# Exámen Unidad#3

## Visualizaciones y análisis de datos

# APLICACIONES DE BIG DATA

Ludwicka Aguirre Meza
307102

## Introducción:

Para este examen seguiremos estos mismos pasos para diversos ejercicios:

1. **Leer los datos**: Utilizaremos la biblioteca Pandas para leer los archivos CSV.
2. **Importaremos librerías de Matplotlib**: Utilizaremos `.pyplot` y `.patches` para la creación y personalización de gráficos y figuras.
3. **Crearemos gráficos**: Utilizaremos Matplotlib para crear un gráfico diferente y específico para cada ejercicio.
4. **Agregar leyenda personalizada**: Agregaremos una leyenda que indique el color de cada categoría.

Librerías de Matplotlib:

`.pyplot` es una colección de funciones que hacen que Matplotlib funcione como MATLAB. Cada función de `pyplot` realiza algún cambio en una figura: por ejemplo, crea una figura, crea un área de trazado en una figura, dibuja algunas líneas en un área de trazado, decora el gráfico con etiquetas, etc.

Las funciones más comunes que se utilizan de `pyplot` son:

- `plt.plot()`: Para crear un gráfico de líneas.
- `plt.bar()`: Para crear un gráfico de barras.
- `plt.scatter()`: Para crear un gráfico de dispersión.
- `plt.hist()`: Para crear un histograma.
- `plt.xlabel()`, `plt.ylabel()`: Para etiquetar los ejes x e y.
- `plt.title()`: Para añadir un título al gráfico.
- `plt.legend()`: Para añadir una leyenda al gráfico.
- `plt.show()`: Para mostrar la figura o gráfico.

`.patches` contiene una colección de formas básicas que se pueden añadir a

las figuras y gráficos de Matplotlib. Estas formas (también llamadas "parches") incluyen rectángulos, círculos, polígonos, etc., y se utilizan para resaltar áreas específicas del gráfico, añadir anotaciones, o simplemente para decorar la visualización.

Algunas clases comunes en `matplotlib.patches` son:

- `mpatches.Rectangle`: Para crear y añadir un rectángulo.
- `mpatches.Circle`: Para crear y añadir un círculo.
- `mpatches.Polygon`: Para crear y añadir un polígono.
- `mpatches.Ellipse`: Para crear y añadir una elipse.
- `mpatches.FancyBboxPatch`: Para crear y añadir cajas con bordes decorativos.

En este examen se utilizó mpatches.Patch, el cual crea un parche (patch) de Matplotlib con un color y una etiqueta específicos. Un parche en este contexto es simplemente una forma gráfica que puede ser representada en una leyenda de un gráfico. La clase Patch es una clase base para todas las formas en matplotlib.patches, y aquí se está utilizando para crear rectángulos de color que representan diferentes categorías o elementos en la leyenda.

Ahora después de un poco de contexto, el examen se dividirá en cada ejercicio analizando el problema y a través del código ya comentado llevaremos a cabo el desarrollo del problema.

# Ejercicio 1 – Comparación de Algoritmos

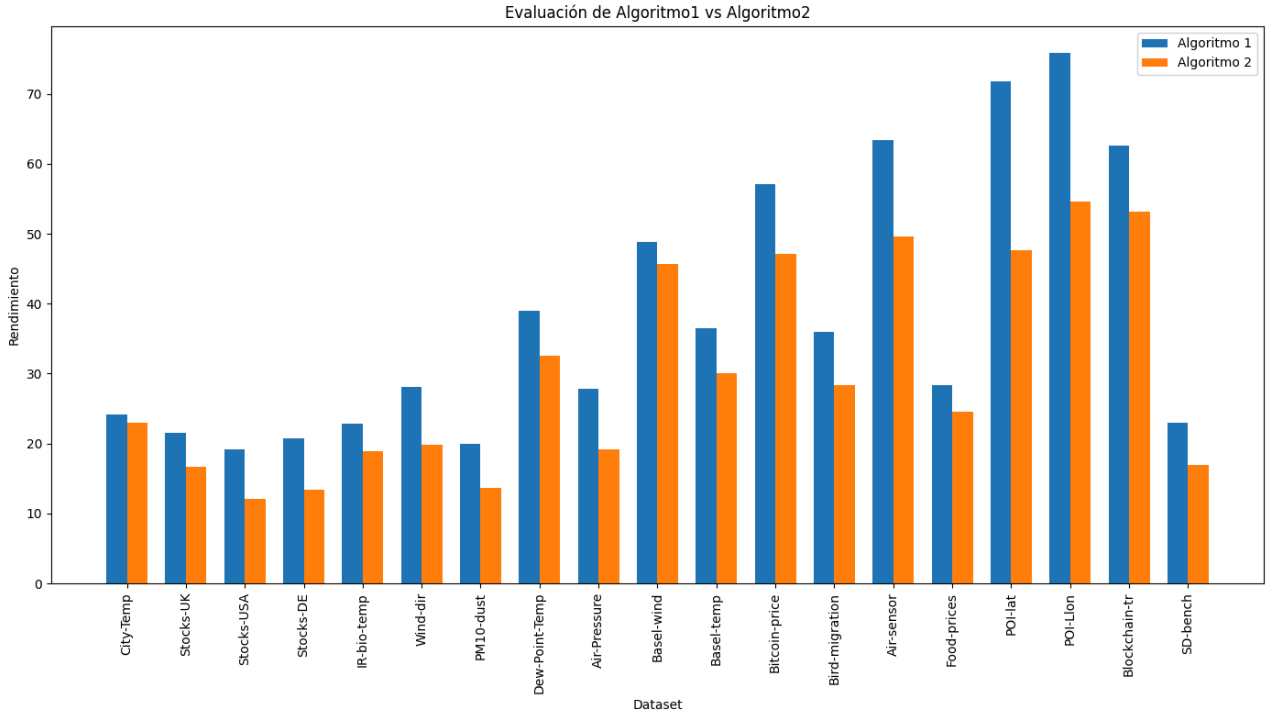
## Análisis del problema

El objetivo de este problema es comparar el rendimiento de dos algoritmos de compresión utilizando un conjunto de datos. El archivo `comparasion_algoritmos.csv` contiene los nombres de estos conjuntos de datos y el rendimiento correspondiente de cada algoritmo. Queremos visualizar esta comparación mediante un gráfico de barras.


| | dataset | algoritmo1 | algoritmo2 |
|---|---|---|---|
| 0 | City-Temp | 24.17 | 22.92 |
| 1 | Stocks-UK | 21.51 | 16.70 |
| 2 | Stocks-USA | 19.22 | 12.06 |
| 3 | Stocks-DE | 20.78 | 13.46 |
| 4 | IR-bio-temp | 22.90 | 18.94 |
| 5 | Wind-dir | 28.15 | 19.80 |
| 6 | PM10-dust | 19.92 | 13.68 |
| 7 | Dew-Point-Temp | 39.03 | 32.49 |
| 8 | Air-Pressure | 27.89 | 19.23 |
| 9 | Basel-wind | 48.87 | 45.65 |
| 10 | Basel-temp | 36.51 | 30.12 |

```python
#Configuramos el tamaño de la figura
plt.figure(figsize=(14, 8))
#Creamos la gráfica de barras
bar_width = 0.35
index = range(len(df['dataset']))
#Declaramos las barras para el algoritmo1
plt.bar(index, df['algoritmo1'], bar_width, label='Algoritmo 1')
#Declaramos las barras para el algoritmo2
plt.bar([i + bar_width for i in index], df['algoritmo2'], bar_width, label='Algoritmo 2')
#Configuramos los ejes y las etiquetas
plt.xlabel('Dataset')
plt.ylabel('Rendimiento')
plt.title('Evaluación de Algoritmo1 vs Algoritmo2')
plt.xticks([i + bar_width / 2 for i in index], df['dataset'], rotation=90)
plt.legend()
#Mostramos la gráfica
plt.tight_layout()
plt.show()
```

0.3s



# Ejercicio 2 - Comparación de categorías

## Análisis del problema

En este ejercicio, queremos visualizar cómo se distribuyen las categorías dentro de diferentes conjuntos de datos. El archivo `datasets_top_4.csv` contiene el nombre de los conjuntos de datos y el porcentaje de datos que pertenecen a cada una de las cuatro categorías. Queremos generar un gráfico de barras apiladas para mostrar esta distribución.

## 2. Comparación de categorías

```
#Realizamos la lectura de los datos
df = pd.read_csv('./datasets_top_4.csv')
df
```

[36] 0.0s

| | dataset | cat1 | cat2 | cat3 | otros |
|---|---|---|---|---|---|
| 0 | arade4 | 49.759495 | 20.770575 | 6.280610 | 23.189320 |
| 1 | basel_temp_f | 73.695473 | 22.569984 | 0.291341 | 3.443202 |
| 2 | basel_wind_f | 67.390949 | 24.599608 | 2.985026 | 5.024417 |
| 3 | bird_migration_f | 44.973576 | 28.504147 | 13.051469 | 13.470807 |
| 4 | bitcoin_f | 68.701150 | 7.519535 | 1.025393 | 22.753922 |
| 5 | bitcoin_transactions_f | 59.006077 | 10.651040 | 7.901476 | 22.441407 |
| 6 | city_temperature_f | 55.766159 | 18.470318 | 15.352960 | 10.410563 |
| 7 | cms1 | 50.301599 | 11.414799 | 4.405325 | 33.878277 |
| 8 | cms25 | 8.793950 | 3.629491 | 1.931263 | 85.645296 |
| 9 | cms9 | 99.889249 | 0.073619 | 0.017582 | 0.019549 |
| 10 | food_prices | 63.307736 | 17.856899 | 8.406486 | 10.428878 |

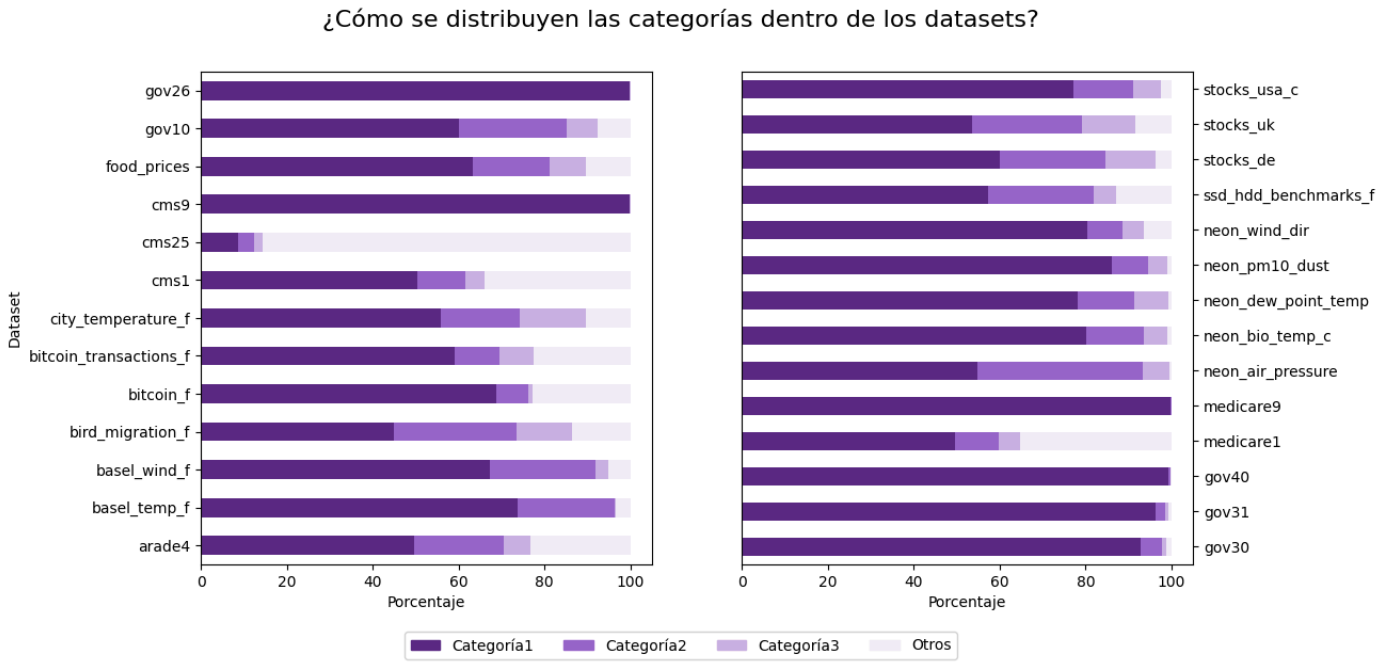
```python
# Dividimos el dataframe en dos partes
n = len(df) // 2
df_p1 = df.iloc[:n]
df_p2 = df.iloc[n:]
```

0.0s

```python
#Creamos la figura y dos subgráficos con tamaño ajustado
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 6))

#Creamos el primer gráfico de barras apiladas
df_p1.plot.barh(
    stacked=True,
    color=['#5A2C84', '#9A65CC', '#CCB2E5', '#F2EBF8'],
    xlabel='Porcentaje',
    ylabel='Dataset',
    ax=ax1
)

#Configuramos los nombres de los datasets en el eje y para la parte 1
ax1.set_yticklabels(df_p1['dataset'])

#Creamos el primer gráfico de barras apiladas horizontalmente en el segundo subgráfico
df_p2.plot.barh(
    stacked=True,
    color=['#5A2C84', '#9A65CC', '#CCB2E5', '#F2EBF8'],
    xlabel='Porcentaje',
    ax=ax2
)
```

```
)

#Configuramos los nombres de los datasets en el eje y para la parte 2
ax2.set_yticklabels(df_p2['dataset'])

#Eliminamos la leyenda de colores de cada subgráfico individual para mejor presentación visual
ax1.get_legend().remove()
ax2.get_legend().remove()

#Creamos una leyenda personalizada para los colores de las categorías en la figura
legend_labels = ['Categoría1', 'Categoría2', 'Categoría3', 'Otros']
legend_colors = ['#5A2C84', '#9A65CC', '#CCB2E5', '#F2EBF8']
legend_patches = [mpatches.Patch(color=color, label=label) for color, label in zip(legend_colors, legend_labels)]

#Agregamos la leyenda a la figura como un texto adicional
fig.legend(handles=legend_patches, loc='lower center', bbox_to_anchor=(0.5, -0.05), ncol=4)

#Añadimos un título para ambas subgráficas
fig.suptitle('¿Cómo se distribuyen las categorías dentro de los datasets?', fontsize=16)

#Posicionamos los nombres de los datasets a la derecha en el gráfico de la derecha
ax2.yaxis.tick_right()

#Mostrar la figura
plt.show()
```

0.4s



# Ejercicio 3 – Surfistas

## Análisis del problema

En este ejercicio, queremos visualizar el porcentaje de surfistas por país de procedencia. El archivo `viajes_surfistas.csv` contiene datos de diferentes surfistas, incluido su país de procedencia. Queremos generar un gráfico de tarta para mostrar cómo se distribuyen los surfistas entre los países.

3. Surfistas

```
#Realizamos la lectura de los datos
df = pd.read_csv('./viajes_surfistas.csv')
df
```

[39] ✓ 0.0s

...

| | homename | homecontinent | homecountry | homelat | homelon | travelcontinent |
|---|---|---|---|---|---|---|
| 0 | Canadian in La La Land | North America | United States of America | 33.930030 | -118.280993 | North America |
| 1 | Canadian in La La Land | North America | United States of America | 33.930030 | -118.280993 | North America |
| 2 | Canadian in La La Land | North America | United States of America | 33.930030 | -118.280993 | North America |
| 3 | Canadian in La La Land | North America | United States of America | 33.930030 | -118.280993 | North America |
| 4 | Canadian in La La Land | North America | United States of America | 33.930030 | -118.280993 | North America |
| ... | ... | ... | ... | ... | ... | ... |
| 9506 | Bridgetown, Western Australia | Australia | Australia | -33.954448 | 116.131214 | Australia |
| 9507 | Lille, France | Europe | France | 50.629250 | 3.057256 | Europe |
| 9508 | MX | South America | Mexico | 23.634501 | -102.552784 | South America |
| 9509 | Kuta - Bali - Indonesia | Asia | Indonesia | -8.739184 | 115.171130 | Asia |
| 9510 | Kuta - Bali - Indonesia | Asia | Indonesia | -8.739184 | 115.171130 | Asia |

9511 rows × 9 columns

```python
#Contamos el número de surfistas por país de procedencia
surfistas_por_pais = df['homecountry'].value_counts()
surfistas_por_pais
```

0.0s

```
homecountry
United States of America    2826
Japan                        893
Australia                    809
Spain                        752
United Kingdom               496
                            ...
Grenada                        1
Kenya                          1
Fiji                           1
Pakistan                       1
Andorra                        1
Name: count, Length: 101, dtype: int64
```

```python
#Calculamos el porcentaje de surfistas por país
porcentaje_surfistas_por_pais = (surfistas_por_pais / surfistas_por_pais.sum()) * 100
porcentaje_surfistas_por_pais
```

0.0s

```
homecountry
United States of America    29.712964
Japan                        9.389128
Australia                    8.505940
Spain                        7.906634
United Kingdom               5.215014
                               ...
Grenada                      0.010514
Kenya                        0.010514
Fiji                         0.010514
```

```
#Seleccionamos los cuatro países con los porcentajes más altos
top_paises = porcentaje_surfistas_por_pais.nlargest(4)
top_paises
```

[2] 0.0s

```
homecountry
United States of America    29.712964
Japan                        9.389128
Australia                    8.505940
Spain                        7.906634
Name: count, dtype: float64
```

```
#Calculamos la suma de los porcentajes de los demás países
otros_paises = porcentaje_surfistas_por_pais.drop(top_paises.index).sum()
otros_paises
```

[3] 0.0s

```
44.48533277257911
```

```
#Reseteamos el índice del dataframe
otros_serie = pd.Series({ 'Otros': otros_paises })
otros_serie
```

[4] 0.0s

```
Otros    44.485333
dtype: float64
```

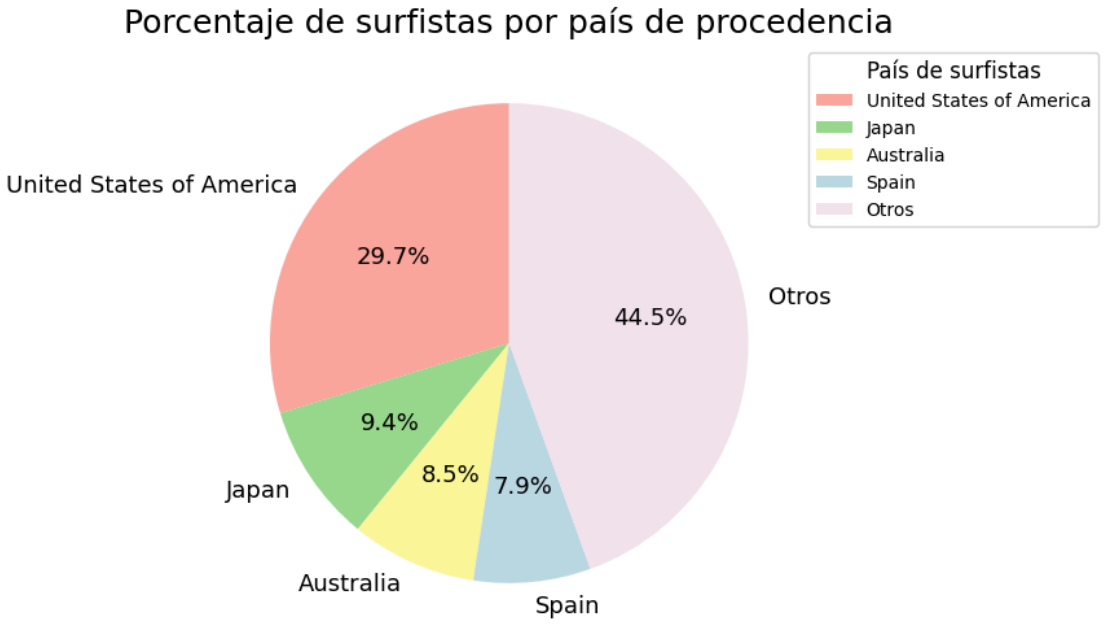
```
#Eliminamos la columna 'count'
catFinales = pd.concat([top_paises, otros_serie], axis=0)
catFinales
```

45] 0.0s

```
United States of America    29.712964
Japan                        9.389128
Australia                    8.505940
Spain                        7.906634
Otros                       44.485333
dtype: float64
```

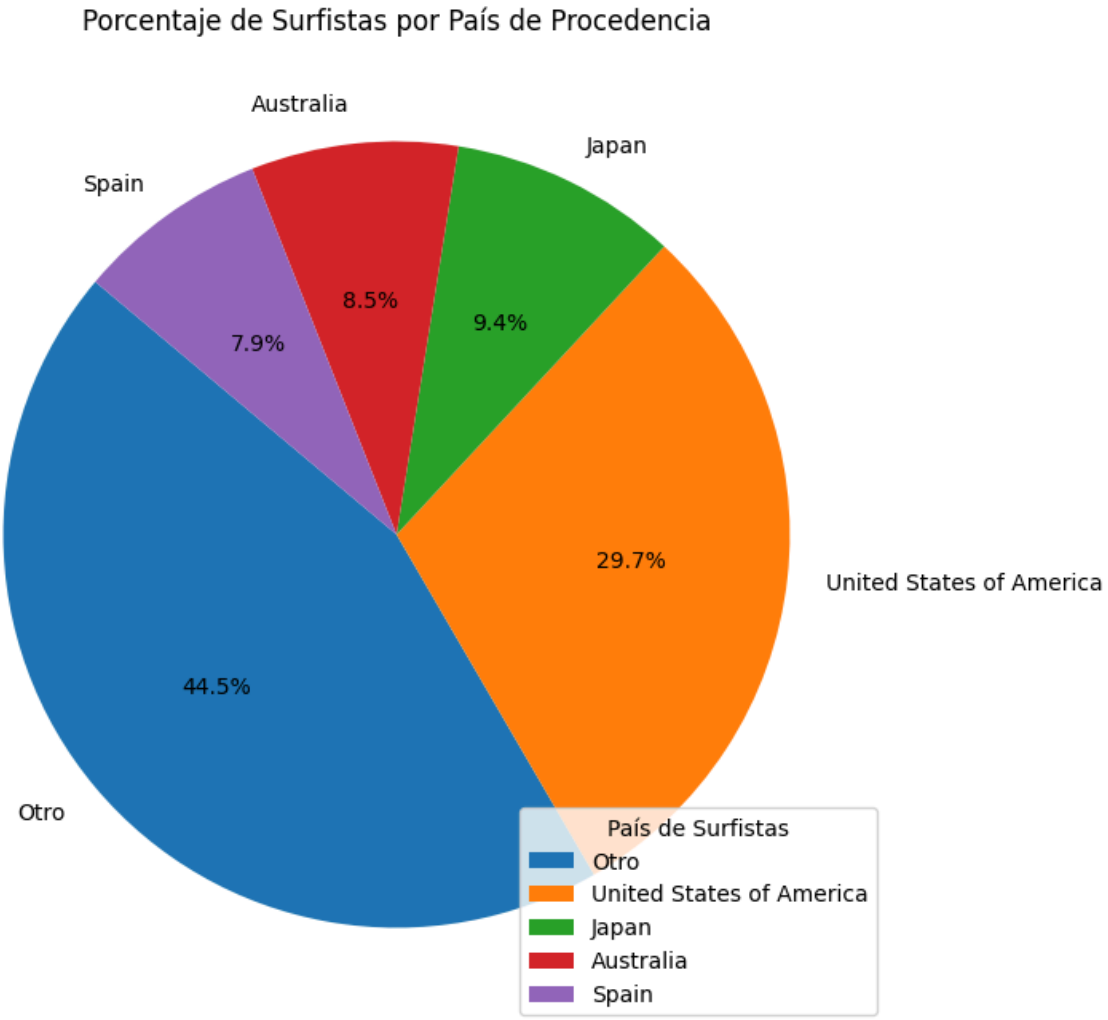
```
#Realizamos una lista con los colores deseados para la gráfica
colores_personalizados = ['#FAA59D', '#9AD78D', '#FEF999', '#BDD8E5', '#F3E1ED']

#Creamos un gráfico de pastel con los colores personalizados
catFinales.plot.pie(autopct='%1.1f%%',
                    figsize=(10, 6),
                    legend=True,
                    startangle=90,
                    colors=colores_personalizados,
                    fontsize='13',
                   )

#Ajustamos el tamaño del título
plt.title('Porcentaje de surfistas por país de procedencia', fontsize='18')

#Ajustamos la posición de la leyenda, definimos su titulo y el tamaño de letra
plt.legend(title="País de surfistas", loc="upper right", title_fontsize='12', bbox_to_anchor=(1.5, 1))

#Mostramos el gráfico
plt.show()
```

46] 0.1s



```python
#Otra forma para realizar el gráfico de pastel podría ser la siguiente:

#Reemplazamos los países que no son "United States of America", "Japan", "Australia" o "Spain" con "Otro"
df.loc[~df['homecountry'].isin(['United States of America', 'Japan', 'Australia', 'Spain']), 'homecountry'] = 'Otro'

#Contamos las ocurrencias de cada país en la columna 'homecountry'
country_counts = df['homecountry'].value_counts()

#Creamos el gráfico de pastel
plt.figure(figsize=(10, 8))
plt.pie(country_counts, labels=country_counts.index, autopct='%1.1f%%', startangle=140)
plt.title('Porcentaje de Surfistas por País de Procedencia')
plt.legend(title='País de Surfistas', loc='lower right')

#Mostramos el gráfico
plt.show()
```

0.1s



# Ejercicio 4 – Ventas

## Análisis del problema

El objetivo es analizar las ventas realizadas en diferentes tiendas a lo largo del tiempo y determinar cuáles son las tiendas que han tenido un mayor número de ventas históricamente. Utilizaremos el archivo `tienda_ventas.csv`, que contiene datos de ventas con información sobre la fecha de la venta, la tienda y el monto de la venta.

4. Ventas

```
#Realizamos la lectura de los datos
df = pd.read_csv('./tienda_ventas.csv')
df
```

[48] ✓ 1.0s

...

| | id | date | store_nbr | family |
|---|---|---|---|---|
| 0 | 0 | 2013-01-01 | 1 | AUTOMOTIVE |
| 1 | 1 | 2013-01-01 | 1 | BABY CARE |
| 2 | 2 | 2013-01-01 | 1 | BEAUTY |
| 3 | 3 | 2013-01-01 | 1 | BEVERAGES |
| 4 | 4 | 2013-01-01 | 1 | BOOKS |
| ... | ... | ... | ... | ... |
| 2161561 | 2161561 | 2016-04-30 | 9 | POULTRY |
| 2161562 | 2161562 | 2016-04-30 | 9 | PREPARED FOODS |
| 2161563 | 2161563 | 2016-04-30 | 9 | PRODUCE |
| 2161564 | 2161564 | 2016-04-30 | 9 | SCHOOL AND OFFICE SUPPLIES |
| 2161565 | 2161565 | 2016-04-30 | 9 | SEAFOOD |

2161566 rows × 6 columns

```
#Obtenemos las 5 tiendas agrupando desde family, con la suma más alta de ventas
top_families = df.groupby('family')['sales'].sum().nlargest(5).index
top_families
```

49] 0.1s

```
Index(['GROCERY I', 'BEVERAGES', 'CLEANING', 'PRODUCE', 'DAIRY'], dtype='object', na
```

```
#Convertimos la columna 'date' a tipo datetime para agrupar por mes
df['date'] = pd.to_datetime(df['date'])

#Agregamos una columna para el mes y el año
df['year_month'] = df['date'].dt.to_period('M')

#Filtramos las ventas solo de las 5 tiendas principales
dataVentas_top_families = df[df['family'].isin(top_families)]
dataVentas_top_families
```

60] 0.4s

| | id | date | store_nbr | family | sales | onpromotion | year_mor |
|---|---|---|---|---|---|---|---|
| 3 | 3 | 2013-01-01 | 1 | BEVERAGES | 0.000 | 0 | 2013- |
| 7 | 7 | 2013-01-01 | 1 | CLEANING | 0.000 | 0 | 2013- |
| 8 | 8 | 2013-01-01 | 1 | DAIRY | 0.000 | 0 | 2013- |
| 12 | 12 | 2013-01-01 | 1 | GROCERY I | 0.000 | 0 | 2013- |

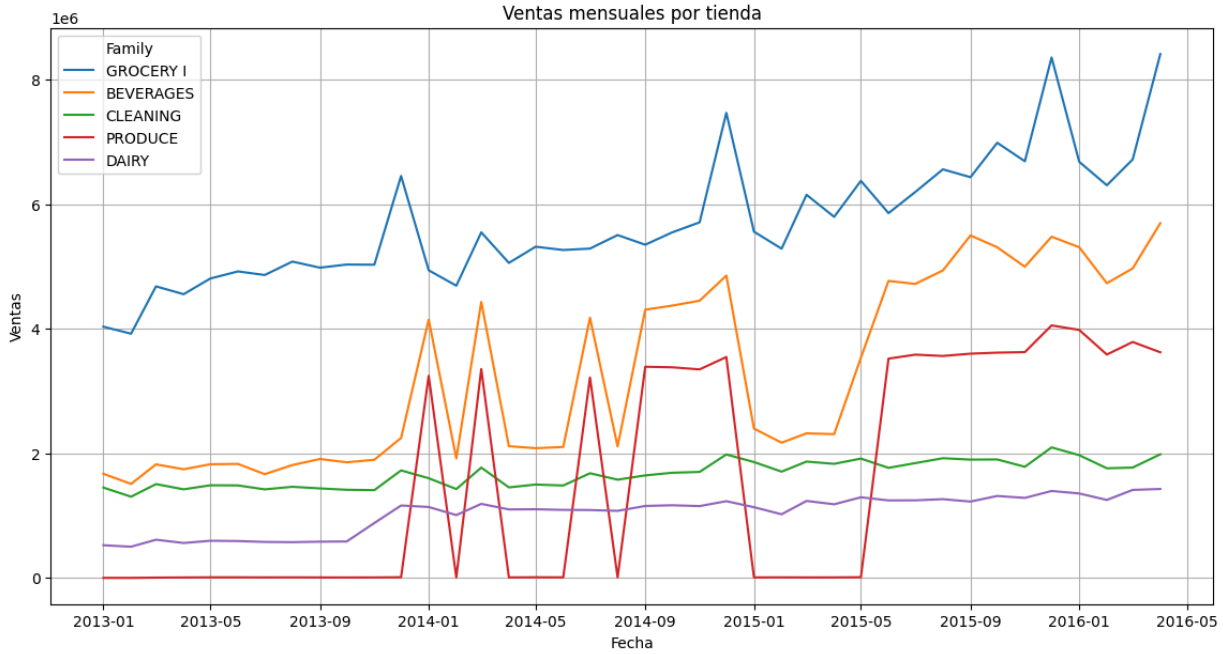
```
#Agrupamos y sumamos las ventas por tienda en intervalos de 1 mes
ventas_mensuales = dataVentas_top_families.groupby(['family', 'year_month'])['sales'].sum().reset_index()

#Convertimos 'year_month' a una fecha de tipo datetime para la gráfica
ventas_mensuales['year_month'] = ventas_mensuales['year_month'].dt.to_timestamp()
ventas_mensuales
```

1] 0.0s

| | family | year_month | sales |
|---|---|---|---|
| 0 | BEVERAGES | 2013-01-01 | 1.670653e+06 |
| 1 | BEVERAGES | 2013-02-01 | 1.508254e+06 |
| 2 | BEVERAGES | 2013-03-01 | 1.822220e+06 |
| 3 | BEVERAGES | 2013-04-01 | 1.743121e+06 |
| 4 | BEVERAGES | 2013-05-01 | 1.824123e+06 |
| ... | ... | ... | ... |
| 195 | PRODUCE | 2015-12-01 | 4.052565e+06 |

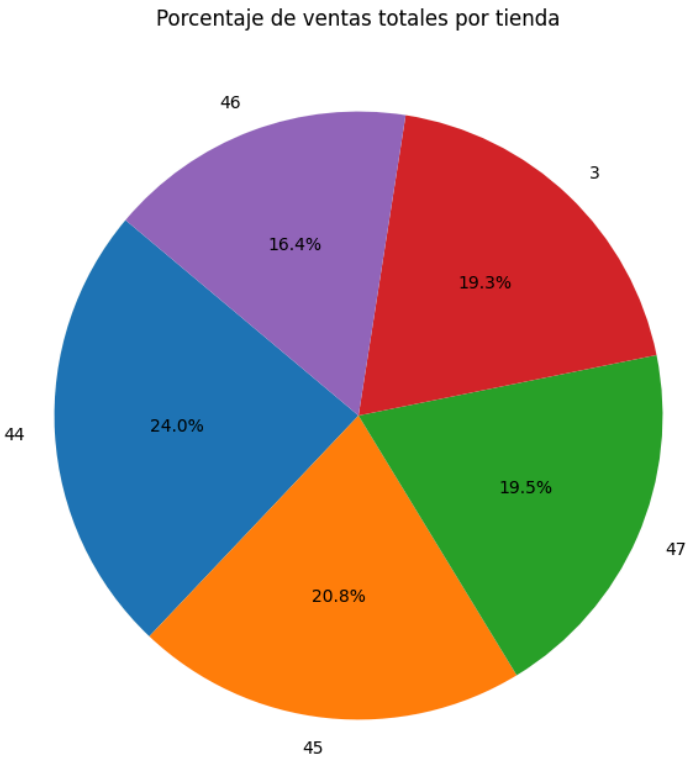
```python
#Creamos la gráfica de líneas
plt.figure(figsize=(14, 7))

for family in top_families:
    family_data = ventas_mensuales[ventas_mensuales['family'] == family]
    plt.plot(family_data['year_month'], family_data['sales'], label=family)

plt.xlabel('Fecha')
plt.ylabel('Ventas')
plt.title('Ventas mensuales por tienda')
plt.legend(title='Family')
plt.grid(True)
plt.show()
```

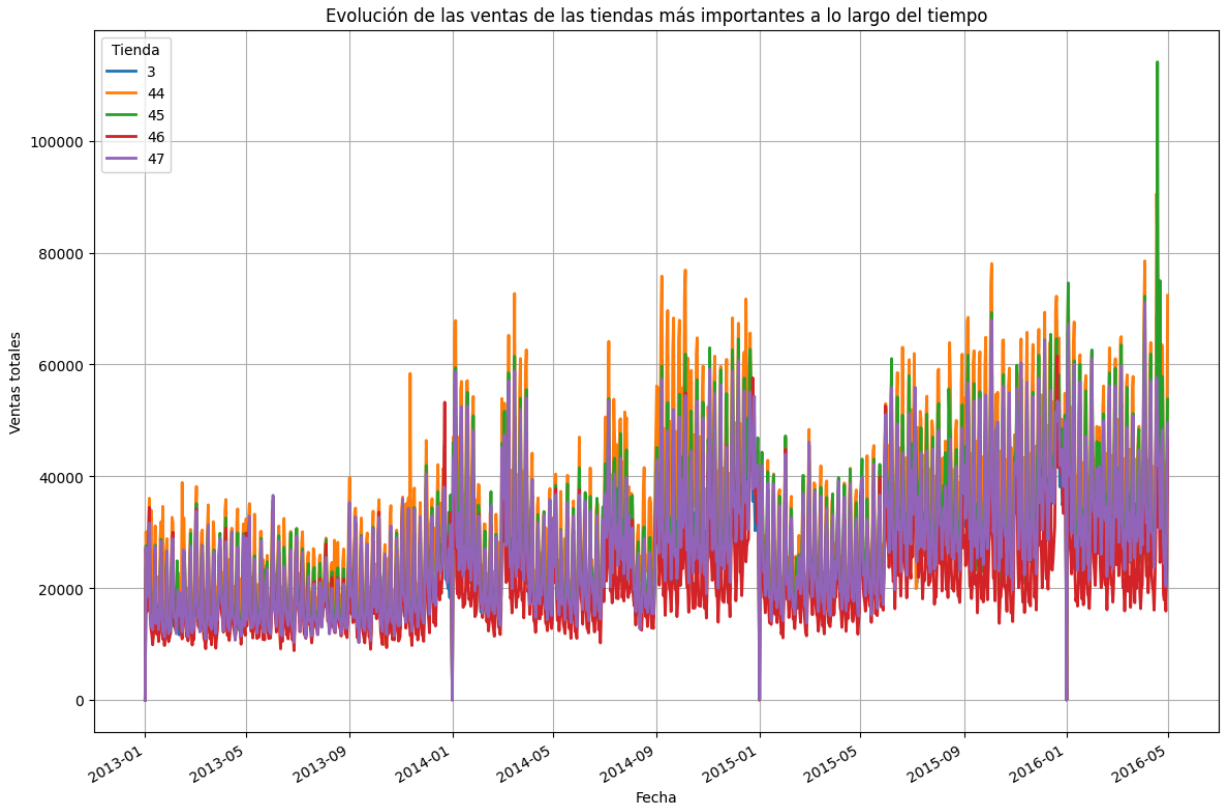
0.4s



```python
#Otra forma de hacerlo podría ser la siguiente:

#Agrupamos por tienda y sumar las ventas
total_sales_by_store = df.groupby('store_nbr')['sales'].sum()

#Identificamos las tiendas con las ventas totales más altas
top_stores = total_sales_by_store.nlargest(5)

#Filtramos los datos para las tiendas con las ventas totales más altas
top_stores_data = df[df['store_nbr'].isin(top_stores.index)]

#Agrupamos los datos filtrados por fecha y tienda y sumar las ventas
sales_by_date_store = top_stores_data.groupby(['date', 'store_nbr'])['sales'].sum().unstack()

#Creamos un gráfico de pastel para mostrar las ventas totales de las tiendas más importantes
plt.figure(figsize=(10, 8))
plt.pie(top_stores, labels=top_stores.index, autopct='%1.1f%%', startangle=140)
plt.title('Porcentaje de ventas totales por tienda')

#Mostramos el gráfico
plt.show()

#Creamos un gráfico de líneas para mostrar la evolución de las ventas de las tiendas más importantes a lo largo del tiempo
sales_by_date_store.plot(figsize=(12, 8), linewidth=2)  #Aumentamos el grosor de las líneas
plt.xlabel('Fecha')
plt.ylabel('Ventas totales')
plt.title('Evolución de las ventas de las tiendas más importantes a lo largo del tiempo')
plt.legend(title='Tienda', loc='upper left')  #Cambiamos la ubicación de la leyenda
plt.grid(True)  # Agregamos una cuadrícula para una mejor referencia visual
plt.tight_layout()  #Ajustamos el diseño para evitar superposiciones

#Mostramos el gráfico
plt.show()
```

0.6s






Para consultar el código puedes acceder al siguiente link: https://github.com/LudAcrist/ExamenU3BD