Análisis Exploratorio (EDA)

Breve introducción al análisis exploratorio de datos - Feregrino
https://www.youtube.com/watch?v=s7A1OxEeT6o

¿Cómo hacer el ANÁLISIS EXPLORATORIO DE DATOS?: guía paso a paso - Codificando Bits
https://www.youtube.com/watch?v=_-KW4gT_oGU&t=352s

¿Qué es el Análisis Exploratorio de Datos (EDA)?

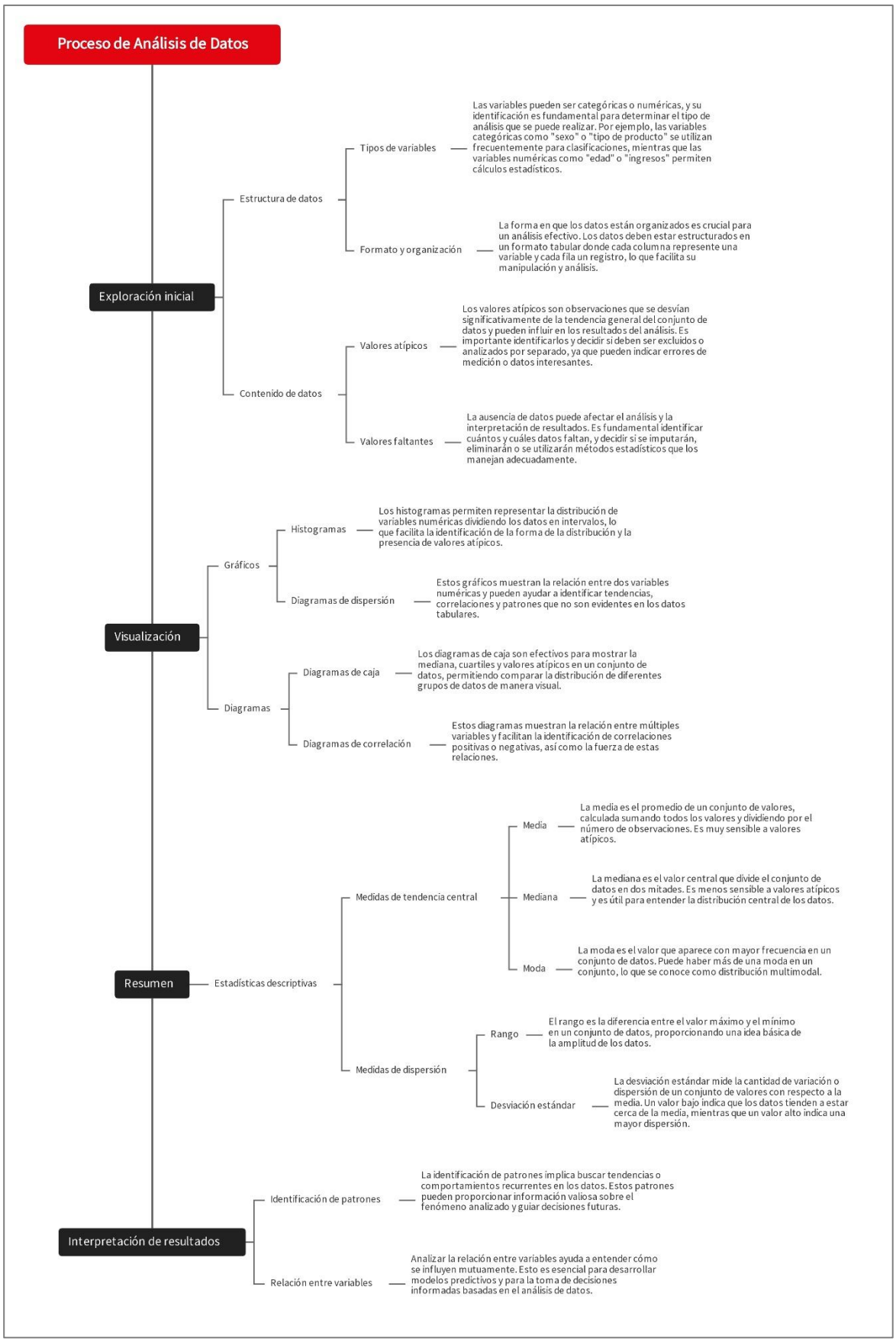
El EDA es como una lupa para científicos de datos. Nos permite sumergirnos en nuestros conjuntos de datos y descubrir patrones, tendencias y anomalías que a simple vista podrían pasar desapercibidos. Al visualizar y analizar la información de formas creativas, los científicos de datos obtienen una comprensión profunda de sus datos, lo que les permite tomar decisiones más informadas y construir modelos más precisos.

¿Cómo funciona?

El EDA es un proceso iterativo que implica:

1. **Exploración inicial:** Un primer vistazo a los datos para familiarizarse con su estructura y contenido.
2. **Visualización:** Crear gráficos y diagramas que revelen patrones y relaciones entre las variables.
3. **Resumen:** Calcular estadísticas descriptivas para obtener una visión general de los datos.
4. **Transformación:** Limpiar y transformar los datos si es necesario para mejorar su calidad.

Análisis exploratorio - Python - Actividad No 1



Análisis exploratorio - Python - Actividad No 1

Tipos de Archivos para Almacenar Grandes Cantidades de Datos

La elección del formato de archivo para almacenar grandes cantidades de datos depende en gran medida de la naturaleza de los datos, el tamaño del conjunto de datos, y el propósito para el cual se almacenarán. A continuación, te presento algunos de los formatos más comunes y sus características:

Formatos de Texto

- ✓ **CSV (Comma-Separated Values):**
  - Muy común y fácil de leer tanto por humanos como por máquinas.
  - Cada línea representa una fila y cada valor está separado por una coma.
  - Ideal para datos tabulares simples.
- ✓ **TSV (Tab-Separated Values):**
  - Similar a CSV, pero utiliza tabulaciones como separadores.
  - Puede ser más legible cuando los datos contienen comas.
- ✓ **JSON (JavaScript Object Notation):**
  - Formato ligero y legible por humanos.
  - Utiliza pares clave-valor para estructurar los datos.
  - Ideal para datos estructurados en formato jerárquico.
- ✓ **XML (eXtensible Markup Language):**
  - Muy flexible y extensible.
  - Utiliza etiquetas para definir la estructura de los datos.
  - Puede ser más complejo que JSON.

Formatos Binarios

- ✓ **HDF5 (Hierarchical Data Format 5):**
  - Diseñado para almacenar grandes cantidades de datos numéricos.
  - Permite crear jerarquías de grupos y datasets.
  - Ideal para datos científicos y de alta dimensión.
- ✓ **Parquet:**
  - Formato columnar eficiente para almacenar grandes conjuntos de datos.
  - Compresión eficiente y soporte para esquemas.
  - Utilizado ampliamente en el ecosistema de big data.
- ✓ **ORC (Optimized Row Columnar):**
  - Formato columnar similar a Parquet.
  - Desarrollado por Apache Hive.
  - Ofrece una buena compresión y rendimiento.

Bases de Datos

- ✓ **Relacionales:**
  - MySQL, PostgreSQL, SQL Server.
  - Almacenan datos en tablas relacionadas.
  - Ideal para datos estructurados y consultas complejas.
- ✓ **NoSQL:**
  - MongoDB, Cassandra, HBase.
  - Más flexibles que las bases de datos relacionales.
  - Ideales para grandes volúmenes de datos no estructurados o semiestructurados.
- ✓ **Cloud-based:**
  - Amazon S3, Google Cloud Storage, Azure Blob Storage.
  - Almacenamiento en la nube escalable y duradero.
  - Ideal para grandes volúmenes de datos y análisis de datos.

El EDA se utiliza para:

Descubrir patrones ocultos: Identificar relaciones entre variables que podrían ser útiles para la predicción o la clasificación.

Detectar anomalías: Encontrar valores atípicos que podrían indicar errores en los datos o fenómenos interesantes.

Verificar hipótesis: Confirmar o refutar supuestos sobre los datos.

Seleccionar las técnicas de análisis adecuadas: Elegir las herramientas estadísticas más apropiadas para responder a las preguntas de investigación.

El EDA es una herramienta fundamental en el arsenal de cualquier científico de datos, ya que proporciona una base sólida para cualquier análisis posterior.

EJERCICIO

A continuación, se llevará a cabo un análisis exploratorio de los datos utilizando el lenguaje de programación Python, soportado por entornos de desarrollo como Anaconda Navigator, Colaboratory o Visual Studio Code.

1. Configuración del Entorno de Trabajo

Selecciona tu herramienta: Elige entre Anaconda Navigator, Colaboratory o Visual Studio Code. Cada una ofrece ventajas diferentes en términos de facilidad de uso, recursos y funcionalidades.

Crea un nuevo proyecto: Inicia un nuevo proyecto en tu herramienta seleccionada. Esto te permitirá organizar tus archivos y código de manera eficiente.

Crea un entorno: crea un entorno para instalar lo necesario.

Instala las librerías necesarias: Asegúrate de tener instaladas las siguientes librerías en tu entorno, por ejemplo:

Pandas: Para manipulación y análisis de datos.

NumPy: Para operaciones numéricas.

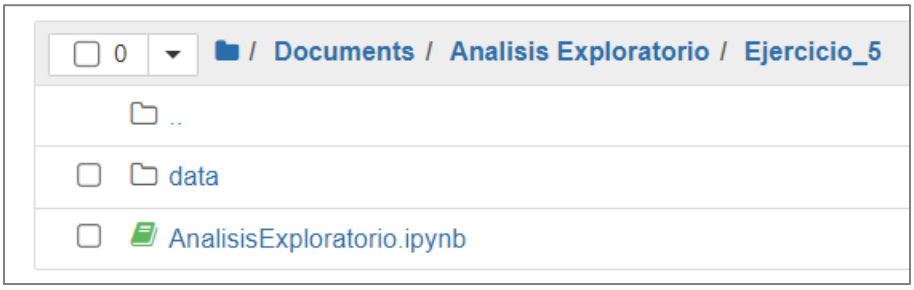
Matplotlib: Para crear visualizaciones estáticas.

Seaborn: Para crear visualizaciones más estéticas y complejas.

2. Carga de los Datos

El conjunto de datos que utilizaremos para este ejercicio se encuentra en la plataforma de datos abiertos de Colombia, para explorar las tendencias en la adopción de tecnologías digitales en Colombia, el siguiente es el enlace donde encontraras el conjunto de datos https://www.datos.gov.co/Ciencia-Tecnolog-a-e-Innovacin/Ciudadan-a-Digital/g4cd-bvpd/about_data. Este conjunto de datos, proporcionado por el gobierno colombiano, contiene información detallada sobre los programas de alfabetización digital, el acceso a internet y el uso de tecnologías de la información y la comunicación por parte de los ciudadanos.

Crea la estructura del proyecto como por ejemplo, si utilizamos ANACONDA.NAVIGATOR :



Análisis exploratorio - Python - Actividad No 1

Para iniciar la implementación práctica, crea un nuevo cuaderno de Jupyter. En este cuaderno, podrás escribir y ejecutar el código Python necesario para cargar los datos, realizar la limpieza y exploración, y obtener los resultados del análisis.

Importa las librerías:

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

- ✓ Escribe para que se utilizan las anteriores librerías.

Carga el dataset:

```
df = pd.read_csv('./data/Ciudadan_a_Digital_20241006.csv', encoding='utf-8')
df
```

Impresión de los datos:

```
# Primeras 5 registros del dataframe
df.head(5)
```

```
# Ultimas 5 registros del dataframe
df.tail(5)
```

```
# 5 registros del dataframe aleatoriamente
df.sample(5)
```

Exploración inicial:

```
df.shape
```

```
df.info()
```

```
df.describe()
```

```
print(f"Cantidad de filas: {len(df)}")
print(f"Dimensiones: {df.shape}")
print(f"Cantidad de valores: {df.size}")
print("Columnas: \n", *df.columns + '\n')
```

- ✓ Escribe con detalle cuales son los resultados.

Contar valores nulos:

```
# Contar los valores nulos por columna
conteo_nulos = df.isnull().sum()
print(conteo_nulos)
```

Es importante conocer cuántos datos nulos hay en una columna durante un análisis exploratorio de datos (EDA) por varias razones:

Calidad de los datos: La presencia de datos nulos indica posibles problemas de calidad, como registros incompletos o errores en la recolección de datos. Detectar los valores faltantes es el primer paso para decidir cómo tratarlos (eliminarlos, imputarlos, etc.).

Decisiones sobre el tratamiento: Dependiendo de la cantidad de valores nulos, puedes decidir si:

Eliminar las filas o columnas afectadas (si el porcentaje de datos nulos es bajo).

Imputar valores (por ejemplo, con la media, mediana, moda o utilizando modelos de predicción).

Mantener los datos nulos si son significativos o reflejan una característica importante del conjunto de datos.

Evitar sesgos: Si los valores nulos no se gestionan adecuadamente, pueden generar sesgos en los resultados del análisis. Por ejemplo, si eliminas demasiadas filas, puedes perder información valiosa o subrepresentar ciertas características.

Impacto en los modelos: Muchos algoritmos de machine Learning no manejan bien los valores nulos, lo que puede provocar errores o resultados inexactos. Conocer cuántos valores nulos tienes te permite prepararte para limpiarlos o imputarlos antes de entrenar modelos.

Exploración inicial: El análisis de los valores nulos te da una visión clara del estado general de tus datos y te ayuda a identificar patrones o posibles causas de los datos faltantes.

Valores almacenados en columnas:

Es importante obtener información sobre la cantidad de valores únicos (distintos) en cada columna.

```python
# Al iterar un df, se obtienen los nombres de las columnas
for columna in df:

    # Una forma de acccesos a las columnas es como usarlo como un diccionario
    datos_columna = df[columna]

    # Cantidad de valores unicos
    distintos = datos_columna.nunique()

    print(f"La columna {columna} tiene {distintos} valores diferentes.")
```

- ✓ Escribe con detalle cuales son los resultados.

Eliminación de columnas irrelevantes:

Durante el análisis exploratorio de datos, es común identificar columnas que no aportan valor al estudio. Si una columna contiene un alto porcentaje de datos nulos o está completamente vacía, puede considerarse irrelevante para el análisis. En estos casos, una práctica recomendada es eliminarla, ya que la falta de información útil impide que contribuya de manera significativa a nuestros objetivos. Esta técnica ayuda a optimizar el dataset, reduciendo su complejidad y enfocándonos en las variables que realmente influyen en el ejercicio.

Para practicar con los siguientes códigos, es importante que determines qué columnas deseas eliminar según la cantidad de valores nulos que contienen. Evalúa el umbral adecuado de datos faltantes para cada caso. Por ejemplo, puedes decidir eliminar columnas que tengan más de un cierto porcentaje de valores nulos o aquellas que estén completamente vacías. La elección dependerá de la relevancia de

la información en cada columna y de cuántos datos faltantes puedes tolerar sin comprometer la calidad del análisis.

Borrar columnas con al menos un dato vacío:

```
# Si deseas eliminar las columnas que tienen al menos un valor nulo
df_sin_nulos = df.dropna(axis=1)
```

Con el siguiente código puedes determinar si las columnas fueron borradas.

```
# Contar los valores nulos por columna
conteo_nulos = df_sin_nulos.isnull().sum()
print(conteo_nulos)
```

Borrar columnas con al menos un datos vacíos:

```
# Si deseas eliminar las columnas que tienen todos sus valores vacios
df_sin_nulos_totales = df.dropna(axis=1, how='all')
```

Con el siguiente código puedes determinar si las columnas fueron borradas.

```
# Contar los valores nulos por columna
conteo_nulos = df_sin_nulos_totales.isnull().sum()
print(conteo_nulos)
```

Eliminar columnas con un porcentaje alto de datos nulos:

```
# solo se mantendrá las columnas que tienen al menos el 70% de sus datos no nulos
limite = len(df) * 0.7
df_sin_muchos_nulos = df.dropna(axis=1, thresh=limite)
```

Con el siguiente código puedes determinar si las columnas fueron borradas.

```
# Contar los valores nulos por columna
conteo_nulos = df_sin_muchos_nulos.isnull().sum()
print(conteo_nulos)
```

Análisis exploratorio - Python - Actividad No 1

Bibliografía

https://www.ibm.com/mx-es/topics/exploratory-data-analysis

https://www.datos.gov.co/Salud-y-Protecci-n-Social/Casos-positivos-de-COVID-19-en-Colombia-/gt2j-8ykr/data

https://www.datos.gov.co/Ciencia-Tecnolog-a-e-Innovaci-n/Ciudadan-a-Digital/g4cd-bvpd/about_data