Scikit-Stats: Biblioteca Estadística para Python

Documentación Técnica y Ejemplos Prácticos

Juan Diego Avila Tafur

Curso: Probabilidad y Estadística

Universidad: Universidad Distrital

18 de septiembre de 2025

Índice

1 Introducción a Scikit-Stats

1.1 ¿Qué es Scikit-Stats?

Scikit-stats es una biblioteca de Python especializada en análisis estadístico y cálculo de probabilidades. Forma parte del ecosistema scikit y está diseñada específicamente para aplicaciones educativas y de investigación.

1.2 Historia y Creadores

- **Creador**: Brian Sipos
- **Primera versión**: 2020
- **Versión actual**: 0.1.3
- **Licencia**: BSD 3-Clause

1.3 Objetivos Principales

- Proporcionar funciones estadísticas sencillas de usar
- Facilitar el aprendizaje de probabilidad y estadística
- Ofrecer una alternativa más simple a SciPy.stats
- Ser accesible para estudiantes e investigadores

2 Instalación y Configuración

2.1 Requisitos del Sistema

- Python 3.6 o superior
- pip (gestor de paquetes de Python)

2.2 Instalación

```
# Instalaci n desde PyPI
pip install scikit-stats

# Verificaci n de la instalaci n
python -c "import scikit_stats; print('Scikit-stats instalado
    correctamente')"
```

2.3 Importación en Python

```
# Importaci n est ndar
from scikit_stats import dists
import scikit_stats as sks
```

3 Funciones y Módulos Principales

3.1 Módulo dists (Distribuciones)

Este módulo contiene todas las distribuciones de probabilidad.

3.1.1. Distribuciones Disponibles

- `norm` - Distribución Normal
- `poisson` - Distribución de Poisson
- `binom` - Distribución Binomial
- `multivariate_hypergeom` - Hipergeométrica Multivariada
- `expon` - Distribución Exponencial
- `uniform` - Distribución Uniforme

3.1.2. Funciones Comunes por Distribución

- `.pmf()` - Función de Masa de Probabilidad
- `.pdf()` - Función de Densidad de Probabilidad
- `.cdf()` - Función de Distribución Acumulada
- `.ppf()` - Función de Percentil
- `.sf()` - Función de Supervivencia
- `.rvs()` - Generador de Variables Aleatorias

3.2 Módulo tests (Pruebas Estadísticas)

- `t_test()` - Prueba t de Student
- `chi2_test()` - Prueba Chi-cuadrado
- `anova()` - Análisis de Varianza

3.3 Módulo intervals (Intervalos)

- `mean_ci()` - Intervalo de confianza para la media
- `proportion_ci()` - Intervalo para proporciones

4 Ejemplos Prácticos de Probabilidad

4.1 Ejemplo 1: Distribución Normal

Cálculo de probabilidades usando distribución normal estándar.

```
from scikit_stats import dists

# Probabilidad acumulada
prob = dists.norm.cdf(1.96)
print(f"P(Z < 1.96) = {prob:.4f}")

# Funci n de densidad
densidad = dists.norm.pdf(0)
print(f"f(0) = {densidad:.4f}")

# Percentil 95%
percentil = dists.norm.ppf(0.95)
print(f"Percentil 95% = {percentil:.4f}")
```

4.2 Ejemplo 2: Distribución de Poisson

Cálculo de probabilidades para eventos raros.

```
from scikit_stats import dists

# Probabilidad de exactamente 3 eventos
prob_exacta = dists.poisson.pmf(3, mu=2.5)
print(f"P(X=3) = {prob_exacta:.4f}")

# Probabilidad acumulada ( 2  eventos)
prob_acum = dists.poisson.cdf(2, mu=2.5)
print(f"P( X 2 ) = {prob_acum:.4f}")

# Probabilidad de m s de 4 eventos
prob_cola = dists.poisson.sf(4, mu=2.5)
print(f"P(X>4) = {prob_cola:.4f}")
```

4.3 Ejemplo 3: Distribución Hipergeométrica Multivariada

Ejercicio de selección de estudiantes.

```
from scikit_stats import dists

# Datos: 3 Sistemas, 8 Electr nica, 9 Industrial
categorias = [3, 8, 9]

# a) Probabilidad de 3 Electr nica
prob_3_electronica = dists.multivariate_hypergeom.pmf(
    x=[0, 3, 0],
    n=categorias,
```

```
    size=3
)
print(f"P(3 Electr nica) = {prob_3_electronica:.6f}")

# b) Probabilidad de uno de cada especialidad
prob_uno_cada = dists.multivariate_hypergeom.pmf(
    x=[1, 1, 1],
    n=categorias,
    size=3
)
print(f"P(uno de cada) = {prob_uno_cada:.6f}")
```

5 Aplicaciones en Probabilidad y Estadística

5.1 Variables Aleatorias Discretas

Scikit-stats es ideal para trabajar con:

- Distribución Binomial
- Distribución de Poisson
- Distribución Hipergeométrica
- Distribución Geométrica

5.2 Variables Aleatorias Continuas

- Distribución Normal
- Distribución Exponencial
- Distribución Uniforme
- Distribución t de Student

5.3 Combinatoria y Probabilidad

La biblioteca facilita el cálculo de:

- Probabilidades exactas
- Probabilidades acumuladas de confianza
- Simulaciones de Monte Carlo

6 Ventajas y Desventajas

6.1 Ventajas

- Sintaxis intuitiva y sencilla
- Documentación clara y ejemplos prácticos
- Especializada en aplicaciones educativas
- No requiere conocimientos avanzados de programación
- Compatible con el ecosistema científico de Python

6.2 Desventajas

- Menos funciones que SciPy.stats
- Comunidad más pequeña
- Menos opciones para análisis avanzado

7 Conclusión

Scikit-stats es una excelente biblioteca para estudiantes de probabilidad y estadística. Su sintaxis clara y funciones bien documentadas la hacen ideal para el aprendizaje de conceptos estadísticos fundamentales. Aunque no tiene todas las funcionalidades de bibliotecas más complejas como SciPy, su enfoque educativo la convierte en la herramienta perfecta para cursos introductorios.

8 Referencias y Recursos

8.1 Enlaces Importantes

- **Documentación oficial**: `https://pypi.org/project/scikit-stats/`
- **Código fuente**: `https://github.com/bsipos/scikit-stats`
- **Ejemplos adicionales**: `https://scikit-stats.readthedocs.io/`

8.2 Bibliografía Recomendada

- Ross, S. M. (2019). Ïntroduction to Probability and Statistics"
- Montgomery, D. C. (2019). .ªpplied Statistics and Probability for Engineers"
- Devore, J. L. (2015). "Probability and Statistics for Engineering and the Sciences"

A Apéndice: Códigos Completos

A.1 Código del Ejercicio de Hipergeométrica

```python
# ejercicio_hipergeometrica.py
"""
Ejercicio completo de distribuci n hipergeom trica multivariada
"""
from scikit_stats import dists

def main():
    print("=== EJERCICIO HIPERGEOM TRICA MULTIVARIADA ===")

    # Datos del problema
    sistemas = 3
    electronica = 8
    industrial = 9
    total = sistemas + electronica + industrial

    categorias = [sistemas, electronica, industrial]
    nombres = ["Sistemas", "Electr nica", "Industrial"]

    print(f"Total estudiantes: {total}")
    print(f"Distribuci n: {sistemas} Sistemas, {electronica}
       Electr nica, {industrial} Industrial\n")

    # a) Probabilidad de 3 Electr nica
    prob_a = dists.multivariate_hypergeom.pmf(
        x=[0, 3, 0],
        n=categorias,
        size=3
    )
    print(f"a) P(3 Electr nica) = {prob_a:.6f}")
    print(f"   ({prob_a*100:.2f}%)\n")

    # b) Probabilidad de uno de cada uno
    prob_b = dists.multivariate_hypergeom.pmf(
        x=[1, 1, 1],
        n=categorias,
        size=3
    )
    print(f"b) P(uno de cada especialidad) = {prob_b:.6f}")
    print(f"   ({prob_b*100:.2f}%)\n")

    # c) Probabilidad de al menos uno de industrial
    # Para 3 selecciones, calcular 1 - P(ning n industrial)
    prob_ningun_industrial = dists.multivariate_hypergeom.pmf(
        x=[1, 2, 0],  # Combinaciones sin industrial
        n=categorias,
        size=3
    )
```

```
    prob_c = 1 - prob_ningun_industrial
    print(f"c) P(al menos 1 Industrial) = {prob_c:.6f}")
    print(f"   ({prob_c*100:.2f}%)\n")

if __name__ == "__main__":
    main()
```