MAESTRÍA EN INTELIGENCIA DE NEGOCIOS Y CIENCIA DE DATOS

Analítica predictiva

TEMA:

S4- CASO PRACTICO-BDD FINAL

DOCENTE:

Andrea Escobar García PHD, MSc.

AUTORES:

Nathaly Vanessa Espinosa Quijano

Miguel Sebastián Valenzuela Sánchez

Álvaro Paul Angamarca Pinos

Adrián Alejandro Vacacela Baquero

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

```
df=pd.read_csv("/content/drive/MyDrive/CLASE MASTER/Walmart(1).csv")
df
```

	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
...	...	...	...	...	...	...	...	...
6430	45	28-09-2012	713173.95	0	64.88	3.997	192.013558	8.684
6431	45	05-10-2012	733455.07	0	64.89	3.985	192.170412	8.667
6432	45	12-10-2012	734464.36	0	54.47	4.000	192.327265	8.667
6433	45	19-10-2012	718125.53	0	56.47	3.969	192.330854	8.667
6434	45	26-10-2012	760281.43	0	58.85	3.882	192.308899	8.667

Cambiamos la variable Data a tipo fecha:

```
df['Date'] = pd.to_datetime(df['Date'], format='%d-%m-%Y')
```

Cambiamos las variables Store y Holiday_Flag a tipo objeto:

```
df['Store'] = df['Store'].astype('object')
```

```
df['Holiday_Flag'] = df['Holiday_Flag'].astype('object')
```

Quedando de la siguiente manera los tipos de variables:

```
df.dtypes
```

```
Store                 object
Date          datetime64[ns]
Weekly_Sales         float64
Holiday_Flag          object
Temperature          float64
Fuel_Price           float64
CPI                  float64
Unemployment         float64
dtype: object
```

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

```
df.shape
```

```
(6435, 8)
```

La base de datos contiene 6435 registros con un total de 8 variables, las cuales se describen a continuación:

```
df.describe()
```

	Date	Weekly_Sales	Temperature	Fuel_Price	CPI	Unemployment
count	6435	6.435000e+03	6435.000000	6435.000000	6435.000000	6435.000000
mean	2011-06-17 00:00:00	1.046965e+06	60.663782	3.358607	171.578394	7.999151
min	2010-02-05 00:00:00	2.099862e+05	-2.060000	2.472000	126.064000	3.879000
25%	2010-10-08 00:00:00	5.533501e+05	47.460000	2.933000	131.735000	6.891000
50%	2011-06-17 00:00:00	9.607460e+05	62.670000	3.445000	182.616521	7.874000
75%	2012-02-24 00:00:00	1.420159e+06	74.940000	3.735000	212.743293	8.622000
max	2012-10-26 00:00:00	3.818686e+06	100.140000	4.468000	227.232807	14.313000
std	NaN	5.643666e+05	18.444933	0.459020	39.356712	1.875885

Las ventas semanales para Walmart se produjeron durante un período de tres años (2010-2012) en 45 tiendas, presentando ventas semanales máximas de 3,8 millones de dólares y el día más caluroso fue de 100°F.

Dividimos la variable Date en dos para poder analizar los datos por año y mes.

```
df['year'], df['month'] = df['Date'].dt.year, df['Date'].dt.month
df.head()
```

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment	year	month
0	1	2010-02-05	1643690.90	0	42.31	2.572	211.096358	8.106	2010	2
1	1	2010-02-12	1641957.44	1	38.51	2.548	211.242170	8.106	2010	2
2	1	2010-02-19	1611968.17	0	39.93	2.514	211.289143	8.106	2010	2
3	1	2010-02-26	1409727.59	0	46.63	2.561	211.319643	8.106	2010	2
4	1	2010-03-05	1554806.68	0	46.50	2.625	211.350143	8.106	2010	3

Los tipos de variables quedan de la siguiente manera:

```
var_cuantitativas = df.select_dtypes('number').columns
var_cualitativas  =df.select_dtypes('object').columns
```

```
var_cualitativas
```

```
Index(['Store', 'Holiday_Flag', 'year', 'month'], dtype='object')
```

```
var_cuantitativas
```

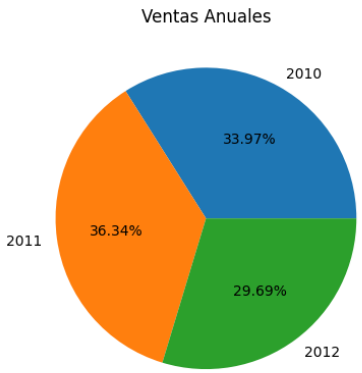
```
Index(['Weekly_Sales', 'Temperature', 'Fuel_Price', 'CPI', 'Unemployment'], dtype='object')
```

Al realizar un análisis de las ventas consolidadas anuales, se identificó que el año con mayor aporte fue 2011, con un porcentaje de 36.34%.

```
df.groupby('year')['Weekly_Sales'].sum()
```

```
year
2010    2.288886e+09
2011    2.448200e+09
2012    2.000133e+09
Name: Weekly_Sales, dtype: float64
```

```
plt.pie(df.groupby('year')['Weekly_Sales'].sum(),labels=df['year'].unique(),normalize=True,autopct='%1.2f%%')
plt.title('Ventas Anuales')
```



Al realizar un análisis de las ventas consolidadas mensuales, se identificó que el mes con mayor número de ventas es abril.

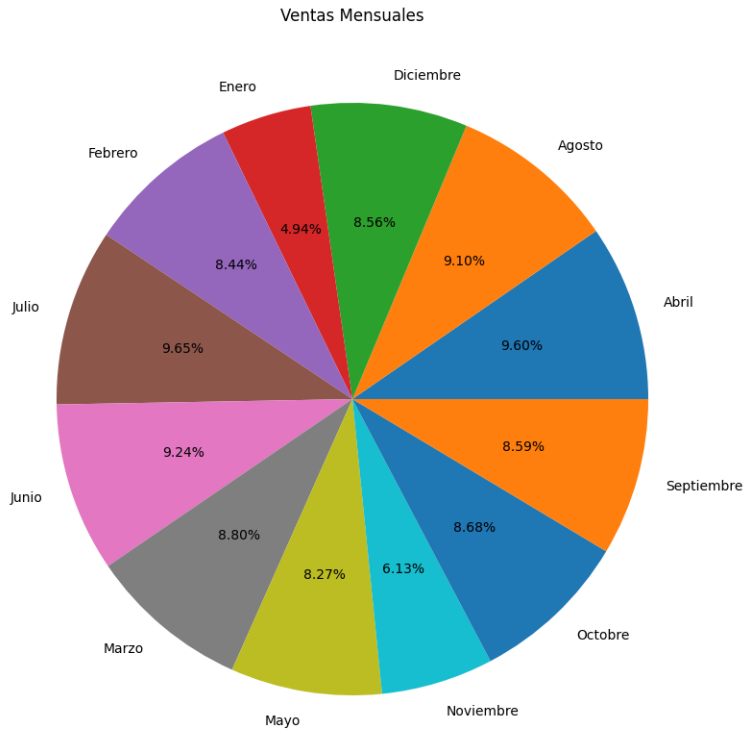
```
df3 = df.groupby('month')['Weekly_Sales'].sum().reset_index()

df3.head()
```

	month	Weekly_Sales
0	Abril	6.468598e+08
1	Agosto	6.130902e+08
2	Diciembre	5.768386e+08
3	Enero	3.325984e+08
4	Febrero	5.687279e+08

os siguientes: Generar código con df3 Ver gráficos recomendados New interactive sheet

```
plt.figure(figsize=(10,10))
plt.pie(df3['Weekly_Sales'],labels=df3['month'],normalize=True,autopct='%1.2f%%')
plt.title('Ventas Mensuales')
```



3. **Evalúe si la base contiene datos perdidos.**

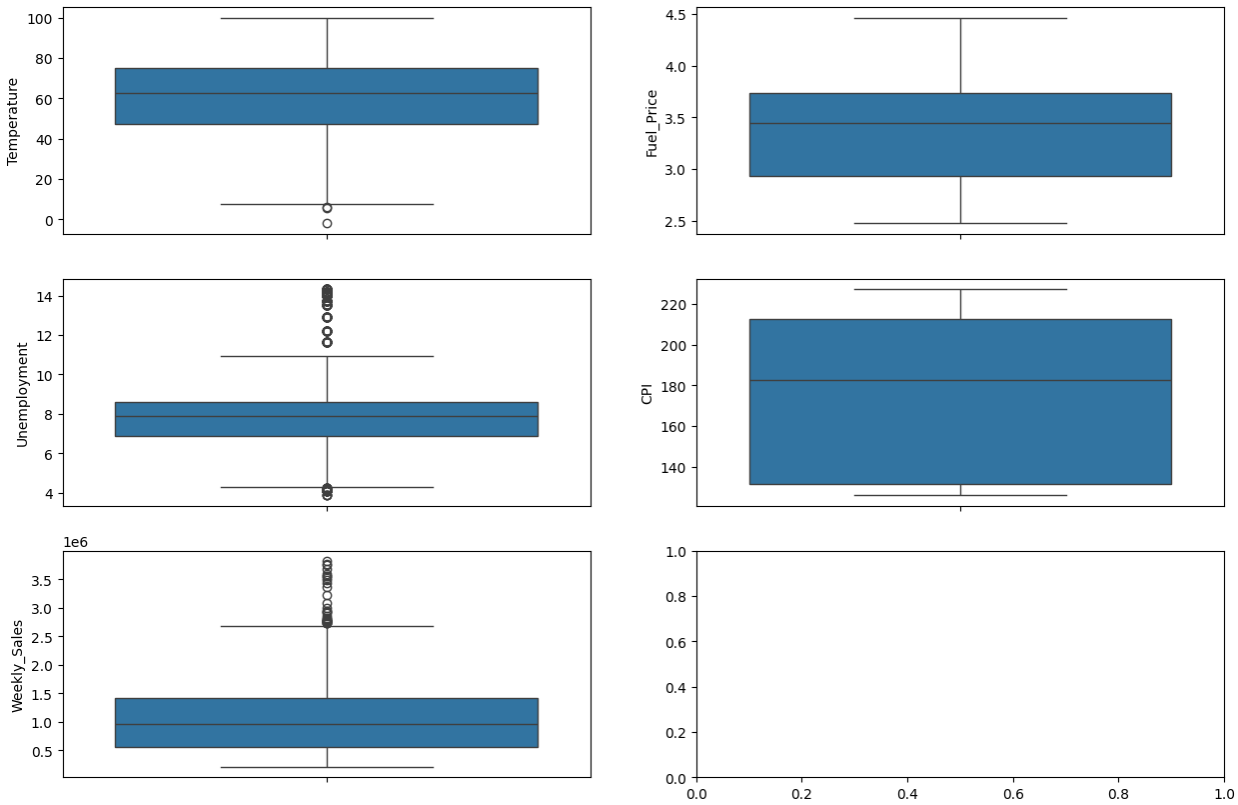
   Base de datos de Walmart no presenta valores perdidos.

```
df.isna().sum()
```

```
Store           0
Date            0
Weekly_Sales    0
Holiday_Flag    0
Temperature     0
Fuel_Price      0
CPI             0
Unemployment    0
year            0
month           0
dtype: int64
```

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

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



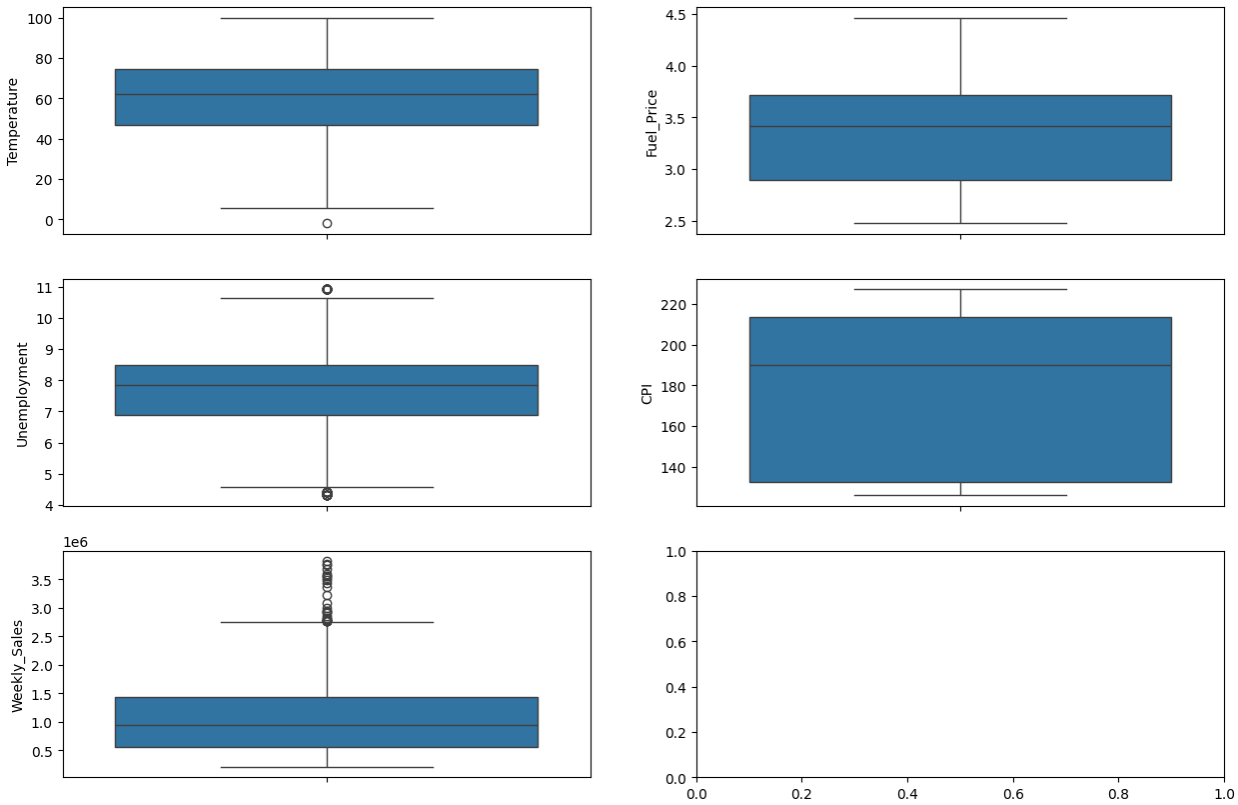
El marco de datos anterior muestra que las columnas weekly_sales, temperature y unemployment tienen valores atípicos, siendo unemployment la variable con el mayor porcentaje de valores atípicos. Por lo que se procede a quitar los valores atípicos de la variable unemployment.

```
# Calculamos el Quartil 1 y Quartil 3 que son aquellos que nos permiten estimar los límites de los datos atípicos
Q1 = df.Unemployment.quantile(0.25)
Q3 = df.Unemployment.quantile(0.75)
IQR = Q3 - Q1  #rango intercuartil
print(IQR)
```

```
1.7309999999999999
```

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

```
(5954, 10)
```

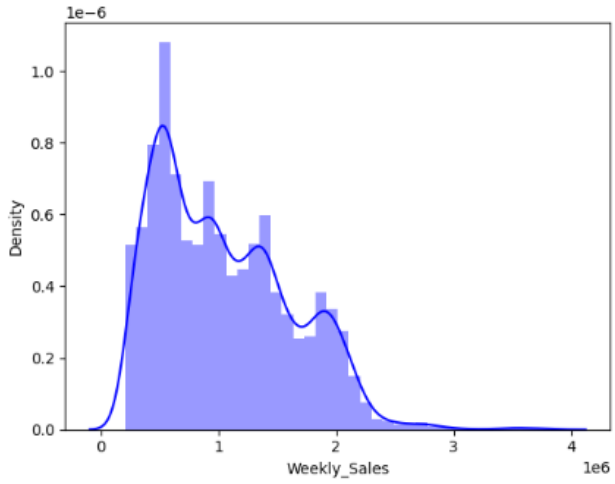


5. Grafique las distribuciones de las variables y a priori comente sobre ellas.

weekly_sales: En el gráfico de distribución de la variable weekly_sales se observa una distribución asimétrica hacia la derecha, lo que indica que la mayoría de las semanas tienen ventas más bajas, la presencia de varios picos en el histograma podría indicar mayor porcentaje dentro de este conjunto de ventas semanales.

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

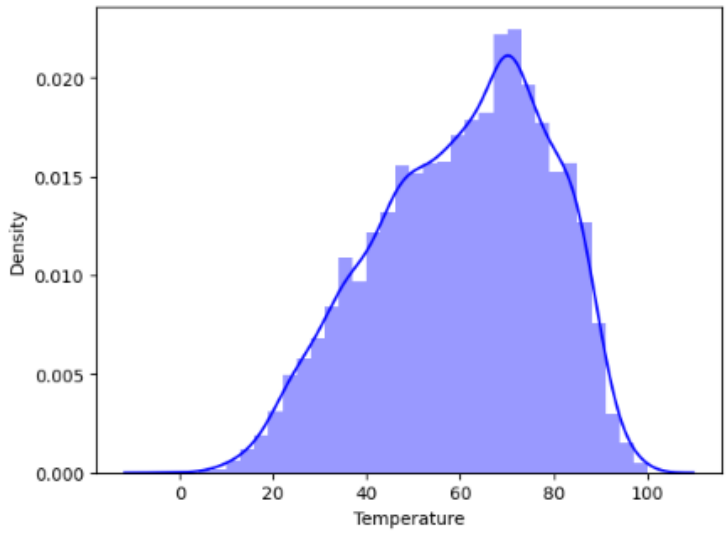
```
<Axes: xlabel='Weekly_Sales', ylabel='Density'>
```



Temperature: En el gráfico de distribución de la variable temperature se observa una distribución simétrica normal, la mayoría de las temperaturas están concentradas alrededor de 70 grados.

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

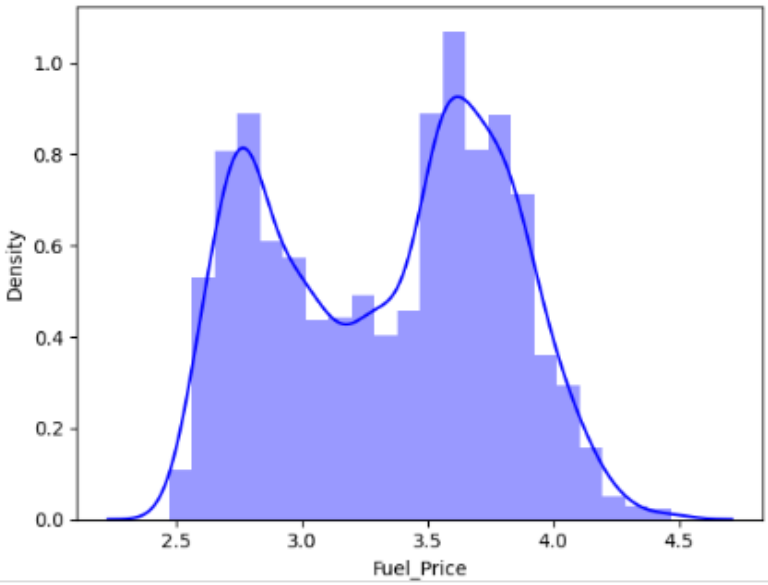
```
<Axes: xlabel='Temperature', ylabel='Density'>
```



Fuel_Price: En el gráfico de distribución de la variable fuel_price se observa una distribución bimodal por lo que sugiere que hay dos grupos de precios de combustible en el conjunto de datos, concentrándose uno en 2.75 y otro alrededor de 3.5.

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

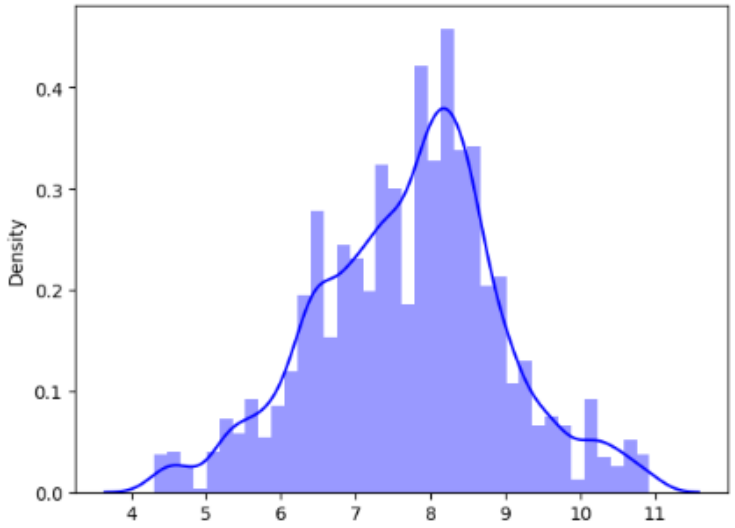
```
<Axes: xlabel='Fuel_Price', ylabel='Density'>
```



Unemployment: En el gráfico de distribución de la variable unemployment se observa una distribución simétrica, indica que la mayoría de los datos se agrupan alrededor del valor central (entre una tasa de desempleo de 8 y 9).

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

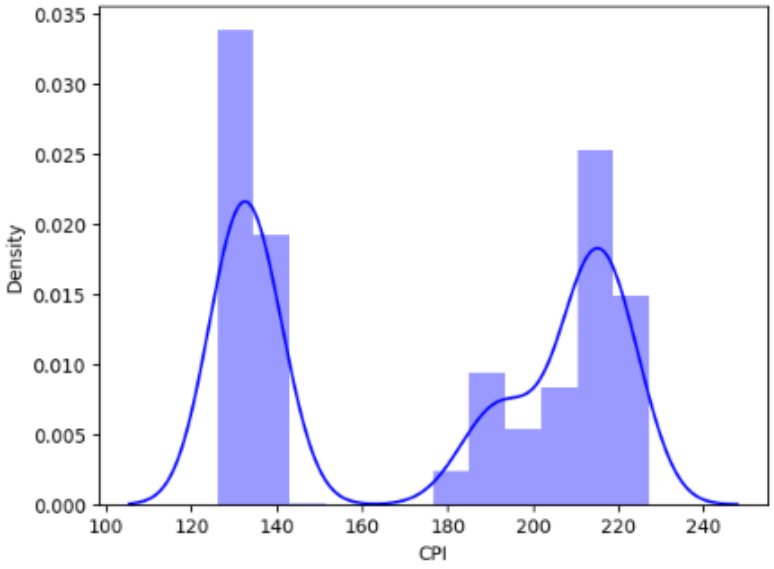
```
<Axes: xlabel='Unemployment', ylabel='Density'>
```



CPI: En el gráfico de distribución de la variable CPI se observa dos principales picos que indican mayores concentraciones de índices de precios al consumidor esta en 130 y 210.

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

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



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

```
df.corr(numeric_only=True).style.background_gradient(cmap='coolwarm')
```

	Weekly_Sales	Temperature	Fuel_Price	CPI	Unemployment
Weekly_Sales	1.000000	-0.061389	0.011257	-0.087443	-0.074999
Temperature	-0.061389	1.000000	0.147560	0.218762	0.026236
Fuel_Price	0.011257	0.147560	1.000000	-0.142689	-0.104268
CPI	-0.087443	0.218762	-0.142689	1.000000	-0.216206
Unemployment	-0.074999	0.026236	-0.104268	-0.216206	1.000000

- Existe una correlación negativa débil entre weekly_sales y temperature. Las ventas tienden a disminuir ligeramente conforme aumenta la temperatura.

- La correlación entre weekly_sales y Fuel_Price es muy débil y positiva. Esto indica una relación casi nula entre estas dos variables.
- Existe una correlación negativa moderada entre weekly_sales y CPI. Esto sugiere que cuando el CPI aumenta, las ventas tienden a disminuir.

- La correlación entre weekly_sales y Unemployment es negativa y débil. Esto indica una ligera tendencia a que las ventas disminuyan cuando aumenta el desempleo.

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

   Variable Dependiente.

   weekly_sales

   Variables Independientes.

```
X = df_.drop(['Date','Weekly_Sales','year','month','Store_1','Holiday_Flag_0'],axis=1)
y = df_.Weekly_Sales
```

```
var_X = X.columns
var_X
```

```
Index(['Temperature', 'Fuel_Price', 'CPI', 'Unemployment', 'Store_2',
       'Store_3', 'Store_4', 'Store_5', 'Store_6', 'Store_7', 'Store_8',
       'Store_9', 'Store_10', 'Store_11', 'Store_12', 'Store_13', 'Store_14',
       'Store_15', 'Store_16', 'Store_17', 'Store_18', 'Store_19', 'Store_20',
       'Store_21', 'Store_22', 'Store_23', 'Store_24', 'Store_25', 'Store_26',
       'Store_27', 'Store_28', 'Store_29', 'Store_30', 'Store_31', 'Store_32',
       'Store_33', 'Store_34', 'Store_35', 'Store_36', 'Store_37', 'Store_38',
       'Store_39', 'Store_40', 'Store_41', 'Store_42', 'Store_43', 'Store_44',
       'Store_45', 'Holiday_Flag_1'],
      dtype='object')
```

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

Utilizamos el modelo de **regresión lineal de Scikit-learn** para predecir las ventas de Walmart debido a su simplicidad y efectividad. La regresión lineal es fácil de implementar y proporciona resultados interpretables, permitiendo identificar claramente cómo diferentes factores influyen en las ventas. Además, su eficiencia computacional y facilidad de uso en Scikit-learn permiten un desarrollo rápido y una evaluación sencilla del modelo.

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

   No se escogió un modelo econométrico.

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

    Train/test split, separando un 90% de los datos para la submuestra de entrenamiento y 10% para la submuestra de prueba.

```
from sklearn.model_selection import train_test_split
```

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.10,random_state=123)
```

```
print(X_train.shape,"",type(X_train))
print(y_train.shape,"\t ",type(y_train))
print(X_test.shape,"",type(X_test))
print(y_test.shape,"\t ",type(y_test))
```

```
(5358, 49)  <class 'pandas.core.frame.DataFrame'>
(5358,)          <class 'pandas.core.series.Series'>
(596, 49)  <class 'pandas.core.frame.DataFrame'>
(596,)    <class 'pandas.core.series.Series'>
```

Entrenamiento del modelo de regresión lineal por sklearn.

```
modelo_regresion = LinearRegression()
modelo_regresion.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
predicciones_train = modelo_regresion.predict(X_train)
predicciones_test = modelo_regresion.predict(X_test)
```

Resultados de Errores:

```
MSE_train = mean_squared_error(y_train, predicciones_train)
MSE_test = mean_squared_error(y_test, predicciones_test)
print(MSE_train)
print(MSE_test)
```

```
26061606578.7453
26541682094.386703
```

```
RMSE_train = np.sqrt(MSE_train)
RMSE_test = np.sqrt(MSE_test)
print(RMSE_train)
print(RMSE_test)
```

```
161436.07582800474
162916.18119262034
```

```
MAE_train = mean_absolute_error(y_train, predicciones_train)
MAE_test = mean_absolute_error(y_test, predicciones_test)
print(MAE_train)
print(MAE_test)
```

```
91395.82746256446
94078.9788347326
```

Al realizar un análisis del error absoluto, el modelo se equivoca más menos en 94078, lo que es un margen de error pequeño considerando los valores de la variable weekly_sales.

R cuadrado:

```
r_square_train = r2_score(y_train, predicciones_train)
r_square_test  = r2_score(y_test, predicciones_test)
print('El R^2 del subconjunto de entrenamiento es:' , r_square_train.round(2))
print('El R^2 del subconjunto de prueba es:' , r_square_test.round(2))
```

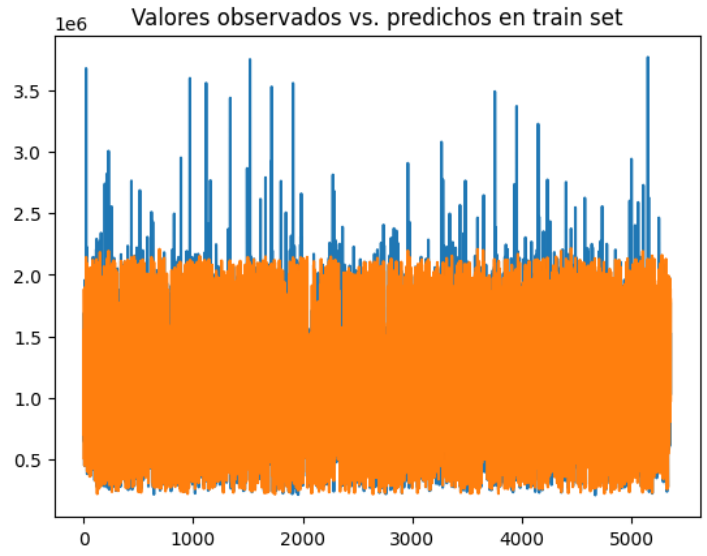
```
El R^2 del subconjunto de entrenamiento es: 0.92
El R^2 del subconjunto de prueba es: 0.92
```

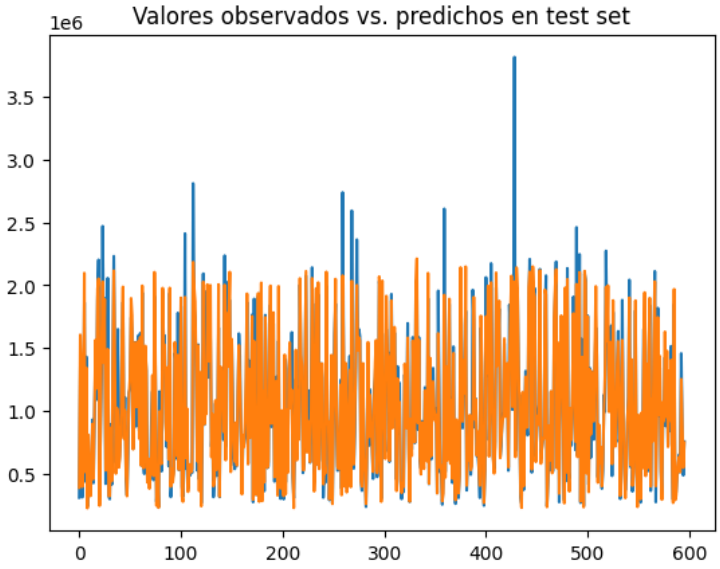
El R cuadrado de la base de entrenamiento me indica un valor de 0.92, lo que indica un gran ajuste del modelo.

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.

    La línea naranja (valores predichos) tiende a seguir de cerca la línea azul (valores observados), lo que indica que el modelo ha capturado cierta parte de la variabilidad de los datos.





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.

Para realizar el análisis de "feature importance" y calcular la importancia de cada variable independiente en nuestro modelo. Ajustamos nuestro modelo de regresión lineal y extraemos los coeficientes de los predictores, los cuales proporcionan la base para una puntuación de importancia de características. Para este tipo de análisis requerimos que las variables de entrada tengan la misma escala por lo que procedemos a normalizar:

Normalización:

```
sc = StandardScaler()
```

```
X_train_std = sc.fit_transform(X_train)
X_test_std = sc.transform(X_test)
```

```
modelo_regresion_std = LinearRegression()
modelo_regresion_std.fit(X_train_std, y_train)
```

LinearRegression
LinearRegression()

```
predicciones_train_std = modelo_regresion_std.predict(X_train_std)
predicciones_test_std = modelo_regresion_std.predict(X_test_std)
```

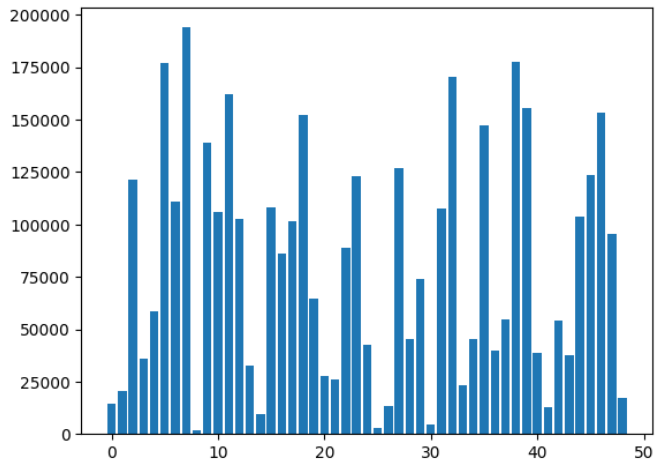
```
importancia = modelo_regresion_std.coef_
```

```
modelo_regresion_std.coef_.flags
```

```
  C_CONTIGUOUS : True
  F_CONTIGUOUS : True
  OWNDATA : True
  WRITEABLE : True
  ALIGNED : True
  WRITEBACKIFCOPY : False
```

```
# Resumen
for i,v in enumerate(importancia):
    print('Variable explicativa No. %0d, Score: %.5f' % (i,v))
```

```
Variable explicativa No. 0, Score: -14364.37709
Variable explicativa No. 1, Score: -20715.16672
Variable explicativa No. 2, Score: 121211.84891
Variable explicativa No. 3, Score: -35885.87124
Variable explicativa No. 4, Score: 58290.16451
Variable explicativa No. 5, Score: -176782.57021
Variable explicativa No. 6, Score: 110636.28764
Variable explicativa No. 7, Score: -193882.71924
Variable explicativa No. 8, Score: -1879.52680
Variable explicativa No. 9, Score: -139070.88661
Variable explicativa No. 10, Score: -106154.86107
Variable explicativa No. 11, Score: -161900.00430
Variable explicativa No. 12, Score: 102518.64088
Variable explicativa No. 13, Score: -32840.24181
Variable explicativa No. 14, Score: -9677.25703
Variable explicativa No. 15, Score: 108290.99569
Variable explicativa No. 16, Score: 86209.36505
Variable explicativa No. 17, Score: -101591.15572
Variable explicativa No. 18, Score: -152382.26711
Variable explicativa No. 19, Score: -64412.52021
Variable explicativa No. 20, Score: -27802.42255
Variable explicativa No. 21, Score: 25789.86682
Variable explicativa No. 22, Score: 89074.77294
Variable explicativa No. 23, Score: -122731.56725
Variable explicativa No. 24, Score: -42761.95813
Variable explicativa No. 25, Score: 2900.55408
Variable explicativa No. 26, Score: 13414.84378
Variable explicativa No. 27, Score: -126705.40703
Variable explicativa No. 28, Score: -45381.58478
Variable explicativa No. 29, Score: 74063.13053
Variable explicativa No. 30, Score: 4393.31146
Variable explicativa No. 31, Score: -107518.57167
Variable explicativa No. 32, Score: -170381.19424
Variable explicativa No. 33, Score: -23295.02810
Variable explicativa No. 34, Score: -45542.73654
Variable explicativa No. 35, Score: -147056.78965
Variable explicativa No. 36, Score: -39758.84528
Variable explicativa No. 37, Score: -54548.40385
Variable explicativa No. 38, Score: -177569.14495
Variable explicativa No. 39, Score: -155710.15771
Variable explicativa No. 40, Score: -38976.05513
Variable explicativa No. 41, Score: -12992.40256
Variable explicativa No. 42, Score: -54255.52651
Variable explicativa No. 43, Score: -37758.59859
Variable explicativa No. 44, Score: -103817.67111
Variable explicativa No. 45, Score: -123642.61387
Variable explicativa No. 46, Score: -153331.22165
Variable explicativa No. 47, Score: -95673.11979
Variable explicativa No. 48, Score: 17212.48513
```



Las variables 10 variables más importantes son:

```
# Ordena los scores y toma los más grandes
indices_mayores = np.argsort(abs(importancia))[::-1][:10]

# Imprime los scores más grandes y sus variables correspondientes
for i in indices_mayores:
    print('Variable explicativa No. %0d, Score: %.5f' % (i, abs(importancia[i])))
```

```
Variable explicativa No. 7, Score: 193882.71924
Variable explicativa No. 38, Score: 177569.14495
Variable explicativa No. 5, Score: 176782.57021
Variable explicativa No. 32, Score: 170381.19424
Variable explicativa No. 11, Score: 161900.00430
Variable explicativa No. 39, Score: 155710.15771
Variable explicativa No. 46, Score: 153331.22165
Variable explicativa No. 18, Score: 152382.26711
Variable explicativa No. 35, Score: 147056.78965
Variable explicativa No. 9, Score: 139070.88661
```

13. Suba su proyecto final en su cuenta de Github y adjunte una captura de pantalla en esta plataforma.