

Universidad de las Américas – UDLA



Estudiantes:

Cahueñas Vizuite Stephanie Mishell

Toral Rodriguez Stephany Haydee

Posgrado: Maestría en Inteligencia de Negocios y Ciencia de Datos.

2023 – 2024

Módulo: Analítica Predictiva

Tema: Caso Final – Regresión Lineal Múltiple

Fecha: 6 agosto 2024

En primera instancia, hay que denotar que intentamos hacer dos posibles modelos usando dos metodologías distintas. En primer lugar, se realizó con la base datos una regresión logística la cual no funcionó dado que se concluyó que no teníamos una variable categórica lo cual dificulta el análisis en general. Por lo cual, se decidió modelar por medio de regresión lineal múltiple, estos dos modelos se van a mostrar a continuación como muestra de los intentos realizados para el ejercicio correspondiente al reto final.

En la primera parte consta el intento #1, correspondiente a la regresión logística.

1. Importar la base de datos a una base en Jupyter con pandas

En primer lugar, se llama a la librería pandas para poder trabajar con la base de datos Walmart

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

import warnings
warnings.filterwarnings("ignore")

[ ] import statsmodels.stats.api as sms
import statsmodels.api as sm
from statsmodels.formula.api import ols
from statsmodels.compat import lzip
```

Igualmente se puede visualizar una tabla resumen en donde constan las variables a analizar:

```
df = pd.read_csv("/content/Walmart(1).csv")
df.head(10)
```

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
0	1	05-02-2010	1643690.90	0	42.31	2.572	211.096358	8.106
1	1	12-02-2010	1641957.44	1	38.51	2.548	211.242170	8.106
2	1	19-02-2010	1611968.17	0	39.93	2.514	211.289143	8.106
3	1	26-02-2010	1409727.59	0	46.63	2.561	211.319643	8.106
4	1	05-03-2010	1554806.68	0	46.50	2.625	211.350143	8.106
5	1	12-03-2010	1439541.59	0	57.79	2.667	211.380643	8.106
6	1	19-03-2010	1472515.79	0	54.58	2.720	211.215635	8.106
7	1	26-03-2010	1404429.92	0	51.45	2.732	211.018042	8.106
8	1	02-04-2010	1594968.28	0	62.27	2.719	210.820450	7.808
9	1	09-04-2010	1545418.53	0	65.86	2.770	210.622857	7.808

Adicionalmente, para facilidad de trabajo con la base de datos, se cambió los nombres a español:

```
df.rename({'Store':'Tienda', 'Date':'Fecha', 'Weekly_Sales':'Ventas', 'Holiday_Flag': 'Festivo',
          'Temperature': 'Temperatura', 'Fuel_Price': 'Combustible',
          'Unemployment':'Desempleo', }, axis=1, inplace=True)
df.head(10)
```

	Tienda	Fecha	Ventas	Festivo	Temperatura	Combustible	CPI	Desempleo
0	1	05-02-2010	1643690.90	0	42.31	2.572	211.096358	8.106
1	1	12-02-2010	1641957.44	1	38.51	2.548	211.242170	8.106
2	1	19-02-2010	1611968.17	0	39.93	2.514	211.289143	8.106
3	1	26-02-2010	1409727.59	0	46.63	2.561	211.319643	8.106
4	1	05-03-2010	1554806.68	0	46.50	2.625	211.350143	8.106
5	1	12-03-2010	1439541.59	0	57.79	2.667	211.380643	8.106
6	1	19-03-2010	1472515.79	0	54.58	2.720	211.215635	8.106
7	1	26-03-2010	1404429.92	0	51.45	2.732	211.018042	8.106
8	1	02-04-2010	1594968.28	0	62.27	2.719	210.820450	7.808
9	1	09-04-2010	1545418.53	0	65.86	2.770	210.622857	7.808

2. Obtenga los descriptivos resumen de la base de datos e identifique las variables numéricas y categóricas. ¿Hay algo que le llame la atención?

Como se observa en las tablas anteriores, las variables ya se pueden observar, aun no se conoce si son numéricas o categóricas, pero si se puede notar que se requiere hacer un tratamiento con la variable fecha para poder trabajar de manera más sencilla posteriormente, por lo cual, se decide separar la fecha en día, mes y año para que formen columnas completas, del siguiente modo:

```
[ ] df['Fecha'] = pd.to_datetime(df['Fecha'], format='%d-%m-%Y') # Specify the correct date format
```

```
df['Dia'] = df['Fecha'].dt.day
df['Mes'] = df['Fecha'].dt.month
df['Año'] = df['Fecha'].dt.year

df
```

De modo que, las variables se pueden visualizar de la siguiente manera:

	Tienda	Fecha	Ventas	Festivo	Temperatura	Combustible	CPI	Desempleo	Dia	Mes	Año
0	1	2010-02-05	1643690.90	0	42.31	2.572	211.096358	8.106	5	2	2010
1	1	2010-02-12	1641957.44	1	38.51	2.548	211.242170	8.106	12	2	2010
2	1	2010-02-19	1611968.17	0	39.93	2.514	211.289143	8.106	19	2	2010
3	1	2010-02-26	1409727.59	0	46.63	2.561	211.319643	8.106	26	2	2010
4	1	2010-03-05	1554806.68	0	46.50	2.625	211.350143	8.106	5	3	2010
...
6430	45	2012-09-28	713173.95	0	64.88	3.997	192.013558	8.684	28	9	2012
6431	45	2012-10-05	733455.07	0	64.89	3.985	192.170412	8.667	5	10	2012
6432	45	2012-10-12	734464.36	0	54.47	4.000	192.327265	8.667	12	10	2012
6433	45	2012-10-19	718125.53	0	56.47	3.969	192.330854	8.667	19	10	2012
6434	45	2012-10-26	760281.43	0	58.85	3.882	192.308899	8.667	26	10	2012

6435 rows × 11 columns

Como se observa, la variable Fecha se encuentra ya separada en sus componentes, lo que facilitará el análisis y evitará la formación de alrededor de 200 variables por las fechas. A continuación se puede visualizar la clase de variables con las que se cuenta.

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6435 entries, 0 to 6434
Data columns (total 11 columns):
#   Column          Non-Null Count  Dtype
---  -
0   Tienda           6435 non-null   int64
1   Fecha            6435 non-null   datetime64[ns]
2   Ventas           6435 non-null   float64
3   Festivo          6435 non-null   int64
4   Temperatura      6435 non-null   float64
5   Combustible      6435 non-null   float64
6   CPI              6435 non-null   float64
7   Desempleo        6435 non-null   float64
8   Dia              6435 non-null   int32
9   Mes              6435 non-null   int32
10  Año              6435 non-null   int32
dtypes: datetime64[ns](1), float64(5), int32(3), int64(2)
memory usage: 477.7 KB
```

De este modo, se puede tener una idea general del tipo de variables que tiene la base de datos, a continuación se muestran las distintas variables, en primera instancia se encuentran las variables numéricas:

Numéricas

```
df.describe()
```

	Tienda	Fecha	Ventas	Festivo	Temperatura	Combustible	CPI	Desempleo	Dia	Mes	Año
count	6435.000000	6435	6.435000e+03	6435.000000	6435.000000	6435.000000	6435.000000	6435.000000	6435.000000	6435.000000	6435.000000
mean	23.000000	2011-06-17 00:00:00	1.046965e+06	0.069930	60.663782	3.358607	171.578394	7.999151	15.678322	6.447552	2010.965035
min	1.000000	2010-02-05 00:00:00	2.099862e+05	0.000000	-2.060000	2.472000	126.064000	3.879000	1.000000	1.000000	2010.000000
25%	12.000000	2010-10-08 00:00:00	5.533501e+05	0.000000	47.460000	2.933000	131.735000	6.891000	8.000000	4.000000	2010.000000
50%	23.000000	2011-06-17 00:00:00	9.607460e+05	0.000000	62.670000	3.445000	182.616521	7.874000	16.000000	6.000000	2011.000000
75%	34.000000	2012-02-24 00:00:00	1.420159e+06	0.000000	74.940000	3.735000	212.743293	8.622000	23.000000	9.000000	2012.000000
max	45.000000	2012-10-26 00:00:00	3.818686e+06	1.000000	100.140000	4.468000	227.232807	14.313000	31.000000	12.000000	2012.000000
std	12.988182	NaN	5.643666e+05	0.255049	18.444933	0.459020	39.356712	1.875885	8.755780	3.238308	0.797019

Adicionalmente se analiza las variables categóricas:

Categóricas

```
df.describe(include='object')

ValueError                                Traceback (most recent call last)
<ipython-input-93-e9c15d751cf5> in <cell line: 1>()
----> 1 df.describe(include='object')

4 frames
/usr/local/lib/python3.10/dist-packages/pandas/core/reshape/concat.py in __init__(self, objs, axis, join, keys, levels, names, ignore_index, verify_integrity, copy, sort)
427
428     if len(objs) == 0:
--> 429         raise ValueError("No objects to concatenate")
430
431     if keys is None:

ValueError: No objects to concatenate
```

Como se puede observar no poseemos en la base de datos variables categóricas, esto específicamente debido a que se hizo un tratamiento a la fecha y adicionalmente no se posee ninguna variable adicional que tenga letras o caracteres en su conformación, por lo cual descartamos la idea de tener esta variable. Se podría generar una variable categórica si se desea, aun así, por fines de experimento seguimos trabajando con estos datos sin transformar nada hasta llegar al final de la regresión logística.

3. Evalúe si la base de datos contiene datos perdidos

A continuación, se evalúa los datos perdidos en base al siguiente código:

```
df.isnull().sum()

Tienda      0
Fecha       0
Ventas      0
Festivo     0
Temperatura 0
Combustible 0
CPI         0
Desempleo   0
Dia         0
Mes         0
Año         0
dtype: int64
```

Como se puede observar no se posee datos perdidos en la base de datos

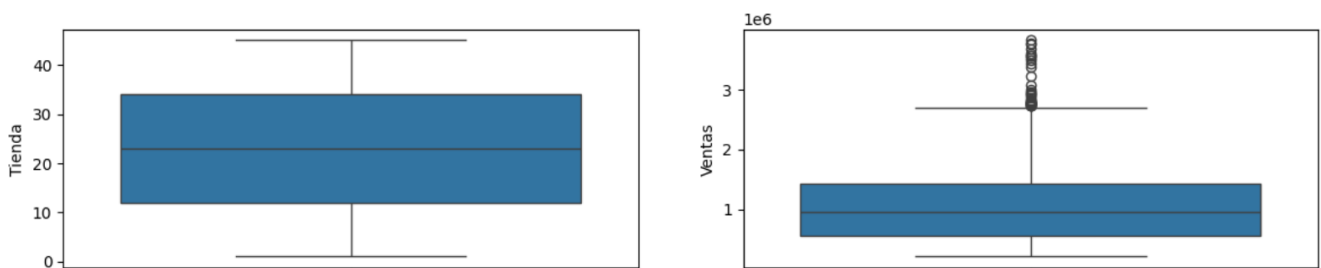
4. Evalúe si alguna de las variables contiene datos atípicos (outliners)

a. De ser el caso, detalle cuáles y qué método estadístico aplicará para corregir

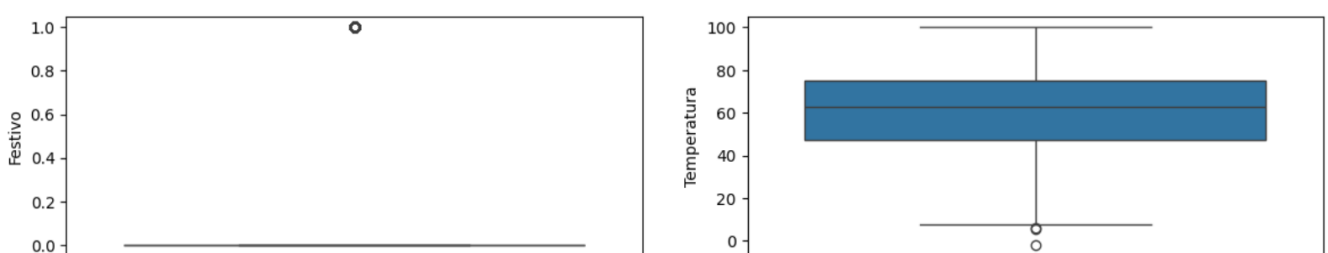
A continuación, se genera la evaluación respectiva de los datos atípicos, por lo cual se usa el siguiente código para visualizar de mejor manera las dispersiones por medio de una gráfica Bix Plot:

```
fig, axs = plt.subplots(5,2, figsize = (12,12))
plt1 = sns.boxplot(df['Tienda'], ax = axs[0,0])
plt2 = sns.boxplot(df['Ventas'], ax = axs[0,1])
plt1 = sns.boxplot(df['Festivo'], ax = axs[1,0])
plt2 = sns.boxplot(df['Temperatura'], ax = axs[1,1])
plt1 = sns.boxplot(df['Combustible'], ax = axs[2,0])
plt2 = sns.boxplot(df['CPI'], ax = axs[2,1])
plt1 = sns.boxplot(df['Desempleo'], ax = axs[3,0])
plt2 = sns.boxplot(df['Mes'], ax = axs[3,1])
plt1 = sns.boxplot(df['Dia'], ax = axs[4,0])
plt2 = sns.boxplot(df['Año'], ax = axs[4,1])
plt.tight_layout()
```

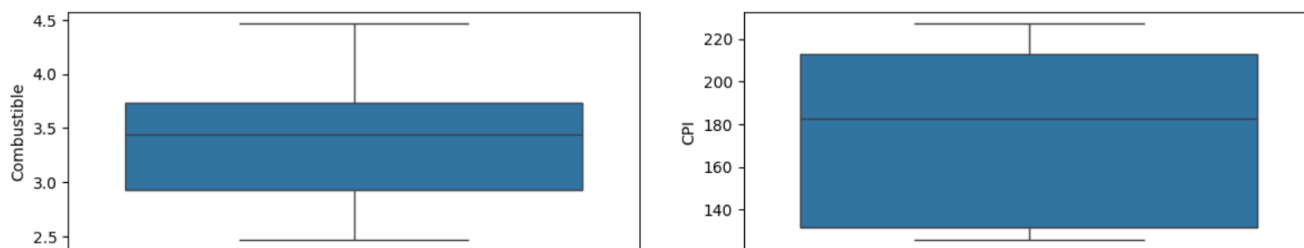
Con lo cual se genera los siguientes resultados:



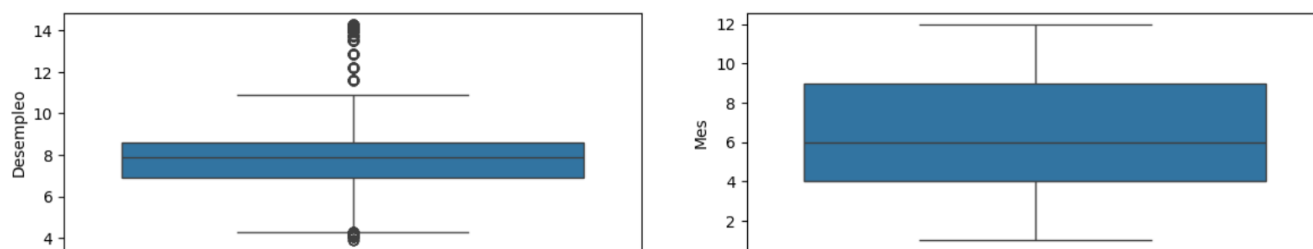
Como se observa la gráfica correspondiente a Tienda no cuenta con datos atípicos, mientras que, ventas si cuenta con datos por fuera de los límites superiores del box plot, estos deben ser tratados para normalizarlos y futuramente poder trabajar con ellos.



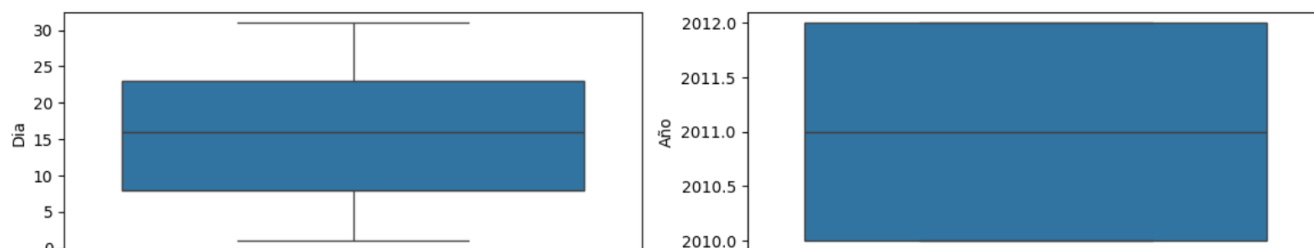
En el caso de la variable festivos no se muestra una gráfica realmente ya que festivo cuenta con dos opciones siendo estas 1 (Festivo) y 0 (No Festivo), por lo cual el Box Plot no sería una gráfica adecuada de análisis. En el caso contrario, la variable temperatura se puede observar que relativamente tiene valores muy normales y tiene unos dos puntos atípicos que podrían no ser de mucha preocupación para el estudio.



La variable combustible como se observa no cuenta con datos atípicos, así como, la variable CPI tampoco cuenta con datos atípicos.



La variable Desempleo como se observa en la gráfica si cuenta con valores atípicos dado que tiene valores por fuera de los límites superior e inferior del Box Plot. En el caso de la variable mes no se cuenta con datos atípicos.



Finalmente, las variables día y Año no poseen valores atípicos, De este modo se debería tratar a las dos variables que más incidencia de valores atípicos poseen, siendo estas Desempleo y Ventas. Para efectuar esta corrección se aplica en el código la eliminación de outliers:

```
[ ] cuartiles = df.select_dtypes(include=['number']).quantile([0.25, 0.5, 0.75])
print(cuartiles)
```

	Tienda	Ventas	Festivo	Temperatura	Combustible	CPI \
0.25	12.0	553350.105	0.0	47.46	2.933	131.735000
0.50	23.0	960746.040	0.0	62.67	3.445	182.616521
0.75	34.0	1420158.660	0.0	74.94	3.735	212.743293

	Desempleo	Dia	Mes	Año
0.25	6.891	8.0	4.0	2010.0
0.50	7.874	16.0	6.0	2011.0
0.75	8.622	23.0	9.0	2012.0

En primer lugar se debe identificar los outliers para poder eliminarlos posteriormente:

```

# Identificar outliers
def identify_outliers(df):
    outliers = {}
    for column in df.select_dtypes(include=[np.number]).columns:
        Q1 = df[column].quantile(0.25)
        Q3 = df[column].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        outliers[column] = df[(df[column] < lower_bound) | (df[column] > upper_bound)][column]
    return outliers

```

Paso seguido, se busca la eliminación de los mismos:

```

# Eliminar outliers
def remove_outliers(df):
    for column in df.select_dtypes(include=[np.number]).columns:
        Q1 = df[column].quantile(0.25)
        Q3 = df[column].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        df = df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]
    return df

df_cleaned = df.copy()

# Eliminar outliers iterativamente hasta que no queden más
max_iterations = 10
iteration = 0

while iteration < max_iterations:
    iteration += 1
    df_cleaned = remove_outliers(df_cleaned)
    outliers_after_cleaning = identify_outliers(df_cleaned)

```

De igual manera, se valida la eliminación de los outliers:

```

# Verificar si quedan outliers
if all(len(values) == 0 for values in outliers_after_cleaning.values()):
    break

for column, values in outliers_after_cleaning.items():
    print(f'Outliers for {column}: {len(values)}')

```

De este modo se obtiene el siguiente resultado en el cual validamos que no existan valores atípicos y que se hayan tratado:

```

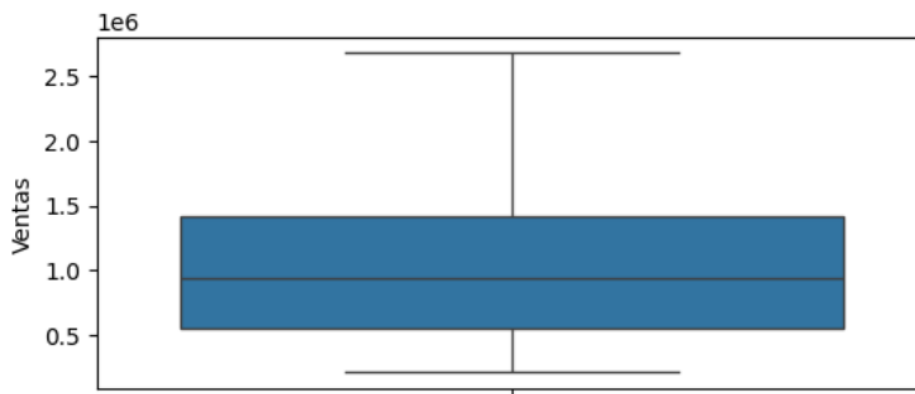
➡ Outliers for Tienda: 0
  Outliers for Ventas: 0
  Outliers for Festivo: 0
  Outliers for Temperatura: 0
  Outliers for Combustible: 0
  Outliers for CPI: 0
  Outliers for Desempleo: 0
  Outliers for Dia: 0
  Outliers for Mes: 0
  Outliers for Año: 0

```

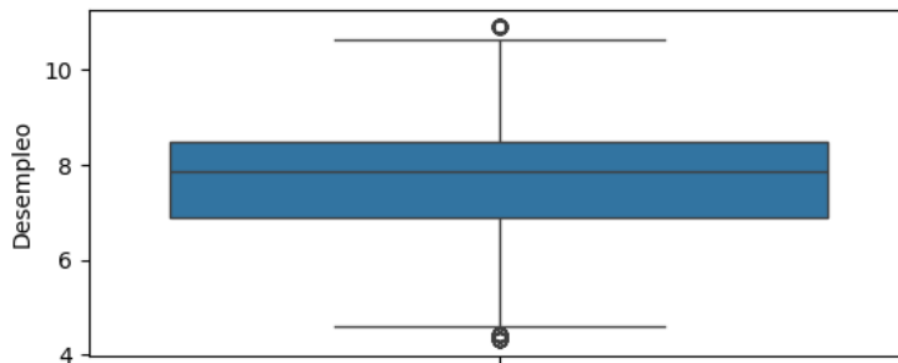
Adicionalmente, para validar nuevamente se corre un código que muestre los cambios generados por la eliminación de los valores que tienen datos atípicos:

```
# Crear la gráfica de los datos sin outliers
fig, axs = plt.subplots(5,2, figsize = (12,12))
plt1 = sns.boxplot(df_sin_outliers['Tienda'], ax = axs[0,0])
plt2 = sns.boxplot(df_sin_outliers['Ventas'], ax = axs[0,1])
plt1 = sns.boxplot(df_sin_outliers['Festivo'], ax = axs[1,0])
plt2 = sns.boxplot(df_sin_outliers['Temperatura'], ax = axs[1,1])
plt1 = sns.boxplot(df_sin_outliers['Combustible'], ax = axs[2,0])
plt2 = sns.boxplot(df_sin_outliers['CPI'], ax = axs[2,1])
plt1 = sns.boxplot(df_sin_outliers['Desempleo'], ax = axs[3,0])
plt2 = sns.boxplot(df_sin_outliers['Mes'], ax = axs[3,1])
plt1 = sns.boxplot(df_sin_outliers['Dia'], ax = axs[4,0])
plt2 = sns.boxplot(df_sin_outliers['Año'], ax = axs[4,1])
plt.tight_layout()
plt.title('Datos sin Outliers')
plt.show()
```

Siguiendo este código se muestra principalmente las gráficas de las variables que tenían que ser tratadas:



Como se puede observar ya no se poseen valores atípicos en la variable Ventas.

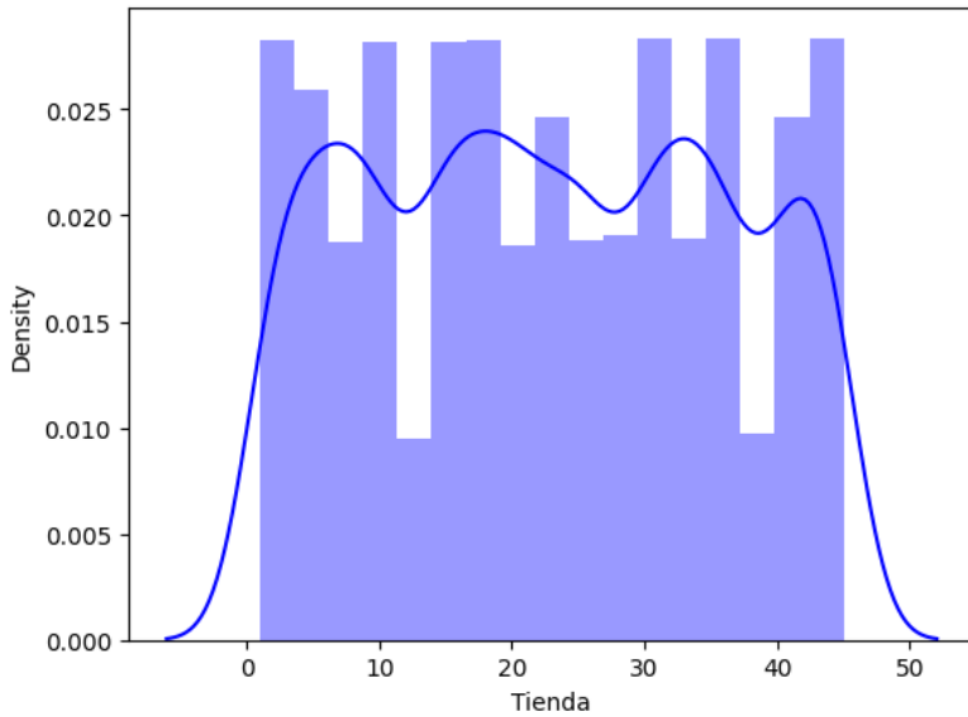


La variable desempeño también tiene una mejor conformación después del tratamiento dado.

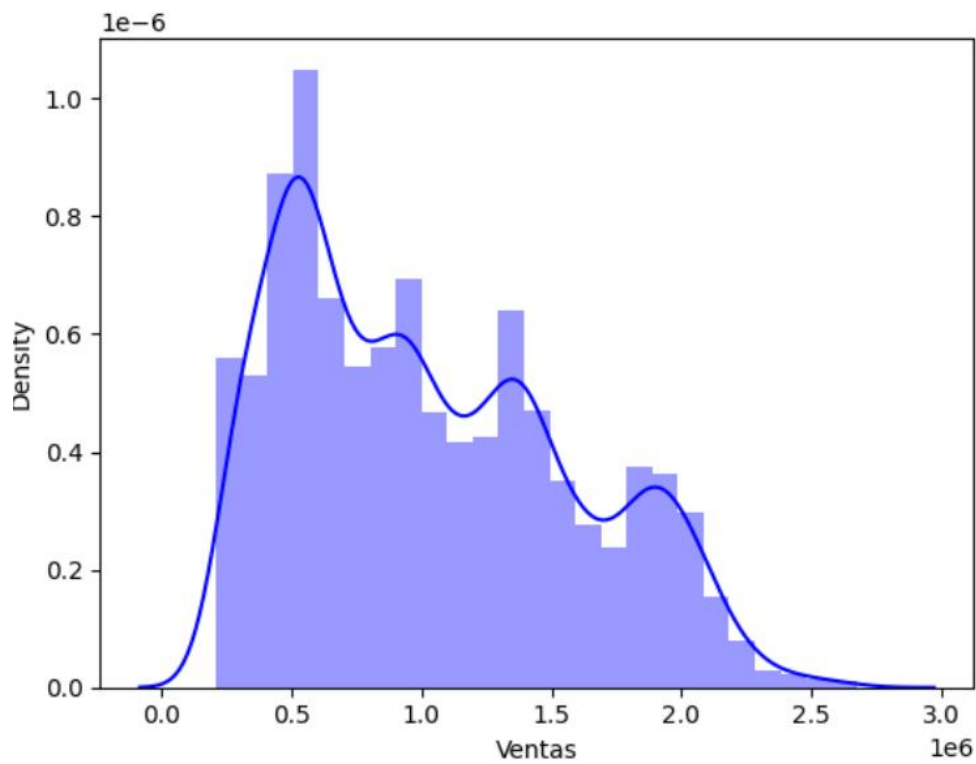
5. Grafique las distribuciones de las variables y a priori comente sobre ellas

Se genera las distribuciones de las variables numéricas haciendo uso del siguiente código:

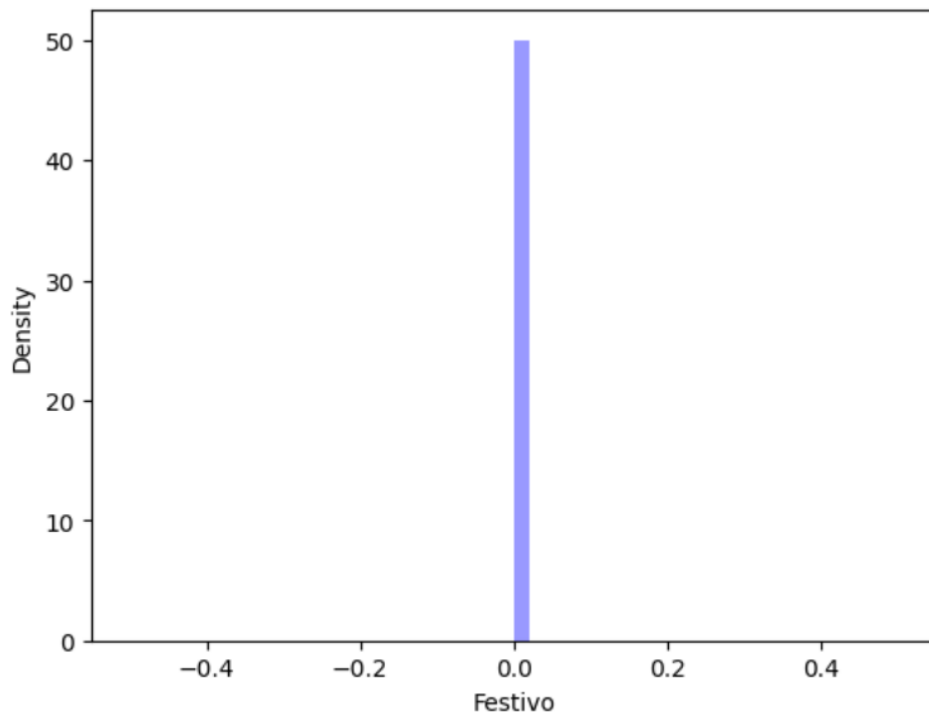
```
[ ] numeric_cols = df_cleaned.select_dtypes(include=['number']).columns
for col in numeric_cols:
    plt.figure()
    sns.distplot(df_cleaned[col], kde=True, color="blue")
    plt.show()
```

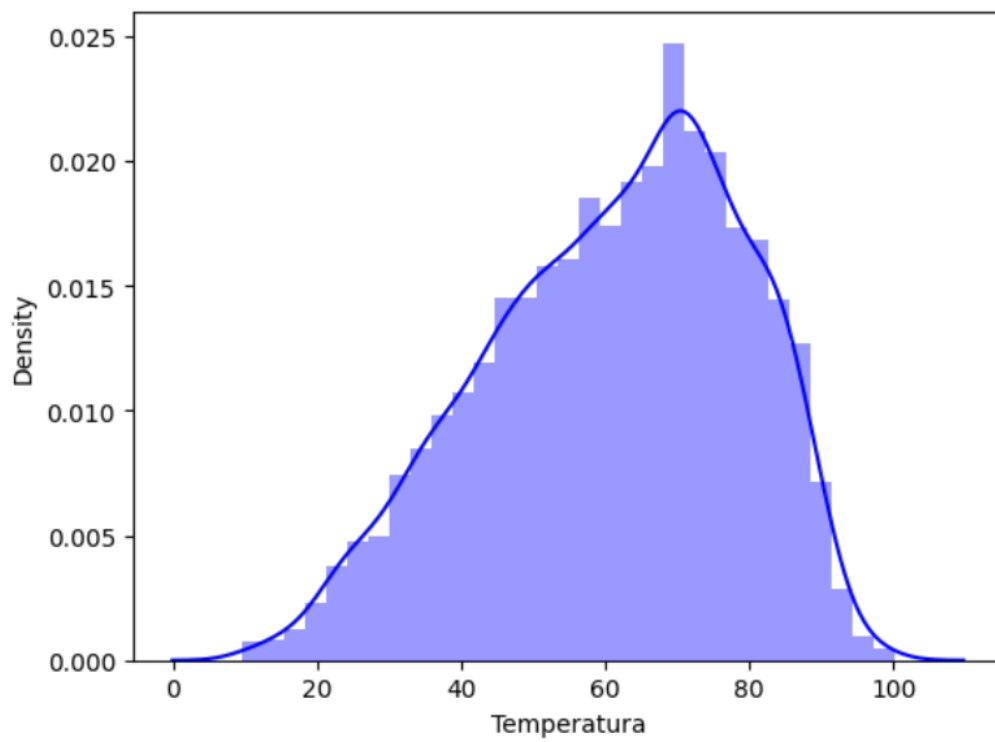
La primera variable a analizar es Tienda, se puede observar que la distribución de la misma tiende al centro como se observa en la campana de Gauss.



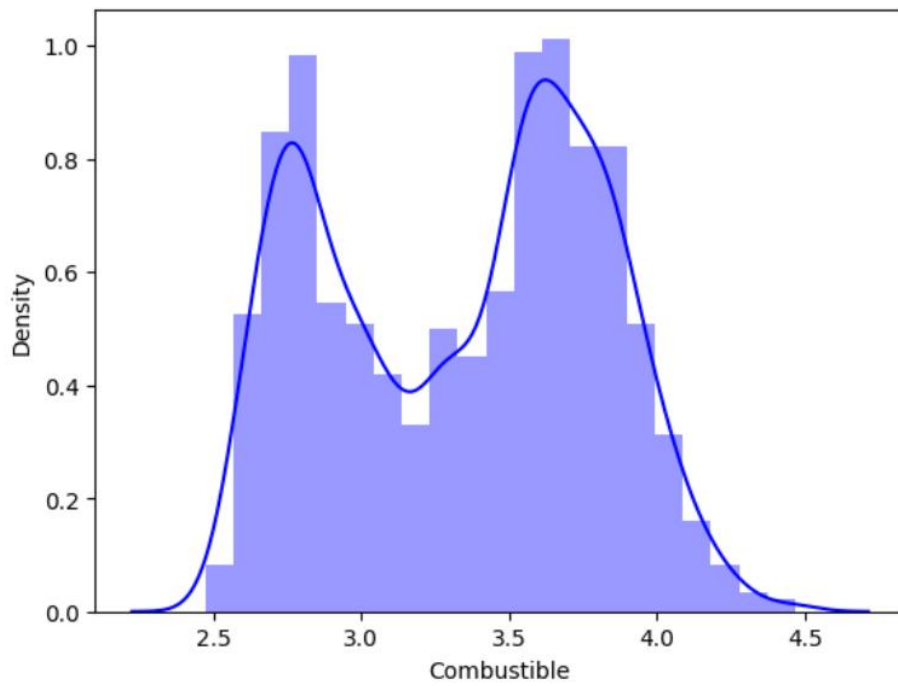
La variable Ventas se puede observar que tiene una distribución oscilante que tiende su cola al lado izquierdo de la campana de Gauss.



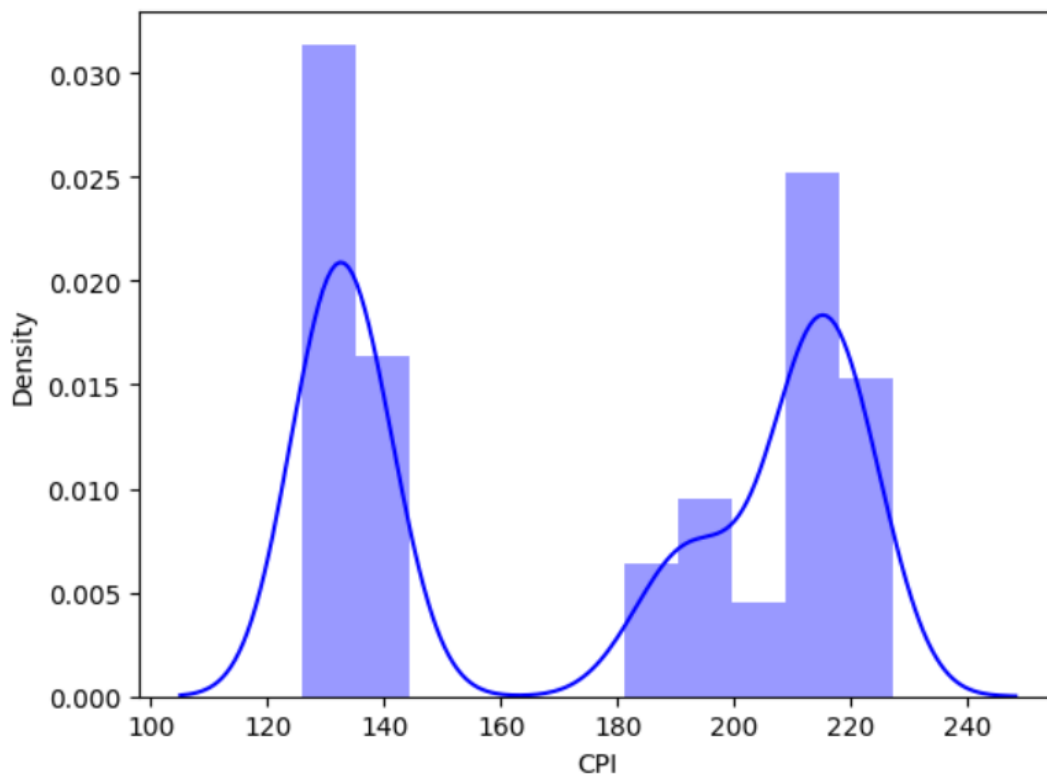
Se puede observar que la variable Festivo está conformada por 1 solo valor.



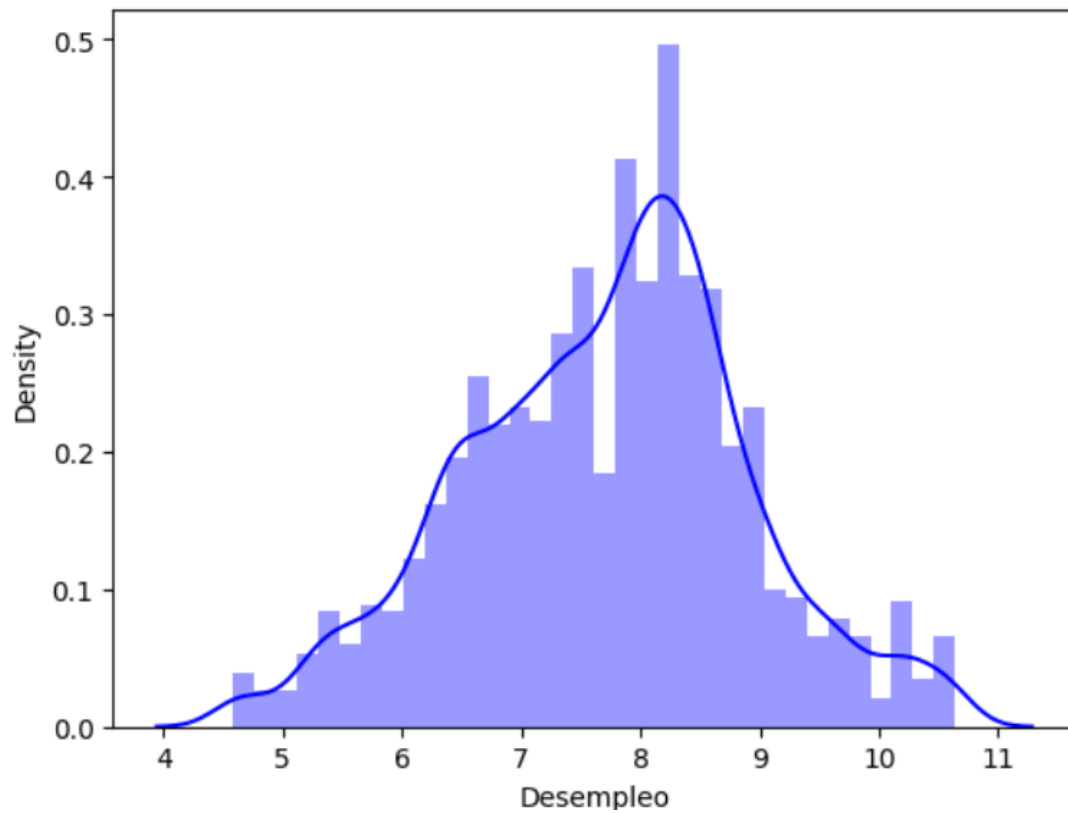
La variable Temperatura tiene una distribución que tiende al centro derecho de la gráfica de la campana de Gauss.



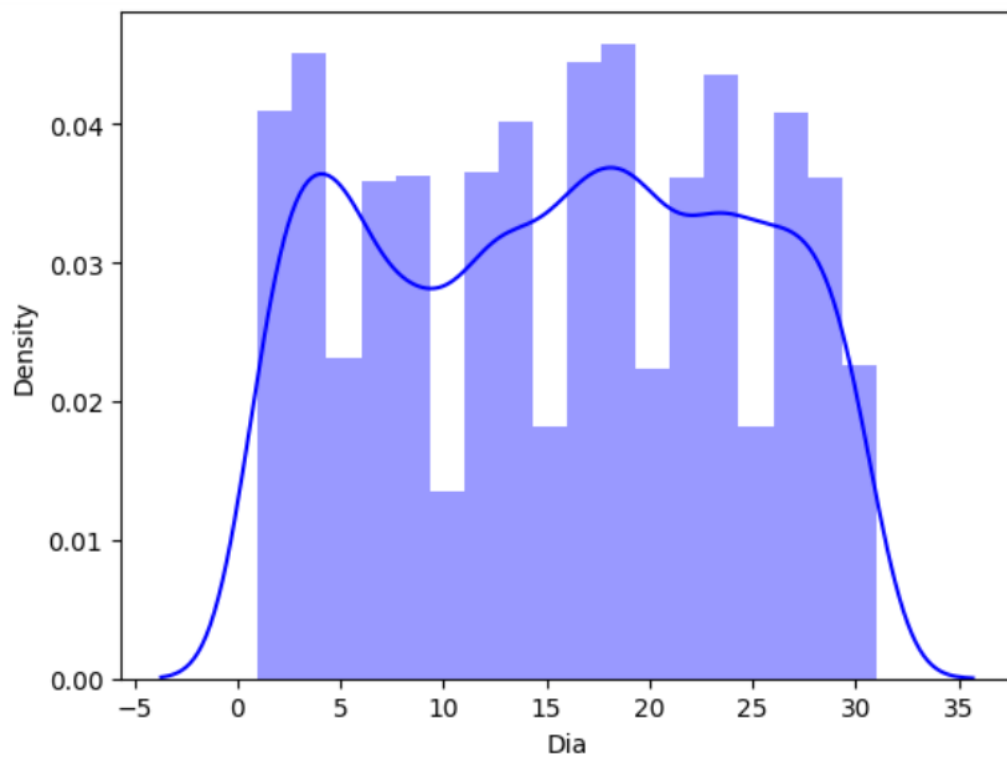
La variable Combustible como se puede observar posee una distribución de dos colas centrales en su composición de campana de Gauss.



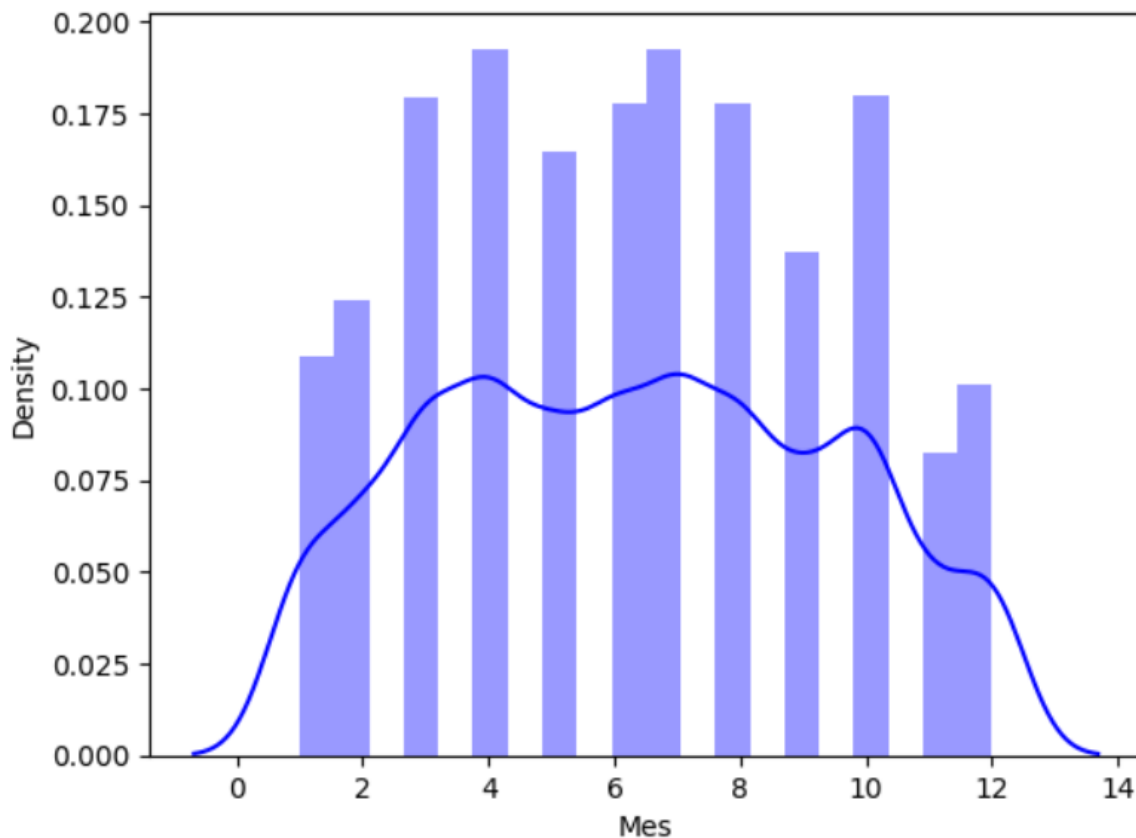
La variable CPI muestra una distribución de binomial que muestra dos colas en su distribución de campana de Gauss.



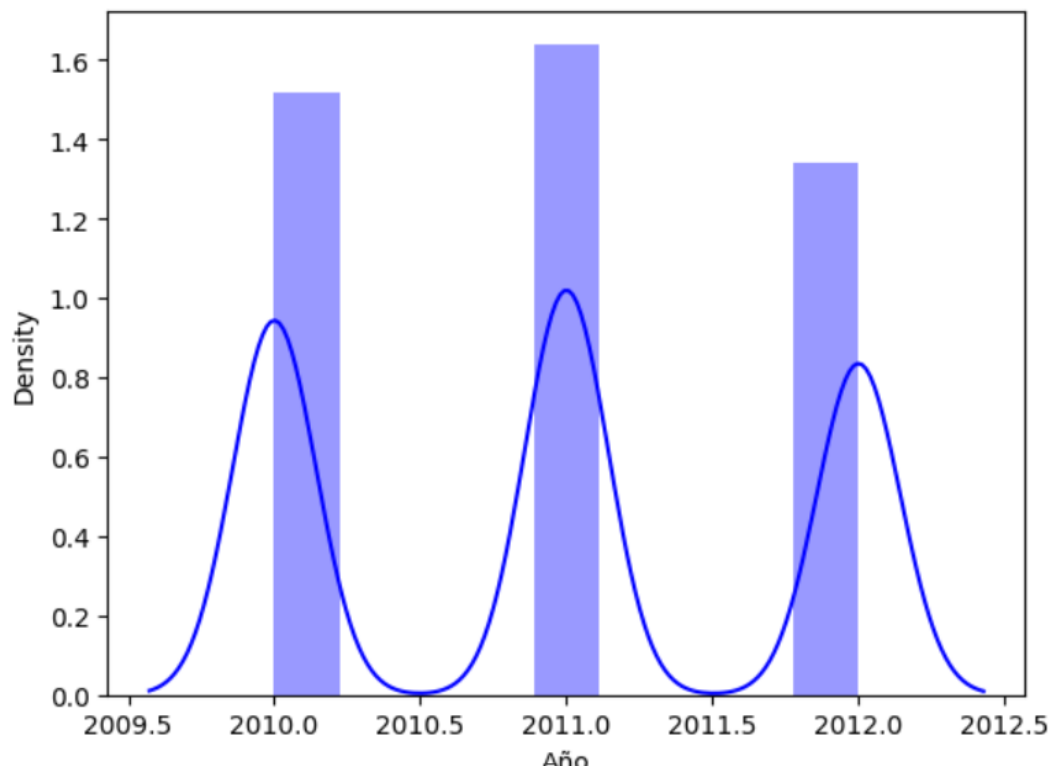
La variable desempeño se puede observar que tiende su distribución hacia el centro de los datos analizados.



La variable Día se observa que su distribución tiende hacia el centro de los datos evaluados.



La variable Mes como se observa tienen una tendencia más amplia con respecto a su distribución, aun así se va concentrando en los distintos meses dependiendo de las ventas en general.



La variable año cuenta con una distribución pronunciada que se visualiza en tres puntas distintas a lo largo de la misma, es trinomial.

Adicionalmente, no se posee gráficas resultantes del análisis de variables categóricas al no contar con una:

Frecuencia de variables categóricas

```
[ ] categorical_cols = df_cleaned.select_dtypes(include=['object']).columns
for col in categorical_cols:
    plt.figure()
    df_cleaned[col].value_counts().plot(kind='bar')
    plt.xlabel(col)
    plt.ylabel('Frecuencia')
    plt.show()
```

6. Obtenga las correlaciones entre los datos de corte numérico

A continuación, se genera el código que permite determinar la correlación de las variables:

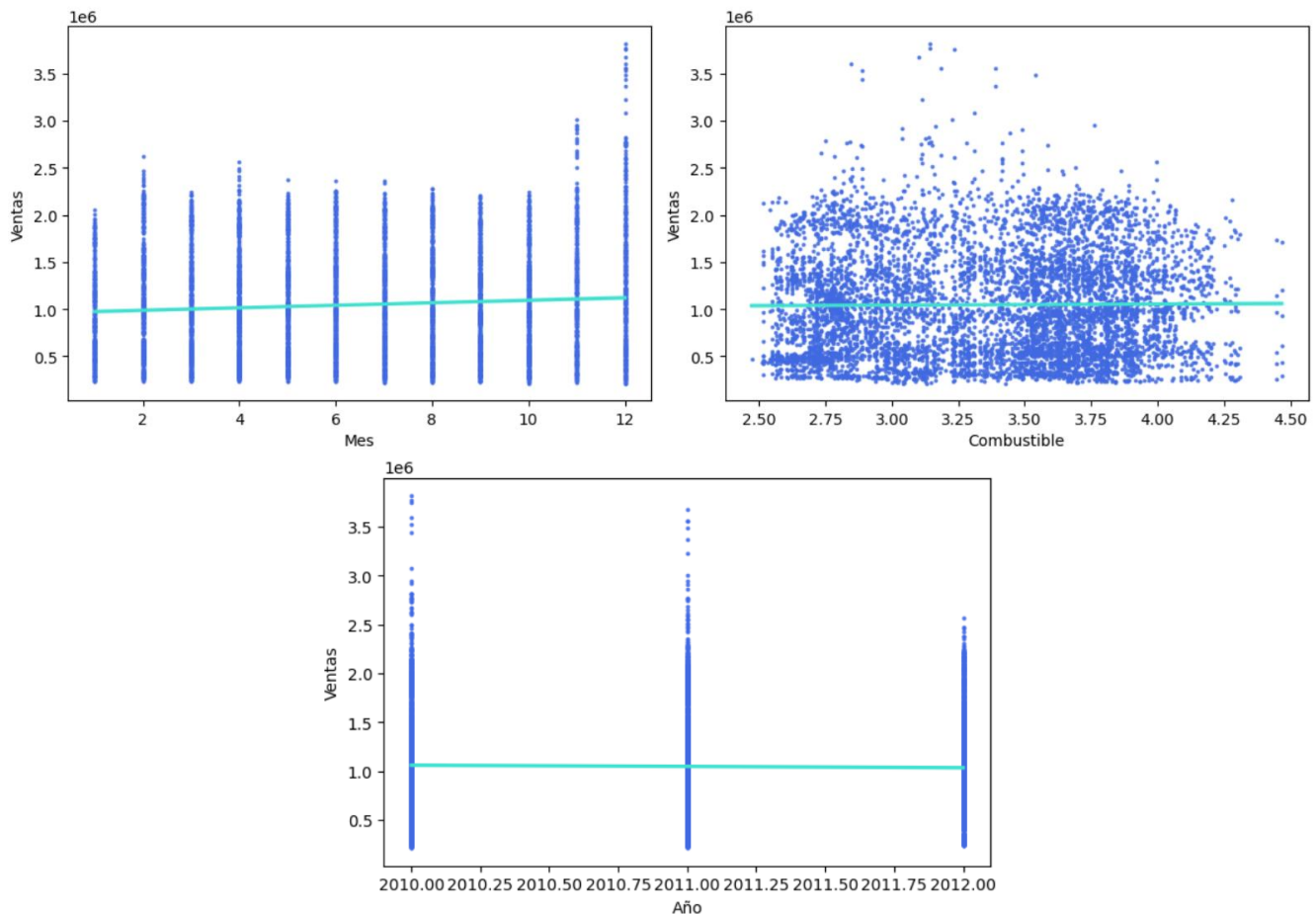
```
numerical_df = df_cleaned.select_dtypes(include=['number'])
numerical_df.corr()
```

	Tienda	Ventas	Festivo	Temperatura	Combustible	CPI	Desempleo	Dia	Mes	Año
Tienda	1.000000	-0.316222	NaN	-0.025456	0.046444	-0.209478	0.321092	0.002084	0.006140	-0.004571
Ventas	-0.316222	1.000000	NaN	-0.037851	0.018721	-0.079723	-0.051682	-0.037204	0.043021	-0.026434
Festivo	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Temperatura	-0.025456	-0.037851	NaN	1.000000	0.114122	0.235501	-0.006634	0.073314	0.272601	0.053267
Combustible	0.046444	0.018721	NaN	0.114122	1.000000	-0.131896	-0.123973	0.050815	-0.039182	0.780313
CPI	-0.209478	-0.079723	NaN	0.235501	-0.131896	1.000000	-0.231608	0.006222	0.008309	0.102760
Desempleo	0.321092	-0.051682	NaN	-0.006634	-0.123973	-0.231608	1.000000	-0.009263	-0.005210	-0.271142
Dia	0.002084	-0.037204	NaN	0.073314	0.050815	0.006222	-0.009263	1.000000	-0.062178	0.040727
Mes	0.006140	0.043021	NaN	0.272601	-0.039182	0.008309	-0.005210	-0.062178	1.000000	-0.178965
Año	-0.004571	-0.026434	NaN	0.053267	0.780313	0.102760	-0.271142	0.040727	-0.178965	1.000000

```
numerical_df.corr().style.background_gradient(cmap='coolwarm')
```

	Tienda	Ventas	Festivo	Temperatura	Combustible	CPI	Desempleo	Dia	Mes	Año
Tienda	1.000000	-0.316222	nan	-0.025456	0.046444	-0.209478	0.321092	0.002084	0.006140	-0.004571
Ventas	-0.316222	1.000000	nan	-0.037851	0.018721	-0.079723	-0.051682	-0.037204	0.043021	-0.026434
Festivo	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
Temperatura	-0.025456	-0.037851	nan	1.000000	0.114122	0.235501	-0.006634	0.073314	0.272601	0.053267
Combustible	0.046444	0.018721	nan	0.114122	1.000000	-0.131896	-0.123973	0.050815	-0.039182	0.780313
CPI	-0.209478	-0.079723	nan	0.235501	-0.131896	1.000000	-0.231608	0.006222	0.008309	0.102760
Desempleo	0.321092	-0.051682	nan	-0.006634	-0.123973	-0.231608	1.000000	-0.009263	-0.005210	-0.271142
Dia	0.002084	-0.037204	nan	0.073314	0.050815	0.006222	-0.009263	1.000000	-0.062178	0.040727
Mes	0.006140	0.043021	nan	0.272601	-0.039182	0.008309	-0.005210	-0.062178	1.000000	-0.178965
Año	-0.004571	-0.026434	nan	0.053267	0.780313	0.102760	-0.271142	0.040727	-0.178965	1.000000

Se genera igualmente una gradiente de colores con la finalidad de observar mejor las correlaciones, se puede determinar que existe una correlación positiva con la variable Tienda. Adicionalmente, se genera las gráficas de correlación para visualizar de mejor manera:



7. Comente qué variable escogerán como variable dependiente y a que variables introducirán a su modelo

De manera general, se cuenta con variables numéricas, aun así, se llegó al final del ejercicio tratándolo como si se tratase de una regresión logística por fines educativos y de experimentación con el código. Se genera LabelEncoder y OneHotEncoder. Hay que tomar en cuenta que la variable objetivo es Ventas, las demás se consideran como variables independientes.

```
[ ] from sklearn.preprocessing import LabelEncoder
```

```
[ ] var_cuantitativas = df_cleaned.select_dtypes('number').columns
var_cualitativas = df_cleaned.select_dtypes('object').columns
```

```
df[var_cualitativas]=df[var_cualitativas].apply(LabelEncoder().fit_transform)
df
```

	Tienda	Fecha	Ventas	Festivo	Temperatura	Combustible	CPI	Desempleo	Dia	Mes	Año
0	1	2010-02-05	1643690.90	0	42.31	2.572	211.096358	8.106	5	2	2010
1	1	2010-02-12	1641957.44	1	38.51	2.548	211.242170	8.106	12	2	2010
2	1	2010-02-19	1611968.17	0	39.93	2.514	211.289143	8.106	19	2	2010
3	1	2010-02-26	1409727.59	0	46.63	2.561	211.319643	8.106	26	2	2010
4	1	2010-03-05	1554806.68	0	46.50	2.625	211.350143	8.106	5	3	2010
...
6430	45	2012-09-28	713173.95	0	64.88	3.997	192.013558	8.684	28	9	2012
6431	45	2012-10-05	733455.07	0	64.89	3.985	192.170412	8.667	5	10	2012

Se generó el primer modelado con LabelEncoder, haciendo uso de todas las variables haciendo uso del siguiente código:

```

# Excluyendo variables correlacionadas
X = df[['Tienda', 'Ventas', 'Temperatura', 'Combustible', 'CPI',
        'Desempleo', 'Festivo', 'Dia', 'Mes', 'Año']]

# Normalizing 'Ventas' to be within the range [0,1]
y = df['Ventas'] / df['Ventas'].max() # Assuming 'Ventas' are non-negative

logit_model = sm.Logit(y, X)
result = logit_model.fit()

print(result.summary())

```

Del cual se obtuvo el siguiente resultado:

Optimization terminated successfully.

Current function value: 0.434982

Iterations 5

Logit Regression Results

```

=====
Dep. Variable:          Ventas    No. Observations:          6435
Model:                  Logit     Df Residuals:              6425
Method:                  MLE      Df Model:                  9
Date:                   Tue, 06 Aug 2024    Pseudo R-squ.:          -1.213
Time:                   00:32:24    Log-Likelihood:         -2799.1
converged:              True      LL-Null:                 -1264.9
Covariance Type:        nonrobust    LLR p-value:            1.000
=====

```

	coef	std err	z	P> z	[0.025	0.975]
Tienda	0.0012	0.003	0.452	0.651	-0.004	0.006
Ventas	1.358e-06	5.93e-08	22.888	0.000	1.24e-06	1.47e-06
Temperatura	-0.0008	0.002	-0.456	0.649	-0.004	0.003
Combustible	0.0124	0.068	0.183	0.855	-0.120	0.145
CPI	3.219e-05	0.001	0.038	0.969	-0.002	0.002
Desempleo	0.0052	0.017	0.299	0.765	-0.029	0.039
Festivo	-0.0213	0.119	-0.179	0.858	-0.255	0.212
Dia	0.0002	0.003	0.054	0.957	-0.006	0.007
Mes	-6.244e-05	0.010	-0.006	0.995	-0.019	0.019
Año	-0.0013	0.000	-6.989	0.000	-0.002	-0.001

```

=====

```

Como se observa, se obtuvo un R2 negativo lo cual indica que el modelo no tiene un buen ajuste con respecto al promedio, incluso se puede interpretar este resultado como valor 0. Desde este punto pudimos verificar que el modelo escogido no era muy adecuado, aun así, seguimos evaluando con regresión logística hasta el final para denegar la idea completamente. Aún así, siguiendo la estructura de revisión de p-valor, se eliminó las variables no significativas y se obtuvo lo siguiente:

Optimization terminated successfully.

Current function value: 0.434919

Iterations 5

Logit Regression Results

```

=====
Dep. Variable:          Ventas    No. Observations:          6435
Model:                  Logit     Df Residuals:              6433
Method:                  MLE      Df Model:                  1
Date:                   Tue, 06 Aug 2024    Pseudo R-squ.:          -1.213
Time:                   00:36:29    Log-Likelihood:         -2798.7
converged:              True      LL-Null:                 -1264.9
Covariance Type:        nonrobust    LLR p-value:            1.000
=====

```

	coef	std err	z	P> z	[0.025	0.975]
Ventas	1.345e-06	5.39e-08	24.962	0.000	1.24e-06	1.45e-06
Año	-0.0012	3.53e-05	-35.131	0.000	-0.001	-0.001

```

=====

```


Con lo cual se obtuvo los Odd Ratios:

```
# Calcular los ratios de odds
odds_ratios = pd.DataFrame({
    'Coeficientes': result.params,
    'Ratios de Odds': np.exp(result.params)
})
```

```
# Mostrar los ratios de odds
print(odds_ratios)
```

	Coeficientes	Ratios de Odds
Ventas	0.000001	1.000001
Año	-0.001240	0.998761

Posteriormente se usó, OneHotEncoder para evaluar mediante otro método, haciendo uso del siguiente código:

```
from sklearn.preprocessing import OneHotEncoder
```

```
# Creamos la instancia del objeto OneHotEncoder
onehotencoder = OneHotEncoder()

# Assuming 'y' is a column in your DataFrame and you want to exclude it from one-hot encoding
# Select categorical columns except 'y'
categorical_columns = [col for col in var_cualitativas if col != 'y']
df_categorical = df[categorical_columns].reset_index(drop=True)

# Aplicar OneHotEncoder a las variables categóricas
encoded_data = onehotencoder.fit_transform(df_categorical)
```

```
# Obtener los nombres de las categorías después de la codificación
categories = onehotencoder.categories_

# Generar los nombres de las columnas
column_names = []
for i, (col, categories_array) in enumerate(zip(df_categorical.columns, categories)):
    column_names.extend([f"{col}_{category}" for category in categories_array])

# Convertir el resultado de la codificación en un DataFrame de pandas
df_encoded = pd.DataFrame(encoded_data.toarray(), columns=column_names, index=df.index)

# Agregar los nombres de las columnas
df_encoded.columns = column_names

# Concatenate the encoded data with the original DataFrame, excluding original categorical columns
df_ = pd.concat([df.drop(var_cualitativas.drop("y", errors='ignore').tolist(), axis=1), df_encoded], axis=1)
```

```
df_encoded.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6435 entries, 0 to 6434
Empty DataFrame
```

De modo que se obtiene las siguientes columnas:

```
df_.columns
```

```
Index(['Tienda', 'Fecha', 'Ventas', 'Festivo', 'Temperatura', 'Combustible',
      'CPI', 'Desempleo', 'Dia', 'Mes', 'Año', 'HighSales'],
      dtype='object')
```

8. Identifique que tipo de modelación realizarán y porqué

De esta manera, se eligió seguir evaluando mediante el uso de LabelEncoder, debido a que puede generar un buen ajuste del modelado:

```
import statsmodels.api as sm
import statsmodels.formula.api as smf # Import formula API for easier model specification

# Rename the 'logit' variable to avoid conflicts
my_logit_array = logit

# Use the formula API to fit the logit model
regression = smf.logit("Ventas ~ Tienda+Festivo+Temperatura+Combustible+CPI+Desempleo+Dia+Mes+Año", data=df_)
results = regression.fit()
```

```
Optimization terminated successfully.
Current function value: 0.426398
Iterations 8
```

Obteniendo los siguientes resultados del modelado definitivo:

Logit Regression Results						
=====						
Dep. Variable:	Ventas		No. Observations:	6435		
Model:	Logit		Df Residuals:	6425		
Method:	MLE		Df Model:	9		
Date:	Tue, 06 Aug 2024		Pseudo R-squ.:	-1.912		
Time:	01:00:45		Log-Likelihood:	-2743.9		
converged:	True		LL-Null:	-942.25		
Covariance Type:	nonrobust		LLR p-value:	1.000		
=====						
	coef	std err	z	P> z	[0.025	0.975]

Intercept	117.9997	139.252	0.847	0.397	-154.929	390.929
Tienda	-0.0251	0.002	-10.123	0.000	-0.030	-0.020
Festivo	0.0696	0.118	0.592	0.554	-0.161	0.300
Temperatura	-0.0030	0.002	-1.682	0.093	-0.006	0.000
Combustible	0.1208	0.118	1.025	0.305	-0.110	0.352
CPI	-0.0035	0.001	-3.944	0.000	-0.005	-0.002
Desempleo	-0.0356	0.018	-1.962	0.050	-0.071	-4.43e-05
Dia	-0.0020	0.003	-0.588	0.557	-0.009	0.005
Mes	0.0220	0.010	2.212	0.027	0.003	0.041
Año	-0.0587	0.069	-0.846	0.397	-0.195	0.077
=====						

Como observamos seguimos manteniendo el mismo R2 por lo cual se puede concluir que el modelo no es el adecuado, lo cual se verificará igualmente con el análisis de los supuestos. Adicionalmente se analiza nuevamente los Odd Ratios

```
# Your logistic regression code
regression_2 = smf.logit("Ventas ~ Tienda+CPI+Desempleo+\
Mes", data=df_) # Use smf instead of sms
results_2 = regression_2.fit()
```

```
Optimization terminated successfully.
Current function value: 0.426640
Iterations 6
```

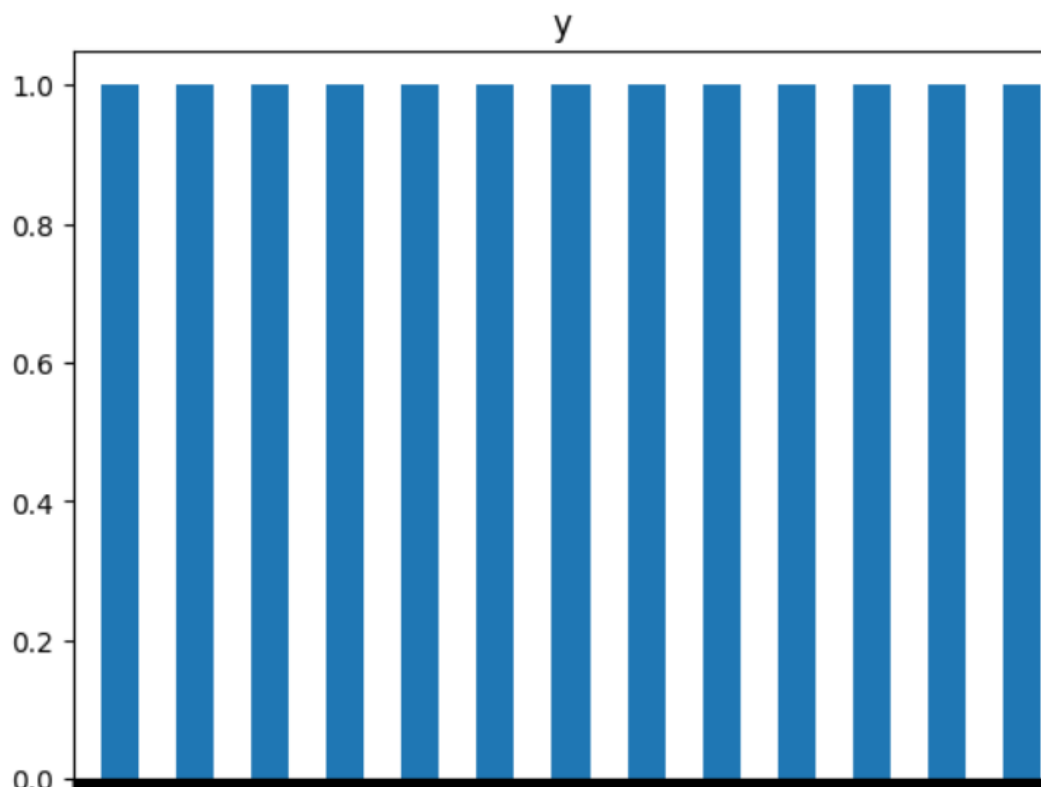
```
coeficientes = results_2.params
odds_ratios = np.exp(coeficientes)
resultados_df = pd.DataFrame({'Coeficiente': coeficientes, 'Odds Ratio': odds_ratios})
print(resultados_df)
```

	Coeficiente	Odds Ratio
Intercept	0.197451	1.218293
Tienda	-0.024948	0.975360
CPI	-0.004048	0.995960
Desempleo	-0.036791	0.963878
Mes	0.021021	1.021244

Como se observa los Odds Ratio no son resultados tan malos en general, aun así no es en realidad el mejor modelo.

9. Verifique los supuestos, de haber escogido el enfoque econométrico

En el primer supuesto se pretende determinar que la variable respuesta es binaria:



Como se observa, no se cuenta con una variable objetivo binaria, por lo cual no cumple con el primer supuesto.

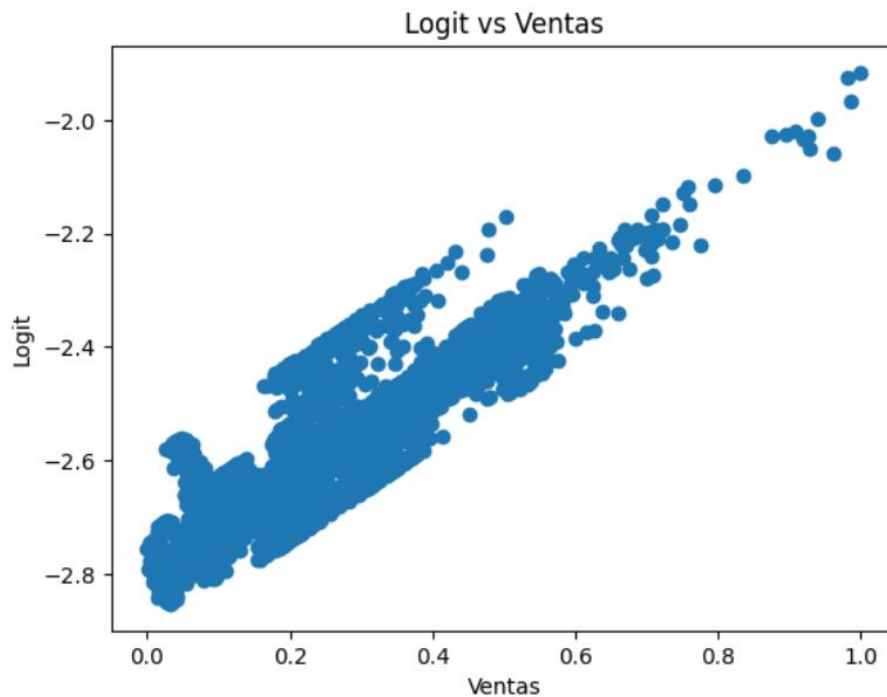
El segundo supuesto es la relación entre odds y variable continua, que se visualiza de este modo:

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LogisticRegression # Example model

# Assuming 'df_' contains the features you want to use for prediction
# Replace 'feature1', 'feature2', and 'target_variable' with the correct column names from df_
X = df_[['Ventas', 'Desempleo']] # Example feature columns, adjust as needed
y = df_['Festivo'] # Example target variable column, adjust as needed
model = LogisticRegression()
model.fit(X, y)

# Now you can use the trained model
y_pred = model.predict_proba(X)[: , 1] # Probabilidades predichas for the positive class
logit = np.log(y_pred / (1 - y_pred)) # Logit

plt.scatter(df_['Ventas'], logit)
plt.xlabel('Ventas')
plt.ylabel('Logit')
plt.title('Logit vs Ventas') # Changed title to reflect the x-axis variable
plt.show()
```



En este gráfico se puede observar que los datos tienen una creciente tendencia positiva, aun así, están entre dispersos y continuos, podría estar cumpliendo con este supuesto.

El tercer supuesto es el de colinealidad perfecta entre variables que se ve de la siguiente manera:

```
[ ] df_.dtypes
```

Tienda	int64
Fecha	datetime64[ns]
Ventas	float64
Festivo	int64
Temperatura	float64
Combustible	float64
CPI	float64
Desempleo	float64
Dia	int32
Mes	int32
Año	int32
HighSales	int64
dtype: object	

10. Obtenga el modelo definitivo, prediga los valores y comente el grado de ajuste del modelo. Justifique con métricas su respuesta

Se generó con el último modelo las predicciones de los valores y los valores VIF, de hecho, en este punto se decidió cambiar de modelo de análisis de la base datos debido a que no obtuvo los mejores resultados. Las respuesta se generaron con el siguiente código:

```

import pandas as pd
from statsmodels.stats.outliers_influence import variance_inflation_factor

# Verifica si el DataFrame está vacío
if df.empty:
    print("Error: El DataFrame está vacío.")
else:
    # Selecciona todas las columnas numéricas como variables predictoras (excepto 'Ventas' si es la variable objetivo)
    var_predictoras = df.select_dtypes(include=['number']).columns.tolist()
    if 'Ventas' in var_predictoras:
        var_predictoras.remove('Ventas')

    if len(var_predictoras) < 2:
        print("Error: Se necesitan al menos dos variables predictoras para calcular el VIF.")
    else:
        # Calcula el VIF para cada variable predictora
        vif = pd.DataFrame()
        vif["Variable"] = var_predictoras
        vif["VIF"] = [variance_inflation_factor(df[var_predictoras].values, i) for i in range(df[var_predictoras].shape[1])]

    # Muestra los resultados
    print(vif)

```

	Variable	VIF
0	Tienda	4.760990
1	Festivo	1.141299
2	Temperatura	14.533803
3	Combustible	60.133870
4	CPI	25.589341
5	Desempleo	22.889805
6	Día	4.225965
7	Mes	5.490918
8	Año	147.357231
9	HighSales	2.172661

Como se observa, los valores VIF son muy altos lo cual indica que existe una gran colinealidad entre las variables, lo cual es otra muestra de que el modelo no es bueno.

Posteriormente, se decidió generar otro modelo con la base de datos haciendo uso de regresión lineal múltiple:

REGRESIÓN LINEAL MULTIPLE

1. Importe la base de datos a una base en Jupyter Notebook con pandas.


Comenzamos a importar nuestra base de datos de Walmart, y se implementa el statsmodel

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

import warnings
warnings.filterwarnings("ignore")

[ ] import statsmodels.stats.api as sms
import statsmodels.api as sm
from statsmodels.formula.api import ols
from statsmodels.compat import lzip

[ ] df = pd.read_csv("/content/Walmart(1).csv")
df.head(10)
```



	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
0	1	05-02-2010	1643690.90	0	42.31	2.572	211.096358	8.106
1	1	12-02-2010	1641957.44	1	38.51	2.548	211.242170	8.106
2	1	19-02-2010	1611968.17	0	39.93	2.514	211.289143	8.106
3	1	26-02-2010	1409727.59	0	46.63	2.561	211.319643	8.106
4	1	05-03-2010	1554806.68	0	46.50	2.625	211.350143	8.106
5	1	12-03-2010	1439541.59	0	57.79	2.667	211.380643	8.106
6	1	19-03-2010	1472515.79	0	54.58	2.720	211.215635	8.106
7	1	26-03-2010	1404429.92	0	51.45	2.732	211.018042	8.106
8	1	02-04-2010	1594968.28	0	62.27	2.719	210.820450	7.808
9	1	09-04-2010	1545418.53	0	65.86	2.770	210.622857	7.808

Para mayor entendimiento, se procede a recodificar los nombres de las variables de inglés a español.

```
df.rename({'Store':'Tienda', 'Date':'Fecha', 'Weekly_Sales':'Ventas', 'Holiday_Flag': 'Festivo',
          'Temperature': 'Temperatura', 'Fuel_Price': 'Combustible',
          'Unemployment':'Desempleo', }, axis=1, inplace=True)
df.head(10)
```

	Tienda	Fecha	Ventas	Festivo	Temperatura	Combustible	CPI	Desempleo
0	1	05-02-2010	1643690.90	0	42.31	2.572	211.096358	8.106
1	1	12-02-2010	1641957.44	1	38.51	2.548	211.242170	8.106
2	1	19-02-2010	1611968.17	0	39.93	2.514	211.289143	8.106
3	1	26-02-2010	1409727.59	0	46.63	2.561	211.319643	8.106
4	1	05-03-2010	1554806.68	0	46.50	2.625	211.350143	8.106
5	1	12-03-2010	1439541.59	0	57.79	2.667	211.380643	8.106
6	1	19-03-2010	1472515.79	0	54.58	2.720	211.215635	8.106
7	1	26-03-2010	1404429.92	0	51.45	2.732	211.018042	8.106
8	1	02-04-2010	1594968.28	0	62.27	2.719	210.820450	7.808
9	1	09-04-2010	1545418.53	0	65.86	2.770	210.622857	7.808

Se revisa que la base de datos cuenta con variable “Fecha”, el cual se encuentra en diferentes formatos y no nos permite realizar un correcto análisis del mismo, procedemos a implementar un código de transformación, para que nos muestre Día, Mes y Año.

```
[ ] df['Fecha'] = pd.to_datetime(df['Fecha'], format='%d-%m-%Y') # Specify the correct date format
```

```
df['Dia'] = df['Fecha'].dt.day
df['Mes'] = df['Fecha'].dt.month
df['Año'] = df['Fecha'].dt.year

df
```

	Tienda	Fecha	Ventas	Festivo	Temperatura	Combustible	CPI	Desempleo	Día	Mes	Año
0	1	2010-02-05	1643690.90	0	42.31	2.572	211.096358	8.106	5	2	2010
1	1	2010-02-12	1641957.44	1	38.51	2.548	211.242170	8.106	12	2	2010
2	1	2010-02-19	1611968.17	0	39.93	2.514	211.289143	8.106	19	2	2010
3	1	2010-02-26	1409727.59	0	46.63	2.561	211.319643	8.106	26	2	2010
4	1	2010-03-05	1554806.68	0	46.50	2.625	211.350143	8.106	5	3	2010
...
6430	45	2012-09-28	713173.95	0	64.88	3.997	192.013558	8.684	28	9	2012
6431	45	2012-10-05	733455.07	0	64.89	3.985	192.170412	8.667	5	10	2012
6432	45	2012-10-12	734464.36	0	54.47	4.000	192.327265	8.667	12	10	2012
6433	45	2012-10-19	718125.53	0	56.47	3.969	192.330854	8.667	19	10	2012
6434	45	2012-10-26	760281.43	0	58.85	3.882	192.308899	8.667	26	10	2012

6435 rows x 11 columns

2. Obtenga los descriptivos resumen de la base de datos e identifique a las variables numéricas y categóricas. ¿Hay algo que le llame la atención?

Se procede a revisar los descriptivos resumen y se identifica lo siguiente:

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6435 entries, 0 to 6434
Data columns (total 11 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   Tienda      6435 non-null   int64
1   Fecha      6435 non-null   datetime64[ns]
2   Ventas      6435 non-null   float64
3   Festivo     6435 non-null   int64
4   Temperatura 6435 non-null   float64
5   Combustible 6435 non-null   float64
6   CPI         6435 non-null   float64
7   Desempleo   6435 non-null   float64
8   Dia         6435 non-null   int32
9   Mes         6435 non-null   int32
10  Año         6435 non-null   int32
dtypes: datetime64[ns](1), float64(5), int32(3), int64(2)
memory usage: 477.7 KB

[ ] NUMERICAS

[ ] df.describe()
```

	Tienda	Fecha	Ventas	Festivo	Temperatura	Combustible	CPI	Desempleo	Dia	Mes	Año
count	6435.000000	6435	6.435000e+03	6435.000000	6435.000000	6435.000000	6435.000000	6435.000000	6435.000000	6435.000000	6435.000000
mean	23.000000	2011-06-17 00:00:00	1.046965e+06	0.069930	60.663782	3.358607	171.578394	7.999151	15.678322	6.447552	2010.965035
min	1.000000	2010-02-05 00:00:00	2.099862e+05	0.000000	-2.060000	2.472000	126.064000	3.879000	1.000000	1.000000	2010.000000
25%	12.000000	2010-10-08 00:00:00	5.533501e+05	0.000000	47.460000	2.933000	131.735000	6.891000	8.000000	4.000000	2010.000000
50%	23.000000	2011-06-17 00:00:00	9.607460e+05	0.000000	62.670000	3.445000	182.616521	7.874000	16.000000	6.000000	2011.000000
75%	34.000000	2012-02-24 00:00:00	1.420159e+06	0.000000	74.940000	3.735000	212.743293	8.622000	23.000000	9.000000	2012.000000
max	45.000000	2012-10-26 00:00:00	3.818686e+06	1.000000	100.140000	4.468000	227.232807	14.313000	31.000000	12.000000	2012.000000
std	12.988182	NaN	5.643666e+05	0.255049	18.444933	0.459020	39.356712	1.875885	8.755780	3.238308	0.797019

Se cuenta con 10 variables, las cuales todas son numéricas y dada la transformación que se realizó a “fecha”, no existen categóricas.

```
[ ] df.describe(include='object')

-----
ValueError                                Traceback (most recent call last)
<ipython-input-10-e9c15d751cf5> in <cell line: 1>()
----> 1 df.describe(include='object')

-----
5 frames -----
/usr/local/lib/python3.10/dist-packages/pandas/core/reshape/concat.py in _clean_keys_and_objs(self, objs, keys)
    503
    504     if len(objs_list) == 0:
-> 505         raise ValueError("No objects to concatenate")
    506
    507     if keys is None:

ValueError: No objects to concatenate
```

3. Evalúe si la base contiene datos perdidos.

No contiene datos perdidos

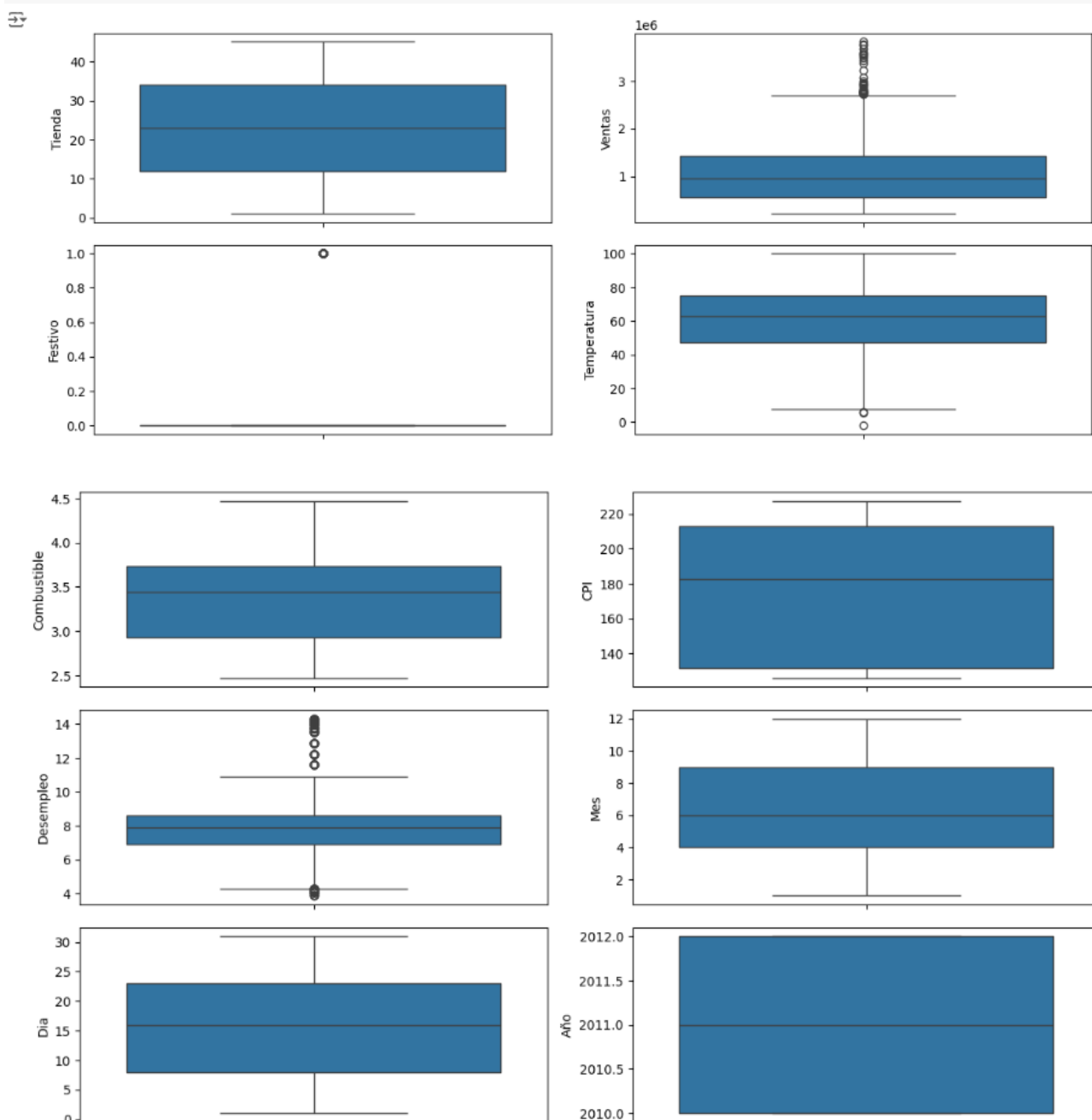
```
df.isnull().sum()

Tienda      0
Fecha      0
Ventas      0
Festivo     0
Temperatura 0
Combustible 0
CPI         0
Desempleo   0
Dia         0
Mes         0
Año         0
dtype: int64
```


4. Evalúe si alguna de las variables contiene datos atípicos (outliers)

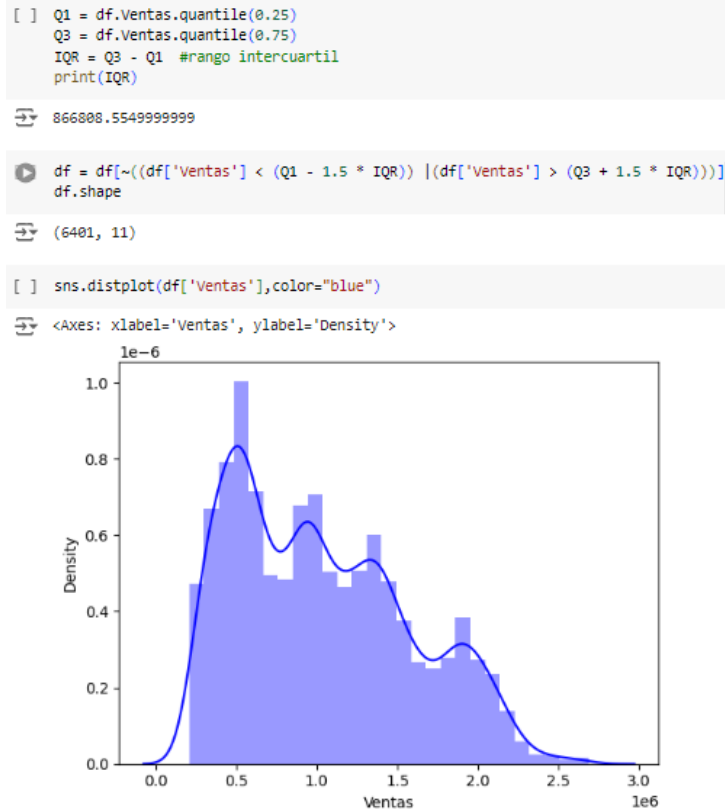
Del boxplot se puede evidenciar que las variables que requieren el tratamiento son Ventas y Desempleo

```
fig, axs = plt.subplots(5,2, figsize = (12,12))
plt1 = sns.boxplot(df['Tienda'], ax = axs[0,0])
plt2 = sns.boxplot(df['Ventas'], ax = axs[0,1])
plt1 = sns.boxplot(df['Festivo'], ax = axs[1,0])
plt2 = sns.boxplot(df['Temperatura'], ax = axs[1,1])
plt1 = sns.boxplot(df['Combustible'], ax = axs[2,0])
plt2 = sns.boxplot(df['CPI'], ax = axs[2,1])
plt1 = sns.boxplot(df['Desempleo'], ax = axs[3,0])
plt2 = sns.boxplot(df['Mes'], ax = axs[3,1])
plt1 = sns.boxplot(df['Dia'], ax = axs[4,0])
plt2 = sns.boxplot(df['Año'], ax = axs[4,1])
plt.tight_layout()
```



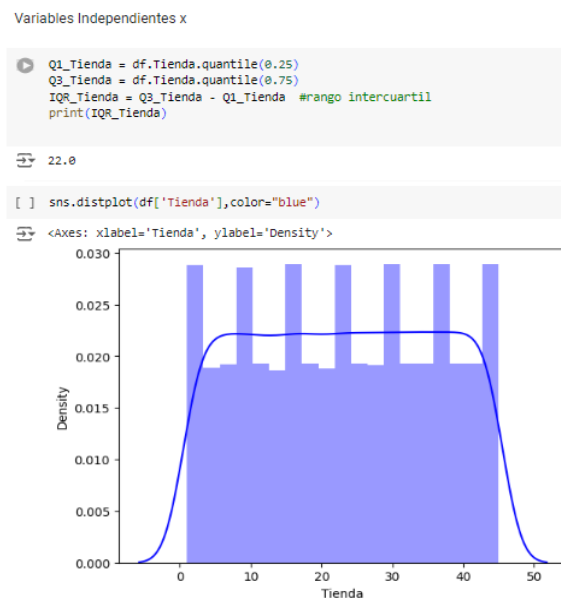
5. Grafique las distribuciones de las variables y a priori comente sobre ellas.

Se procede, primero a graficar la variable dependiente “Ventas”



Se distribuye la variable Ventas en 866808, se visualiza que los datos tienden a la izquierda de la campana de Gauss.

A continuación, variables independientes:



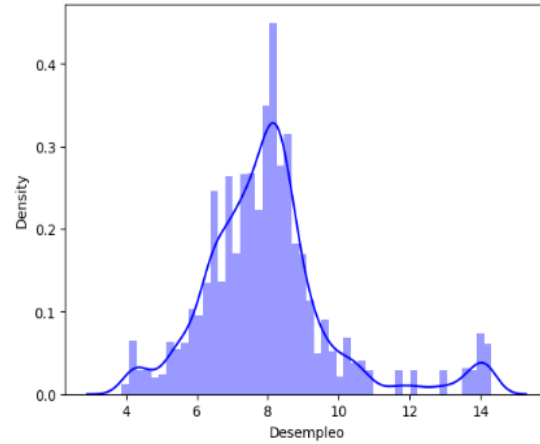
La variable Tienda muestra una distribución de 22, adicionalmente se puede observar que la tendencia de los datos se concentra en el centro.

```
[ ] Q1_Desempleo = df.Desempleo.quantile(0.25)
     Q3_Desempleo = df.Desempleo.quantile(0.75)
     IQR_Desempleo = Q3_Desempleo - Q1_Desempleo #rango intercuartil
     print(IQR_Desempleo)
```

```
1.7309999999999999
```

```
[ ] sns.distplot(df['Desempleo'],color="blue")
```

```
<Axes: xlabel='Desempleo', ylabel='Density'>
```



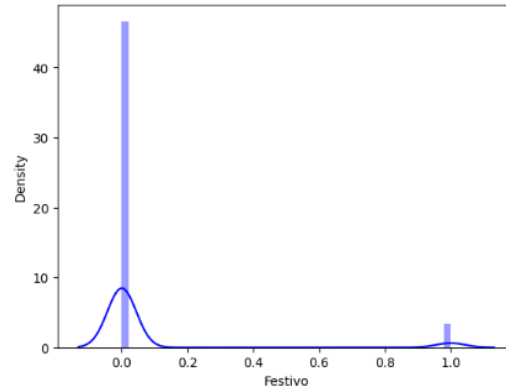
La variable Desempleo muestra una distribución de 1.73, se visualiza que la tendencia es informe

```
[ ] Q1_Festivo = df.Festivo.quantile(0.25)
     Q3_Festivo = df.Festivo.quantile(0.75)
     IQR_Festivo = Q3_Festivo - Q1_Festivo #rango intercuartil
     print(IQR_Festivo)
```

```
-6.891
```

```
sns.distplot(df['Festivo'],color="blue")
```

```
<Axes: xlabel='Festivo', ylabel='Density'>
```



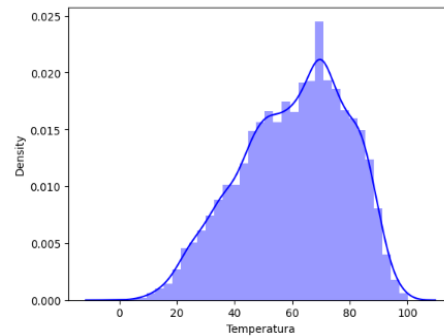
La variable Festivo, muestra una distribución de -6.891, lo cual podría mostrar pérdidas, se observa que es una variable binomial

```
[ ] Q1_Temperatura = df.Temperatura.quantile(0.25)
     Q3_Temperatura = df.Temperatura.quantile(0.75)
     IQR_Temperatura = Q3_Temperatura - Q1_Temperatura #rango intercuartil
     print(IQR_Temperatura)
```

```
-39.038
```

```
[ ] sns.distplot(df['Temperatura'],color="blue")
```

```
<Axes: xlabel='Temperatura', ylabel='Density'>
```



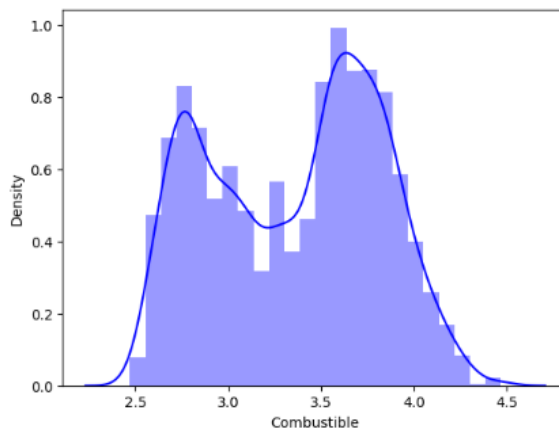
La variable Temperatura muestra una distribución de -39.038, lo cual podría mostrar pérdidas, se puede visualizar que tiene una distribución que tiende a la derecha de los datos.

```
Q1_Combustible = df.Combustible.quantile(0.25)
Q3_Combustible = df.Combustible.quantile(0.75)
IQR_Combustible = Q3_Combustible - Q1_Combustible #rango intercuartil
print(IQR_Combustible)
```

5.689

```
[ ] sns.distplot(df['Combustible'],color="blue")
```

<Axes: xlabel='Combustible', ylabel='Density'>



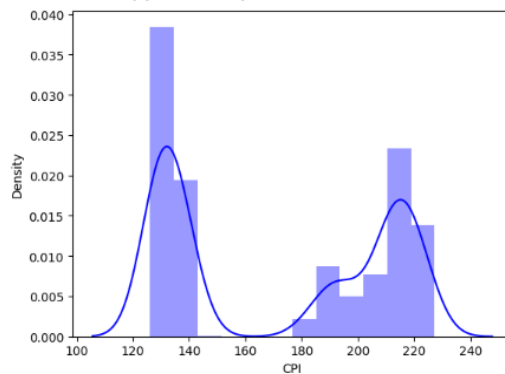
La variable Combustible muestra una distribución de 5.6, se observa que tiene una tendencia que forma dos colas en el centro de los datos.

```
Q1_CPI = df.CPI .quantile(0.25)
Q3_CPI = df.CPI .quantile(0.75)
IQR_CPI = Q3_CPI - Q1_CPI #rango intercuartil
print(IQR_CPI)
```

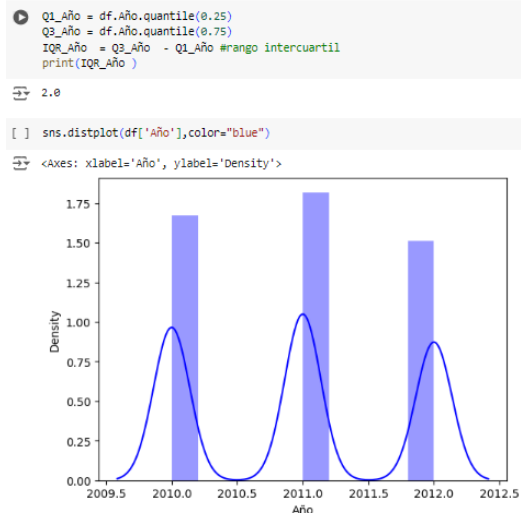
81.04963990000002

```
[ ] sns.distplot(df['CPI'],color="blue")
```

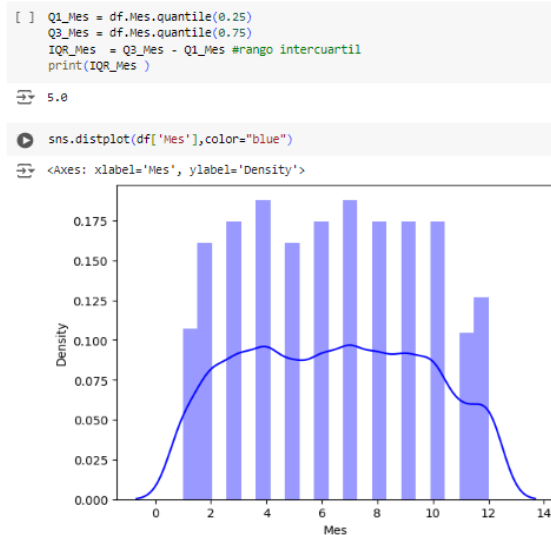
<Axes: xlabel='CPI', ylabel='Density'>



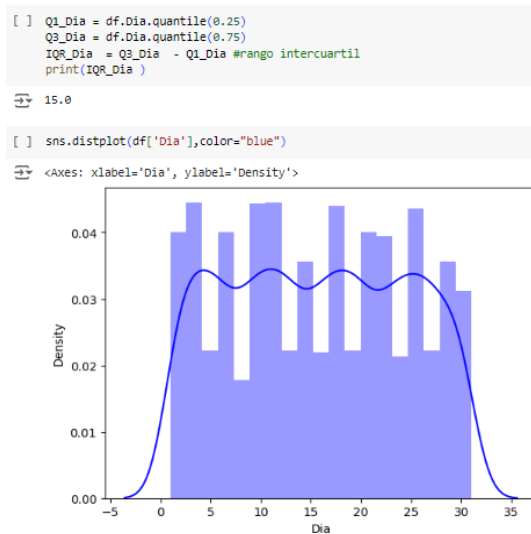
La variable CPI muestra una distribución de 81.04, igualmente forma dos colas en la distribución



La variable Año, muestra una distribución de 2, se observa que tiene la formación de tres campanas.



La variable mes muestra una distribución de 5, se observa que la distribución de los datos se concentra en el centro de los datos.



La variable Día muestra una distribución de 15, se observa que los datos se concentran en el centro de la distribución.

6. Obtenga las correlaciones entre los datos de corte numérico.

Obtenemos la correlación de las variables y procedemos a sacar el gráfico de coolwarm

```
[ ] numerical_df = df.select_dtypes(include=['number'])
numerical_df.corr()
```

```
[ ] numerical_df.corr().style.background_gradient(cmap='coolwarm')
```



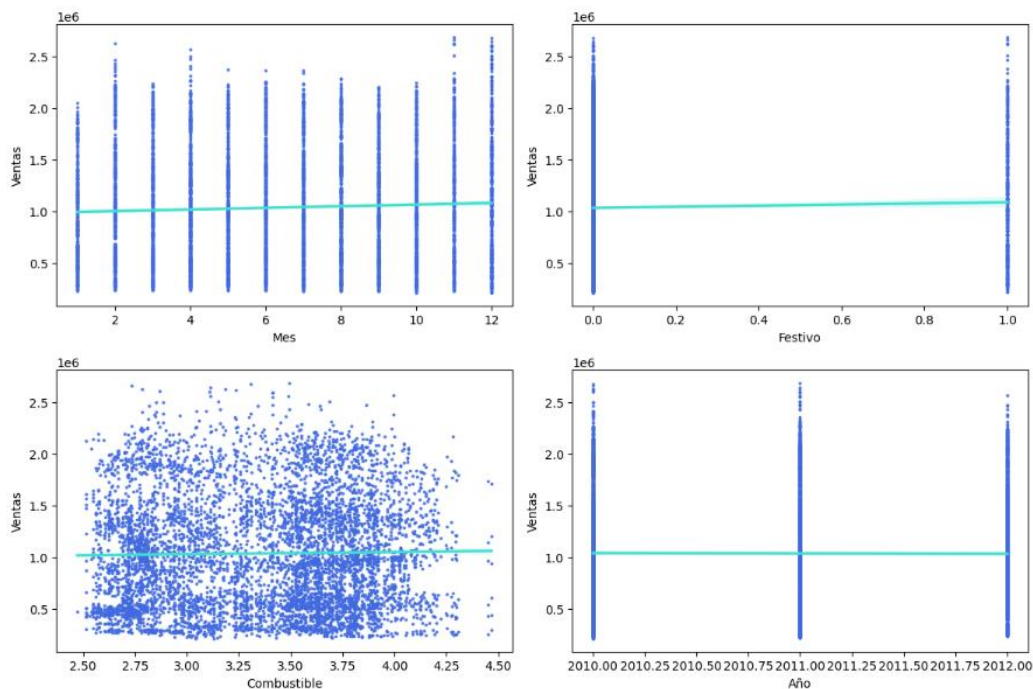
	Tienda	Ventas	Festivo	Temperatura	Combustible	CPI	Desempleo	Día	Mes	Año
Tienda	1.000000	-0.332881	0.003566	-0.026652	0.057863	-0.212481	0.222746	0.003165	0.006855	-0.002834
Ventas	-0.332881	1.000000	0.025358	-0.044340	0.018189	-0.069617	-0.104298	-0.033247	0.046662	-0.005401
Festivo	0.003566	0.025358	1.000000	-0.154556	-0.077808	0.000121	0.012385	0.039835	0.119601	-0.054836
Temperatura	-0.026652	-0.044340	-0.154556	1.000000	0.143080	0.176510	0.099266	0.030957	0.248394	0.060465
Combustible	0.057863	0.018189	-0.077808	0.143080	1.000000	-0.172078	-0.035469	0.029759	-0.038797	0.779444
CPI	-0.212481	-0.069617	0.000121	0.176510	-0.172078	1.000000	-0.304158	0.004109	0.007620	0.074081
Desempleo	0.222746	-0.104298	0.012385	0.099266	-0.035469	-0.304158	1.000000	-0.003121	-0.010091	-0.242957
Día	0.003165	-0.033247	0.039835	0.030957	0.029759	0.004109	-0.003121	1.000000	0.009126	0.009128
Mes	0.006855	0.046662	0.119601	0.248394	-0.038797	0.007620	-0.010091	0.009126	1.000000	-0.190219
Año	-0.002834	-0.005401	-0.054836	0.060465	0.779444	0.074081	-0.242957	0.009128	-0.190219	1.000000

Se puede observar que existe una correlación positiva con la variable Tienda en cuanto a la variable dependiente considerada como Ventas.

7. Comente qué variable escogerán como variable dependiente y que variables introducirán a su modelo.

Se escoge como variable dependiente a Ventas y a variables independientes a Tienda, Festivo, Temperatura, Combustible, CPI, Desempleo, Día, Mes y Año

```
n = 10
fig = plt.figure(figsize=(12,12))
# Correlaciones en pares
corr = numerical_df.corr()
#
cols = corr.nlargest(6, "Ventas")["Ventas"].index
# Calculate correlation
for i in np.arange(1,6):
    regline = df[cols[i]] # Indent this line to include it in the for loop
    ax = fig.add_subplot(3,2,i)
    sns.regplot(x=regline, y=df['Ventas'], scatter_kws={"color": "royalblue", "s": 3},
                line_kws={"color": "turquoise"})
plt.tight_layout()
plt.show()
```



8. Indique que tipo de modelación realizarán y porqué.

Se implementa relación lineal múltiple dado que parece comprobar los supuestos definidos

```
[ ] log_Ventas=np.log(df.Ventas)
df['Ventas']=log_Ventas

regression= ols("Ventas ~ Tienda+Festivo+Temperatura+Combustible+CPI+Desempleo+Dia+Mes+Año", data=df)
results= regression.fit()

[ ] print(results.summary())
```

OLS Regression Results

Dep. Variable:	Ventas	R-squared:	0.115
Model:	OLS	Adj. R-squared:	0.114
Method:	Least Squares	F-statistic:	92.22
Date:	Tue, 06 Aug 2024	Prob (F-statistic):	2.96e-162
Time:	02:34:01	Log-Likelihood:	-5236.1
No. Observations:	6401	AIC:	1.049e+04
Df Residuals:	6391	BIC:	1.056e+04
Df Model:	9		
Covariance Type:	nonrobust		

	coef	std err	t	P> t	[0.025	0.975]
Intercept	60.1130	31.794	1.891	0.059	-2.214	122.440
Tienda	-0.0141	0.001	-25.601	0.000	-0.015	-0.013
Festivo	0.0136	0.028	0.487	0.626	-0.041	0.068
Temperatura	-0.0026	0.000	-6.160	0.000	-0.003	-0.002
Combustible	0.0727	0.027	2.695	0.007	0.020	0.126
CPI	-0.0019	0.000	-9.427	0.000	-0.002	-0.002
Desempleo	-0.0136	0.004	-3.288	0.001	-0.022	-0.006
Dia	-0.0019	0.001	-2.475	0.013	-0.003	-0.000
Mes	0.0098	0.002	4.199	0.000	0.005	0.014
Año	-0.0228	0.016	-1.437	0.151	-0.054	0.008

Omnibus: 489.387 Durbin-Watson: 0.061
Prob(Omnibus): 0.000 Jarque-Bera (JB): 336.586
Skew: -0.451 Prob(JB): 8.15e-74
Kurtosis: 2.330 Cond. No. 9.36e+06

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 9.36e+06. This might indicate that there are strong multicollinearity or other numerical problems.

Se evidencia un R2 bajo, lo cual motiva a que se debe mejorar el modelo, se evidencia también un p valor alto en las variables Festivo y Año, para lo cual se decide separar.

```
regression_2= ols("Ventas ~ Tienda+Temperatura+Combustible+CPI+Desempleo+Dia+Mes", data=df)
results_2= regression_2.fit()

print(results_2.summary())
```

OLS Regression Results

Dep. Variable:	Ventas	R-squared:	0.115
Model:	OLS	Adj. R-squared:	0.114
Method:	Least Squares	F-statistic:	118.2
Date:	Tue, 06 Aug 2024	Prob (F-statistic):	7.59e-164
Time:	02:54:04	Log-Likelihood:	-5237.3
No. Observations:	6401	AIC:	1.049e+04
Df Residuals:	6393	BIC:	1.054e+04
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P> t	[0.025	0.975]
Intercept	14.4357	0.002	176.860	0.000	14.276	14.596
Tienda	-0.0141	0.001	-25.691	0.000	-0.015	-0.013
Temperatura	-0.0026	0.000	-6.305	0.000	-0.003	-0.002
Combustible	0.0409	0.016	2.619	0.009	0.010	0.071
CPI	-0.0020	0.000	-10.192	0.000	-0.002	-0.002
Desempleo	-0.0119	0.004	-2.995	0.003	-0.020	-0.004
Dia	-0.0019	0.001	-2.420	0.016	-0.003	-0.000
Mes	0.0108	0.002	4.897	0.000	0.006	0.015

Omnibus: 493.988 Durbin-Watson: 0.061
Prob(Omnibus): 0.000 Jarque-Bera (JB): 335.056
Skew: -0.447 Prob(JB): 1.75e-73
Kurtosis: 2.325 Cond. No. 2.26e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.26e+03. This might indicate that there are strong multicollinearity or other numerical problems.

Se tiene un p valor aceptable para todas las variables significativas en nuestro modelo, pero nuestro R2 sigue siendo muy bajo, se ha hecho varias iteraciones y el R2 no logra ser bueno, se recomienda revisar la base de datos,

que cumpla con la calidad y la precisión, por otro lado se recomienda agregar valores relevantes y validar si se puede elevar el R2

9. Verifique los supuestos, de haber escogido el enfoque econométrico.

Los supuestos que se van a definir son:

Multicolinealidad:

Se va a importar la librería statsmodel para revisar el VIF, para poder analizar el VIF es necesario tener variables numéricas

```
[ ] from statsmodels.stats.outliers_influence import variance_inflation_factor

[ ] df2=df[df.columns.difference(['Ventas', 'log_Ventas'])]
df2
```

	Año	CPI	Combustible	Desempleo	Dia	Fecha	Festivo	Mes	Temperatura	Tienda
0	2010	211.096358	2.572	8.106	5	2010-02-05	0	2	42.31	1
1	2010	211.242170	2.548	8.106	12	2010-02-12	1	2	38.51	1
2	2010	211.289143	2.514	8.106	19	2010-02-19	0	2	39.93	1
3	2010	211.319643	2.561	8.106	26	2010-02-26	0	2	46.63	1
4	2010	211.350143	2.625	8.106	5	2010-03-05	0	3	46.50	1
...
6430	2012	192.013558	3.997	8.684	28	2012-09-28	0	9	64.88	45
6431	2012	192.170412	3.985	8.667	5	2012-10-05	0	10	64.89	45
6432	2012	192.327265	4.000	8.667	12	2012-10-12	0	10	54.47	45
6433	2012	192.330854	3.969	8.667	19	2012-10-19	0	10	56.47	45
6434	2012	192.308899	3.882	8.667	26	2012-10-26	0	10	58.85	45

6401 rows x 10 columns

```
[ ] df2.dtypes
```

	Año	CPI	Combustible	Desempleo	Dia	Fecha	Festivo	Mes	Temperatura	Tienda
	int32	float64	float64	float64	int32	datetime64[ns]	int64	int32	float64	int64
dtype:	object									

Ahora si se calcula el VIF de las variables, y se puede evidenciar un VIF >1 y <5, que muestra que la correlación es moderada y no requiere atención.

```
] # Creamos el dataframe del VIF
vif_data = pd.DataFrame()
vif_data["feature"] = df2.columns

# Calculamos el VIF por c/variable
vif_data["VIF"] = [variance_inflation_factor(df2.values, i) for i in range(len(df2.columns))]

print(vif_data)
```

	feature	VIF
0	Año	0.000448
1	CPI	1.005970
2	Combustible	1.244165
3	Desempleo	0.955121
4	Dia	1.001742
5	Fecha	9481.851068
6	Festivo	0.999840
7	Mes	1.009912
8	Temperatura	1.014566
9	Tienda	0.998822

La variable fecha no es una variable analizada potencial, dado a que está dividida según nuestro tratamiento de datos. Procedemos a generar nuevamente nuestro modelo.


```
regresion_2= ols("Ventas ~ Tienda+Temperatura+Combustible+CPI+Desempleo+Dia+Mes", data=df)
results_2= regresion_2.fit()
```

```
print(results_2.summary())
```

```

=====
                        OLS Regression Results
=====
Dep. Variable:          Ventas      R-squared:                0.115
Model:                  OLS        Adj. R-squared:             0.114
Method:                 Least Squares   F-statistic:             118.2
Date:                  Tue, 06 Aug 2024   Prob (F-statistic):       7.59e-164
Time:                  02:54:04         Log-Likelihood:          -5237.3
No. Observations:      6401          AIC:                    1.049e+04
Df Residuals:          6393          BIC:                    1.054e+04
Df Model:              7
Covariance Type:       nonrobust
=====
               coef      std err          t      P>|t|      [0.025      0.975]
-----
Intercept      14.4357      0.082     176.860      0.000      14.276      14.596
Tienda        -0.0141      0.001    -25.691      0.000      -0.015      -0.013
Temperatura   -0.0026      0.000     -6.305      0.000      -0.003      -0.002
Combustible    0.0409      0.016      2.619      0.009      0.010      0.071
CPI           -0.0020      0.000    -10.192      0.000      -0.002      -0.002
Desempleo     -0.0119      0.004     -2.995      0.003      -0.020      -0.004
Dia           -0.0019      0.001     -2.420      0.016      -0.003      -0.000
Mes            0.0108      0.002      4.897      0.000      0.006      0.015
=====
Omnibus:                 493.988   Durbin-Watson:           0.061
Prob(Omnibus):            0.000   Jarque-Bera (JB):        335.056
Skew:                    -0.447   Prob(JB):                 1.75e-73
Kurtosis:                 2.325   Cond. No.                 2.26e+03
=====

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.26e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Se muestran los resultados quitando las variables, y a pesar de aquello mi R2 no mejora, de hecho, sigue siendo suficientemente bajo como para predecir que el modelo no es bueno y que se deberían hacer tratamientos en las bases de datos o escoger diferentes datos de análisis.

Normalidad en los residuos

Se procede aplicar el test de Jarque Bera.

```

nombres = ["Jarque-Bera", "Chi^2 two-tail prob.", "Skew", "Kurtosis"]
jarque_bera = sms.jarque_bera(results.resid)
lzip(nombres, jarque_bera)

[('Jarque-Bera', 336.58560593463324),
 ('Chi^2 two-tail prob.', 8.153880217361672e-74),
 ('Skew', -0.45083599982149036),
 ('Kurtosis', 2.3299354956448264)]

[52] results.resid.mean()

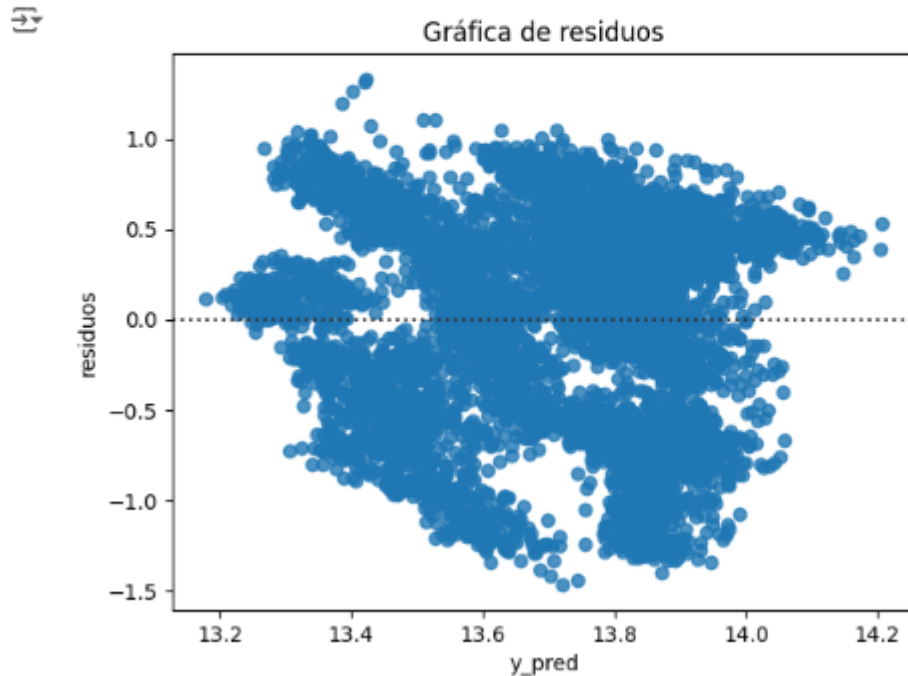
2.8291944634574674e-12
```

Los datos no se distribuyen normalmente, su media es 2.8 lo cual nos indica una vez más que el modelo no es adecuado, los datos analizados no nos están ayudando a predecir buenos resultados.

Homocedasticidad

```
[53] y_pred=results.predict()
```

```
sns.residplot(x=y_pred, y=results.resid)  
plt.xlabel("y_pred")  
plt.ylabel("residuos")  
plt.title("Gráfica de residuos")  
plt.show()
```



```
[54] nombres = ["Lagrange multiplier statistic", "p-value", "f-value", "f p-value"]  
breuschpagan = sms.het_breuschpagan(results.resid, results.model.exog)  
lzip(nombres, breuschpagan)
```

```
[('Lagrange multiplier statistic', 290.1341603974956),  
 ('p-value', 3.225122290430188e-57),  
 ('f-value', 33.7149425987981),  
 ('f p-value', 1.3646080719257607e-58)]
```

El modelo si tiene homocedasticidad, se observa la formación de una nube de puntos alrededor de la línea de tendencia central. Cumple con este supuesto.

10. Obtenga el modelo definitivo, prediga los valores y comente el grado de ajuste del modelo. Justifique con métricas su respuesta.

Se muestran los resultados quitando las variables, y a pesar de aquello mi R2 no mejora, por lo cual sigo las recomendaciones previamente mencionadas:

OLS Regression Results

Dep. Variable:

Ventas

R-squared:

0.115

Model:

OLS

Adj. R-squared:

0.114

Method:

Least Squares

F-statistic:

118.2

Date:

Tue, 06 Aug 2024

Prob (F-statistic):

7.59e-164

Time:

02:54:04

Log-Likelihood:

-5237.3

No. Observations:

6401

AIC:

1.049e+04

Df Residuals:

6393

BIC:

1.054e+04

Df Model:

7

Covariance Type:

nonrobust

coef

std err

t

P>|t|

[0.025

0.975]

Intercept

14.4357

0.082

176.860

0.000

14.276

14.596

Tienda

-0.0141

0.001

-25.691

0.000

-0.015

-0.013

Temperatura

-0.0026

0.000

-6.305

0.000

-0.003

-0.002

Combustible

0.0409

0.016

2.619

0.009

0.010

0.071

CPI

-0.0020

0.000

-10.192

0.000

-0.002

-0.002

Desempleo

-0.0119

0.004

-2.995

0.003

-0.020

-0.004

Dia

-0.0019

0.001

-2.420

0.016

-0.003

-0.000

Mes

0.0108

0.002

4.897

0.000

0.006

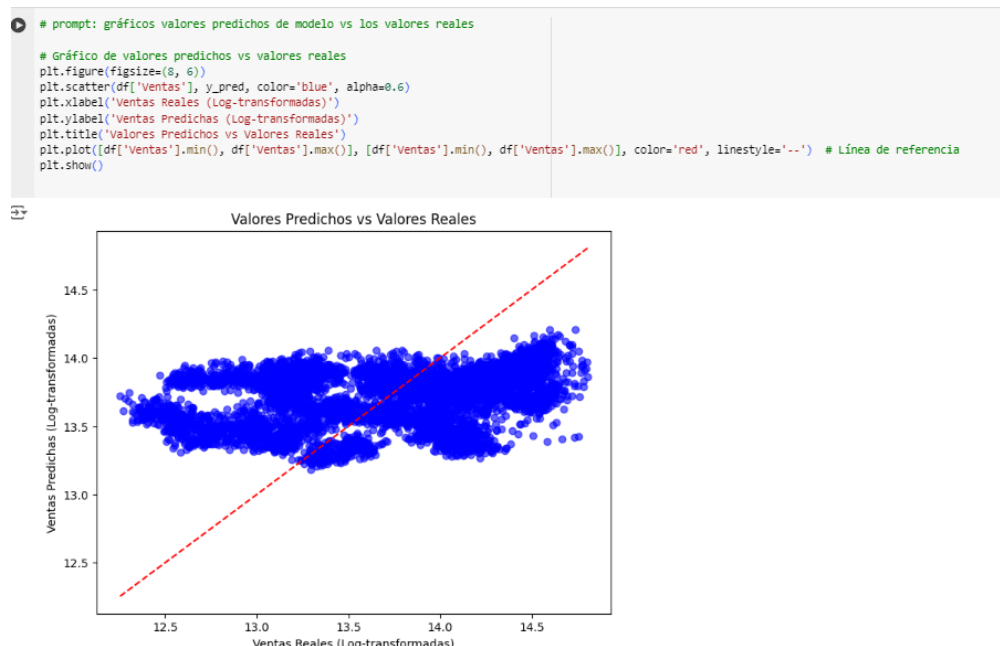
0.015

✓ 0s

completed at 9:54 PM

Como se observa se ha trabajado con el último modelo generado, aun así no se ha logrado obtener un modelo que aumente su valor R2 con lo cual después de varias iteraciones y con los mismos valores generados de R2, se destaca que el modelo no ha mejorado.

11. Grafique a los valores predicho de modelo vs los valores reales.



Este gráfico puede estar reflejado el mal ajuste que se ve con R2 bajo.

12. ¿Cómo se ven una vez graficados frente a los valores reales? Argumente su respuesta.

Los valores no se ven correctos, dado que deberían estar más apegados a la línea, de hecho deberían tener la misma forma de la línea de tendencia de los valores reales, aun así, se muestra que el modelo no es un buen predictor de valores futuros, lo cual indica que se podría seguir intentando mejorar el modelo usando otro tipo de metodologías, se podría intentar generar un modelo con IA entrenado para considerar mejoras.

13. Concluya sobre su modelo. Para ello, si escogió el enfoque econométrico, interprete coeficientes, por el contrario si escogió el enfoque de machine learning, determine cuáles son las variables que tienen mayor poder explicativo sobre su variable objetivo.

```
# Assuming 'results_2' is the fitted OLS model
print(results_2.params)
```

Intercept	60.112963
Tienda	-0.014068
Festivo	0.013597
Temperatura	-0.002552
Combustible	0.072738
CPI	-0.001911
Desempleo	-0.013632
Día	-0.001942
Mes	0.009775
Año	-0.022764

dtype: float64

Los coeficientes muestran que mis variables Festivo, Combustible, y Mes son las variables con impacto positivo, por otro lado, mis variables Tienda, Temperatura, CPI, Desempleo, Día y año tendrán un impacto negativo.

Finalmente, se puede mencionar que, dados los dos intentos de trabajo con la base de datos dada, no se ha logrado tener un buen modelo, se ha intentado por dos caminos y en el primero se consiguió obtener un R2 negativo que no es un buen indicador del funcionamiento general de la base de datos, de hecho, se puede considerar como 0 al valor R2 en el modelo de regresión logística, adicionando a esto, en los análisis de supuestos no se ha obtenido buenos resultados.

En la segunda prueba se trato con regresión lineal múltiple, esto específicamente a que las variables independientes son todas numéricas, de este modo, consideramos que la mejor manera de analizar estos datos es mediante el uso de esta herramienta, como se ha visualizado a lo largo de este informe, tampoco se logró obtener buenos resultados, a comparación del método de regresión logística, si se obtuvo un R2 positivo, sin embargo, no es un porcentaje suficiente como para considerar que el modelo es adecuado y que por ende en fases de predicción se comportará de manera adecuada, a pesar de aquello el modelo si muestra homocedasticidad.

Se sugiere agregar variables con las cuales se pueda analizar, o a su vez aplicar transformaciones a las mismas, esto con el fin de tener mayor calidad en los datos, verificar y corregir datos previamente, también se sugiere agregar mas variables a la muestra.

El gráficos de los valores predichos vs valores reales puede estar reflejando el mal ajuste que se ve con el R2.

Se recomienda seguir tratando la base de datos por distintos medios de preproceso de datos de modo que se efectúen transformaciones de datos para obtener una normalidad en los mismos, se recomienda igualmente trabajar con divisiones de los datos como en el caso de la fecha que permite una facilidad al usar los datos en pasos posteriores. Adicionalmente, se sugiere tratar con otras herramientas para la base de datos como el análisis con machine learning para crear un set de entrenamiento y ver como se comportan los datos en general, quizás esto permita un mejor uso de los mismos y mejores resultados finales.