

Laboratorio20

Sesión # 20 Introducción a la visualización de datos con Python

Título del Laboratorio: Aplicación y uso de la herramienta de Python para hacer el proceso de ETL y visualizaciones.

Duración: 2 horas

Objetivos del Laboratorio:

1. *Afianzar los conocimientos y manejo básico para realizar las visualizaciones de los datos con Python: data, exploración, limpieza y gráficas personalizadas. usando los datos con ejercicios prácticos planteados de acuerdo con el escenario.*

Materiales Necesarios:

1. *Computador con acceso a internet.*
2. *Colocarlo en el repositorio de Github*
3. *Ampliar el conocimiento con el curso de datos en AWS y Cisco.*
4. *Python*

Estructura del Laboratorio:

Parte 1

En la primera parte se aplicarán los temas vistos en la sesión como la Introducción a la visualización de los datos con Python: data, exploración, limpieza y gráficas personalizadas usando los datos, se deberá realizar el paso a paso con las respectivas capturas de pantalla, esta aplicación es de acuerdo con los escenarios planteados.

1. Ejercicio de práctica 1.

Realizar el paso a paso de la visualización de los datos con Python: data, exploración, limpieza y gráficas personalizadas, deberás realizar las respectivas capturas de pantalla, conclusión, guardar el archivo.

Escenario 1: Sector Financiero

Una institución bancaria desea analizar la relación entre las características demográficas de sus clientes y su historial crediticio para mejorar la gestión del riesgo de crédito. El banco ha recolectado datos sobre la edad, el género, el historial crediticio, el salario anual y la deuda de sus clientes. Se desea estudiar la relación entre estas variables para identificar patrones de comportamiento y mejorar el proceso de toma de decisiones en la concesión de créditos.

- **Data:** datos_financieros.csv

Paso a paso

Conclusión

El análisis reveló que los clientes con mayores ingresos solicitan más crédito, mientras que los jóvenes presentan un historial crediticio menos sólido. El género no influyó en el riesgo, y altos niveles de deuda están asociados con un menor historial crediticio. Las visualizaciones facilitaron la comprensión de estos patrones, ayudando a optimizar las políticas de crédito.

Escenario 2: Sector Salud

Un hospital está llevando a cabo un estudio para analizar cómo diferentes factores afectan el riesgo de desarrollar diabetes en pacientes. Se han recolectado datos sobre la edad, el género, los niveles de glucosa en sangre, el índice de masa corporal (IMC) y si el paciente tiene antecedentes familiares de diabetes. El objetivo es identificar patrones en los datos para detectar a los pacientes con mayor riesgo de desarrollar esta enfermedad.

- **Data:** datos_salud glu.csv.

Paso a paso

Conclusión

En el sector salud, se identificó que altos niveles de glucosa e IMC son factores clave para el riesgo de diabetes, junto con la edad y antecedentes familiares. Las visualizaciones aclararon las tendencias y permitieron segmentar a los pacientes en riesgo. Estos hallazgos apoyarán la implementación de intervenciones personalizadas para mejorar la prevención de la diabetes.

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

df_financiero = pd.read_csv('./source/datos_financieros.csv')
print(df_financiero)

print(f"""\nResumen de los datos:
{df_financiero.describe()}""")

print(f"""\nVerificación de valores nulos:
{df_financiero.isnull().sum()}""")

df_financiero['Salario_Anual'].fillna(df_financiero['Salario_Anual'].median(), inplace=True)
df_financiero['Deuda'].fillna(df_financiero['Deuda'].median(), inplace=True)

print(f"""\nVerificación de valores nulos después de limpieza
{df_financiero.isnull().sum()}""")

print(f"""\nGuardado de datos limpios
{df_financiero.to_csv('./source/clean/datos_financieros_limpios.csv')}""")
```

| | Edad | Genero | Historial_Crediticio | Salario_Anual | Deuda |
|------|------|--------|----------------------|---------------|---------|
| 0 | 56 | M | Regular | 61037.0 | 22443.0 |
| 1 | 69 | F | Bueno | 31805.0 | 39725.0 |
| 2 | 46 | F | Bueno | 63147.0 | 49212.0 |
| 3 | 32 | M | Malo | 94280.0 | 18595.0 |
| 4 | 60 | M | Regular | 66053.0 | 9740.0 |
| ... | ... | ... | ... | ... | ... |
| 4995 | 24 | M | Regular | 116777.0 | 27655.0 |
| 4996 | 66 | F | Malo | 101909.0 | 29143.0 |
| 4997 | 26 | F | Regular | 107409.0 | 15698.0 |
| 4998 | 53 | F | Malo | 57664.0 | NaN |
| 4999 | 36 | F | Malo | 21293.0 | 9102.0 |

[5000 rows x 5 columns]

```
Resumen de los datos:
      Edad  Salario_Anual  Deuda
count  5000.000000    4800.000000  4850.000000
mean    43.584600    69131.827917  25137.928454
std     14.919094    29221.920701  14478.122739
min     18.000000    20005.000000   18.000000
25%     31.000000    43049.500000  12817.500000
50%     43.000000    69636.500000  24841.500000
75%     56.000000    94543.000000  37690.250000
max     69.000000   119987.000000  49999.000000

Verificación de valores nulos:
Edad      0
Genero    0
Historial_Crediticio  0
Salario_Anual    200
Deuda      150
dtype: int64

Verificación de valores nulos después de limpieza
Edad      0
Genero    0
Historial_Crediticio  0
Salario_Anual    0
Deuda      0
dtype: int64

Guardado de datos limpios
None
<ipython-input-6-9e3e1a895789>:10: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment...
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values is a copy.
For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value, inplace=True)

df_financiero['Salario_Anual'].fillna(df_financiero['Salario_Anual'].median(), inplace=True)
<ipython-input-6-9e3e1a895789>:11: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment...
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values is a copy.
For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value, inplace=True)
```

31/10/24, 1:40 p.m.

Laboratorio Sesión 20 - Colab

```
plt.figure(figsize=(10, 6))
plt.scatter(df_financiero['Edad'], df_financiero['Deuda'], c='green', alpha=0.5)
plt.title('Relación entre Edad y Deuda', fontsize=15)
plt.xlabel('Edad', fontsize=12)
plt.ylabel('Deuda', fontsize=12)
plt.grid(True)

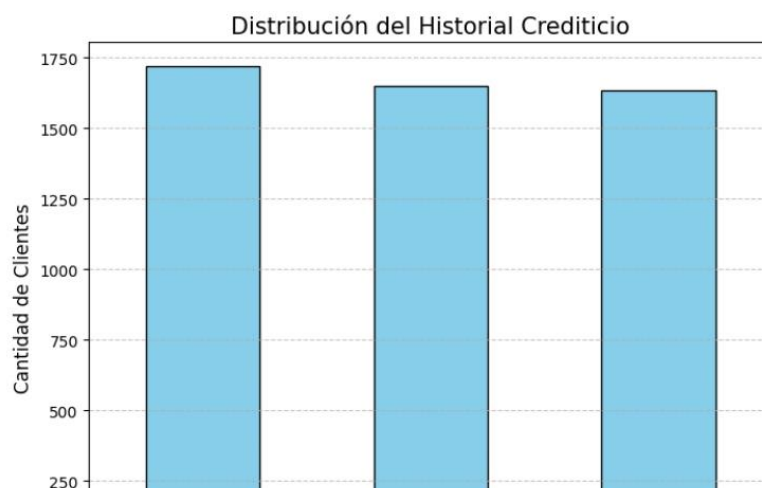
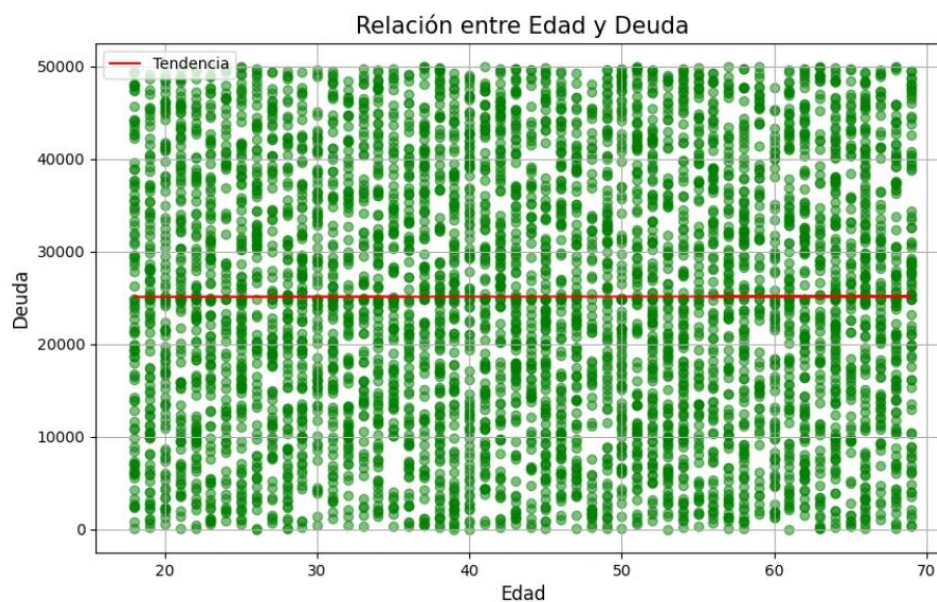
z = np.polyfit(df_financiero['Edad'], df_financiero['Deuda'], 1)
p = np.polyd(z)
plt.plot(df_financiero['Edad'], p(df_financiero['Edad']), color='red', label='Tendencia')
plt.legend()

plt.show()

plt.figure(figsize=(8, 6))
df_financiero['Historial_Crediticio'].value_counts().plot(kind='bar', color='skyblue', edgecolor='black')
plt.title('Distribución del Historial Crediticio', fontsize=15)
plt.xlabel('Historial Crediticio', fontsize=12)
plt.ylabel('Cantidad de Clientes', fontsize=12)

plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.show()
```



<https://colab.research.google.com/drive/1apXFINGo05R1798rhj1WkcRSg5IMt5qa?usp=sharing#printMode=true>

2/5

Parte 2

En la segunda parte una vez realizado el proceso de la visualización de los datos con Python: data, exploración, limpieza y gráficas personalizadas, se deberán guardar los archivos en el repositorio, deberás anexar la captura de pantalla con el nombre del archivo.

```
31/10/24, 1:40 p.m. Laboratorio Sesión 20 - Colab
```

```
df_salud = pd.read_csv('./source/datos_salud glu.csv')
print(f"Valores Originales:
(df_salud.describe())")

print(f"Verificación valores nulos:
(df_salud.isnull().sum())")

df_salud['Glucosa'].fillna(df_salud['Glucosa'].median(), inplace=True)
df_salud['IMC'].fillna(df_salud['IMC'].median(), inplace=True)

print(f"Verificación valores nulos después de limpieza:
(df_salud.isnull().sum())")

print(f"Guardado de datos limpios:
(df_salud.to_csv('./source/clean/datos_salud_limpios.csv'))")
```

| Valores Originales: | | | |
|---------------------|-------------|-------------|-------------|
| | Edad | Glucosa | IMC |
| count | 5000.000000 | 4800.000000 | 4850.000000 |
| mean | 53.299000 | 133.858125 | 26.708239 |
| std | 20.646851 | 37.196858 | 4.808900 |
| min | 18.000000 | 70.000000 | 18.502603 |
| 25% | 36.000000 | 101.000000 | 22.515387 |
| 50% | 53.000000 | 134.000000 | 26.613110 |
| 75% | 71.000000 | 165.000000 | 30.979322 |
| max | 89.000000 | 199.000000 | 34.998760 |

```
Verificación valores nulos:
Edad      0
Genero    0
Glucosa   200
IMC       150
Antecedentes_Familiares  0
dtype: int64

Verificación valores nulos después de limpieza:
Edad      0
Genero    0
Glucosa   0
IMC       0
Antecedentes_Familiares  0
dtype: int64

Guardado de datos limpios:
None
<ipython-input-27-981a0346ab86>:8: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting value
For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value, inplace=True)

df_salud['Glucosa'].fillna(df_salud['Glucosa'].median(), inplace=True)
<ipython-input-27-981a0346ab86>:9: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting value
For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value, inplace=True)

df_salud['IMC'].fillna(df_salud['IMC'].median(), inplace=True)
```

```
plt.figure(figsize=(10, 6))
plt.scatter(df_salud['Edad'], df_salud['Glucosa'], c='blue', alpha=0.5)
plt.title('Relación entre Edad y Niveles de Glucosa', fontsize=15)
plt.xlabel('Edad', fontsize=12)
plt.ylabel('Niveles de Glucosa', fontsize=12)
plt.grid(True)

z = np.polyfit(df_salud['Edad'], df_salud['Glucosa'], 1)
p = np.poly1d(z)
plt.plot(df_salud['Edad'], p(df_salud['Edad']), color='red', label='Tendencia')
plt.legend()

plt.show()

plt.figure(figsize=(8, 6))
df_salud['Antecedentes_Familiares'].value_counts().plot(kind='bar', color='green', edgecolor='black')
plt.title('Distribución de Antecedentes Familiares de Diabetes', fontsize=15)
plt.xlabel('Antecedentes Familiares', fontsize=12)
```

<https://colab.research.google.com/drive/1apXFINGo05R1798rhj1WkCRSg5IMt5qa?usp=sharing#printMode=true>

3/5


```
plt.ylabel('Cantidad de Pacientes', fontsize=12)
```

```
plt.grid(axis='y', linestyle='--', alpha=0.7)
```

```
plt.show()
```

(2)

