Universidad del Valle de Guatemala

Facultad de Ingeniería

Departamento de Computación

Laboratorio 2: Series de Tiempo

Integrantes:

- Diego Alexander Hernández Silvestre, 21270
- Linda Inés Jiménez Vides, 21169

Curso: Data Science
Sección: 10

Guatemala, 1 de agosto de 2024

En primera instancia, se importan las librerías requeridas para poder realizar los cálculos durante el laboratorio.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sweetviz as sv
import statsmodels.api as sm
import matplotlib.pyplot as plt
from autoviz.AutoViz_Class import AutoViz_Class
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.stattools import adfuller
from pmdarima import auto_arima
from prophet import Prophet
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LinearRegression
from statsmodels.tsa.arima.model import ARIMA
%matplotlib inline
```

Dataset: CONSUMO-2024-05.xlsx

Las columnas que contienen la información sobre el diésel, se encuentran separadas en dos variables *Diesel alto azufre* y *Diesel bajo azufre* por lo que se implementa una función para llevar a cambo la combinación en una resultante.

```
def combineDieselColumns(csvPath):
    print(" Iniciando la combinación de las columnas de Diesel...")

    try:
        # Leer el archivo CSV en un DataFrame
        df = pd.read_csv(csvPath)
        print(f" Archivo cargado: {csvPath}")
    except Exception as e:
        print(f" Error al cargar {csvPath}: {e}")
        return None

    if 'Diesel alto azufre' not in df.columns or 'Diesel bajo azufre'
not in df.columns:
        print(" Las columnas 'Diesel alto azufre' o 'Diesel bajo
azufre' no existen en el archivo CSV.")
        return None

    # Crear la nueva columna 'Diesel'
    df['Diesel'] = df['Diesel alto azufre'].replace(0,
pd.NA).combine_first(df['Diesel bajo azufre'].replace(0, pd.NA))

    print(" Combinación de columnas completada.")
    return df

csvPath = 'Consumo/CONSUMO-2024-05.csv'
dataConsumo = combineDieselColumns(csvPath)
```

```
 Iniciando la combinación de las columnas de Diesel...
 Archivo cargado: Consumo/CONSUMO-2024-05.csv
 Combinación de columnas completada.
```

Luego, se realiza un filtrado de las columnas relevantes del dataset.

```
dataConsumo = dataConsumo[['Fecha', 'Gasolina regular', 'Gasolina
superior', 'Diesel', 'Gas licuado de petróleo']]
dataConsumo['Fecha'] = pd.to_datetime(dataConsumo['Fecha'])
dataConsumo.head()
```

```
    Fecha     Gasolina regular  Gasolina superior   Diesel    \
0 2000-01-01      202645.20         308156.82      634667.06
1 2000-02-01      205530.96         307766.31      642380.66
2 2000-03-01      229499.56         331910.29      699807.25
3 2000-04-01      210680.40         315648.08      586803.98
```

```
4 2000-05-01      208164.34         319667.97       656948.2

   Gas licuado de petróleo
0       194410.476190
1       174710.552381
2       189234.066667
3       174330.607143
4       191745.147619
```

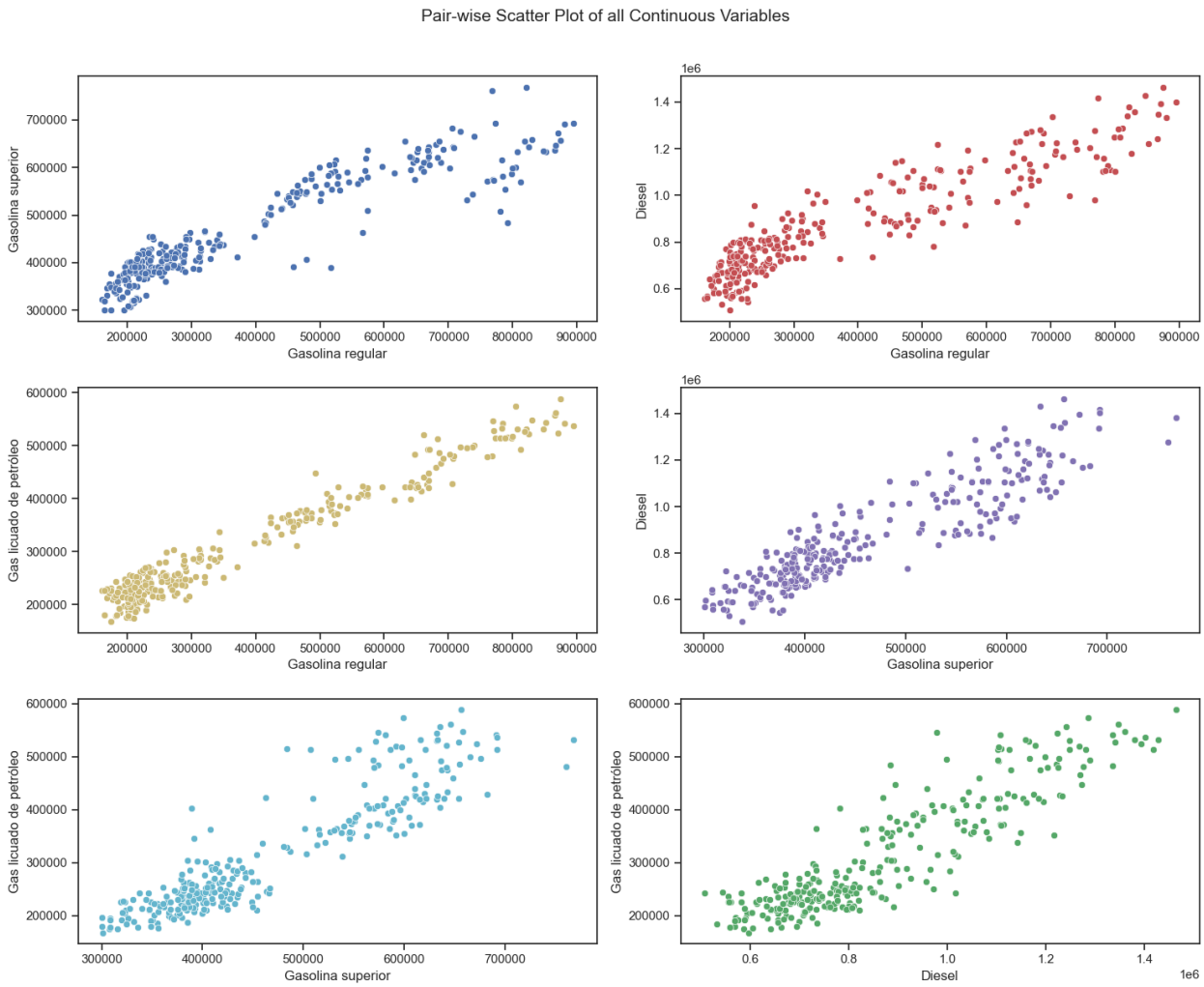
Este es un método alternativo que se puede utilizar para hacer un EDA automatizado. Sin embargo, en comparativa con otros métodos, se decidió no utilizar pero se dejó a manera de proporcionar un recurso extra.

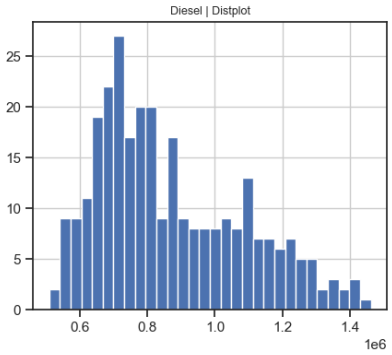
```
#plt.figure(figsize=(12, 8))

#for i, column in enumerate(['Gasolina regular', 'Gasolina superior',
'Diesel', 'Gas licuado de petróleo'], 1):
#    plt.subplot(2, 2, i)
#    sns.histplot(dataConsumo[column], kde=True)
#    plt.title(f'Distribución de {column}')
#    plt.xlabel(column)

#plt.tight_layout()
#plt.show()
```

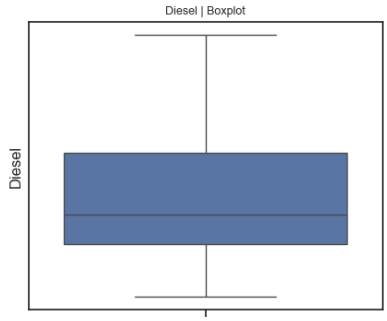
Para facilitar el EDA, se utilizó la librería AutoViz para automatizar este proceso. De este se puede destacar que en el dataset de Consumo, todas las variables cuentan con distribuciones asimétricas positivas, por lo que se puede deducir que existe una mayor concentración de valores bajos y algunos valores muy altos. Los graficos de caja y bigotes indican la presencia de valores atipicos en todas las variables y los Q-Q plots muestran que ninguna de las variables cuenta con una distribución normal.

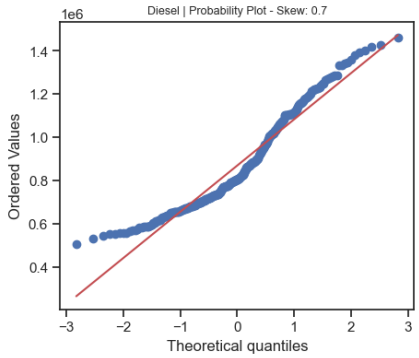
```
csvPath = 'Consumo/dataConsumo.csv'
dataConsumo.to_csv(csvPath, index=False)
AV = AutoViz_Class()
AV.AutoViz(csvPath)

Shape of your Data Set loaded: (293, 5)
############################################################################
#################
######################### C L A S S I F Y I N G  V A R I A B L E S
####################
############################################################################
#################
Classifying variables in data set...
    Number of Numeric Columns =  4
    Number of Integer-Categorical Columns =  0
    Number of String-Categorical Columns =  0
    Number of Factor-Categorical Columns =  0
```

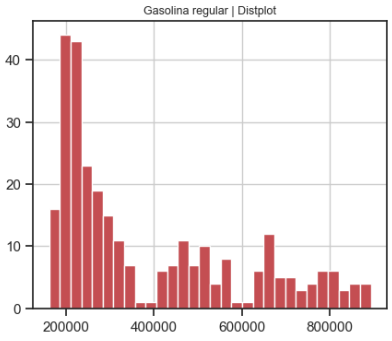
```
    Number of String-Boolean Columns =  0
    Number of Numeric-Boolean Columns =  0
    Number of Discrete String Columns =  0
    Number of NLP String Columns =  0
    Number of Date Time Columns =  0
    Number of ID Columns =  1
    Number of Columns to Delete =  0
    5 Predictors classified...
        1 variable(s) removed since they were ID or low-information
variables
        List of variables removed: ['Fecha']
To fix these data quality issues in the dataset, import FixDQ from
autoviz...
    All variables classified into correct types.

<pandas.io.formats.style.Styler at 0x15458640650>

Number of All Scatter Plots = 10
```

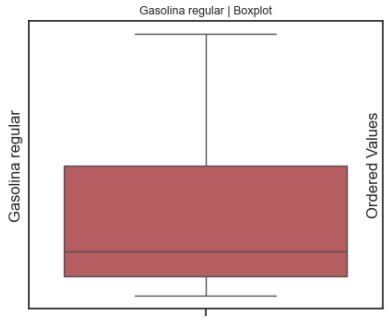


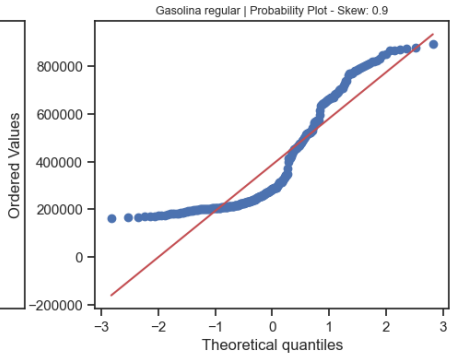

Diesel | Distplot
1e6

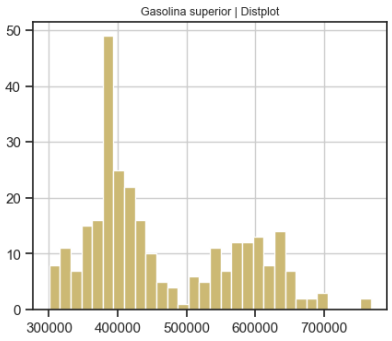

Diesel | Boxplot
Diesel

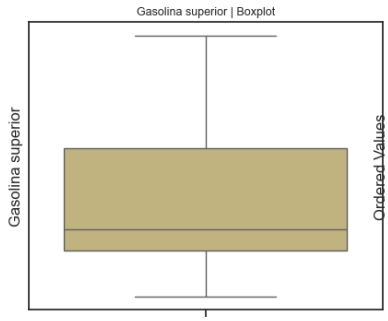

Diesel | Probability Plot - Skew: 0.7
1e6
Ordered Values
Theoretical quantiles

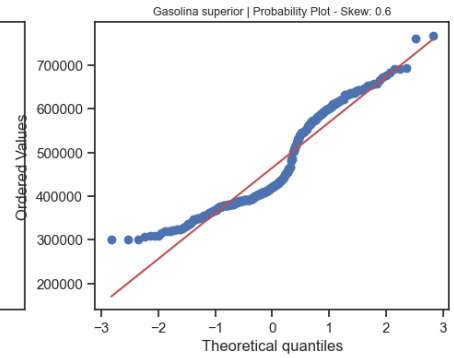

Gasolina regular | Distplot

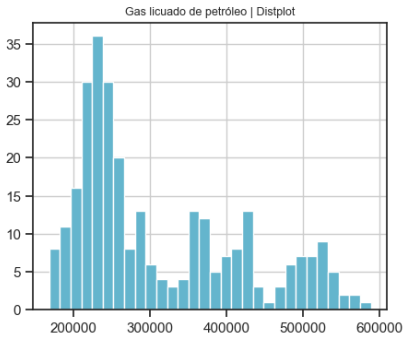

Gasolina regular | Boxplot
Gasolina regular

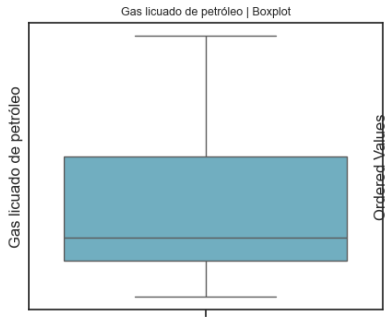

Gasolina regular | Probability Plot - Skew: 0.9
Ordered Values
Theoretical quantiles

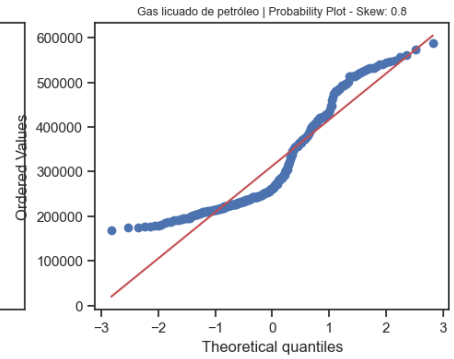

Gasolina superior | Distplot

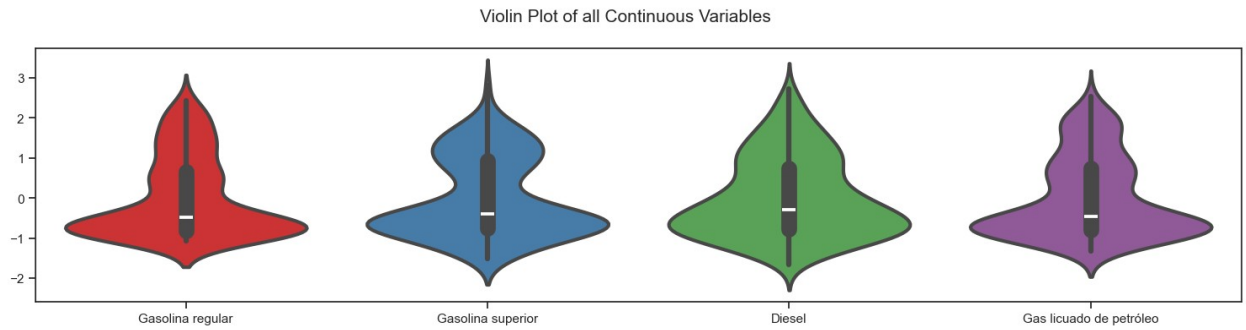

Gasolina superior | Boxplot
Gasolina superior

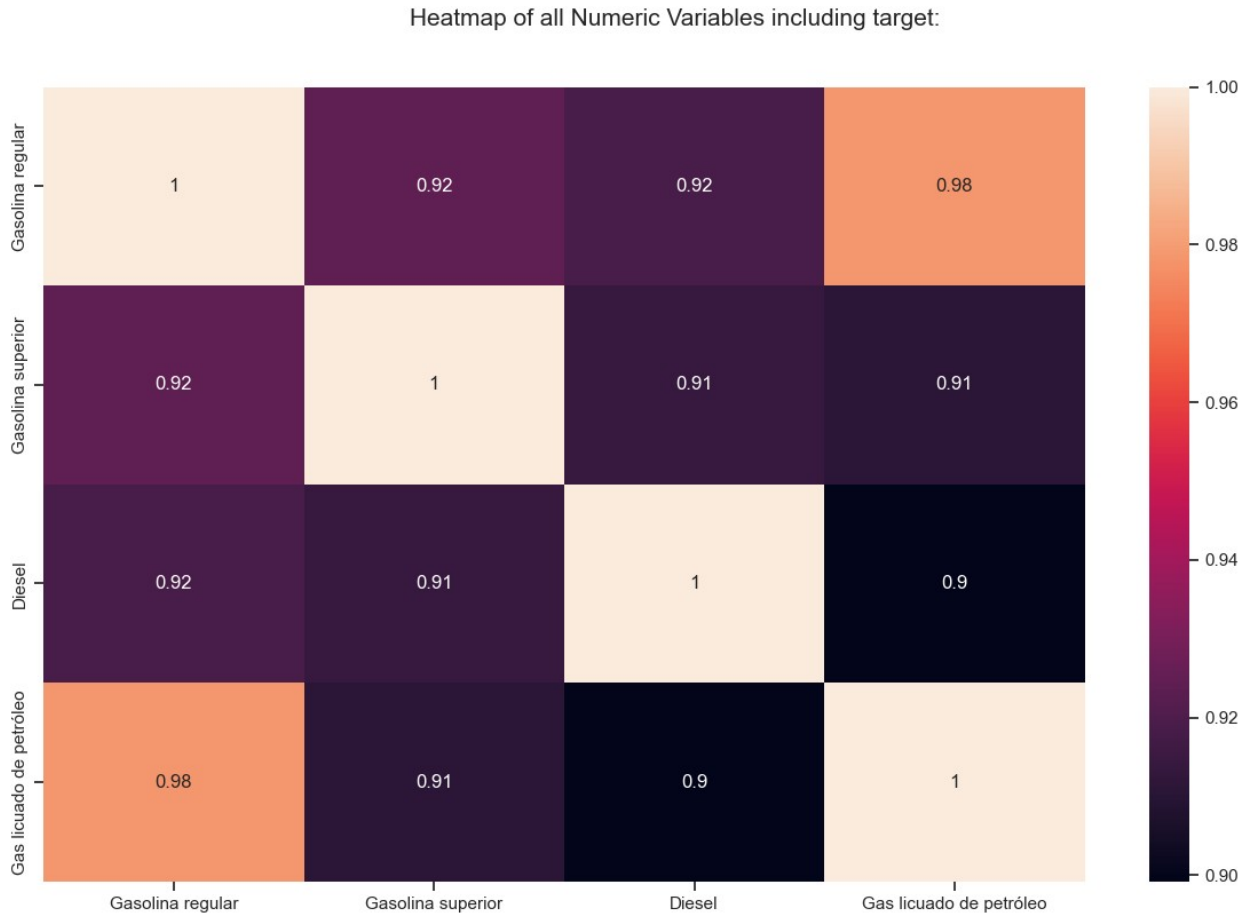

Gasolina superior | Probability Plot - Skew: 0.6
Ordered Values
Theoretical quantiles


Gas licuado de petróleo | Distplot


Gas licuado de petróleo | Boxplot
Gas licuado de petróleo


Gas licuado de petróleo | Probability Plot - Skew: 0.8
Ordered Values
Theoretical quantiles






```
All Plots done
Time to run AutoViz = 1 seconds

 ###################### AUTO VISUALIZATION Completed
########################
```

	Fecha	Gasolina regular	Gasolina superior	Diesel	\
0	2000-01-01	202645.200000	308156.820000	6.346671e+05	
1	2000-02-01	205530.960000	307766.310000	6.423807e+05	
2	2000-03-01	229499.560000	331910.290000	6.998072e+05	
3	2000-04-01	210680.400000	315648.080000	5.868040e+05	

4	2000-05-01	208164.340000	319667.970000	6.569482e+05
5	2000-06-01	195088.740000	300347.700000	5.709199e+05
6	2000-07-01	204556.770000	308305.930000	5.764360e+05
7	2000-08-01	218388.510000	323011.860000	5.937708e+05
8	2000-09-01	213906.960000	326455.720000	5.901454e+05
9	2000-10-01	213606.650000	323843.630000	6.411134e+05
10	2000-11-01	209227.480000	320788.080000	6.570863e+05
11	2000-12-01	230020.700000	366987.560000	6.732823e+05
12	2001-01-01	164547.351667	300242.781667	5.693288e+05
13	2001-02-01	174838.811667	301483.401667	5.977498e+05
14	2001-03-01	201992.651667	355572.831667	7.360860e+05
15	2001-04-01	198683.391667	355826.761667	6.062577e+05
16	2001-05-01	182504.091667	336040.041667	6.640827e+05
17	2001-06-01	201379.481667	308804.701667	5.593330e+05
18	2001-07-01	214980.241667	350501.971667	5.894527e+05
19	2001-08-01	204646.771667	348414.651667	5.685464e+05
20	2001-09-01	188292.351667	324393.961667	5.331084e+05
21	2001-10-01	225240.951667	366261.631667	6.033712e+05
22	2001-11-01	210324.951667	351890.631667	6.261562e+05
23	2001-12-01	223209.951667	403561.631667	6.641902e+05
24	2002-01-01	200309.851667	368935.525833	7.266726e+05
25	2002-02-01	183119.341667	348917.785833	6.943448e+05
26	2002-03-01	212711.711667	385044.965833	6.563793e+05
27	2002-04-01	204765.621667	379741.975833	6.836462e+05
28	2002-05-01	201083.511667	375054.635833	7.071655e+05
29	2002-06-01	187629.421667	371138.585833	6.678207e+05
30	2002-07-01	201692.641667	399944.645833	6.567616e+05
31	2002-08-01	205240.771667	392355.615833	6.570997e+05
32	2002-09-01	177614.801667	347191.435833	6.536646e+05
33	2002-10-01	203097.241667	381595.195833	6.530533e+05
34	2002-11-01	207629.811667	384334.215833	6.714013e+05
35	2002-12-01	199619.271667	400645.415833	6.761924e+05
36	2003-01-01	193713.620000	376516.580000	7.698433e+05
37	2003-02-01	184354.610000	355760.810000	7.097836e+05
38	2003-03-01	201443.370000	361880.870000	7.864845e+05
39	2003-04-01	219004.312500	377395.170000	7.096264e+05
40	2003-05-01	200407.180000	390624.270000	7.559821e+05
41	2003-06-01	168327.720000	346941.840000	6.415401e+05
42	2003-07-01	174317.570000	377178.840000	6.317041e+05
43	2003-08-01	171665.050000	355682.720000	6.134286e+05
44	2003-09-01	174757.950000	348659.020000	5.829895e+05
45	2003-10-01	209314.450000	392041.780000	7.227636e+05
46	2003-11-01	168510.000000	330640.230000	6.398284e+05
47	2003-12-01	196081.990000	391327.660000	6.528724e+05
48	2004-01-01	203975.570000	381547.720000	6.569874e+05
49	2004-02-01	180290.240000	339576.460000	6.595404e+05
50	2004-03-01	206455.620000	379687.460000	7.462368e+05
51	2004-04-01	182379.920000	357595.810000	6.337661e+05
52	2004-05-01	160741.900000	322596.730000	5.563697e+05

53	2004-06-01	164782.500000	318811.500000	5.603935e+05
54	2004-07-01	182642.410000	346413.660000	5.830985e+05
55	2004-08-01	207432.800000	390304.280000	6.996108e+05
56	2004-09-01	194240.550000	364689.840000	6.126396e+05
57	2004-10-01	200323.530000	380441.550000	6.190848e+05
58	2004-11-01	214509.650000	378498.600000	7.052982e+05
59	2004-12-01	236733.030000	429109.100000	7.618643e+05
60	2005-01-01	209901.600000	321382.140000	7.246924e+05
61	2005-02-01	198775.610000	375768.830000	7.171168e+05
62	2005-03-01	231358.580000	410241.928000	8.021396e+05
63	2005-04-01	225644.890000	389737.160000	8.039969e+05
64	2005-05-01	187048.260000	343762.520000	7.153604e+05
65	2005-06-01	202161.790000	351906.820000	6.817434e+05
66	2005-07-01	213821.770000	389172.650000	6.786145e+05
67	2005-08-01	212696.380000	396871.120000	6.936811e+05
68	2005-09-01	196776.870000	378887.620000	6.662584e+05
69	2005-10-01	192270.240000	328052.560000	5.898302e+05
70	2005-11-01	185251.120000	360869.140000	7.012555e+05
71	2005-12-01	241035.130000	448688.520000	7.750272e+05
72	2006-01-01	211152.720000	382078.880000	7.509464e+05
73	2006-02-01	200551.400000	367944.250000	7.345308e+05
74	2006-03-01	233082.860000	427567.580000	8.750473e+05
75	2006-04-01	223479.450000	392050.290000	7.111852e+05
76	2006-05-01	222710.900000	379832.550000	7.342892e+05
77	2006-06-01	196511.220000	364359.350000	6.724289e+05
78	2006-07-01	210255.700000	392252.910000	6.844387e+05
79	2006-08-01	211202.140000	394585.280000	7.072197e+05
80	2006-09-01	204369.990000	382724.970000	6.507914e+05
81	2006-10-01	203061.390000	400793.810000	6.835431e+05
82	2006-11-01	205541.910000	401922.570000	7.391820e+05
83	2006-12-01	233936.830000	454373.130000	7.763303e+05
84	2007-01-01	216560.790000	418412.860000	8.103727e+05
85	2007-02-01	204991.910000	401425.140000	7.993946e+05
86	2007-03-01	239019.910000	454731.090000	9.566802e+05
87	2007-04-01	217536.230000	403732.560000	7.589886e+05
88	2007-05-01	224400.730000	426414.710000	8.117037e+05
89	2007-06-01	217473.910000	390991.360000	7.208968e+05
90	2007-07-01	219874.230000	416497.410000	7.215506e+05
91	2007-08-01	236565.570000	411088.530000	7.366209e+05
92	2007-09-01	207241.840000	390568.990000	6.695879e+05
93	2007-10-01	224750.770000	419001.810000	7.387029e+05
94	2007-11-01	218116.790000	417490.770000	7.795471e+05
95	2007-12-01	252394.114286	438542.793333	7.688869e+05
96	2008-01-01	213645.390000	403266.740000	7.976689e+05
97	2008-02-01	215873.670000	407735.550000	7.940002e+05
98	2008-03-01	233761.070000	424157.280000	7.760268e+05
99	2008-04-01	233687.280000	399193.530000	8.077258e+05
100	2008-05-01	236635.850000	390882.630000	6.823262e+05
101	2008-06-01	200656.570000	337464.320000	5.076627e+05

102	2008-07-01	228640.210000	374847.050000	5.443993e+05
103	2008-08-01	227499.000000	378363.760000	5.561744e+05
104	2008-09-01	218214.640000	367962.730000	5.578026e+05
105	2008-10-01	238955.440000	386430.400000	6.178631e+05
106	2008-11-01	238626.160000	375438.690000	6.881041e+05
107	2008-12-01	298245.470000	463343.730000	7.789538e+05
108	2009-01-01	273494.160000	424264.500000	8.005265e+05
109	2009-02-01	254667.700000	424438.710000	7.950505e+05
110	2009-03-01	290343.280000	434248.690000	8.659816e+05
111	2009-04-01	290887.080000	454265.400000	8.221097e+05
112	2009-05-01	276970.670000	432401.490000	7.706873e+05
113	2009-06-01	252551.990000	393140.380000	7.038155e+05
114	2009-07-01	283142.050000	435256.090000	7.408681e+05
115	2009-08-01	267830.840000	409182.960000	6.923767e+05
116	2009-09-01	248994.420000	420238.710000	6.874351e+05
117	2009-10-01	269008.520000	421753.290000	7.757396e+05
118	2009-11-01	212149.905000	391754.810000	7.386786e+05
119	2009-12-01	296805.290000	449842.290000	8.916704e+05
120	2010-01-01	243341.490000	389068.410000	7.777413e+05
121	2010-02-01	248207.930000	391195.690000	8.076921e+05
122	2010-03-01	320985.410000	465664.590000	1.017058e+06
123	2010-04-01	254872.610000	387063.180000	7.627700e+05
124	2010-05-01	238903.310000	381811.180000	7.216682e+05
125	2010-06-01	251896.390000	385981.100000	6.832365e+05
126	2010-07-01	278700.490000	422481.800000	7.215846e+05
127	2010-08-01	268917.790000	410176.390000	6.957963e+05
128	2010-09-01	266975.250000	396429.750000	6.875539e+05
129	2010-10-01	275031.210000	418305.070000	7.454155e+05
130	2010-11-01	273452.940000	409652.730000	7.859174e+05
131	2010-12-01	320807.170000	466766.780000	8.437620e+05
132	2011-01-01	285089.040000	411737.330000	8.506968e+05
133	2011-02-01	269301.380000	391131.100000	8.199405e+05
134	2011-03-01	281453.750000	393160.910000	9.004859e+05
135	2011-04-01	285598.370000	386749.270000	7.975272e+05
136	2011-05-01	253277.880000	371876.750000	8.038495e+05
137	2011-06-01	290137.490000	380800.430000	7.296768e+05
138	2011-07-01	279283.360000	389294.640000	7.182551e+05
139	2011-08-01	256380.270000	408494.510000	6.997361e+05
140	2011-09-01	256728.250000	380453.230000	6.603849e+05
141	2011-10-01	242071.010000	381309.500000	6.722149e+05
142	2011-11-01	231413.830000	381871.840000	7.945072e+05
143	2011-12-01	291678.030000	449904.100000	8.623188e+05
144	2012-01-01	261933.680000	393781.940000	8.149567e+05
145	2012-02-01	253733.220000	402633.550000	8.480672e+05
146	2012-03-01	290286.650000	429534.500000	9.250653e+05
147	2012-04-01	262615.280000	389264.020000	7.922017e+05
148	2012-05-01	235374.040000	405877.410000	7.715425e+05
149	2012-06-01	261012.580000	409111.240000	7.124649e+05
150	2012-07-01	233061.350000	420370.600000	6.991300e+05

151	2012-08-01	259165.940000	433828.020000	7.167204e+05
152	2012-09-01	228205.770000	378478.800000	6.321204e+05
153	2012-10-01	261347.220000	414523.220000	7.284149e+05
154	2012-11-01	272657.820000	394751.470000	8.125234e+05
155	2012-12-01	310407.040000	442191.430000	8.482629e+05
156	2013-01-01	287985.590000	408557.450000	8.637694e+05
157	2013-02-01	259659.660000	361215.690000	8.088540e+05
158	2013-03-01	313075.450000	413744.860000	8.983522e+05
159	2013-04-01	330070.650000	409238.300000	9.661039e+05
160	2013-05-01	313131.360000	403354.790000	8.276012e+05
161	2013-06-01	287596.640000	402012.070000	7.751871e+05
162	2013-07-01	314185.610000	425154.220000	7.325445e+05
163	2013-08-01	288400.670000	416968.150000	7.731479e+05
164	2013-09-01	249357.420000	379479.780000	6.721252e+05
165	2013-10-01	314518.720000	430179.690000	8.000491e+05
166	2013-11-01	306638.450000	404692.460000	8.214575e+05
167	2013-12-01	339969.850000	448135.350000	8.603519e+05
168	2014-01-01	312139.770000	412617.930000	9.172515e+05
169	2014-02-01	312142.650000	385430.580000	8.903623e+05
170	2014-03-01	335869.630000	434863.960000	1.004937e+06
171	2014-04-01	349155.300000	437470.070000	9.720394e+05
172	2014-05-01	332737.370000	427487.200000	8.772633e+05
173	2014-06-01	303016.820000	387299.100000	7.335087e+05
174	2014-07-01	344659.640000	439137.130000	8.269188e+05
175	2014-08-01	326356.060000	442953.100000	7.936940e+05
176	2014-09-01	371327.130000	411466.280000	7.284988e+05
177	2014-10-01	343812.670000	459922.480000	8.362781e+05
178	2014-11-01	343918.560000	436467.900000	8.857137e+05
179	2014-12-01	464223.150000	538899.260000	1.021378e+06
180	2015-01-01	419102.720000	503135.700000	1.014706e+06
181	2015-02-01	398167.690000	454190.880000	9.801498e+05
182	2015-03-01	457246.140000	521879.260000	1.142551e+06
183	2015-04-01	413451.110000	486779.410000	1.011168e+06
184	2015-05-01	415570.950000	483594.830000	9.443167e+05
185	2015-06-01	414466.180000	480892.110000	8.800165e+05
186	2015-07-01	440729.530000	515322.660000	9.025864e+05
187	2015-08-01	439588.900000	513413.000000	8.877968e+05
188	2015-09-01	422826.585476	501658.043333	7.346542e+05
189	2015-10-01	448971.880000	532741.140000	8.342859e+05
190	2015-11-01	423841.820000	516486.950000	9.254310e+05
191	2015-12-01	499839.170000	600161.720000	1.047479e+06
192	2016-01-01	433180.340000	545737.530000	1.084044e+06
193	2016-02-01	450916.350000	540459.190000	1.055673e+06
194	2016-03-01	486255.940000	575720.920000	1.107297e+06
195	2016-04-01	467472.200000	545605.470000	1.149175e+06
196	2016-05-01	468710.380000	551000.790000	1.019984e+06
197	2016-06-01	450758.180000	537583.250000	8.885381e+05
198	2016-07-01	465270.480000	562778.840000	8.837160e+05
199	2016-08-01	485995.830000	585748.500000	8.662700e+05

200	2016-09-01	456946.750000	548491.650000	8.778006e+05
201	2016-10-01	476277.640000	573754.730000	9.164107e+05
202	2016-11-01	463726.210000	547323.530000	1.019672e+06
203	2016-12-01	524242.310000	615251.510000	1.113827e+06
204	2017-01-01	477291.930000	546212.160000	1.079339e+06
205	2017-02-01	453665.320000	526888.940000	1.052536e+06
206	2017-03-01	523943.600000	592249.340000	1.216423e+06
207	2017-04-01	500063.030000	530345.960000	1.032813e+06
208	2017-05-01	516884.370000	588861.620000	1.034806e+06
209	2017-06-01	478491.950000	549378.050000	8.975119e+05
210	2017-07-01	521730.410000	607702.700000	9.388876e+05
211	2017-08-01	514086.570000	604650.340000	9.501881e+05
212	2017-09-01	492618.550000	560892.670000	8.943636e+05
213	2017-10-01	512762.320000	583658.830000	9.359862e+05
214	2017-11-01	510631.290000	563345.610000	1.038598e+06
215	2017-12-01	574097.700000	635423.760000	1.117711e+06
216	2018-01-01	528763.300000	581802.980000	1.105629e+06
217	2018-02-01	502041.670000	545210.000000	1.072072e+06
218	2018-03-01	571052.130000	618402.180000	1.194136e+06
219	2018-04-01	526979.670000	565665.720000	1.108453e+06
220	2018-05-01	544395.170000	570129.660000	1.009750e+06
221	2018-06-01	518911.440000	554431.500000	9.300650e+05
222	2018-07-01	541811.120000	589775.970000	9.503435e+05
223	2018-08-01	570224.107000	593659.630000	9.918181e+05
224	2018-09-01	531285.291000	551547.830000	8.823035e+05
225	2018-10-01	573751.472000	579474.430000	9.690950e+05
226	2018-11-01	563417.572000	573687.350000	1.061942e+06
227	2018-12-01	632670.174000	654281.250000	1.106130e+06
228	2019-01-01	596682.750000	601714.900000	1.152950e+06
229	2019-02-01	559109.220000	565425.020000	1.103358e+06
230	2019-03-01	645266.770000	610720.580000	1.228025e+06
231	2019-04-01	657570.180000	598956.300000	1.160125e+06
232	2019-05-01	666748.630000	611592.320000	1.134666e+06
233	2019-06-01	641136.560000	595744.380000	1.012030e+06
234	2019-07-01	650150.780000	615559.330000	1.030650e+06
235	2019-08-01	668783.390000	632415.700000	1.070198e+06
236	2019-09-01	616079.300000	587861.840000	9.733584e+05
237	2019-10-01	669008.770000	643044.330000	1.043630e+06
238	2019-11-01	643070.350000	620572.210000	1.122735e+06
239	2019-12-01	706020.770000	682468.140000	1.175725e+06
240	2020-01-01	640011.730000	622323.790000	1.184419e+06
241	2020-02-01	652028.000000	633100.050000	1.233250e+06
242	2020-03-01	573586.730000	509752.950000	1.100814e+06
243	2020-04-01	458420.850000	391773.230000	8.675923e+05
244	2020-05-01	478771.830000	407482.520000	8.287439e+05
245	2020-06-01	516933.600000	388952.810000	7.823280e+05
246	2020-07-01	567052.500000	462380.810000	8.703549e+05
247	2020-08-01	647375.580000	574379.330000	8.852007e+05
248	2020-09-01	661665.580000	610445.100000	9.590133e+05

249	2020-10-01	680362.070000	648582.820000	1.064005e+06
250	2020-11-01	652630.160000	639856.620000	1.072873e+06
251	2020-12-01	768193.030000	760695.450000	1.276291e+06
252	2021-01-01	685942.970000	655512.660000	1.220638e+06
253	2021-02-01	669312.880000	621712.090000	1.272609e+06
254	2021-03-01	773453.520000	692240.490000	1.417380e+06
255	2021-04-01	688389.030000	610310.750000	1.268491e+06
256	2021-05-01	707336.380000	643076.840000	1.188223e+06
257	2021-06-01	692598.710000	637332.810000	1.128077e+06
258	2021-07-01	740193.130000	665472.940000	1.197398e+06
259	2021-08-01	668221.540000	636824.610000	1.104626e+06
260	2021-09-01	670418.800000	605257.760000	1.102655e+06
261	2021-10-01	718676.320000	675821.540000	1.166983e+06
262	2021-11-01	708246.270000	641916.400000	1.223817e+06
263	2021-12-01	821882.150000	768102.830000	1.380065e+06
264	2022-01-01	683031.840000	621512.780000	1.281083e+06
265	2022-02-01	661521.580000	592106.140000	1.267195e+06
266	2022-03-01	701819.650000	598038.470000	1.335368e+06
267	2022-04-01	812059.060000	569199.430000	1.288233e+06
268	2022-05-01	791824.980000	483991.630000	1.109577e+06
269	2022-06-01	781009.470000	507260.420000	1.102689e+06
270	2022-07-01	788841.460000	554496.900000	1.125984e+06
271	2022-08-01	784468.320000	581844.420000	1.107349e+06
272	2022-09-01	760396.150000	570632.200000	1.202883e+06
273	2022-10-01	768784.360000	573943.480000	9.792433e+05
274	2022-11-01	771173.120000	572163.230000	1.165294e+06
275	2022-12-01	865851.440000	635953.460000	1.241282e+06
276	2023-01-01	804524.730000	599501.290000	1.285933e+06
277	2023-02-01	737345.640000	544144.000000	1.226885e+06
278	2023-03-01	875010.340000	656941.640000	1.463008e+06
279	2023-04-01	798128.360000	586642.200000	1.247899e+06
280	2023-05-01	866826.790000	646221.200000	1.346554e+06
281	2023-06-01	782745.170000	616290.210000	1.159046e+06
282	2023-07-01	825219.050000	642098.160000	1.178454e+06
283	2023-08-01	851754.020000	632333.400000	1.222453e+06
284	2023-09-01	799852.730000	597670.260000	1.103622e+06
285	2023-10-01	728741.160000	531542.580000	9.981134e+05
286	2023-11-01	807947.980000	632180.030000	1.248224e+06
287	2023-12-01	880233.120000	691389.620000	1.334692e+06
288	2024-01-01	830708.130000	658083.660000	1.359012e+06
289	2024-02-01	818740.160000	654059.600000	1.340174e+06
290	2024-03-01	870771.700000	671997.050000	1.393325e+06
291	2024-04-01	847353.150000	633520.570000	1.428143e+06
292	2024-05-01	894533.140000	692427.940000	1.401052e+06

	Gas licuado de petróleo
0	194410.476190
1	174710.552381
2	189234.066667

3	174330.607143
4	191745.147619
5	196321.933333
6	192344.502381
7	203727.080952
8	194565.097619
9	210261.826190
10	205027.952619
11	200247.555000
12	179405.990278
13	167818.137897
14	186397.325992
15	175820.345040
16	178303.330754
17	177826.928373
18	191041.990278
19	190295.990278
20	184039.990278
21	206176.990278
22	201335.990278
23	209792.990278
24	196068.608730
25	180209.801587
26	186680.965873
27	196167.161111
28	201612.356349
29	193024.399206
30	204196.244444
31	213625.432540
32	209849.758730
33	212932.327778
34	210000.546825
35	223830.396825
36	213268.978571
37	206815.011905
38	216427.950000
39	204403.511905
40	216452.752143
41	212069.040476
42	231172.014286
43	222312.707143
44	225703.050000
45	239665.042857
46	229612.354762
47	247544.638095
48	235284.965714
49	213288.683333
50	233426.354762
51	217530.876190

52	225959.050238
53	225673.464048
54	243064.072857
55	236324.768095
56	225204.180238
57	229626.058333
58	225593.823095
59	241915.005238
60	227154.969762
61	211814.375952
62	224387.028333
63	218555.206429
64	230568.152381
65	223151.259524
66	236140.078571
67	242734.740476
68	223283.869048
69	213139.580952
70	222988.526190
71	225410.429048
72	217867.309524
73	212328.607143
74	232405.357143
75	207612.269048
76	232489.833333
77	236563.828571
78	239767.854762
79	245672.016667
80	256040.102381
81	238514.633333
82	228417.438095
83	236333.111905
84	217344.655952
85	211149.360714
86	264115.983333
87	224818.876190
88	259749.480952
89	244505.283333
90	261555.621429
91	267634.121429
92	254081.758571
93	264936.659524
94	250108.611905
95	253659.528571
96	252508.333333
97	241978.083333
98	247679.038095
99	243178.030952
100	240798.459524

101	242422.571429
102	243531.971429
103	237062.519048
104	225262.458571
105	263100.502381
106	230092.621429
107	245852.645238
108	217372.416667
109	212038.241928
110	235475.316667
111	209739.173810
112	221884.466190
113	221329.626190
114	241510.204762
115	232174.109524
116	225886.426190
117	235899.133333
118	232172.833333
119	215923.378571
120	228078.623810
121	214409.685714
122	242081.585714
123	216150.666667
124	215523.538095
125	226415.073810
126	243063.350000
127	245908.416667
128	233404.895238
129	236852.345238
130	255841.845238
131	251825.204762
132	244328.914286
133	228478.995238
134	255792.916667
135	237582.676190
136	247975.276190
137	247082.276190
138	259033.888095
139	259688.742857
140	251882.535714
141	262255.138095
142	252218.214286
143	263892.433333
144	242524.898571
145	243810.952381
146	270906.083810
147	249493.658571
148	261657.406667
149	255333.426190

150	271101.936190
151	280184.268571
152	268875.095238
153	297829.174048
154	303422.012381
155	281757.561905
156	291843.997619
157	253221.892857
158	286869.621429
159	287216.303333
160	301770.525476
161	275921.369048
162	292790.104762
163	282844.319048
164	278403.420238
165	285195.011905
166	253022.225476
167	282335.469524
168	288260.804286
169	304312.128571
170	284201.430952
171	250902.831190
172	305911.840476
173	285640.435714
174	289827.598571
175	272617.597143
176	271079.594762
177	337006.144524
178	303666.704286
179	311419.960476
180	317220.912857
181	315364.363333
182	338142.317143
183	320920.883810
184	328177.803810
185	331140.045714
186	363251.960000
187	333215.607143
188	364351.758095
189	364469.545714
190	353643.078571
191	355233.649048
192	346512.219048
193	368867.548095
194	372223.424048
195	357084.223333
196	377629.016905
197	355848.993333
198	350429.766667
199	363859.513095

200	356426.233571
201	373503.775714
202	372904.634286
203	372690.455714
204	358706.818571
205	357497.970229
206	351791.925033
207	360957.704524
208	377492.160714
209	378328.463810
210	369948.637476
211	385834.284762
212	447373.592143
213	393901.921429
214	409332.858095
215	404574.361905
216	421848.920476
217	379239.067143
218	415125.896429
219	370434.605238
220	403094.744048
221	390070.171667
222	381128.075238
223	407124.746190
224	385717.790714
225	408674.336524
226	408049.579000
227	421872.087524
228	421555.410000
229	402371.330000
230	428149.450000
231	414018.680000
232	426587.330000
233	399031.610000
234	419958.020000
235	419493.820000
236	396860.640000
237	433917.150000
238	430725.890000
239	428697.640000
240	421644.720000
241	426533.150000
242	420637.140000
243	345933.610000
244	362987.370000
245	402541.020000
246	423328.220000
247	483695.900000
248	439581.360000

```
249       459312.450000
250       421504.590000
251       481068.860000
252       485759.130000
253       447847.050000
254       513966.390000
255       466275.270000
256       475018.360000
257       475462.790000
258       500478.600000
259       492629.810000
260       493058.470000
261       497279.700000
262       480432.290000
263       531775.930000
264       513015.930000
265       519921.040000
266       483657.430000
267       493178.400000
268       515616.020000
269       513914.270000
270       513619.250000
271       541273.940000
272       479529.680000
273       545823.450000
274       528393.770000
275       556737.220000
276       574148.000000
277       497273.190000
278       588892.160000
279       513919.890000
280       561767.170000
281       532138.900000
282       521833.560000
283       543825.640000
284       517699.480000
285       495261.140000
286       530542.420000
287       541263.180000
288       548124.450000
289       526897.850000
290       523990.910000
291       531880.190000
292       536754.380000

#report = sv.analyze(dataConsumo)
#report.show_html('Consumo/EDAConsumo.html')
```

Dataset: IMPORTACION-VOLUMEN-2024-05.xlsx

Este dataset cuenta con una estructura similar a la de datos de consumo, por lo que tambien se combinan las columnas de diesel.

```
csvPath = 'Importacion/IMPORTACION-VOLUMEN-2024-05.csv'

dataImportacion = combineDieselColumns(csvPath)
```

```
 Iniciando la combinación de las columnas de Diesel...
 Archivo cargado: Importacion/IMPORTACION-VOLUMEN-2024-05.csv
 Combinación de columnas completada.
```

```
dataImportacion = dataImportacion[['Fecha', 'Gasolina regular',
'Gasolina superior', 'Diesel', 'Gas licuado de petróleo']]
dataImportacion['Fecha'] = pd.to_datetime(dataImportacion['Fecha'])
dataImportacion.head()
```

```
    Fecha     Gasolina regular  Gasolina superior   Diesel    \
0 2001-01-01      177776.50         373963.96      566101.99
1 2001-02-01      123115.99         243091.07       489525.8
2 2001-03-01      161726.42         312084.38      575559.68
3 2001-04-01      127338.74         285054.89      437745.42
4 2001-05-01      168730.19         300913.67      552609.13

   Gas licuado de petróleo
0        194065.738095
1        170703.380952
2        161837.371429
3        163048.642857
4        171518.861905
```

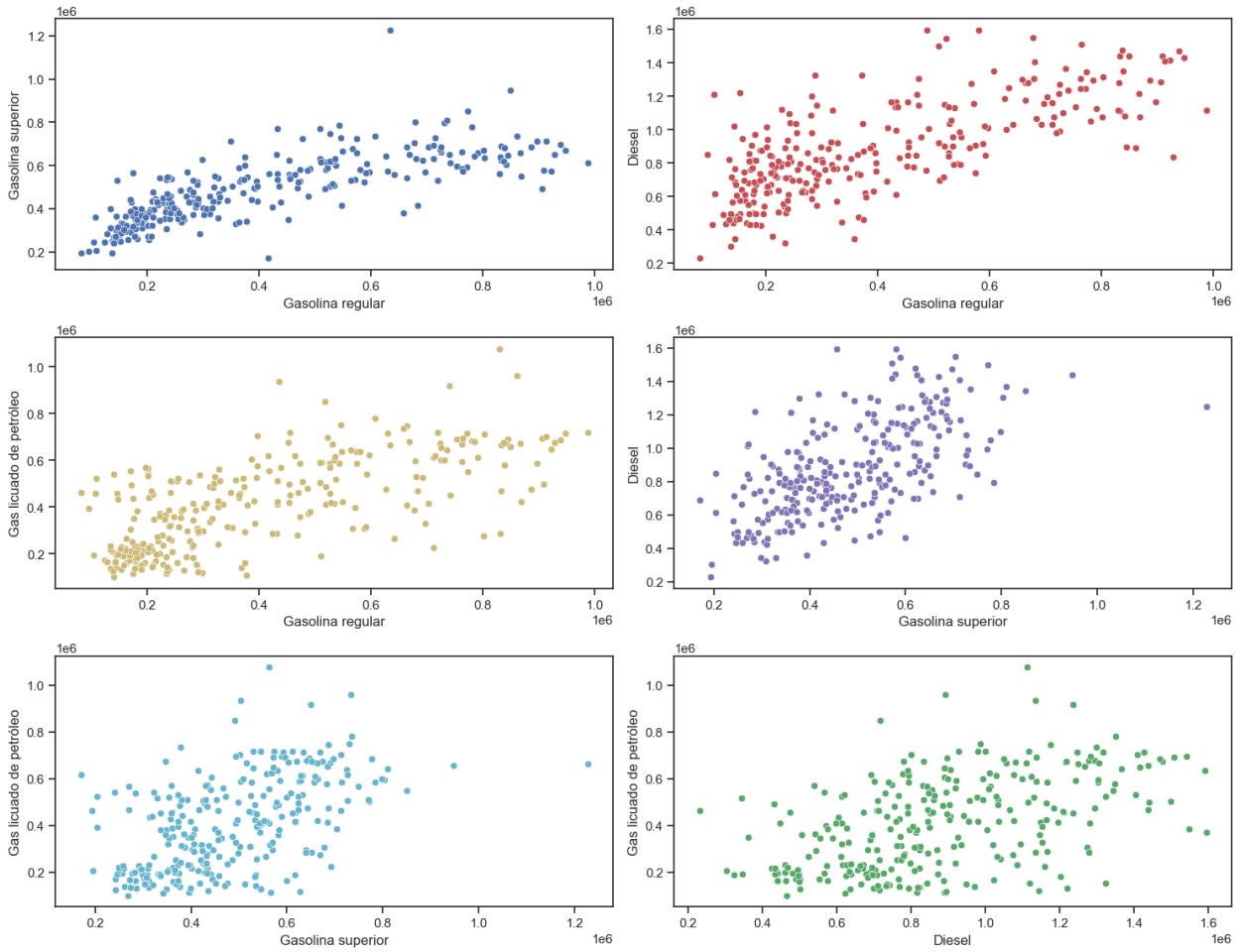
En este EDA se destaca que en el dataset de Importación, similarmente al dataset de Consumo, todas las variables cuentan con distribuciones asimétricas positivas, aunque algunas cuentan con una menor asimetría. Los gráficos de caja y bigotes indican la presencia de valores atipicos principalmente en *Gasolina superior* y *Gas licuado de petróleo*. Al igual que el dataset anterior, los Q-Q plots muestran que ninguna de las variables cuenta con una distribución normal.

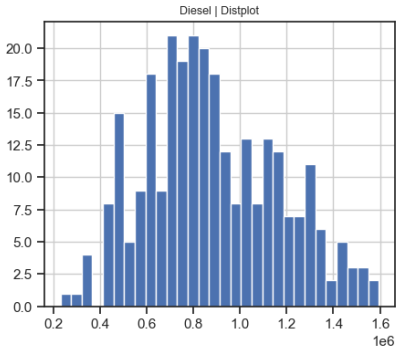
```
csvPath = 'Importacion/dataImportacion.csv'
dataImportacion.to_csv(csvPath, index=False)
AV = AutoViz_Class()
AV.AutoViz(csvPath)
```

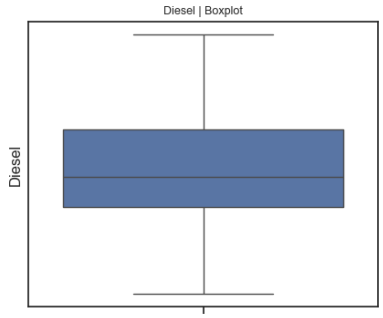
```
Shape of your Data Set loaded: (281, 5)
############################################################################
#################
######################## C L A S S I F Y I N G  V A R I A B L E S
```

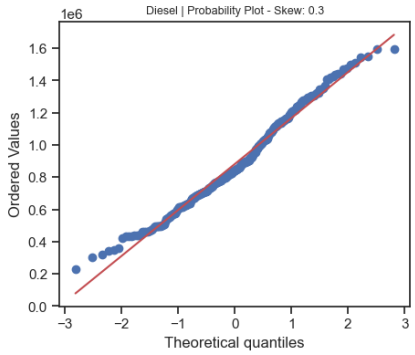
```
####################
#########################################################################
#################
Classifying variables in data set...
    Number of Numeric Columns =  4
    Number of Integer-Categorical Columns =  0
    Number of String-Categorical Columns =  0
    Number of Factor-Categorical Columns =  0
    Number of String-Boolean Columns =  0
    Number of Numeric-Boolean Columns =  0
    Number of Discrete String Columns =  0
    Number of NLP String Columns =  0
    Number of Date Time Columns =  0
    Number of ID Columns =  1
    Number of Columns to Delete =  0
    5 Predictors classified...
        1 variable(s) removed since they were ID or low-information
variables
        List of variables removed: ['Fecha']
To fix these data quality issues in the dataset, import FixDQ from
autoviz...
    All variables classified into correct types.

<pandas.io.formats.style.Styler at 0x15462526590>

Number of All Scatter Plots = 10
```

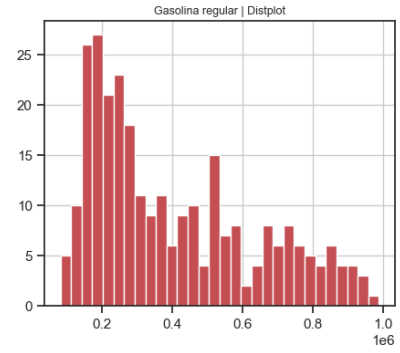
Pair-wise Scatter Plot of all Continuous Variables

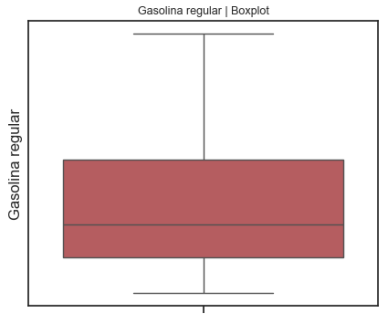


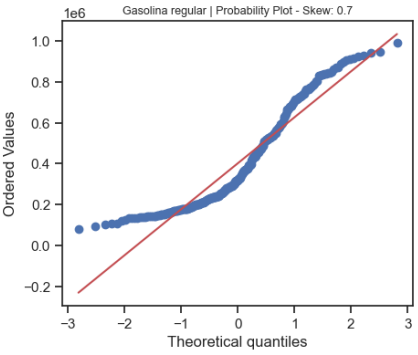

Diesel | Distplot
1e6

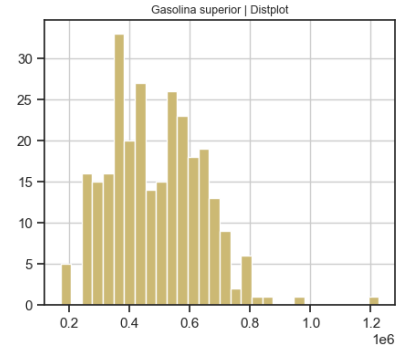

Diesel | Boxplot
Diesel

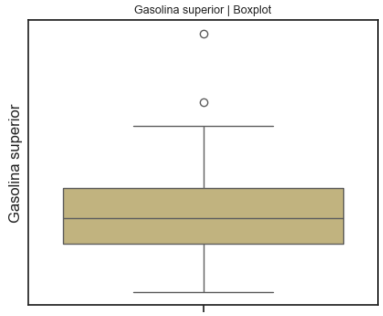

Diesel | Probability Plot - Skew: 0.3
1e6
Ordered Values
Theoretical quantiles

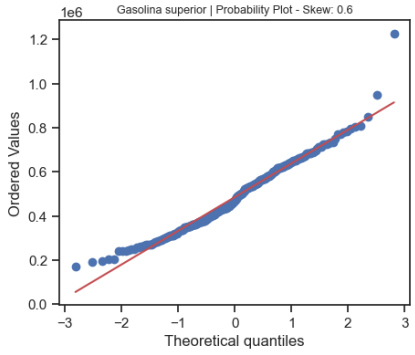

Gasolina regular | Distplot
1e6

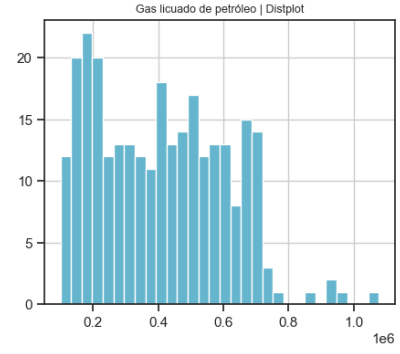

Gasolina regular | Boxplot
Gasolina regular

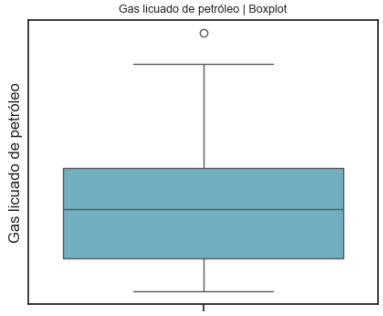

Gasolina regular | Probability Plot - Skew: 0.7
1e6
Ordered Values
Theoretical quantiles

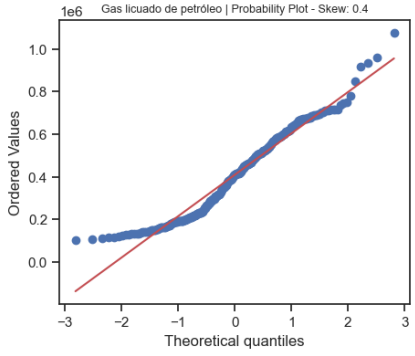

Gasolina superior | Distplot
1e6

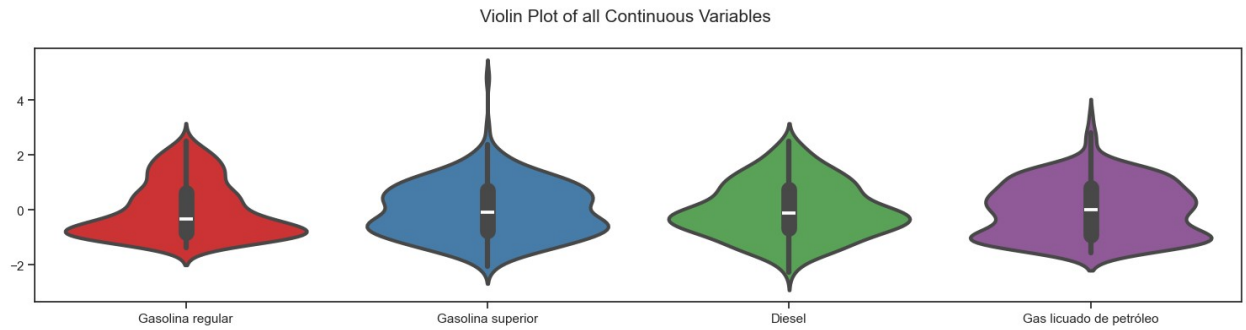

Gasolina superior | Boxplot
Gasolina superior

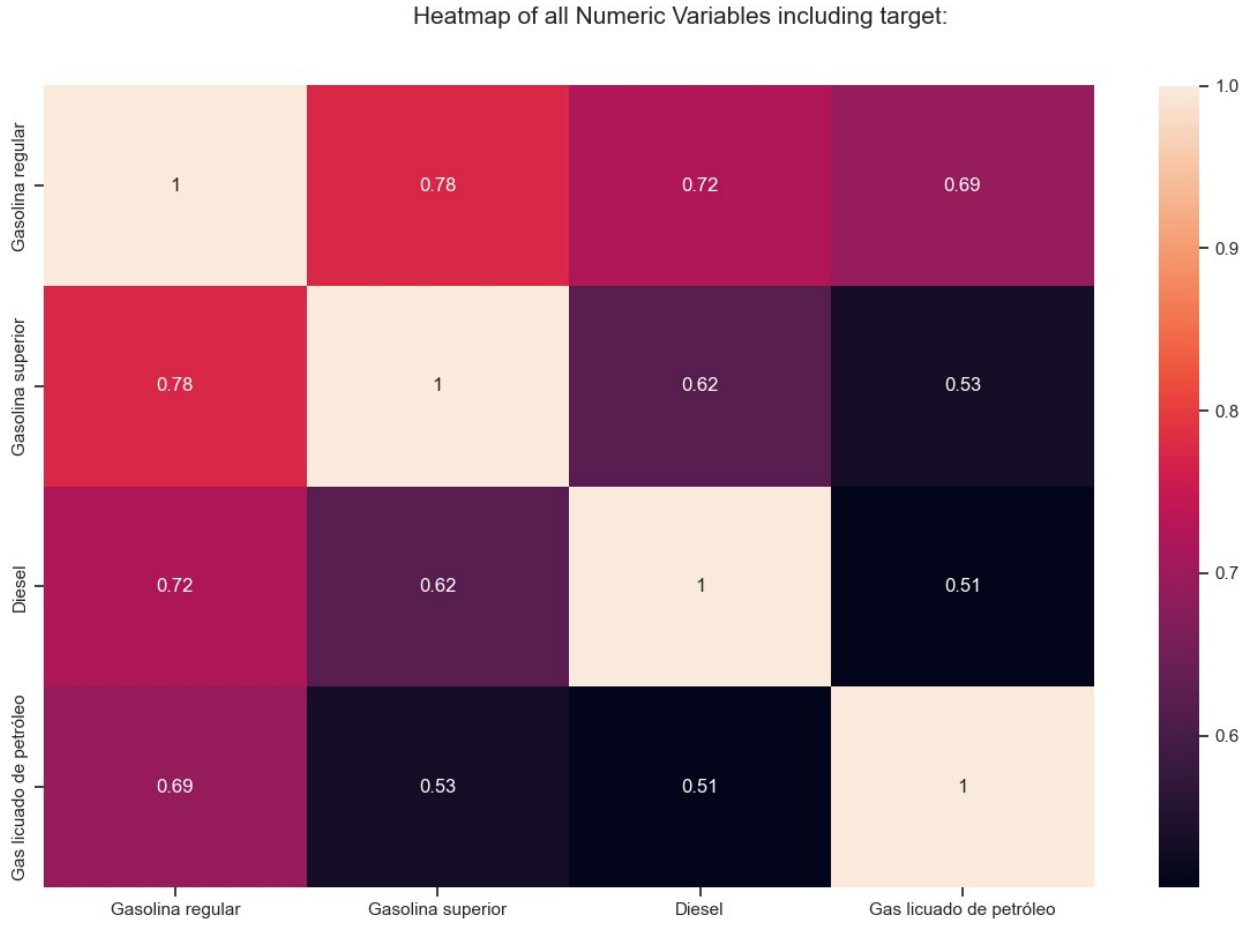

Gasolina superior | Probability Plot - Skew: 0.6
1e6
Ordered Values
Theoretical quantiles


Gas licuado de petróleo | Distplot
1e6


Gas licuado de petróleo | Boxplot
Gas licuado de petróleo


Gas licuado de petróleo | Probability Plot - Skew: 0.4
1e6
Ordered Values
Theoretical quantiles






```
All Plots done
Time to run AutoViz = 1 seconds

 ###################### AUTO VISUALIZATION Completed
########################
```

	Fecha	Gasolina regular	Gasolina superior	Diesel	\
0	2001-01-01	177776.500000	373963.960	566101.990	
1	2001-02-01	123115.990000	243091.070	489525.800	
2	2001-03-01	161726.420000	312084.380	575559.680	
3	2001-04-01	127338.740000	285054.890	437745.420	

4	2001-05-01	168730.190000	300913.670	552609.130
5	2001-06-01	152899.090000	333217.190	497855.260
6	2001-07-01	136299.130000	195071.860	302350.020
7	2001-08-01	139365.070000	268153.260	464159.130
8	2001-09-01	233643.750000	308439.070	321952.940
9	2001-10-01	141550.220000	305102.280	438989.550
10	2001-11-01	165841.460000	256638.190	440245.350
11	2001-12-01	149217.670000	277145.050	479983.270
12	2002-01-01	178585.930000	271479.000	500041.320
13	2002-02-01	144447.250000	365657.750	686923.830
14	2002-03-01	104673.640000	246292.600	432538.460
15	2002-04-01	190339.000000	310256.300	424667.360
16	2002-05-01	194830.100000	363311.760	497781.660
17	2002-06-01	137050.800000	250123.190	496826.620
18	2002-07-01	222408.000000	428842.800	636727.840
19	2002-08-01	143795.000000	298280.330	346225.000
20	2002-09-01	156956.900000	331053.700	638933.220
21	2002-10-01	195426.800000	352055.500	703749.810
22	2002-11-01	153185.700000	291188.760	573289.500
23	2002-12-01	167596.710000	440677.810	678749.640
24	2003-01-01	228269.500000	425185.500	797344.380
25	2003-02-01	214490.700000	443849.850	765912.660
26	2003-03-01	161257.480000	363352.660	693942.310
27	2003-04-01	189824.893000	335444.690	873103.096
28	2003-05-01	269981.860000	459025.690	625190.080
29	2003-06-01	182254.230000	372486.530	671882.830
30	2003-07-01	133668.200000	400922.970	822014.550
31	2003-08-01	169903.860000	292365.040	494087.130
32	2003-09-01	177217.530000	398192.160	614913.680
33	2003-10-01	181668.026000	260219.920	433084.990
34	2003-11-01	205624.080000	393951.960	627189.780
35	2003-12-01	161477.730000	296780.150	745524.880
36	2004-01-01	173427.230000	368235.140	762385.450
37	2004-02-01	147939.230000	281891.970	460947.500
38	2004-03-01	275539.380000	486880.820	772128.260
39	2004-04-01	245156.340000	415426.000	809194.950
40	2004-05-01	178387.800000	350052.950	634209.640
41	2004-06-01	133341.000000	311238.010	461381.790
42	2004-07-01	152075.750000	249710.687	467154.310
43	2004-08-01	208592.850000	434578.120	669462.530
44	2004-09-01	218499.240000	420550.290	714039.850
45	2004-10-01	201907.670000	366886.300	655715.050
46	2004-11-01	215838.320000	434710.940	726697.880
47	2004-12-01	227094.710000	433149.960	781266.610
48	2005-01-01	202374.330000	257867.990	663058.790
49	2005-02-01	178104.720000	306660.390	683664.390
50	2005-03-01	217667.690000	431907.810	778645.070
51	2005-04-01	234492.030000	534151.810	708164.740
52	2005-05-01	192734.400000	378616.300	875087.620

53	2005-06-01	243343.680000	537398.430	1041016.720
54	2005-07-01	175946.300000	363410.900	770731.740
55	2005-08-01	173810.380000	566618.550	815435.200
56	2005-09-01	81015.300000	193484.500	229764.740
57	2005-10-01	145325.650000	528877.090	671848.940
58	2005-11-01	266018.370000	348687.810	635553.460
59	2005-12-01	107374.000000	360807.000	1212610.530
60	2006-01-01	348503.850000	711860.770	708730.840
61	2006-02-01	145048.200000	262170.900	767710.630
62	2006-03-01	285783.250000	408065.340	744721.940
63	2006-04-01	238824.610000	421300.820	935306.580
64	2006-05-01	134212.300000	312024.700	822556.450
65	2006-06-01	211562.740000	394014.850	361207.590
66	2006-07-01	180942.180000	375951.160	971716.720
67	2006-08-01	147245.490000	313326.860	603747.330
68	2006-09-01	366193.770000	525623.150	473431.310
69	2006-10-01	153997.350000	285505.000	1220719.640
70	2006-11-01	171327.010000	428590.500	431874.860
71	2006-12-01	234319.300000	395132.090	890336.680
72	2007-01-01	289550.590000	523730.310	818060.570
73	2007-02-01	299077.040000	401855.410	822325.650
74	2007-03-01	326317.750000	541817.850	569451.850
75	2007-04-01	94844.900000	203441.330	850138.110
76	2007-05-01	238404.830000	545587.630	793887.040
77	2007-06-01	170064.980000	441537.390	855757.140
78	2007-07-01	227660.470000	451180.820	1118432.250
79	2007-08-01	351306.270000	501840.270	824047.470
80	2007-09-01	181015.830000	336120.810	497008.840
81	2007-10-01	312120.180000	376377.750	885924.860
82	2007-11-01	289282.550000	512550.510	665114.830
83	2007-12-01	222108.350000	542916.970	812776.590
84	2008-01-01	107885.650000	204788.300	613819.330
85	2008-02-01	234830.100000	545139.830	500922.230
86	2008-03-01	241800.330000	405713.480	1095743.350
87	2008-04-01	271554.270000	570256.980	682042.070
88	2008-05-01	242628.160000	424954.880	710464.760
89	2008-06-01	177838.700000	273398.730	817804.840
90	2008-07-01	232129.200000	384549.700	541486.150
91	2008-08-01	138376.720000	241881.470	713650.520
92	2008-09-01	290621.260000	423244.170	544521.540
93	2008-10-01	358036.440000	329762.300	344273.020
94	2008-11-01	170365.320000	386365.050	809439.420
95	2008-12-01	238398.920000	458949.420	524038.420
96	2009-01-01	202484.534762	382300.560	941535.710
97	2009-02-01	588346.360000	522820.810	914059.400
98	2009-03-01	298459.250000	626630.990	893473.800
99	2009-04-01	182218.960000	348267.660	816840.020
100	2009-05-01	377337.480000	341700.480	621339.930
101	2009-06-01	363676.290000	589337.790	735260.510

102	2009-07-01	363911.510000	337172.650	850806.890
103	2009-08-01	210495.680000	476525.360	842742.170
104	2009-09-01	161783.680000	321728.950	611966.220
105	2009-10-01	375067.230000	639524.610	869366.310
106	2009-11-01	201213.100000	321859.440	894309.880
107	2009-12-01	291525.480000	435416.180	1144960.660
108	2010-01-01	173918.350000	318949.720	905803.650
109	2010-02-01	336005.720000	492452.550	447271.920
110	2010-03-01	157798.770000	303762.940	947702.880
111	2010-04-01	433108.230000	562996.070	613011.210
112	2010-05-01	279926.950000	429729.380	1086036.160
113	2010-06-01	281834.460000	464310.800	587828.260
114	2010-07-01	241994.240000	456588.390	712889.930
115	2010-08-01	282509.240000	475714.120	637252.290
116	2010-09-01	208796.020000	321134.560	741721.420
117	2010-10-01	392100.620000	493794.290	630692.560
118	2010-11-01	139547.730000	241062.760	565627.540
119	2010-12-01	430857.950000	652210.770	1135293.920
120	2011-01-01	253700.820000	391997.540	1033948.690
121	2011-02-01	343582.850000	437769.480	772775.580
122	2011-03-01	291087.160000	445920.270	707536.310
123	2011-04-01	326003.710000	357349.670	767445.360
124	2011-05-01	326566.610000	433038.130	918940.440
125	2011-06-01	230990.130000	354604.940	553883.420
126	2011-07-01	188581.400000	369694.760	734958.540
127	2011-08-01	201844.490000	270765.310	1014324.280
128	2011-09-01	260669.220000	335803.820	795554.710
129	2011-10-01	239711.910000	351555.660	612133.260
130	2011-11-01	237803.480000	450212.800	575406.480
131	2011-12-01	367820.540000	534925.360	732921.030
132	2012-01-01	318283.440000	434648.170	790123.230
133	2012-02-01	253583.960000	372893.490	887963.980
134	2012-03-01	255492.160000	354160.850	978693.290
135	2012-04-01	189948.790000	399701.690	697871.280
136	2012-05-01	241786.290000	380531.850	833370.090
137	2012-06-01	320566.690000	550379.330	767124.650
138	2012-07-01	142232.470000	271509.200	1021995.900
139	2012-08-01	263677.230000	361307.200	626706.230
140	2012-09-01	201347.560000	372989.290	619651.270
141	2012-10-01	374301.510000	600094.030	463054.770
142	2012-11-01	197398.800000	359818.540	538552.890
143	2012-12-01	287980.580000	419029.170	1324780.960
144	2013-01-01	396416.840000	433924.690	749266.680
145	2013-02-01	235502.580000	470483.980	709843.020
146	2013-03-01	439386.620000	429974.170	832627.570
147	2013-04-01	274680.650000	430259.180	860094.780
148	2013-05-01	281823.200000	532379.660	1202187.850
149	2013-06-01	293808.600000	284038.030	729711.880
150	2013-07-01	259025.830000	486075.060	693160.500

151	2013-08-01	221433.200000	347066.110	505133.440
152	2013-09-01	300794.990000	373303.310	803626.980
153	2013-10-01	281168.850000	372946.330	980327.980
154	2013-11-01	390941.150000	496029.720	903218.470
155	2013-12-01	210096.780000	358799.640	791698.890
156	2014-01-01	487880.790000	457365.460	1595698.560
157	2014-02-01	312423.860000	455726.060	688703.180
158	2014-03-01	472437.630000	582572.930	1147182.500
159	2014-04-01	207049.000000	270302.220	845051.330
160	2014-05-01	417767.240000	563319.790	1008148.070
161	2014-06-01	247114.990000	404603.550	694920.680
162	2014-07-01	454196.130000	558403.410	845875.860
163	2014-08-01	305285.530000	398563.480	760353.370
164	2014-09-01	378741.560000	547486.220	598145.890
165	2014-10-01	256340.490000	379572.070	762763.240
166	2014-11-01	372809.930000	533524.400	1033873.160
167	2014-12-01	491072.110000	583081.570	1004032.650
168	2015-01-01	371420.590000	471973.300	1324421.960
169	2015-02-01	548714.530000	664315.850	955286.020
170	2015-03-01	453536.190000	622883.100	1164885.970
171	2015-04-01	537064.590000	599558.550	1130870.790
172	2015-05-01	534049.280000	618319.210	954782.750
173	2015-06-01	394824.330000	525280.850	862071.770
174	2015-07-01	519374.150000	616269.010	1149630.430
175	2015-08-01	328552.070000	467373.130	919285.900
176	2015-09-01	485906.490000	576250.290	754068.770
177	2015-10-01	432666.750000	533620.640	1163425.950
178	2015-11-01	574547.760000	724273.970	904700.420
179	2015-12-01	319329.710000	498798.790	1113876.280
180	2016-01-01	546876.540000	729211.430	987507.780
181	2016-02-01	424074.810000	405559.450	1167227.960
182	2016-03-01	508542.180000	771024.110	1498844.950
183	2016-04-01	527083.640000	502728.880	1137213.100
184	2016-05-01	528347.810000	615900.310	1035544.080
185	2016-06-01	397477.400000	502082.450	800222.530
186	2016-07-01	526652.730000	618366.100	859222.620
187	2016-08-01	512202.570000	625218.900	701185.750
188	2016-09-01	567786.570000	677892.970	1274128.310
189	2016-10-01	455850.760000	546154.650	927923.500
190	2016-11-01	476784.160000	725548.160	957656.050
191	2016-12-01	473659.590000	526573.960	1303735.780
192	2017-01-01	527564.900000	748233.580	842993.490
193	2017-02-01	469140.060000	497707.410	844676.980
194	2017-03-01	608144.980000	734984.240	1351787.920
195	2017-04-01	681303.110000	632843.410	1406022.430
196	2017-05-01	386088.180000	442311.090	889926.860
197	2017-06-01	519389.660000	582374.340	902659.000
198	2017-07-01	433026.080000	769214.060	994338.300
199	2017-08-01	592103.310000	607368.720	846696.350

200	2017-09-01	316435.220000	401678.980	925397.870
201	2017-10-01	574073.610000	657241.760	740198.570
202	2017-11-01	434963.360000	504247.310	1135350.100
203	2017-12-01	679441.470000	803216.880	1305047.550
204	2018-01-01	571858.060000	535784.700	1153831.890
205	2018-02-01	509366.100000	629467.910	921684.120
206	2018-03-01	563899.060000	529507.150	889227.100
207	2018-04-01	510830.000000	617512.280	696962.770
208	2018-05-01	580825.110000	580329.350	1592580.340
209	2018-06-01	463620.610000	558257.590	777679.910
210	2018-07-01	543333.770000	784601.770	793683.410
211	2018-08-01	663834.040000	541557.340	1281364.780
212	2018-09-01	454442.870000	536897.090	896266.430
213	2018-10-01	726044.350000	663440.340	992682.040
214	2018-11-01	534084.490000	509382.860	899588.340
215	2018-12-01	535639.970000	631652.560	792198.660
216	2019-01-01	702923.370000	684504.840	1195728.200
217	2019-02-01	469437.280000	521555.380	1208476.160
218	2019-03-01	678145.380000	704276.530	1550052.840
219	2019-04-01	867761.340000	656495.950	1215209.050
220	2019-05-01	596798.220000	568128.410	1011872.870
221	2019-06-01	712731.800000	629788.430	1029100.900
222	2019-07-01	531577.640000	516273.850	1056569.110
223	2019-08-01	896841.310000	713859.980	1167556.890
224	2019-09-01	518043.680000	491194.710	717490.020
225	2019-10-01	712672.380000	691407.730	1160570.850
226	2019-11-01	791039.020000	658706.420	1122990.690
227	2019-12-01	731012.240000	797580.750	1098972.390
228	2020-01-01	629414.900000	573220.630	999181.200
229	2020-02-01	741509.070000	611314.130	1237018.370
230	2020-03-01	735491.290000	809640.180	1367335.140
231	2020-04-01	329134.180000	453153.090	834366.520
232	2020-05-01	415805.820000	170292.500	691066.440
233	2020-06-01	683322.720000	415672.190	1066148.050
234	2020-07-01	451717.750000	347541.590	779930.570
235	2020-08-01	547899.550000	414416.340	791258.190
236	2020-09-01	591799.830000	563742.500	1021360.830
237	2020-10-01	861840.540000	733692.010	891340.690
238	2020-11-01	696600.140000	570262.170	1152909.510
239	2020-12-01	780347.280000	777450.550	1050560.830
240	2021-01-01	626537.480000	642652.370	1185644.960
241	2021-02-01	715261.340000	726508.780	1076824.670
242	2021-03-01	772750.010000	850334.640	1345110.180
243	2021-04-01	634408.890000	1227173.530	1250171.580
244	2021-05-01	669085.780000	649783.140	1279017.240
245	2021-06-01	833544.640000	623695.200	1440106.470
246	2021-07-01	833732.310000	689717.290	1110409.530
247	2021-08-01	664522.740000	685977.450	1175960.640
248	2021-09-01	844430.030000	646233.310	896539.240

249	2021-10-01	522701.110000	589405.560	1542473.570
250	2021-11-01	692267.310000	623392.630	1029780.800
251	2021-12-01	785313.560000	640981.150	1294042.310
252	2022-01-01	848902.970000	947226.270	1438571.280
253	2022-02-01	723925.740000	673653.800	1217495.310
254	2022-03-01	772406.100000	592318.210	1246975.120
255	2022-04-01	906104.050000	493958.320	1284453.730
256	2022-05-01	762776.120000	663348.940	1306786.770
257	2022-06-01	641348.460000	556311.960	1032070.530
258	2022-07-01	987872.050000	611115.010	1116548.970
259	2022-08-01	657942.720000	378627.690	1299910.960
260	2022-09-01	869579.640000	551589.720	1077517.900
261	2022-10-01	718920.200000	529832.420	980527.080
262	2022-11-01	754135.860000	595870.420	1136760.550
263	2022-12-01	841323.980000	602458.300	1082107.640
264	2023-01-01	909391.130000	578792.140	1442099.080
265	2023-02-01	725101.200000	685183.060	1267967.390
266	2023-03-01	803262.670000	633849.050	1317519.910
267	2023-04-01	922032.390000	572201.360	1417182.730
268	2023-05-01	947633.290000	668478.730	1428099.620
269	2023-06-01	831466.440000	639685.800	1278824.180
270	2023-07-01	830098.860000	562498.090	1113086.840
271	2023-08-01	801621.750000	669240.210	1075379.760
272	2023-09-01	762592.300000	586145.300	1242719.420
273	2023-10-01	928439.180000	652149.450	834686.020
274	2023-11-01	839290.020000	682060.840	1348739.160
275	2023-12-01	763754.270000	571924.920	1509634.280
276	2024-01-01	914133.320000	712333.330	1409097.150
277	2024-02-01	740662.250000	650360.110	1236861.750
278	2024-03-01	838270.930000	620077.740	1477038.000
279	2024-04-01	886132.770000	687017.960	1294706.120
280	2024-05-01	939656.180000	696970.300	1470870.090

	Gas licuado de petróleo
0	1.940657e+05
1	1.707034e+05
2	1.618374e+05
3	1.630486e+05
4	1.715189e+05
5	1.900044e+05
6	2.060228e+05
7	1.005615e+05
8	1.868390e+05
9	1.638642e+05
10	1.947225e+05
11	1.943290e+05
12	1.684226e+05
13	2.001414e+05
14	1.934226e+05

```
15          2.182575e+05
16          1.592994e+05
17          2.108746e+05
18          1.997085e+05
19          1.922137e+05
20          1.906867e+05
21          2.108712e+05
22          2.275959e+05
23          1.922720e+05
24          2.375323e+05
25          1.698190e+05
26          2.210309e+05
27          1.921701e+05
28          1.850167e+05
29          2.318050e+05
30          2.165049e+05
31          2.099524e+05
32          2.314992e+05
33          2.163028e+05
34          2.325270e+05
35          2.144899e+05
36          2.512715e+05
37          1.977303e+05
38          2.533661e+05
39          2.066719e+05
40          2.023682e+05
41          2.318503e+05
42          2.204493e+05
43          2.158811e+05
44          2.520583e+05
45          2.087155e+05
46          1.956455e+05
47          2.786885e+05
48          2.264525e+05
49          1.792128e+05
50          2.711630e+05
51          1.876000e+05
52          2.436253e+05
53          2.560014e+05
54          3.510896e+05
55          1.150986e+05
56          4.619572e+05
57          4.567143e+05
58          1.278198e+05
59          4.564411e+05
60          4.619158e+05
61          1.393182e+05
62          3.852874e+05
63          3.177561e+05
```

64	1.345260e+05
65	3.475493e+05
66	1.400744e+05
67	4.340132e+05
68	4.576832e+05
69	1.309387e+05
70	4.920634e+05
71	1.129128e+05
72	3.083485e+05
73	2.950828e+05
74	3.993872e+05
75	3.917908e+05
76	1.321066e+05
77	5.532520e+05
78	1.531678e+05
79	3.096471e+05
80	1.834849e+05
81	4.861949e+05
82	1.486274e+05
83	3.710984e+05
84	5.226705e+05
85	1.269412e+05
86	3.212948e+05
87	2.965320e+05
88	5.065898e+05
89	1.356305e+05
90	3.651176e+05
91	1.249092e+05
92	2.903307e+05
93	5.158331e+05
94	3.047807e+05
95	2.658542e+05
96	4.408072e+05
97	3.088982e+05
98	1.163574e+05
99	3.827966e+05
100	1.080326e+05
101	4.399545e+05
102	4.518928e+05
103	1.511795e+05
104	2.655124e+05
105	2.942996e+05
106	5.434985e+05
107	1.212566e+05
108	4.370423e+05
109	4.108464e+05
110	1.484954e+05
111	4.100011e+05
112	2.305846e+05

113	3.446831e+05
114	4.619780e+05
115	2.361643e+05
116	1.443818e+05
117	3.284726e+05
118	5.406719e+05
119	2.865017e+05
120	5.115181e+05
121	3.138772e+05
122	2.971903e+05
123	4.263802e+05
124	2.383741e+05
125	3.451963e+05
126	1.489885e+05
127	5.656279e+05
128	3.823104e+05
129	3.190910e+05
130	3.843684e+05
131	1.410525e+05
132	4.817815e+05
133	1.625613e+05
134	5.214937e+05
135	1.319584e+05
136	5.105068e+05
137	5.276723e+05
138	1.654242e+05
139	3.878921e+05
140	5.308978e+05
141	1.622084e+05
142	5.693459e+05
143	1.529268e+05
144	5.753185e+05
145	1.427728e+05
146	5.690132e+05
147	5.086495e+05
148	1.814616e+05
149	5.368472e+05
150	3.529090e+05
151	3.593460e+05
152	3.949859e+05
153	4.158693e+05
154	4.256692e+05
155	4.524141e+05
156	3.718887e+05
157	3.368452e+05
158	3.587558e+05
159	4.667670e+05
160	5.547843e+05
161	3.591356e+05

162	4.175703e+05
163	4.160683e+05
164	4.089989e+05
165	3.148689e+05
166	4.873437e+05
167	3.801860e+05
168	5.233523e+05
169	4.212682e+05
170	4.971686e+05
171	5.196038e+05
172	4.158903e+05
173	5.263585e+05
174	4.119714e+05
175	5.615638e+05
176	4.802689e+05
177	4.232394e+05
178	6.373165e+05
179	5.006843e+05
180	7.489706e+05
181	2.873377e+05
182	5.044511e+05
183	5.874827e+05
184	4.169125e+05
185	7.033192e+05
186	4.558231e+05
187	5.883105e+05
188	3.065434e+05
189	7.173886e+05
190	5.059760e+05
191	5.848072e+05
192	5.673458e+05
193	2.795272e+05
194	7.800486e+05
195	5.305781e+05
196	6.047026e+05
197	4.872442e+05
198	5.104281e+05
199	5.385821e+05
200	3.502946e+05
201	5.847096e+05
202	9.359865e+05
203	5.954214e+05
204	3.963630e+05
205	5.910000e+05
206	6.442454e+05
207	1.885908e+05
208	6.362341e+05
209	6.188113e+05
210	5.843861e+05
211	4.086586e+05

212	4.498468e+05
213	6.002305e+05
214	5.986262e+05
215	6.148870e+05
216	4.140861e+05
217	4.605188e+05
218	3.844256e+05
219	6.706339e+05
220	6.234662e+05
221	6.127323e+05
222	6.674868e+05
223	5.864306e+05
224	8.507782e+05
225	2.252070e+05
226	6.108681e+05
227	6.000039e+05
228	7.156003e+05
229	4.513317e+05
230	6.432789e+05
231	5.103499e+05
232	6.163814e+05
233	4.676648e+05
234	6.753375e+05
235	6.343201e+05
236	3.155271e+05
237	9.608405e+05
238	3.286031e+05
239	6.841601e+05
240	4.756195e+05
241	6.177832e+05
242	5.498153e+05
243	6.644439e+05
244	6.785011e+05
245	4.666585e+05
246	6.652631e+05
247	7.460025e+05
248	6.898295e+05
249	6.963469e+05
250	5.241865e+05
251	6.794151e+05
252	6.571857e+05
253	6.732183e+05
254	7.135534e+05
255	6.939644e+05
256	6.672722e+05
257	2.657738e+05
258	7.176527e+05
259	7.352537e+05
260	4.213032e+05

```
261       7.178492e+05
262       6.912374e+05
263       6.714446e+05
264       4.977807e+05
265       6.529847e+05
266       7.119783e+05
267       6.476663e+05
268       7.133490e+05
269       2.858178e+05
270       1.077123e+06
271       2.749736e+05
272       5.917119e+05
273       6.735812e+05
274       5.787974e+05
275       6.921820e+05
276       7.015708e+05
277       9.165417e+05
278       6.751575e+05
279       4.739407e+05
280       6.848645e+05
```

Dataset: Precios-Promedio-Nacionales-Diarios-2024-3.xlsx

```
csvPath = 'PreciosPromedioNacionales/PreciosPromedioNacionales.csv'

dataPrecios = pd.read_csv(csvPath)
print(f" Archivo cargado: {csvPath}")
```

```
 Archivo cargado:
PreciosPromedioNacionales/PreciosPromedioNacionales.csv
```

```
dataPrecios = dataPrecios[['FECHA', 'Regular GTQ/GALON', 'Superior
GTQ/GALON', 'Diesel GTQ/GALON', 'Glp Cilindro 25Lbs. GTQ/LB']]
dataPrecios['FECHA'] = pd.to_datetime(dataPrecios['FECHA'])
dataPrecios.head()
```

```
    FECHA     Regular GTQ/GALON  Superior GTQ/GALON  Diesel GTQ/GALON
\
0 2021-01-01              21.11               21.91              17.61

1 2021-01-02              21.11               21.91              17.61

2 2021-01-03              21.11               21.91              17.61

3 2021-01-04              21.11               21.91              17.61

4 2021-01-05              21.11               21.91              17.61
```

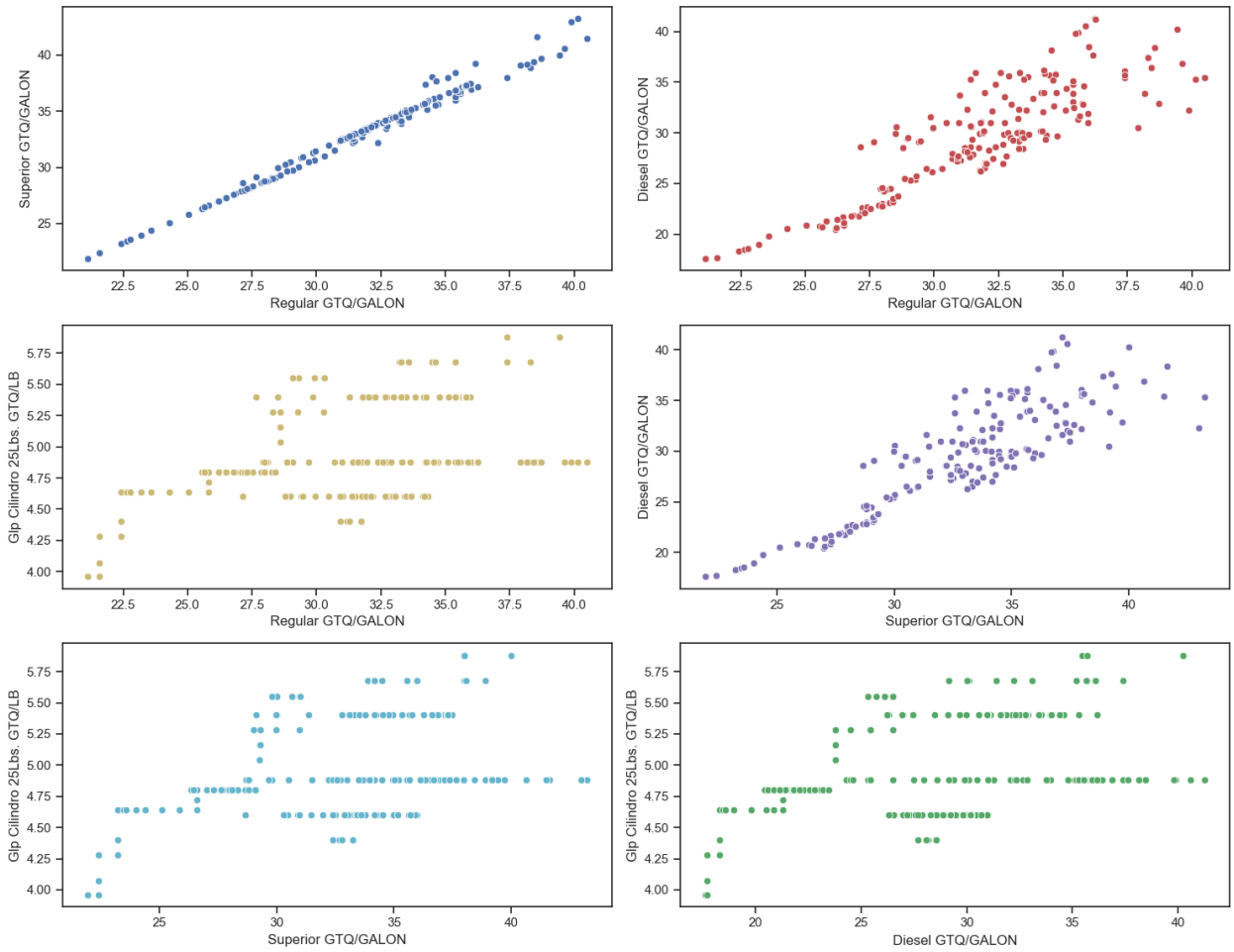
```
   Glp Cilindro 25Lbs. GTQ/LB
0                        3.96
1                        3.96
2                        3.96
3                        3.96
4                        3.96
```

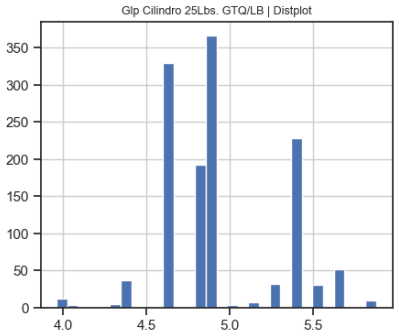
En este análisis se puede destacar que la mayoría de categorías de combustibles cuentan con distribuciones cercanas a una distribucion normal, con la excepción de *Gas Cilindro 25Lbs*. Los valores atípicos se aprecian más en esta última que en las otras categorías. *Superior*, *Regular* y *Diesel* tienen rangos intercuartiles similares y más amplios que en comparación de *Gas Cilindro 25Lbs*.

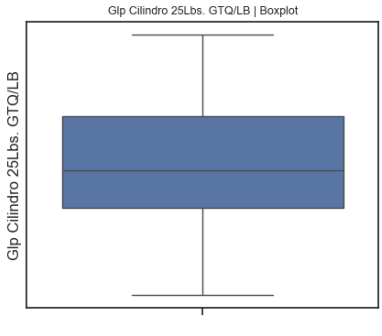
```
csvPath = 'PreciosPromedioNacionales/dataPrecios.csv'
dataPrecios.to_csv(csvPath, index=False)
AV = AutoViz_Class()
AV.AutoViz(csvPath)
```

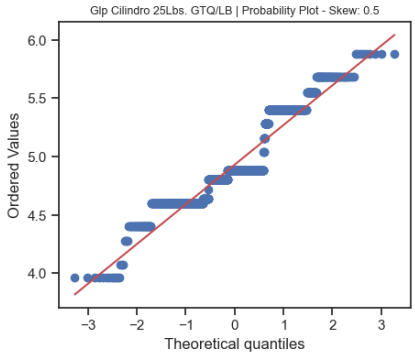
```
Shape of your Data Set loaded: (1305, 5)
############################################################################
#################
######################## C L A S S I F Y I N G  V A R I A B L E S
####################
############################################################################
#################
Classifying variables in data set...
    Number of Numeric Columns =  4
    Number of Integer-Categorical Columns =  0
    Number of String-Categorical Columns =  0
    Number of Factor-Categorical Columns =  0
    Number of String-Boolean Columns =  0
    Number of Numeric-Boolean Columns =  0
    Number of Discrete String Columns =  0
    Number of NLP String Columns =  0
    Number of Date Time Columns =  0
    Number of ID Columns =  1
    Number of Columns to Delete =  0
    5 Predictors classified...
        1 variable(s) removed since they were ID or low-information
variables
        List of variables removed: ['FECHA']
To fix these data quality issues in the dataset, import FixDQ from
autoviz...
    All variables classified into correct types.

<pandas.io.formats.style.Styler at 0x1545e536590>

Number of All Scatter Plots = 10
```

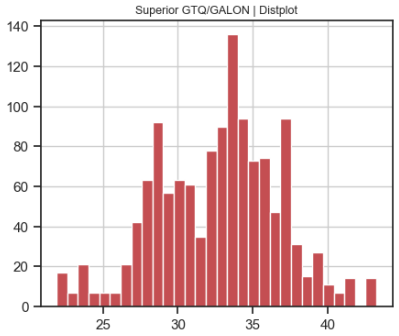
Pair-wise Scatter Plot of all Continuous Variables

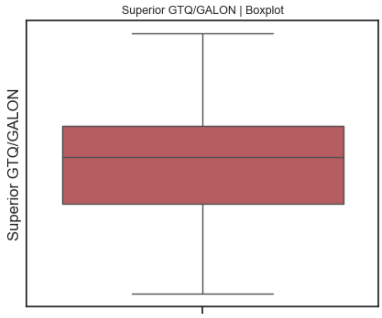


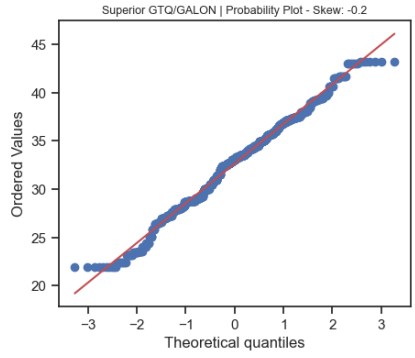

Glp Cilindro 25Lbs. GTQ/LB | Distplot

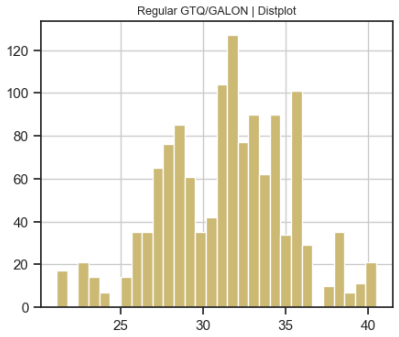

Glp Cilindro 25Lbs. GTQ/LB | Boxplot
Glp Cilindro 25Lbs. GTQ/LB

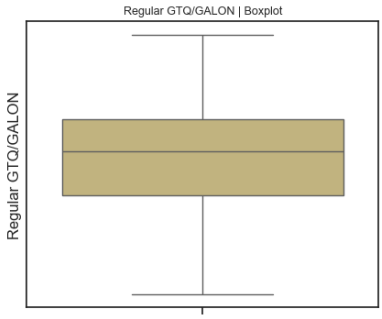

Glp Cilindro 25Lbs. GTQ/LB | Probability Plot - Skew: 0.5
Ordered Values
Theoretical quantiles

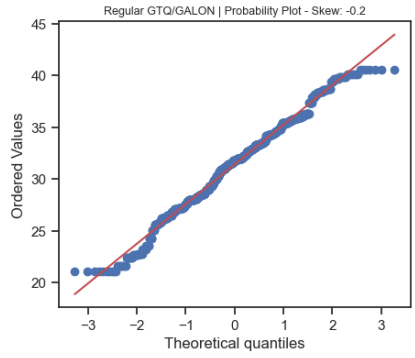

Superior GTQ/GALON | Distplot

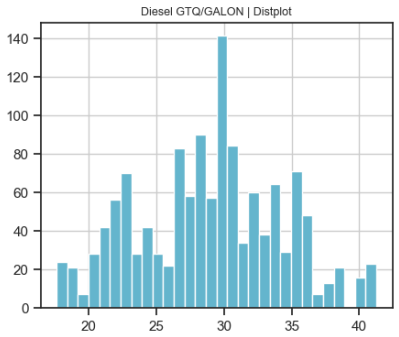

Superior GTQ/GALON | Boxplot
Superior GTQ/GALON

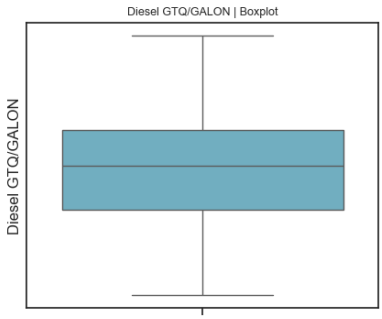

Superior GTQ/GALON | Probability Plot - Skew: -0.2
Ordered Values
Theoretical quantiles

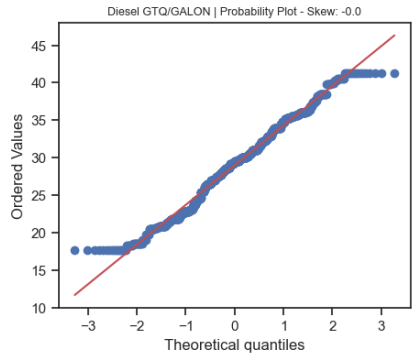

Regular GTQ/GALON | Distplot

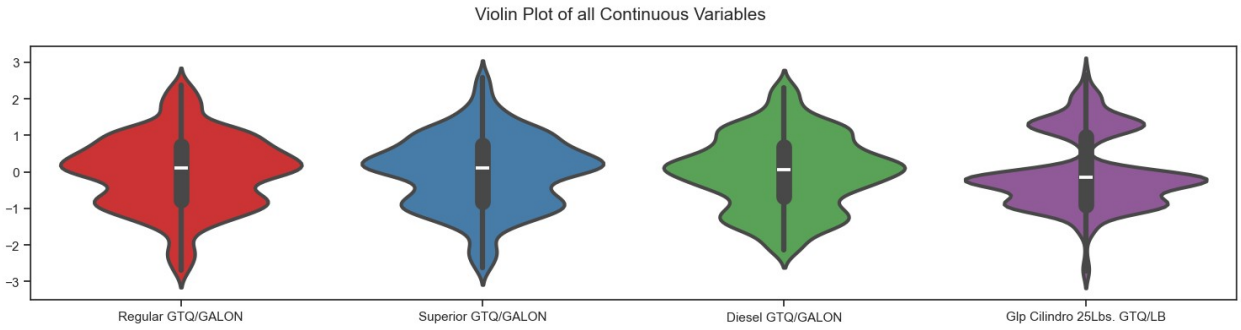

Regular GTQ/GALON | Boxplot
Regular GTQ/GALON

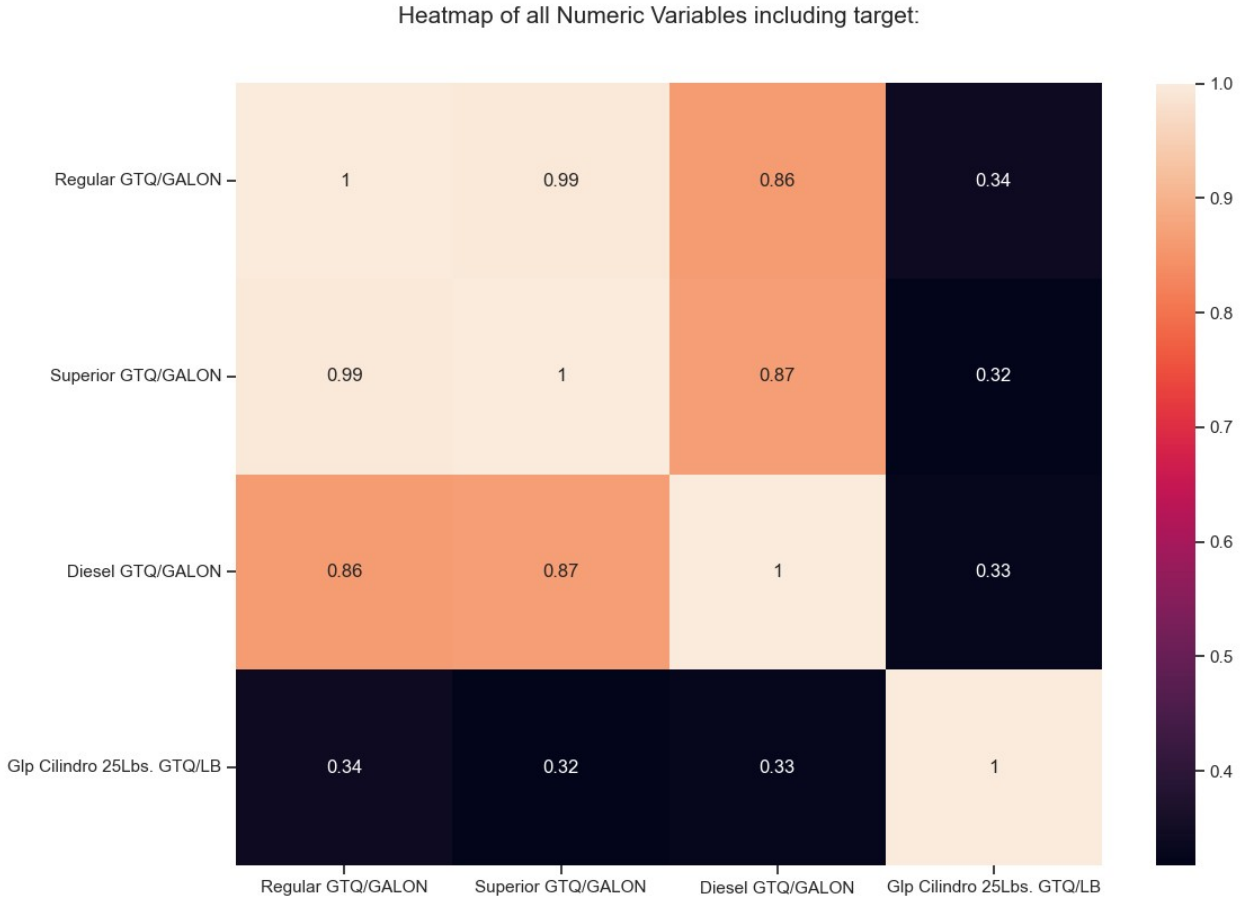

Regular GTQ/GALON | Probability Plot - Skew: -0.2
Ordered Values
Theoretical quantiles


Diesel GTQ/GALON | Distplot


Diesel GTQ/GALON | Boxplot
Diesel GTQ/GALON


Diesel GTQ/GALON | Probability Plot - Skew: -0.0
Ordered Values
Theoretical quantiles






```
All Plots done
Time to run AutoViz = 1 seconds

 ###################### AUTO VISUALIZATION Completed
########################

        FECHA     Regular GTQ/GALON  Superior GTQ/GALON  Diesel
GTQ/GALON  \
0     2021-01-01        21.11               21.91               17.61

1     2021-01-02        21.11               21.91               17.61
```

```
2     2021-01-03        21.11                21.91                17.61

3     2021-01-04        21.11                21.91                17.61

4     2021-01-05        21.11                21.91                17.61

...          ...                  ...                  ...
...
1300  2024-07-24        31.29                32.79                28.09

1301  2024-07-25        31.29                32.79                28.09

1302  2024-07-26        31.29                32.79                28.09

1303  2024-07-27        31.29                32.79                28.09

1304  2024-07-28        31.29                32.79                28.09


      Glp Cilindro 25Lbs. GTQ/LB
0                  3.96
1                  3.96
2                  3.96
3                  3.96
4                  3.96
...                         ...
1300               4.40
1301               4.40
1302               4.40
1303               4.40
1304               4.40

[1305 rows x 5 columns]
```

Datasets: Seleccion de series de tiempo.

Las tres series seleccionadas fueron:

- Dataset consumo = 'Diesel'
- Dataset importación = 'Gasolina regular'
- Dataset precios = 'Superior GTQ/GALON'

```
serieConsumoDiesel = dataConsumo[['Fecha', 'Diesel']]
serieImportacionRegular = dataImportacion[['Fecha', 'Gasolina
regular']]
seriePreciosSuper = dataPrecios[['FECHA', 'Superior GTQ/GALON']]

serieConsumoDiesel.set_index('Fecha', inplace=True)
serieConsumoDiesel.sort_index(inplace=True)
```

```
serieImportacionRegular.set_index('Fecha', inplace=True)
serieImportacionRegular.sort_index(inplace=True)
seriePreciosSuper.set_index('FECHA', inplace=True)
seriePreciosSuper.sort_index(inplace=True)
```

Datos iniciales sobre las series.

Como se puede observar, se cuenta con tres series de tiempo. Se describen las fechas de inicio y finalización de cada serie, teniendo un abanico de datos que van desde el año 2000 hasta mayo de 2024. Las siglas de frecuencia representan lo siguiente:

- MS significa "Month Start", lo que indica que los datos para ese dataset son mensuales y se registran al inicio de cada mes.
- D significa "Daily", lo que indica que los datos se registran diariamente.

```
def mostrarInfoSeriesTiempo(consumoDiesel, importacionRegular,
preciosSuper):
    print(" Información de las series de tiempo:")

    print("\n Consumo Diesel:")
    print(f"   Inicio: {consumoDiesel.index.min()}")
    print(f"   Fin: {consumoDiesel.index.max()}")
    frecuenciaDiesel = consumoDiesel.index.freq or
pd.infer_freq(consumoDiesel.index)
    print(f"   Frecuencia: {frecuenciaDiesel}")

    print("\n Importación Gasolina Regular:")
    print(f"   Inicio: {importacionRegular.index.min()}")
    print(f"   Fin: {importacionRegular.index.max()}")
    frecuenciaRegular = importacionRegular.index.freq or
pd.infer_freq(importacionRegular.index)
    print(f"   Frecuencia: {frecuenciaRegular}")

    print("\n Precios Super:")
    print(f"   Inicio: {preciosSuper.index.min()}")
    print(f"   Fin: {preciosSuper.index.max()}")
    frecuenciaSuper = preciosSuper.index.freq or
pd.infer_freq(preciosSuper.index)
    print(f"   Frecuencia: {frecuenciaSuper}")

# Supongamos que ya tienes las series de tiempo `serieConsumoDiesel`,
`serieImportacionRegular` y `seriePreciosSuper`
mostrarInfoSeriesTiempo(serieConsumoDiesel, serieImportacionRegular,
seriePreciosSuper)
```

```
 Información de las series de tiempo:

 Consumo Diesel:
   Inicio: 2000-01-01 00:00:00
   Fin: 2024-05-01 00:00:00
```

```
   Frecuencia: MS

 Importación Gasolina Regular:
   Inicio: 2001-01-01 00:00:00
   Fin: 2024-05-01 00:00:00
   Frecuencia: MS

 Precios Super:
   Inicio: 2021-01-01 00:00:00
   Fin: 2024-07-28 00:00:00
   Frecuencia: D
```

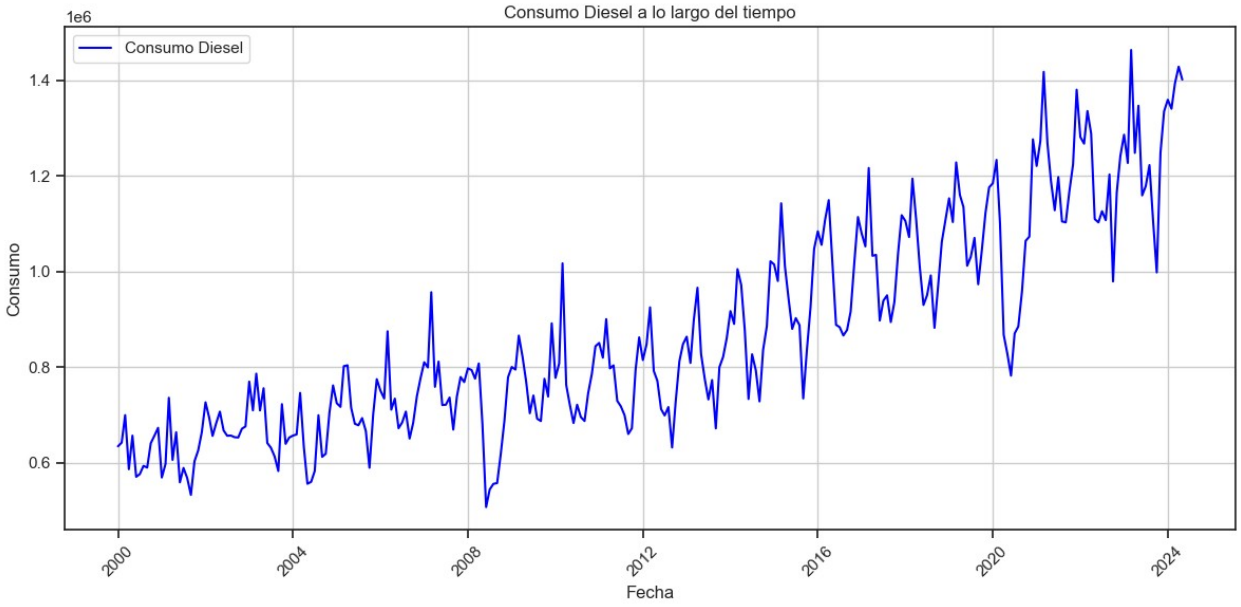
Visualizacion de las series.

Para esta parte, se deciden mostrar las series de tiempo para ver un comportamiento inicial. Pero adicionalmente, se genera graficos que permiten observarlas en distintos componentes.

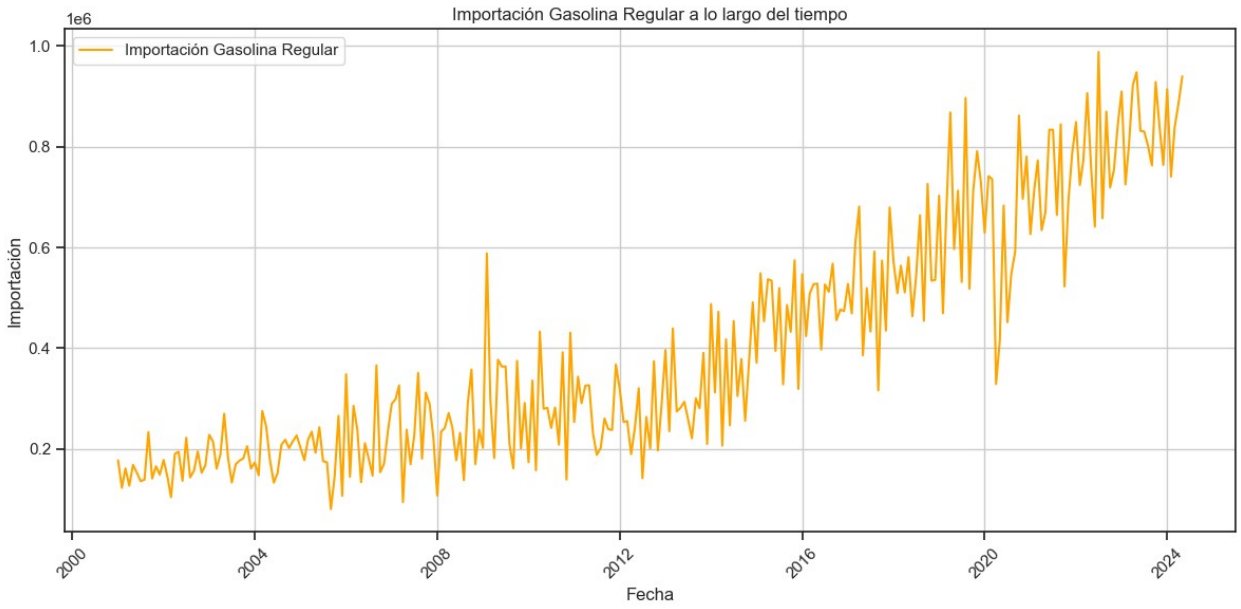
```
def plotTimeSeries(consumoDiesel, importacionRegular, preciosSuper):
    print(" Iniciando la visualización de series de tiempo...")

    # Grafica Consumo Diesel
    plt.figure(figsize=(12, 6))
    print(" Graficando Consumo Diesel...")
    plt.plot(consumoDiesel.index, consumoDiesel['Diesel'],
label='Consumo Diesel', color='blue')
    plt.title('Consumo Diesel a lo largo del tiempo')
    plt.xlabel('Fecha')
    plt.ylabel('Consumo')
    plt.legend()
    plt.grid()
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
    print(" Gráfica de Consumo Diesel completada.")

    # Grafica Importación Gasolina Regular
    plt.figure(figsize=(12, 6))
    print(" Graficando Importación Gasolina Regular...")
    plt.plot(importacionRegular.index, importacionRegular['Gasolina
regular'], label='Importación Gasolina Regular', color='orange')
    plt.title('Importación Gasolina Regular a lo largo del tiempo')
    plt.xlabel('Fecha')
    plt.ylabel('Importación')
    plt.legend()
    plt.grid()
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
    print(" Gráfica de Importación Gasolina Regular completada.")
```

```
    # Grafica Precios Super
    plt.figure(figsize=(12, 6))
    print("📈 Graficando Precios Super...")
    plt.plot(preciosSuper.index, preciosSuper['Superior GTQ/GALON'],
label='Precios Super', color='green')
    plt.title('Precios Super a lo largo del tiempo')
    plt.xlabel('Fecha')
    plt.ylabel('Precio (GTQ/GALON)')
    plt.legend()
    plt.grid()
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
    print("✅ Gráfica de Precios Super completada.")

plotTimeSeries(serieConsumoDiesel, serieImportacionRegular,
seriePreciosSuper)
```

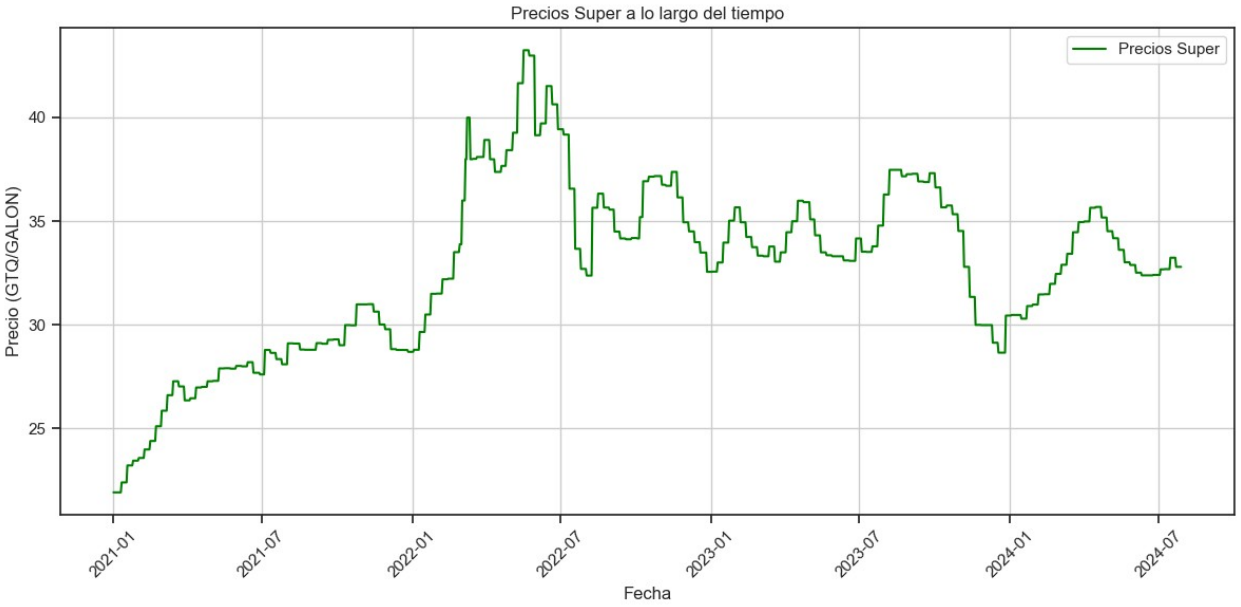
```
🚀 Iniciando la visualización de series de tiempo...
📈 Graficando Consumo Diesel...
```




```
✅ Gráfica de Consumo Diesel completada.
📈 Graficando Importación Gasolina Regular...
```




```
 Gráfica de Importación Gasolina Regular completada.
 Graficando Precios Super...
```

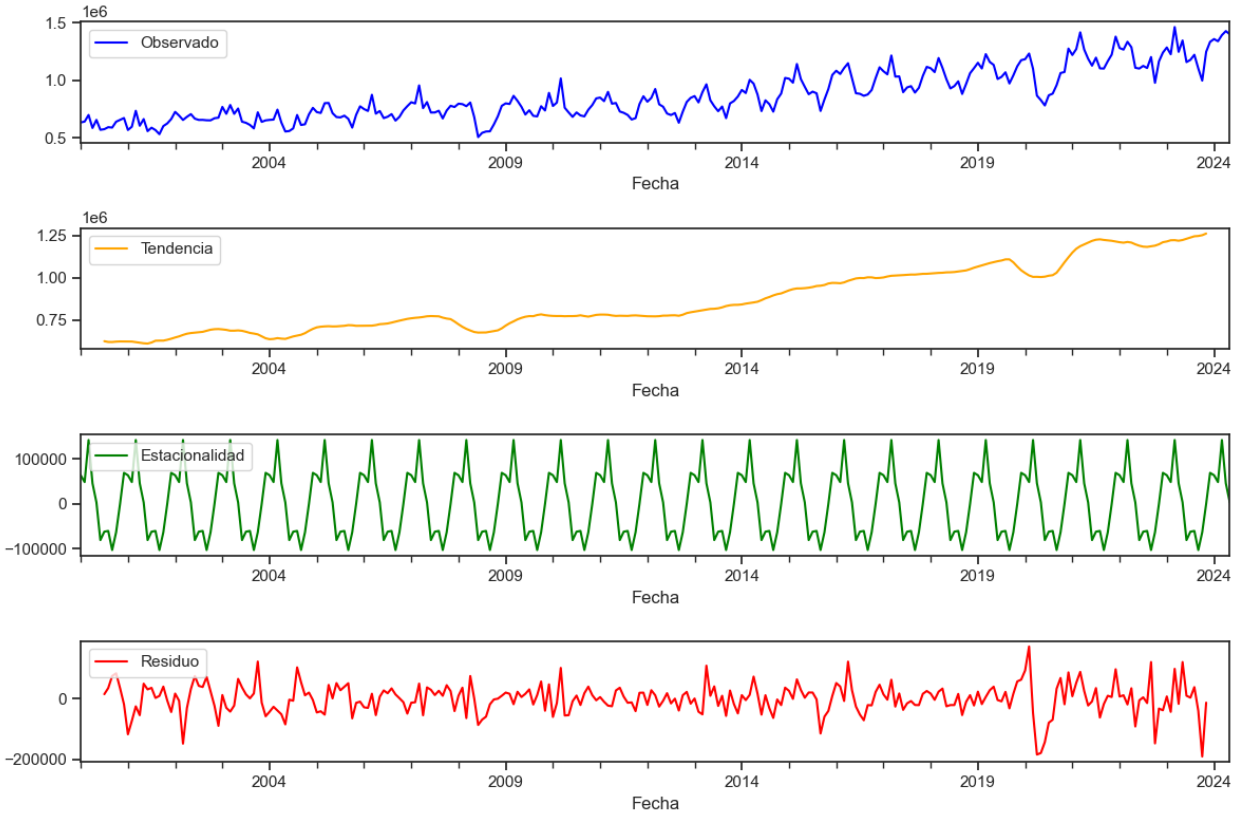



```
 Gráfica de Precios Super completada.
```

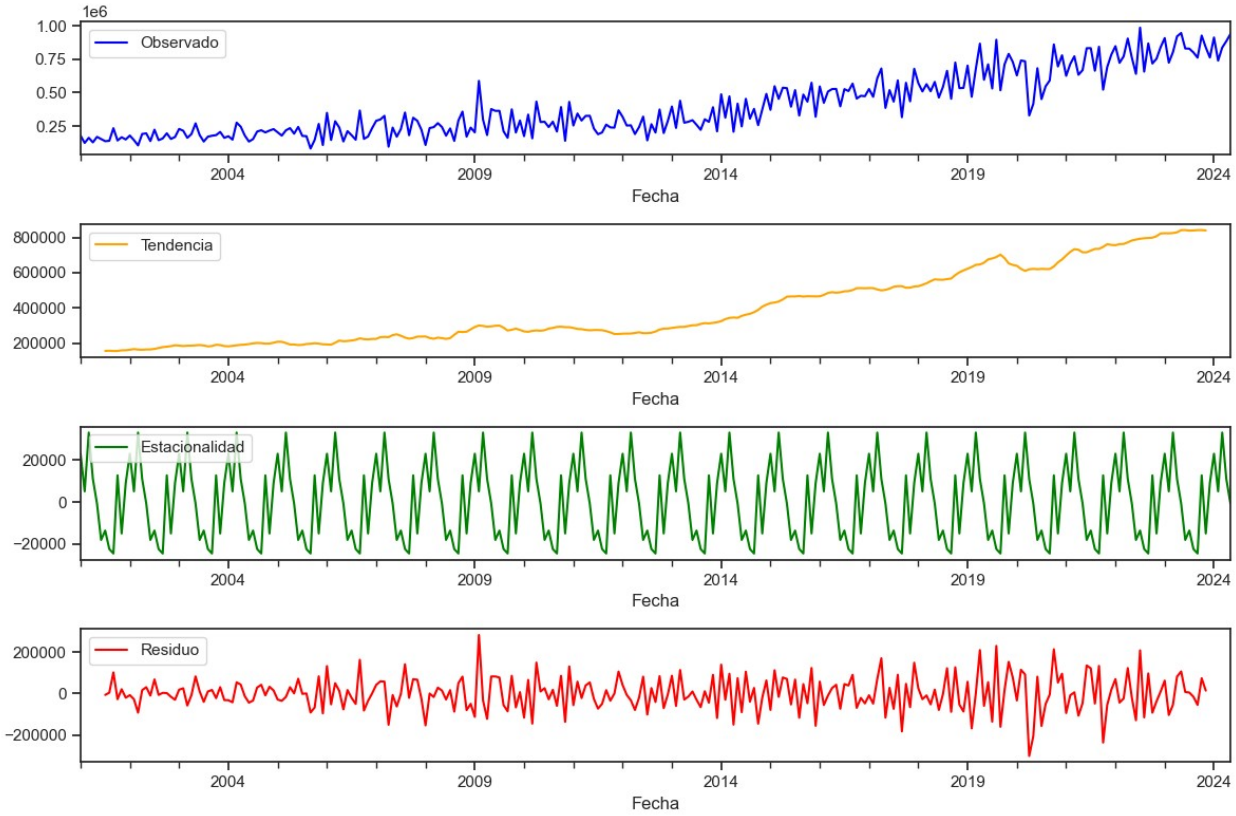
```python
def descomponerSerieTiempo(serieTiempo, descripcion):
    print(f" {descripcion}: Iniciando la descomposición de la serie de tiempo... ")

    # Descomposición de la serie
    descomposicion = sm.tsa.seasonal_decompose(serieTiempo,
```

```
model='additive')

    # Graficar los componentes
    plt.figure(figsize=(12, 8))
    descomposicion.observed.plot(ax=plt.subplot(411),
label='Observado', color='blue')
    plt.legend(loc='upper left')

    descomposicion.trend.plot(ax=plt.subplot(412), label='Tendencia',
color='orange')
    plt.legend(loc='upper left')

    descomposicion.seasonal.plot(ax=plt.subplot(413),
label='Estacionalidad', color='green')
    plt.legend(loc='upper left')

    descomposicion.resid.plot(ax=plt.subplot(414), label='Residuo',
color='red')
    plt.legend(loc='upper left')

    plt.tight_layout()
    plt.show()

    print("□ Descomposición completada.")

descomponerSerieTiempo(serieConsumoDiesel['Diesel'], 'Diesel')
descomponerSerieTiempo(serieImportacionRegular['Gasolina regular'],
'Regular')
descomponerSerieTiempo(seriePreciosSuper['Superior GTQ/GALON'],
'Superior')
```

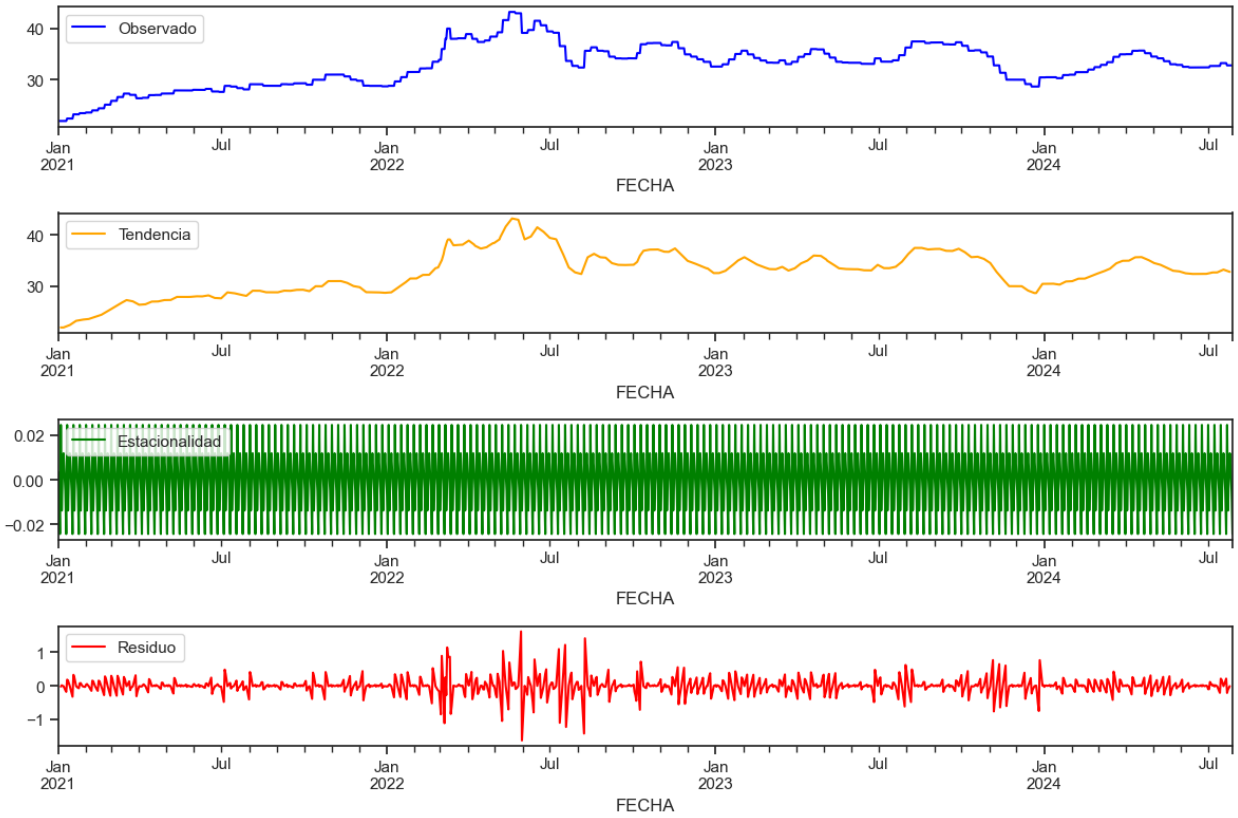
```
□ Diesel: Iniciando la descomposición de la serie de tiempo...
```




```
✅ Descomposición completada.
📊 Regular: Iniciando la descomposición de la serie de tiempo...
```




```
 Descomposición completada.
 Superior: Iniciando la descomposición de la serie de tiempo...
```




```
✅ Descomposición completada.
```

- Consumo de diesel: Se puede observar que lo largo del tiempo, cuenta con una tendencia general ascendente con alguna fluctuaciones durante ciertos años como el 2008, 2012, 2016 y 2020. Seguramente este último debido a la pandemia de COVID-19. Además, se puede observar que cuenta con algunos patrones de estacionalidad con variaciones relativamente regulares en ciertos periodos.

- Importacion de gasolina regular: Al igual que la tendencia del diesel, cuenta con una tendencia creciente en el tiempo y con la característica que es más pronunciada a partir de 2008. Respecto a la estacionalidad de la serie, se puede mencionar que existe mayor variabilidad y picos significativos en los mismos años mencionados del diesel. Estas variaciones se pueden atribuir a cambios en politicas de importacion, eventos fuera de lo ordinario como la pandemia, entre otros.

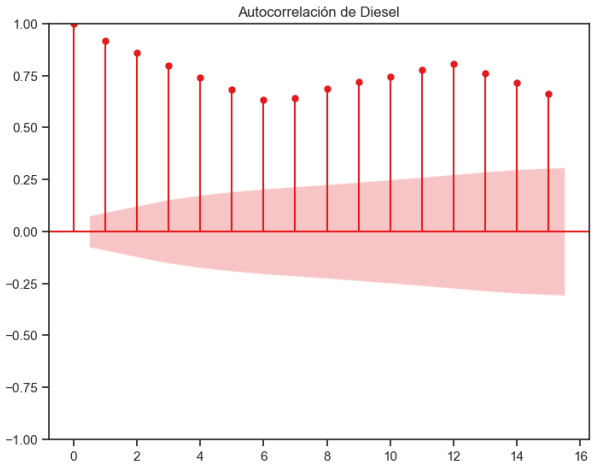
- Precios diarios promedio de super: A diferencia de las últimas dos series, los precios no muestran una tendencia muy clara a largo plazo. Aún asi, cabe desctacar que existen periodos de aumento y disminución significativos. Hablando de la estacinalidad de la serie, no se logra apreciar una estacionalidad clara pero sí es evidente que existen picos y valles notables. Entre 2021 y 2022 se observa que existio un aumento considerable de los precios seguidos de una disminución y varias flucturaciones. Estos cambios siempre se atribuyen a eventos como el COVID-19 o conflictos geopolíticos que afecta la producción y oferta del petróleo.

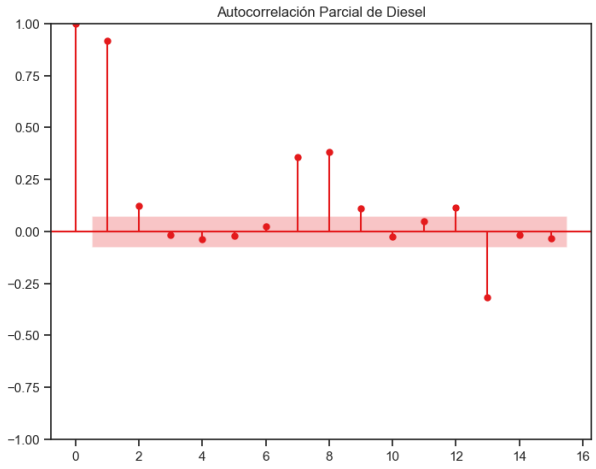
✍ Transformaciones

Se puede observar que todos los diagramas cuentan con un lento decaimiento y las autocorrelaciones son positivas. Esto nos indica que existe autocorrelación significativa y las series no son estacionarias. Además, analizando la autocorrelacion parcial, se puede observar que existe significancia entre el primer y segundo retardo principalmente.

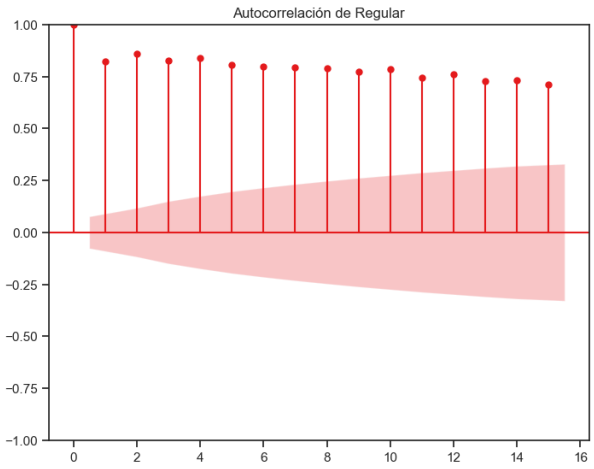
```
def graficarACF_PACF(serieTiempo, descripcion):
    print(f"□ Gráfico de Autocorrelación y Autocorrelación Parcial
para: {descripcion}")

    fig, axes = plt.subplots(1, 2, figsize=(16, 6))

    plot_acf(serieTiempo, lags=15, alpha=0.2, ax=axes[0])
    axes[0].set_title(f'Autocorrelación de {descripcion}')

    plot_pacf(serieTiempo, lags=15, alpha=0.2, ax=axes[1])
    axes[1].set_title(f'Autocorrelación Parcial de {descripcion}')

    plt.show()

# Ejemplos de uso:
graficarACF_PACF(serieConsumoDiesel['Diesel'], 'Diesel')
graficarACF_PACF(serieImportacionRegular['Gasolina regular'],
'Regular')
graficarACF_PACF(seriePreciosSuper['Superior GTQ/GALON'], 'Superior')
```

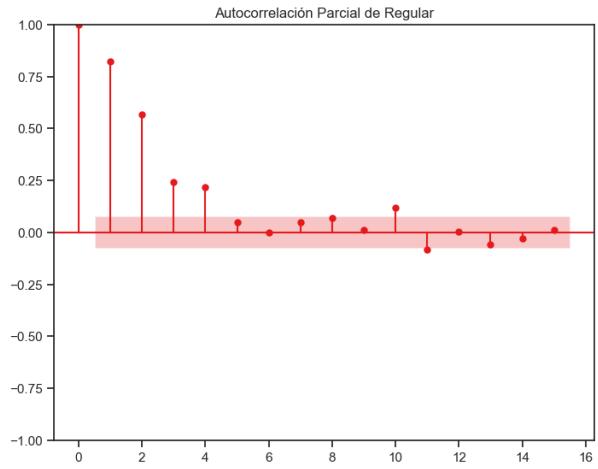
```
□ Gráfico de Autocorrelación y Autocorrelación Parcial para: Diesel
```



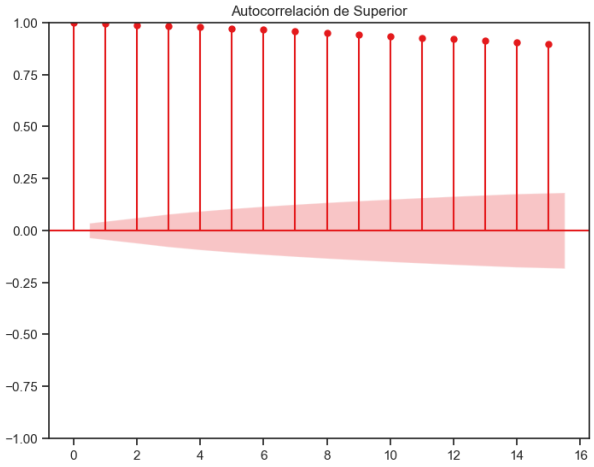


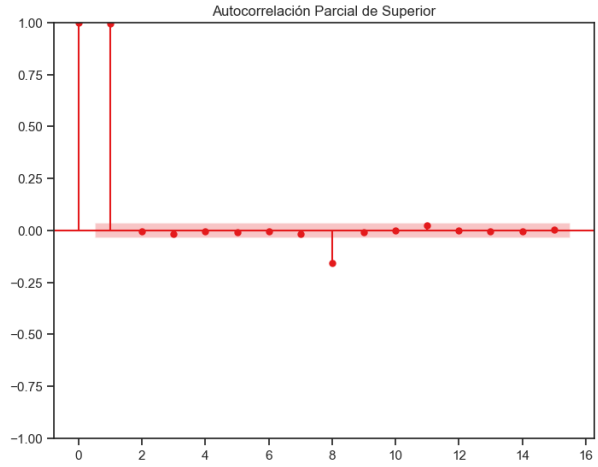

```
□ Gráfico de Autocorrelación y Autocorrelación Parcial para: Regular
```






```
 Gráfico de Autocorrelación y Autocorrelación Parcial para: Superior
```





De la prueba de Dickey-Fuller Aumentada se puede observar que tanto para diesel y gasolina regular, sus series no son estacionarios debido al p-valor determinado, por lo que se puede considerar realizar aplicar diferenciación antes de aplicar los modelos de series temporales. Por otra parte, respecto a la serie de gasolina superior, pareciera ser estacionaria por lo que se podría modelar directamente sin necesidad de diferenciación.

Las trasnformaciones que podrían ser requeridas serían aplicar diferenciaciones para las series *Diesel* y *Regular* o transformaciones logaritmicas para estabilizar varianza. Además, si se quisiera mejorar un poco la serie de *Super* se podría considerar aplicar alguna transformación para ver si mejora el rendimiento.

```
def pruebaDickeyFuller(serieTiempo, descripcion):
    print(f" Prueba de Dickey-Fuller Aumentada para: {descripcion}")
    resultado = adfuller(serieTiempo)
    print(f'Estadístico ADF: {resultado[0]}')
    print(f'p-valor: {resultado[1]}')
    print('Valores críticos:')
    for key, value in resultado[4].items():
```

```
        print(f'  {key}: {value}')
    print('----------------------------------------------------')
    return resultado[1]

print('----------------------------------------------------')
pruebaDickeyFuller(serieConsumoDiesel['Diesel'], 'Diesel')
pruebaDickeyFuller(serieImportacionRegular['Gasolina regular'],
'Regular')
pruebaDickeyFuller(seriePreciosSuper['Superior GTQ/GALON'],
'Superior')
```

```
----------------------------------------------------
 Prueba de Dickey-Fuller Aumentada para: Diesel
Estadístico ADF: 0.14523982117810716
p-valor: 0.9690175028779467
Valores críticos:
  1%: -3.453922368485787
  5%: -2.871918329081633
  10%: -2.5723001147959184
----------------------------------------------------
 Prueba de Dickey-Fuller Aumentada para: Regular
Estadístico ADF: 0.8288289928204905
p-valor: 0.992087554110633
Valores críticos:
  1%: -3.4547128138328875
  5%: -2.8722649771800155
  10%: -2.5724850011573914
----------------------------------------------------
 Prueba de Dickey-Fuller Aumentada para: Superior
Estadístico ADF: -2.9399220843939617
p-valor: 0.040898674582741115
Valores críticos:
  1%: -3.435401880796999
  5%: -2.863770985550096
  10%: -2.567957791647768
----------------------------------------------------

0.040898674582741115
```

En esta parte se realizan una serie de funciones que buscan automatizar el proceso de transformación de series para mejorar su rendimiento y evitar la no estacionalidad. Dentro de estas se mencionan la trasnformacion logaritmica y la diferenciación. Cabe destacar que de acuerdo con los resultados anteriores, si no se logra estabilizar tanto en media como en varianza, se decidieron los valores p=1, d=1 y q=1 para ser utilizados en los modelos requeridos.

```
def detectar_valores_invalidos(serieTemporal, descripcion):
    serieTemporal = pd.to_numeric(serieTemporal, errors='coerce')  #
Convertir a numérico y forzar NaN si hay errores
    n_nan = serieTemporal.isna().sum()
```

```
    n_no_positivos = (serieTemporal <= 0).sum()

    print(f"🔍 Detección de valores inválidos en la serie:
{descripcion}")
    print(f"  Valores NaN: {n_nan}")
    print(f"  Valores no positivos: {n_no_positivos}")

    serieTemporal_sin_nan = serieTemporal.dropna()
    print(f"  Serie temporal sin valores NaN tiene
{len(serieTemporal_sin_nan)} elementos.")

    return serieTemporal_sin_nan, n_nan, n_no_positivos

def varianza_estable(serieTemporal):
    n = len(serieTemporal)
    partes = np.array_split(serieTemporal, 3)
    varianzas = [np.var(parte) for parte in partes]
    return np.allclose(varianzas[0], varianzas[1], rtol=0.1) and
np.allclose(varianzas[1], varianzas[2], rtol=0.1)

def aplicar_transformacion_logaritmica(serieTemporal):
    if (serieTemporal > 0).all():
        return np.log(serieTemporal)
    else:
        print("⚠ La serie contiene valores no positivos, no se puede
aplicar la transformación logarítmica.")
        return serieTemporal

def analizar_y_transformar_serie(serieTemporal, descripcion):
    print(f"🔍 Analizando la serie: {descripcion}")

    serieTemporal, n_nan, n_no_positivos =
detectar_valores_invalidos(serieTemporal, descripcion)

    estacionaria = False
    intentos = 0
    max_intentos = 10

    serieTemporal = aplicar_transformacion_logaritmica(serieTemporal)

    while not estacionaria and intentos < max_intentos:
        intentos += 1
        p_valor = pruebaDickeyFuller(serieTemporal, descripcion)
        if p_valor < 0.05:
            if varianza_estable(serieTemporal):
                estacionaria = True
                print(f"✅ La serie {descripcion} es estacionaria en
media y varianza.")
                break
            else:
```

```
                print(f"□ La varianza no es estable en la serie
{descripcion}.")
        else:
            print(f"□ La serie {descripcion} no es estacionaria (p-
valor: {p_valor}).")

        serieTemporal = serieTemporal.diff().dropna()
        print(f"□ Aplicando diferenciación a {descripcion}. Intento
{intentos}.")

    if not estacionaria:
        print(f"⚠ No se pudo hacer la serie {descripcion} estacionaria
después de {max_intentos} intentos.")
        print("□ Utilizando un modelo ARIMA para tratar con la no
estacionariedad.")
        modelo_arima = ARIMA(serieTemporal, order=(1, 1, 1))
        modelo_arima_fit = modelo_arima.fit()
        print(modelo_arima_fit.summary())
    else:
        modelo_arima_fit = None

    return serieTemporal, modelo_arima_fit

# Asegúrate de que estas variables estén definidas correctamente
# Reemplaza con tus series temporales reales
serieTemporalD, modelo_arima_diesel =
analizar_y_transformar_serie(serieConsumoDiesel['Diesel'], 'Diesel')
#
print("-----------------------------------------------------------------
----------------------------------------")
serieTemporalR, modelo_arima_regular =
analizar_y_transformar_serie(serieImportacionRegular['Gasolina
regular'], 'Regular')
#
print("-----------------------------------------------------------------
----------------------------------------")
serieTemporalS, modelo_arima_super =
analizar_y_transformar_serie(seriePreciosSuper['Superior GTQ/GALON'],
'Superior')
```

```
□ Analizando la serie: Diesel
□ Detección de valores inválidos en la serie: Diesel
  Valores NaN: 0
  Valores no positivos: 0
  Serie temporal sin valores NaN tiene 293 elementos.
Prueba Dickey-Fuller para Diesel: p-valor = 0.901557753919378
□ La serie Diesel no es estacionaria (p-valor: 0.901557753919378).
□ Aplicando diferenciación a Diesel. Intento 1.
Prueba Dickey-Fuller para Diesel: p-valor = 7.851113111171977e-10
□ La varianza no es estable en la serie Diesel.
```

🔄 Aplicando diferenciación a Diesel. Intento 2.
Prueba Dickey-Fuller para Diesel: p-valor = 3.1021697475811376e-15
🔄 La varianza no es estable en la serie Diesel.
🔄 Aplicando diferenciación a Diesel. Intento 3.
Prueba Dickey-Fuller para Diesel: p-valor = 7.339914914662219e-25
🔄 La varianza no es estable en la serie Diesel.
🔄 Aplicando diferenciación a Diesel. Intento 4.
Prueba Dickey-Fuller para Diesel: p-valor = 3.85887345410522e-27
🔄 La varianza no es estable en la serie Diesel.
🔄 Aplicando diferenciación a Diesel. Intento 5.
Prueba Dickey-Fuller para Diesel: p-valor = 3.960328765820069e-24
🔄 La varianza no es estable en la serie Diesel.
🔄 Aplicando diferenciación a Diesel. Intento 6.
Prueba Dickey-Fuller para Diesel: p-valor = 1.693758290598338e-19
🔄 La varianza no es estable en la serie Diesel.
🔄 Aplicando diferenciación a Diesel. Intento 7.
Prueba Dickey-Fuller para Diesel: p-valor = 9.999475164371196e-19
🔄 La varianza no es estable en la serie Diesel.
🔄 Aplicando diferenciación a Diesel. Intento 8.
Prueba Dickey-Fuller para Diesel: p-valor = 6.431107622936416e-23
🔄 La varianza no es estable en la serie Diesel.
🔄 Aplicando diferenciación a Diesel. Intento 9.
Prueba Dickey-Fuller para Diesel: p-valor = 2.6791261349428788e-28
🔄 La varianza no es estable en la serie Diesel.
🔄 Aplicando diferenciación a Diesel. Intento 10.
⚠️ No se pudo hacer la serie Diesel estacionaria después de 10 intentos.
🔄 Utilizando un modelo ARIMA para tratar con la no estacionariedad.
                               SARIMAX Results

==
========
Dep. Variable: Diesel   No. Observations:
283
Model: ARIMA(1, 1, 1)   Log Likelihood   -
1078.599
Date: Sun, 04 Aug 2024   AIC
2163.197
Time: 14:50:24   BIC
2174.123
Sample: 11-01-2000   HQIC
2167.578
                         - 05-01-2024

Covariance Type: opg

==
========
                 coef   std err   z   P>|z| [0.025
0.975]

```
------------------------------------------------------------------------
--------
ar.L1         -0.8892      0.027    -32.441      0.000      -0.943
-0.836
ma.L1         -0.9995      1.502     -0.665      0.506      -3.944
1.945
sigma2       119.3735    179.118      0.666      0.505    -231.692
470.439
========================================================================
=============
Ljung-Box (L1) (Q):                 162.95   Jarque-Bera (JB):
0.09
Prob(Q):                              0.00   Prob(JB):
0.96
Heteroskedasticity (H):               0.54   Skew:
0.02
Prob(H) (two-sided):                  0.00   Kurtosis:
3.08
========================================================================
=============

Warnings:
[1] Covariance matrix calculated using the outer product of gradients
(complex-step).
 Analizando la serie: Regular
 Detección de valores inválidos en la serie: Regular
  Valores NaN: 0
  Valores no positivos: 0
  Serie temporal sin valores NaN tiene 281 elementos.
Prueba Dickey-Fuller para Regular: p-valor = 0.9547166884951503
 La serie Regular no es estacionaria (p-valor: 0.9547166884951503).
 Aplicando diferenciación a Regular. Intento 1.
Prueba Dickey-Fuller para Regular: p-valor = 8.607670489056085e-12
 La varianza no es estable en la serie Regular.
 Aplicando diferenciación a Regular. Intento 2.
Prueba Dickey-Fuller para Regular: p-valor = 1.1514286931342563e-18
 La varianza no es estable en la serie Regular.
 Aplicando diferenciación a Regular. Intento 3.
Prueba Dickey-Fuller para Regular: p-valor = 1.4299582821292605e-21
 La varianza no es estable en la serie Regular.
 Aplicando diferenciación a Regular. Intento 4.
Prueba Dickey-Fuller para Regular: p-valor = 1.3446729756106482e-22
 La varianza no es estable en la serie Regular.
 Aplicando diferenciación a Regular. Intento 5.
Prueba Dickey-Fuller para Regular: p-valor = 2.3198979881378136e-21
 La varianza no es estable en la serie Regular.
 Aplicando diferenciación a Regular. Intento 6.
Prueba Dickey-Fuller para Regular: p-valor = 3.8522581992643975e-23
 La varianza no es estable en la serie Regular.
```

🔄 Aplicando diferenciación a Regular. Intento 7.
Prueba Dickey-Fuller para Regular: p-valor = 4.3203576796413914e-27
🔄 La varianza no es estable en la serie Regular.
🔄 Aplicando diferenciación a Regular. Intento 8.
Prueba Dickey-Fuller para Regular: p-valor = 1.9830657561682155e-29
🔄 La varianza no es estable en la serie Regular.
🔄 Aplicando diferenciación a Regular. Intento 9.
Prueba Dickey-Fuller para Regular: p-valor = 7.512287476786966e-30
🔄 La varianza no es estable en la serie Regular.
🔄 Aplicando diferenciación a Regular. Intento 10.
⚠️ No se pudo hacer la serie Regular estacionaria después de 10 intentos.
🔄 Utilizando un modelo ARIMA para tratar con la no estacionariedad.

SARIMAX Results

Dep. Variable:	Gasolina regular	No. Observations:	271
Model:	ARIMA(1, 1, 1)	Log Likelihood	-1459.374
Date:	Sun, 04 Aug 2024	AIC	2924.747
Time:	14:50:24	BIC	2935.542
Sample:	11-01-2001	HQIC	2929.082
	- 05-01-2024		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.9238	0.022	-42.730	0.000	-0.966	-0.881
ma.L1	-0.9995	2.543	-0.393	0.694	-5.983	3.984
sigma2	2808.7948	7124.660	0.394	0.693	-1.12e+04	1.68e+04

Ljung-Box (L1) (Q):	171.55	Jarque-Bera (JB):	6.09
Prob(Q):	0.00	Prob(JB):	0.05
Heteroskedasticity (H):	0.31	Skew:	

```
-0.02
Prob(H) (two-sided):                  0.00   Kurtosis:
3.73
===============================================================================
=============

Warnings:
[1] Covariance matrix calculated using the outer product of gradients
(complex-step).
🔍 Analizando la serie: Superior
🔍 Detección de valores inválidos en la serie: Superior
  Valores NaN: 0
  Valores no positivos: 0
  Serie temporal sin valores NaN tiene 1305 elementos.
Prueba Dickey-Fuller para Superior: p-valor = 0.021117919557505475
📉 La varianza no es estable en la serie Superior.
🔄 Aplicando diferenciación a Superior. Intento 1.
Prueba Dickey-Fuller para Superior: p-valor = 2.436091753573115e-16
📉 La varianza no es estable en la serie Superior.
🔄 Aplicando diferenciación a Superior. Intento 2.
Prueba Dickey-Fuller para Superior: p-valor = 3.159810795761913e-22
📉 La varianza no es estable en la serie Superior.
🔄 Aplicando diferenciación a Superior. Intento 3.
Prueba Dickey-Fuller para Superior: p-valor = 2.988167990607026e-27
📉 La varianza no es estable en la serie Superior.
🔄 Aplicando diferenciación a Superior. Intento 4.
Prueba Dickey-Fuller para Superior: p-valor = 2.845364748853002e-30
📉 La varianza no es estable en la serie Superior.
🔄 Aplicando diferenciación a Superior. Intento 5.
Prueba Dickey-Fuller para Superior: p-valor = 0.0
📉 La varianza no es estable en la serie Superior.
🔄 Aplicando diferenciación a Superior. Intento 6.
Prueba Dickey-Fuller para Superior: p-valor = 0.0
📉 La varianza no es estable en la serie Superior.
🔄 Aplicando diferenciación a Superior. Intento 7.
Prueba Dickey-Fuller para Superior: p-valor = 0.0
📉 La varianza no es estable en la serie Superior.
🔄 Aplicando diferenciación a Superior. Intento 8.
Prueba Dickey-Fuller para Superior: p-valor = 0.0
📉 La varianza no es estable en la serie Superior.
🔄 Aplicando diferenciación a Superior. Intento 9.
Prueba Dickey-Fuller para Superior: p-valor = 0.0
📉 La varianza no es estable en la serie Superior.
🔄 Aplicando diferenciación a Superior. Intento 10.
⚠️ No se pudo hacer la serie Superior estacionaria después de 10
intentos.
🔄 Utilizando un modelo ARIMA para tratar con la no estacionariedad.
                                SARIMAX Results
```

```
==========================================================================
========
Dep. Variable:     Superior GTQ/GALON   No. Observations:
1295
Model:                  ARIMA(1, 1, 1)   Log Likelihood               -
1718.628
Date:                 Sun, 04 Aug 2024   AIC
3443.255
Time:                         14:50:25   BIC
3458.752
Sample:                     01-11-2021   HQIC
3449.071
                          - 07-28-2024

Covariance Type:                   opg

==========================================================================
========
                 coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------
--------
ar.L1         -0.9007      0.006   -148.573      0.000      -0.913
-0.889
ma.L1         -0.9999      1.317     -0.759      0.448      -3.581
1.581
sigma2         0.8275      1.089      0.760      0.447      -1.307
2.962
==========================================================================
==========
Ljung-Box (L1) (Q):                  755.72   Jarque-Bera (JB):
1613.69
Prob(Q):                               0.00   Prob(JB):
0.00
Heteroskedasticity (H):                0.76   Skew:
0.00
Prob(H) (two-sided):                   0.01   Kurtosis:
8.47
==========================================================================
==========

Warnings:
[1] Covariance matrix calculated using the outer product of gradients
(complex-step).
```

Los resultados nos indican de esta forma de automatizar la busqueda de buenos parametros p, d y q resultó efectiva en todas las series. Esto basados en los resultados de AIC y BIC obtenidos. Aun así, se decidió implementar el modelo auto_arima para comparar la diferencia que existen entre estos métodos.

```
serie_diesel = serieConsumoDiesel['Diesel']

# Ajustar el modelo ARIMA automáticamente
modelo_auto = auto_arima(serie_diesel,
                         seasonal=False,  # Si no es estacional
                         stepwise=True,   # Si utilizar pasos para
encontrar el mejor modelo
                         trace=True)      # Mostrar el progreso

# Resumen del modelo
print(modelo_auto.summary())
```

```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=inf, Time=0.29 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=7447.955, Time=0.01 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=7440.658, Time=0.02 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=7440.979, Time=0.02 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=7446.248, Time=0.01 sec
 ARIMA(2,1,0)(0,0,0)[0] intercept   : AIC=7442.638, Time=0.04 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=7442.627, Time=0.04 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=7444.623, Time=0.08 sec
 ARIMA(1,1,0)(0,0,0)[0]             : AIC=7439.163, Time=0.02 sec
 ARIMA(2,1,0)(0,0,0)[0]             : AIC=7441.126, Time=0.02 sec
 ARIMA(1,1,1)(0,0,0)[0]             : AIC=7441.116, Time=0.04 sec
 ARIMA(0,1,1)(0,0,0)[0]             : AIC=7439.504, Time=0.02 sec
 ARIMA(2,1,1)(0,0,0)[0]             : AIC=7443.087, Time=0.06 sec

Best model:  ARIMA(1,1,0)(0,0,0)[0]
Total fit time: 0.677 seconds
                                SARIMAX Results

==========================================================================
========
Dep. Variable:                           y   No. Observations:
293
Model:                 SARIMAX(1, 1, 0)   Log Likelihood                -
3717.581
Date:                  Sun, 04 Aug 2024   AIC
7439.163
Time:                          17:10:33   BIC
7446.516
Sample:                      01-01-2000   HQIC
7442.108
                           - 05-01-2024

Covariance Type:                     opg

==========================================================================
========
                 coef    std err          z      P>|z|      [0.025
```

```
0.975]
------------------------------------------------------------------------------
--------
ar.L1          -0.1598      0.043     -3.719      0.000       -0.244
-0.076
sigma2      6.671e+09   4.16e-13    1.6e+22      0.000    6.67e+09
6.67e+09
===================================================================================
=============
Ljung-Box (L1) (Q):                   0.35   Jarque-Bera (JB):
1.44
Prob(Q):                              0.56   Prob(JB):
0.49
Heteroskedasticity (H):               2.34   Skew:
-0.14
Prob(H) (two-sided):                  0.00   Kurtosis:
3.19
===================================================================================
=============

Warnings:
[1] Covariance matrix calculated using the outer product of gradients
(complex-step).
[2] Covariance matrix is singular or near-singular, with condition
number    inf. Standard errors may be unstable.
```

```
serie_regular = serieImportacionRegular['Gasolina regular']

# Ajustar el modelo ARIMA automáticamente
modelo_auto = auto_arima(serie_regular,
                         seasonal=False,  # Si no es estacional
                         stepwise=True,   # Si utilizar pasos para
encontrar el mejor modelo
                         trace=True)      # Mostrar el progreso

# Resumen del modelo
print(modelo_auto.summary())
```

```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=7193.005, Time=0.09 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=7412.355, Time=0.01 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=7264.099, Time=0.01 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=7230.431, Time=0.05 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=7410.471, Time=0.01 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=7201.550, Time=0.07 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=7200.752, Time=0.06 sec
 ARIMA(3,1,2)(0,0,0)[0] intercept   : AIC=7193.700, Time=0.14 sec
 ARIMA(2,1,3)(0,0,0)[0] intercept   : AIC=7195.543, Time=0.13 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=7198.250, Time=0.04 sec
 ARIMA(1,1,3)(0,0,0)[0] intercept   : AIC=7200.602, Time=0.10 sec
```

```
 ARIMA(3,1,1)(0,0,0)[0] intercept   : AIC=7199.638, Time=0.06 sec
 ARIMA(3,1,3)(0,0,0)[0] intercept   : AIC=7189.517, Time=0.34 sec
 ARIMA(4,1,3)(0,0,0)[0] intercept   : AIC=inf, Time=0.35 sec
 ARIMA(3,1,4)(0,0,0)[0] intercept   : AIC=inf, Time=0.37 sec
 ARIMA(2,1,4)(0,0,0)[0] intercept   : AIC=inf, Time=0.40 sec
 ARIMA(4,1,2)(0,0,0)[0] intercept   : AIC=7195.774, Time=0.19 sec
 ARIMA(4,1,4)(0,0,0)[0] intercept   : AIC=inf, Time=0.45 sec
 ARIMA(3,1,3)(0,0,0)[0]             : AIC=7199.639, Time=0.16 sec

Best model:  ARIMA(3,1,3)(0,0,0)[0] intercept
Total fit time: 3.051 seconds
                                SARIMAX Results

==============================================================================
========
Dep. Variable:                           y   No. Observations:
281
Model:               SARIMAX(3, 1, 3)   Log Likelihood                -
3586.758
Date:                Sun, 04 Aug 2024   AIC
7189.517
Time:                        17:10:38   BIC
7218.595
Sample:                    01-01-2001   HQIC
7201.180
                         - 05-01-2024

Covariance Type:                   opg

==============================================================================
========
                 coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
--------
intercept    1.22e+04   2792.661      4.367      0.000    6721.770
1.77e+04
ar.L1         -1.4398      0.063    -22.896      0.000      -1.563
-1.317
ar.L2         -1.2383      0.086    -14.318      0.000      -1.408
-1.069
ar.L3         -0.2493      0.059     -4.218      0.000      -0.365
-0.133
ma.L1          0.3535      0.041      8.556      0.000       0.273
0.434
ma.L2         -0.0481      0.045     -1.062      0.288      -0.137
0.041
ma.L3         -0.8335      0.041    -20.336      0.000      -0.914
-0.753
sigma2       8.31e+09      0.001   1.53e+13      0.000    8.31e+09
```

```
8.31e+09
================================================================================
=============
Ljung-Box (L1) (Q):                   0.07   Jarque-Bera (JB):
37.62
Prob(Q):                              0.79   Prob(JB):
0.00
Heteroskedasticity (H):               3.23   Skew:
0.03
Prob(H) (two-sided):                  0.00   Kurtosis:
4.80
================================================================================
=============

Warnings:
[1] Covariance matrix calculated using the outer product of gradients
(complex-step).
[2] Covariance matrix is singular or near-singular, with condition
number 1.1e+29. Standard errors may be unstable.
```

```
serie_super = seriePreciosSuper['Superior GTQ/GALON']

# Ajustar el modelo ARIMA automáticamente
modelo_auto_super = auto_arima(serie_super,
                           seasonal=False,  # Si no es estacional
                           stepwise=True,   # Si utilizar pasos para
encontrar el mejor modelo
                           trace=True)      # Mostrar el progreso

# Resumen del modelo
print(modelo_auto.summary())
```

```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=773.104, Time=0.66 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=774.698, Time=0.05 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=776.697, Time=0.07 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=776.697, Time=0.05 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=773.556, Time=0.03 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=779.616, Time=0.11 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=770.964, Time=0.69 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=778.676, Time=0.12 sec
 ARIMA(2,1,0)(0,0,0)[0] intercept   : AIC=777.650, Time=0.05 sec
 ARIMA(3,1,1)(0,0,0)[0] intercept   : AIC=781.620, Time=0.09 sec
 ARIMA(3,1,0)(0,0,0)[0] intercept   : AIC=779.621, Time=0.08 sec
 ARIMA(3,1,2)(0,0,0)[0] intercept   : AIC=771.049, Time=0.99 sec
 ARIMA(2,1,1)(0,0,0)[0]             : AIC=769.409, Time=0.17 sec
 ARIMA(1,1,1)(0,0,0)[0]             : AIC=769.647, Time=0.21 sec
 ARIMA(2,1,0)(0,0,0)[0]             : AIC=776.461, Time=0.03 sec
 ARIMA(3,1,1)(0,0,0)[0]             : AIC=780.423, Time=0.06 sec
 ARIMA(2,1,2)(0,0,0)[0]             : AIC=771.199, Time=0.25 sec
```

```
 ARIMA(1,1,0)(0,0,0)[0]             : AIC=775.556, Time=0.02 sec
 ARIMA(1,1,2)(0,0,0)[0]             : AIC=778.410, Time=0.04 sec
 ARIMA(3,1,0)(0,0,0)[0]             : AIC=778.423, Time=0.10 sec
 ARIMA(3,1,2)(0,0,0)[0]             : AIC=768.220, Time=0.37 sec
 ARIMA(4,1,2)(0,0,0)[0]             : AIC=770.135, Time=0.55 sec
 ARIMA(3,1,3)(0,0,0)[0]             : AIC=760.596, Time=0.73 sec
 ARIMA(2,1,3)(0,0,0)[0]             : AIC=768.273, Time=0.54 sec
 ARIMA(4,1,3)(0,0,0)[0]             : AIC=753.584, Time=0.58 sec
 ARIMA(5,1,3)(0,0,0)[0]             : AIC=754.952, Time=0.76 sec
 ARIMA(4,1,4)(0,0,0)[0]             : AIC=755.142, Time=0.88 sec
 ARIMA(3,1,4)(0,0,0)[0]             : AIC=753.109, Time=0.72 sec
 ARIMA(2,1,4)(0,0,0)[0]             : AIC=770.272, Time=0.56 sec
 ARIMA(3,1,5)(0,0,0)[0]             : AIC=752.713, Time=0.76 sec
 ARIMA(2,1,5)(0,0,0)[0]             : AIC=750.713, Time=0.48 sec
 ARIMA(1,1,5)(0,0,0)[0]             : AIC=783.759, Time=0.26 sec
 ARIMA(1,1,4)(0,0,0)[0]             : AIC=782.318, Time=0.12 sec
 ARIMA(2,1,5)(0,0,0)[0] intercept   : AIC=751.866, Time=1.66 sec

Best model:  ARIMA(2,1,5)(0,0,0)[0]
Total fit time: 12.850 seconds
                                SARIMAX Results

==========================================================================
========
Dep. Variable:                          y   No. Observations:
281
Model:               SARIMAX(3, 1, 3)   Log Likelihood                -
3586.758
Date:                Sun, 04 Aug 2024   AIC
7189.517
Time:                        17:10:53   BIC
7218.595
Sample:                    01-01-2001   HQIC
7201.180
                         - 05-01-2024

Covariance Type:                   opg

==========================================================================
========
                 coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------
--------
intercept    1.22e+04   2792.661      4.367      0.000    6721.770
1.77e+04
ar.L1         -1.4398      0.063    -22.896      0.000      -1.563
-1.317
ar.L2         -1.2383      0.086    -14.318      0.000      -1.408
-1.069
```

```
ar.L3          -0.2493      0.059     -4.218      0.000      -0.365
-0.133
ma.L1           0.3535      0.041      8.556      0.000       0.273
0.434
ma.L2          -0.0481      0.045     -1.062      0.288      -0.137
0.041
ma.L3          -0.8335      0.041    -20.336      0.000      -0.914
-0.753
sigma2       8.31e+09      0.001   1.53e+13      0.000    8.31e+09
8.31e+09
===================================================================
=============
Ljung-Box (L1) (Q):                   0.07   Jarque-Bera (JB):
37.62
Prob(Q):                              0.79   Prob(JB):
0.00
Heteroskedasticity (H):               3.23   Skew:
0.03
Prob(H) (two-sided):                  0.00   Kurtosis:
4.80
===================================================================
=============

Warnings:
[1] Covariance matrix calculated using the outer product of gradients
(complex-step).
[2] Covariance matrix is singular or near-singular, with condition
number 1.1e+29. Standard errors may be unstable.
```

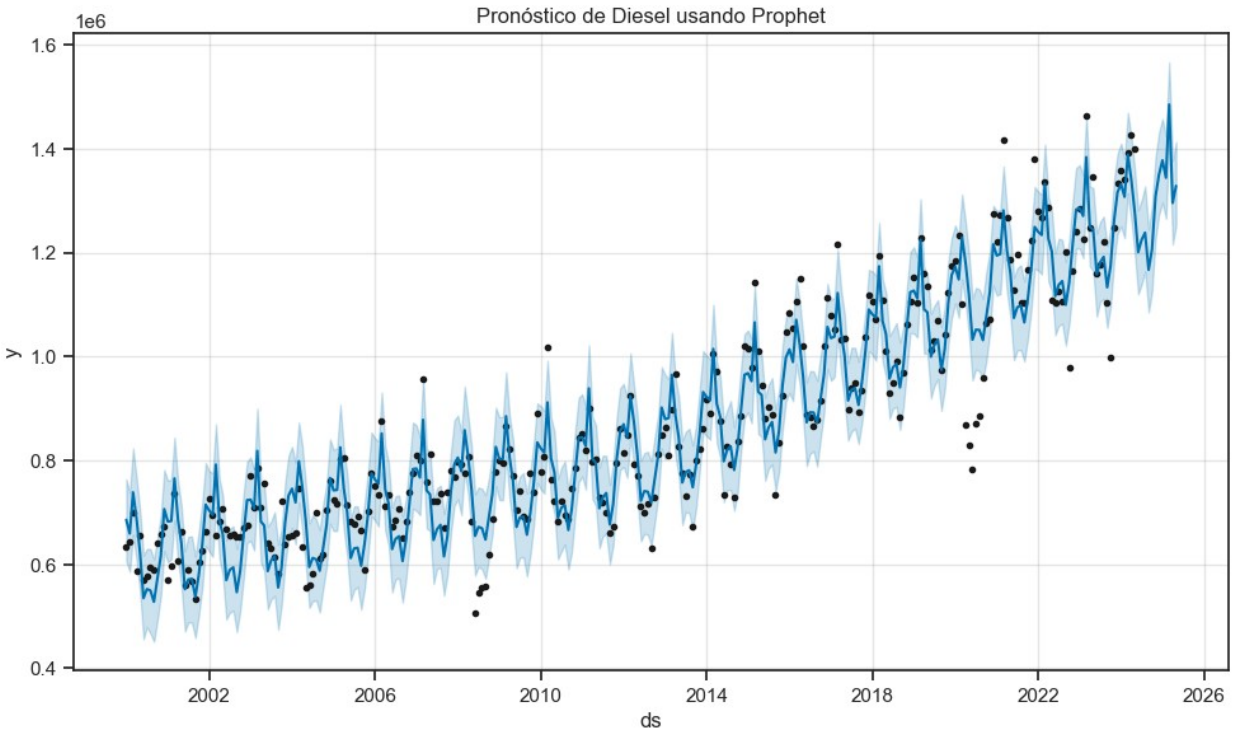
De esta manera se comprueba que la diferencia es significativamente alta, ya que entre los modelos propuestos anteriormente y los generados con el auto arima, existe una diferencia en terminos de AIC y de BIC de 3000 aproximadamente en todos los modelos.

Además, con fines de investigación y de verificar cómo las predicciones automatizadas con librerías funcionan, se decidió utilizar la librería de prophet de Facebook para evaluar su rendimiento.

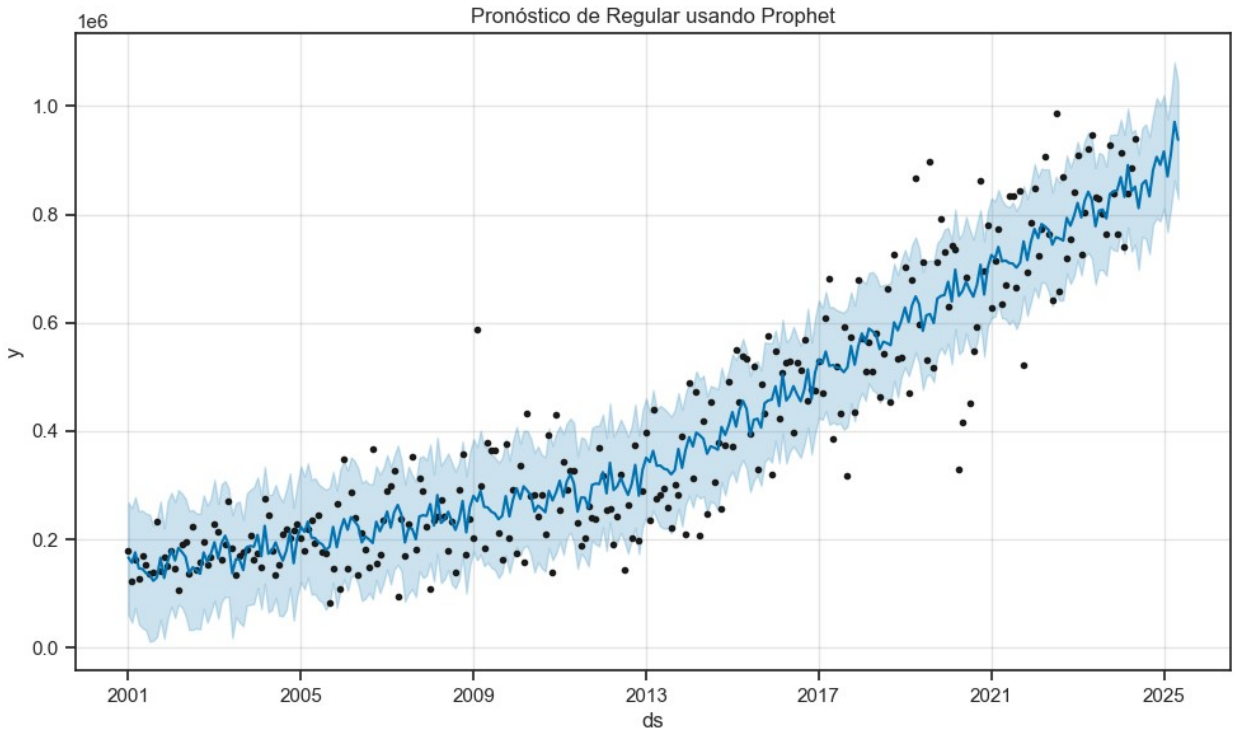
```python
def calcular_aic_bic(y_true, y_pred, num_params):
    # Calcular log-verosimilitud
    residuals = y_true - y_pred
    n = len(y_true)
    log_likelihood = -0.5 * (n * np.log(np.sum(residuals**2) / n))

    # Calcular AIC y BIC
    aic = 2 * num_params - 2 * log_likelihood
    bic = np.log(n) * num_params - 2 * log_likelihood

    return aic, bic
```

```python
def entrenarProphet(serieTiempo, nombre_variable):
    # Preparar los datos
    df_prophet = serieTiempo.reset_index()
    df_prophet.columns = ['ds', 'y']

    # Crear y ajustar el modelo Prophet
    modelo = Prophet()
    modelo.fit(df_prophet)

    # Hacer predicciones
    futuro = modelo.make_future_dataframe(periods=12, freq='M')  #
Cambia el número y la frecuencia según tus necesidades
    pronostico = modelo.predict(futuro)

    # Graficar resultados
    modelo.plot(pronostico)
    plt.title(f'Pronóstico de {nombre_variable} usando Prophet')
    plt.show()

    # Calcular AIC y BIC
    y_true = df_prophet['y'].values
    y_pred = pronostico['yhat'][:len(df_prophet)].values
    num_params = len(modelo.params)  # Número de parámetros del modelo

    aic, bic = calcular_aic_bic(y_true, y_pred, num_params)

    print(f"AIC para {nombre_variable}: {aic}")
    print(f"BIC para {nombre_variable}: {bic}")

# Entrenar el modelo para el Diesel
entrenarProphet(serieConsumoDiesel[['Diesel']], 'Diesel')
entrenarProphet(serieImportacionRegular[['Gasolina regular']],
'Regular')
entrenarProphet(seriePreciosSuper[['Superior GTQ/GALON']], 'Superior')
```

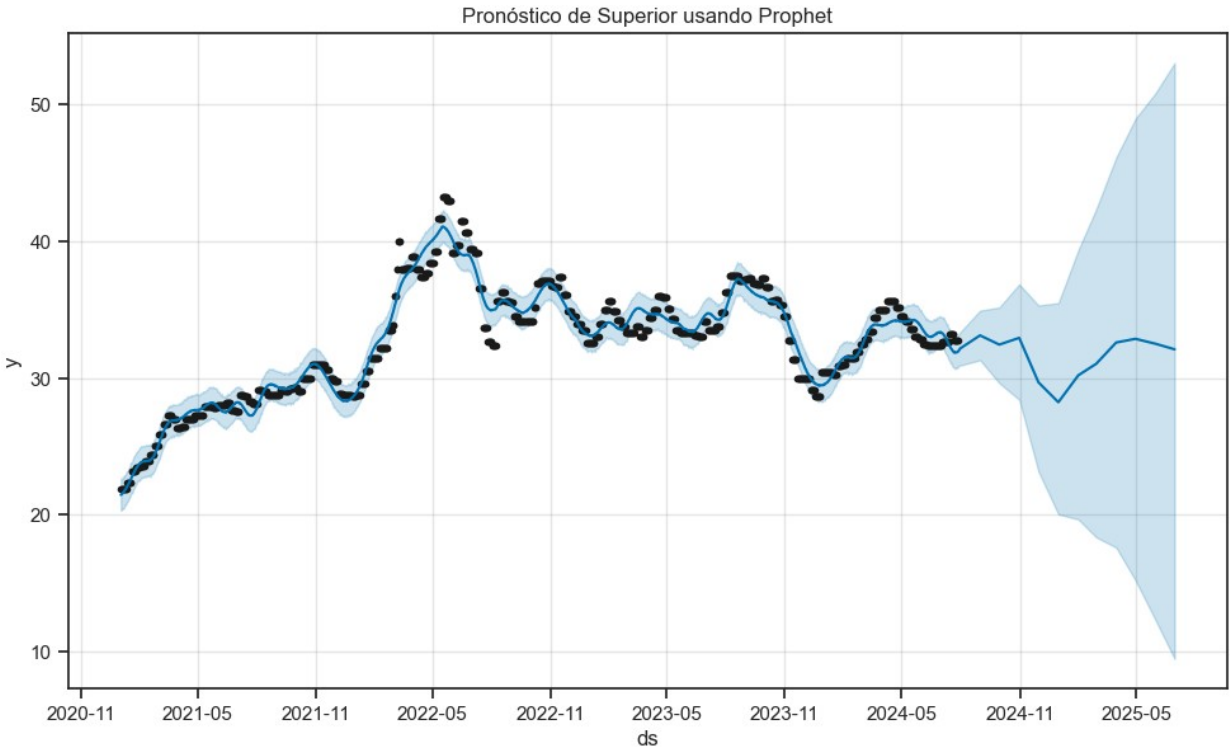
```
17:20:53 - cmdstanpy - INFO - Chain [1] start processing
17:20:53 - cmdstanpy - INFO - Chain [1] done processing
```




```
17:20:53 - cmdstanpy - INFO - Chain [1] start processing
17:20:53 - cmdstanpy - INFO - Chain [1] done processing

AIC para Diesel: 6480.70112158008
BIC para Diesel: 6506.4623298432
```




```
17:20:54 - cmdstanpy - INFO - Chain [1] start processing

AIC para Regular: 6387.850788531148
BIC para Regular: 6413.319271216485

17:20:54 - cmdstanpy - INFO - Chain [1] done processing
```




```
AIC para Superior: -279.6520722629985
BIC para Superior: -243.43436402470093
```

Por una parte, se logran realizar predicciones coherentes como las de diesel y gasolina regular, pero por otra parte, es evidente que las predicciones en algunos casos no cuentan con ningún sentido dado que puede ocurrir cuaquier evento debido al rango que muestran. El ejemplo más evidente es con la serie de precios promedio de la gasolina super. Aún así, es destacable la forma en la que se representan los puntos atípicos, evidenciando las épocas donde se vieron afectadas las series por acotecimientos globales como la pandemia o conflictos geopolíticos.

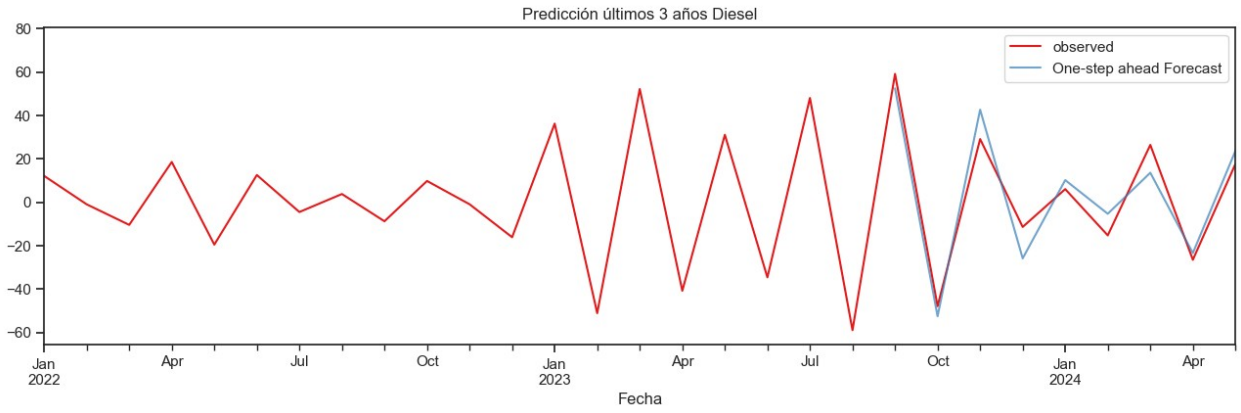
Dejando lo anterior de lado, los valores de AIC y BIC son significativamente mejores que los de auto arima hablando de las series de Diesel y de Gasolina regular, sin embargo, siguen siendo mejores los que se determinaron manualmente. Por otra parte, resulta interesante la predicción de gasolina superior dado que los valores de AIC y BIC son los mejores obtenidos de todos los métodos implementados.

Predicciones ultimos 3 años

```
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_absolute_error,
mean_absolute_percentage_error


ultimos_3_anos = serieTemporalD['2022-01-01':'2024-12-31']

# Dividir los datos en 70% entrenamiento y 30% prueba
n = len(ultimos_3_anos)
```

```
train_size = int(n * 0.7)
train, test = ultimos_3_anos[:train_size], ultimos_3_anos[train_size:]

pred = modelo_arima_diesel.get_prediction(start=test.index[0],
end=test.index[-1])

pred_ci = pred.conf_int()
ax = serieTemporalD['2022':].plot(label='observed')
pred.predicted_mean.plot(ax=ax, label='One-step ahead Forecast',
alpha=.7, figsize=(14, 4))
ax.fill_between( pred_ci.iloc[:,0],
                pred_ci.iloc[:,1], color='k', alpha=.2)

# Evaluar el modelo
rmse = np.sqrt(mean_squared_error(test, pred.predicted_mean))
mae = mean_absolute_error(test, pred.predicted_mean)
mape = mean_absolute_percentage_error(test, pred.predicted_mean)

print(f'RMSE: {rmse:.2f}')
print(f'MAE: {mae:.2f}')
print(f'MAPE: {mape * 100:.2f}%')
plt.legend()
plt.title(f'Predicción últimos 3 años Diesel')
plt.show()
```

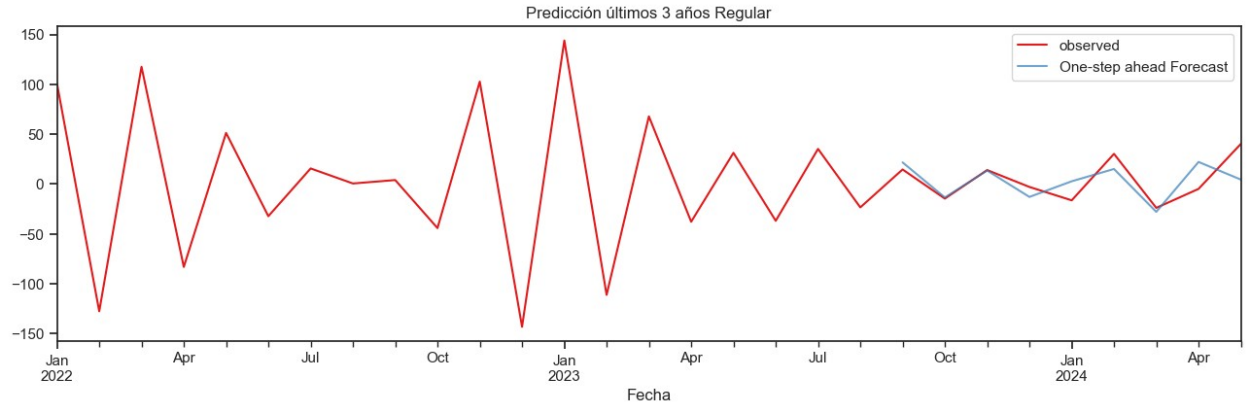
```
RMSE: 9.34
MAE: 8.38
MAPE: 47.00%
```



Para la serie de tiempo de consumo de diesel se utilizó información de los últimos 3 años y concretamente en los meses entre septiembre y abril de 2024, se realizó una predicción y se evaluaron para ver qué tanto se asemeajaba a la información real de la serie de tiempo. En la misma, se puede ver que se asemeja bastante visualmente, además el RMSE y el MAE obtenidos son relativamente bajos, lo que indica una buena prediccion. Por otra parte, el MAPE fué de 47% por lo que se puede decir que la predicción es razonable.

```
ultimos_3_anos = serieTemporalR['2022-01-01':'2024-12-31']

# Dividir los datos en 70% entrenamiento y 30% prueba
n = len(ultimos_3_anos)
train_size = int(n * 0.7)
train, test = ultimos_3_anos[:train_size], ultimos_3_anos[train_size:]

pred = modelo_arima_regular.get_prediction(start=test.index[0],
end=test.index[-1])

pred_ci = pred.conf_int()
ax = serieTemporalR['2022':].plot(label='observed')
pred.predicted_mean.plot(ax=ax, label='One-step ahead Forecast',
alpha=.7, figsize=(14, 4))
ax.fill_between( pred_ci.iloc[:,0],
                pred_ci.iloc[:,1], color='k', alpha=.2)

# Evaluar el modelo
rmse = np.sqrt(mean_squared_error(test, pred.predicted_mean))
mae = mean_absolute_error(test, pred.predicted_mean)
mape = mean_absolute_percentage_error(test, pred.predicted_mean)

print(f'RMSE: {rmse:.2f}')
print(f'MAE: {mae:.2f}')
print(f'MAPE: {mape * 100:.2f}%')

plt.legend()
plt.title(f'Predicción últimos 3 años Regular')
plt.show()

pred
```

```
RMSE: 17.56
MAE: 13.31
MAPE: 136.39%
```

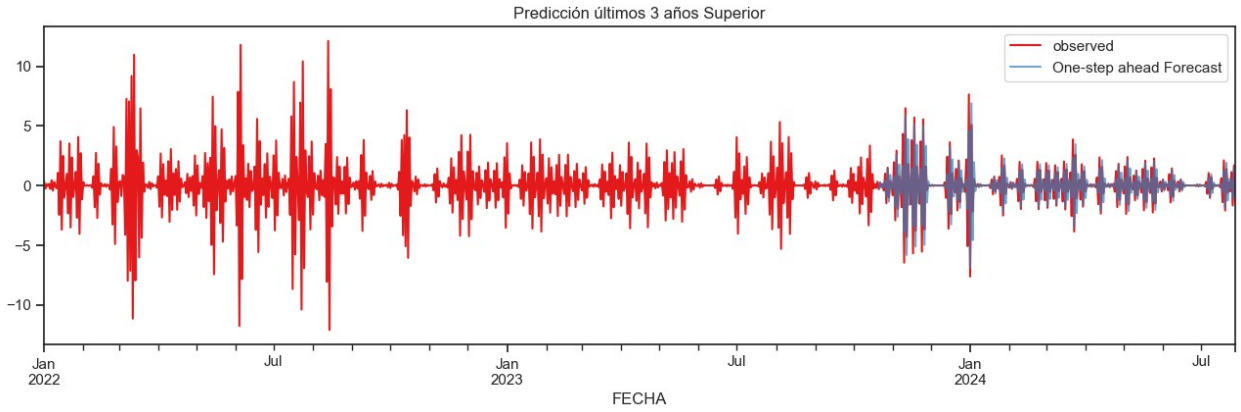


```
<statsmodels.tsa.statespace.mlemodel.PredictionResultsWrapper at
0x2dc4c777f90>
```

Para esta predicción podemos ver en la gráfica de la serie de tiempo, la predicción inicia alineada a los datos reales, sin embargo se desalinea en las siguientes partes. El RMSE y el MAE son relativamente bajos aunque más altos que la anterior, sin embargo el MAPE es casi del 136% por lo que indica una predicción bastante pobre.

```
ultimos_3_anos = serieTemporalS['2022-01-01':'2024-12-31']

# Dividir los datos en 70% entrenamiento y 30% prueba
n = len(ultimos_3_anos)
train_size = int(n * 0.7)
train, test = ultimos_3_anos[:train_size], ultimos_3_anos[train_size:]

pred = modelo_arima_super.get_prediction(start=test.index[0],
end=test.index[-1])

pred_ci = pred.conf_int()
ax = serieTemporalS['2022':].plot(label='observed')
pred.predicted_mean.plot(ax=ax, label='One-step ahead Forecast',
alpha=.7, figsize=(14, 4))
ax.fill_between( pred_ci.iloc[:,0],
                pred_ci.iloc[:,1], color='k', alpha=.2)

# Evaluar el modelo
rmse = np.sqrt(mean_squared_error(test, pred.predicted_mean))
mae = mean_absolute_error(test, pred.predicted_mean)
mape = mean_absolute_percentage_error(test, pred.predicted_mean)

print(f'RMSE: {rmse:.2f}')
print(f'MAE: {mae:.2f}')
print(f'MAPE: {mape * 100:.2f}%')

plt.legend()
plt.title(f'Predicción últimos 3 años Superior')
plt.show()

pred
```

```
RMSE: 0.77
MAE: 0.52
MAPE: 7882774729377.25%
```



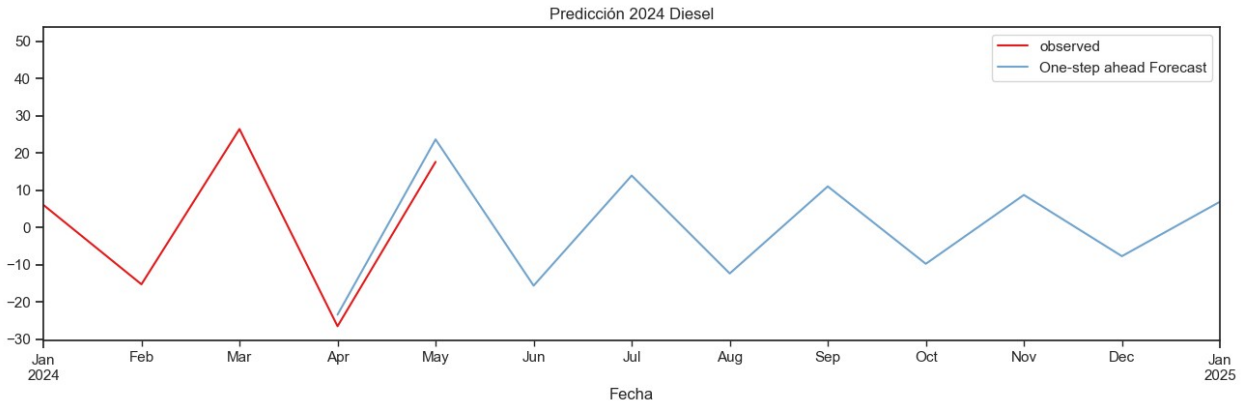

```
<statsmodels.tsa.statespace.mlemodel.PredictionResultsWrapper at
0x2dc68fb6b50>
```

Para esta predicción podemos ver en la gráfica de la serie de tiempo que la predicción se mantiene bastante alineada a los datos reales. El RMSE y el MAE son bastante bajos, lo que podría dar indicios de una buena predicción, sin embargo el MAPE es demasiado grande, indicandonos que igualmente sería una predicción bastante pobre.

Prediccion para 2024

```
actual_year = serieTemporalD['2024-01-01':'2024-12-31']

# Dividir los datos en 70% entrenamiento y 30% prueba
n = len(actual_year)
train_size = int(n * 0.7)
train, test = actual_year[:train_size], actual_year[train_size:]

pred = modelo_arima_diesel.get_prediction(start=test.index[0],
end='2024-12-31')

pred_ci = pred.conf_int()
ax = serieTemporalD['2024':].plot(label='observed')
pred.predicted_mean.plot(ax=ax, label='One-step ahead Forecast',
alpha=.7, figsize=(14, 4))
ax.fill_between( pred_ci.iloc[:,0],
                pred_ci.iloc[:,1], color='k', alpha=.2)




plt.legend()
plt.title(f'Predicción 2024 Diesel')
plt.show()

# Alinear las predicciones y los valores reales para asegurarse de que
tienen la misma longitud
```

```
predictions = pred.predicted_mean.loc[test.index]
true_values = test

# Calcular intervalo de confianza
pred_ci = pred.conf_int().loc[test.index]

# Calcular errores
rmse = np.sqrt(mean_squared_error(true_values, predictions))
mae = mean_absolute_error(true_values, predictions)
mape = np.mean(np.abs((true_values - predictions) / true_values)) *
100

print(f'RMSE: {rmse}')
print(f'MAE: {mae}')
print(f'MAPE: {mape:.2f}%')
```

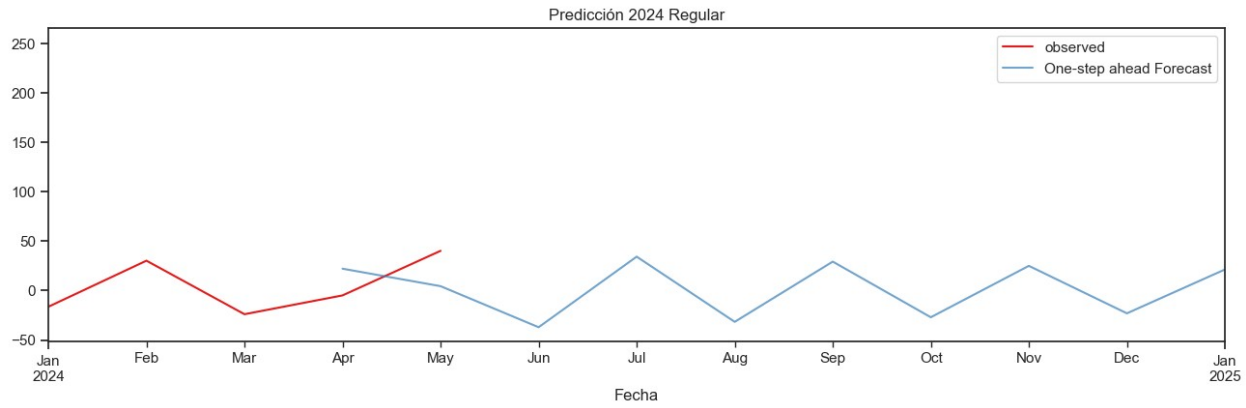



```
RMSE: 4.790903186877741
MAE: 4.561657135595812
MAPE: 22.95%
```

Haciendo la predicción pero solo con el año 2024 para el consumo de diesel, podemos ver que empieza bastante alineada con los datos reales. El RMSE y el MAE son relativamente bajos y el MAPE es del casi 23% lo que nos dice que no es una excelente predicción pero es bastante razonable.

```
actual_year = serieTemporalR['2024-01-01':'2024-12-31']

# Dividir los datos en 70% entrenamiento y 30% prueba
n = len(actual_year)
train_size = int(n * 0.7)
train, test = actual_year[:train_size], actual_year[train_size:]

pred = modelo_arima_regular.get_prediction(start=test.index[0],
end='2024-12-31')

pred_ci = pred.conf_int()
```

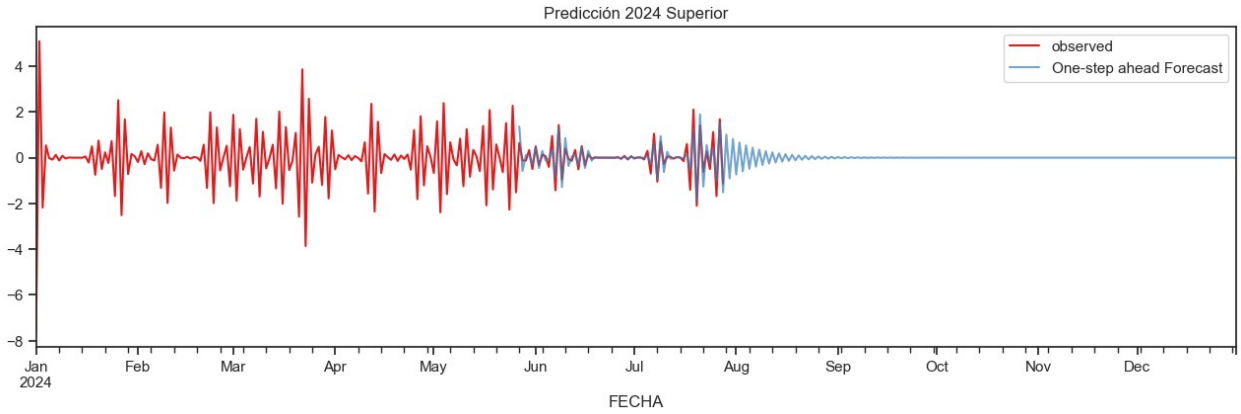
```
ax = serieTemporalR['2024':].plot(label='observed')
pred.predicted_mean.plot(ax=ax, label='One-step ahead Forecast',
alpha=.7, figsize=(14, 4))
ax.fill_between( pred_ci.iloc[:,0],
                pred_ci.iloc[:,1], color='k', alpha=.2)

plt.title(f'Predicción 2024 Regular')
plt.legend()
plt.show()


# Alinear las predicciones y los valores reales para asegurarse de que
tienen la misma longitud
predictions = pred.predicted_mean.loc[test.index]
true_values = test

# Calcular intervalo de confianza
pred_ci = pred.conf_int().loc[test.index]

# Calcular errores
rmse = np.sqrt(mean_squared_error(true_values, predictions))
mae = mean_absolute_error(true_values, predictions)
mape = np.mean(np.abs((true_values - predictions) / true_values)) *
100

print(f'RMSE: {rmse}')
print(f'MAE: {mae}')
print(f'MAPE: {mape:.2f}%')
```




```
RMSE: 31.710895539342097
MAE: 31.408330118821873
MAPE: 321.42%
```

En la predicción de serie de tiempo de importación regular con el año 2024 podemos ver que el RMSE y el MAE son más altos de lo usual y el MAPE sobrepasa el 100%, lo que indicaría una predicción bastante pobre. En comparativa, es mejor la predicción que cuenta con más años antes del 2024.

```
actual_year = serieTemporalS['2024-01-01':'2024-12-31']

# Dividir los datos en 70% entrenamiento y 30% prueba
n = len(actual_year)
train_size = int(n * 0.7)
train, test = actual_year[:train_size], actual_year[train_size:]

pred = modelo_arima_super.get_prediction(start=test.index[0],
end='2024-12-31')

pred_ci = pred.conf_int()
ax = serieTemporalS['2024':].plot(label='observed')
pred.predicted_mean.plot(ax=ax, label='One-step ahead Forecast',
alpha=.7, figsize=(14, 4))
ax.fill_between( pred_ci.iloc[:,0],
                pred_ci.iloc[:,1], color='k', alpha=.2)

plt.legend()
plt.title(f'Predicción 2024 Superior')
plt.show()


# Alinear las predicciones y los valores reales para asegurarse de que
tienen la misma longitud
predictions = pred.predicted_mean.loc[test.index]
true_values = test

# Calcular intervalo de confianza
pred_ci = pred.conf_int().loc[test.index]

# Calcular errores
rmse = np.sqrt(mean_squared_error(true_values, predictions))
mae = mean_absolute_error(true_values, predictions)
mape = np.mean(np.abs((true_values - predictions) / true_values)) *
100

print(f'RMSE: {rmse}')
print(f'MAE: {mae}')
print(f'MAPE: {mape:.2f}%')
```



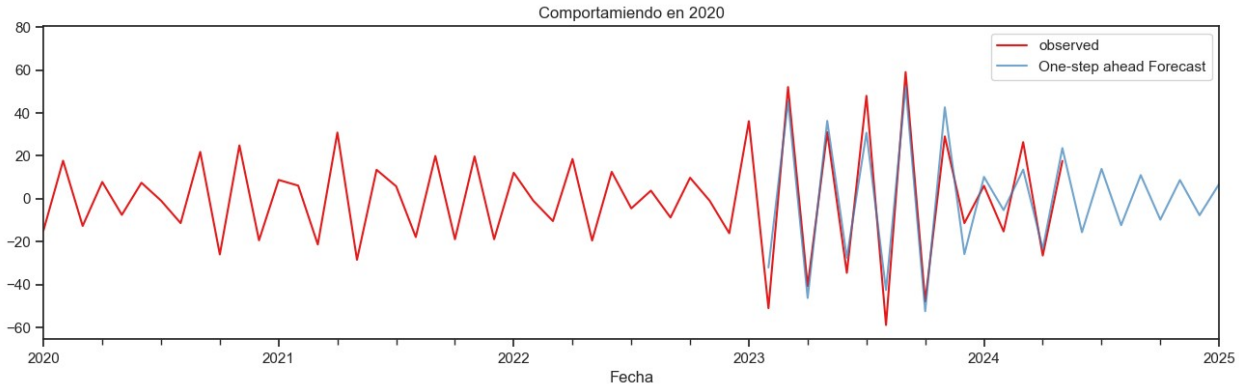

```
RMSE: 0.32406539665899864
MAE: 0.23132556421351486
MAPE: inf%
```

Por último en la predicción de los precios de la gasolina superior, se puede observar que el modelo del año 2024 no fue el adecuado para realizar la predicción, ya que el modelo casi no se ajusta a los datos reales. A pesar que el RMSE y el MAE salieron batante bajos, MAPE llego casi a infinito indicando una predicción muy pobre, igualmente se puede observar esa mala predicción en la gráfica.

Comportamiendo durante pandemia

```
actual_year = serieTemporalD['2020-01-01':]

# Dividir los datos en 70% entrenamiento y 30% prueba
n = len(actual_year)
train_size = int(n * 0.7)
train, test = actual_year[:train_size], actual_year[train_size:]

pred = modelo_arima_diesel.get_prediction(start=test.index[0],
end='2024-12-31')

pred_ci = pred.conf_int()
ax = serieTemporalD['2020':].plot(label='observed')
pred.predicted_mean.plot(ax=ax, label='One-step ahead Forecast',
alpha=.7, figsize=(14, 4))
ax.fill_between( pred_ci.iloc[:,0],
                pred_ci.iloc[:,1], color='k', alpha=.2)


plt.legend()
plt.title(f'Comportamiendo en 2020')
plt.show()

# Alinear las predicciones y los valores reales para asegurarse de que
tienen la misma longitud
predictions = pred.predicted_mean.loc[test.index]
```

```
true_values = test

# Calcular intervalo de confianza
pred_ci = pred.conf_int().loc[test.index]

# Calcular errores
rmse = np.sqrt(mean_squared_error(true_values, predictions))
mae = mean_absolute_error(true_values, predictions)
mape = np.mean(np.abs((true_values - predictions) / true_values)) *
100

print(f'RMSE: {rmse}')
print(f'MAE: {mae}')
print(f'MAPE: {mape:.2f}%')
```

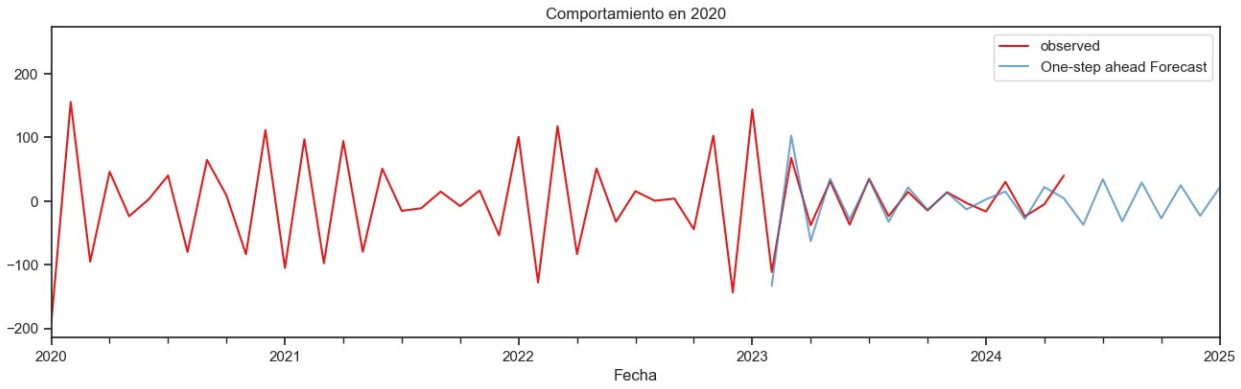



```
RMSE: 10.78909174355549
MAE: 9.528774374289872
MAPE: 36.71%
```

Podemos ver que para el consumo de gasolina diesel durante 2020 fue más bajo de lo normal, luego para 2021 y 2022 se puede ver un poco de aumento, hasta 2023 que este aumento fue significativo. Debido a la baja de consumo durante el año 2020 y la costosa recuperación entre 2021 y 2022, se puede considerar que puede afectar la predicción ya que se pudo ver un abrupto cambio en 2023 a comparación de los años cercanos a la pandemia ya que el valle generado en la pandemia no es considerado un comportamiento usual en el consumo.

```
actual_year = serieTemporalR['2020-01-01':]

# Dividir los datos en 70% entrenamiento y 30% prueba
n = len(actual_year)
train_size = int(n * 0.7)
train, test = actual_year[:train_size], actual_year[train_size:]

pred = modelo_arima_regular.get_prediction(start=test.index[0],
end='2024-12-31')
```

```
pred_ci = pred.conf_int()
ax = serieTemporalR['2020':].plot(label='observed')
pred.predicted_mean.plot(ax=ax, label='One-step ahead Forecast',
alpha=.7, figsize=(14, 4))
ax.fill_between( pred_ci.iloc[:,0],
                pred_ci.iloc[:,1], color='k', alpha=.2)

plt.title(f'Comportamiento en 2020')
plt.legend()
plt.show()


# Alinear las predicciones y los valores reales para asegurarse de que
tienen la misma longitud
predictions = pred.predicted_mean.loc[test.index]
true_values = test

# Calcular intervalo de confianza
pred_ci = pred.conf_int().loc[test.index]

# Calcular errores
rmse = np.sqrt(mean_squared_error(true_values, predictions))
mae = mean_absolute_error(true_values, predictions)
mape = np.mean(np.abs((true_values - predictions) / true_values)) *
100

print(f'RMSE: {rmse}')
print(f'MAE: {mae}')
print(f'MAPE: {mape:.2f}%')
```

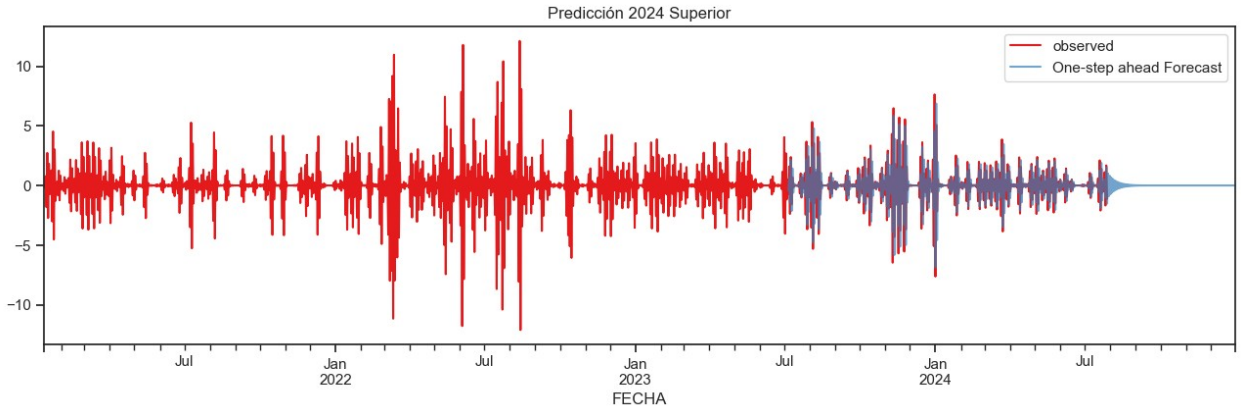



```
RMSE: 18.11992975727239
MAE: 13.964687409995861
MAPE: 90.00%
```

Para la importación de gasolina regular se puede ver que no hubo mayor cambio en la fluctuación de importaciones ya que en 2020 mostraban un alza, luego bajo un poco, a inicios de 2021 volvió a subir y bajo una temporada, luego 2022 vuelve a subir y bajar al final. La pequeña

diferencia es que en 2023, la parte alta fue un poco más que los años anteriores, probablemente por la recuperación que se vió en ese año, hasta llegar a 2024 sin cambiar casi nada a inicios de ese mismo año. Podemos ver que la predicción no es tan buena y puede deberse a las fluctuaciones de cada año.

```
actual_year = serieTemporalS['2021-01-01':]

# Dividir los datos en 70% entrenamiento y 30% prueba
n = len(actual_year)
train_size = int(n * 0.7)
train, test = actual_year[:train_size], actual_year[train_size:]

pred = modelo_arima_super.get_prediction(start=test.index[0],
end='2024-12-31')

pred_ci = pred.conf_int()
ax = serieTemporalS['2021':].plot(label='observed')
pred.predicted_mean.plot(ax=ax, label='One-step ahead Forecast',
alpha=.7, figsize=(14, 4))
ax.fill_between( pred_ci.iloc[:,0],
                pred_ci.iloc[:,1], color='k', alpha=.2)

plt.legend()
plt.title(f'Predicción 2024 Superior')
plt.show()


# Alinear las predicciones y los valores reales para asegurarse de que
tienen la misma longitud
predictions = pred.predicted_mean.loc[test.index]
true_values = test

# Calcular intervalo de confianza
pred_ci = pred.conf_int().loc[test.index]

# Calcular errores
rmse = np.sqrt(mean_squared_error(true_values, predictions))
mae = mean_absolute_error(true_values, predictions)
mape = np.mean(np.abs((true_values - predictions) / true_values)) *
100

print(f'RMSE: {rmse}')
print(f'MAE: {mae}')
print(f'MAPE: {mape:.2f}%')
```




```
RMSE: 0.7509319468696994
MAE: 0.5098981943527374
MAPE: inf%
```

Finalmente tenemos los precios de la gasolina super, la cual es un poco más dificil de ver la predicción de la gráfica ya que solo cuenta con el año 2021 en adelante. Aún así, basandonos en el comportamiento de predicciones anteriores se ve que durante el año 2021 hasta inicios del 2022, los cambios registrados fueron un poco más bajos, lo cual se le podría atribuir a la pandemia. Luego en el 2022 hubo un aumento significativo hasta llegar a 2023 donde se registró una pequeña disminución que se mantuvo hasta 2024. En este modelo podemos ver que tanto en periodos anteriores como actuales, cuentan con cierta incertidumbre de predicción a futuro por lo que es considerado uno de los peores modelos que se pudieron dar entre los de gasolina regular y diesel, por lo que a pesar que no cuenta con el año 2020 para determinar si la pandemia fue lo que afectó principalmente, se puede evidenciar que fué el peor modelo de predicción.

Posiblemente el mejor de todos y el único modelo razonable de predicción fue el de consumo de diesel, ya que fué el que mejor pudo predecir y evidenciar fluctuaciones durante la pandemia.