays.Provincia
       AND DeliveryDay.Delivery_DAY = Holidays.Fecha
```

Vista **Sales_Rotura**:

```
CREATE VIEW Sales_Rotura AS

SELECT
	SalesDay.Sales_DAY,
	SalesDay.Affiliated_Code,
	Affiliated_Outlets.POSTALCODE,
	Postal_Codes.poblacion,
	Postal_Codes.provincia,
	SalesDay.Product_Code,
	SalesDay.Sales_Uds,
	OoS_DAY,
CASE
	WHEN OoSDay.OoS_DAY IS NOT NULL THEN 'Yes'
	ELSE 'No'
END AS Rotura,
ISNULL(Festivo, 'Laborable') AS Festivo
FROM dbo.SalesDay

LEFT JOIN OoSDay ON SalesDay.Sales_DAY = OoSDay.OoS_DAY
	AND SalesDay.Affiliated_Code = OoSDay.Affiliated_Code
	AND SalesDay.Product_Code = OoSDay.Product_Code

LEFT JOIN dbo.Affiliated_Outlets ON SalesDay.Affiliated_Code =
Affiliated_Outlets.Affiliated_Code

LEFT JOIN dbo.Postal_Codes ON Affiliated_Outlets.POSTALCODE = Postal_Codes.PostalCode

LEFT JOIN dbo.Holidays ON Postal_Codes.provincia = Holidays.Provincia
AND SalesDay.Sales_DAY = Holidays.Fecha
```

Python- Jupyter Notebook

Importación y carga de vistas SQL.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import pyodbc

server = 'DESKTOP-VUJ9ETK\SQLEXPRESS'
database = 'TFM_EnriqueRocho'
conn = pyodbc.connect('DRIVER={SQL Server};SERVER=' + server + ';DATABASE=' + database +
';Trusted_Connection=yes;')

query1 = 'SELECT * FROM all_tables'
query2 = 'SELECT * FROM Delivery_Route'
query3 = 'SELECT * FROM Sales_Rotura'
```

```
df_all_tables = pd.read_sql(query1, conn)
df_Delivery_Route = pd.read_sql(query2, conn)
df_Sales_Rotura = pd.read_sql(query3, conn)

conn.close()

print(df_all_tables.head())
print(df_Delivery_Route.head())
print(df_Sales_Rotura.head())
```

Análisis EDA (Exploratory Data Analysis)

Revisión de tipo de datos y nulls

Revisión inicial de información de la tabla df_all_tables

```
print("df_all_tables:")
print(df_all_tables.info())
```

Uso de fillna() con un valor por defecto:

```
df_all_tables['provincia'].fillna('Desconocido', inplace=True)
df_all_tables['poblacion'].fillna('Desconocido', inplace=True)

df_all_tables.isnull().sum()
```

Revisión de información para df_Delivery_Route

```
print("df_Delivery_Route:")
print(df_Delivery_Route.info())
```

Nulls para población y provincia.

```
df_Delivery_Route['provincia'].fillna('Desconocido', inplace=True)
df_Delivery_Route['poblacion'].fillna('Desconocido', inplace=True)
```

Cambio para las columnas de fechas con tipos de dato incorrecto:

```
df_Delivery_Route['Delivery_DAY'] = pd.to_datetime(df_Delivery_Route['Delivery_DAY'])
df_Delivery_Route['Route_DAY'] = pd.to_datetime(df_Delivery_Route['Route_DAY'])
```

Cambio de data type para Sales_DAY y OoS_DAY

```
df_Sales_Rotura['Sales_DAY'] = pd.to_datetime(df_Sales_Rotura['Sales_DAY'])
df_Sales_Rotura['OoS_DAY'] = pd.to_datetime(df_Sales_Rotura['OoS_DAY'])
```

Resolución de Nulls para df_Sales_Rotura

```
df_Sales_Rotura['provincia'].fillna('Desconocido', inplace=True)
df_Sales_Rotura['poblacion'].fillna('Desconocido', inplace=True)
```

Valores únicos

Se revisan los valores únicos en las columnas categóricas:

```
for col in df_all_tables.select_dtypes(include='object').columns:
    print(col, df_all_tables[col].nunique())

for col in df_Delivery_Route.select_dtypes(include='object').columns:
```

```
    print(col, df_Delivery_Route[col].nunique())

for col in df_Sales_Rotura.select_dtypes(include='object').columns:
    print(col, df_Sales_Rotura[col].nunique())
```

Sumario Estadístico

Sumario estadístico para las tablas.

```
print("Sumario estadístico de df_all_tables:")
print(df_all_tables[['Engage','Management_Cluster','Cost_price', 'Sell_price', 'Margin',
'Total_Delivered', 'Total_Sales','Total_OoS']].describe())

print("Sumario estadístico de df_Delivery_Route:")
print(df_Delivery_Route[['Engage','Management_Cluster']].describe())

pd.options.display.float_format = '{:.2f}'.format
print("Sumario estadístico de df_Sales_Rotura:")
print(df_Sales_Rotura['Sales_Uds'].describe())
```

Visualizaciones realizadas para el EDA para df_all_tables:

Visualización de Count por Month

```
plt.figure(figsize=(10, 6))
sns.countplot(x='Month', data=df_all_tables,
order=df_all_tables['Month'].value_counts().index)
plt.xticks(rotation=90)
plt.title('Count of Month')
plt.show()
```

Count por Provincia

```
plt.figure(figsize=(10, 6))
sns.countplot(x='provincia', data=df_all_tables,
order=df_all_tables['provincia'].value_counts().index)
plt.xticks(rotation=90)
plt.title('Count of Provincia')
plt.show()
```

Engage

```
plt.figure(figsize=(10, 6))
sns.countplot(x='Engage', data=df_all_tables,
order=df_all_tables['Engage'].value_counts().index)
plt.xticks(rotation=0)
plt.title('Count of Engage')
plt.show()
```

Management Cluster

```
plt.figure(figsize=(10, 6))
sns.countplot(x='Management_Cluster', data=df_all_tables,
order=df_all_tables['Management_Cluster'].value_counts().index)
plt.xticks(rotation=0)
plt.title('Count of Management_Cluster')
plt.show()
```

Location

```
plt.figure(figsize=(10, 6))
sns.countplot(x='Location', data=df_all_tables,
order=df_all_tables['Location'].value_counts().index)
plt.xticks(rotation=0)
plt.title('Count of Location')
plt.show()
```

Tam_m2

```
plt.figure(figsize=(10, 6))
sns.countplot(x='Tam_m2', data=df_all_tables,
order=df_all_tables['Tam_m2'].value_counts().index)
plt.xticks(rotation=0)
plt.title('Count of Tam_m2')
plt.show()
```

Product_Code

```
plt.figure(figsize=(10, 6))
sns.countplot(x='Product_Code', data=df_all_tables,
order=df_all_tables['Product_Code'].value_counts().index)
plt.xticks(rotation=90)
plt.title('Count of Product_Code')
plt.show()
```

Size

```
plt.figure(figsize=(10, 6))
sns.countplot(x='SIZE', data=df_all_tables, order=df_all_tables['SIZE'].value_counts().index)
plt.xticks(rotation=90)
plt.title('Count of SIZE')
plt.show()
```

Format

```
plt.figure(figsize=(10, 6))
sns.countplot(x='Format', data=df_all_tables,
order=df_all_tables['Format'].value_counts().index)
plt.xticks(rotation=90)
plt.title('Count of Format')
plt.show()
```

Total_OoS

```
plt.figure(figsize=(10, 6))
sns.countplot(x='Total_OoS', data=df_all_tables,
order=df_all_tables['Total_OoS'].value_counts().index)
plt.xticks(rotation=90)
plt.title('Count of Total_OoS')
plt.show()
```

Análisis Univariante

```
df_all_tables[['Cost_price','Sell_price','Margin']].hist(figsize=(20, 20))
plt.figure(figsize=(5, 5))
plt.hist(df_all_tables['Total_Delivered'], bins=20)
plt.xlim(-200, 300)
plt.xlabel('Delivered')
plt.ylabel('Freq')
plt.title('Histogram Total_Delivered')
plt.show()
```

```
plt.figure(figsize=(5, 5))
plt.hist(df_all_tables['Total_Sales'], bins=20)
plt.xlim(-200, 300)
plt.xlabel('Sales')
plt.ylabel('Freq')
plt.title('Histogram Total_Sales')
plt.show()
```

Análisis de Correlación df_all_tables

```
correlation_matrix_1 = df_all_tables[['Engage','Management_Cluster','SIZE',
'Cost_price','Sell_price','Margin',
'Total_Delivered','Total_Sales', 'Total_OoS' ]].corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix_1, annot=True, cmap='coolwarm')
plt.show()
```

Outliers

Detección de Outliers df_all_tables

```
sns.boxplot(x=df_all_tables['Cost_price'])
plt.show()

sns.boxplot(x=df_all_tables['Sell_price'])
plt.show()

sns.boxplot(x=df_all_tables['Margin'])
plt.show()

sns.boxplot(x=df_all_tables['Total_Delivered'])
plt.show()
```

Método IQR para revision de outliers en detalle:

```
Q1 = df_all_tables['Total_Delivered'].quantile(0.25)
Q3 = df_all_tables['Total_Delivered'].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 2 * IQR
upper_bound = Q3 + 2 * IQR

outliers = df_all_tables[(df_all_tables['Total_Delivered'] < lower_bound) |
(df_all_tables['Total_Delivered'] > upper_bound)]
print(outliers)
```

Método Z-Score:

```
from scipy import stats
threshold = 2
outliers2 = df_all_tables[np.abs(stats.zscore(df_all_tables['Total_Delivered'])) > threshold]
print(outliers2)
```

Outliers df_all_tables/Total_Sales

```
sns.boxplot(x=df_all_tables['Total_Sales'])
plt.show()
```

IQR

```
Q1 = df_all_tables['Total_Sales'].quantile(0.25)
Q3 = df_all_tables['Total_Sales'].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 2 * IQR
upper_bound = Q3 + 2 * IQR

outliers3 = df_all_tables[(df_all_tables['Total_Sales'] < lower_bound) |
(df_all_tables['Total_Sales'] > upper_bound)]
print(outliers3)
```

Método Z-Score:

```
threshold = 2
outliers4 = df_all_tables[np.abs(stats.zscore(df_all_tables['Total_Sales'])) > threshold]
print(outliers4)
```

Outliers df_all_tables/Total_OoS

```
sns.boxplot(x=df_all_tables['Total_OoS'])
plt.show()
```

Limpieza de Outliers

```
outliers_indices = outliers2.index.union(outliers4.index)
df_all_tables_no_outliers = df_all_tables.drop(outliers_indices)
```

Limpieza de valores inferiores a 0

```
df_all_tables_no_outliers  = df_all_tables_no_outliers[(df_all_tables_no_outliers
['Total_Delivered'] >= 0)
                                                      & (df_all_tables_no_outliers
['Total_Sales'] >= 0)]
```

PREDICCIÓN Total_Sales

Preparación de datos para los algoritmos

Elección de variables: Para intentar obtener las ventas totales (Total_Sales) en función de características de tiendas, se eliminan columnas no relativas a esas características:

```
df_affiliated_sales = df_all_tables_no_outliers.copy()

df_affiliated_sales =
df_affiliated_sales.drop(['Month','Affiliated_Code','Affiliated_NAME','POSTALCODE',
'poblacion','Product_Code','SIZE', 'Format','Cost_price',
'Sell_price','Margin','Total_Delivered','Total_OoS','Total_Sales'], axis=1)
```

Dummy variables: conversion de columnas categóricas a numéricas bivariables:

```
categorical_columns = ['Engage', 'Management_Cluster', 'provincia','Location', 'Tam_m2']
df_affiliated_sales_cat = pd.get_dummies(df_affiliated_sales, columns=categorical_columns)
```

PCA: se emplea para reducir dimensionalidad de variables.

Applicamos PCA inicialmente con un número de componentes muy alto.

```
from sklearn.decomposition import PCA
pca = PCA(n_components=66)
pca.fit(df_affiliated_sales_cat)
transformed_data = pca.transform(df_affiliated_sales_cat)
```

Revisión de componentes óptimo:

```
import matplotlib.pyplot as plt
pca = PCA()
pca.fit(transformed_data)
```

Calculamos la cumulative explained variance.

```
cumulative_variance = np.cumsum(pca.explained_variance_ratio_)
```

Plot

```
plt.figure(figsize=(8, 6))
plt.plot(cumulative_variance)
plt.xlabel('Number of Components')
plt.ylabel('Cumulative Explained Variance')
plt.title('Explained Variance vs. Number of Components')
plt.grid(True)
plt.show()
```

Aplicamos PCA de Nuevo con 20 componentes

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

pca = PCA(n_components=20)
pca.fit(df_affiliated_sales_cat)
pca_features = pca.transform(df_affiliated_sales_cat)
```

Regresión Lineal Total_Sales

```
X_train, X_test, y_train, y_test = train_test_split(pca_features,
df_all_tables_no_outliers['Total_Sales'], test_size=0.2, random_state=42)
regressor = LinearRegression()
regressor.fit(X_train, y_train)
y_pred = regressor.predict(X_test)
```

Calculamos Mean Squared Error (MSE) y R2 para medir la precisión del modelo

```
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
r2 = r2_score(y_test, y_pred)
print(f"R-squared (R2): {r2}")
```

Plot de resultados

```
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.xlabel('Actual Total Sales')
plt.ylabel('Predicted Total Sales')
plt.title('Actual vs Predicted Total Sales')
```

```
plt.show()
```

Cross-Validation para intentar mejorar el modelo

```
from sklearn.model_selection import cross_val_score, KFold
X = transformed_data
y = df_all_tables_no_outliers['Total_Sales']
model = LinearRegression()
k_folds = 5
kf = KFold(n_splits=k_folds, shuffle=True, random_state=42)

mse_scores = -cross_val_score(model, X, y, cv=kf, scoring='neg_mean_squared_error')
r2_scores = cross_val_score(model, X, y, cv=kf, scoring='r2')

mean_mse = np.mean(mse_scores)
std_mse = np.std(mse_scores)
mean_r2 = np.mean(r2_scores)
std_r2 = np.std(r2_scores)

print(f"Cross-Validation - Mean Squared Error (MSE): {mean_mse} ± {std_mse}")
print(f"Cross-Validation - R-squared (R2): {mean_r2} ± {std_r2}")
```

Baseline: Comparación de valores con Baseline, para establecer un punto de referencia.

```
mean_sales = np.mean(y)
mean_sales_predictions = np.full_like(y_test, mean_sales)

baseline_mse = mean_squared_error(y_test, mean_sales_predictions)
print("Baseline MSE (Predicting Mean):", baseline_mse)
```

Feature Engineering: Polynomial Features. Intento de mejora de la regresión con polynomial features

```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import make_pipeline
from sklearn.metrics import mean_squared_error, r2_score

# Definimos los grados de polynomial features
degree = 2

poly_reg = make_pipeline(PolynomialFeatures(degree), LinearRegression())
poly_reg.fit(X_train, y_train)
predictions = poly_reg.predict(X_test)

poly_mse = mean_squared_error(y_test, predictions)
poly_r2 = r2_score(y_test, predictions)

print(f"Polynomial Regression - Mean Squared Error (MSE): {poly_mse}")
print(f"Polynomial Regression - R-squared (R2): {poly_r2}")
```

Random Forest Total_Sales

```
from sklearn.ensemble import RandomForestRegressor

rf_model = RandomForestRegressor(n_estimators=25, random_state=42)
rf_model.fit(X_train, y_train)
rf_predictions = rf_model.predict(X_test)

rf_mse = mean_squared_error(y_test, rf_predictions)
rf_r2 = r2_score(y_test, rf_predictions)

print(f"Random Forest - Mean Squared Error (MSE): {rf_mse}")
print(f"Random Forest - R-squared (R2): {rf_r2}")

plt.figure(figsize=(8, 6))
plt.scatter(y_test, rf_predictions, alpha=0.5)
plt.xlabel('Actual Total Sales')
plt.ylabel('Predicted Total Sales')
plt.title('Actual vs Predicted Total Sales (Random Forest)')
plt.show()
```

Revisión de features para mejorar resultados del modelo

```
df_affiliated_sales2 = df_all_tables_no_outliers.copy()

df_affiliated_sales2 = df_affiliated_sales2.drop(['Affiliated_NAME','POSTALCODE', 'poblacion',
'Product_Code','SIZE', 'Format'], axis=1)

categorical_columns2 = ['Month', 'Affiliated_Code', 'provincia', 'Engage',
'Management_Cluster', 'Location', 'Tam_m2']
df_affiliated_sales_cat2 = pd.get_dummies(df_affiliated_sales2, columns=categorical_columns2)
```

SelectKBest para revisión de columnas con más peso en Total_Sales

```
from sklearn.feature_selection import SelectKBest, f_regression

X2 = df_affiliated_sales_cat2.drop(columns=['Total_Sales'])  # Features
y2 = df_affiliated_sales_cat2['Total_Sales']  # Variable Objetivo

selector = SelectKBest(score_func=f_regression, k=10)
X_new = selector.fit_transform(X2, y2)
selected_columns = X2.columns[selector.get_support()]
selected_features_df = pd.DataFrame(X_new, columns=selected_columns)

print("Selected Features:")
print(selected_features_df.head())
```

Random Forest para revisión de features importantes

```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(n_estimators=10, random_state=42)
rf.fit(X2, y2)

feature_importances = pd.Series(rf.feature_importances_, index=X2.columns)
feature_importances = feature_importances.sort_values(ascending=False)
```

```
# Plot
plt.figure(figsize=(10, 8))
feature_importances.head(10).plot(kind='barh')
plt.title('Top 10 Feature Importances')
plt.xlabel('Relative Importance')
plt.ylabel('Features')
plt.show()
```

Selección de nuevas features

```
specified_columns = ['Margin','Total_Delivered','Total_OoS','Month_5','Month_10','Engage_1',
                     'Engage_2','Engage_3','provincia_Madrid','Location_CITY',
                     ]
df_affiliated_sales_selected = df_affiliated_sales_cat2[specified_columns].copy()
```

Normalizamos las columnas numéricas que utilizaremos:

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
df_affiliated_sales_selected[['Margin','Total_Delivered','Total_OoS']] =
scaler.fit_transform(df_affiliated_sales_selected[['Margin','Total_Delivered','Total_OoS']])
```

PCA inicial

```
pca2 = PCA(n_components=10)
pca_result2 = pca2.fit_transform(df_affiliated_sales_selected)
pca2 = PCA()
pca2.fit(pca_result2)
```

Varianza acumulada

```
cumulative_variance = np.cumsum(pca2.explained_variance_ratio_)

# Plot
plt.figure(figsize=(8, 6))
plt.plot(cumulative_variance)
plt.xlabel('Number of Components')
plt.ylabel('Cumulative Explained Variance')
plt.title('Explained Variance vs. Number of Components')
plt.grid(True)
plt.show()
```

Aplicamos PCA para 10, ya que se ha comprobado una bajada de precisión muy acusada con unos 7.

```
pca2 = PCA(n_components=10)
pca_result2 = pca2.fit_transform(df_affiliated_sales_selected)
```

Nueva regression Lineal para Total_Sales con reducción de features

```
y2 = df_all_tables_no_outliers['Total_Sales']
X_train2, X_test2, y_train2, y_test2 = train_test_split(pca_result2, y2, test_size=0.2,
random_state=42)

model = LinearRegression()
model.fit(X_train2, y_train2)
predictions = model.predict(X_test2)
```

```
mse = mean_squared_error(y_test2, predictions)
print(f"Mean Squared Error (MSE): {mse}")

r2 = r2_score(y_test2, predictions)
print(f"R-squared (R2): {r2}")

# Plot
plt.figure(figsize=(8, 6))
plt.scatter(y_test2, predictions, color='blue', alpha=0.5)  # Scatter plot
plt.plot([min(y_test2), max(y_test2)], [min(y_test2), max(y_test2)], color='red')  # Diagonal
line for reference
plt.title('Actual vs Predicted Total_Sales')
plt.xlabel('Actual Total_Sales')
plt.ylabel('Predicted Total_Sales')
plt.show()
```

Cross-Validation del nuevo modelo

```
model = LinearRegression()

scores = cross_val_score(model, pca_result2, y2, cv=3, scoring='r2')
print("R-squared scores for each fold:", scores)

mean_r2 = np.mean(scores)
std_r2 = np.std(scores)
print(f"\nMean R-squared (R2) across folds: {mean_r2}")
print(f"Standard deviation of R-squared (R2) across folds: {std_r2}")
```

Nueva Polynomial Regression

```
degree = 2
poly = PolynomialFeatures(degree=degree)
model2 = make_pipeline(poly, LinearRegression())

model2.fit(X_train2, y_train2)
predictions2 = model2.predict(X_test2)

mse = mean_squared_error(y_test2, predictions2)
print(f"Mean Squared Error (MSE): {mse}")

r2 = r2_score(y_test2, predictions2)
print(f"R-squared (R2): {r2}")
```

Nuevo Random Forest

```
rf_model2 = RandomForestRegressor(n_estimators=10, max_depth=10, random_state=42)
rf_model2.fit(X_train2, y_train2)
rf_predictions2 = rf_model2.predict(X_test2)

rf_mse = mean_squared_error(y_test2, rf_predictions2)
print(f"Random Forest Mean Squared Error (MSE): {rf_mse}")

rf_r2 = r2_score(y_test2, rf_predictions2)
print(f"Random Forest R-squared (R2): {rf_r2}")
```

```
# Plot
plt.figure(figsize=(8, 6))
plt.scatter(y_test2, rf_predictions2, color='blue', alpha=0.5)  # Scatter plot
plt.plot([min(y_test2), max(y_test2)], [min(y_test2), max(y_test2)], color='red')  # Diagonal
line for reference
plt.title('Actual vs Predicted Total_Sales')
plt.xlabel('Actual Total_Sales')
plt.ylabel('Predicted Total_Sales')
plt.show()

rf_cv_scores2 = cross_val_score(rf_model2, pca_result2, y, cv=3, scoring='r2')
print("Random Forest R-squared scores for each fold:", rf_cv_scores2)
rf_mean_r2 = np.mean(rf_cv_scores2)
rf_std_r2 = np.std(rf_cv_scores2)
print(f"\nRandom Forest Mean R-squared (R2) across folds: {rf_mean_r2}")
print(f"Random Forest Standard deviation of R-squared (R2) across folds: {rf_std_r2}")
```

Predicciones nuevo modelo con Random Forest

Creación de la **nueva columna Month_11** a partir de datos de una **regresión logístic**a a partir de Month_10:

```
from sklearn.linear_model import LogisticRegression
columns_for_prediction = [col for col in df_affiliated_sales_selected.columns if col !=
'Month_11']

X_train3 = df_affiliated_sales_selected[columns_for_prediction]
df_with_predicted_month_11 = df_affiliated_sales_selected.copy()

model = LogisticRegression()
model.fit(X_train3, df_affiliated_sales_selected['Month_10'])

predicted_month_11 = model.predict(X_train3)
df_with_predicted_month_11['Month_11'] = predicted_month_11

print(df_with_predicted_month_11)
```

Aplicamos PCA para 10 componentes

```
pca3 = PCA(n_components=10)
pca_result_predicted_month_11 = pca3.fit_transform(df_with_predicted_month_11)
```

Predecimos 'Total_Sales' para el nuevo df

```
predicted_total_sales = rf_model2.predict(pca_result_predicted_month_11)
df_with_predicted_month_11['Predicted_Total_Sales'] = predicted_total_sales
```

Filtramos los valores para Month_11 para obtener solo las filas que conntienen este mes

```
df_with_predicted_month_11 = df_with_predicted_month_11[df_with_predicted_month_11['Month_11']
== 1]

df_with_predicted_month_11.drop(['Month_5', 'Month_10'], axis=1, inplace=True)
```

Carga de predicciones en MS SQL Server

```
from sqlalchemy import create_engine
import pyodbc

server = 'DESKTOP-VUJ9ETK\\SQLEXPRESS'
database = 'TFM_EnriqueRocho'
driver = '{SQL Server}'
trusted_connection = 'yes'

conn_str =
f'DRIVER={driver};SERVER={server};DATABASE={database};Trusted_Connection={trusted_connection};
'
conn = pyodbc.connect(conn_str)
engine = create_engine(f'mssql+pyodbc://', creator=lambda: conn)

table_name = 'Prediccion_Sales'df_with_predicted_month_11.to_sql(table_name, engine,
index=False, if_exists='replace')
```

PREDICCIÓN TOTAL_OoS

Predicción de uds de rotura de stock en función de características de los productos.

Preparación de datos para los algoritmos

```
df_product = df_all_tables_no_outliers.copy()

df_product = df_product.drop(['Affiliated_Code', 'Affiliated_NAME', 'POSTALCODE',
       'poblacion', 'provincia', 'Engage', 'Management_Cluster', 'Location',
       'Tam_m2'], axis=1)
```

Dummy Variables

```
categorical_columns = ['Month','Product_Code', 'SIZE', 'Format']
df_product_cat = pd.get_dummies(df_product, columns=categorical_columns)
```

SelectKBest

```
X = df_product_cat.drop(columns=['Total_OoS'])  # Features
y = df_product_cat['Total_OoS']  # Variable Objetivo

selector = SelectKBest(score_func=f_regression, k=10)
X_new = selector.fit_transform(X, y)
selected_columns = X.columns[selector.get_support()]
selected_features_df = pd.DataFrame(X_new, columns=selected_columns)
```

Creamos el nuevo subset:

```
specified_columns2 = ['Total_OoS','Cost_price', 'Sell_price','Month_3', 'Month_4',
'Month_5','Month_6','Month_7','Month_8','Month_9','Month_10','Product_Code_Natu408','Product_C
ode_Dome206','Product_Code_Dome427','Product_Code_Natu079','Product_Code_Brit627','Product_Cod
e_Natu969','Product_Code_Dome164','Product_Code_Natu122','Product_Code_Brit700','Product_Code_
Dome213',Product_Code_Inte404','Product_Code_Dome363','Product_Code_Dome762','Product_Code_Dom
e459','Product_Code_Natu723','Product_Code_Inte327','Product_Code_Dome615','Product_Code_Dome7
97','Product_Code_Dome527','Product_Code_Brit555','Product_Code_Natu461','Product_Code_Dome104
','SIZE_85','SIZE_142','SIZE_190','SIZE_317','SIZE_283','SIZE_125','SIZE_114','SIZE_481','Form
at_ASL', 'Format_ATA','Format_ETO']
```

```
df_product_oos = df_product_cat[specified_columns2].copy()
```

StandardScaler, Clustering y PCA

```
y = df_product_oos['Total_OoS']
X_for_clustering = df_product_oos.drop(columns=['Total_OoS'])

numerical_columns = ['Cost_price', 'Sell_price']
scaler_for_pca = StandardScaler()
X_for_pca_scaled = X_for_clustering.copy()
X_for_pca_scaled[numerical_columns] =
scaler_for_pca.fit_transform(X_for_clustering[numerical_columns])

kmeans = KMeans(n_clusters=3, random_state=42)
clusters = kmeans.fit_predict(X_for_clustering)
X_for_clustering['Cluster'] = clusters

wcss = []

max_clusters = 10
for i in range(1, max_clusters + 1):
    kmeans = KMeans(n_clusters=i, random_state=42)
    kmeans.fit(X_for_clustering)
    wcss.append(kmeans.inertia_)

# Plot de inertia:
plt.figure(figsize=(8, 6))
plt.plot(range(1, max_clusters + 1), wcss, marker='o')
plt.title('Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('Within-Cluster Sum of Squares (WCSS)')
plt.grid(True)
plt.show()
```

Ajuste del num máximo de componentes

```
optimal_components =  15
pca = PCA(n_components=optimal_components)
pca_result = pca.fit_transform(X_for_pca_scaled)

# Plot
plt.figure(figsize=(8, 6))
plt.plot(range(1, len(pca.explained_variance_ratio_) + 1),
pca.explained_variance_ratio_.cumsum(), marker='o', linestyle='--')
plt.xlabel('Number of Components')
plt.ylabel('Cumulative Explained Variance')
plt.title('Explained Variance Ratio')
plt.show()
```

Aplicamos de nuevo PCA con los componentes óptimos

```
optimal_components = 11
pca = PCA(n_components=optimal_components)
pca_result = pca.fit_transform(X_for_pca_scaled)

df_pca = pd.DataFrame(data=pca_result, columns=[f'PCA_{i+1}' for i in
range(optimal_components)], index=X_for_clustering.index)

X_clustered = pd.concat([df_pca, X_for_clustering['Cluster']], axis=1)
```

```
df_processed = pd.concat([X_clustered, y], axis=1)
```

Aplicación de Algoritmos

Regresión Lineal

```
X2 = df_processed.drop(columns=['Total_OoS'])
y2 = df_processed['Total_OoS']
X_train2, X_test2, y_train2, y_test2 = train_test_split(X2, y2, test_size=0.2,
random_state=42)

reg = LinearRegression()
reg.fit(X_train2, y_train2)
y_pred2 = reg.predict(X_test2)

mse = mean_squared_error(y_test2, y_pred2)
r2 = r2_score(y_test2, y_pred2)

#plot Predicted vs. Actual Total_OoS
plt.scatter(y_test2, y_pred2)
plt.title('Linear Regression: Predicted vs. Actual Total_OoS')
plt.xlabel('Actual Total_OoS')
plt.ylabel('Predicted Total_OoS')
plt.show()
```

Visualización 3 primeros componentes

```
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(df_pca['PCA_1'], df_pca['PCA_2'], df_pca['PCA_3'], c=X_for_clustering['Cluster'],
cmap='viridis', alpha=0.5)
ax.set_xlabel('PCA_1')
ax.set_ylabel('PCA_2')
ax.set_zlabel('PCA_3')
ax.set_title('PCA Visualization with Cluster Assignments')
plt.show()
```

Random Forest

```
forest = RandomForestRegressor(random_state=42)
forest.fit(X_train2, y_train2)
feature_importances = forest.feature_importances_

feature_importance_df = pd.DataFrame({'Feature': X2.columns, 'Importance':
feature_importances})
```

Selección de features por importancia

```
feature_importance_df = feature_importance_df.sort_values(by='Importance', ascending=False)
top_features = feature_importance_df.head(10)['Feature'].tolist()
```

Vovemos a entrenar el modelo con los features seleccionados

```
reg = LinearRegression()
reg.fit(X_train2[top_features], y_train2)
y_pred3 = reg.predict(X_test2[top_features])

mse = mean_squared_error(y_test2, y_pred3)
r2 = r2_score(y_test2, y_pred3)
```

GradientBoostingRegressor

```
from sklearn.ensemble import GradientBoostingRegressor

grad_boost = GradientBoostingRegressor(random_state=42)
grad_boost.fit(X_train2, y_train2)
y_pred4 = grad_boost.predict(X_test2)

mse = mean_squared_error(y_test2, y_pred4)
r2 = r2_score(y_test2, y_pred4)
```

Mejoras en PCA

```
selector = SelectKBest(score_func=f_regression, k='all')
selector.fit(df_pca, y2)
```

Conseguimos puntuaciones y p-values para cada feature

```
feature_scores = pd.DataFrame({'Feature': df_pca.columns, 'Score': selector.scores_, 'p-
value': selector.pvalues_})
feature_scores.sort_values(by='Score', ascending=False, inplace=True)
```

Usamos los componentes top 5 según los Score obtenidos

```
k = 5
top_components = feature_scores['Feature'][:k].tolist()
selected_pca_components = df_pca[top_components]

X_train3, X_test3, y_train3, y_test3 = train_test_split(selected_pca_components, y2,
test_size=0.2, random_state=42)
```

Inicializamos la regresión lineal

```
regression_model = LinearRegression()
regression_model.fit(X_train3, y_train3)
predictions3 = regression_model.predict(X_test3)

mse = mean_squared_error(y_test3, predictions3)
r2 = r2_score(y_test3, predictions3)
```

Elección de **nuevas variables/features**

```
X3 = df_product_cat.drop(columns=['Total_OoS'])  # Features
y3 = df_product_cat['Total_OoS']  # Variable Objetivo

selector = SelectKBest(score_func=f_regression, k=10)
X_new = selector.fit_transform(X3, y3)
```

```
selected_columns = X3.columns[selector.get_support()]
selected_features_df = pd.DataFrame(X_new, columns=selected_columns)

print("Selected Features:")
print(selected_features_df.head())
```

Revisión de features con Random Forest

```
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X3, y3)

feature_importances = pd.Series(rf.feature_importances_, index=X3.columns)
feature_importances = feature_importances.sort_values(ascending=False)

# Plot
plt.figure(figsize=(10, 8))
feature_importances.head(20).plot(kind='barh')
plt.title('Top 20 Feature Importances')
plt.xlabel('Relative Importance')
plt.ylabel('Features')
plt.show()

X4 = df_product_cat[['Total_Sales', 'Total_Delivered', 'Cost_price', 'Sell_price', 'Margin',
'Month_3', 'Month_4', 'Month_5', 'Month_6', 'Month_7', 'Month_8', 'Month_9', 'Month_10',
'Product_Code_Natu408', 'Product_Code_Dome206', 'Product_Code_Dome427',
'Product_Code_Natu079', 'Product_Code_Brit627', 'Product_Code_Natu969',
'Product_Code_Dome164', 'Product_Code_Natu122', 'Product_Code_Brit700',
'Product_Code_Dome213', 'Product_Code_Inte404', 'Product_Code_Dome363',
'Product_Code_Dome762', 'Product_Code_Dome459', 'Product_Code_Natu723',
'Product_Code_Inte327', 'Product_Code_Dome615', 'Product_Code_Dome797',
'Product_Code_Dome527', 'Product_Code_Brit555', 'Product_Code_Natu461',
'Product_Code_Dome104','SIZE_85', 'SIZE_142', 'SIZE_190', 'SIZE_317', 'SIZE_283', 'SIZE_125',
'SIZE_114', 'SIZE_481']].copy()

y4 = df_product_cat[['Total_OoS']].copy()
```

Decision Tree

```
from sklearn.tree import DecisionTreeRegressor

X_train4, X_test4, y_train4, y_test4 = train_test_split(X4, y4, test_size=0.2,
random_state=42)

numerical_columns = ['Total_Sales','Cost_price', 'Sell_price','Margin']
```

Escalamos las variables numéricas

```
scaler = StandardScaler()
X_train4[numerical_columns] = scaler.fit_transform(X_train4[numerical_columns])
X_test4[numerical_columns] = scaler.transform(X_test4[numerical_columns])

decision_tree = DecisionTreeRegressor(random_state=42)
decision_tree.fit(X_train4, y_train4)
tree_preds = decision_tree.predict(X_test4)
```

```
mse = mean_squared_error(y_test4, tree_preds)
print(f"Decision Tree MSE: {mse}")

r2 = r2_score(y_test4, tree_preds)
print(f"Decision Tree R-squared Score {r2}")
```

Resampling y nuevo Decision Tree: cambio de peso de los valores de entrenamiento.

```
from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler
from imblearn.pipeline import Pipeline
```

Definimos la estrategia de resampling con oversampling para valores que no son 0 y undersample para valores 0

```
resampling_strategy = [('over', RandomOverSampler()), ('under', RandomUnderSampler())]

resampling_pipeline = Pipeline(steps=resampling_strategy)
X_train_resampled4, y_train_resampled4 = resampling_pipeline.fit_resample(X_train4, y_train4)

clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train_resampled4, y_train_resampled4)
y_pred4 = clf.predict(X_test4)

print(classification_report(y_test4, y_pred4))

precision = precision_score(y_test4, y_pred4, average='weighted

plt.scatter(y_test4, y_pred4)
plt.title('Linear Regression: Predicted vs. Actual Total_OoS')
plt.xlabel('Actual Total_OoS')
plt.ylabel('Predicted Total_OoS')
plt.show()
```

GaussianNB sobre resampled

```
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report

naive_bayes = GaussianNB()

naive_bayes.fit(X_train_resampled4, y_train_resampled4)
nb_preds = naive_bayes.predict(X_test4)

accuracy = accuracy_score(y_test4, nb_preds)
print(f"Gaussian Naive Bayes Accuracy: {accuracy}")
print(classification_report(y_test4, nb_preds))
```

SVM sobre resampled

```
from sklearn.svm import SVC
from sklearn.metrics import classification_report, precision_score
from imblearn.pipeline import Pipeline
```

Seleccionamos las columnas de nuevo

```
specified_columns3 = df_product_cat[['Total_OoS','Total_Sales', 'Total_Delivered',
'Cost_price', 'Sell_price', 'Margin','Month_3', 'Month_4', 'Month_5', 'Month_6', 'Month_7',
'Month_8', 'Month_9', 'Month_10','Product_Code_Natu408', 'Product_Code_Dome206',
'Product_Code_Dome427','Product_Code_Natu079', 'Product_Code_Brit627', 'Product_Code_Natu969',
'Product_Code_Dome164', 'Product_Code_Natu122', 'Product_Code_Brit700',
'Product_Code_Dome213', 'Product_Code_Inte404', 'Product_Code_Dome363',
'Product_Code_Dome762', 'Product_Code_Dome459', 'Product_Code_Natu723',
'Product_Code_Inte327', 'Product_Code_Dome615', 'Product_Code_Dome797',
'Product_Code_Dome527', 'Product_Code_Brit555', 'Product_Code_Natu461',
'Product_Code_Dome104','SIZE_85', 'SIZE_142', 'SIZE_190', 'SIZE_317', 'SIZE_283', 'SIZE_125',
'SIZE_114', 'SIZE_481']].copy()
```

Se realiza sample de los datos para evitar problemas de procesamiento con SVM

```
sampled_data = specified_columns3.sample(frac=0.03, random_state=42)

X5 = sampled_data.drop(columns=['Total_OoS'])
y5 = sampled_data['Total_OoS']
X_train5, X_test5, y_train5, y_test5 = train_test_split(X5, y5, test_size=0.2,
random_state=42)

numerical_columns = ['Total_Sales', 'Total_Delivered', 'Cost_price', 'Sell_price', 'Margin']
scaler = StandardScaler()
X_train5[numerical_columns] = scaler.fit_transform(X_train5[numerical_columns])
X_test5[numerical_columns] = scaler.transform(X_test5[numerical_columns])
```

Resampling

```
resampling_strategy = [('over', RandomOverSampler()), ('under', RandomUnderSampler())]
resampling_pipeline = Pipeline(steps=resampling_strategy)
X_train_resampled5, y_train_resampled5 = resampling_pipeline.fit_resample(X_train5, y_train5)

svm_classifier = SVC(random_state=42)
svm_classifier.fit(X_train_resampled5, y_train_resampled5)
y_pred5 = svm_classifier.predict(X_test5)

print(classification_report(y_test5, y_pred5))
precision = precision_score(y_test5, y_pred5, average='weighted')
print(f"Precision: {precision}")

plt.scatter(y_test5, y_pred5)
plt.title('Linear Regression: Predicted vs. Actual Total_OoS')
plt.xlabel('Actual Total_OoS')
plt.ylabel('Predicted Total_OoS')
plt.show()
```

PREDICCIONES Total_OoS: se usa un código similar que en Total_Sales para crear la nueva columna Month_11 con regresión logística y se predicen los valores de Total_OoS para ese nuevo mes.

Finalmente se realiza la subida a MS SQL Server de las predicciones.