EDA

June 12, 2025

Importing libraries

[207]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import statsmodels.api as sm
```

[179]:
```
data = pd.read_csv("./data/data.csv")

data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6240 entries, 0 to 6239
Data columns (total 16 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   geo                        6240 non-null   object
 1   date                       6240 non-null   object
 2   tv_impression              6240 non-null   int64
 3   radio_impression           6240 non-null   int64
 4   print_impression           6240 non-null   int64
 5   search_impression          6240 non-null   int64
 6   social_impression          6240 non-null   int64
 7   competitor_sales_control   6240 non-null   float64
 8   sentiment_score_control    6240 non-null   float64
 9   tv_spend                   6240 non-null   float64
 10  radio_spend                6240 non-null   float64
 11  print_spend                6240 non-null   float64
 12  search_spend               6240 non-null   float64
 13  social_spend               6240 non-null   float64
 14  population                 6240 non-null   float64
 15  sales                      6240 non-null   float64
dtypes: float64(9), int64(5), object(2)
memory usage: 780.1+ KB
```

[180]:
```
data.describe()
```

[180]:
```
       tv_impression  radio_impression  print_impression  search_impression  \
count   6.240000e+03      6.240000e+03      6.240000e+03       6.240000e+03
mean    8.851635e+05      5.206055e+05      2.605050e+05       1.806810e+06
std     8.193451e+05      6.206231e+05      5.551842e+05       1.304818e+06
min     0.000000e+00      0.000000e+00      0.000000e+00       0.000000e+00
25%     2.536458e+05      0.000000e+00      0.000000e+00       8.083565e+05
50%     6.796705e+05      3.106525e+05      0.000000e+00       1.505379e+06
75%     1.324174e+06      8.075525e+05      2.547340e+05       2.566118e+06
max     5.192032e+06      4.397540e+06      5.610156e+06       7.635147e+06

       social_impression  competitor_sales_control  sentiment_score_control  \
count       6.240000e+03                6240.000000              6240.000000
mean        9.816780e+05                  -0.044412                -0.033675
std         9.141768e+05                   1.210164                 1.175572
min         0.000000e+00                  -4.825634                -4.230392
25%         2.675945e+05                  -0.872736                -0.808085
50%         7.462475e+05                  -0.050346                -0.036433
75%         1.476708e+06                   0.743633                 0.739391
max         6.975542e+06                   3.924682                 4.320259

           tv_spend   radio_spend   print_spend  search_spend  social_spend  \
count   6240.000000   6240.000000   6240.000000   6240.000000   6240.000000
mean    6490.659147   5019.278558   1935.793164  14080.185617   7649.147993
std     6008.031135   5983.571424   4125.531614  10168.239254   7123.184932
min        0.000000      0.000000      0.000000      0.000000      0.000000
25%     1859.914025      0.000000      0.000000   6299.393250   2085.072625
50%     4983.835500   2995.072650      0.000000  11731.178500   5814.694600
75%     9709.803000   7785.800675   1892.909150  19997.351500  11506.376500
max    38071.730000  42397.700000  41688.645000  59499.484000  54352.810000

          population          sales
count    6240.000000    6240.000000
mean   542418.832750  211330.038098
std    242839.719887  121605.908396
min    136670.940000    9742.282233
25%    335176.345000  114323.174425
50%    560478.825000  193995.582650
75%    736033.810000  287918.213625
max    994048.940000  739823.338900
```

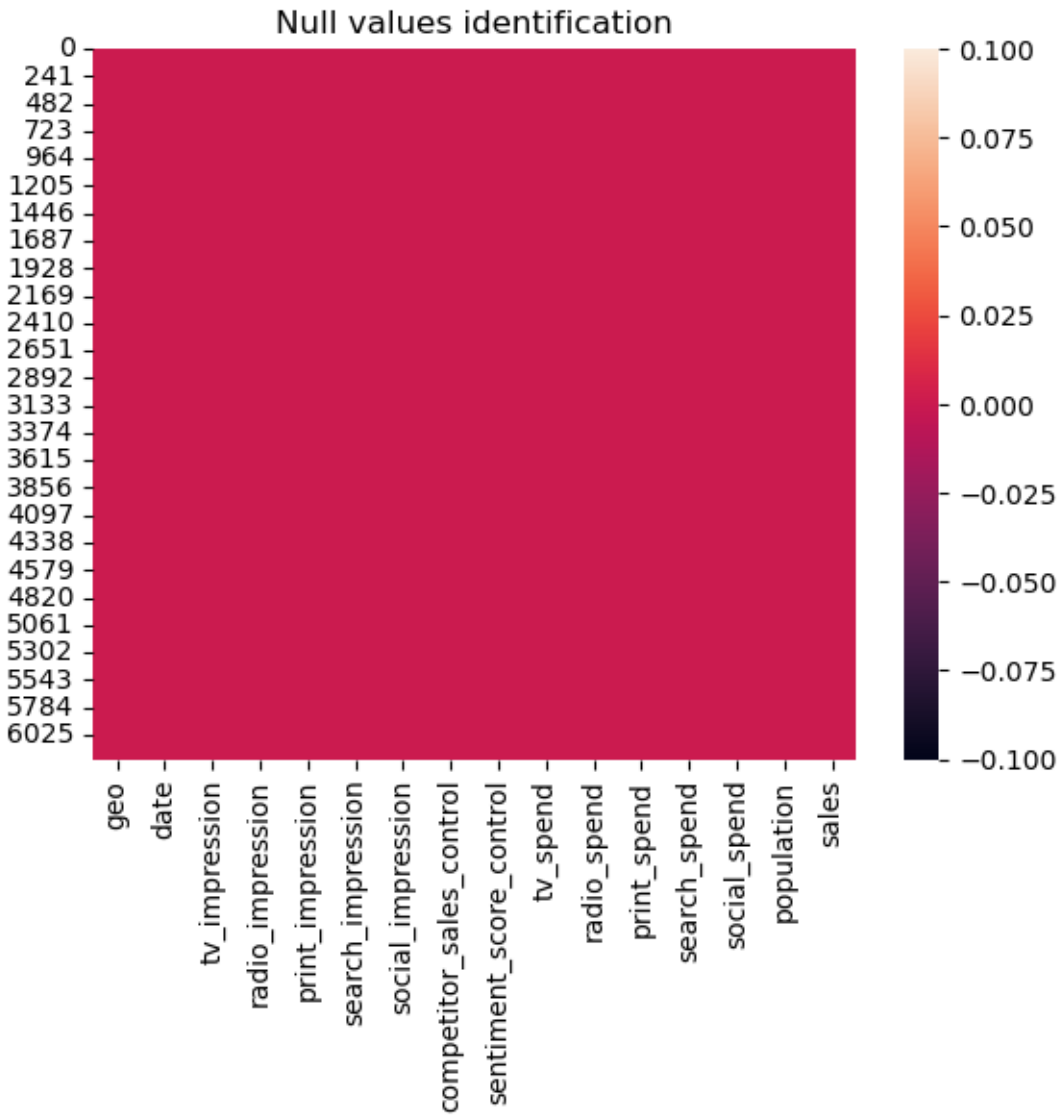
[181]:
```
data.head()
```

[181]:
```
    geo     date  tv_impression  radio_impression  print_impression  \
0  Geo0  1/25/21         280668                 0                 0
1  Geo0   2/1/21         366206            182108             19825
2  Geo0   2/8/21         197565            230170                 0
3  Geo0  2/15/21         140990             66643                 0
```

```
4  Geo0  2/22/21         399116            164991                  0

   search_impression  social_impression  competitor_sales_control  \
0             470611             108010                 -1.338765
1             527702             252506                  0.893645
2             393618             184061                 -0.284549
3             326034             201729                 -1.034740
4             381982             153973                 -0.319276

   sentiment_score_control   tv_spend  radio_spend  print_spend  search_spend  \
0                 0.115581  2058.0608      0.00000      0.00000     3667.3965
1                 0.944224  2685.2874   1755.74540    147.31808     4112.2974
2                -1.290579  1448.6895   2219.12230      0.00000     3067.4023
3                -1.084514  1033.8406    642.52057      0.00000     2540.7310
4                -0.017503  2926.6072   1590.71640      0.00000     2976.7249

   social_spend  population        sales
0     841.6044   136670.94  39198.55690
1    1967.5044   136670.94  41497.96063
2    1434.1870   136670.94  41579.08885
3    1571.8545   136670.94  56492.86151
4    1199.7440   136670.94  71039.82718
```

Identifying Null values

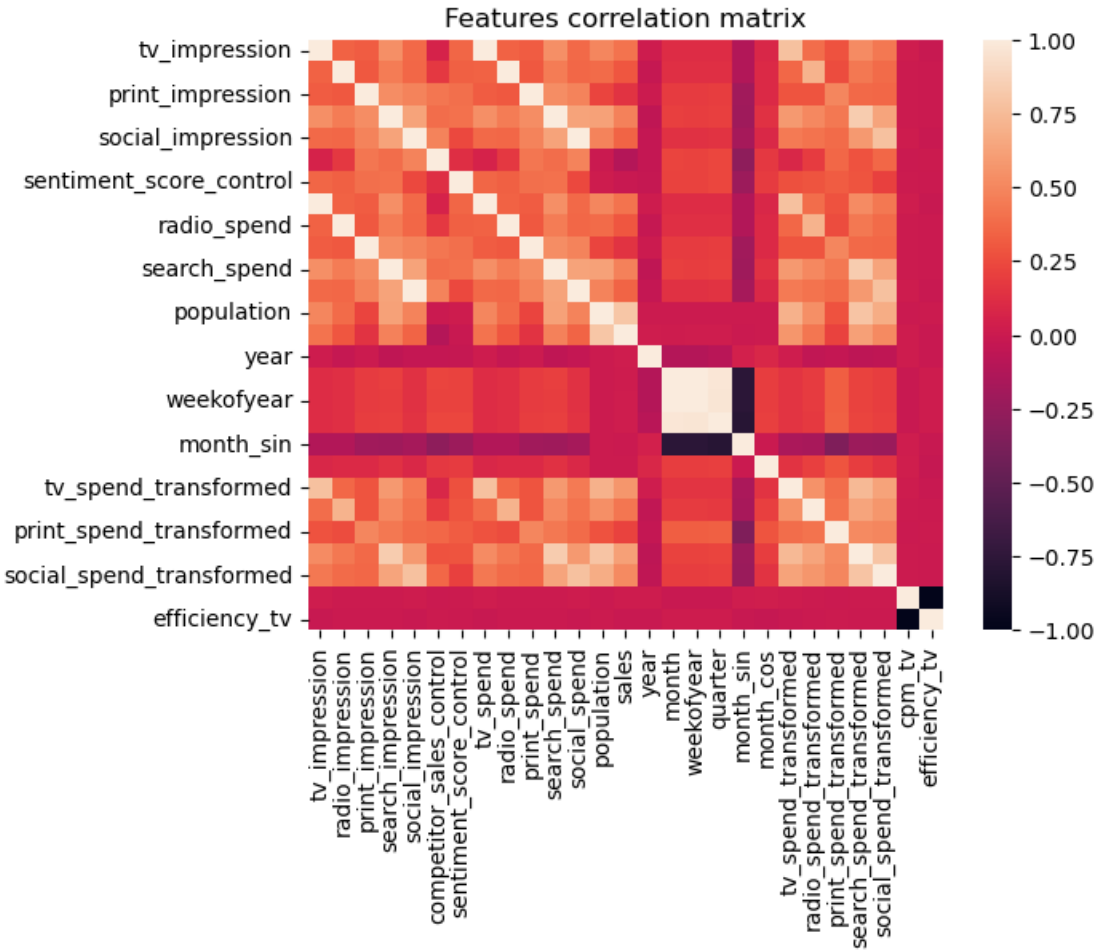
[182]:
```
sns.heatmap(data.isnull())
plt.title("Null values identification")
plt.show()
```



0.1 Feature correlation analysis

The purpose is to identify the feature relations and its relevance with the sales number

[246]:
```
correlations = data.select_dtypes(include=['number']).corr()

sns.heatmap(correlations)
plt.title("Features correlation matrix")
plt.show()
```



[247]:
```
numeric_cols = data.select_dtypes(include=['number'])

correlations = numeric_cols.corr()['sales'].drop('sales')  # quitamos la␣
↪diagonal

top_positive = correlations.sort_values(ascending=False).head(10)

top_negative = correlations.sort_values(ascending=True).head(5)

print("  Top 5 variables más positivamente correlacionadas con 'sales':")
print(top_positive)

print("\n  Top 5 variables más negativamente correlacionadas con 'sales':")
print(top_negative)
```

```
  Top 5 variables más positivamente correlacionadas con 'sales':
population                  0.798669
```

```
search_spend_transformed    0.610072
tv_spend_transformed        0.570103
social_spend_transformed    0.510654
search_spend                0.468858
search_impression           0.468858
radio_spend_transformed     0.425363
tv_spend                    0.413745
tv_impression               0.413745
social_spend                0.348154
Name: sales, dtype: float64

 Top 5 variables más negativamente correlacionadas con 'sales':
competitor_sales_control   -0.116704
efficiency_tv              -0.013164
sentiment_score_control    -0.009409
month_sin                  -0.000891
month_cos                   0.002995
Name: sales, dtype: float64
```

0.2 Feature Engineering

This process involves change the geo and date format to work with them in next analysis

[183]:
```
print(f"Unique 'geo' values:  {len(data["geo"].unique())}")
print(f"Unique 'population' values:  {len(data["population"].unique())}")
```

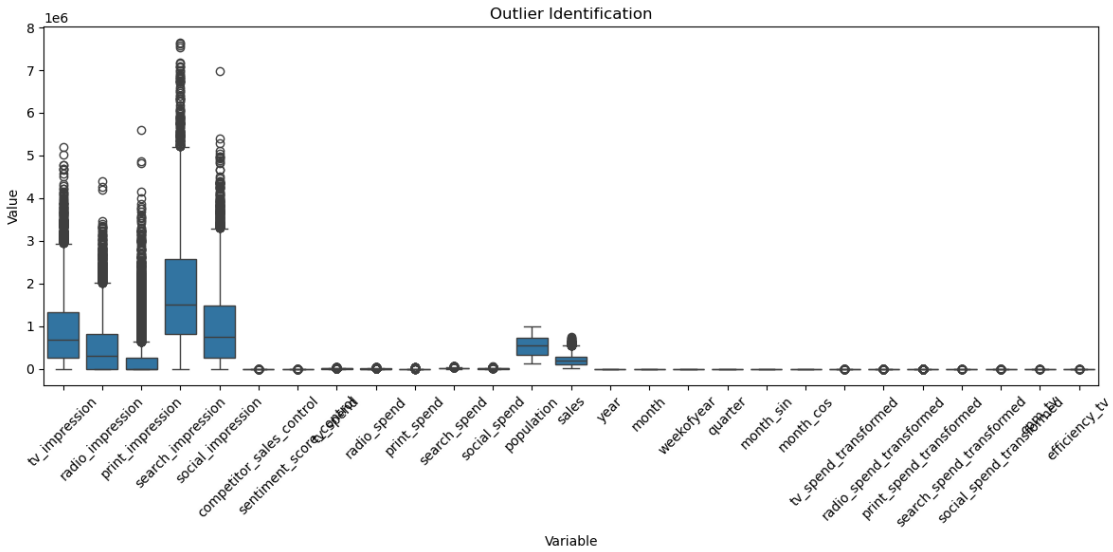
```
Unique 'geo' values:  40
Unique 'population' values:  40
```

[236]:
```
data['date'] = pd.to_datetime(data['date'])
data['year'] = data['date'].dt.year
data['month'] = data['date'].dt.month
data['weekofyear'] = data['date'].dt.isocalendar().week.astype(int)


#data['is_weekend'] = data['date'].dt.dayofweek >= 5
#data['is_weekend'] = data['is_weekend'].astype(int)
data['quarter'] = data['date'].dt.quarter
data['month_sin'] = np.sin(2 * np.pi * data['month'] / 12)
data['month_cos'] = np.cos(2 * np.pi * data['month'] / 12)
```

0.3 Seasonality Analysis

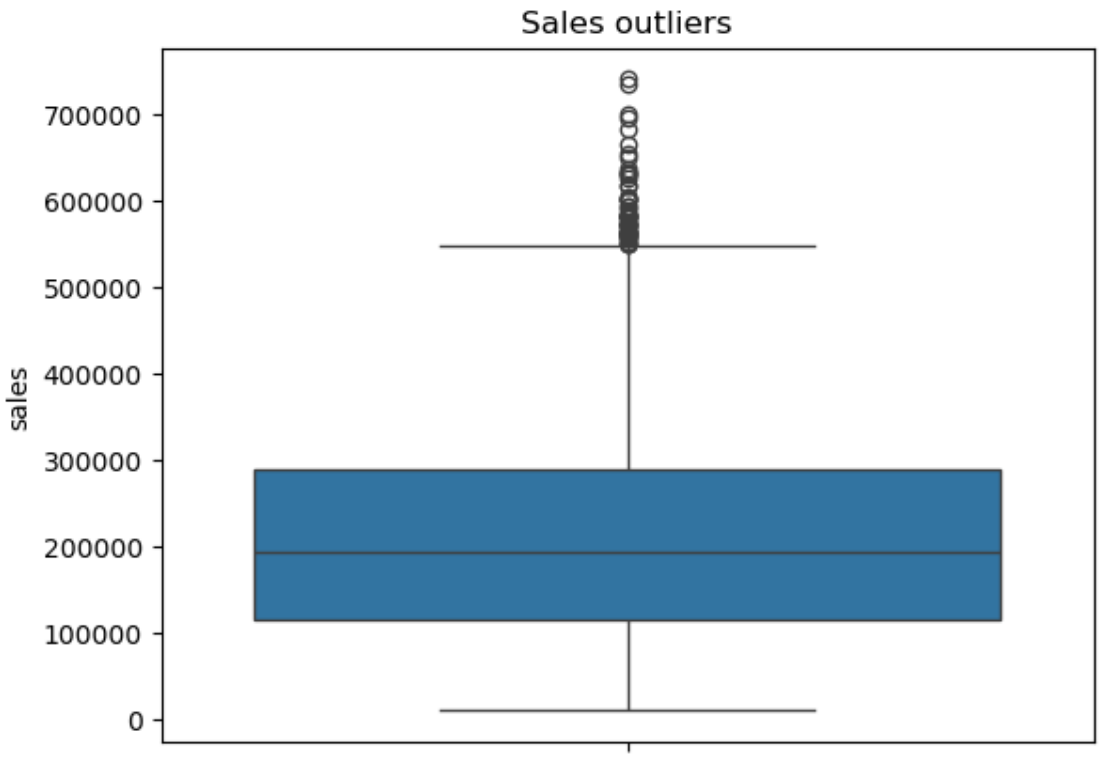
[248]:
```
numeric_df = data.select_dtypes(include='number')

melted = numeric_df.melt(var_name='Variable', value_name='Value')

plt.figure(figsize=(12, 6))
```

```
sns.boxplot(x='Variable', y='Value', data=melted)
plt.title("Outlier Identification")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

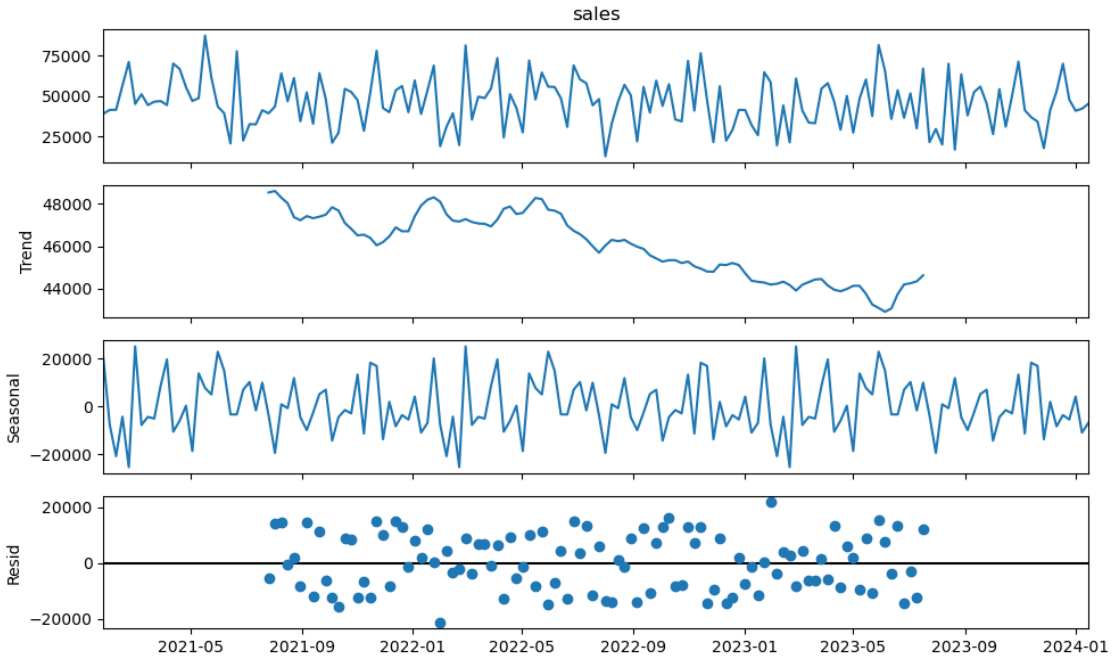


[ ]:

[249]:
```
sns.boxplot(data["sales"])
plt.title("Sales outliers")
plt.show()
```



[241]:
```
from statsmodels.tsa.seasonal import seasonal_decompose

# Filtra por geo y agrega ventas por fecha
geo_df = data[data['geo'] == 'Geo0'].groupby('date')['sales'].sum()

# Asegura que el índice esté en formato datetime y ordenado
geo_df = geo_df.sort_index()

# Descomposición estacional (weekly, suponiendo 1 punto por semana)
result = seasonal_decompose(geo_df, model='additive', period=52)

# Mostrar gráfico de tendencia, estacionalidad y residuales
fig = result.plot()
fig.set_size_inches(10, 6)
plt.tight_layout()
plt.show()
```



0.4 Coefficient Analysis

[242]:
```
import statsmodels.api as sm

# Variables seleccionadas
features = [
    'tv_spend', 'radio_spend', 'print_spend',
    'search_spend', 'social_spend',
    'competitor_sales_control', 'sentiment_score_control',"weekofyear",
    'population', 'month_sin', 'month_cos'
]

# Separar X e y
X = data[features]
y = data['sales']

# Agregar constante al modelo
X = sm.add_constant(X)

# Ajustar modelo OLS
model = sm.OLS(y, X).fit()

# Mostrar resumen con coeficientes y significancia
print(model.summary())
```

OLS Regression Results

```
==============================================================================
Dep. Variable:                  sales   R-squared:                       0.656
Model:                            OLS   Adj. R-squared:                  0.656
Method:                 Least Squares   F-statistic:                     1082.
Date:                Thu, 12 Jun 2025   Prob (F-statistic):               0.00
Time:                        19:23:01   Log-Likelihood:                -78582.
No. Observations:                6240   AIC:                         1.572e+05
Df Residuals:                    6228   BIC:                         1.573e+05
Df Model:                          11
Covariance Type:            nonrobust
==============================================================================
============
                               coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
------------
const                    -1.255e+04   3415.979     -3.673      0.000   -1.92e+04
-5851.108
tv_spend                     0.6451      0.193      3.350      0.001       0.268
1.023
radio_spend                  0.2253      0.177      1.271      0.204      -0.122
0.573
print_spend                  0.6206      0.282      2.201      0.028       0.068
1.173
search_spend                 0.3945      0.161      2.451      0.014       0.079
0.710
social_spend                 0.4145      0.185      2.242      0.025       0.052
0.777
competitor_sales_control -1.624e+04   1023.174    -15.873      0.000   -1.82e+04
-1.42e+04
sentiment_score_control  -5875.3428   1005.590     -5.843      0.000   -7846.646
-3904.039
weekofyear                 228.1944     98.627      2.314      0.021      34.852
421.537
population                   0.3718      0.006     61.615      0.000       0.360
0.384
month_sin                -3032.2411   2122.479     -1.429      0.153   -7193.032
1128.550
month_cos                 3885.6260   1379.008      2.818      0.005    1182.294
6588.958
==============================================================================
Omnibus:                      211.793   Durbin-Watson:                   1.757
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              581.625
Skew:                           0.075   Prob(JB):                    5.03e-127
Kurtosis:                       4.488   Cond. No.                     2.44e+06
==============================================================================

Notes:
```

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.44e+06. This might indicate that there are strong multicollinearity or other numerical problems.

0.5 Adstock, Saturation and Coefficient analysis

[243]:
```
def apply_adstock(series, decay= 0.5):
    result = []
    for i, val in enumerate(series):
        if i == 0:
            result.append(val)
        else:
            result.append(val + decay * result[i-1])

    return pd.Series(result, index=series.index)

spend_channels = ['tv_spend', 'radio_spend', 'print_spend', 'search_spend',␣
 ↪'social_spend']

decay = 0.5
for channel in spend_channels:
    adstocked = apply_adstock(data[channel], decay)
    data[f'{channel}_transformed'] = np.log1p(adstocked)  # Saturación␣
 ↪logarítmica
```

[244]:
```
features_adstock = ["tv_spend_transformed", "radio_spend_transformed",
    "print_spend_transformed", "search_spend_transformed",␣
 ↪"social_spend_transformed",
    'competitor_sales_control', 'sentiment_score_control', 'population',
    'month_sin', 'month_cos'
]

X_adstock = data[features_adstock]
y_adstock = data['sales']
X_adstock = sm.add_constant(X_adstock)

model_adstock = sm.OLS(y_adstock, X_adstock).fit()

print(model_adstock.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  sales   R-squared:                       0.655
Model:                            OLS   Adj. R-squared:                  0.654
Method:                 Least Squares   F-statistic:                     1181.
Date:                Thu, 12 Jun 2025   Prob (F-statistic):               0.00
Time:                        19:23:01   Log-Likelihood:                -78597.
```

```
No. Observations:                6240   AIC:                         1.572e+05
Df Residuals:                    6229   BIC:                         1.573e+05
Df Model:                          10
Covariance Type:            nonrobust
==============================================================================
============
                                coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
------------
const                     -8.729e+04   2.91e+04     -2.999      0.003   -1.44e+05
-3.02e+04
tv_spend_transformed       4893.2143   1952.947      2.506      0.012    1064.765
8721.663
radio_spend_transformed    1547.2075   1289.377      1.200      0.230    -980.416
4074.831
print_spend_transformed    1275.0343    571.206      2.232      0.026     155.274
2394.795
search_spend_transformed    181.7293   3382.299      0.054      0.957   -6448.743
6812.202
social_spend_transformed   2486.6020   2094.781      1.187      0.235   -1619.892
6593.095
competitor_sales_control -1.432e+04    913.950    -15.670      0.000   -1.61e+04
-1.25e+04
sentiment_score_control  -3619.5570    906.991     -3.991      0.000   -5397.572
-1841.542
population                   0.3776      0.008     47.253      0.000       0.362
0.393
month_sin                -5364.8144   1451.347     -3.696      0.000   -8209.956
-2519.673
month_cos                 3038.2383   1399.094      2.172      0.030     295.532
5780.945
==============================================================================
Omnibus:                      213.977   Durbin-Watson:                   1.754
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              579.906
Skew:                           0.094   Prob(JB):                    1.19e-126
Kurtosis:                       4.482   Cond. No.                     1.92e+07
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.92e+07. This might indicate that there are
strong multicollinearity or other numerical problems.
```

0.6 ROI calculation by channel

[245]:
```
spend_channels = ['tv_spend', 'radio_spend', 'print_spend', 'search_spend',␣
 ↪'social_spend']
transformed_channels = [f'{ch}_transformed' for ch in spend_channels]

features_adstock = transformed_channels + [
    'competitor_sales_control', 'sentiment_score_control', 'population',
    'month_sin', 'month_cos'
]

X_adstock = sm.add_constant(data[features_adstock])
y_adstock = data['sales']

model_adstock = sm.OLS(y_adstock, X_adstock).fit()

coeffs = model_adstock.params

contributions = {}
roi = {}

for ch in spend_channels:
    ch_trans = f"{ch}_transformed"

    avg_trans = data[ch_trans].mean()
    avg_spend = data[ch].mean()

    contrib = coeffs[ch_trans] * avg_trans
    contributions[ch] = contrib
    roi[ch] = contrib / avg_spend if avg_spend != 0 else np.nan

df_contrib_roi = pd.DataFrame({
    "channel": spend_channels,
    'contribution': [contributions[ch] for ch in spend_channels],
    'roi': [roi[ch] for ch in spend_channels]
})

print(df_contrib_roi)
```

```
        channel  contribution       roi
0      tv_spend  45166.112549  6.958633
1   radio_spend  13700.820908  2.729639
2   print_spend   9025.087838  4.662217
3  search_spend   1827.975399  0.129826
4  social_spend  23329.146063  3.049901
```