guayerd

Fundamentos IA

Análisis con Python

Clase 8

En colaboración con

IBM **SkillsBuild**

¡Bienvenidos!

¿Nos presentamos?

- ¿Qué recuerdan de la clase anterior?
- ¿Qué esperan aprender?
- ¿Tienen alguna pregunta?

guayerd

En colaboración con

IBM **SkillsBuild**

Contenidos

Por temas

05
- Copilot Chat y prompts
- Demo asincrónica

06
- Limpieza y transformación

07
- Estadística aplicada

08
- Visualización

guayerd

En colaboración con

IBM **SkillsBuild**

Objetivos de la clase

- Visualización
- Matplotlib
- Seaborn

guayerd

En colaboración con

IBM **SkillsBuild**

Análisis con Python

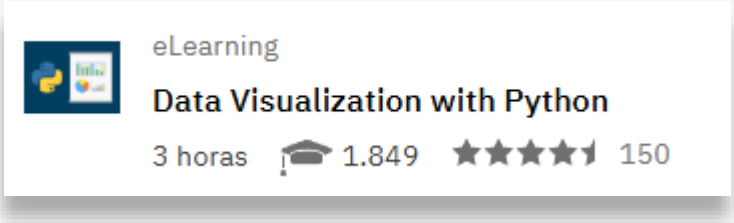
Visualización

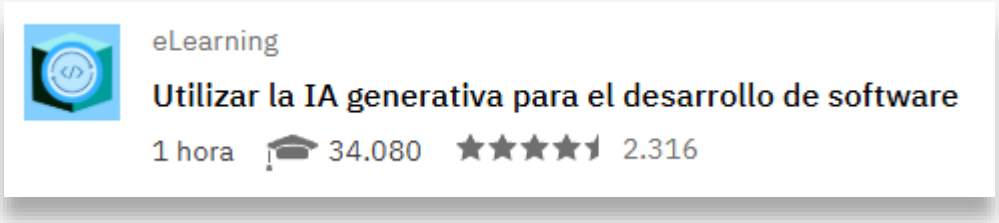
guayerd

En colaboración con

IBM **SkillsBuild**

Plataforma Skill Build: Python





guayerd

En colaboración con

IBM **SkillsBuild**

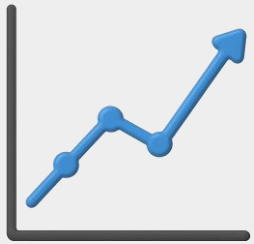
Visualización de datos

Transforma **datos en representaciones gráficas** interpretables.

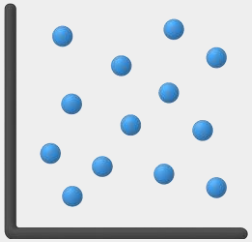
- Revela patrones no detectables
- Identifica valores atípicos y distribuciones anómalas
- Comunica hallazgos de forma comprensible
- Valida hipótesis sobre comportamiento de datos

Patrones, outliers, hallazgos

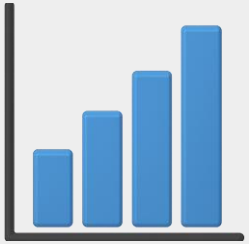
Tipos de visualizaciones



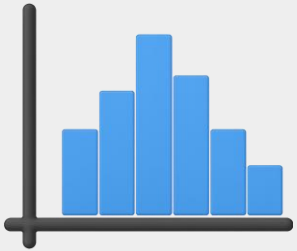
Evolución temporal



Relación entre variables



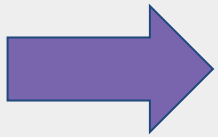
Comparación categórica



Distribución de datos

Identifica el gráfico

- Evolución de ventas durante 12 meses
- Comprar satisfacción entre 5 sucursales
- Analizar relación precio-demanda
- Distribución de edades de clientes
- Comparar ventas promedio entre temporadas
- Mostrar cambio de precios a lo largo de 2 años



- Líneas
- B
- Dispers
- Hist ma
- Barr
- Líneas

Matplotlib

¿Qué es Matplotlib?
Matplotlib es una biblioteca de visualización en Python que permite crear gráficos estáticos, animados e interactivos de alta calidad. Es ampliamente utilizada en ciencia, ingeniería y análisis de datos para representar visualmente patrones, distribuciones y relaciones entre variables.

Librería fundamental para **crear gráficos en Python.**

- Compatible con arrays NumPy y DataFrames pandas
- Exportación en formatos profesionales (PNG, PDF, SVG)

Comandos

- **Instalación:** pip install matplotlib
- **Uso:** import matplotlib.pyplot as plt

💡 **Bonus**: Matplotlib se integra perfectamente con **NumPy**, **pandas** y **seaborn**, lo que permite enriquecer tus análisis con visualizaciones personalizadas y potentes.

Tipos de visualizaciones con Mathplotlib

📈 **Gráfico de líneas**	Evolución temporal, tendencias	plot()
📉 **Gráfico de dispersión**	Relación entre dos variables continuas	scatter()
📊 **Gráfico de barra**	Comparación entre categorías	bar() / barh()
🧱 **Histograma**	Distribución de frecuencias	hist()
📦 **Boxplot (diagrama de caja)**	Resumen estadístico de una variable numérica	boxplot()
🌀 **Gráfico de torta**	Proporciones de un total	pie()
🏞️ **Gráfico de área**	Acumulación de valores a lo largo del tiempo	stackplot()
🧭 **Gráfico polar**	Datos angulares o cíclicos	polar()
🧩 **Subplots**	Composición de múltiples gráficos en una figura	subplot() / subplots()

Elementos fundamentales

Elemento	Código	Propósito
Crear figura	plt.figure(figsize=(8, 6))	Define tamaño del gráfico
Agregar datos	plt.plot(x, y)	Dibuja líneas y puntos
Títulos	plt.title('Texto')	Contexto del gráfico
Ejes	plt.xlabel('Variable'), plt.ylabel('Variable')	Claridad en interpretación
Mostrar	plt.show()	Renderizar gráfico final

Gráficos de líneas

Componente	Sintaxis	Descripción
Línea básica	plt.plot(x, y)	Conexión simple entre puntos
Con marcadores	plt.plot(x, y, marker='o')	Puntos visibles en cada dato
Múltiples series	plt.plot(x, y1), plt.plot(x, y2)	Comparar tendencias

Series temporales, métricas en evolución y comparación de tendencias

Gráficos de barras

Tipo	Sintaxis	Aplicación
Verticales	plt.bar(categorias, valores)	Comparaciones estándar
Horizontales	plt.barh(categorias, valores)	Etiquetas largas
Agrupadas	plt.bar(x, y1), plt.bar(x+width, y2)	Múltiples variables

Rankings, comparaciones categóricas y distribuciones por grupos

Gráficos de dispersión

Elemento	Sintaxis	Función
Puntos básicos	plt.scatter(x, y)	Relación simple
Tamaño variable	plt.scatter(x, y, s=tamaños)	Tercera dimensión
Colores por grupo	plt.scatter(x, y, c=categorias)	Segmentación

Correlaciones, segmentaciones y análisis multivariado

Histogramas

Parámetro	Sintaxis	Efecto
Básico	plt.hist(datos)	Frecuencias automáticas
Intervalos	plt.hist(datos, bins=20)	Control de granularidad
Transparencia	plt.hist(datos, alpha=0.7)	Comparar distribuciones

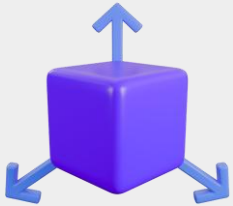
Análisis de normalidad, detección de valores atípicos y comparación de grupos

Mejores prácticas

Títulos
Claros y con contexto

Colores
Paletas consistentes y accesibles

Texto
Legible y con buen contraste



Ejes
Escalas completas y proporcionales

Claridad
Gráficos simples y directos

Seaborn

¿Qué es Seaborn?
Seaborn es una biblioteca de visualización estadística en Python construida sobre Matplotlib. Ofrece una interfaz más sencilla y estética para crear gráficos complejos con menos código, integrándose perfectamente con estructuras de datos tipo `DataFrame` (especialmente de pandas).

Librería basada en Matplotlib para **gráficos estadísticos con diseño optimizado.**

- Sintaxis simplificada para gráficos complejos
- Integración directa con DataFrames de Pandas

Comandos

- **Instalación:** pip install seaborn
- **Uso:** import seaborn as sns

💡 **Bonus:** Seaborn automatiza el cálculo de estadísticas como medias, desviaciones, regresiones y densidades, lo que lo hace ideal para análisis exploratorio rápido y visualmente atractivo.

Tipos de visualizaciones con Seaborn

Tipo de gráfico	Uso principal	Función típica (sns.)
📈 **Gráfico de líneas**	Tendencias temporales con bandas de confianza	lineplot()
📉 **Gráfico de dispersión**	Relación entre variables, con agrupación por categoría	scatterplot()
📊 **Gráfico de barras**	Comparación entre categorías con agregación estadística	barplot()
🧱 **Histograma / KDE**	Distribución de datos con suavizado opcional	histplot() / kdeplot()
📦 **Boxplot / Violinplot**	Resumen estadístico y forma de la distribución	boxplot() / violinplot()
🧩 **Gráfico de puntos categóricos**	Comparación entre grupos con dispersión	stripplot() / swarmplot()
🧠 **Mapa de calor (heatmap)**	Matrices de correlación o intensidad	heatmap()
🔗 **Gráfico de regresión**	Relación lineal con ajuste automático	regplot() / lmplot()
🧬 **Gráfico de pares (pairplot)**	Exploración multivariada entre todas las variables	pairplot()
🧪 **Gráfico de cajas múltiples**	Comparación entre grupos y subgrupos	catplot()

Gráficos estadísticos principales

Boxplot

Outliers y cuartiles por grupo

```
sns.boxplot(data=df,
x='cat', y='num')
```

Heatmap

Matriz de correlaciones

```
sns.heatmap(df.corr(),
annot=True)
```

Violinplot

Densidad de distribuciones

```
sns.violinplot(data=d
f, x='cat', y='num')
```

Rendimiento E-commerce

- Identificar mes con mayor eficiencia (ventas/gasto publicidad)
- Determinar mes con mejor tasa de conversión y analizar causa
- Calcular ticket promedio (ventas/productos) por mes
- Evaluar relación entre visitantes y ventas

Mes	Ventas ($)	Visitantes	Conversión (%)	Gasto Publicidad ($)	Productos Vendidos
Ene	45,000	15,000	3.2	8,500	450
Feb	52,000	18,200	2.9	9,800	520
Mar	38,000	12,500	3.8	7,200	380
Abr	61,000	20,500	3.1	11,200	610
May	48,000	16,800	2.7	9,500	480

Visualizar rendimiento E-commerce

A partir del ejercicio trabajado en la clase anterior, se aplicarán distintas visualizaciones:

- **Líneas:** Evolución de ventas mensuales
- **Barras:** Comparación de eficiencia publicitaria (ventas/gasto)
- **Dispersión:** Relación visitantes vs ventas
- **Heatmap:** Correlaciones entre ventas, visitantes y gasto

Proyecto

Tienda Aurelion

- **Documentación:** notebook Markdown
- **Desarrollo técnico:** programa Python
- **Visualización de datos:** dashboard en Power BI
- **Presentación oral:** problema, solución y hallazgos

guayerd

En colaboración con

IBM **SkillsBuild**

Aplicando visualizaciones

Trabajo en equipo

1. Seleccionar **variables clave** de su dataset
2. Crear al menos **3 visualizaciones**
3. Identificar **patrones, tendencias o correlaciones**
4. **Interpretar resultados** en contexto de su problema
5. **Documentar con Copilot** hallazgos y decisiones

Preparación demo

próxima clase

guayerd

En colaboración con

IBM **SkillsBuild**

Características

2° demo: sincrónica

- **Tema:** python
- **Medio:** oral
- **Devolución:** en vivo
- **Entrega:** archivo .py o Notebook Jupyter
- **Presentación:** Power Point (opcional)
- **Exposición:** Cada equipo debe organizarse
- **Tiempo:** máximo 15 minutos por equipo

Contenido

2° demo: sincrónica

Documentación actualizada (.md)

- Estadísticas descriptivas básicas calculadas
- Identificación del tipo de distribución de variables
- Análisis de correlaciones entre variables principales
- Detección de outliers (valores extremos)
- Al menos 3 gráficos representativos
- Interpretación de resultados orientada al problema

Base de datos (.csv)

- Limpia y lista para análisis

Programa actualizado (.py)

- Información actualizada del proyecto

Retro

¿Cómo nos vamos?

- ¿Qué fue lo más útil de la clase?
- ¿Qué parte te costó más?
- ¿Qué te gustaría repasar o reforzar?

guayerd

En colaboración con

IBM **SkillsBuild**