Universidad Nacional Autónoma de México

Facultad de Ingeniería

División de Ingeniería Eléctrica

Ingeniería en Computación

Práctica 3.3
“Series de tiempo”

Integrantes:

- Castelan Ramos Carlos
- Reyes Esquivel Ana Karen
- Rocandio Montiel Jesús Alexis
- Romero López Alexis Uriel

Materia: Minería de Datos
Grupo: 03
Semestre: 2024-1

Fecha de entrega: 11 de octubre2023

Práctica 3.3 – Series de tiempo

UBICACIÓN DEL ARCHIVO:

https://colab.research.google.com/drive/1mNRDKhVT788IFvT7MU9PKZOvz4KuYpBW?usp=sharing

Instrucciones.

Realizar pronósticos de ventas (importe) utilizando Series de tiempo de Triple Exponential Smoothing y ARIMA para los siguientes escenarios:

1.Seleccionar 1 o 2 variables objetivo que consideren entre las siguientes: Sector o Línea
2.Para cada variable seleccionada:
1.Describir el caso de negocio, del porque se desea realizar un pronostico
2.Ejecutar el pronóstico ambas Series de Tiempos
3.En el caso de Triple Exponential Smoothing descomponer cada elemento como: Tendencia, Ciclicidad, Estacionalidad y el componente Irregular
4.En el caso de ARIMA incluir ACF y PACF
5.Incluir métricas para medir la calidad de cada modelo
6.Dar sus comentarios de los resultados obtenidos

Desarrollo:

Para la siguiente actividad se realizaron varios ejemplos, implementados diferentes modelos de clasificación, donde las variables objetivas y las variables dependientes fueron elegidas a partir de las características por las cuales trabajan los modelos de clasificación.

Título del Caso de Negocio: Optimización de Inventario y Estrategia de Precios para Carnes y Quesos

Introducción:

Una cadena de supermercados regional está buscando mejorar la gestión de sus productos cárnicos y lácteos, específicamente carnes y quesos. La empresa ha experimentado fluctuaciones en las ventas mensuales de estos productos y está interesada en maximizar sus beneficios mediante el análisis de datos.

Objetivos del Caso de Negocio:

- Analizar las ventas mensuales de carnes y quesos en los últimos 2 años.
- Identificar tendencias estacionales en la demanda de estos productos.

- Optimizar el inventario para reducir costos y evitar escasez o exceso de stock.
- Desarrollar una estrategia de precios competitiva y rentable para carnes y quesos.

Beneficios Esperados:

- Reducción de costos de almacenamiento al evitar exceso de stock.
- Aumento de las ventas al garantizar la disponibilidad de productos en temporada alta.
- Mayor rentabilidad a través de una estrategia de precios efectiva.
- Mayor satisfacción del cliente debido a la disponibilidad constante de productos.

SERIES DE TIEMPO

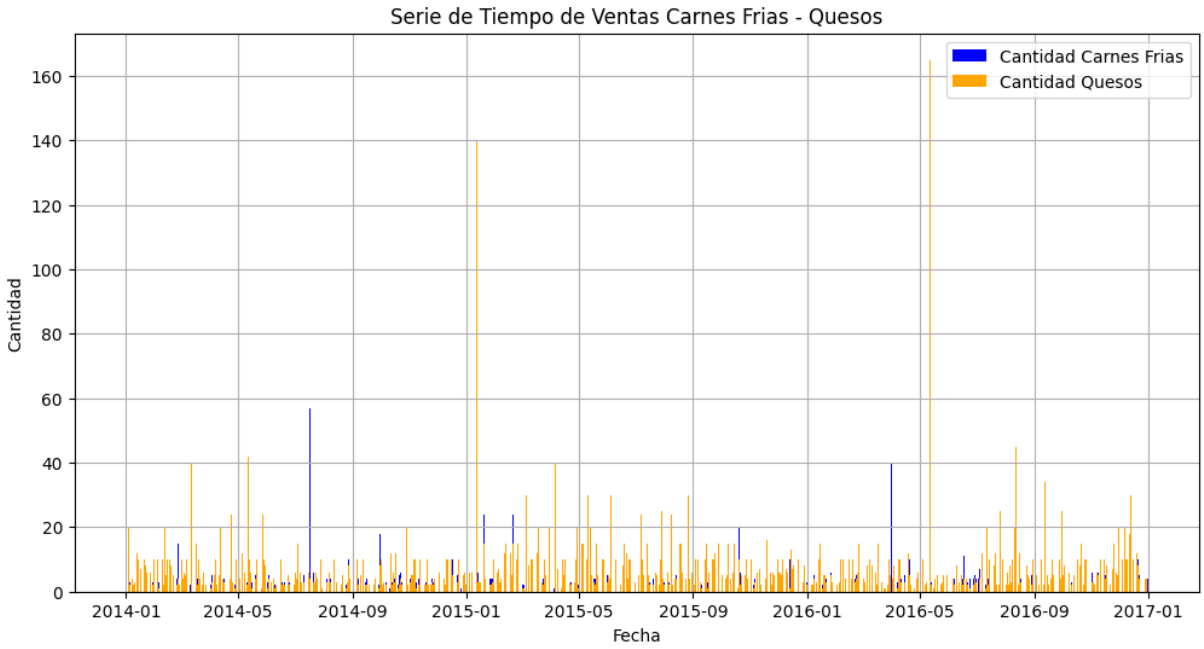
Código usado para serie de tiempo:

```python
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.dates as mdates

# Cargar el archivo CSV inicial
data0 = pd.read_csv("transacciones.csv", encoding='latin1')

# Definir la clase que te interesa (por ejemplo, 'Clase_A')
claseCarnes = ['CARNES FRIAS']
claseQuesos = ['QUESOS']

# Filtrar las filas que tienen la clase deseada carnes y quesos
filasCarnes = data0[data0['Sector'].isin(claseCarnes)]
filasQuesos = data0[data0['Sector'].isin(claseQuesos)]

#print(filas_objetivo)
# Guardar las filas identificadas en un nuevo archivo CSV_objetivo2
filasCarnes.to_csv('filasCarnes.csv', index=False)
filasQuesos.to_csv('filasQuesos.csv', index=False)

# Cargar el archivo CSV (reemplaza 'tu_archivo.csv' con la ubicación de tu archivo)
dataCarnes = pd.read_csv('filasCarnes.csv', encoding='latin1')
dataQuesos = pd.read_csv('filasQuesos.csv', encoding='latin1')

#Generamos SERIE DE TIEMPO
dataCarnes['Fecha'] = pd.to_datetime(dataCarnes['Fecha'])
dataCarnes.set_index('Fecha', inplace=True)

dataQuesos['Fecha'] = pd.to_datetime(dataQuesos['Fecha'])
dataQuesos.set_index('Fecha', inplace=True)
```

```python
# Graficar la serie de tiempo de ventas
plt.figure(figsize=(12, 6))
#plt.plot(data1.index, data1['Cantidad'], label='Cantidad', color='orange')
#plt.plot(data2.index, data2['Cantidad'], label='Cantidad', color='blue')
plt.bar(dataCarnes.index, dataCarnes['Cantidad'], label='Cantidad Carnes Frias',color='blue', width=0.8)
plt.bar(dataQuesos.index, dataQuesos['Cantidad'], label='Cantidad Quesos',color='orange', width=0.8)
#plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%b %Y'))  # Formato de fechas de meses
plt.title('Serie de Tiempo de Ventas Carnes Frias - Quesos')
plt.xlabel('Fecha')
plt.ylabel('Cantidad')
plt.legend()
plt.grid(True)
plt.show()

# Guardar la serie de tiempo en un nuevo archivo CSV
dataCarnes.to_csv('serieTiempoCarnes.csv')
dataQuesos.to_csv('serieTiempoQuesos.csv')
```

Serie de tiempo de Ventas de Carnes Frías- Quesos

A continuación, se muestra los resultados del algoritmo de series de tiempo respecto a ventas de carnes frías y quesos. En donde podemos observar que en el período de 2016-05 tenemos la más alta cantidad de consumo con un valor por arriba de 160, además se puede observar que en el principio de tres periodos hay una alta cantidad de ventas.
Respecto al valor más alto del sector 'quesos' se dio en el período de 2014–05, con un consumo por arriba de 40. También podemos observar que el consumo de carnes frías supera al de quesos en la mayoría de los años.



Muestra de datos de las carnes:

Fecha	Sector	Marca	Linea_2	Sublinea_2	Presentacion	Gramaje_2	Empresa	AÃ±o	Cliente	Nota	...	Fecha_2	Year	Quarter	Month	Day	Dia_Num_Sem	Dia_Semana_	IND_Compra	IND_Precio_Sospech
2014-01-01	CARNES FRIAS	MARCA MAY	JAMONES	PECHUGA	PAQUETERIA	0.26	303012 CORPORATIVO BETA COMERCIAL	2014	129070 LOPEZ RA	542421	...	01-01-14	2014.0	1.0	1.0	1.0	3.0	Miercoles	0.0	
2014-01-01	CARNES FRIAS	MARCA HID	SALCHICHA	SALCHICHA TRADICIONA	PAQUETERIA	0.55	303012 CORPORATIVO BETA COMERCIAL	2014	129070 LOPEZ RA	542421	...	01-01-14	2014.0	1.0	1.0	1.0	3.0	Miercoles	0.0	
2014-01-01	CARNES FRIAS	MARCA HID	SALCHICHA	SALCHICHA TRADICIONA	PAQUETERIA	0.36	303012 CORPORATIVO BETA COMERCIAL	2014	129070 LOPEZ RA	542421	...	01-01-14	2014.0	1.0	1.0	1.0	3.0	Miercoles	0.0	
2014-03-01	CARNES FRIAS	MARCA JAL	JAMONES	JAMON VIRGINIA	PAQUETERIA	0.29	957936 ALFA TORRE	2014	168833 XIOMARA	542432	...	03-01-14	2014.0	1.0	1.0	3.0	5.0	Viernes	0.0	
2014-03-01	CARNES FRIAS	MARCA HID	SALCHICHA	SALCHICHA TIPO PAVO	PAQUETERIA	0.50	957936 ALFA TORRE	2014	168833 XIOMARA	542432	...	03-01-14	2014.0	1.0	1.0	3.0	5.0	Viernes	0.0	

5 rows × 42 columns

Muestra de datos de los quesos:

Fecha	Sector	Marca	Linea_2	Sublinea_2	Presentacion	Gramaje_2	Empresa	AÃ±o	Cliente	Nota	...	Fecha_2	Year	Quarter	Month	Day	Dia_Num_Sem	Dia_Semana_	IND_Compra	IND_Precio_Sospe
2014-01-01	QUESOS	MARCA TLA	Q-MADURADOS	MANCHEGO	PAQUETERIA	0.4	303012 CORPORATIVO BETA COMERCIAL	2014	129070 LOPEZ RA	542421	...	01-01-14	2014.0	1.0	1.0	1.0	3.0	Miercoles	0.0	
2014-01-01	QUESOS	MARCA CHIA	Q-FRESCOS	PANELA	PAQUETERIA	0.4	303012 CORPORATIVO BETA COMERCIAL	2014	129070 LOPEZ RA	542421	...	01-01-14	2014.0	1.0	1.0	1.0	3.0	Miercoles	0.0	
2014-03-01	QUESOS	MARCA CHIA	Q-MADURADOS	RALLADO	PAQUETERIA	0.3	957936 ALFA TORRE	2014	168833 XIOMARA	542432	...	03-01-14	2014.0	1.0	1.0	3.0	5.0	Viernes	0.0	
2014-03-01	QUESOS	MARCA TAM	Q-FRESCOS	PANELA	PAQUETERIA	0.4	957936 ALFA TORRE	2014	168833 XIOMARA	542432	...	03-01-14	2014.0	1.0	1.0	3.0	5.0	Viernes	0.0	
2014-03-01	QUESOS	MARCA CHIA	Q-FRESCOS	PANELA	PAQUETERIA	0.4	436463 ALFA CARRIZALEJO	2014	139044 EDGARDO	542440	...	03-01-14	2014.0	1.0	1.0	3.0	5.0	Viernes	0.0	

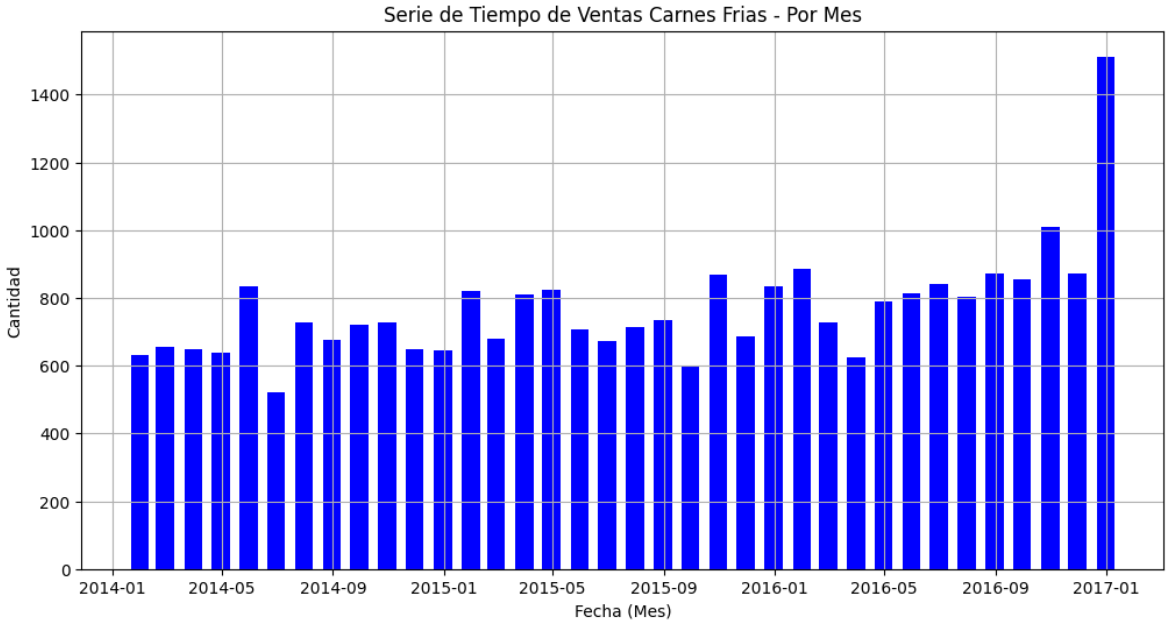
5 rows × 42 columns

Separando datos por mes:

```
# Cargar el archivo CSV con la serie de tiempo
dataMesCarne = pd.read_csv('serieTiempoCarnes.csv')
dataMesQuesos = pd.read_csv('serieTiempoQuesos.csv')

#Obtenemos columna fecha
dataMesCarne['Fecha'] = pd.to_datetime(dataMesCarne['Fecha'])
dataMesQuesos['Fecha'] = pd.to_datetime(dataMesQuesos['Fecha'])

# Calcular el total de ventas por mes
ventasCarnesMes = dataMesCarne.resample('M', on='Fecha')['Cantidad'].sum().reset_index()
ventasQuesosMes = dataMesQuesos.resample('M', on='Fecha')['Cantidad'].sum().reset_index()

# Guardar el resultado en un nuevo archivo CSV (reemplaza 'ventas_por_mes.csv' con el nombre que desees)
ventasCarnesMes.to_csv('ventasCarnesMes.csv', index=False)
ventasQuesosMes.to_csv('ventasQuesosMes.csv', index=False)
```

Serie de tiempo de Ventas de Carnes Frías

Ahora, para entrar más a detalle y tener una mejor visualización de los datos, nos centraremos específicamente en los en el sector de 'Carnes Frías' en donde agrupamos los datos por meses para tener una mejor visualización de las ventas a lo largo de tiempo. Podemos observar que el consumo a lo largo del tiempo es variable respecto a temporadas, Sin embargo existe una tendencia de crecimiento a la compra.



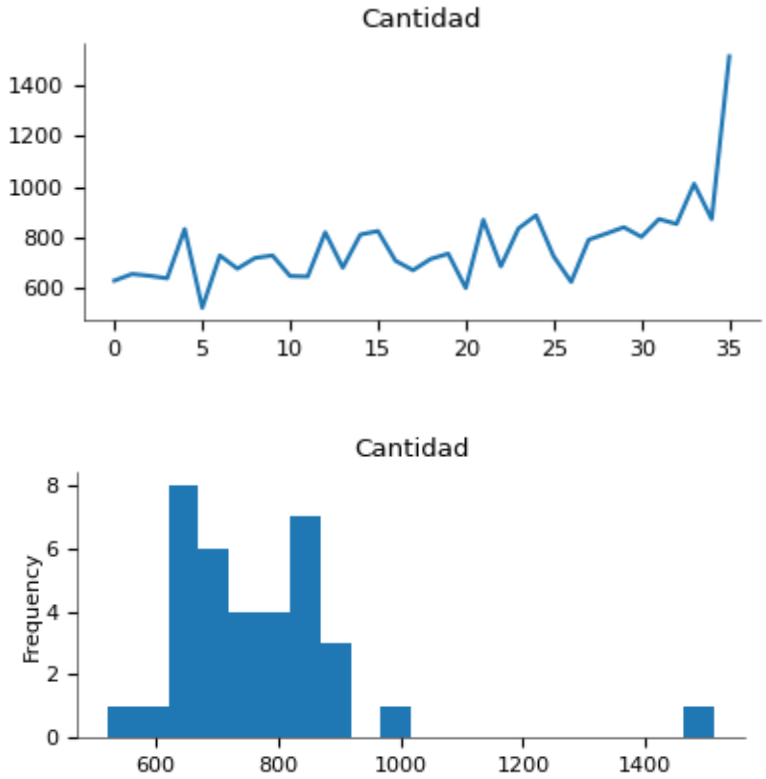

```
ventasCarnesMes.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36 entries, 0 to 35
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Fecha     36 non-null     datetime64[ns]
 1   Cantidad  36 non-null     int64
dtypes: datetime64[ns](1), int64(1)
memory usage: 704.0 bytes
```

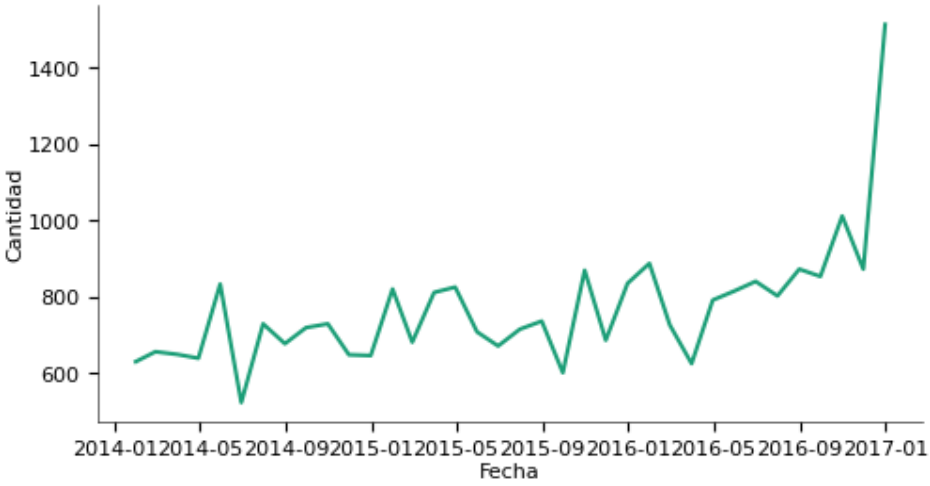
	Fecha	Cantidad
0	2014-01-31	630
1	2014-02-28	656
2	2014-03-31	649
3	2014-04-30	639
4	2014-05-31	833
5	2014-06-30	523
6	2014-07-31	729
7	2014-08-31	677
8	2014-09-30	719
9	2014-10-31	729
10	2014-11-30	648
11	2014-12-31	646
12	2015-01-31	820
13	2015-02-28	681
14	2015-03-31	811
15	2015-04-30	825
16	2015-05-31	708
17	2015-06-30	671
18	2015-07-31	715
19	2015-08-31	736
20	2015-09-30	601
21	2015-10-31	869
22	2015-11-30	686

22	2015-11-30	686
23	2015-12-31	835
24	2016-01-31	887
25	2016-02-29	726
26	2016-03-31	625
27	2016-04-30	791
28	2016-05-31	815
29	2016-06-30	840
30	2016-07-31	802
31	2016-08-31	872
32	2016-09-30	853
33	2016-10-31	1011
34	2016-11-30	872
35	2016-12-31	1512

Después, podemos observar de manera gráfica el comportamiento continuo de las ventas a lo largo del tiempo. Donde del lado ‘x’, cada valor numérico del 0 al 35 representa una fecha específica durante a lo largo del tiempo. Esto nos facilita identificar con mayor claridad los puntos con crecimiento y decremento. En las cuales podemos observar tres caídas abruptas en los períodos de 2014-05, 2015-09 y 2016-12. Además, punto con mayor crecimiento en el período 2014-05 y 2015-09. Esto nos permite identificar exactamente los períodos con menor y mayor ganancias y así aprovecharlos para planeación y toma de decisiones a futuro

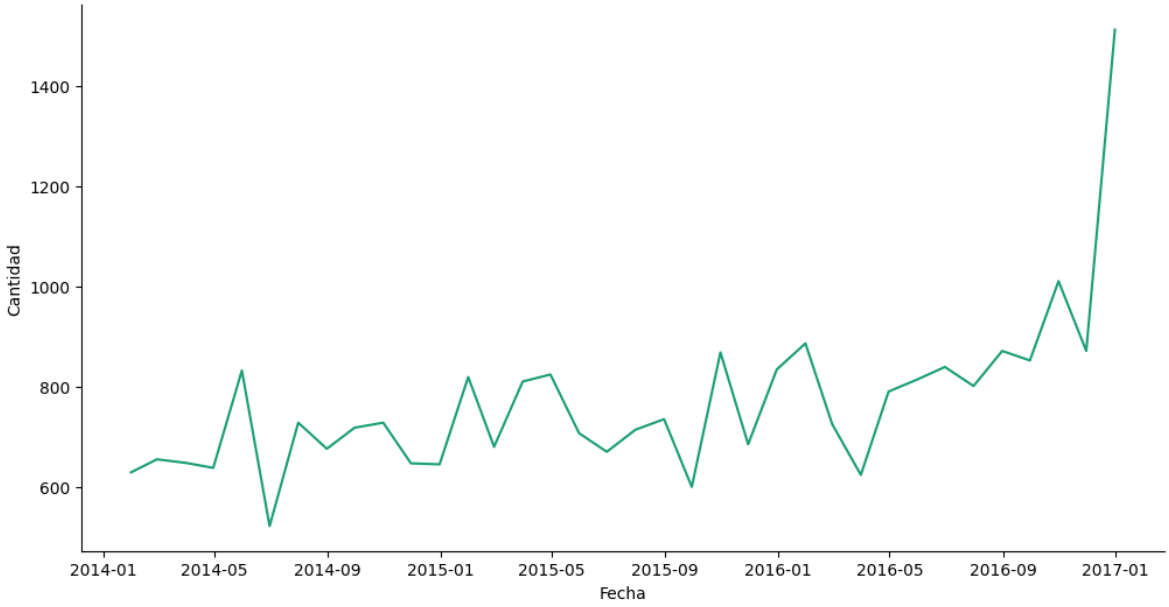


Se puede observar de manera gráfica el comportamiento continuo de las ventas a lo largo del tiempo. Esta gráfica lo que nos facilita es ver los puntos con crecimiento y decremento. En las cuales podemos observar tres caídas abruptas en los períodos de 2014-05, 2015-09 y 2016-12. Además, punto con mayor crecimiento en el período 2014-05 y 2015-09. Esto nos permite identificar exactamente los períodos con menor y mayor ganancias y así aprovecharlos para planeación y toma de decisiones a futuro



Series de tiempo – Carnes Frías – método alterno

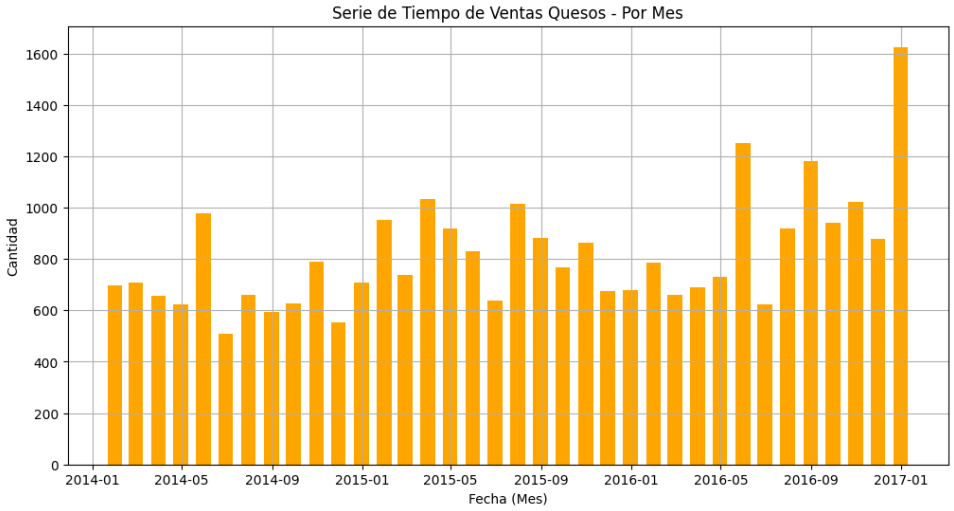
```
import numpy as np
from google.colab import autoviz

def time_series_multiline(df, timelike_colname, value_colname, series_colname, figscale=1, mpl_palette_name='Dark2'):
  from matplotlib import pyplot as plt
  import seaborn as sns
  figsize = (10 * figscale, 5.2 * figscale)
  palette = list(sns.palettes.mpl_palette(mpl_palette_name))
  def _plot_series(series, series_name, series_index=0):
    if value_colname == 'count()':
      counted = (series[timelike_colname]
                 .value_counts()
                 .reset_index(name='counts')
                 .rename({'index': timelike_colname}, axis=1)
                 .sort_values(timelike_colname, ascending=True))
      xs = counted[timelike_colname]
      ys = counted['counts']
    else:
      xs = series[timelike_colname]
      ys = series[value_colname]
    plt.plot(xs, ys, label=series_name, color=palette[series_index % len(palette)])

  fig, ax = plt.subplots(figsize=figsize, layout='constrained')
  df = df.sort_values(timelike_colname, ascending=True)
  if series_colname:
    for i, (series_name, series) in enumerate(df.groupby(series_colname)):
      _plot_series(series, series_name, i)
    fig.legend(title=series_colname, bbox_to_anchor=(1, 1), loc='upper left')
  else:
    _plot_series(df, '')
  sns.despine(fig=fig, ax=ax)
  plt.xlabel(timelike_colname)
  plt.ylabel(value_colname)
  return autoviz.MplChart.from_current_mpl_state()

chart = time_series_multiline(_df_2, *['Fecha', 'Cantidad', None], **{})
chart
```



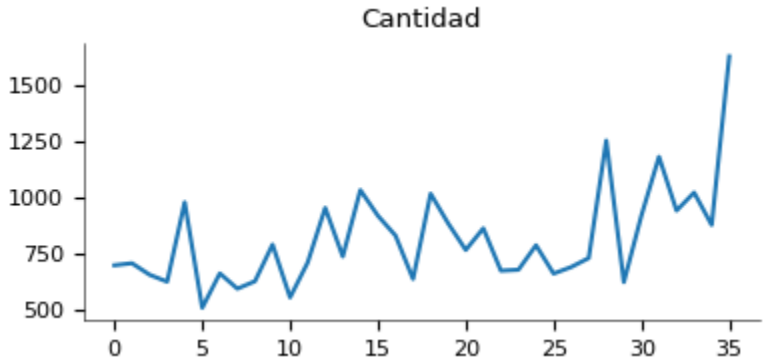
Series de tiempo- Quesos por mes

Ahora, vamos a pasar los datos del sector de 'Quesos' donde clasificamos los datos por meses para tener una mejor visualización de las ventas a lo largo de tiempo. Podemos observar que el consumo en el segundo período tuvo un decremento, pero a partir de ahí las ventas empiezan a incrementar de manera lenta y esto lo podemos comparando el primero período, 2014-1, con el último, 2017-01. Al igual que las carnes, estas tienen tendencia de crecimiento a la temporada, generando estacionalidad.

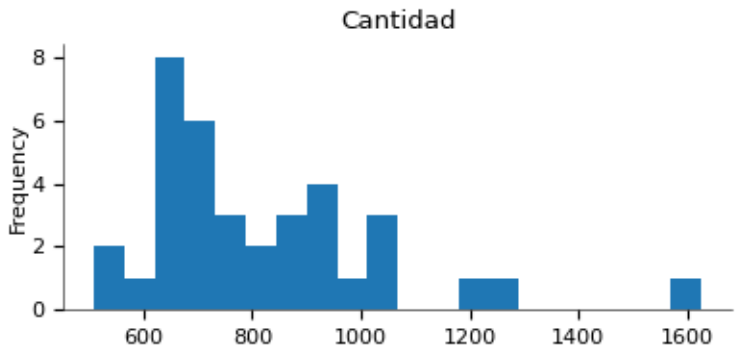
```
# Graficar la serie de tiempo en mes QUESOS
plt.figure(figsize=(12, 6))
plt.bar(ventasQuesosMes['Fecha'], ventasQuesosMes['Cantidad'], width=20, align='center', color='orange')
plt.title('Serie de Tiempo de Ventas Quesos - Por Mes')
plt.xlabel('Fecha (Mes)')
plt.ylabel('Cantidad')
plt.grid(True)
plt.show()
```



Posteriormente, podemos observar de manera gráfica el comportamiento continuo de las ventas a lo largo del tiempo. Donde del lado de la variable independiente, cada valor numérico del 0 al 35 representa una fecha específica a lo largo del tiempo. Esto nos facilita identificar con mayor claridad los puntos con crecimiento y decremento. En las cuales podemos observar tres caídas abruptas en los períodos de 2014-05, 2015-09 y 2016-12. Además, punto con mayor crecimiento en el período 2014-05 y 2015-09. Esto nos permite identificar exactamente los períodos con menor y mayor ganancias y así aprovecharlos para planeación y toma de decisiones a futuro

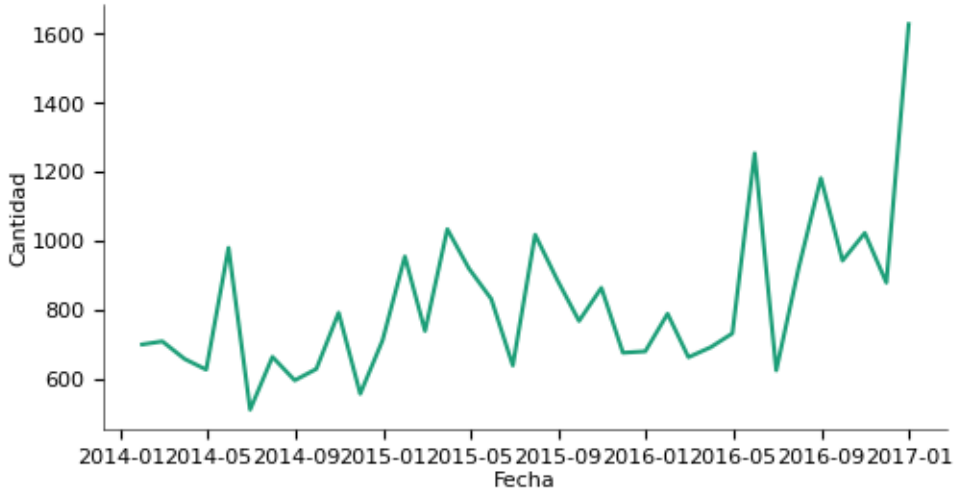


distribución

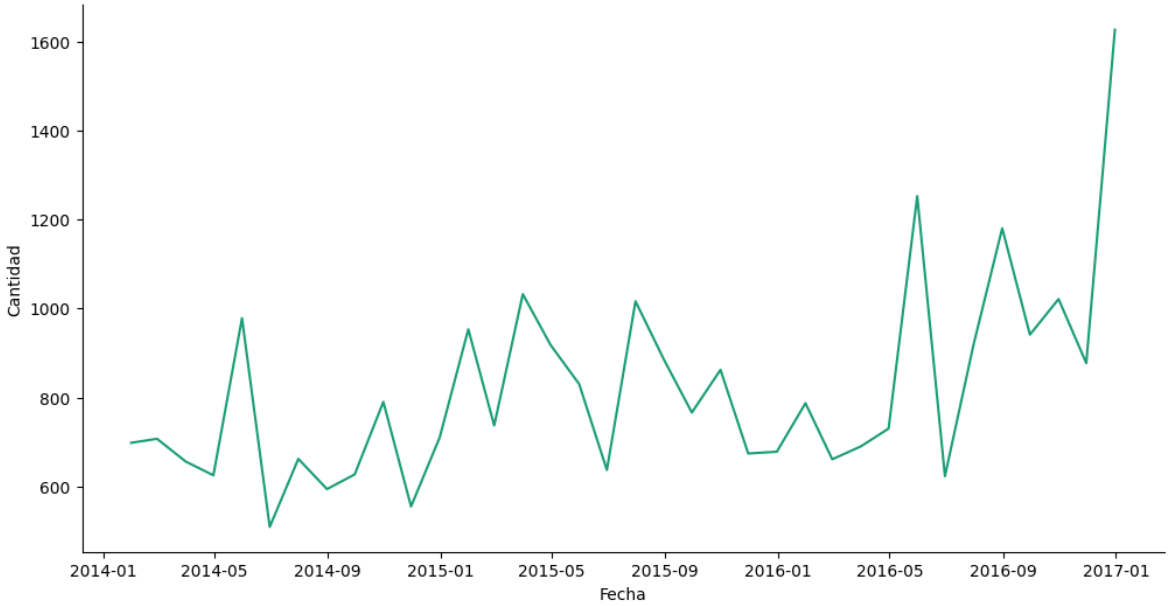


En seguida, podemos observar de manera gráfica el comportamiento continuo de las ventas del sector 'Quesos' a lo largo del tiempo. Esta gráfica nos facilita ver los puntos con crecimiento y decremento. En las cuales podemos observar cuatro caídas abruptas en los períodos de 2014-05, 2014-09, 2015-09 y 2016-05. Por el contrario se puede identificar tres crecimientos inmediatos en el período 2014-05, 2015-05 y 2016-05. Esto nos permite identificar exactamente los períodos con menor ganancias y considerarlos para la toma de decisiones a futuro.

Series de tiempo

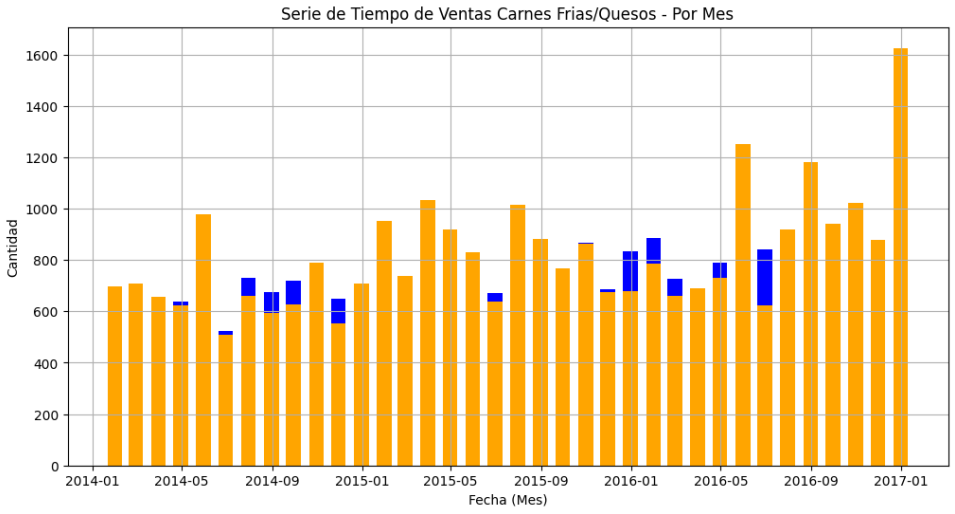


Otro método:



En la siguiente gráfica se muestra una comparación de ventas (cantidad) por mes comparando el sector: “Carnes frías” con “Quesos” donde se puede observar con mayor claridad que las ventas en carnes frías tuvo mayor cantidad de ventas a comparación del otro sector, además podríamos decir que en los próximos años habrá continuará el consumo en el primer sector mencionado.

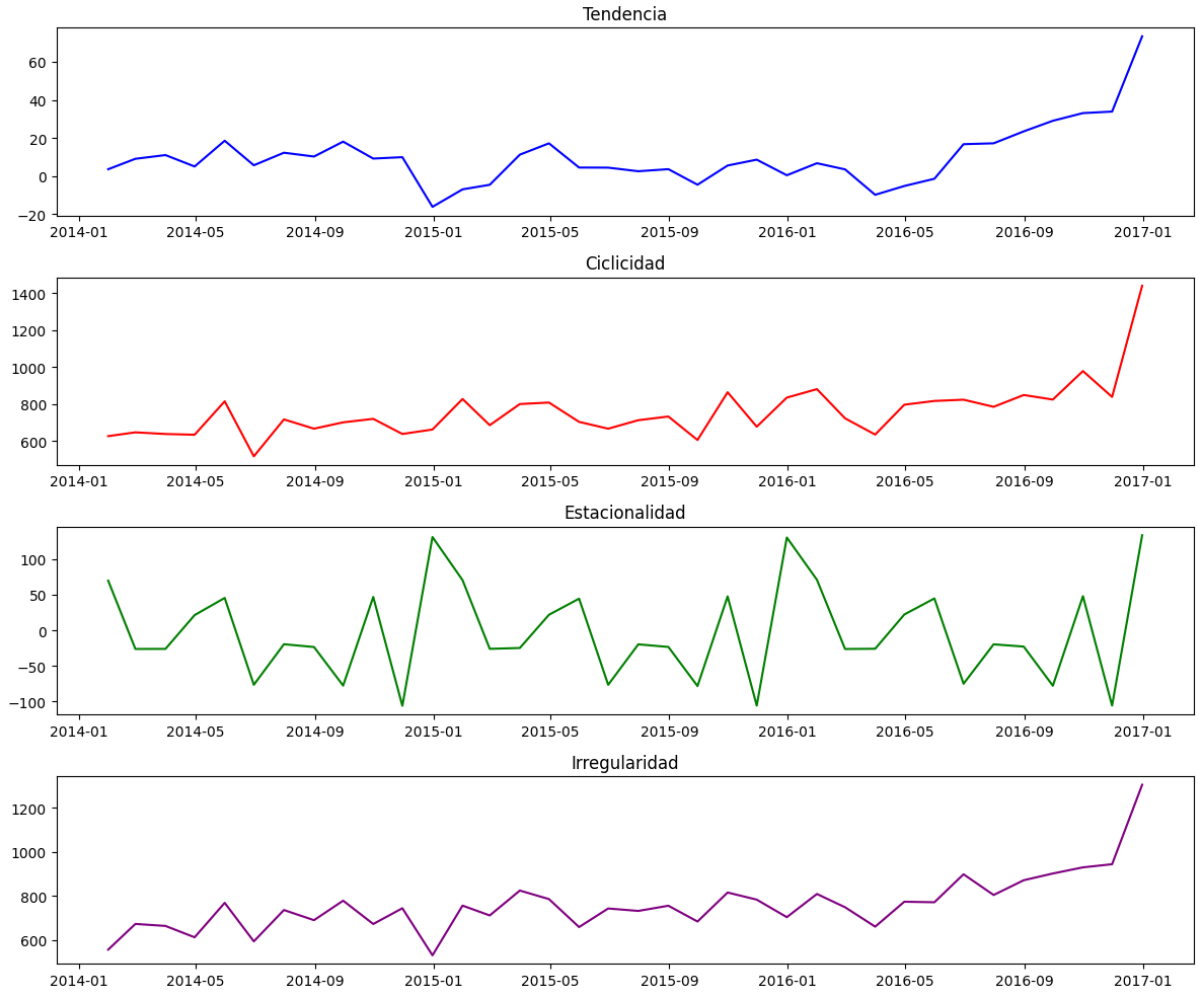
```
plt.figure(figsize=(12, 6))
plt.bar(ventasCarnesMes['Fecha'], ventasCarnesMes['Cantidad'], width=20, align='center', color='blue')
plt.bar(ventasQuesosMes['Fecha'], ventasQuesosMes['Cantidad'], width=20, align='center', color='orange')
plt.title('Serie de Tiempo de Ventas Carnes Frias/Quesos - Por Mes')
plt.xlabel('Fecha (Mes)')
plt.ylabel('Cantidad')
plt.grid(True)
plt.show()
```



Modelo triple Exponential Smoothing - Carnes

Posteriormente, se muestran cuatro subgráficos apilados verticalmente, cada uno mostrando una de las componentes descompuestas, cada uno mostrando una de las componentes descompuestas. Estos gráficos proporcionan una visualización de cómo, cada componente contribuye a la serie de tiempo y permiten analizar la tendencia, la estacionalidad la ciclicidad y la irregularidad por separado.
El primer subgráfico muestra la tendencia, la segunda muestra la ciclicidad, la tercera la estacionalidad y por último la irregularidad.

```
from statsmodels.tsa.holtwinters import ExponentialSmoothing

# Cargar el archivo CSV con las columnas "Fecha" y "Cantidad" (reemplaza 'tu_archivo.csv' con la ubicación de tu archivo)
TEScarnes = pd.read_csv('ventasCarnesMes.csv')

# Asegúrate de que el DataFrame tenga una columna de fecha y una columna de cantidad
# Reemplaza 'Fecha' y 'Cantidad' con los nombres reales de las columnas en tu archivo
TEScarnes['Fecha'] = pd.to_datetime(TEScarnes['Fecha'])
TEScarnes.set_index('Fecha', inplace=True)

# Ajustar el modelo Triple Exponential Smoothing
model = ExponentialSmoothing(TEScarnes['Cantidad'], trend='add', seasonal='add', seasonal_periods=12)
fit = model.fit()

# Descomponer la serie de tiempo en tendencia, ciclicidad, estacionalidad e irregularidad
tendencia = fit.trend
ciclicidad = TEScarnes['Cantidad'] - tendencia
estacionalidad = fit.season
irregularidad = TEScarnes['Cantidad'] - tendencia - estacionalidad

# Graficar las componentes descompuestas
plt.figure(figsize=(12, 10))
plt.subplot(4, 1, 1)
plt.plot(tendencia, label='Tendencia', color='blue')
plt.title('Tendencia')

plt.subplot(4, 1, 2)
plt.plot(ciclicidad, label='Ciclicidad', color='red')
plt.title('Ciclicidad')

plt.subplot(4, 1, 3)
plt.plot(estacionalidad, label='Estacionalidad', color='green')
plt.title('Estacionalidad')

plt.subplot(4, 1, 4)
plt.plot(irregularidad, label='Irregularidad', color='purple')
plt.title('Irregularidad')

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

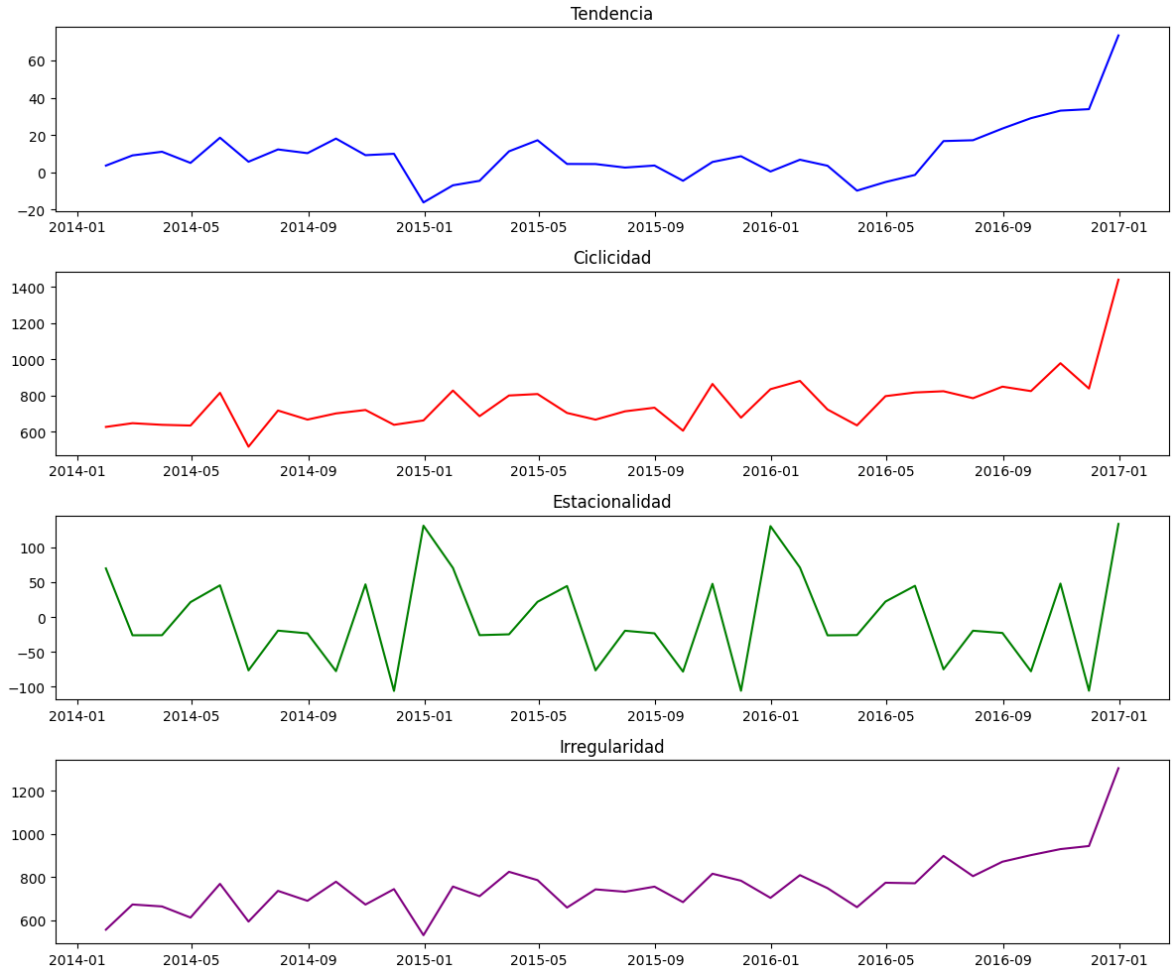


Triple Exponential Smoothing- Quesos

```
from statsmodels.tsa.holtwinters import ExponentialSmoothing

# Cargar el archivo CSV con las columnas "Fecha" y "Cantidad" (reemplaza 'tu_archivo.csv' con la ubicación de tu archivo)
TESquesos = pd.read_csv('ventasQuesosMes.csv')

# Asegúrate de que el DataFrame tenga una columna de fecha y una columna de cantidad
# Reemplaza 'Fecha' y 'Cantidad' con los nombres reales de las columnas en tu archivo
TESquesos['Fecha'] = pd.to_datetime(TESquesos['Fecha'])
TESquesos.set_index('Fecha', inplace=True)

# Ajustar el modelo Triple Exponential Smoothing
model = ExponentialSmoothing(TEScarnes['Cantidad'], trend='add', seasonal='add', seasonal_periods=12)
fit = model.fit()

# Descomponer la serie de tiempo en tendencia, ciclicidad, estacionalidad e irregularidad
tendencia = fit.trend
ciclicidad = TEScarnes['Cantidad'] - tendencia
estacionalidad = fit.season
irregularidad = TEScarnes['Cantidad'] - tendencia - estacionalidad

# Graficar las componentes descompuestas
plt.figure(figsize=(12, 10))
plt.subplot(4, 1, 1)
plt.plot(tendencia, label='Tendencia', color='blue')
plt.title('Tendencia')

plt.subplot(4, 1, 2)
plt.plot(ciclicidad, label='Ciclicidad', color='red')
plt.title('Ciclicidad')

plt.subplot(4, 1, 3)
plt.plot(estacionalidad, label='Estacionalidad', color='green')
plt.title('Estacionalidad')

plt.subplot(4, 1, 4)
plt.plot(irregularidad, label='Irregularidad', color='purple')
plt.title('Irregularidad')

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

Después, se muestran cuatro subgráficos apilados verticalmente, cada uno mostrando una de las componentes descompuestas, cada uno mostrando una de las componentes descompuestas. Estos gráficos proporcionan una visualización de cómo, cada componente contribuye a la serie de tiempo y permiten analizar la tendencia, la estacionalidad la ciclicidad y la irregularidad por separado.

El primer subgráfico muestra la tendencia, la segunda muestra la ciclicidad, la tercera la estacionalidad y por último la irregularidad.



ARIMA:

ARIMA basado en carnes.

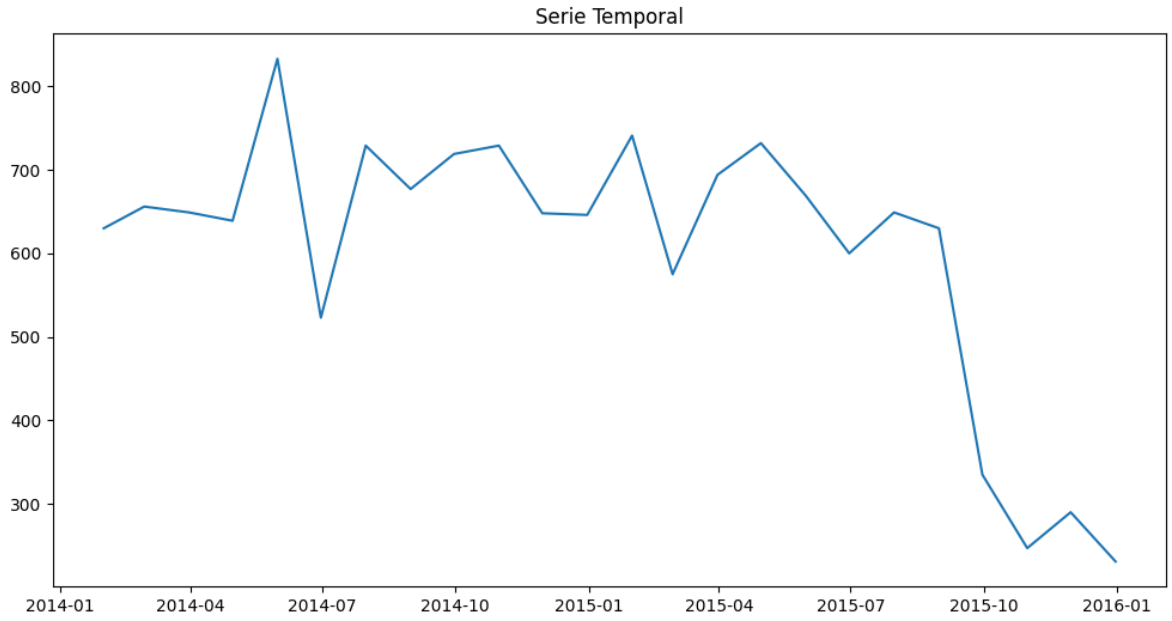
Como es costumbre importamos las librerías que siempre usamos, pero en este casi usamos ARIMA para que el código sepa que modelo usamos. Pandas para manipular los datos, numpy para realizar operaciones matemáticas y matplotlib.pylot para visualizar datos mediante la creación de gráficos y trazados.

se convierte la columna 'Fecha' en valores de tipo datetime utilizando la función pd.to_datetime().
A continuación, se establece la columna 'Fecha' como el índice del DataFrame utilizando el método set_index().

Se crea una variable ts que contiene la columna 'Cantidad' del DataFrame. Esta columna representa la cantidad de ventas de carne en cada fecha.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

# Cargar los datos desde el archivo CSV
data = pd.read_csv('ventasCarnesMes.csv')
data['Fecha'] = pd.to_datetime(data['Fecha'])
data.set_index('Fecha', inplace=True)
ts = data['Cantidad']

# Visualizar la serie temporal
plt.figure(figsize=(12, 6))
plt.plot(ts)
plt.title('Serie Temporal')
plt.show()
```

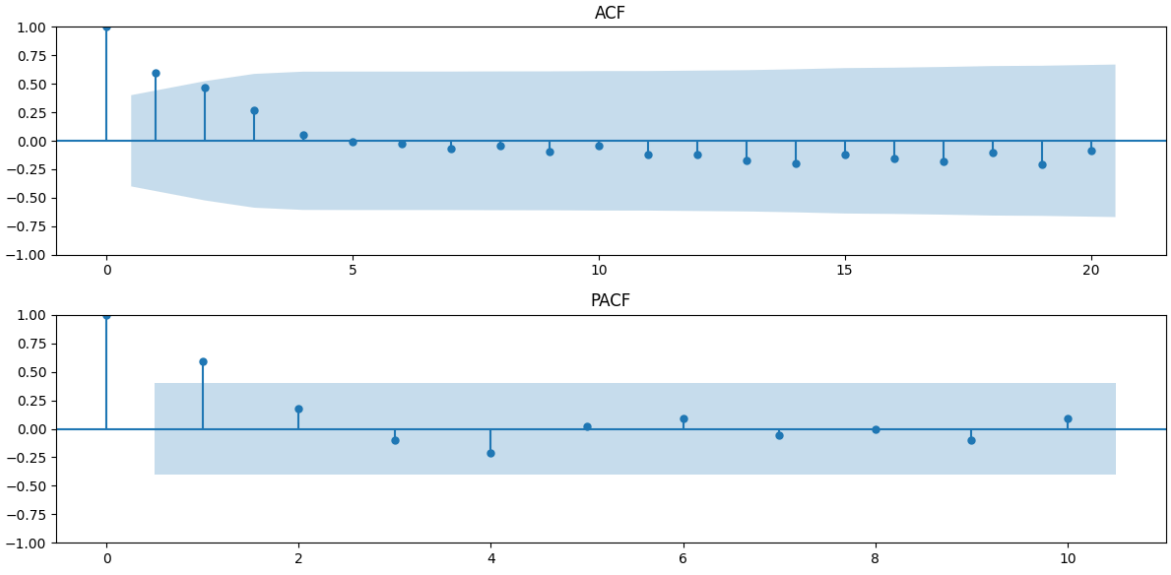
Serie temporal:



Este fragmento de código calcula y muestra dos gráficos: el gráfico de la función de autocorrelación (ACF) y el gráfico de la función de autocorrelación parcial (PACF) de la serie temporal ts. Estos gráficos son útiles para identificar posibles patrones de autocorrelación en los datos y ayudar en la selección de los parámetros del modelo ARIMA.

```
# Calcular y graficar ACF y PACF
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 6))

plot_acf(ts, lags=20, ax=ax1)
ax1.set_title('ACF')

plot_pacf(ts, lags=10, ax=ax2)
ax2.set_title('PACF')

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

Calculando y graficando ACF y PACF:



Este fragmento de código ajusta un modelo ARIMA a la serie temporal ts utilizando los valores seleccionados de 'p' y 'q'. Luego, muestra un resumen del modelo, que proporciona información sobre la calidad del ajuste y los coeficientes estimados.

```python
# Basado en los gráficos de ACF y PACF, selecciona los valores de p y q
p = 1  # Valor adecuado de p basado en el gráfico de PACF
q = 1  # Valor adecuado de q basado en el gráfico de ACF

# Entrenar el modelo ARIMA
model = ARIMA(ts, order=(p, 0, q))  # Nota: Utilizamos 0 como el valor de 'd'
results = model.fit()

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

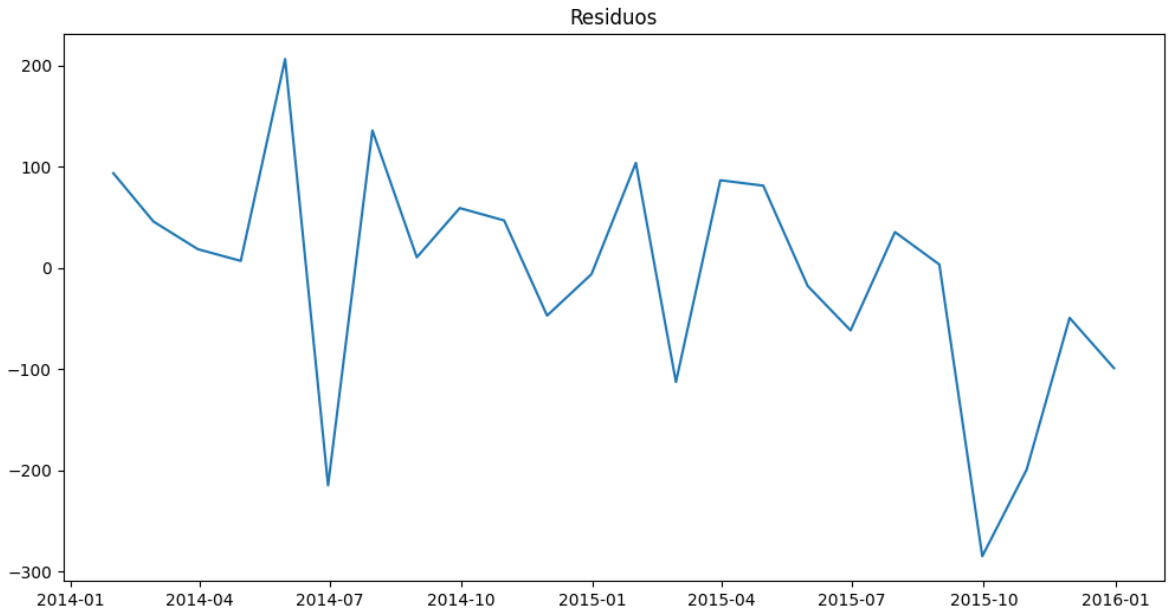
Resumen del modelo:

```
                               SARIMAX Results
==============================================================================
Dep. Variable:               Cantidad   No. Observations:                   24
Model:                 ARIMA(1, 0, 1)   Log Likelihood                -147.669
Date:                Thu, 12 Oct 2023   AIC                            303.338
Time:                        02:35:16   BIC                            308.050
Sample:                    01-31-2014   HQIC                           304.588
                         - 12-31-2015
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const        536.4486    204.962      2.617      0.009     134.731     938.166
ar.L1          0.9020      0.184      4.912      0.000       0.542       1.262
ma.L1         -0.3211      0.258     -1.246      0.213      -0.826       0.184
sigma2      1.236e+04   3712.633      3.328      0.001    5078.517    1.96e+04
===================================================================================
Ljung-Box (L1) (Q):                   0.06   Jarque-Bera (JB):                 1.97
Prob(Q):                              0.80   Prob(JB):                         0.37
Heteroskedasticity (H):               1.23   Skew:                            -0.68
Prob(H) (two-sided):                  0.78   Kurtosis:                         3.34
===================================================================================

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

Este fragmento de código muestra el gráfico de los residuos del modelo ARIMA y realiza un pronóstico utilizando el modelo entrenado. Los residuos representan la diferencia entre los valores observados y los valores predichos por el modelo. El pronóstico muestra las estimaciones para los próximos pasos en la serie temporal.

```python
# Gráfico de residuos
residuals = pd.Series(results.resid, index=ts.index)
plt.figure(figsize=(12, 6))
plt.plot(residuals)
plt.title('Residuos')
plt.show()

# Pronóstico
forecast = results.forecast(steps=10)  # Cambia 10 al número de pasos que necesites pronosticar
print("Pronóstico:", forecast)
```

Grafica de residuos



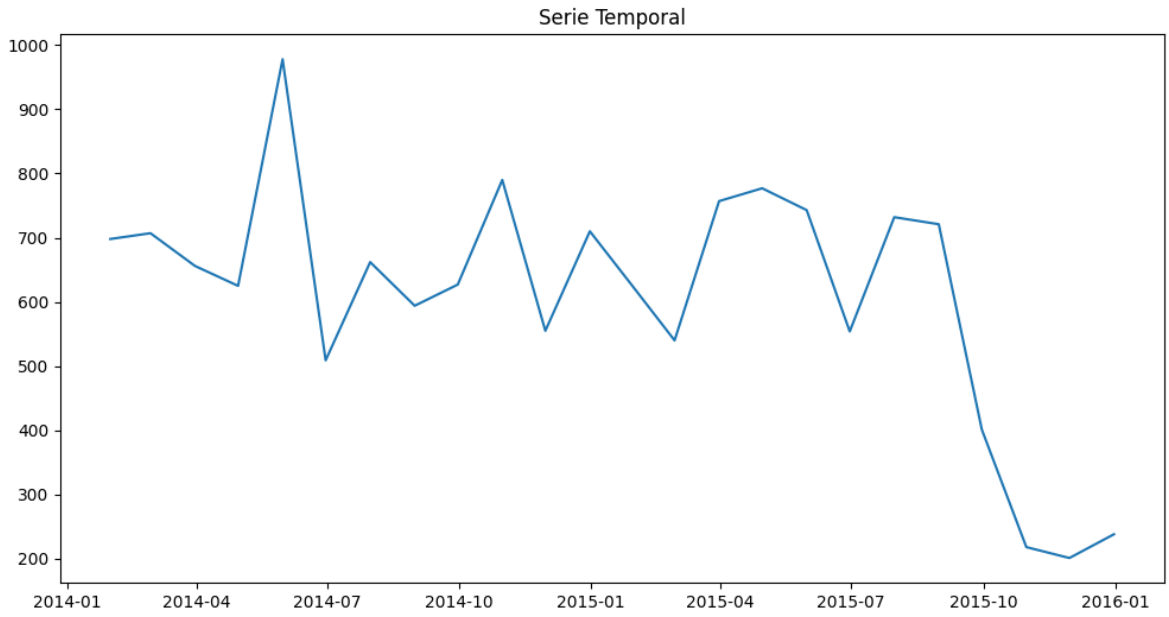
Pronostico:

```
Pronóstico: 2016-01-31    292.714384
2016-02-29    316.596076
2016-03-31    338.137780
2016-04-30    357.568774
2016-05-31    375.095869
2016-06-30    390.905616
2016-07-31    405.166283
2016-08-31    418.029654
2016-09-30    429.632640
2016-10-31    440.098735
Freq: M, Name: predicted_mean, dtype: float64
```

ARIMA basado en Quesos

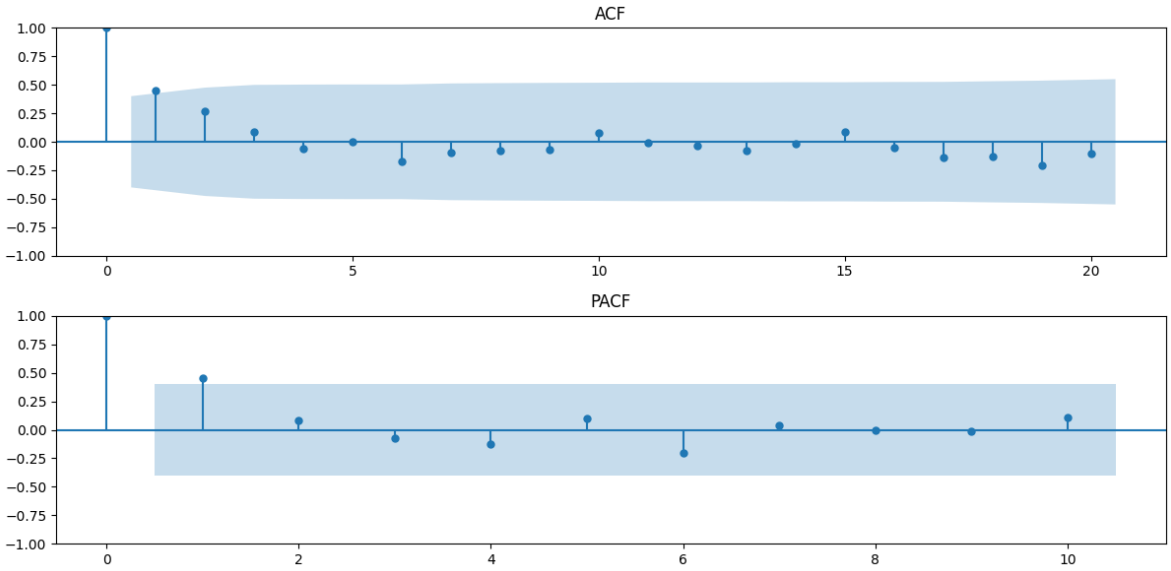
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

# Cargar los datos desde el archivo CSV
data = pd.read_csv('ventasQuesosMes.csv')
data['Fecha'] = pd.to_datetime(data['Fecha'])
data.set_index('Fecha', inplace=True)
ts = data['Cantidad']

# Visualizar la serie temporal
plt.figure(figsize=(12, 6))
plt.plot(ts)
plt.title('Serie Temporal')
plt.show()
```

Serie temporal



```
# Calcular y graficar ACF y PACF
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 6))

plot_acf(ts, lags=20, ax=ax1)
ax1.set_title('ACF')

plot_pacf(ts, lags=10, ax=ax2)
ax2.set_title('PACF')

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

Calculando y graficando ACF y PACF



Entrenando el modelo y mostrando el resumen del modelo

```
# Basado en los gráficos de ACF y PACF, selecciona los valores de p y q
p = 1  # Valor adecuado de p basado en el gráfico de PACF
q = 1  # Valor adecuado de q basado en el gráfico de ACF

# Entrenar el modelo ARIMA
model = ARIMA(ts, order=(p, 0, q))  # Nota: Utilizamos 0 como el valor de
results = model.fit()

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

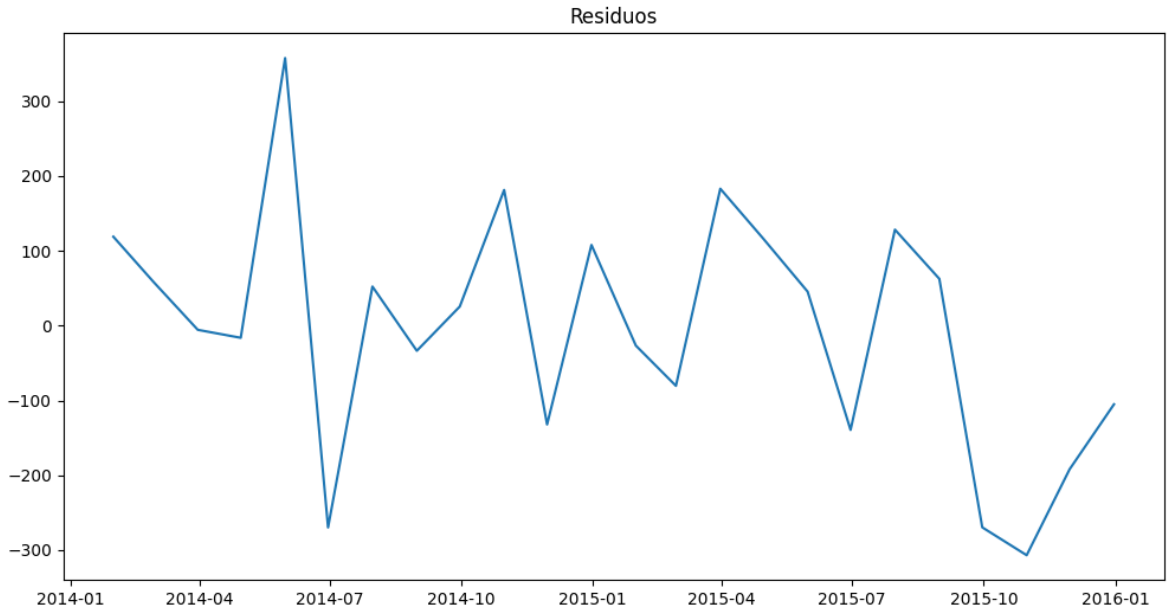
Resumen del modelo:

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                Cantidad   No. Observations:                   24
Model:                 ARIMA(1, 0, 1)   Log Likelihood                -155.633
Date:                Thu, 12 Oct 2023   AIC                            319.266
Time:                        02:38:46   BIC                            323.978
Sample:                    01-31-2014   HQIC                           320.516
                         - 12-31-2015
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const        579.0863    145.927      3.968      0.000     293.075     865.097
ar.L1          0.7853      0.477      1.645      0.100      -0.150       1.721
ma.L1         -0.3177      0.525     -0.605      0.545      -1.347       0.711
sigma2      2.461e+04   7524.651      3.271      0.001    9865.380    3.94e+04
===================================================================================
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):                 0.05
Prob(Q):                              0.94   Prob(JB):                         0.97
Heteroskedasticity (H):               1.18   Skew:                            -0.06
Prob(H) (two-sided):                  0.82   Kurtosis:                         2.80
===================================================================================
```

Graficando residuos y mostrando pronostico:

```python
# Gráfico de residuos
residuals = pd.Series(results.resid, index=ts.index)
plt.figure(figsize=(12, 6))
plt.plot(residuals)
plt.title('Residuos')
plt.show()

# Pronóstico
forecast = results.forecast(steps=10)  # Cambia 10 al número de pasos que necesites pronosticar
print("Pronóstico:", forecast)
```

Residuos:



Pronostico:

```
Pronóstico: 2016-01-31    344.624939
2016-02-29    394.957515
2016-03-31    434.485035
2016-04-30    465.527054
2016-05-31    489.905182
2016-06-30    509.049978
2016-07-31    524.084899
2016-08-31    535.892225
2016-09-30    545.164834
2016-10-31    552.446862
Freq: M, Name: predicted_mean, dtype: float64
```

Conclusiones generales.

En la realización de esta práctica implementamos diferentes técnicas para el tratamiento de datos, desde detectar nuestra variable objetivo siendo Sector, hasta definir las clases a analizar para generar series de tiempo. Por otro lado, la capacidad de generar series de tiempo es de vital importancia, aplicando un segundo tratamiento a los datos, para generar una serie de tiempo clasificada por meses. Al obtener una serie clasificada por mes de las ventas, logramos identificar casi de manera inmediata la tendencia y la estacionalidad de la

serie de tiempo. Finalmente, la aplicación de modelos nos permitió identificar características individuales de las series de tiempo mostrándonos el comportamiento de las ventas de Carnes Frías y Quesos.
Por lo que podemos concluir que las ventas de estos seguirán creciendo conforme pasen los meses, y que en épocas especiales como las de fin de año existe un crecimiento en la demanda de los mismos. Por lo que se puede obtener una predicción para años futuros y así cumplir con los objetivos planteados.

Referencias.

Siregar, B., Butar-Butar, I. A., Rahmat, R. F., Andayani, U., & Fahmi, F. (2017). Comparison of exponential smoothing methods in forecasting palm oil real production. In *Journal of Physics: Conference Series* (Vol. 801, No. 1, p. 012004). IOP Publishing.

De Arce, R., & Mahía, R. (2003). Modelos Arima. *Programa CITUS: Técnicas de Variables Financieras*, 5-6.

Esling, P., & Agon, C. (2012). Time-series data mining. *ACM Computing Surveys (CSUR)*, *45*(1), 1-34.