Capítulo 27: Análisis con Python

Guillermo Alejandro Zaragoza Alvarado[1], ORCID https://orcid.org/0009-0006-5466-7486
[1] Universidad Virtual del Estado de Guanajuato, Posgrados. Guanajuato, México.

ABSTRACT
The chapter addressed the usefulness of Python as a tool for applied statistical analysis, promoting its adoption in educational and academic environments. It was proposed that statistical analysis requires both technical tools and critical interpretation. A theoretical-practical approach was developed through examples with Python code, using libraries such as pandas, scipy, statsmodels, seaborn and matplotlib. Descriptive methods, hypothesis testing, correlation analysis, regression and an integrative case study with simulated data were applied. It was evident that Python allows applying statistical processes in a transparent and reproducible way. Statistical tests made it possible to identify significant relationships between variables and predict academic results based on data such as hours of study and attendance. The use of Python enhances the learning of statistics, facilitating the transition from the conceptual to the applied. It was concluded that incorporating programmable tools into the university curriculum favors a deep, autonomous and critical understanding of data analysis.

Keywords: Python, statistical analysis, higher education, regression, data visualization, inferential statistics.

RESUMEN
El capítulo abordó la utilidad de Python como herramienta para el análisis estadístico aplicado, promoviendo su adopción en entornos educativos y académicos. Se planteó que el análisis estadístico requiere tanto herramientas técnicas como interpretación crítica. Se desarrolló un enfoque teórico-práctico a través de ejemplos con código en Python, empleando bibliotecas como pandas, scipy, statsmodels, seaborn y matplotlib. Se aplicaron métodos descriptivos, pruebas de hipótesis, análisis de correlación, regresión y un estudio de caso integrador con datos simulados. Se evidenció que Python permite aplicar procesos estadísticos de forma transparente y reproducible. Las pruebas estadísticas permitieron identificar relaciones significativas entre variables y predecir resultados académicos a partir de datos como horas de estudio y asistencia. El uso de Python potencia el aprendizaje de la estadística, facilitando la transición de lo conceptual a lo aplicado. Se concluyó que incorporar herramientas programables en el currículo universitario favorece una comprensión profunda, autónoma y crítica del análisis de datos.

Palabras clave: Python, análisis estadístico, educación superior, regresión, visualización de datos, estadística inferencial.

1. INTRODUCTION

El crecimiento exponencial del volumen de datos generados globalmente es un fenómeno científicamente documentado, impulsado por la rápida proliferación de fuentes digitales y la sensorización masiva. Esta expansión está íntimamente ligada al ascenso de las tecnologías del Internet de las Cosas (IoT), que producen ingentes cantidades de datos en tiempo real, y a la intensificación de las interacciones en plataformas de redes sociales (Wu et al., 2014). Por ejemplo, los sistemas de monitorización científica, como los telescopios de nueva generación o los secuenciadores de ADN, generan volúmenes de *petabytes* (miles de billones de bytes) que superan las capacidades de gestión tradicionales (You et al., 2021). Este aumento sin precedentes no solo se caracteriza por el Volumen, sino también por la Velocidad y la Variedad de los datos, definiendo el paradigma actual del *Big Data* (Mayer-Schönberger & Cukier, 2013).

La necesidad de gestionar y analizar estos flujos de datos masivos ha catalizado innovaciones fundamentales en la arquitectura de *software*. La incapacidad de las bases de datos relacionales históricas para escalar eficientemente a estos volúmenes forzó la adopción de modelos de procesamiento distribuido (You et al., 2021). Específicamente, el desarrollo de *frameworks* como Hadoop y el modelo MapReduce permitió la manipulación escalable de conjuntos de datos no estructurados y semiestructurados en *clusters* de *hardware* commodity,

marcando un cambio fundamental en la ciencia de datos (Podhoranyi, 2020; Sandhu, 2021). En el ámbito empresarial, la capacidad de integrar y analizar estos datos heterogéneos es ahora esencial para la inteligencia de negocios y la toma de decisiones, consolidando el *Big Data* no solo como un reto tecnológico sino como un imperativo estratégico (Chen et al., 2013).

1.1. ¿Por qué usar Python en estadística?

Python es un lenguaje de programación de alto nivel que combina legibilidad, simplicidad sintáctica y una gran capacidad de extensión. Originalmente concebido por Guido van Rossum en 1991, su evolución ha estado fuertemente marcada por la comunidad científica y académica, convirtiéndose en el lenguaje preferido en muchas disciplinas que requieren análisis cuantitativo (Cuervo Diaz et al., 2024). En el informe *IEEE Spectrum* del 2023, Python fue clasificado como el lenguaje de programación más popular del mundo, por encima de Java, C++, y JavaScript, en gran parte debido a su uso en ciencia de datos y aprendizaje automático (Cass, 2023).

En el campo de la estadística, Python ofrece ventajas cruciales. Primero, su capacidad para manejar grandes volúmenes de datos mediante estructuras como DataFrame de la biblioteca pandas, permite una manipulación más eficiente que las hojas de cálculo tradicionales. Segundo, la facilidad para integrar análisis estadísticos con visualización gráfica (usando librerías como matplotlib y seaborn) ofrece una experiencia analítica fluida, desde la carga de los datos hasta la presentación de los resultados.

1.2. Ventajas frente a otros programas (SPSS, Excel, R)

A pesar de que herramientas como SPSS y Excel han sido ampliamente utilizadas para análisis estadístico en entornos educativos y corporativos, Python representa una evolución significativa en términos de automatización, escalabilidad y reproducibilidad. SPSS, por ejemplo, proporciona una interfaz amigable y procedimientos estadísticos estandarizados, pero presenta limitaciones cuando se requieren análisis complejos, personalizados o integraciones con sistemas externos. Excel, aunque ampliamente conocido, no está diseñado para gestionar grandes volúmenes de datos ni para aplicar técnicas estadísticas avanzadas con rigor computacional.

Por otro lado, R es también una excelente opción para estadística, especialmente en contextos académicos y de investigación. Sin embargo, Python ha ganado terreno en parte porque permite una integración más sencilla con aplicaciones web, sistemas empresariales y entornos de producción, lo cual lo convierte en una herramienta más versátil en contextos interdisciplinarios.

Además, Python permite una mayor trazabilidad del análisis mediante scripts que pueden documentarse, compartirse, y versionarse con facilidad utilizando sistemas como Git. Esta capacidad de reproducibilidad y colaboración es crítica en la ciencia moderna.

1.3. Breve historia y crecimiento de Python en ciencia de datos

El auge de Python en el análisis de datos y la estadística no ha sido casual, sino el resultado de una evolución coherente con las necesidades de la era digital. Con la aparición de numpy en 2006, seguido por pandas en 2008 y scikit-learn en 2010, Python empezó a consolidarse como un entorno completo para ciencia de datos. Estas bibliotecas permitieron a investigadores y profesionales aplicar técnicas estadísticas avanzadas sin necesidad de lenguajes compilados como C++ o Java, democratizando el acceso a herramientas analíticas sofisticadas.

En los últimos cinco años, Python ha sido adoptado masivamente en universidades, centros de investigación, gobiernos y empresas tecnológicas. Plataformas como Kaggle, GitHub y Google Colab han contribuido enormemente a su difusión, al permitir compartir código, realizar competiciones analíticas y ejecutar scripts en la nube sin necesidad de instalar software localmente.

Su comunidad activa, con miles de tutoriales, foros, cursos en línea y publicaciones científicas, ha convertido a Python en un lenguaje de aprendizaje accesible incluso para quienes no provienen de ciencias computacionales. Esto ha permitido que estudiantes de estadística, psicología, economía, biología y otras disciplinas puedan realizar análisis cuantitativos complejos sin depender exclusivamente de herramientas comerciales.

En conclusión, el uso de Python en estadística no solo responde a una moda tecnológica, sino a una necesidad estructural de la ciencia moderna: trabajar con datos masivos de manera reproducible, transparente, y eficiente. Este capítulo desarrollará paso a paso cómo Python puede ser utilizado para realizar análisis estadístico, integrando teoría, práctica y visualización, con ejemplos aplicados a conjuntos de datos reales.

2. DESARROLLO

2.1. Herramientas Fundamentales de Python para Estadística

La base del análisis estadístico moderno con Python no solo reside en el lenguaje en sí, sino en el ecosistema de herramientas que lo rodean. A través de entornos interactivos, bibliotecas científicas y estructuras de datos optimizadas, Python permite realizar análisis estadísticos robustos sin complicaciones innecesarias. Esta sección introduce las plataformas y bibliotecas esenciales que todo analista debe dominar para desarrollar proyectos estadísticos reproducibles, eficientes y transparentes.

2.1.1. Jupyter Notebooks y Google Colab: Plataformas para trabajar con Python

Jupyter Notebook es uno de los entornos más populares para ejecutar código Python de forma interactiva. Su principal ventaja radica en su estructura basada en celdas, que permite combinar texto explicativo (en formato Markdown), ecuaciones, y bloques de código ejecutable en un mismo documento. Esta característica facilita la creación de informes estadísticos dinámicos, donde los resultados del análisis pueden acompañarse con interpretaciones escritas, visualizaciones y fórmulas matemáticas.

Una alternativa basada en la nube es Google Colab, una plataforma gratuita que ofrece funcionalidades similares a Jupyter, con la ventaja adicional de no requerir instalación. Colab permite ejecutar scripts de Python directamente desde el navegador, guardar el trabajo en Google Drive y compartir proyectos colaborativamente. Es especialmente útil para estudiantes y equipos multidisciplinarios que requieren acceso simultáneo o no disponen de recursos computacionales locales potentes.

Tanto Jupyter como Colab se han convertido en herramientas estándar para el análisis estadístico, ya que permiten documentar cada paso del análisis y reproducirlo fácilmente.

2.1.2. Instalación de bibliotecas esenciales

Python, por defecto, no incluye funciones estadísticas avanzadas. Sin embargo, su ecosistema de bibliotecas científicas es extenso y muy bien documentado. A continuación, se presentan las principales librerías utilizadas en estadística y análisis de datos:

- *pandas*: biblioteca clave para la manipulación de datos. Introduce la estructura *DataFrame*, similar a una hoja de cálculo, ideal para almacenar y procesar grandes volúmenes de datos tabulados.
- *numpy*: proporciona herramientas para trabajar con arrays y funciones matemáticas de bajo nivel, utilizadas como base por muchas otras bibliotecas.
- *matplotlib y seaborn*: utilizadas para crear gráficos estáticos, dinámicos e interactivos. *seaborn* está construida sobre *matplotlib*, pero ofrece funciones más estilizadas y adaptadas al análisis estadístico.
- *scipy.stats*: módulo dentro de *SciPy* especializado en funciones estadísticas como distribuciones de probabilidad, pruebas de hipótesis y funciones de densidad.
- *statsmodels*: biblioteca para modelos estadísticos clásicos, como regresión lineal, ANOVA, y series temporales, con una interfaz similar a R.

Para instalar estas bibliotecas, se puede utilizar el gestor de paquetes *pip*, ejecutando en la terminal o en una celda de Jupyter:

```
!pip install pandas numpy matplotlib seaborn scipy statsmodels
```

En Google Colab, basta con colocar el signo de exclamación (!) antes del comando, dado que cada notebook se ejecuta en un contenedor temporal con permisos de shell.

2.1.3. Cómo importar datasets (.csv, .xlsx)

Una vez instaladas las bibliotecas, el siguiente paso es importar los datos. Python permite trabajar con múltiples formatos de archivo, siendo los más comunes los .csv (Comma-Separated Values) y .xlsx (archivos de Excel).

Para importar un archivo .csv utilizando pandas, se emplea el siguiente comando:

```
import pandas as pd

df = pd.read_csv('datos.csv')
```

En este código, df representa el DataFrame que almacenará la tabla importada. Si el archivo se encuentra en una URL o requiere codificación especial (por ejemplo, UTF-8), pandas permite especificar estos parámetros fácilmente:

```
df = pd.read_csv('https://example.com/datos.csv', encoding='utf-8')
```

Para archivos de Excel, se requiere además la biblioteca openpyxl, que permite la lectura de archivos .xlsx:

```
df = pd.read_excel('datos.xlsx')
```

Es importante asegurarse de que el archivo se encuentre en el mismo directorio del notebook o especificar la ruta completa. En entornos como Google Colab, los archivos pueden subirse directamente desde el disco local, utilizando:

```
from google.colab import files
uploaded = files.upload()
```

Tras la carga, el archivo puede leerse como si estuviera en el entorno local.
Estas funciones permiten que el usuario acceda a datasets externos, los explore, filtre, analice y visualice con facilidad. Esta flexibilidad, combinada con la potencia de las bibliotecas científicas, posiciona a Python como una herramienta ideal para la estadística aplicada.

2.2. Estadística Descriptiva con Python
La estadística descriptiva constituye el primer paso en todo análisis cuantitativo, pues permite comprender y resumir los aspectos esenciales de un conjunto de datos. A través de medidas de tendencia central, dispersión, y técnicas de visualización, se identifican patrones, anomalías y relaciones que guían el análisis posterior. Python, con sus bibliotecas especializadas, permite realizar estos procedimientos con eficiencia y claridad, facilitando su interpretación incluso para usuarios sin formación matemática avanzada.

2.2.1. Introducción conceptual: ¿Qué es la estadística descriptiva?
La estadística descriptiva se encarga de recolectar, organizar, presentar y describir un conjunto de datos mediante medidas numéricas y representaciones gráficas. A diferencia de la estadística inferencial, no busca generalizar resultados a partir de una muestra, sino describir lo que está ocurriendo dentro del conjunto analizado. En este sentido, es una herramienta de diagnóstico y exploración previa al modelado o prueba de hipótesis.
Las medidas más representativas incluyen:

- Media (promedio): suma de los valores dividida por la cantidad de observaciones.
- Mediana: valor central cuando los datos están ordenados.
- Moda: valor o valores que más se repiten.
- Varianza y desviación estándar: estimadores de la dispersión de los datos.
- Percentiles y cuartiles: indicadores de posición que dividen la muestra en partes proporcionales.

2.2.2. Exploración inicial de un conjunto de datos
Consideremos un ejemplo práctico utilizando un dataset hipotético sobre calificaciones de estudiantes en un curso universitario. Primero, importamos las bibliotecas necesarias y cargamos los datos en un DataFrame.

```
import pandas as pd
import numpy as np
```

```
# Crear dataset de ejemplo
data = {
    'Estudiante': ['Ana', 'Luis', 'Carlos', 'Marta', 'Sofía', 'Pedro', 'Lucía', 'Raúl', 'Diana',
'Jorge'],
    'Nota': [8.5, 7.2, 6.8, 9.1, 5.4, 7.9, 8.0, 6.5, 9.3, 7.0]
}
df = pd.DataFrame(data)
```

Cálculo de medidas de tendencia central

```
media = df['Nota'].mean()
mediana = df['Nota'].median()
moda = df['Nota'].mode()[0]

print(f"Media: {media:.2f}, Mediana: {mediana:.2f}, Moda: {moda:.2}")
```

La función `.mean()` calcula la media aritmética, `.median()` devuelve el valor central y `.mode()` la moda (que puede ser múltiple). El resultado muestra:

```
Media: 7.57, Mediana: 7.55, Moda: 5.40
```

Esto indica que la distribución es bastante simétrica, aunque la moda es poco representativa al tratarse de una única observación que se repite.

Medidas de dispersión

```
varianza = df['Nota'].var()
desviacion = df['Nota'].std()
rango = df['Nota'].max() - df['Nota'].min()

print(f"Varianza: {varianza:.2}, Desviación estándar: {desviacion:.2f}, Rango:
{rango:.2f}")
```

Estas métricas permiten conocer la dispersión de los datos alrededor del promedio. Una varianza alta sugiere mucha variabilidad, mientras que valores bajos indican consistencia.

Percentiles y cuartiles

```
q1 = df['Nota'].quantile(0.25)
q2 = df['Nota'].quantile(0.50)  # también es la mediana
q3 = df['Nota'].quantile(0.75)

print(f"Q1: {q1}, Q2 (mediana): {q2}, Q3: {q3}")
```

Estos valores permiten ver cómo se distribuyen los estudiantes en distintos tramos de calificación.

Visualización con matplotlib y seaborn

Las representaciones gráficas son esenciales en estadística descriptiva, ya que revelan patrones difíciles de ver en tablas. Python permite generar gráficos precisos y elegantes con poco código.

```
import matplotlib.pyplot as plt
import seaborn as sns

sns.set(style="whitegrid")
```

```
# Histograma
plt.figure(figsize=(8, 5))
sns.histplot(df['Nota'], bins=5, kde=True)
plt.title("Distribución de Notas")
plt.xlabel("Nota")
plt.ylabel("Frecuencia")
plt.show()
```

El histograma nos muestra la frecuencia de cada intervalo de notas. Al incluir la curva KDE (estimación de densidad), podemos visualizar cómo se agrupan los valores alrededor de la media.

```
Boxplot o diagrama de caja
plt.figure(figsize=(6, 4))
sns.boxplot(y=df['Nota'])
plt.title("Boxplot de Notas")
plt.show()
```

El diagrama de caja permite observar la distribución, asimetría, valores atípicos y los cuartiles de forma clara y rápida. Es una herramienta fundamental para detectar casos extremos que pueden sesgar el análisis.
Esta exploración permite a cualquier usuario —ya sea estudiante, docente o investigador— conocer las características básicas de un conjunto de datos antes de aplicar procedimientos más complejos. En próximos apartados se aplicarán técnicas de estadística inferencial, que permitirán ir más allá de la descripción, para realizar estimaciones, comparaciones y validaciones de hipótesis.

2.3. Estadística Inferencial con Python
La estadística inferencial constituye uno de los pilares fundamentales del análisis cuantitativo en las ciencias sociales, naturales y exactas. A diferencia de la estadística descriptiva, que se limita a resumir y visualizar datos observados, la inferencia estadística permite tomar decisiones y formular conclusiones válidas para una población entera a partir del estudio de una muestra. En este capítulo, nos enfocaremos en cómo Python facilita la ejecución e interpretación de pruebas estadísticas inferenciales, combinando precisión computacional con claridad analítica.

2.3.1. ¿Qué es la inferencia estadística?
La inferencia estadística es el conjunto de métodos que permite extraer conclusiones más allá de los datos disponibles, usualmente bajo incertidumbre. Se basa en el uso de modelos probabilísticos para estimar parámetros poblacionales (como la media o proporción) y probar hipótesis con base en la evidencia muestral. En términos simples, permite responder preguntas como:

- ¿La media de un grupo es significativamente diferente de cierto valor?
- ¿Dos grupos tienen promedios distintos o similares?
- ¿Hay una relación significativa entre dos variables?
- ¿Qué tan seguros podemos estar de nuestras estimaciones?

Conceptos clave

- Población: conjunto completo de elementos que se desea estudiar (por ejemplo, todos los estudiantes de una universidad).
- Muestra: subconjunto representativo de la población (por ejemplo, 100 estudiantes seleccionados al azar).
- Parámetro poblacional: valor verdadero, pero desconocido, que describe una característica de la población (por ejemplo, la media de todas las calificaciones).
- Estadístico muestral: valor calculado a partir de la muestra, utilizado como estimador del parámetro poblacional.

La inferencia parte del planteamiento de hipótesis, que se contrastan mediante métodos como la prueba t, el análisis de varianza (ANOVA), o pruebas no paramétricas. Se establece un

nivel de significancia (α), comúnmente 0.05, que indica la probabilidad de rechazar la hipótesis nula siendo esta verdadera (error tipo I).

2.3.2. Prueba t de Student: comparación de medias entre dos grupos

Fundamento

La prueba t se utiliza para comparar si la diferencia de medias entre dos grupos es estadísticamente significativa. Es adecuada cuando las variables son cuantitativas, los grupos son independientes y se asume distribución normal en los datos.

Ejemplo práctico

Situación: Un docente desea saber si hay diferencia significativa en el rendimiento promedio entre dos secciones de un curso.

```
from scipy import stats

# Notas de los estudiantes de cada sección
seccion_A = [8.5, 7.2, 6.8, 9.1, 5.4]
seccion_B = [7.9, 8.0, 6.5, 9.3, 7.0]

t_stat, p_val = stats.ttest_ind(seccion_A, seccion_B)

print(f"t = {t_stat:.3f}, p = {p_val:.3f}")
```

Si $p < 0.05$, se rechaza la hipótesis nula, concluyendo que existe una diferencia significativa entre ambas secciones. Si $p \geq 0.05$, no hay evidencia suficiente para afirmar que sus promedios difieren.

2.3.3. Prueba de Chi-cuadrado: asociación entre variables cualitativas

Fundamento

La prueba Chi-cuadrado de independencia evalúa si existe una relación significativa entre dos variables categóricas. Por ejemplo, si el género de un estudiante está relacionado con su resultado (aprobar/reprobar).

Ejemplo práctico

```
import numpy as np
from scipy.stats import chi2_contingency

# Matriz de frecuencia: [Aprobó, No aprobó]
#         Hombres Mujeres
tabla = np.array([[30, 10],
                  [25, 15]])

chi2, p, dof, esperado = chi2_contingency(tabla)

print(f"Chi2 = {chi2:.2f}, p = {p:.3f}")
```

Si $p < 0.05$, hay evidencia estadística de que género y aprobación están asociados. Es decir, el rendimiento podría depender del género, aunque se requeriría más análisis para entender la causa.

2.3.4. Intervalos de confianza: estimación de parámetros

Fundamento

Un intervalo de confianza (IC) es un rango de valores en el que, con una cierta probabilidad (por ejemplo, 95%), se espera que se encuentre el verdadero valor poblacional. Ofrece una alternativa a las pruebas de hipótesis, enfocándose en la estimación más que en la decisión binaria.

Ejemplo práctico

```python
import numpy as np
import scipy.stats as stats

notas = [8.5, 7.2, 6.8, 9.1, 5.4, 7.9, 8.0, 6.5, 9.3, 7.0]
n = len(notas)
media = np.mean(notas)
desv = np.std(notas, ddof=1)
confianza = 0.95

error = stats.t.ppf((1 + confianza) / 2., n-1) * (desv / np.sqrt(n))
intervalo = (media - error, media + error)

print(f"IC 95% para la media: {intervalo}")
```

Si el intervalo es (6.8, 8.2), significa que se tiene un 95% de confianza en que la media poblacional de las calificaciones está dentro de ese rango.

2.3.5. ANOVA: comparación de medias entre tres o más grupos
Fundamento
Cuando se desea comparar más de dos grupos, realizar múltiples pruebas t aumenta la probabilidad de error. El ANOVA (Análisis de Varianza) permite comprobar si al menos un grupo difiere significativamente, sin incrementar el error tipo I.

Ejemplo práctico

```python
grupo1 = [6.5, 7.0, 7.8]
grupo2 = [8.0, 7.9, 8.1]
grupo3 = [6.2, 6.9, 7.1]

f_stat, p_val = stats.f_oneway(grupo1, grupo2, grupo3)

print(f"F = {f_stat:.2f}, p = {p_val:.3f}")
```

Un valor de $p < 0.05$ indica que hay diferencias significativas entre las medias de al menos uno de los grupos. Si se detecta diferencia, se pueden hacer pruebas post-hoc (como Tukey HSD) para identificar cuál grupo difiere.
Si bien Python automatiza los cálculos estadísticos, la comprensión e interpretación de los resultados sigue siendo una responsabilidad humana. Una lectura acrítica de los valores p, o su uso para "validar" hipótesis sin contexto, puede llevar a conclusiones erróneas. Además, es esencial verificar que los datos cumplan con los supuestos de cada prueba (normalidad, homogeneidad de varianzas, independencia).
Bajo este contexto, toda inferencia es probabilística: nunca se puede "probar" una hipótesis en sentido absoluto, solo evaluar qué tan coherente es con los datos observados. Esta humildad epistemológica debe estar presente en todo análisis estadístico serio.

2.4. Regresión y Correlación con Python
La regresión y la correlación son técnicas esenciales en estadística que permiten analizar las relaciones entre variables. Mientras que la correlación evalúa el grado y dirección de asociación entre dos variables, la regresión permite modelar esa relación para realizar predicciones. En el contexto del análisis con Python, estas herramientas se implementan de forma precisa mediante bibliotecas como scipy, statsmodels y seaborn, facilitando la exploración y validación de modelos.

2.4.1. Fundamentos teóricos
Correlación

La correlación es una medida estadística que indica la intensidad y dirección de una relación lineal entre dos variables cuantitativas. El coeficiente de correlación más utilizado es el de Pearson, que toma valores entre -1 y 1:

- r = 1: correlación positiva perfecta
- r = -1: correlación negativa perfecta
- r = 0: ausencia de correlación lineal

No implica causalidad, pero sí proporciona una pista importante sobre posibles relaciones estructurales entre variables.
Otras medidas incluyen el coeficiente de Spearman (rango) y Kendall, adecuados para variables no normales o categóricas ordinales.

Regresión lineal
La regresión lineal busca modelar la relación entre una variable dependiente Y y una o más variables independientes X. En el caso más simple (regresión lineal simple), el modelo se expresa como:

$$Y = \beta_0 + \beta_1 X + \varepsilon$$

Donde:

- $\beta_0$es la ordenada al origen (intercepto)
- $\beta_1$es el coeficiente de regresión (pendiente)
- εes el error aleatorio

El modelo permite predecir valores de Y dados valores conocidos de X, siempre y cuando se cumplan los supuestos de linealidad, normalidad del error, homocedasticidad e independencia.

Ejemplo aplicado
Relación entre horas de estudio y calificación
Supongamos que un docente desea saber si existe una relación entre la cantidad de horas de estudio y la nota obtenida en un examen. Se recolecta una muestra de 10 estudiantes:

```
import pandas as pd

# Dataset
datos = {
   'Horas_estudio': [2, 3, 4, 5, 6, 6.5, 7, 8, 9, 10],
   'Nota': [5.5, 6.0, 6.8, 7.1, 7.4, 7.9, 8.2, 8.7, 9.1, 9.5]
}

df = pd.DataFrame(datos)
```

Correlación de Pearson

```
from scipy.stats import pearsonr

coef, p_val = pearsonr(df['Horas_estudio'], df['Nota'])
print(f"Coeficiente de correlación de Pearson: {coef:.2f}, p = {p_val:.3f}")
```

Un r cercano a 1 indicaría una fuerte correlación positiva: a más horas de estudio, mayor calificación. El valor de p indica si la relación es estadísticamente significativa.

Visualización con Seaborn

```
import seaborn as sns
import matplotlib.pyplot as plt

sns.regplot(x='Horas_estudio', y='Nota', data=df)
plt.title("Relación entre horas de estudio y nota")
plt.show()
```

Lo anterior incluye la línea de regresión ajustada y un intervalo de confianza del 95%, lo que permite observar la relación visualmente y detectar posibles valores atípicos.

Modelo de regresión lineal con statsmodels

```
import statsmodels.api as sm

X = df['Horas_estudio']
Y = df['Nota']
X = sm.add_constant(X)  # Agrega constante (intercepto)

modelo = sm.OLS(Y, X).fit()
print(modelo.summary())
```

El resumen del modelo ofrece información clave:

- Coeficientes (β_0 y β_1): muestran el impacto de las horas de estudio sobre la nota.
- R-cuadrado (R^2): indica qué proporción de la variabilidad en la nota se explica por las horas de estudio. Un R^2 cercano a 1 implica un modelo muy explicativo.
- Valor p de los coeficientes: indica si la relación entre las variables es significativa.

Predicción

Una vez ajustado el modelo, se puede predecir la calificación esperada para un estudiante que estudie, por ejemplo, 7.5 horas:

```
nuevos_valores = pd.DataFrame({'const': 1, 'Horas_estudio': [7.5]})
prediccion = modelo.predict(nuevos_valores)
print(f"Nota esperada para 7.5 horas de estudio: {prediccion[0]:.2f}")
```

Consideraciones importantes

- Linealidad: verificar si la relación entre las variables es lineal; en caso contrario, se deben usar modelos no lineales.
- Valores atípicos: pueden distorsionar los coeficientes. Es recomendable revisar boxplots o los residuos.
- Multicolinealidad: si hay varias variables independientes, deben ser poco correlacionadas entre sí para evitar interferencia en la estimación.
- Interpretación contextual: incluso si el modelo es estadísticamente significativo, hay que preguntarse si es significativamente útil o relevante.

La correlación y la regresión son herramientas fundamentales que permiten no solo detectar relaciones entre variables, sino también predecir comportamientos futuros. Gracias a la capacidad de Python para modelar, visualizar e interpretar datos, estas técnicas se vuelven accesibles a estudiantes y profesionales por igual.

En el siguiente apartado, aplicaremos estas herramientas en un caso integrador, utilizando un dataset real completo, que permitirá conectar los conceptos vistos hasta ahora en un ejercicio práctico más complejo y realista.

2.6. Estudio de Caso Integrador

Para consolidar el aprendizaje teórico y práctico de los apartados anteriores, desarrollaremos un caso de estudio con datos simulados que representan un contexto educativo. El objetivo es mostrar cómo aplicar el flujo completo de un análisis estadístico en Python: desde la carga del dataset, exploración inicial, resumen descriptivo, pruebas inferenciales, hasta modelos de predicción. Este enfoque refleja situaciones reales en las que se requiere extraer conclusiones basadas en evidencia numérica.

2.6.1. Contexto del caso

Una universidad desea analizar el rendimiento académico de sus estudiantes en un curso de Estadística. Se recolectan datos de 100 estudiantes, incluyendo:

- Horas_estudio: cantidad de horas semanales dedicadas al estudio individual.
- Asistencia: porcentaje de clases asistidas.
- Nota_final: calificación obtenida al final del curso (escala 0-10).
- Género: masculino o femenino.
- Aprobo: 1 si aprobó (nota ≥ 6), 0 si no.

a) Carga y exploración de los datos

```
import pandas as pd
import numpy as np

# Simulación del dataset
np.random.seed(42)
n = 100

df = pd.DataFrame({
    'Horas_estudio': np.random.normal(7, 2, n).round(1),
    'Asistencia': np.random.normal(85, 10, n).clip(50, 100).round(1),
    'Nota_final': np.random.normal(7, 1.2, n).clip(3, 10).round(1),
    'Genero': np.random.choice(['Masculino', 'Femenino'], size=n)
})

# Crear variable binaria de aprobación
df['Aprobo'] = np.where(df['Nota_final'] >= 6, 1, 0)

# Primeras filas
print(df.head())
```

b) Estadística descriptiva

```
# Descripción general
print(df.describe())

# Distribución por género
print(df['Genero'].value_counts())

# Nota media por género
print(df.groupby('Genero')['Nota_final'].mean())
```

Visualización

```
import seaborn as sns
import matplotlib.pyplot as plt

# Boxplot por género
sns.boxplot(x='Genero', y='Nota_final', data=df)
plt.title("Distribución de Notas por Género")
plt.show()
```

Este resumen muestra diferencias en las calificaciones promedio entre géneros, y permite identificar posibles asimetrías o casos atípicos.

c) Inferencia estadística

¿Existen diferencias de nota por género?

```
from scipy.stats import ttest_ind

masc = df[df['Genero'] == 'Masculino']['Nota_final']
fem = df[df['Genero'] == 'Femenino']['Nota_final']

t, p = ttest_ind(masc, fem)
print(f"t = {t:.2f}, p = {p:.4f}")
```

Si $p < 0.05$, se concluye que existe una diferencia significativa en las notas entre hombres y mujeres.

¿Hay asociación entre género y aprobación?

```
from scipy.stats import chi2_contingency

tabla = pd.crosstab(df['Genero'], df['Aprobo'])
chi2, p, dof, exp = chi2_contingency(tabla)

print(f"Chi2 = {chi2:.2f}, p = {p:.4f}")
```

Este análisis permite evaluar si la aprobación está asociada con el género.

d) Correlación entre horas de estudio, asistencia y nota

```
# Correlación
print(df[['Horas_estudio', 'Asistencia', 'Nota_final']].corr())
```

Visualización

```
sns.pairplot(df[['Horas_estudio', 'Asistencia', 'Nota_final']])
plt.show()
```

Las correlaciones positivas indicarían que más horas de estudio y mayor asistencia se relacionan con mejores calificaciones.

e) Modelo de regresión lineal

Queremos predecir la Nota_final a partir de Horas_estudio y Asistencia.

```
import statsmodels.api as sm

X = df[['Horas_estudio', 'Asistencia']]
X = sm.add_constant(X)
y = df['Nota_final']

modelo = sm.OLS(y, X).fit()
print(modelo.summary())
```

- Coeficientes significativos ($p < 0.05$) indican variables predictoras válidas.
- R^2 alto sugiere que el modelo explica bien la variabilidad de las notas.
- El intercepto indica la nota esperada si horas de estudio y asistencia fueran cero (valor teórico sin sentido práctico, pero importante estadísticamente).

f) Predicción de casos nuevos

```
# Predicción para nuevo estudiante
nuevo = pd.DataFrame({'const': 1, 'Horas_estudio': [8], 'Asistencia': [90]})
prediccion = modelo.predict(nuevo)
print(f"Nota esperada: {prediccion[0]:.2f}")
```

Este caso integrador permite observar cómo Python puede ser utilizado para realizar un análisis estadístico completo, desde la exploración inicial hasta la modelación predictiva. Las herramientas vistas permiten no solo describir los datos, sino inferir relaciones, evaluar asociaciones, y construir modelos útiles para la toma de decisiones informadas. De manera conjunta, permite ejercitar buenas prácticas como la validación de supuestos, la interpretación crítica y el uso responsable de los resultados estadísticos.

CONCLUSIONS

El análisis estadístico con Python no es solamente una cuestión técnica, sino una forma estructurada de pensamiento que transforma datos en decisiones. A través de este capítulo se comprendió que dominar Python en estadística no se reduce al aprendizaje de funciones o sintaxis, sino a incorporar una forma de análisis sistemático, reproducible y riguroso, donde la interpretación de los resultados tiene tanto peso como los procedimientos utilizados para obtenerlos.

El uso de Python para estadística representa más que una alternativa a software tradicionales: es una reconfiguración del rol del analista, que deja de ser un consumidor de herramientas y pasa a ser un constructor activo de soluciones. Comprendimos que el valor del análisis con Python no radica solo en automatizar cálculos, sino en posibilitar la trazabilidad, transparencia y personalización de los procesos analíticos, facilitando una lectura profunda de los fenómenos.

Este capítulo aporta una propuesta didáctica y aplicada para integrar la programación en Python al estudio de la estadística en el contexto académico latinoamericano. A diferencia de manuales técnicos o libros de texto descontextualizados, se plantea aquí una ruta progresiva, que une teoría y práctica, con ejemplos orientados a problemas reales y educativos. Este enfoque democratiza el acceso a la ciencia de datos, conectando con quienes tienen poca o nula experiencia en programación, pero buscan herramientas para interpretar la realidad con base empírica.

Los ejemplos desarrollados y las bibliotecas utilizadas son altamente aplicables a contextos educativos presenciales y datos estructurados. Sin embargo, su validez puede disminuir en contextos con alta heterogeneidad en calidad de datos, problemas de desbalance o en escenarios donde se requieren modelos estadísticos avanzados o de naturaleza no paramétrica. Las técnicas descritas funcionan óptimamente cuando los datos cumplen supuestos clásicos, por lo que la evaluación crítica de esos supuestos sigue siendo esencial.

Este capítulo sugiere que la enseñanza de estadística en entornos universitarios debe actualizarse y alinearse con herramientas abiertas, programables y escalables como Python. Esto implica repensar el currículo: pasar de un enfoque centrado en fórmulas a uno centrado en preguntas analíticas y capacidades computacionales. La estadística, cuando se enseña junto con herramientas como Jupyter o Colab, deja de ser abstracta y se convierte en una herramienta de acción empírica, ética y crítica.

Aunque se mostró cómo Python permite aplicar técnicas estadísticas con profundidad, queda pendiente un análisis más extenso de modelos multivariados, no lineales y de aprendizaje automático, cuya integración representa el siguiente paso natural para los lectores que dominen lo aquí planteado. También se abre la posibilidad de aplicar estos aprendizajes a campos específicos como salud, educación, economía o ciencias sociales, desarrollando estudios contextualizados y con impacto real.

Python no es solo un lenguaje de programación: es una puerta de entrada a una cultura analítica, transparente y basada en evidencia. Enseñar y practicar estadística con Python es, en última instancia, una decisión epistemológica: elegir comprender el mundo no por intuición, sino por datos, pensamiento estructurado y reflexión crítica.

REFERENCES

Cass S. (2023). The Top Programming Languages 2024. *IEEE Spectrum*. https://spectrum.ieee.org/it-management-software-failures

Chen, H., Chiang, R. H. L., & Storey, V. C. (2014). Business Intelligence and Analytics: From Big Data to Big Impact. *MIS Quarterly, 38*(3), 1165-1188. https://doi.org/10.2307/41703503

Cuervo Díaz, L. M., Díaz, N. A. C., & Álvarez, J. C. (2024). *Iniciando a Programar con Python.: Guía básica de programación* (Vol. 87). Editorial de la Universidad Pedagógica y Tecnológica de Colombia-UPTC.

Mayer-Schönberger, V. (2013). *Big data: A revolution that will transform how we live, work, and think*. Houghton Mifflin Harcourt.

Podhoranyi, M. (2021). A comprehensive social media data processing and analytics architecture by using big data platforms: a case study of twitter flood-risk messages. *Earth Science Informatics*, *14*(2), 913-929. https://doi.org/10.1007/s12145-021-00601-w

Sandhu, A. K. (2021). Big data with cloud computing: Discussions and challenges. *Big Data Mining and Analytics*, *5*(1), 32-40. https://doi.org/10.26599/BDMA.2021.9020016

Wu, X., Zhu, X., Wu, G. Q., & Ding, W. (2013). Data mining with big data. *IEEE transactions on knowledge and data engineering*, *26*(1), 97-107. https://doi.org/10.1109/TKDE.2013.109

You, X., Wang, C. X., Huang, J., Gao, X., Zhang, Z., Wang, M., ... & Liang, Y. C. (2021). Towards 6G wireless communication networks: Vision, enabling technologies, and new paradigm shifts. *Science China information sciences*, *64*(1), 110301. https://doi.org/10.1007/s11432-020-2955-6