
SKILLING
CENTER
TECMILENIO

SKILLING CENTER TECMILENIO

Visualización y Programación en Python

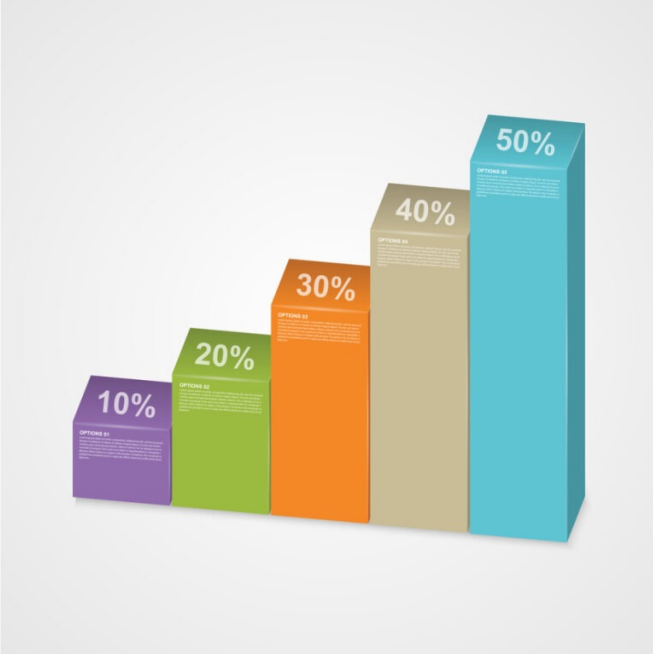
Uso de gráficos en Python

Introducción

Los gráficos son una herramienta poderosa para comunicar información de forma clara y efectiva. Al elegir el tipo de gráfico adecuado y seguir algunos consejos básicos, se pueden crear gráficos que sean informativos y fáciles de comprender.

Imagina que estás preparando una presentación con los resultados de un modelo analítico que ayuda a hacer más eficiente la gestión de logística de tu empresa. Has detectado áreas de oportunidad muy interesantes, y quieres comunicar los resultados de manera que refleje adecuadamente las propuestas y hallazgos que has obtenido y quieres comunicar.

¿Qué tipos de gráficos puedes realizar con Python y cuáles son los adecuados para tu presentación?



Explicación

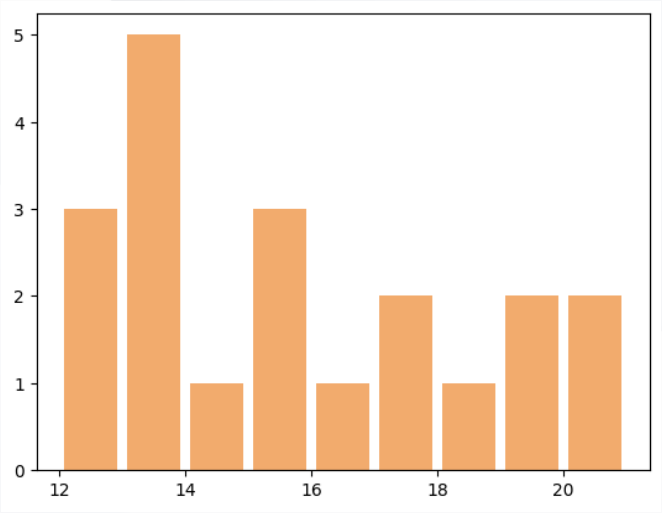
- La visualización de los datos te permite interpretar y confirmar tus hallazgos, además de comunicar los resultados.
- Es muy común preparar y compartir *dashboards* con texto, tablas y gráficos.
- En Python, además de analizar datos, puedes generar gráficos para diferentes propósitos.

Explicación

Histogramas

- Muestran la distribución de datos con base en la frecuencia de rangos específicos.
- Una librería muy común para graficar en Python es Matplotlib.
- Como ejemplo, se quiere graficar la frecuencia de edades de alumnos de un salón de clases:
  - **import matplotlib.pyplot as plt # se importa el módulo pyplot de la librería matplotlib**
  - **edades = [12, 13, 12, 18, 19, 20, 13, 12, 17, 15, 20, 16, 13, 14, 13, 17, 15, 19, 13, 15]**
  - **intervalos = range(min(edades), max(edades) + 2) # se calculan los extremos de los intervalos**
  - **plt.hist(x=edades, bins=intervalos, color='#F2AB6D', rwidth=0.85)**
  - **plt.show() # genera el histograma**

:



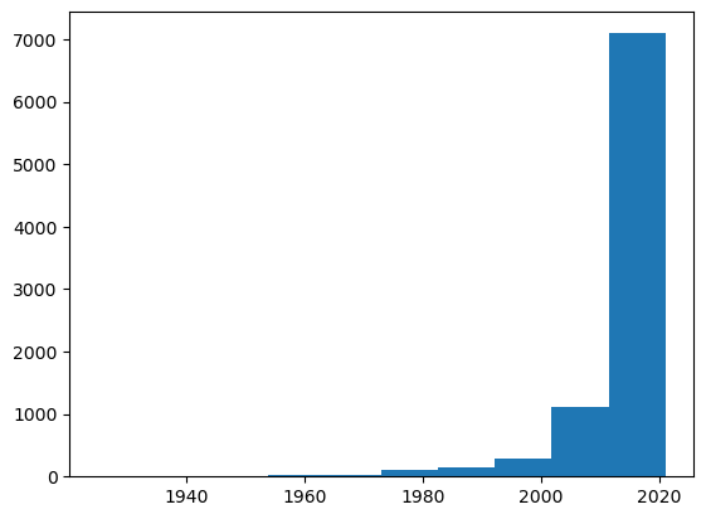
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Explicación

En caso de que los datos provengan de un archivo, se utiliza la librería Pandas para leerlo y en seguida se realiza el histograma. Por ejemplo, al archivo 'netflix_titles' tiene los datos de los lanzamientos de nuevos programas en la columna 'release_year', lo cual se va a graficar como histograma.

- **Import matplotlib.pyplot as plt**
- **Import pandas as pd**
- **df = pd.read_csv('Netflix_titles.csv')**
- **plt.hist(df['release_year])**
- **plt.show()**

Y en la gráfica de salida se puede observar que los rangos de fechas son de 10 años, se estrenaron 1000 programas entre 2000 y 2010, y se estrenaron 7000 programas entre 2010 y 2020.

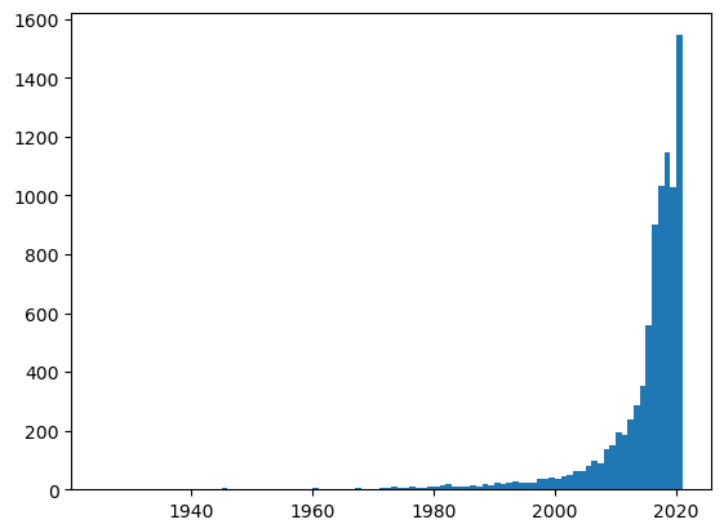


Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Explicación

Con el parámetro ‘bins’ se puede modificar el rango de las barras. Para el ejemplo, se va establecer el incremento de 1 año, y se utilizará la librería Numpy para generar la serie en incrementos de 1 año.

- **import matplotlib.pyplot as plt**
- **import pandas as pd**
- **import numpy as np**
- **df = pd.read_csv('netflix_titles.csv’)**
- **anio = df['release_year’]**
- **plt.hist(anio, bins = np.arrange(min(anio), max(anio) + 1, 1))**
- **plt.show()**

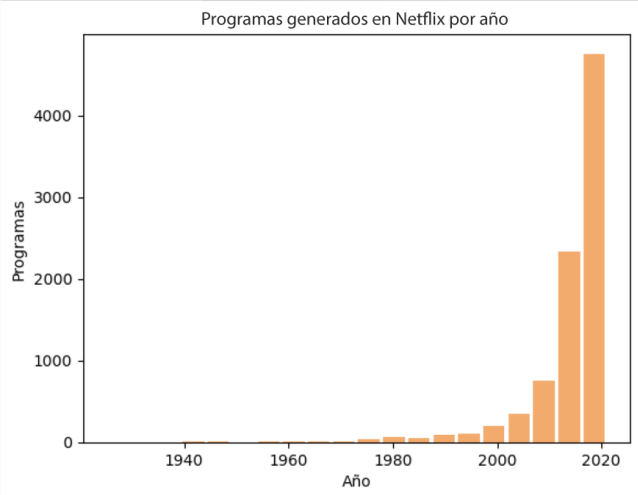


Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Explicación

Por otro lado, es posible fijar la cantidad de intervalos, por ejemplo, en 20, estableciendo la variable bins como bins = 20. Adicionalmente, se completa la gráfica agregando el título y los nombres a los ejes horizontal y vertical.

- **import matplotlib.pyplot as plt**
- **import pandas as pd**
- **import numpy as np**
- **df = pd.read_csv('netflix_titles.csv')**
- **anio = df['release_year']**
- **plt.hist(anio, bins = 20, color='#F2AB6D', rwidth=0.85)**
- **plt.title('Programas generados en Netflix por año')**
- **plt.ylabel('Programas') # etiqueta del eje y**
- **plt.xlabel('Año') # etiqueta del eje x**
- **plt.show()**



Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Explicación

Gráficas de series de tiempo

Este tipo de gráficas son muy empleadas en negocios, finanzas, metrología, monitoreo de desempeño de procesos, de salud, entre muchos otros. Representan el comportamiento de una o más variables en el tiempo. Prabhakaran (2019) menciona que se puede considerar que una serie de tiempo es una secuencia de observaciones registradas a intervalos de tiempo regulares.
Como ejemplo, considera analizar los datos de las rentas de bicicletas en la plataforma Ecobici de la Ciudad de México.

Se importan las librerías y se prepara el estilo de gráfico.

- **import pandas as pd**
- **import matplotlib.pyplot as plt**
- **import seaborn as sns**
- **sns.set_style("darkgrid") # estilo de salida de las gráficas**
- **from datetime import datetime**

Se importa los datos de las rentas del sitio oficial del programa Ecobici y se guarda en un archivo de Excel, para luego leerlo en Python:

- **viajes = pd.read_csv('2018-10.csv')**
- **viajes.head()**

Con la función head() se muestran los primeros cinco renglones de la tabla:

Explicación

```
viaje = pd.read_csv('2018-10.csv')
viaje.head()
```

✓ 2.0s

	Genero_Usuario	Edad_Usuario	Bici	Ciclo_Estacion_Retiro	Fecha_Retiro	Hora_Retiro	Ciclo_Estacion_Arribo	Fecha_Arribo	Hora_Arribo
0	M	58	2176	299	01/10/2018	0:00:02	75	01/10/2018	0:12:02
1	F	42	9816	43	01/10/2018	0:00:23	256	01/10/2018	0:15:02
2	M	30	11219	37	01/10/2018	0:00:34	1	01/10/2018	0:09:09
3	M	26	3910	123	01/10/2018	0:01:36	182	01/10/2018	0:37:22
4	M	29	11487	41	01/10/2018	0:02:39	30	01/10/2018	0:11:29

Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Para obtener el número de viajes es necesario establecer un índice temporal con la fecha y la hora. También es importante cambiar el formato a la fecha de *string* a formato *datetime*.

```
# concatenar Hora_Retiro y Fecha_Retiro
viaje['fecha_hora_retiro'] = viaje.Fecha_Retiro + ' ' + viaje.Hora_Retiro

# cambiar de str a datetime
viaje['fecha_hora'] = viaje.fecha_hora_retiro \
                          .map(lambda x : datetime.strptime(x, '%d/%m/%Y %H:%M:%S'))

# reindexar el dataframe
viaje.index = viaje.fecha_hora

# limpiar valores de otros años
viaje = viaje.loc['2018-10']
viaje.head()
```

✓ 16.3s

fecha_hora	Genero_Usuario	Edad_Usuario	Bici	Ciclo_Estacion_Retiro	Fecha_Retiro	Hora_Retiro	Ciclo_Estacion_Arribo	Fecha_Arribo	Hora_Arribo	fecha_hora_retiro	fecha_hora
2018-10-01 00:00:02	M	58	2176	299	01/10/2018	0:00:02	75	01/10/2018	0:12:02	01/10/2018 0:00:02	2018-10-01 00:00:02
2018-10-01 00:00:23	F	42	9816	43	01/10/2018	0:00:23	256	01/10/2018	0:15:02	01/10/2018 0:00:23	2018-10-01 00:00:23
2018-10-01 00:00:34	M	30	11219	37	01/10/2018	0:00:34	1	01/10/2018	0:09:09	01/10/2018 0:00:34	2018-10-01 00:00:34
2018-10-01 00:01:36	M	26	3910	123	01/10/2018	0:01:36	182	01/10/2018	0:37:22	01/10/2018 0:01:36	2018-10-01 00:01:36
2018-10-01 00:02:39	M	29	11487	41	01/10/2018	0:02:39	30	01/10/2018	0:11:29	01/10/2018 0:02:39	2018-10-01 00:02:39

Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Explicación

La función *lambda* está creando una función anónima y el método .map() está aplicando esta función para el cambio de formato a cada elemento de la columna fecha_hora. El siguiente paso es agrupar los viajes por hora:

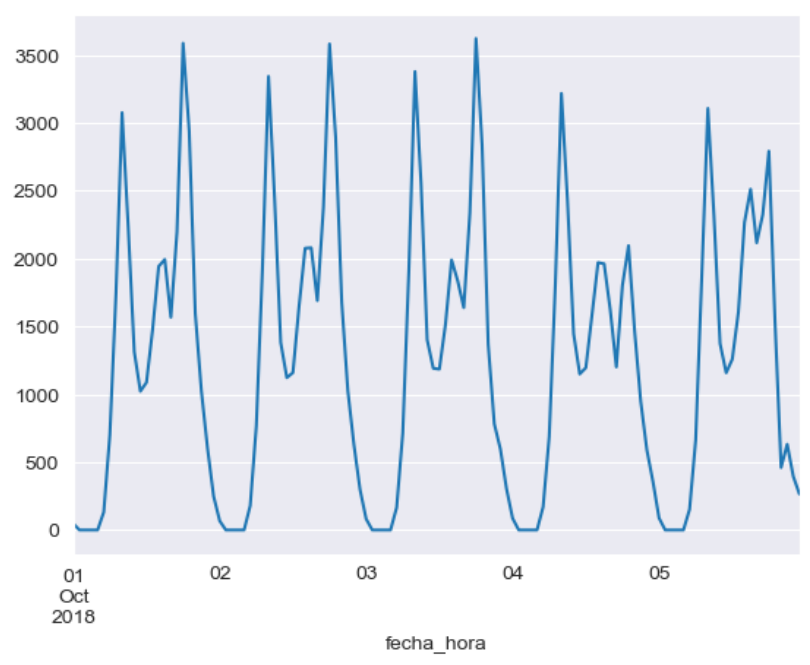
- **# resample y conteo de viajes**
- **viajes_resample_day = viajes.Bici.resample('H').count()**
- **# asignar el nombre del día de la semana**
- **df_resample = pd.concat([viajes_resample_day], axis=1)**
- **df_resample['dayofweek'] = df_resample.index.dayofweek # el 0 equivale a lunes**
- **# de lunes a viernes**
- **df_mon_to_fri = df_resample[df_resample.dayofweek.isin([0,1,2,3,4])].Bici**

Explicación

Por último, se genera la gráfica, y como muestra del resultado, se presentan los días de la primera semana de octubre:

- **df_mon_to_fri[0:(24*5)].plot()**
- **plt.show()**

Observando la gráfica, se ven ciertos patrones que se repiten diariamente a determinadas horas que son entradas y salidas de trabajo y escuelas.

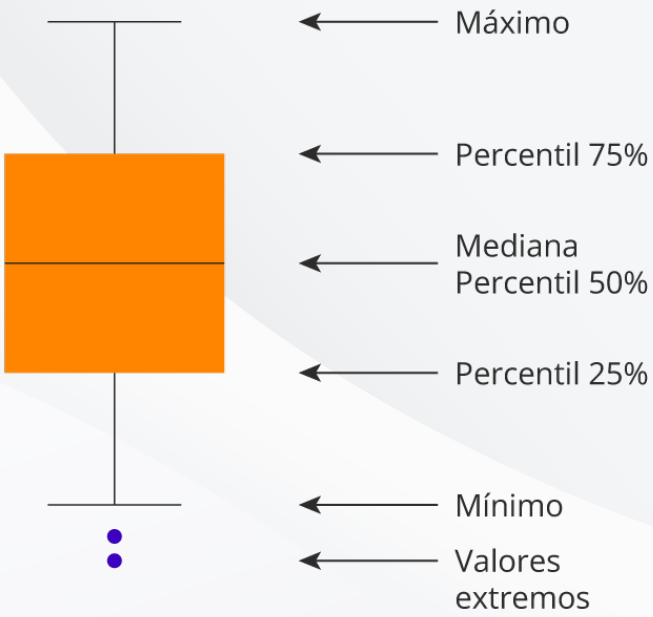


Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Explicación

Gráfica de caja (box plot)

- Muestra múltiples métricos estadísticos, visualizando la distribución de los datos.
- Muestra los valores de una variable que corresponden a cada cuartil, así como los valores máximo y mínimo. Cada cuartil corresponde a 25 % de los datos.

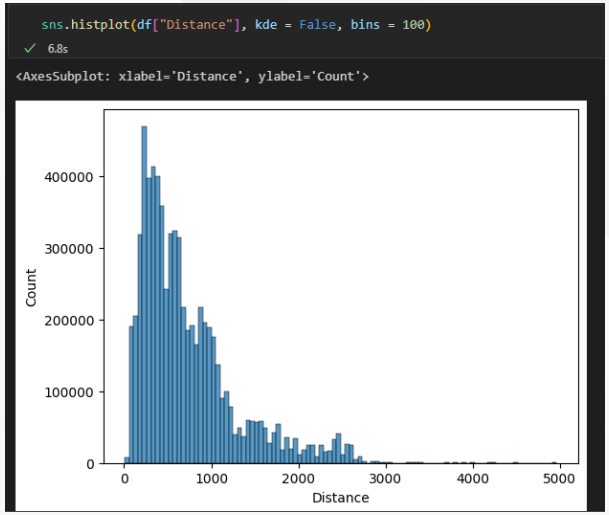


- Las líneas horizontales extremas son valores calculados y se conocen como bigotes.
- La diferencia entre el tercer y primer cuartil se conoce como rango intercuartil.
- Regularmente los bigotes se calculan como 1.5 veces el rango intercuartil.
- Los puntos muestran datos extremos fuera de los bigotes, y se conocen como *outliers*.

Explicación

En otro ejemplo, considera que quieres analizar los registros de vuelos en determinado aeropuerto, siguiendo los siguientes pasos:

- Importar librerías y leer el archivo “base_datos_2008.csv” que contiene la información. En este ejemplo, se va utilizar otra librería de Python para graficar histogramas llamada *seaborn*.
- Crear el dataframe con la información completa de los vuelos de salida y llegada del aeropuerto.
- Limpiar los datos eliminando valores nulos.
- Dado que la base de datos contiene mucha información que no se requiere, se crea un subconjunto de variables para el análisis.
  - **import pandas as pd**
  - **import seaborn as sns**
  - **import matplotlib.pyplot as plt**
  - **df = pd.read_csv('base_datos_2008.csv')**
  - **df.dropna(inplace=True, subset=["ArrDelay","DepDelay","Distance"])**
  - **sns.histoplot(df[‘Distance’], kde = False, bins=100)**

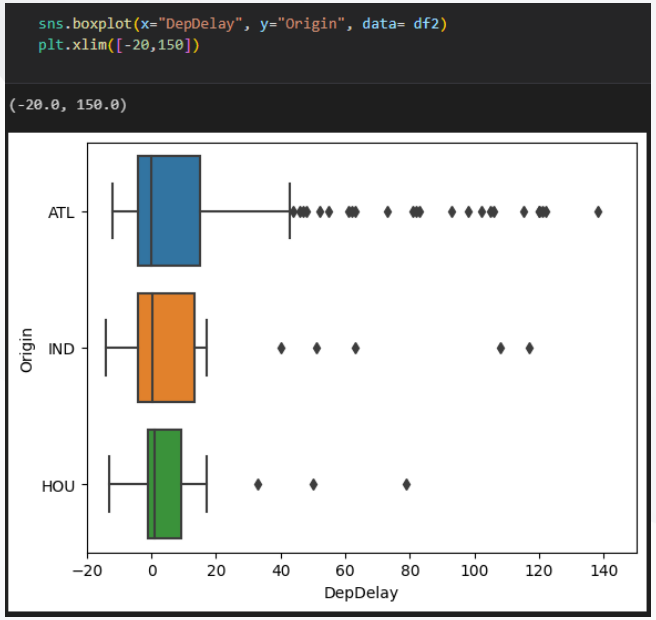


Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Explicación

Para realizar una gráfica de caja con datos más específicos, se crea una muestra de la base de datos, tomando solamente los primeros 500 vuelos que salieron con retraso (DepDelay) de los estados de Atlanta, Houston e Indiana (Origin).

- df2= df[df["Origin"].isin(["ATL","HOU","IND"])].sample(frac = 1).head(500)
- sns.boxplot(x="DepDelay", y="Origin", data= df2)
- plt.xlim([-20,150]) # se limitan los valores del eje x



Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Se puede apreciar que Atlanta tiene un problema para el control de despegues, mientras que en Houston los vuelos se demoran entre 0 y 20 minutos, con muy pocos vuelos demorados de más de 20 minutos.

Actividad

Para reforzar tu aprendizaje del tema, realiza la siguiente actividad:

Considera que te están contratando por la mesa de seguridad de la ciudad en donde radicas, participando en el proyecto para prevenir y reducir la ocurrencia de conductas delictivas por medio del análisis de datos históricos que identifiquen patrones de la ocurrencia de eventos y el lugar en donde ocurre el evento.

Consigue o genera una tabla con datos ficticios que contengan registros de la ocurrencia de conductas delictivas de los últimos 90 días, que incluya las variables de fecha de ocurrencia, edad del delincuente y código postal del evento.

1. Elabora el histograma con la frecuencia de eventos en relación con la edad del delincuente.
2. Elabora la gráfica de serie de tiempo con la cantidad de eventos registrados en períodos semanales, ordenados cronológicamente del más antiguo al más reciente.
3. Elabora la gráfica de caja para mostrar la distribución de la edad del delincuente en tres códigos postales.
4. Realiza tus comentarios y recomendaciones sobre tus hallazgos.

Presenta tu análisis mostrando el código Python que utilizaste, así como la gráfica correspondiente.

Cierre

Con la práctica de este tema tienes el conocimiento necesario para realizar análisis de datos y complementarlos con elementos gráficos en Python para comunicar tus hallazgos que apoyan la gestión de logística, identificando las rutas más eficientes considerando las métricas del tiempo de servicio, el combustible consumido y el horario de servicio. Además, uno de los beneficios que la dirección aprecia de tu modelo analítico, es que las gráficas se van a estar actualizando automáticamente conforme se actualice nueva información y, por lo tanto, se propuso instalar un monitor a la vista de todos para compartir tu modelo a todo el departamento.

Bibliografía

- Prabhakaran, S. (2019). *Time series analysis in Python – a comprehensive guide with examples.* Recuperado de https://www.machinelearningplus.com/time-series/time-series-analysis-python/

SKILLING CENTER TECMILENIO

Visualización y Programación en Python

Graficación en Python

Introducción

Karla ha estado tomando algunos cursos de Python debido a que quiere obtener una promoción en su empresa, sin embargo, en su examen para el puesto deseado, se encontró con que debía graficar algunas funciones matemáticas, así como crear y explicar algunos gráficos aplicados al área financiera.

Ahora Karla se encuentra en búsqueda de algún curso que le ayude a conocer de manera fácil algunos tipos de gráficos que le ayuden a mostrar resultados de datos y de funciones matemáticas calculadas.

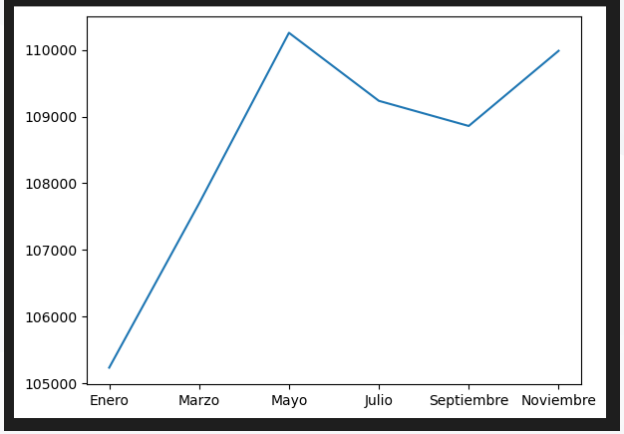
Explicación

Graficación en Python

- En Python puedes generar diversos tipos de gráficos a partir de datos contenidos en un dataframe.
- La visualización te permite mostrar el resultado obtenido de manera clara y objetiva.
- Por ejemplo, utilizando el módulo pyplot de la biblioteca matplotlib, puedes generar la siguiente gráfica:
  - **import matplotlib.pyplot**
  - **meses = [‘Enero', ’Marzo', ’Mayo', ’Julio’, ‘Septiembre', ’Noviembre']**
  - **valores = [105235, 107697, 110256, 109236, 108859, 109986]**
  - **matplotlib.pyplot.plot(meses, valores)**
- Se obtiene la siguiente gráfica:

```
import matplotlib.pyplot
meses = ['Enero', 'Marzo', 'Mayo', 'Julio','Septiembre', 'Noviembre']
valores = [105235, 107697, 110256, 109236, 108859, 109986]

matplotlib.pyplot.plot(meses, valores)
```

```
[<matplotlib.lines.Line2D at 0x18d49c845e0>]
```



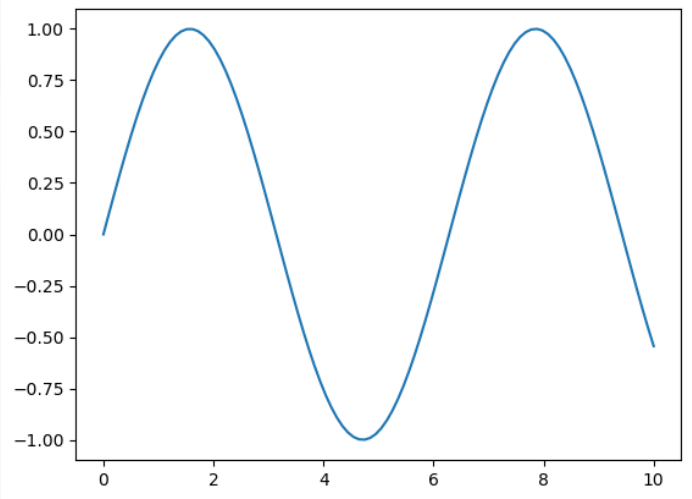
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Explicación

Graficación de funciones matemáticas

- Se utilizan librerías especializadas.
- Por ejemplo, para graficar la función sen(x) se utilizan las librerías matplotlib y numpy, de la siguiente forma:
  - **import numpy as np**
  - **import matplotlib.pyplot as plt**
  - **def h(x):**
  - **return np.sin(x)**
- Y para generar la gráfica se define el dominio con la función linspace y los valores del eje x y y:
  - **x= np.linspace(0,10,100)**
  - **plt.plot(x,h(x))**

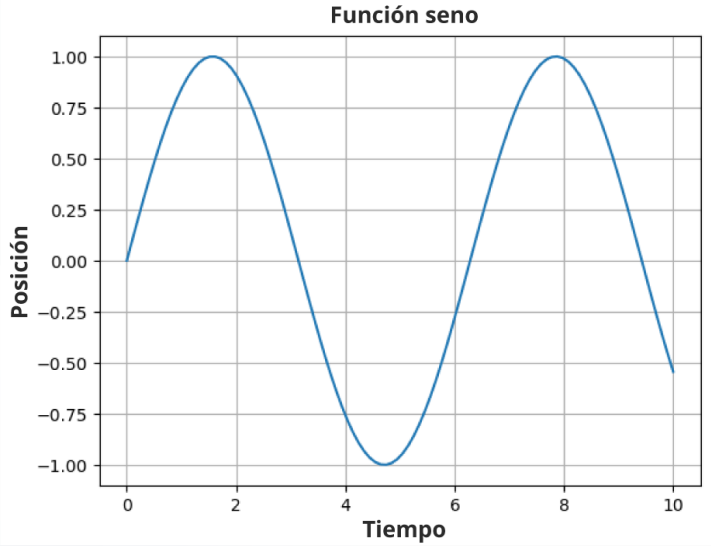
- La gráfica resultante es la siguiente:



Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Explicación

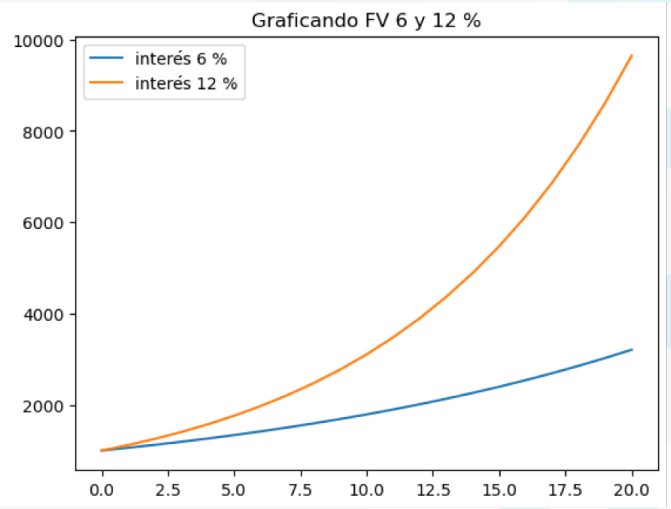
- Lo mismo se puede aplicar para graficar las funciones de coseno y exponencial.
  - **return np.cos(x)**
  - **return np.exp(x)**
- Adicionalmente, se puede completar la gráfica con etiquetas en los ejes y título:
  - **import numpy as np**
  - **import matplotlib.pyplot as plt**
  - **def h(x):**
  - **return np.sin(x)**
  - **x= np.linspace(0,10,100)**
  - **plt.plot(x,h(x))**
  - **plt.xlabel('tiempo')**
  - **plt.ylabel('posicion')**
  - **plt.title('Funcion seno')**
  - **plt.grid()**
- La gráfica resultante es la siguiente:



Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Explicación

- En Python, es posible graficar cualquier tipo de función matemática y/o financiera.
- Como ejemplo, se crea una función financiera para calcular el valor futuro equivalente de un depósito de $1,000 a una tasa de interés del 6 % anual y por un plazo de 3 años.
  - **import numpy_financial as npf**
  - **import matplotlib.pyplot as plt**
  - **x = -1000 # deposito**
  - **r = 0.06 # tasa de interes**
  - **n = 3 # cantidad de años**
  - **fv = npf.fv(pv=x, rate=r, nper=n, pmt=0)**
  - **fv**
- El resultado es $1,191.016.
- Para graficar la función financiera del valor futuro en función del plazo en años, se van a considerar valores de r=6% y 12% y el plazo a 20 años.
  - **# Grafica del valor futuro con interes de 6 y 12% a 20 años**
  - **t = list(range(0, 21)) # genera valores del 0 al 20**
  - **def fv6(num): # función del valor futuro para r=6%**
  - **return npf.fv(pv=-1000, rate=0.03, pmt=0, nper=num)**
  - **def fv12(num): # función del valor futuro para r=12%**
  - **return npf.fv(pv=-1000, rate=0.12, pmt=0, nper=num)**
  - **plt.title("Graficando FV 6 y 12%")**
  - **plt.plot(t,fv6(t), label="interes 6%")**
  - **plt.plot(t, fv12(t), label="interes 12%")**
  - **plt.legend(loc='upper left')**
  - **plt.show()** asdf



Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Explicación

Subgráficas

- Es posible graficar un conjunto de gráficas en el mismo marco, creado con la funcion add_subplot.
- Como ejemplo, se van a graficar los precios de las acciones de cuatro empresas.
- Los valores se descargan del sitio web de Yahoo Finance con la librería yfinance, el módulo yf.download y los símbolos de cada acción.
- Con el siguiente código Python se importan las librerías necesarias y se descargan los precios entre el 1 y el 15 de febrero del 2023.

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

%matplotlib inline

from datetime import datetime
import yfinance as yf
inicio='2023-02-01'   # fecha inicial
fin='2023-02-15'      # fecha final

microsoft = yf.download('MSFT', start = inicio, end=fin)
amazon = yf.download('AMZN', start = inicio , end=fin)
google = yf.download('GOOGL', start = inicio , end=fin)
meta = yf.download('META', start = inicio , end=fin)
```

Explicación

- Se configura el índice con la fecha y año:
  - **fechas=amazon.index # se puede elegir cualquiera de los 4 dataframes**
  - **x = []**
  - **for date in fechas:**
  - **   x.append(date.year)**

- Se genera el mosaico 2x2 con las cuatro gráficas:

- **plt.figure(figsize=(16,18))**

- **#Plot 1 #**
- **plt.subplot(2,2,1)**
- **plt.xticks(rotation='vertical')**
- **plt.plot(fechas, google)**
- **plt.title('Precio en el mercado de Google (GOOGL)')**

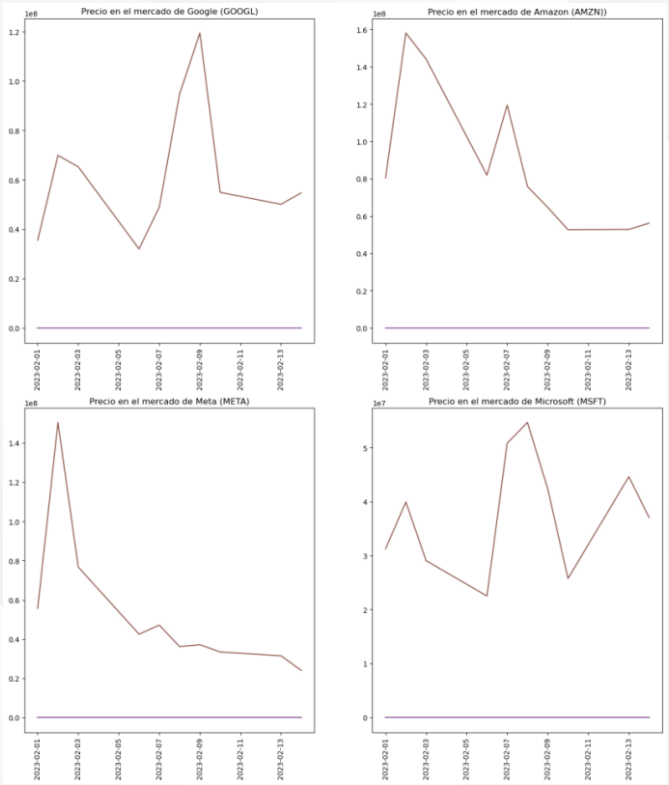
- **#Plot 2**
- **plt.subplot(2,2,2)**
- **plt.xticks(rotation='vertical')**
- **plt.plot(fechas, amazon)**
- **plt.title('Precio en el mercado de Amazon (AMZN))')**

Explicación

- Continuación del código para el mosaico de cuatro gráficas:

- **#Plot 3**
- **plt.subplot(2,2,3)**
- **plt.xticks(rotation='vertical')**
- **plt.plot(fechas, meta)**
- **plt.title('Precio en el mercado de Meta(META)')**

- **#Plot 4**
- **plt.subplot(2,2,4)**
- **plt.xticks(rotation='vertical')**
- **plt.plot(fechas, microsoft)**
- **plt.title('Precio en el mercado de Microsoft (MSFT)')**



Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

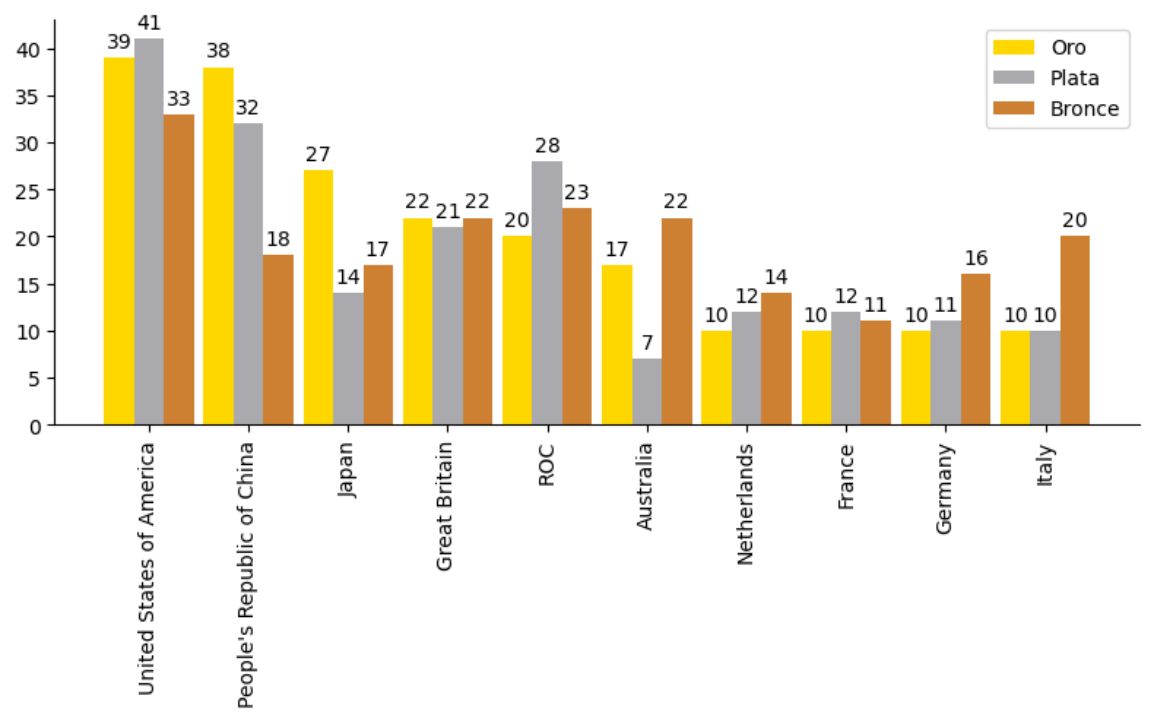
Explicación

Otro tipo de gráficas de dos dimensiones

- *Gráfico de barras*. Son adecuadas cuando se quiere mostrar la comparación de valores de las dimensiones de una variable.
- Como ejemplo, se obtienen los datos de las medallas olímpicas ganadas por cada país:
  - **df = pd.read_excel('Medals.xlsx') # leer el archivo Excel con la librería pandas**
  - **df_mejores = df.nsmallest(10, 'Rank') # se rankean los 10 primeros del total de medallas**
  - Se genera la gráfica con el módulo subplots:
    - **fig, ax = plt.subplots(figsize=(8, 5), dpi=100) # 800x500 px**
    - **bar_width = 0.30**
    - **x = np.arange(df_mejores.index.size)**
    - **medallas_oro = ax.bar(x - bar_width, df_mejores['Gold'],**
    - **bar_width, label='Oro', color='#ffd700')**
    - **medallas_plata = ax.bar(x, df_mejores['Silver'],**
    - **bar_width, label='Plata', color='#aaa9ad')**
    - **medallas_bronce = ax.bar(x + bar_width, df_mejores['Bronze'],**
    - **bar_width, label='Bronce', color='#cd7f32')**
    - **ax.set_xticks(x)**
    - **ax.set_xticklabels(df_mejores.Country, rotation=90)**
    - **ax.legend()**
    - **# Etiquetas en barras**
    - **ax.bar_label(medallas_oro, padding=3)**
    - **ax.bar_label(medallas_plata, padding=3)**
    - **ax.bar_label(medallas_bronce, padding=3)**
    - **ax.spines['right'].set_visible(False) # ocultar borde derecho**
    - **ax.spines['top'].set_visible(False) # ocultar borde superior**
    - **fig.tight_layout() # ajustar elementos al tamaño de la figura**

Explicación

La gráfica resultante es la siguiente:



Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

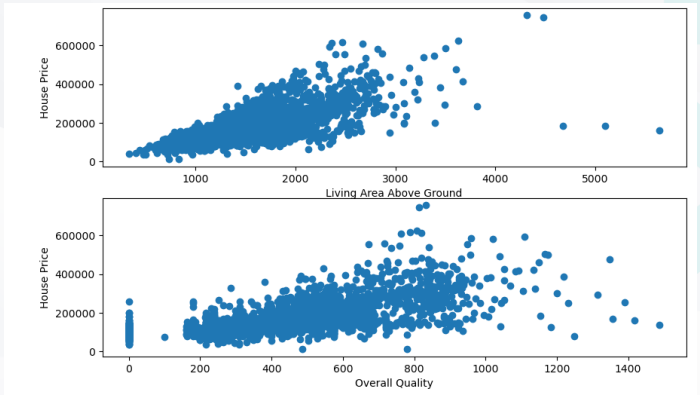
Explicación

Gráficos de dispersión

- Son adecuadas para identificar la relación entre los valores de determinadas variables. Para la visualización, conviene hacerlo con dos variables.
- Validan la semejanza del comportamiento y tendencia de las variables.
- Considera establecer la correlación que existe entre el precio de la vivienda y el área de construcción.
- El tipo de gráfica en Python es scatter.

El código utilizado para elaborar la gráfica es el siguiente:

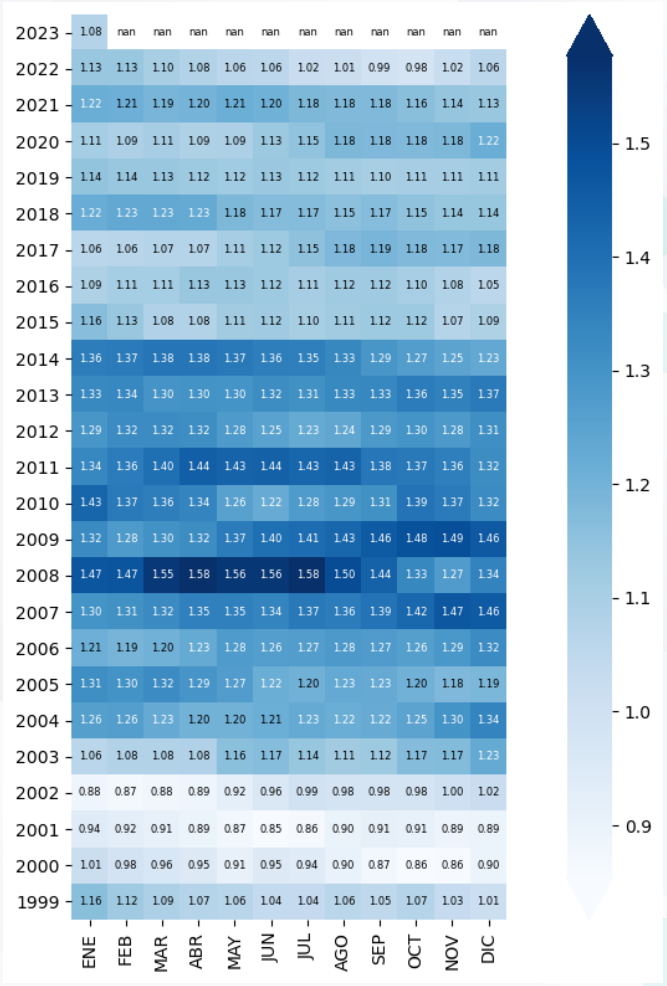
```
import matplotlib.pyplot as plt
import pandas as pd
df = pd.read_csv('AmesHousing.csv')
fig, ax = plt.subplots(2, figsize=(10, 6))
ax[0].scatter(x = df['Gr Liv Area'], y = df['SalePrice'])
ax[0].set_xlabel("Living Area Above Ground")
ax[0].set_ylabel("House Price")

ax[1].scatter(x = df['Garage Area'], y = df['SalePrice'])
ax[1].set_xlabel("Overall Quality")
ax[1].set_ylabel("House Price")

plt.show()
```



Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Explicación

- **Gráficos de mapas de calor**
  - Son adecuadas cuando se quiere comparar los valores de una variable por medio de color o saturación de color.
  - Como ejemplo, se quiere hacer una tabla con valores históricos de la cotización del euro frente al dólar americano.
  - Los renglones son los años y las columnas son los meses.
  - Cada celda contiene el valor de la cotización promedio correspondiente al mes-año.
  - El código Python se puede descargar de la siguiente liga de Kaggle:



Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

El siguiente enlace es externo a la Universidad Tecmilenio, al acceder a él considera que debes apegarte a sus términos y condiciones.

Chemkaeva, D. (s.f.). *Daily Exchange Rates per Euro 1999-2023*. Recuperado de https://www.kaggle.com/datasets/lsind18/euro-exchange-daily-rates-19992020/data

Actividad

Para reforzar tu aprendizaje del tema, realiza lo siguiente:

1. Elabora el código Python y la gráfica la función de tangente de x.
2. Utiliza la librería de numpy_financial para calcular el valor anual equivalente de un crédito por $200,000 a una tasa de interés del 8 % anual y un plazo de 6 años.
3. Por medio de la librería yfinance:
    a) Descarga la cotización del Bitcoin y del índice Nasdaq de Estados Unidos de los últimos 30 días.
    b) Elabora la gráfica de la serie de tiempo para cada uno.
    c) Elabora la gráfica de correlación entre los 2 activos.
    d) Elabora tus comentarios sobre tus hallazgos en la correlación entre estos activos.

Cierre

Con el aprendizaje de este tema, conociste una parte del alcance que tiene el lenguaje Python para representar gráficas de tus análisis, con algunas gráficas de las más utilizadas en la ciencia de datos. Así también Karla ya tiene el requisito del perfil para crear funciones matemáticas y financieras, y tiene toda la confianza de que es la persona indicada para ocupar el puesto de la promoción en su empresa.

Tecmilenio no guarda relación alguna con las marcas mencionadas como ejemplo. Las marcas son propiedad de sus titulares conforme a la legislación aplicable, estas se utilizan con fines académicos y didácticos, por lo que no existen fines de lucro, relación publicitaria o de patrocinio.
