



[LCD] MÓDULO 6

Bienvenido al **Módulo 6** del curso **Lenguaje de Ciencia de Datos I.**

INTRODUCCIÓN

Estimado estudiante:

En el presente módulo aprenderás a trabajar con diferentes fuentes de información mediante Python, comprendiendo cómo leer, limpiar y exportar archivos en diversos formatos, acceder a bases de datos relacionales y NoSQL, y aplicar modelos de inteligencia artificial preentrenados como los de Hugging Face. Además, explorarás cómo los modelos predictivos permiten anticipar resultados y optimizar decisiones basadas en datos.

Te deseamos una excelente experiencia de aprendizaje en este curso.

¡Muchos éxitos!

≡ **TEMA 6: MANEJO DE DATOS Y APLICACIONES MODERNAS**

≡ **CIERRE**

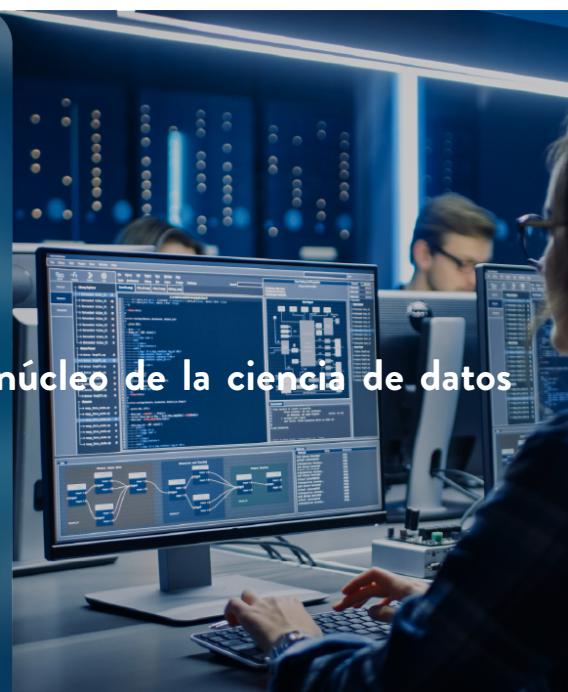
TEMA 6: MANEJO DE DATOS Y APLICACIONES MODERNAS

MANEJO DE DATOS Y APLICACIONES MODERNAS

Llegó el momento para que te conviertas en el protagonista de tu aprendizaje.

- Descubre y analiza la información que se presenta en la lección interactiva para transformarla en nuevo conocimiento.

Transformar datos en conocimiento es el núcleo de la ciencia de datos moderna.





El manejo moderno de datos es una competencia esencial en la ciencia de datos y la analítica avanzada. En este tema se exploran las herramientas y técnicas que permiten integrar, limpiar, transformar y analizar información proveniente de diversas fuentes. Desde la lectura de archivos y la conexión con bases de datos, hasta el uso de modelos de inteligencia artificial y algoritmos predictivos, el objetivo es convertir los datos en conocimiento útil para la toma de decisiones estratégicas.

Lectura y Escritura de Archivos

La lectura y escritura de archivos constituye una de las tareas fundamentales en la ciencia de datos, ya que permite acceder, transformar y almacenar información en diversos formatos según el tipo de proyecto o fuente de origen. Esta etapa inicial garantiza la correcta manipulación de los datos antes de aplicar cualquier tipo de análisis o modelo predictivo.





Archivos planos

Los **archivos planos** como .csv y .txt son los más utilizados para representar datos tabulares de manera sencilla. Se emplean ampliamente por su compatibilidad con casi todas las herramientas de análisis.



Archivos jerárquicos

Los **archivos jerárquicos**, como `.json`, son esenciales para intercambiar información entre aplicaciones web y APIs, ya que permiten estructurar los datos en formato de árbol o diccionario.



Archivos binarios

Los **archivos binarios**, como .xlsx o .parquet, se caracterizan por su capacidad para almacenar grandes volúmenes de datos con alta eficiencia y menor tamaño, lo que los hace ideales para entornos empresariales y proyectos de **Big Data**.



Librerías

Python, mediante librerías como **pandas**, **json** y **openpyxl**, ofrece una amplia versatilidad para leer, procesar y exportar datos en múltiples formatos, facilitando su integración en flujos de trabajo analíticos automatizados.

La **entrada/salida de datos (I/O)** representa la primera fase de cualquier pipeline de ciencia de datos, ya que de su correcta ejecución depende la calidad del análisis posterior.

Principales formatos

Los formatos de archivo son esenciales en el manejo y análisis de datos, ya que determinan la forma en que la información se organiza, almacena y comparte entre distintas herramientas y sistemas. Conocer

sus características y usos permite seleccionar el tipo de archivo más adecuado según el contexto del proyecto, optimizando la eficiencia del procesamiento y la compatibilidad con las librerías de análisis utilizadas en Python.



CSV (Comma-Separated Values)

Formato estándar para datos tabulares, simple y legible.



JSON (JavaScript Object Notation)

Usado en APIs, ideal para representar estructuras complejas o anidadas.



TXT

Archivos de texto plano, útiles para procesamiento de lenguaje natural (NLP).



XLSX

Formato de Excel, común en entornos corporativos.



Parquet/Feather

Formatos binarios de alto rendimiento utilizados en proyectos de Big Data.

Librerías de Python

En la ciencia de datos, las librerías de Python son herramientas fundamentales que facilitan la manipulación, transformación y almacenamiento de la información. Cada librería ofrece funciones especializadas que automatizan tareas comunes como la lectura de archivos, la limpieza de datos o la exportación de resultados. Dominar su uso permite optimizar los procesos analíticos y trabajar de manera más eficiente con grandes volúmenes de información.

- **pandas:** manipulación de datos tabulares y operaciones de limpieza.
- **json:** lectura y escritura de estructuras jerárquicas (diccionarios y listas).
- **openpyxl:** manejo de hojas de cálculo en formato Excel.



En la práctica, dominar la lectura y escritura de archivos permite al analista integrar datos heterogéneos, garantizar su consistencia y preparar una base sólida para el análisis estadístico y el aprendizaje automático.

Ejemplos

EJEMPLO 1 EN PYTHON

EJEMPLO 2 EN PYTHON

APLICACIÓN PRÁCTICA
DESARROLLADA

```
import pandas as pd
import json

# Lectura de archivo CSV
df = pd.read_csv("ventas.csv")

# Lectura de JSON
with open("config.json", "r") as file:
    config = json.load(file)

# Escritura en Excel
df.to_excel("ventas_procesadas.xlsx", index=False)

# Escritura en formato Parquet (óptimo para Big Data)
df.to_parquet("ventas.parquet", index=False)
```

EJEMPLO 1 EN PYTHON

EJEMPLO 2 EN PYTHON

APLICACIÓN PRÁCTICA
DESARROLLADA

```
import pandas as pd
import json

# Lectura de CSV
ventas = pd.read_csv("ventas.csv")
```

```
# Lectura de JSON  
with open("productos.json") as f:  
    productos = json.load(f)  
  
# Escritura en múltiples formatos  
ventas.to_excel("ventas.xlsx", index=False)  
ventas.to_json("ventas.json", orient="records")
```

EJEMPLO 1 EN PYTHON

EJEMPLO 2 EN PYTHON

APLICACIÓN PRÁCTICA
DESARROLLADA

Un analista recibe un archivo ventas.csv con errores en mayúsculas/minúsculas en los nombres de las columnas y valores nulos. La tarea es **normalizar los nombres, eliminar nulos y exportar un archivo final en Excel para el área de finanzas.**

```
# Normalización y limpieza  
df.columns = [col.strip().lower() for col in df.columns]  
df = df.dropna()  
df.to_excel("ventas_limpias.xlsx", index=False)
```

Aplicaciones prácticas

Las aplicaciones prácticas del manejo de archivos en ciencia de datos permiten trasladar los conocimientos teóricos a contextos reales de trabajo, en los que la correcta lectura, integración y exportación de

información garantiza la eficiencia de los procesos analíticos. Estas prácticas son esenciales para profesionales que gestionan grandes volúmenes de datos y buscan optimizar la toma de decisiones basada en evidencia.

1. **Finanzas:** consolidar reportes financieros provenientes de múltiples hojas de cálculo, automatizar su limpieza con pandas y almacenarlos en formatos eficientes como Parquet o CSV para facilitar su análisis posterior.
2. **Salud:** combinar datos clínicos estructurados en CSV con información jerárquica en JSON procedente de sensores o aplicaciones médicas, permitiendo un monitoreo más preciso del estado del paciente.
3. **Educación:** procesar bases de datos de calificaciones en Excel, identificar tendencias de rendimiento académico y generar visualizaciones que apoyen la toma de decisiones pedagógicas e institucionales.



Estas aplicaciones demuestran cómo la correcta gestión de archivos impulsa la productividad y la calidad del análisis en distintos sectores profesionales.

CONTINUAR

Acceso a Bases de Datos

El acceso a bases de datos constituye una habilidad clave para los científicos de datos, ya que la mayoría de la información en entornos reales se encuentra almacenada en sistemas de gestión estructurados. Comprender cómo conectarse, consultar y manipular datos directamente desde Python permite integrar el análisis estadístico con la infraestructura tecnológica de una organización, reduciendo errores y tiempos de procesamiento.

- Los **datos en producción** suelen residir en bases de datos que almacenan grandes volúmenes de información de manera organizada y segura.
- Conectar Python a estos sistemas permite automatizar consultas, actualizar registros y extraer datos relevantes para el análisis sin necesidad de exportarlos manualmente.

Bases de datos relacionales (SQL)

Estas bases de datos almacenan la información en tablas relacionadas entre sí mediante claves primarias y foráneas, lo que garantiza la integridad y consistencia de los datos.

Se utilizan ampliamente en aplicaciones empresariales, financieras y administrativas.

Organizan datos en filas y columnas, siguiendo un esquema estructurado.

Ejemplos: MySQL, PostgreSQL, SQLite, MariaDB.

Bases de datos (NoSQL)

Diseñadas para trabajar con datos no estructurados o semiestructurados, estas bases de datos ofrecen mayor flexibilidad y escalabilidad en comparación con las tradicionales. Son ideales para proyectos que manejan grandes volúmenes de datos o información diversa, como texto, imágenes o JSON.

Permiten un acceso rápido y flexible, adaptándose a diferentes tipos de datos.

Ejemplos: MongoDB, Cassandra, Redis.

Conectores en Python

Python proporciona diversas bibliotecas que permiten conectar el entorno de análisis con las bases de datos y ejecutar consultas SQL o NoSQL dentro del mismo flujo de trabajo:



Dominar la conexión e integración de Python con diferentes tipos de bases de datos garantiza un flujo de análisis más eficiente, seguro y adaptable a las necesidades empresariales y científicas.

Ejemplos

EJEMPLO 1 EN PYTHON (SQLITE Y SQLALCHEMY)

```
import sqlite3
```

EJEMPLO 2 EN PYTHON (MONGODB)

APLICACIÓN PRÁCTICA
DESARROLLADA

```
import pandas as pd
from sqlalchemy import create_engine

# Conexión SQLite
conn = sqlite3.connect("empresa.db")
df = pd.read_sql_query("SELECT * FROM empleados", conn)

# Conexión SQLAlchemy (ejemplo con PostgreSQL)
engine = create_engine("postgresql://user:pass@localhost:5432/empresa")
df_postgres = pd.read_sql("SELECT nombre, salario FROM empleados", engine)
```

EJEMPLO 1 EN PYTHON (SQLITE Y
SQLALCHEMY)

EJEMPLO 2 EN PYTHON (MONGODB)

APLICACIÓN PRÁCTICA
DESARROLLADA

```
from pymongo import MongoClient

# Conexión a MongoDB
cliente = MongoClient("mongodb://localhost:27017/")
db = cliente["empresa"]
colección = db["empleados"]

# Consulta
for empleado in colección.find({"salario": {"$gt": 5000}}):
    print(empleado)
```

EJEMPLO 1 EN PYTHON (SQLITE Y
SQLALCHEMY)

EJEMPLO 2 EN PYTHON (MONGODB)

APLICACIÓN PRÁCTICA
DESARROLLADA

En **Recursos Humanos**, se consulta la tabla de empleados para calcular el **salario promedio por departamento**.

```
query = """
SELECT departamento, AVG(salario) as salario_promedio
FROM empleados
GROUP BY departamento
"""

salarios = pd.read_sql_query(query, conn)
print(salarios)
```

Aplicaciones prácticas

El acceso a bases de datos permite integrar el análisis de datos con los sistemas reales de una organización, automatizando la recolección y actualización de información. Su aplicación práctica abarca diversos sectores productivos, donde los datos constituyen la base para la toma de decisiones estratégicas y la optimización de procesos.

1. **Recursos humanos:** extraer datos de empleados desde bases SQL para calcular métricas de desempeño, identificar patrones de rotación y diseñar políticas de retención basadas en evidencia.

2. **Retail:** consultar inventarios en bases SQL para planificar el abastecimiento de productos, prever la demanda estacional y mantener niveles óptimos de stock mediante modelos predictivos.
3. **Logística:** integrar datos en tiempo real provenientes de sensores IoT (Internet of Things) almacenados en bases NoSQL, permitiendo monitorear el transporte, la ubicación y el estado de los productos en toda la cadena de suministro.



Estas aplicaciones demuestran cómo la conexión entre Python y las bases de datos impulsa la eficiencia operativa, mejora la gestión de información y convierte los datos en herramientas de valor estratégico.

CONTINUAR

Introducción a Hugging Face

Hugging Face es una de las plataformas más relevantes en el campo de la **inteligencia artificial (IA)**, especialmente en el área del **procesamiento del lenguaje natural (NLP)**, la **visión por computadora** y el **reconocimiento de voz**.

(speech). Su principal contribución radica en ofrecer una amplia colección de **modelos preentrenados** que pueden ser reutilizados y adaptados a diversas tareas de análisis sin necesidad de iniciar el entrenamiento desde cero.

La plataforma funciona a través de su librería **Transformers**, que permite descargar modelos avanzados —como **BERT**, **GPT**, **DistilBERT** o **T5**— mediante unas pocas líneas de código en Python. Estos modelos, entrenados con millones de datos, facilitan la automatización de tareas complejas y el acceso a tecnologías de vanguardia para el análisis de texto, audio e imágenes.

Ventajas

Antes de enumerar sus ventajas, es importante destacar que Hugging Face ha transformado la forma en que se desarrollan aplicaciones de inteligencia artificial. Gracias a su librería Transformers, los usuarios pueden acceder a modelos ya entrenados por expertos y aplicarlos directamente a sus proyectos. Esto facilita el trabajo en áreas como el análisis de texto, la traducción o la generación automática de contenido, acelerando el desarrollo y mejorando la precisión de los resultados.

- Ahorro de recursos: evita el entrenamiento desde cero, reduciendo tiempo y costos computacionales.
- Comunidad activa y datasets gratuitos: permite compartir, mejorar y adaptar modelos con soporte constante.

- Compatibilidad multilingüe: ofrece modelos entrenados en varios idiomas, incluido el español.
- Modelos de última generación: actualizados con las arquitecturas más modernas de IA.
- Fácil integración: los modelos pueden incorporarse directamente a pipelines de análisis en Python o sistemas empresariales.



Áreas de aplicación



Análisis de sentimientos:

Clasificación automática de opiniones positivas, negativas o neutras en redes sociales o encuestas.



Chatbots:

Creación de asistentes conversacionales inteligentes que comprenden y responden en lenguaje natural.



Resumen automático de documentos:

Condensación de textos extensos en versiones más breves y comprensibles.



Traducción automática:

Conversión de textos entre idiomas de forma precisa y contextual.



Clasificación de texto:

Identificación de categorías o temas en grandes volúmenes de información.

 Hugging Face representa una herramienta democratizadora del aprendizaje automático, que permite a los científicos de datos y desarrolladores aplicar modelos de inteligencia artificial de alto nivel sin requerir una infraestructura compleja o conocimiento profundo de redes neuronales.

Ejemplos

EJEMPLO 1 EN PYTHON

EJEMPLO 2 EN PYTHON

APLICACIÓN PRÁCTICA
DESARROLLADA

```
from transformers import pipeline

# Análisis de sentimientos
análisis = pipeline("sentiment-analysis")
print(análisis("El producto llegó tarde y estaba dañado."))

# Traducción
traductor = pipeline("translation_en_to_es")
print(traductor("Data science is the new oil."))
```

EJEMPLO 1 EN PYTHON

EJEMPLO 2 EN PYTHON

APLICACIÓN PRÁCTICA
DESARROLLADA

```
from transformers import pipeline

# Resumen automático
resumen = pipeline("summarization")
texto = """La inteligencia artificial está revolucionando la educación,
permitiendo personalizar el aprendizaje y optimizar los recursos educativos."""
print(resumen(texto, max_length=20, min_length=10))
```

```
# Clasificación de texto
clasificador = pipeline("text-classification")
print(clasificador("El servicio fue excelente y rápido."))
```

EJEMPLO 1 EN PYTHON

EJEMPLO 2 EN PYTHON

APLICACIÓN PRÁCTICA
DESARROLLADA

Una empresa de **e-commerce** quiere analizar las opiniones de clientes en Amazon.

```
comentarios = [
    "The laptop works perfectly, I am very satisfied!",
    "Terrible service, I will not buy again."
]
```

```
sentimientos = análisis(comentarios)
print(sentimientos)
```

Interpretación: los resultados se pueden usar para construir un **dashboard de satisfacción de clientes** en tiempo real.

Aplicaciones prácticas

El uso de modelos preentrenados de Hugging Face permite implementar soluciones de inteligencia artificial en distintos sectores, sin necesidad de contar con infraestructura avanzada o grandes volúmenes de datos de entrenamiento. Estas herramientas aceleran los procesos analíticos y mejoran la toma de decisiones basada en información textual.

- **Marketing digital:** clasificar reseñas de clientes como positivas o negativas para medir la satisfacción, identificar áreas de mejora y diseñar estrategias de comunicación más efectivas. También puede emplearse para monitorear menciones de marca en redes sociales.
- **Educación:** generar resúmenes automáticos de artículos académicos, tesis o informes extensos, facilitando la comprensión y el acceso rápido a la información clave. Asimismo, puede utilizarse para clasificar textos educativos por nivel o tema.
- **Gobierno:** analizar la opinión pública en redes sociales y medios digitales, detectar tendencias de conversación ciudadana y evaluar la percepción de políticas públicas mediante análisis de sentimientos y clasificación temática.



Estas aplicaciones evidencian cómo los modelos de Hugging Face amplían las posibilidades del análisis

automatizado del lenguaje en contextos reales, impulsando la eficiencia y la innovación en distintos ámbitos profesionales.

CONTINUAR

Aplicación Práctica de Modelos Predictivos

Los **modelos predictivos** son una de las herramientas más poderosas de la ciencia de datos, ya que permiten **anticipar comportamientos o eventos futuros** a partir del análisis de patrones históricos. Su aplicación se extiende a diversos campos como las **ventas, la gestión de riesgos, el mantenimiento industrial, la salud o la detección de fraudes financieros**.

Un **modelo predictivo** es una función matemática o computacional que, utilizando un conjunto de variables independientes, estima o predice el valor de una variable dependiente. Estos modelos ayudan a las organizaciones a **tomar decisiones proactivas**, basadas en la evidencia y no solo en la intuición.

Principales algoritmos

Los algoritmos predictivos son el núcleo del aprendizaje automático, ya que permiten a los sistemas identificar patrones en los datos y realizar estimaciones sobre eventos futuros. Cada algoritmo posee características y enfoques distintos, por lo que su elección depende del tipo de problema, la naturaleza de los datos y el objetivo del análisis.

Regresión lineal y logística

Se utilizan para predecir valores numéricos o clasificar resultados en categorías (por ejemplo, probabilidad de compra o riesgo crediticio).

Árboles de decisión y Random Forest

Permiten dividir los datos en grupos jerárquicos para obtener predicciones más precisas y fáciles de interpretar.

Redes neuronales

Imitan el funcionamiento del cerebro humano y se emplean para resolver problemas complejos con grandes volúmenes de datos, como el reconocimiento de imágenes o el análisis de texto.

Principales enfoques

Los **enfoques de modelado predictivo** determinan la forma en que un algoritmo aprende de los datos y genera resultados. Comprender estos enfoques es esencial para seleccionar la técnica adecuada según el tipo de información disponible y el objetivo del análisis, ya sea predecir valores conocidos, descubrir patrones ocultos o procesar grandes volúmenes de datos mediante redes neuronales.

Modelos supervisados

Requieren datos etiquetados para aprender la relación entre las variables (ejemplo: regresión lineal, SVM, árboles de decisión).

Modelos no supervisados

Identifican patrones ocultos o grupos naturales en los datos sin etiquetas previas, como en el algoritmo K-Means.

Deep Learning

Utiliza redes neuronales profundas capaces de descubrir relaciones complejas en grandes conjuntos de datos, siendo ampliamente usado en visión por computadora y procesamiento del lenguaje natural.

Ciclo de construcción de un modelo

El **ciclo de construcción de un modelo predictivo** comprende una serie de etapas que garantizan la calidad y confiabilidad de los resultados. Cada fase, desde la preparación de los datos hasta la evaluación final, cumple un papel clave en la creación de modelos capaces de aprender de la información y generar predicciones precisas aplicables a contextos reales.



En la práctica, el desarrollo de modelos predictivos permite transformar los datos históricos en

información estratégica, impulsando decisiones más informadas, precisas y alineadas con los objetivos de negocio o investigación.

Ejemplos

EJEMPLO 1 EN PYTHON (REGRESIÓN LINEAL)

```
from sklearn.linear_model import LinearRegression
import numpy as np
import pandas as pd

# Datos ficticios de ventas
X = np.array([[1], [2], [3], [4], [5]]) # semanas
y = np.array([200, 250, 300, 350, 400]) # ventas

# Entrenar modelo
modelo = LinearRegression()
modelo.fit(X, y)

# Predicción
print("Predicción para semana 6:", modelo.predict([[6]])[0])
```

EJEMPLO 2 EN PYTHON (ÁRBOL DE DECISIÓN)

EJEMPLO 1 EN PYTHON (REGRESIÓN LINEAL)

EJEMPLO 2 EN PYTHON (ÁRBOL DE DECISIÓN)

```
from sklearn.tree import DecisionTreeClassifier  
import numpy as np  
  
# Datos ficticios: [edad, ingresos]  
X = np.array([[25, 2000], [40, 4000], [30, 3000], [50, 6000]])  
y = ["No compra", "Compra", "Compra", "Compra"]  
  
modelo = DecisionTreeClassifier()  
modelo.fit(X, y)  
  
print(modelo.predict([[35, 3500]])) # ['Compra']
```

CONTINUAR

El manejo moderno de datos combina conocimiento técnico y pensamiento analítico para transformar la información en conocimiento útil. A través de herramientas como Python, bases de datos y modelos de inteligencia artificial, el estudiante adquiere la capacidad de integrar, analizar y predecir comportamientos, convirtiendo los datos en un recurso estratégico para la toma de decisiones en contextos reales.



CONTINUAR

Para finalizar el tema, recordemos algunas ideas claves.

1

Lectura y escritura de archivos: base de todo flujo de trabajo en ciencia de datos, con múltiples formatos (CSV, JSON, Excel, Parquet).

2

Acceso a bases de datos: integra Python con infraestructura empresarial, permitiendo consultas directas y análisis en tiempo real.

3

Hugging Face: democratiza el acceso a NLP de vanguardia para análisis de sentimientos, clasificación y traducción.

4

Modelos predictivos: transforman datos históricos en valor estratégico mediante la predicción de escenarios futuros.

CONTINUAR

CIERRE

