ANALITICA PREDICTIVA

TAREA: FINAL SEMANA 4

INTEGRANTES:

Ana Prado
Margarita Yambay

CASO FINAL DE ANALISIS

Modelar a su preferencia (pero justificado datos de la cadena de supermercados de EEUU, Wallmart.)

La base con la que trabajaremos este caso práctico contiene información sobre datos históricos de las ventas de Wallmart desde 2010-02-05 hasta 2012-11-01, en el archivo WalmartStoresales. Dentro de este archivo encontrará los siguientes campos:

- Tienda-el número de la tienda.
- Fecha-la semana de ventas
- Weekly_Sales - ventas para la tienda dada.
- Holiday_Flag: si la semana es una semana especial de vacaciones 1 – Semana de vacaciones 0 – Semana no festiva.
- Temperatura - Temperatura el día de la venta.
- Fuel_price -costo del combustible en la región.
- IPC-índice de precios al consumidor vigente.
- Desempleo - tasa de desempleo predominante.
- Eventos festivos.
  - Super bowl: 12 de febrero de 2010, 11 de febrero de 2011, 10 de febrero de 2012, 8 de febrero de 2013\
  - Día del Trabajo: 10-sep-10, 9-sep-11, 7-sep-12, 6-sep-13\
  - Acción de Gracias: 26-nov-10, 25-nov-11, 23-nov-12, 29-nov-13\
  - Navidad: 31-dic-10, 30-dic-11, 28-dic-12, 27-dic-13

1.- Importe la base de datos a una base en Jupyter Notebook con pandas.(Walmart) .

In [194…
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings("ignore")
```

In [194…
```
df=pd.read_csv("data/Walmart.csv")
df
```

Out[1946]:

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unem
0	1	05-02-2010	1643690.90	0	42.31	2.572	211.096358	
1	1	12-02-2010	1641957.44	1	38.51	2.548	211.242170	
2	1	19-02-2010	1611968.17	0	39.93	2.514	211.289143	
3	1	26-02-2010	1409727.59	0	46.63	2.561	211.319643	
4	1	05-03-2010	1554806.68	0	46.50	2.625	211.350143	
...	...	...	...	...	...	...	...	
6430	45	28-09-2012	713173.95	0	64.88	3.997	192.013558	
6431	45	05-10-2012	733455.07	0	64.89	3.985	192.170412	
6432	45	12-10-2012	734464.36	0	54.47	4.000	192.327265	
6433	45	19-10-2012	718125.53	0	56.47	3.969	192.330854	
6434	45	26-10-2012	760281.43	0	58.85	3.882	192.308899	

6435 rows × 8 columns

2.- Obtenga los descriptivos resumen de la base de datos e identifique las variables numérica y categóricas. Indique , hay algo que le llame la atención?

In [194...
```
df.describe()
```

Out[1947]:

	Store	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemplo
count	6435.000000	6.435000e+03	6435.000000	6435.000000	6435.000000	6435.000000	6435.
mean	23.000000	1.046965e+06	0.069930	60.663782	3.358607	171.578394	7.
std	12.988182	5.643666e+05	0.255049	18.444933	0.459020	39.356712	1.
min	1.000000	2.099862e+05	0.000000	-2.060000	2.472000	126.064000	3.
25%	12.000000	5.533501e+05	0.000000	47.460000	2.933000	131.735000	6.
50%	23.000000	9.607460e+05	0.000000	62.670000	3.445000	182.616521	7.
75%	34.000000	1.420159e+06	0.000000	74.940000	3.735000	212.743293	8.
max	45.000000	3.818686e+06	1.000000	100.140000	4.468000	227.232807	14.

In [194...
```
df.rename({'Store':'nro_tienda', 'Date': 'fecha', 'Weekly_Sales':'ventas', 'Holiday
df
```

Out[1948]:

	nro_tienda	fecha	ventas	feriado	temperatura	precio_combustible	ind_precio_c
0	1	05-02-2010	1643690.90	0	42.31	2.572	
1	1	12-02-2010	1641957.44	1	38.51	2.548	
2	1	19-02-2010	1611968.17	0	39.93	2.514	
3	1	26-02-2010	1409727.59	0	46.63	2.561	
4	1	05-03-2010	1554806.68	0	46.50	2.625	
...	...	...	...	...	...	...	
6430	45	28-09-2012	713173.95	0	64.88	3.997	
6431	45	05-10-2012	733455.07	0	64.89	3.985	
6432	45	12-10-2012	734464.36	0	54.47	4.000	
6433	45	19-10-2012	718125.53	0	56.47	3.969	
6434	45	26-10-2012	760281.43	0	58.85	3.882	

6435 rows × 8 columns

In [194…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6435 entries, 0 to 6434
Data columns (total 8 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   nro_tienda            6435 non-null   int64  
 1   fecha                 6435 non-null   object 
 2   ventas                6435 non-null   float64
 3   feriado               6435 non-null   int64  
 4   temperatura           6435 non-null   float64
 5   precio_combustible    6435 non-null   float64
 6   ind_precio_consumidor 6435 non-null   float64
 7   tasa_desempleo        6435 non-null   float64
dtypes: float64(5), int64(2), object(1)
memory usage: 402.3+ KB
```

In [195…

```
var_cuantitativas = df.select_dtypes('number').columns
print("Variables cuantitativas:", var_cuantitativas)

var_cualitativas  =df.select_dtypes('object').columns
print("Variables cualitativas:", var_cualitativas)
```

```
Variables cuantitativas: Index(['nro_tienda', 'ventas', 'feriado', 'temperatura',
'precio_combustible',
       'ind_precio_consumidor', 'tasa_desempleo'],
      dtype='object')
Variables cualitativas: Index(['fecha'], dtype='object')
```

Los tipos de variables identificadas son:

VARIABLES NUMERICAS: nro_tienda, ventas, feriado, temperatura, precio_combustible, ind_precio_consumidor, tasa_desempleo.

VARIABLES CATEGORICAS: fecha

```
Se puede observar completitud de los datos (6435 registros).
La variable feriado esta dumificada.
Se puede visualizar información en grupo respecto a las tiendas
(total 45 tiendas o almacenes)
```

3._Evalúe si la base contiene datos perdidos.

In [195…
```
df.isna().sum()
```

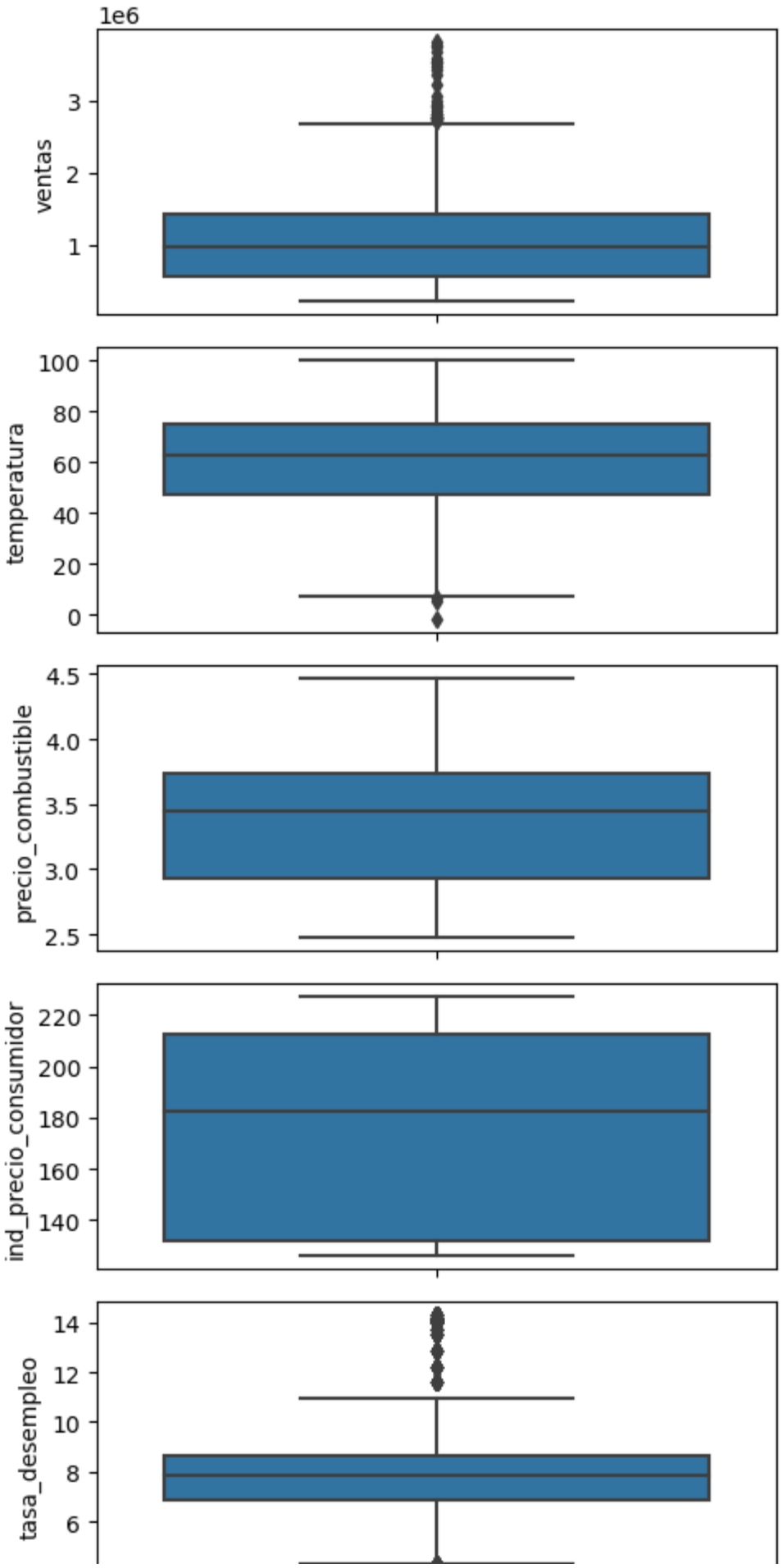
Out[1951]:
```
nro_tienda              0
fecha                   0
ventas                  0
feriado                 0
temperatura             0
precio_combustible      0
ind_precio_consumidor   0
tasa_desempleo          0
dtype: int64
```

En la data no hay valores nulos o perdidos.

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

-De ser el caso, detalle cuáles y qué método estadístico aplicarán para corregir.

In [195…
```
# Identificamos a través de la función Box plots

fig, axs = plt.subplots(5, figsize = (5,10))

plt1 = sns.boxplot(y=df['ventas'], ax = axs[0])
plt2 = sns.boxplot(y=df['temperatura'], ax = axs[1])
plt3 = sns.boxplot(y=df['precio_combustible'], ax = axs[2])
plt4 = sns.boxplot(y=df['ind_precio_consumidor'], ax = axs[3])
plt5 = sns.boxplot(y=df['tasa_desempleo'], ax = axs[4])

plt.tight_layout()
```


1e6
ventas
3
2
1
temperatura
100
80
60
40
20
0
precio_combustible
4.5
4.0
3.5
3.0
2.5
ind_precio_consumidor
220
200
180
160
140
tasa_desempleo
14
12
10
8
6

En la gráfica podemos observar datos atípicos en las variables de: ventas,temperatura, tasa de desempleo.

Para corregir estos valores atípicos, removemos las observaciones que se encuentran fuera del rango 1.5xIQR.

In [195…
```
# Calculamos el Quartil 1 y Quartil 3 que son aquellos que nos permiten estimar los
Q1 = df['ventas'].quantile(0.25)
Q3 = df['ventas'].quantile(0.75)
IQR = Q3 - Q1  #rango intercuartil
print(IQR)
```

```
866808.5549999999
```

In [195…
```
# Ahora removemos aquellas observaciones que se encuentran por fuera del rango: 1.5
df = df[~((df['ventas'] < (Q1 - 1.5 * IQR)) |(df['ventas'] > (Q3 + 1.5 * IQR)))]
df.shape
```

Out[1954]: (6401, 8)

In [195…
```
# Calculamos el Quartil 1 y Quartil 3 que son aquellos que nos permiten estimar los
Q1 = df['temperatura'].quantile(0.25)
Q3 = df['temperatura'].quantile(0.75)
IQR = Q3 - Q1  #rango intercuartil
print(IQR)
```

```
27.340000000000003
```

In [195…
```
#Ahora removemos aquellas observaciones que se encuentran por fuera del rango: 1.5
df = df[~((df['temperatura'] < (Q1 - 1.5 * IQR)) |(df['temperatura'] > (Q3 + 1.5 *
df.shape
```

Out[1956]: (6398, 8)

In [195…
```
# Calculamos el Quartil 1 y Quartil 3 que son aquellos que nos permiten estimar los
Q1 = df['tasa_desempleo'].quantile(0.25)
Q3 = df['tasa_desempleo'].quantile(0.75)
IQR = Q3 - Q1  #rango intercuartil
print(IQR)
```

```
1.7309999999999999
```

In [195…
```
# Ahora removemos aquellas observaciones que se encuentran por fuera del rango: 1.5
df = df[~((df['tasa_desempleo'] < (Q1 - 1.5 * IQR)) |(df['tasa_desempleo'] > (Q3 +
df.shape
```

Out[1958]: (5917, 8)

In [195…
```
#PARA USO DE REGRESION LINEAL
dfr = df
```

In [196…
```
#Revisión de los valores actualizadas de las variables
df.describe()
```

Out[1960]:

	nro_tienda	ventas	feriado	temperatura	precio_combustible	ind_precio_consu
count	5917.000000	5.917000e+03	5917.000000	5917.000000	5917.000000	5917.
mean	22.801251	1.039313e+06	0.069123	60.433407	3.340543	175.
std	13.094060	5.519450e+05	0.253684	18.386455	0.458200	39.
min	1.000000	2.099862e+05	0.000000	7.460000	2.472000	126.
25%	11.000000	5.525292e+05	0.000000	46.980000	2.891000	132.
50%	22.000000	9.472292e+05	0.000000	62.620000	3.420000	190.
75%	34.000000	1.427624e+06	0.000000	74.730000	3.721000	213.
max	45.000000	2.685352e+06	1.000000	100.140000	4.468000	227.

Revisión descriptivos:

Se observa que el estudio esta realizado sobre 45 tiendas (max nro_tienda= 45 almacenes)

También se observa que las ventas (semanales) tienen una media de alrededor 1039313 unidades, desviación estándar de aproximadamente 551945 unidades.

La tasa de desempleo va de un valor mínimo de 1.243 a un valor máximo considerable 10.926.

El indice de precio al consumidor va de un valor mínimo de 39.023 a un valor máximo considerable 227.23.

El precio del combustible tiene valores promedio de 3.34 unidades.

Existen cambios de temperatura que va de 7.46 grados a 100.14 grados.

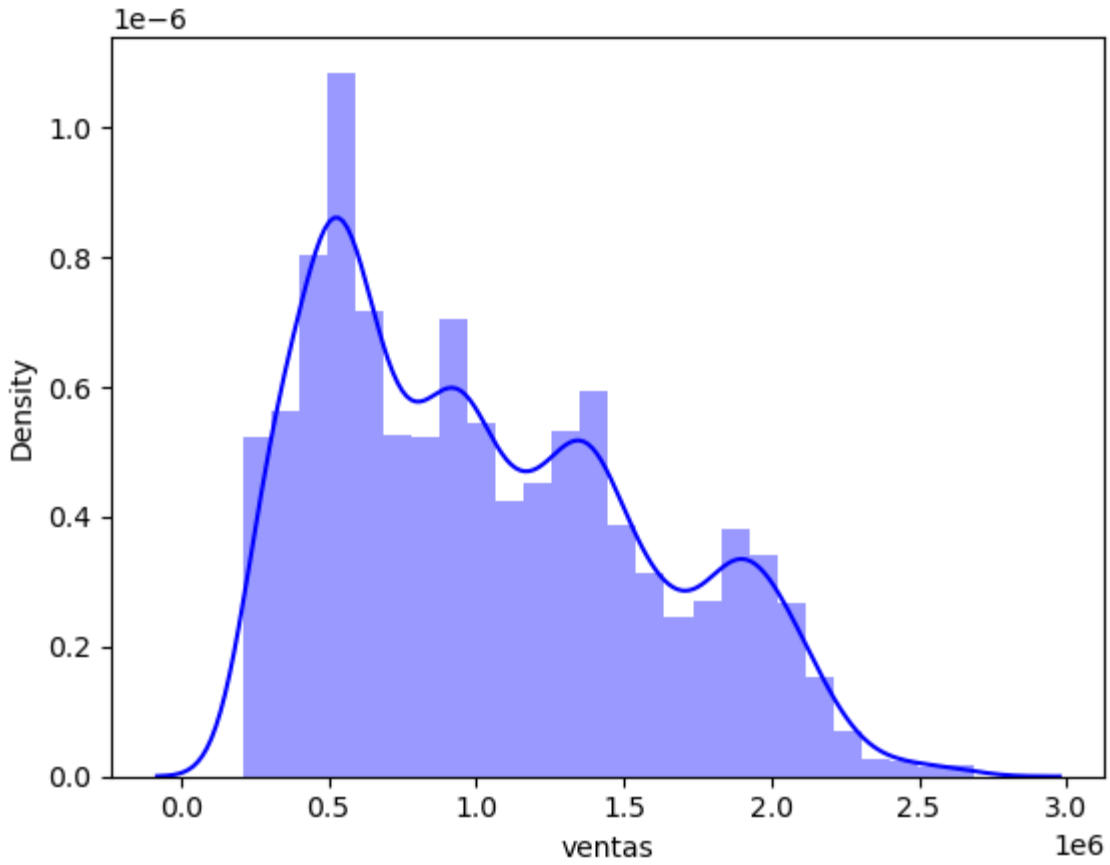
5.Grafique las distribuciones de las variables y a priori comente sobre ellas.

In [196…

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

Out[1961]:

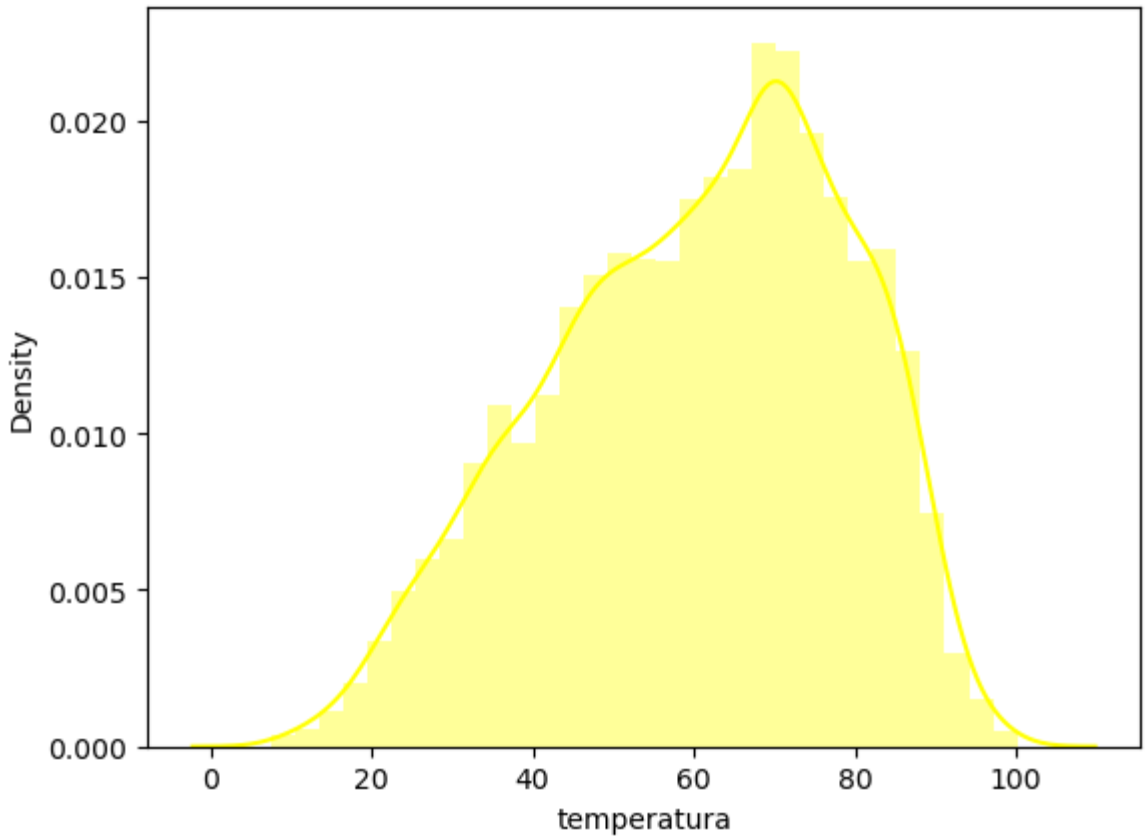
```
<Axes: xlabel='ventas', ylabel='Density'>
```



- Del gráfico podríamos indicar que la distribución multimodal, están sesgados a la derecha, el sesgo estadístico es positivo.

In [196…
```
sns.distplot(df['temperatura'],color="yellow")
```

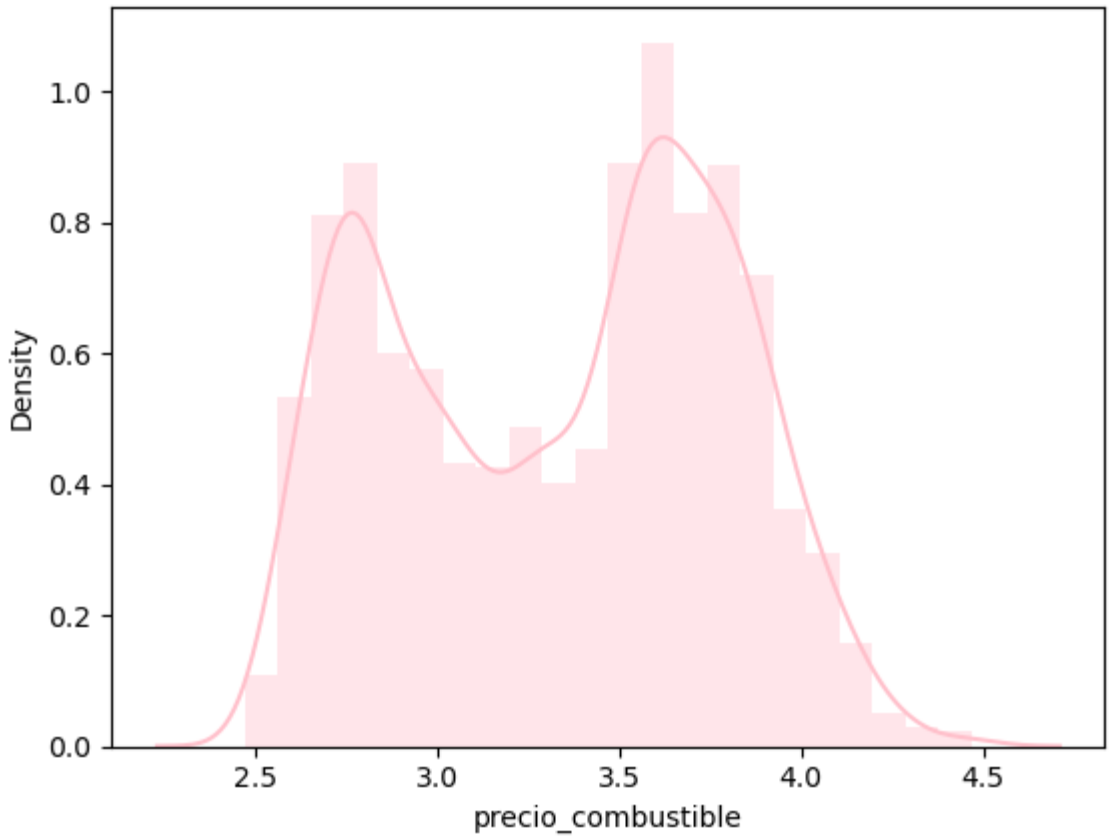
Out[1962]:
```
<Axes: xlabel='temperatura', ylabel='Density'>
```



- Del gráfico con la variable temperatura podríamos indicar que la distribución esta sesgada hacia la izquierda lo que nos indica que hay pocos valores bajos y muchos valores altos.

In [196…
```
sns.distplot(df['precio_combustible'],color="pink")
```

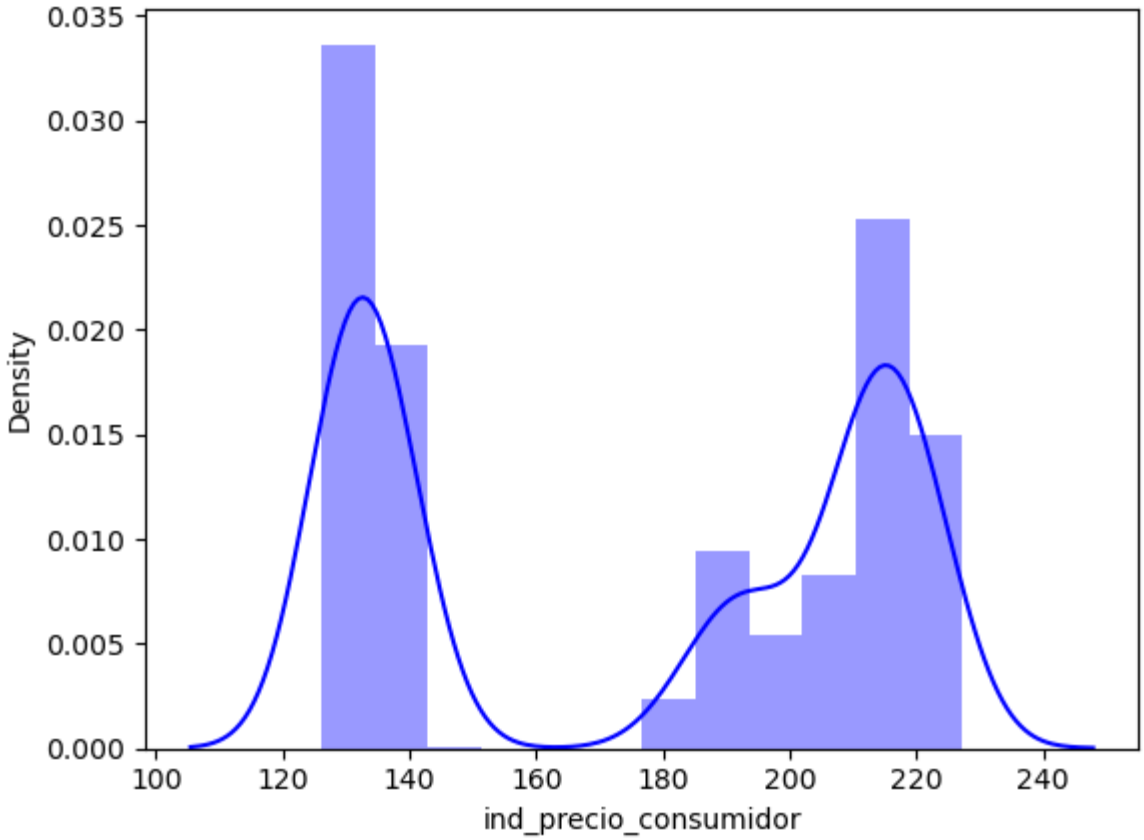
Out[1963]:
```
<Axes: xlabel='precio_combustible', ylabel='Density'>
```



- Del gráfico con la variable temperatura podríamos indicar que la distribución es bimodal, tiene dos picos.

In [196…
```
sns.distplot(df['ind_precio_consumidor'],color="blue")
```

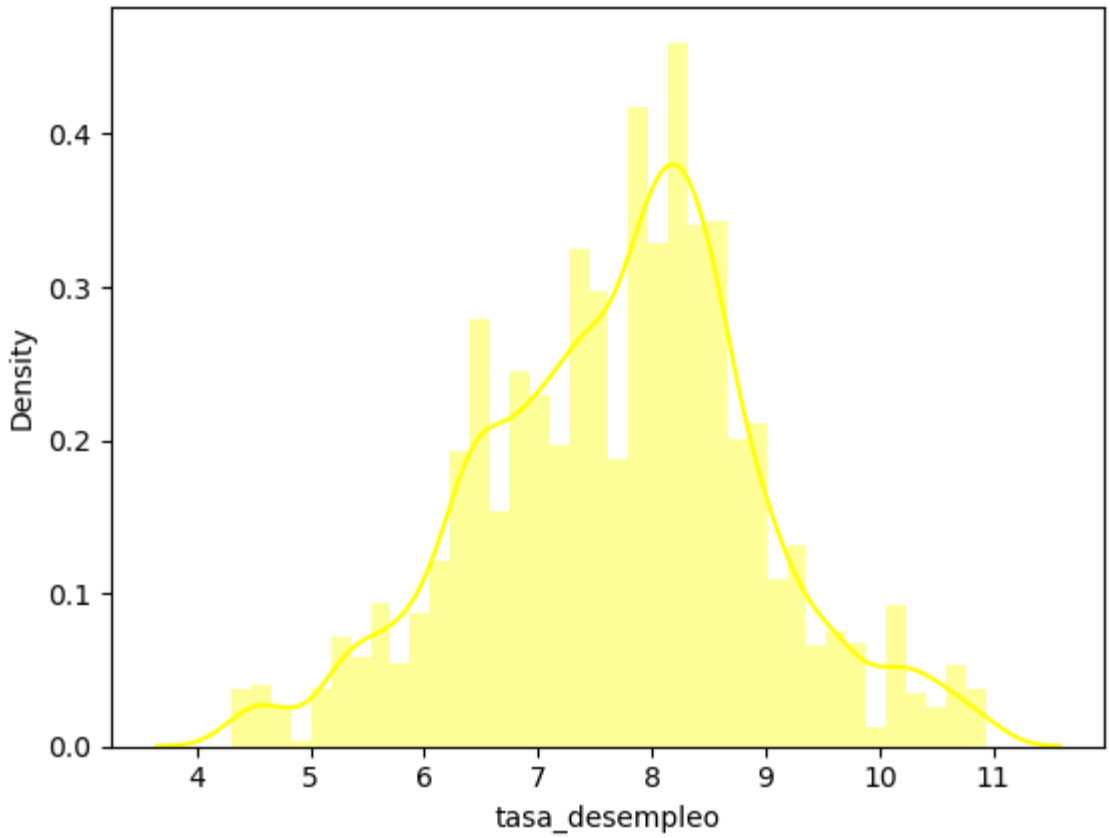
Out[1964]:
```
<Axes: xlabel='ind_precio_consumidor', ylabel='Density'>
```



- Del gráfico con la variable temperatura podríamos indicar que la distribución es bimodal, se puede identificar dos picos.

In [196…
```
sns.distplot(df['tasa_desempleo'],color="yellow")
```

Out[1965]:
```
<Axes: xlabel='tasa_desempleo', ylabel='Density'>
```



- Del gráfico con la variable temperatura podríamos indicar que la distribución esta sesgada hacia la izquierda lo que nos indica que hay pocos valores bajos y muchos valores altos.

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

In [196…

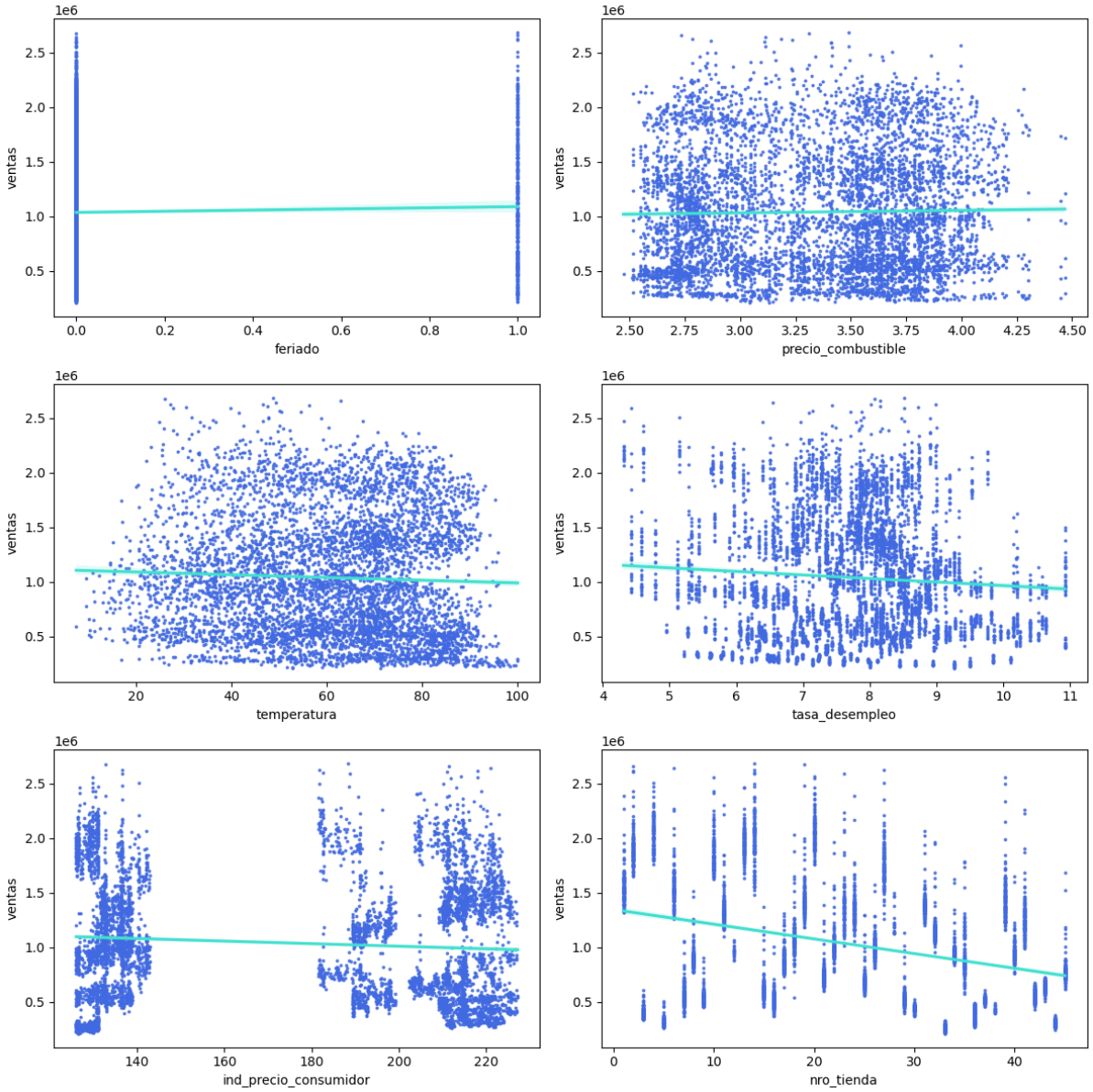
```
#Obtenémos el gráfico de correlación de las variables del modelo

dfc=df[["ventas","nro_tienda","temperatura","precio_combustible","ind_precio_consum
dfc.corr().style.background_gradient(cmap='coolwarm')
```

Out[1966]:

	ventas	nro_tienda	temperatura	precio_combustible	ind_precio_consun
ventas	1.000000	-0.319354	-0.041686	0.019664	-0.08
nro_tienda	-0.319354	1.000000	-0.027045	0.047519	-0.20
temperatura	-0.041686	-0.027045	1.000000	0.145157	0.21
precio_combustible	0.019664	0.047519	0.145157	1.000000	-0.14
ind_precio_consumidor	-0.082977	-0.208637	0.217847	-0.144515	1.00
tasa_desempleo	-0.073092	0.309645	0.024204	-0.105214	-0.21
feriado	0.024390	0.004289	-0.157220	-0.076529	-0.00

In [196…

```
# Número de las variables
n = 7
fig = plt.figure(figsize=(12,12))
# Correlaciones en pares
corr = dfc.corr()
#
cols = corr.nlargest(7, "ventas")["ventas"].index
# Calculate correlation
for i in np.arange(1,7):
    regline = dfc[cols[i]]
    ax = fig.add_subplot(3,2,i)
    sns.regplot(x=regline, y=dfc['ventas'], scatter_kws={"color": "royalblue", "s":
                line_kws={"color": "turquoise"})
plt.tight_layout()
plt.show()
```



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

Como variable dependiente (y) se escoge **ventas**, las ventas son el objeto de estudio en el modelo. Como variables explicativas o independientes (x) **precio_combustible, temperatura, tasa_desempleo, ind_precio_consumidor, feriado**, se observa por los gráficos donde se analiza la correlación y dispersión, las variables que tienen mayor incidencia en el modelo según se aprecia inicialmente son precio y temperatura.

8.-Indique que tipo de modelación realizarán y porqué.

En vista que tenemos en la data una variable fecha y además tienen variable de grupo de número de tienda y hay una variable ventas que depende de varias factores independientes como son precio_combustible, temperatura, tasa_desempleo, ind_precio_consumidor, feriado, a priori vamos a probar el modelo de panel.

1-MODELO DE DATOS DE PANEL CON VARIABLE DEPENDIENTE VENTAS

Requerimos especificar los índice para nuestra base de datos. En este caso, setiaremos la fecha y el número de tienda

In [196…
```
df=df.set_index(['fecha', 'nro_tienda'])
```

In [196…
```
fecha = df.index.get_level_values('fecha').to_list()
df['fecha'] = pd.Categorical(fecha)
```

In [197…
```
!pip install linearmodels
from linearmodels import PooledOLS
import statsmodels.api as sm
```

```
Requirement already satisfied: linearmodels in d:\aplicacion\lib\site-packages (5.
4)
Requirement already satisfied: numpy>=1.22.0 in d:\aplicacion\lib\site-packages (fr
om linearmodels) (1.24.3)
Requirement already satisfied: pandas>=1.3.0 in d:\aplicacion\lib\site-packages (fr
om linearmodels) (2.0.3)
Requirement already satisfied: scipy>=1.5.0 in d:\aplicacion\lib\site-packages (fro
m linearmodels) (1.11.1)
Requirement already satisfied: statsmodels>=0.12.0 in d:\aplicacion\lib\site-packag
es (from linearmodels) (0.14.0)
Requirement already satisfied: mypy-extensions>=0.4 in d:\aplicacion\lib\site-packa
ges (from linearmodels) (1.0.0)
Requirement already satisfied: Cython>=0.29.37 in d:\aplicacion\lib\site-packages
(from linearmodels) (3.0.9)
Requirement already satisfied: pyhdfe>=0.1 in d:\aplicacion\lib\site-packages (from
linearmodels) (0.2.0)
Requirement already satisfied: formulaic>=0.6.5 in d:\aplicacion\lib\site-packages
(from linearmodels) (1.0.1)
Requirement already satisfied: setuptools-scm[toml]<9.0.0,>=8.0.0 in d:\aplicacion
\lib\site-packages (from linearmodels) (8.0.4)
Requirement already satisfied: interface-meta>=1.2.0 in d:\aplicacion\lib\site-pack
ages (from formulaic>=0.6.5->linearmodels) (1.3.0)
Requirement already satisfied: typing-extensions>=4.2.0 in d:\aplicacion\lib\site-p
ackages (from formulaic>=0.6.5->linearmodels) (4.7.1)
Requirement already satisfied: wrapt>=1.0 in d:\aplicacion\lib\site-packages (from
formulaic>=0.6.5->linearmodels) (1.14.1)
Requirement already satisfied: python-dateutil>=2.8.2 in d:\aplicacion\lib\site-pac
kages (from pandas>=1.3.0->linearmodels) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in d:\aplicacion\lib\site-packages (fro
m pandas>=1.3.0->linearmodels) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in d:\aplicacion\lib\site-packages (f
rom pandas>=1.3.0->linearmodels) (2023.3)
Requirement already satisfied: packaging>=20 in d:\aplicacion\lib\site-packages (fr
om setuptools-scm[toml]<9.0.0,>=8.0.0->linearmodels) (23.1)
Requirement already satisfied: setuptools in d:\aplicacion\lib\site-packages (from
setuptools-scm[toml]<9.0.0,>=8.0.0->linearmodels) (68.0.0)
Requirement already satisfied: patsy>=0.5.2 in d:\aplicacion\lib\site-packages (fro
m statsmodels>=0.12.0->linearmodels) (0.5.3)
Requirement already satisfied: six in d:\aplicacion\lib\site-packages (from patsy>=
0.5.2->statsmodels>=0.12.0->linearmodels) (1.16.0)
```

In [197…

```
dfc1=df[["temperatura","precio_combustible","ind_precio_consumidor","tasa_desempleo
X = sm.tools.tools.add_constant(dfc1)
y = df.ventas
```

In [197…

```
y
```

Out[1972]:
```
fecha       nro_tienda
05-02-2010  1             1643690.90
12-02-2010  1             1641957.44
19-02-2010  1             1611968.17
26-02-2010  1             1409727.59
05-03-2010  1             1554806.68
                             ...
28-09-2012  45             713173.95
05-10-2012  45             733455.07
12-10-2012  45             734464.36
19-10-2012  45             718125.53
26-10-2012  45             760281.43
Name: ventas, Length: 5917, dtype: float64
```

In [197…
```
X
```

Out[1973]:

		const	temperatura	precio_combustible	ind_precio_consumidor	tasa_des
fecha	**nro_tienda**					
05-02-2010	**1**	1.0	42.31	2.572	211.096358	
12-02-2010	**1**	1.0	38.51	2.548	211.242170	
19-02-2010	**1**	1.0	39.93	2.514	211.289143	
26-02-2010	**1**	1.0	46.63	2.561	211.319643	
05-03-2010	**1**	1.0	46.50	2.625	211.350143	
…	**…**	…	…	…	…	
28-09-2012	**45**	1.0	64.88	3.997	192.013558	
05-10-2012	**45**	1.0	64.89	3.985	192.170412	
12-10-2012	**45**	1.0	54.47	4.000	192.327265	
19-10-2012	**45**	1.0	56.47	3.969	192.330854	
26-10-2012	**45**	1.0	58.85	3.882	192.308899	

5917 rows × 6 columns

In [197…
```
modelo1 = PooledOLS(y, X)
```

In [197…
```
resultados_pooled_OLS = modelo1.fit(cov_type='clustered', cluster_entity=True)
```

In [197…
```
predicciones_pooled_OLS = resultados_pooled_OLS.predict().fitted_values
```

In [197…
```
residuos_pooled_OLS = resultados_pooled_OLS.resids
resultados_pooled_OLS
```

Out[1977]:

PooledOLS Estimation Summary

Dep. Variable:	ventas	**R-squared:**	0.0165
Estimator:	PooledOLS	**R-squared (Between):**	0.0140
No. Observations:	5917	**R-squared (Within):**	0.0166
Date:	Tue, Mar 19 2024	**R-squared (Overall):**	0.0165
Time:	23:48:15	**Log-likelihood**	-8.658e+04
Cov. Estimator:	Clustered		
		F-statistic:	19.780
Entities:	143	**P-value**	0.0000
Avg Obs:	41.378	**Distribution:**	F(5,5911)
Min Obs:	33.000		
Max Obs:	42.000	**F-statistic (robust):**	233.60
		P-value	0.0000
Time periods:	45	**Distribution:**	F(5,5911)
Avg Obs:	131.49		
Min Obs:	17.000		
Max Obs:	143.00		

Parameter Estimates

	Parameter	Std. Err.	T-stat	P-value	Lower CI	Upper CI
const	1.642e+06	4.46e+04	36.816	0.0000	1.555e+06	1.73e+06
temperatura	-406.53	426.04	-0.9542	0.3400	-1241.7	428.67
precio_combustible	-1478.0	9504.0	-0.1555	0.8764	-2.011e+04	1.715e+04
ind_precio_consumidor	-1429.4	74.153	-19.277	0.0000	-1574.8	-1284.1
tasa_desempleo	-4.23e+04	2495.5	-16.950	0.0000	-4.719e+04	-3.741e+04
feriado	5.042e+04	3.786e+04	1.3316	0.1830	-2.381e+04	1.246e+05

id: 0x165fc0e4d50

De acuerdo al resultado del modelo1 se identifica las variables mas significativas, y se diseña el modelo 2. El r2 es bajo al momento 0.0165.

In [197…

```
dfc2=df[["ind_precio_consumidor","tasa_desempleo"]]
X = sm.tools.tools.add_constant(dfc2)
y = df.ventas
```

In [197…

```
modelo2 = PooledOLS(y, X)
```

In [198…
```
resultados_pooled_OLS_2 = modelo2.fit(cov_type='clustered', cluster_entity=True)
```

In [198…
```
# checking homoskedasticity graphically
predicciones_pooled_OLS_2 = resultados_pooled_OLS_2.predict().fitted_values
```

In [198…
```
residuos_pooled_OLS_2 = resultados_pooled_OLS_2.resids
resultados_pooled_OLS_2
```

Out[1982]:

PooledOLS Estimation Summary

Dep. Variable:	ventas	**R-squared:**	0.0156
Estimator:	PooledOLS	**R-squared (Between):**	-0.0322
No. Observations:	5917	**R-squared (Within):**	0.0166
Date:	Tue, Mar 19 2024	**R-squared (Overall):**	0.0156
Time:	23:48:16	**Log-likelihood**	-8.658e+04
Cov. Estimator:	Clustered		
		F-statistic:	46.964
Entities:	143	**P-value**	0.0000
Avg Obs:	41.378	**Distribution:**	F(2,5914)
Min Obs:	33.000		
Max Obs:	42.000	**F-statistic (robust):**	509.33
		P-value	0.0000
Time periods:	45	**Distribution:**	F(2,5914)
Avg Obs:	131.49		
Min Obs:	17.000		
Max Obs:	143.00		

Parameter Estimates

	Parameter	Std. Err.	T-stat	P-value	Lower CI	Upper CI
const	1.625e+06	2.5e+04	65.020	0.0000	1.576e+06	1.674e+06
ind_precio_consumidor	-1470.6	47.327	-31.073	0.0000	-1563.4	-1377.8
tasa_desempleo	-4.256e+04	2540.9	-16.749	0.0000	-4.754e+04	-3.758e+04

id: 0x165fc6dba50

Una vez ejecutado el modelo final con datos de Panel, con las variables significativas: ind_precio_consumidor y tasa_desempleo, se observa que el r2 sigue siendo bajo 0.0156, NO ES ACEPTABLE EL MODELO.

Por lo que ahora intentaremos con REGRESION LINEAL-Machine Learning el realizar el análisis de la información de Walmart.

2-MODELO REGRESION LINEAL VARIABLE DEPENDIENTE VENTAS

In [198…
```
from sklearn.linear_model import LinearRegression
```

In [198…
```
dfr[['dia', 'mes', 'anio']] = dfr['fecha'].str.split('-', expand=True)
dfr
```

Out[1984]:

	nro_tienda	fecha	ventas	feriado	temperatura	precio_combustible	ind_precio_c
0	1	05-02-2010	1643690.90	0	42.31	2.572	
1	1	12-02-2010	1641957.44	1	38.51	2.548	
2	1	19-02-2010	1611968.17	0	39.93	2.514	
3	1	26-02-2010	1409727.59	0	46.63	2.561	
4	1	05-03-2010	1554806.68	0	46.50	2.625	
…	…	…	…	…	…	…	
6430	45	28-09-2012	713173.95	0	64.88	3.997	
6431	45	05-10-2012	733455.07	0	64.89	3.985	
6432	45	12-10-2012	734464.36	0	54.47	4.000	
6433	45	19-10-2012	718125.53	0	56.47	3.969	
6434	45	26-10-2012	760281.43	0	58.85	3.882	

5917 rows × 11 columns

In [198…
```
var_cuantitativas_r = dfr.select_dtypes('number').columns
var_cualitativas_r  =dfr.select_dtypes('object').columns
print("Variables cuantitativas:", var_cuantitativas_r )
print("Variables cualitativas", var_cualitativas_r )
```

```
Variables cuantitativas: Index(['nro_tienda', 'ventas', 'feriado', 'temperatura',
'precio_combustible',
       'ind_precio_consumidor', 'tasa_desempleo'],
      dtype='object')
Variables cualitativas Index(['fecha', 'dia', 'mes', 'anio'], dtype='object')
```

In [198…
```
from sklearn.preprocessing import LabelEncoder
```

In [198…
```
labelencoder = LabelEncoder()
```

In [198…
```
dfr[var_cualitativas_r] = dfr[var_cualitativas_r].apply(labelencoder.fit_transform)
```

In [198…
```
dfr.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 5917 entries, 0 to 6434
Data columns (total 11 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   nro_tienda             5917 non-null   int64
 1   fecha                  5917 non-null   int32
 2   ventas                 5917 non-null   float64
 3   feriado                5917 non-null   int64
 4   temperatura            5917 non-null   float64
 5   precio_combustible     5917 non-null   float64
 6   ind_precio_consumidor  5917 non-null   float64
 7   tasa_desempleo         5917 non-null   float64
 8   dia                    5917 non-null   int32
 9   mes                    5917 non-null   int32
 10  anio                   5917 non-null   int32
dtypes: float64(5), int32(4), int64(2)
memory usage: 462.3 KB
```

In [199…
```
Xrl = dfr[["nro_tienda","ind_precio_consumidor","precio_combustible","temperatura"]
yrl = dfr.ventas
```

In [199…
```
dfr.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 5917 entries, 0 to 6434
Data columns (total 11 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   nro_tienda             5917 non-null   int64
 1   fecha                  5917 non-null   int32
 2   ventas                 5917 non-null   float64
 3   feriado                5917 non-null   int64
 4   temperatura            5917 non-null   float64
 5   precio_combustible     5917 non-null   float64
 6   ind_precio_consumidor  5917 non-null   float64
 7   tasa_desempleo         5917 non-null   float64
 8   dia                    5917 non-null   int32
 9   mes                    5917 non-null   int32
 10  anio                   5917 non-null   int32
dtypes: float64(5), int32(4), int64(2)
memory usage: 462.3 KB
```

In [199…
```
from sklearn.model_selection import train_test_split
```

In [199…
```
Xrl_train , Xrl_test , yrl_train , yrl_test = train_test_split(Xrl , yrl , test_siz
```

In [199…
```
print(Xrl_train.shape,"",type(Xrl_train))
print(yrl_train.shape,"\t ",type(yrl_train))
print(Xrl_test.shape,"",type(Xrl_test))
print(yrl_test.shape,"\t ",type(yrl_test))
```

```
(4733, 4)  <class 'pandas.core.frame.DataFrame'>
(4733,)          <class 'pandas.core.series.Series'>
(1184, 4)  <class 'pandas.core.frame.DataFrame'>
(1184,)          <class 'pandas.core.series.Series'>
```

In [199…
```
modelo_regresion = LinearRegression()
modelo_regresion.fit(Xrl_train, yrl_train)
```

Out[1995]:
```
▾ LinearRegression

LinearRegression()
```

In [199…
```
predicciones_train_rl = modelo_regresion.predict(Xrl_train)
predicciones_test_rl = modelo_regresion.predict(Xrl_test)
```

In [199…
```
from sklearn.metrics import r2_score
```

In [199…
```
r_square_train = r2_score(yrl_train, predicciones_train_rl)
r_square_test  = r2_score(yrl_test, predicciones_test_rl)
print('El R^2 del subconjunto de entrenamiento es:' , r_square_train)
print('El R^2 del subconjunto de prueba es:' , r_square_test)
```

```
El R^2 del subconjunto de entrenamiento es: 0.12423530186261478
El R^2 del subconjunto de prueba es: 0.13183381206206424
```

CON REGRESION LINEAL tampoco obtenemos un r2 aceptable ya que es menor a 0.50, NO ES ACEPTABLE EL MODELO

3_MODELO DE DATOS DE PANEL CON VARIABLE DEPENDIENTE INDICE DE PRECIO AL CONSUMIDOR

In [199…
```
dfp=df[["temperatura","precio_combustible","ventas","tasa_desempleo","feriado"]]
Xp= sm.tools.tools.add_constant(dfp)
yp= df.ind_precio_consumidor
```

In [200…
```
modelop1 = PooledOLS(yp, Xp)
resultados_pooled_OLSp = modelop1.fit(cov_type='clustered', cluster_entity=True)
predicciones_pooled_OLSp = resultados_pooled_OLSp.predict().fitted_values
residuos_pooled_OLSp = resultados_pooled_OLSp.resids
resultados_pooled_OLSp
```

Out[2000]:

PooledOLS Estimation Summary

Dep. Variable:	ind_precio_consumidor	**R-squared:**	0.1477
Estimator:	PooledOLS	**R-squared (Between):**	-7.1192
No. Observations:	5917	**R-squared (Within):**	0.2108
Date:	Tue, Mar 19 2024	**R-squared (Overall):**	0.1477
Time:	23:48:16	**Log-likelihood**	-2.96e+04
Cov. Estimator:	Clustered		
		F-statistic:	204.87
Entities:	143	**P-value**	0.0000
Avg Obs:	41.378	**Distribution:**	F(5,5911)
Min Obs:	33.000		
Max Obs:	42.000	**F-statistic (robust):**	345.21
		P-value	0.0000
Time periods:	45	**Distribution:**	F(5,5911)
Avg Obs:	131.49		
Min Obs:	17.000		
Max Obs:	143.00		

Parameter Estimates

	Parameter	Std. Err.	T-stat	P-value	Lower CI	Upper CI
const	268.01	6.0581	44.240	0.0000	256.14	279.89
temperatura	0.5402	0.0377	14.327	0.0000	0.4663	0.6141
precio_combustible	-17.390	1.9257	-9.0304	0.0000	-21.165	-13.615
ventas	-6.192e-06	3.879e-07	-15.961	0.0000	-6.952e-06	-5.431e-06
tasa_desempleo	-7.9526	0.3732	-21.311	0.0000	-8.6842	-7.2211
feriado	4.4406	3.1639	1.4035	0.1605	-1.7618	10.643

id: 0x165fc6fbed0

De acuerdo al resultado del modelop1 se identifica que las variables seleccionadas tienen un p-value < 0.05 (a priori todas son significativas), y a continuación se diseña el modelo 2 sin la variable feriado que según p.value no es significativa.

In [200…

```
dfp2=df[["temperatura","precio_combustible","ventas","tasa_desempleo"]]
Xp = sm.tools.tools.add_constant(dfp2)
yp = df.ind_precio_consumidor
```

In [200…

```
modelop2 = PooledOLS(yp, Xp)
resultados_pooled_OLS_2p = modelop2.fit(cov_type='clustered', cluster_entity=True)
predicciones_pooled_OLS_2p = resultados_pooled_OLS_2p.predict().fitted_values
residuos_pooled_OLS_2p = resultados_pooled_OLS_2p.resids
resultados_pooled_OLS_2p
```

Out[2002]:

PooledOLS Estimation Summary

Dep. Variable:	ind_precio_consumidor	**R-squared:**	0.1469
Estimator:	PooledOLS	**R-squared (Between):**	-7.1383
No. Observations:	5917	**R-squared (Within):**	0.2102
Date:	Tue, Mar 19 2024	**R-squared (Overall):**	0.1469
Time:	23:48:16	**Log-likelihood**	-2.961e+04
Cov. Estimator:	Clustered		
		F-statistic:	254.49
Entities:	143	**P-value**	0.0000
Avg Obs:	41.378	**Distribution:**	F(4,5912)
Min Obs:	33.000		
Max Obs:	42.000	**F-statistic (robust):**	421.68
		P-value	0.0000
Time periods:	45	**Distribution:**	F(4,5912)
Avg Obs:	131.49		
Min Obs:	17.000		
Max Obs:	143.00		

Parameter Estimates

	Parameter	Std. Err.	T-stat	P-value	Lower CI	Upper CI
const	269.20	5.9859	44.973	0.0000	257.47	280.93
temperatura	0.5310	0.0373	14.230	0.0000	0.4579	0.6042
precio_combustible	-17.523	1.9201	-9.1261	0.0000	-21.287	-13.759
ventas	-6.151e-06	3.953e-07	-15.560	0.0000	-6.926e-06	-5.376e-06
tasa_desempleo	-7.9432	0.3732	-21.284	0.0000	-8.6748	-7.2116

id: 0x165fc76dcd0

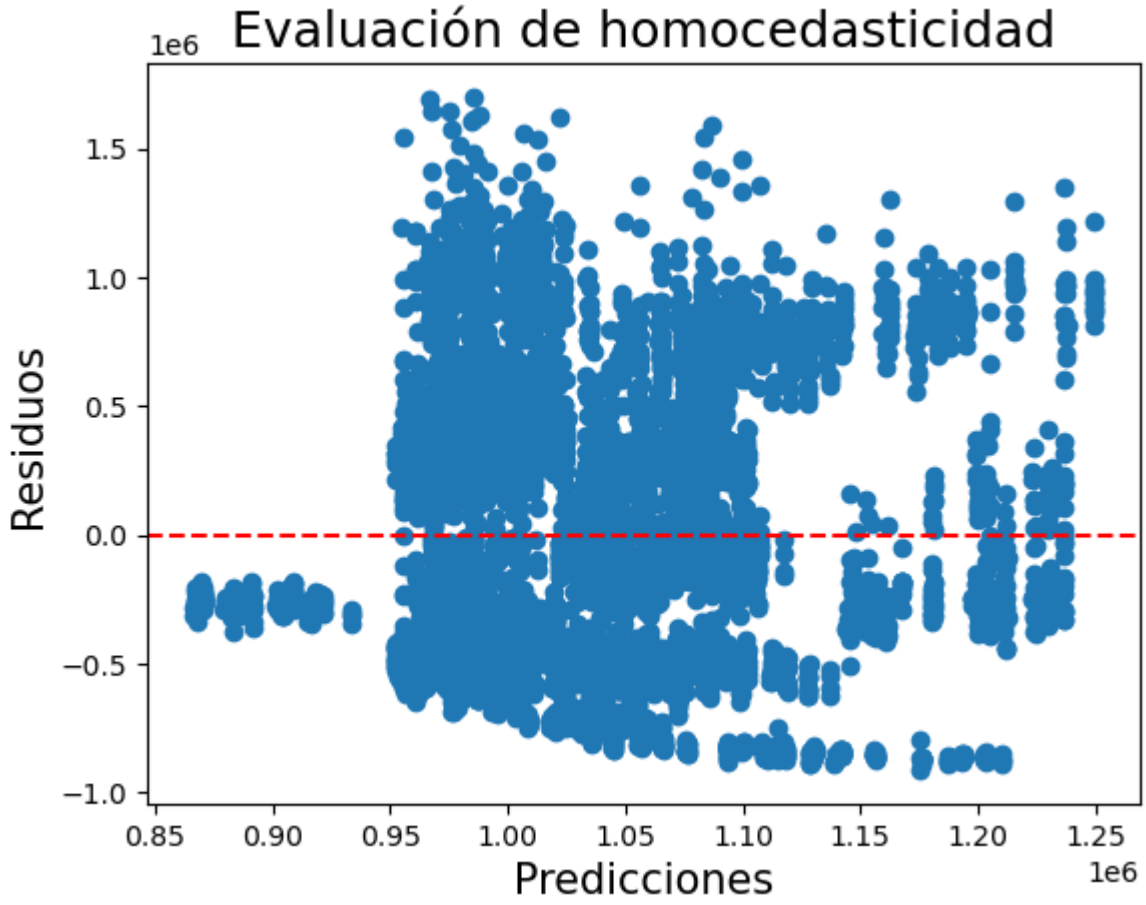
9. Verifique los supuestos, de haber escogido el enfoque econométrico

SUPUESTOS DE DATOS DE PANEL CON VARIABLE DEPENDIENTE VENTAS

Homocedasticidad

Para validar este supuesto, primero graficaremos los residuos y validaremos la prueba gráfica con el test estadístico de Breusch-Pagan.

In [200…
```
fig, ax = plt.subplots()
ax.scatter(predicciones_pooled_OLS_2, residuos_pooled_OLS_2)
ax.axhline(0, color = 'r', ls = '--')
ax.set_xlabel('Predicciones', fontsize = 15)
ax.set_ylabel('Residuos', fontsize = 15)
ax.set_title('Evaluación de homocedasticidad', fontsize = 18)
plt.show()
```



Comprobemos esta intuición gráfcia con el test de Breusch-Pagan:

In [200…
```
from statsmodels.stats.diagnostic import het_breuschpagan
```

In [200…
```
pooled_OLS_df_2 = pd.concat([df, residuos_pooled_OLS_2], axis=1)
```

In [200…
```
pooled_OLS_df_2 = pooled_OLS_df_2.drop(['fecha'], axis = 1).fillna(0)
```

In [200…
```
X_ = sm.tools.tools.add_constant(dfc1).fillna(0)
```

In [200…
```
pooled_OLS_df_2
```

Out[2008]:

fecha	nro_tienda	ventas	feriado	temperatura	precio_combustible	ind_precio_consum
05-02-2010	**1**	1643690.90	0	42.31	2.572	211.096
12-02-2010	**1**	1641957.44	1	38.51	2.548	211.242
19-02-2010	**1**	1611968.17	0	39.93	2.514	211.289
26-02-2010	**1**	1409727.59	0	46.63	2.561	211.319
05-03-2010	**1**	1554806.68	0	46.50	2.625	211.350
...	**...**	...	...	...	...	
28-09-2012	**45**	713173.95	0	64.88	3.997	192.013
05-10-2012	**45**	733455.07	0	64.89	3.985	192.170
12-10-2012	**45**	734464.36	0	54.47	4.000	192.327
19-10-2012	**45**	718125.53	0	56.47	3.969	192.330
26-10-2012	**45**	760281.43	0	58.85	3.882	192.308

5917 rows × 7 columns

In [200…

```
breusch_pagan = het_breuschpagan(pooled_OLS_df_2.residual, X_)
labels = ['LM-Stat', 'LM p-val', 'F-Stat', 'F p-val']
print(dict(zip(labels, breusch_pagan)))
```

```
{'LM-Stat': 113.22834242120994, 'LM p-val': 8.511074831773667e-23, 'F-Stat': 23.064
061494486413, 'F p-val': 5.2031365818457886e-23}
```

Como el p-valor del estadístico de Breush-Pagan es < 0.05 (8.511074831773667e-23), entonces se rechaza la hipótesis nula, y por lo tanto estamos en la presencia de heterocedasticidad.

```
Ho:Las varianzas del error son iguales, hay presencia de
homocedasticidad
Hi:Las varianzas del error no son iguales, hay presencia de
heterocedasticidad
```

No-autocorrelación

In [201…

```
from statsmodels.stats.stattools import durbin_watson
```

In [201…

```
durbin_watson = durbin_watson(pooled_OLS_df_2.residual)
print(durbin_watson)
```

```
0.08702185504450755
```

Según la prueba de Durbin Watson se puede concluir que existe una correlación positiva de 0.087 entre las variables seleccionadas.

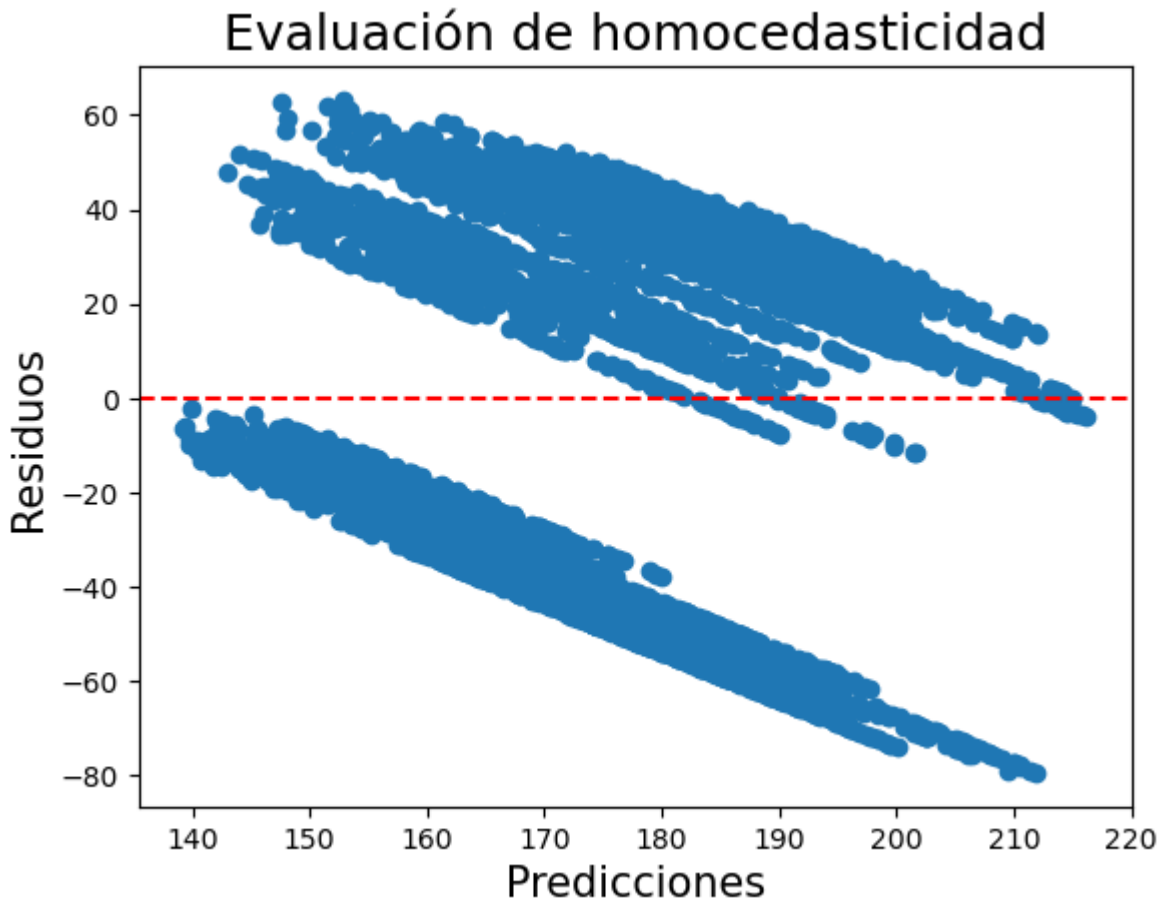
- 0 - 2 significa una autocorrelación positiva (cuanto más cerca de cero, mayor es la correlación)

SUPUESTOS DE DATOS DE PANEL CON VARIABLE DEPENDIENTE INDICE DE PRECIO AL CONSUMIDOR

Homocedasticidad

In [201…

```
fig, ax = plt.subplots()
ax.scatter(predicciones_pooled_OLS_2p, residuos_pooled_OLS_2p)
ax.axhline(0, color = 'r', ls = '--')
ax.set_xlabel('Predicciones', fontsize = 15)
ax.set_ylabel('Residuos', fontsize = 15)
ax.set_title('Evaluación de homocedasticidad', fontsize = 18)
plt.show()
```



In [201…

```
from statsmodels.stats.diagnostic import het_breuschpagan
```

In [201…

```
pooled_OLS_df_2p = pd.concat([df, residuos_pooled_OLS_2p], axis=1)
pooled_OLS_df_2p = pooled_OLS_df_2p.drop(['fecha'], axis = 1).fillna(0)
X_ = sm.tools.tools.add_constant(dfp2).fillna(0)
```

In [201…]
```
pooled_OLS_df_2p
```

Out[2015]:

fecha	nro_tienda	ventas	feriado	temperatura	precio_combustible	ind_precio_consum
05-02-2010	**1**	1643690.90	0	42.31	2.572	211.096
12-02-2010	**1**	1641957.44	1	38.51	2.548	211.242
19-02-2010	**1**	1611968.17	0	39.93	2.514	211.289
26-02-2010	**1**	1409727.59	0	46.63	2.561	211.319
05-03-2010	**1**	1554806.68	0	46.50	2.625	211.350
…	**…**	…	…	…	…	
28-09-2012	**45**	713173.95	0	64.88	3.997	192.013
05-10-2012	**45**	733455.07	0	64.89	3.985	192.170
12-10-2012	**45**	734464.36	0	54.47	4.000	192.327
19-10-2012	**45**	718125.53	0	56.47	3.969	192.330
26-10-2012	**45**	760281.43	0	58.85	3.882	192.308

5917 rows × 7 columns

In [201…]
```
breusch_pagan = het_breuschpagan(pooled_OLS_df_2p.residual, X_)
labels = ['LM-Stat', 'LM p-val', 'F-Stat', 'F p-val']
print(dict(zip(labels, breusch_pagan)))
```

```
{'LM-Stat': 425.4243328099318, 'LM p-val': 8.914749483575605e-91, 'F-Stat': 114.498
49769889663, 'F p-val': 3.4791595375803306e-94}
```

Como el p-valor del estadístico de Breush-Pagan es < 0.05 (8.914749483575605e-91), entonces se rechaza la hipótesis nula, y por lo tanto estamos en la presencia de heterocedasticidad.

No-autocorrelación

In [201…]
```
from statsmodels.stats.stattools import durbin_watson
```

In [201…]
```
durbin_watson = durbin_watson(pooled_OLS_df_2p.residual)
print(durbin_watson)
```

```
0.023139144128356635
```

Según la prueba de Durbin Watson se puede concluir que existe una correlación positiva de 0.023 entre las variables seleccionadas.

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

DATOS DE PANEL CON VARIABLE DEPENDIENTE VENTAS

Para identificar el modelo definitivo es necesario realizar los modelos:

Modelo de Efectos Fijos "Fixed effects"

In [201…
```
from linearmodels import PanelOLS
```

In [202…
```
modelo_fe = PanelOLS(y, X, entity_effects = True)
resultados_fe = modelo_fe.fit()
```

In [202…
```
resultados_fe
```

Out[2021]:

PanelOLS Estimation Summary

Dep. Variable:	ventas	**R-squared:**	0.0167
Estimator:	PanelOLS	**R-squared (Between):**	-0.0353
No. Observations:	5917	**R-squared (Within):**	0.0167
Date:	Tue, Mar 19 2024	**R-squared (Overall):**	0.0156
Time:	23:48:18	**Log-likelihood**	-8.651e+04
Cov. Estimator:	Unadjusted		
		F-statistic:	48.877
Entities:	143	**P-value**	0.0000
Avg Obs:	41.378	**Distribution:**	F(2,5772)
Min Obs:	33.000		
Max Obs:	42.000	**F-statistic (robust):**	48.877
		P-value	0.0000
Time periods:	45	**Distribution:**	F(2,5772)
Avg Obs:	131.49		
Min Obs:	17.000		
Max Obs:	143.00		

Parameter Estimates

	Parameter	Std. Err.	T-stat	P-value	Lower CI	Upper CI
const	1.647e+06	6.281e+04	26.217	0.0000	1.524e+06	1.77e+06
ind_precio_consumidor	-1477.0	187.39	-7.8819	0.0000	-1844.3	-1109.6
tasa_desempleo	-4.517e+04	6063.4	-7.4499	0.0000	-5.706e+04	-3.329e+04

F-test for Poolability: 0.9123
P-value: 0.7626
Distribution: F(142,5772)

Included effects: Entity

id: 0x165fc2ecd50

Modelo de Efectos Aleatoreos "RandomEffects"

In [202…

```
from linearmodels import RandomEffects
```

In [202…

```
modelo_re = RandomEffects(y, X)
resultados_re = modelo_re.fit()
```

In [202…

```
resultados_re
```

Out[2024]:

RandomEffects Estimation Summary

Dep. Variable:	ventas	**R-squared:**	0.0156
Estimator:	RandomEffects	**R-squared (Between):**	-0.0322
No. Observations:	5917	**R-squared (Within):**	0.0166
Date:	Tue, Mar 19 2024	**R-squared (Overall):**	0.0156
Time:	23:48:18	**Log-likelihood**	-8.658e+04
Cov. Estimator:	Unadjusted		
		F-statistic:	46.964
Entities:	143	**P-value**	0.0000
Avg Obs:	41.378	**Distribution:**	F(2,5914)
Min Obs:	33.000		
Max Obs:	42.000	**F-statistic (robust):**	46.964
		P-value	0.0000
Time periods:	45	**Distribution:**	F(2,5914)
Avg Obs:	131.49		
Min Obs:	17.000		
Max Obs:	143.00		

Parameter Estimates

	Parameter	Std. Err.	T-stat	P-value	Lower CI	Upper CI
const	1.625e+06	6.186e+04	26.274	0.0000	1.504e+06	1.747e+06
ind_precio_consumidor	-1470.6	187.02	-7.8633	0.0000	-1837.2	-1104.0
tasa_desempleo	-4.256e+04	5869.7	-7.2501	0.0000	-5.406e+04	-3.105e+04

id: 0x165fc32dfd0

Test de Haussman

Como ambos modelos devuelven resultados similares, para seleccionar el modelo, aplicaremos el test de Hausman.

Con las siguientes hipótesis:

- Ho: El modelo preferido es el de efectos aleatorios
- Hi: El modelo preferido es el de efectos fijos

In [202…
```
import numpy.linalg as la
from scipy import stats
import numpy as np
```

In [202…
```
def hausman(fe, re):
    b = fe.params
    B = re.params
    v_b = fe.cov
    v_B = re.cov
    df = b[np.abs(b) < 1e8].size
    chi2 = np.dot((b - B).T, la.inv(v_b - v_B).dot(b - B))

    pval = stats.chi2.sf(chi2, df)
    return chi2, df, pval
```

In [202…
```
hausman = hausman(resultados_fe, resultados_re)
```

In [202…
```
print('chi-Squared: ' + str(hausman[0]))
print('degrees of freedom: ' + str(hausman[1]))
print('p-Value:' + str(hausman[2]))
```

```
chi-Squared: 5.563096942504234
degrees of freedom: 3
p-Value:0.13491305222506733
```

Considerando que p-valor es (0.13491305222506733). En consecuencia, el modelo de efectos randómicos parece ser el más adecuado.

DATOS DE PANEL CON VARIABLE DEPENDIENTE INDICE DE PRECIO AL CONSUMIDOR

Modelo de Efectos Fijos

In [202…
```
from linearmodels import PanelOLS
```

In [203…
```
modelo_fep = PanelOLS(yp, Xp, entity_effects = True)
resultados_fep = modelo_fep.fit()
```

In [203…
```
resultados_fep
```

Out[2031]:

PanelOLS Estimation Summary

Dep. Variable:	ind_precio_consumidor	**R-squared:**	0.4686
Estimator:	PanelOLS	**R-squared (Between):**	-264.08
No. Observations:	5917	**R-squared (Within):**	0.4686
Date:	Tue, Mar 19 2024	**R-squared (Overall):**	-1.8229
Time:	23:48:18	**Log-likelihood**	-2.818e+04
Cov. Estimator:	Unadjusted		
		F-statistic:	1272.0
Entities:	143	**P-value**	0.0000
Avg Obs:	41.378	**Distribution:**	F(4,5770)
Min Obs:	33.000		
Max Obs:	42.000	**F-statistic (robust):**	1272.0
		P-value	0.0000
Time periods:	45	**Distribution:**	F(4,5770)
Avg Obs:	131.49		
Min Obs:	17.000		
Max Obs:	143.00		

Parameter Estimates

	Parameter	Std. Err.	T-stat	P-value	Lower CI	Upper CI
const	585.92	8.6096	68.054	0.0000	569.04	602.80
temperatura	1.0877	0.0363	29.981	0.0000	1.0166	1.1589
precio_combustible	-135.86	2.5182	-53.952	0.0000	-140.80	-130.92
ventas	-1.769e-06	6.9e-07	-2.5633	0.0104	-3.121e-06	-4.16e-07
tasa_desempleo	-2.7131	0.3274	-8.2873	0.0000	-3.3549	-2.0713

F-test for Poolability: 25.161
P-value: 0.0000
Distribution: F(142,5770)

Included effects: Entity

id: 0x165f4ce0ad0

Modelo de Efectos Aleatoreos "RandomEffects"

In [203…

```
modelo_rep = RandomEffects(yp, Xp)
resultados_rep = modelo_rep.fit()
resultados_rep
```

Out[2032]:

RandomEffects Estimation Summary

Dep. Variable:	ind_precio_consumidor	**R-squared:**	0.1469
Estimator:	RandomEffects	**R-squared (Between):**	-7.1383
No. Observations:	5917	**R-squared (Within):**	0.2102
Date:	Tue, Mar 19 2024	**R-squared (Overall):**	0.1469
Time:	23:48:19	**Log-likelihood**	-2.961e+04
Cov. Estimator:	Unadjusted		
		F-statistic:	254.49
Entities:	143	**P-value**	0.0000
Avg Obs:	41.378	**Distribution:**	F(4,5912)
Min Obs:	33.000		
Max Obs:	42.000	**F-statistic (robust):**	254.49
		P-value	0.0000
Time periods:	45	**Distribution:**	F(4,5912)
Avg Obs:	131.49		
Min Obs:	17.000		
Max Obs:	143.00		

Parameter Estimates

	Parameter	Std. Err.	T-stat	P-value	Lower CI	Upper CI
const	269.20	4.9769	54.090	0.0000	259.44	278.96
temperatura	0.5310	0.0258	20.574	0.0000	0.4804	0.5816
precio_combustible	-17.523	1.0405	-16.841	0.0000	-19.563	-15.483
ventas	-6.151e-06	8.524e-07	-7.2160	0.0000	-7.822e-06	-4.48e-06
tasa_desempleo	-7.9432	0.3804	-20.883	0.0000	-8.6889	-7.1976

id: 0x165fc206950

Test de Haussman

In [203…
```python
def hausman(fep, rep):
    b = fep.params
    B = rep.params
    v_b = fep.cov
    v_B = rep.cov
    df = b[np.abs(b) < 1e8].size
    chi2 = np.dot((b - B).T, la.inv(v_b - v_B).dot(b - B))

    pval = stats.chi2.sf(chi2, df)
    return chi2, df, pval
```

In [203…
```python
hausman = hausman(resultados_fep, resultados_rep)
```

In [203…
```python
print('chi-Squared: ' + str(hausman[0]))
print('degrees of freedom: ' + str(hausman[1]))
print('p-Value:' + str(hausman[2]))
```

```
chi-Squared: 4432.635586560019
degrees of freedom: 5
p-Value:0.0
```

Considerando que p-valor es 0. En consecuencia, el modelo de efectos fijos parece ser el más adecuado.

11. Grafique a los valores predicho de modelo vs los valores reales. ¿Cómo se ven una vez graficados frente a los valores reales? Argumente su respuesta.

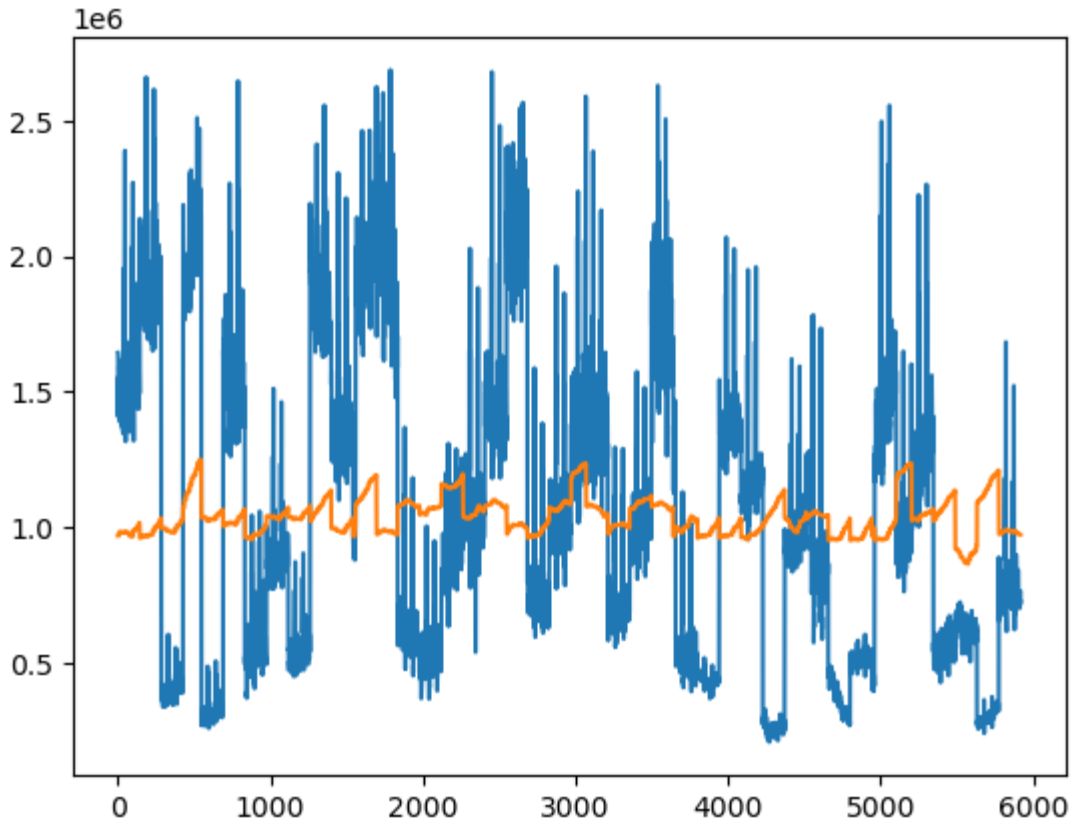
MODELO DE DATOS DE PANEL CON VARIABLE DEPENDIENTE VENTAS

In [203…
```python
from sklearn.model_selection import train_test_split
```

In [203…
```python
#Datos Observados
X = df[df.columns.difference(['ventas'])]
y = df.ventas
```

In [203…
```python
predicciones_re = resultados_re.predict()
fig, ax = plt.subplots()
ax.plot(y.values)
ax.plot(predicciones_re.values)
```

Out[2038]:
```
[<matplotlib.lines.Line2D at 0x165fc0c8590>]
```

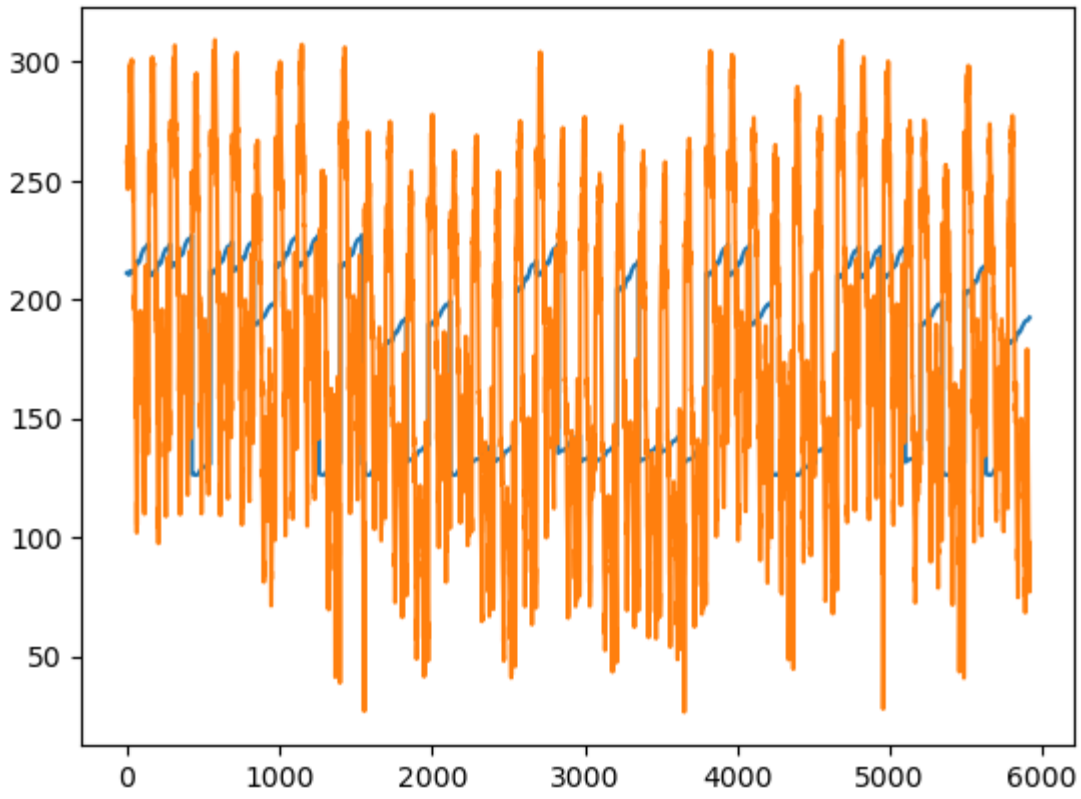


MODELO DE DATOS DE PANEL CON VARIABLE DEPENDIENTE INDICE DE PRECIO CONSUMIDOR

In [203…
```
#Datos Observados
X = df[df.columns.difference(['ind_precio_consumidor'])]
y = df.ind_precio_consumidor
```

In [204…
```
predicciones_fep = resultados_fep.predict()
fig, ax = plt.subplots()
ax.plot(y.values)
ax.plot(predicciones_fep.values)
```

Out[2040]:
```
[<matplotlib.lines.Line2D at 0x165fc1f6450>]
```



12.-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.

Conclusiones

Inicialmente se escoge el modelo de panel de control (enfoque econométrico), ya que se observo variables como fecha y variable de grupo como es el número de tienda o almacen, que indicaba a priori que las ventas era la variable dependiente a predecir.

- Variables con Mayor Significancia

Entre variables que se revisan su importancia, encontramos estas: ind_precio_consumidor, tasa_desempleo, precio_combustible y temperatura.

- Modelamientos

Sin embargo luego de obtener el análisis en el primer modelo de datos de panel con variable dependiente ventas, obtuvimos un r2 muy pequeño por lo que se hizo un par de análisis más para tratar de obtener mejores resultados en el modelamiento.

En el segundo modelo de regresion lineal-machine learning, no se pudo obtener un r cuadrado significativo que de como aceptable al modelo.

En el tercer modelo de datos de panel, tratamos de obtener un mejor r cuadrado con la variable dependiente indice de precio al consumidor, si se consiguió incrementar el r cuadrado, sin embargo al ser R-squared:0.4686 menor a 0.50 tampoco se considera un modelo aceptable.

Se llega a la conclusión que para realizar un análisis más detallado tal vez es necesario incluir más variables o factores específicos que contribuyan a un mejor resultado que permitan predecir de mejor las ventas.

- Nota

Se investigo este modelo un poco más a fondo, llegando a obtener un input, de que el modelamiento adecuado para esta data teniendo como variable dependiente las ventas, el modelo más acertado es el Random Forest, el cual no logramos ver en clase por lo que no se pudo desarrollar el mismo en la tarea planteada.