Introducción a la Ciencia de Datos con Python

Acurio Solar Manuel

Escuela de Ingenieria Física
Universidad Nacional de Ingeniería
Lima, Perú
manuel.acurio.s@uni.pe

Ahon Malca Daniel Guillermo

Escuela de Ingenieria Física
Universidad Nacional de Ingeniería
Lima, Perú
daniel.ahon.m@uni.pe

Estrada Lopez Marco Josbel

Escuela de Física
Universidad Nacional de Ingeniería
Lima, Perú
marco.estrada.l@uni.pe

Mozo Perez Willian Alessandro

Escuela de Física
Universidad Nacional de Ingeniería
Lima, Perú
willian.mozo.p@uni.pe

***Resumen*—Este proyecto tiene como objetivo realizar un análisis exhaustivo de datos utilizando bibliotecas de Python como pandas, numpy y matplotlib. El análisis incluyó la carga y limpieza de datos, verificando y manejando valores nulos, ajustando tipos de datos y calculando estadísticas descriptivas para comprender la estructura de los datos. Posteriormente, se llevó a cabo un análisis exploratorio de datos (EDA) mediante visualizaciones como histogramas, boxplots y pairplots para explorar distribuciones y relaciones entre características, así como la creación de una matriz de correlación para identificar correlaciones entre variables. Los resultados se visualizaron mediante gráficos de dispersión y mapas de calor, proporcionando una representación clara de los hallazgos clave. Todo el proceso fue documentado adecuadamente para asegurar su reproducibilidad, con comentarios en el código y recomendaciones para el uso de herramientas como Jupyter Notebooks y sistemas de control de versiones. Este proyecto demuestra el uso de técnicas fundamentales de análisis de datos para explorar y visualizar patrones y relaciones en un conjunto de datos, proporcionando una base sólida para futuros análisis y desarrollos en el campo del análisis de datos.**

I. Introducción

Python es un lenguaje de programación interpretado, de alto nivel y de propósito general que ha ganado una popularidad significativa en diversos campos, incluyendo la ciencia de datos, el desarrollo web, la automatización y la inteligencia artificial. Desarrollado por Guido van Rossum y lanzado por primera vez en 1991, Python se caracteriza por su sintaxis sencilla y legible, lo que facilita el aprendizaje y la implementación rápida de soluciones. Entre las numerosas bibliotecas disponibles para Python, tres se destacan por su importancia en el análisis de datos y la visualización: Pandas, NumPy y Matplotlib. Estas librerías forman el núcleo de muchas aplicaciones de ciencia de datos y análisis numérico, proporcionando herramientas poderosas y flexibles para manejar, analizar y visualizar datos. En conjunto, estas librerías permiten a los desarrolladores y científicos de datos llevar a cabo un flujo de trabajo completo que abarca desde la manipulación y el análisis de datos hasta la visualización y la presentación de resultados. La sinergia entre Pandas, NumPy y Matplotlib hace que Python sea una de las opciones más potentes y versátiles para la ciencia de datos y el análisis numérico.

Figura 1. Imagen ilustrativa del analisis de datos en Python

II. Marco Teórico

II-A. Pandas

Pandas es una biblioteca de código abierto en Python diseñada para facilitar la manipulación y el análisis de datos estructurados. A continuación, se describen sus características principales y conceptos fundamentales:

Estructuras de Datos Principales:

- **Series:**
  - Una Serie es un objeto unidimensional etiquetado, similar a un array o una lista en Python.
  - Puede contener cualquier tipo de datos y se puede crear a partir de listas, arrays de NumPy, diccionarios, etc.
- **DataFrame:**
  - Un DataFrame es una estructura de datos bidimensional similar a una tabla de base de datos o una hoja de cálculo de Excel.
  - Contiene filas y columnas etiquetadas, permitiendo el acceso flexible a los datos.
  - Es ideal para la manipulación y análisis de datos tabulares en Python.

Funcionalidades Clave:

- **Carga y Almacenamiento de Datos:** Pandas puede leer datos desde varios formatos como CSV, Excel, bases de datos SQL, JSON, HTML, entre otros, y escribir datos en estos formatos.
- **Indexación y Selección:** Permite seleccionar y manipular datos usando etiquetas de índice (por ejemplo, nombres de columnas) o índices numéricos.
- **Operaciones sobre Datos:** Soporta operaciones como agrupamiento, filtrado, agregación (sumas, promedios, etc.) y combinaciones de datos.
- **Manejo de Datos Faltantes:** Proporciona herramientas para detectar y manejar valores nulos o datos faltantes eficientemente.
- **Operaciones de Series Temporales:** Ofrece funcionalidades específicas para trabajar con datos de series temporales, incluyendo manejo de fechas y tiempos, remuestreo y desplazamiento.
- **Visualización de Datos:** Integración con bibliotecas como Matplotlib y Seaborn para crear gráficos y visualizaciones desde los datos en un DataFrame.

Uso Común: Pandas es ampliamente utilizado en campos como análisis financiero, investigación académica, ciencia de datos, análisis de redes sociales, bioinformática, entre otros. Su flexibilidad y potencia lo convierten en una herramienta fundamental para cualquier persona que trabaje con datos en Python.

Conclusión: Pandas simplifica la manipulación de datos, facilita el análisis exploratorio y proporciona una estructura intuitiva para trabajar con datos tabulares en Python. Esencial para proyectos que involucren grandes volúmenes de datos estructurados.

II-B. NumPy

NumPy es una biblioteca fundamental para la computación científica en Python. A continuación, se describen sus características principales y conceptos fundamentales:

Arrays de NumPy:

- NumPy proporciona el objeto **array**, que es una estructura de datos multidimensional eficiente para almacenar y manipular datos.
- Los arrays de NumPy son más eficientes que las listas de Python para operaciones numéricas y matemáticas.
- Pueden contener elementos de un solo tipo de datos, lo que permite operaciones rápidas y optimizadas.
- Se pueden crear desde listas de Python, otros arrays de NumPy, o utilizando funciones específicas de creación de arrays.

Operaciones y Funcionalidades Clave:

- **Operaciones Matemáticas:** NumPy proporciona funciones para realizar operaciones matemáticas eficientes en arrays, como sumas, productos, operaciones trigonométricas, álgebra lineal, entre otros.
- **Indexación y Selección:** Permite acceder y manipular datos en arrays utilizando índices y rebanadas, de manera similar a las listas de Python.
- **Broadcasting:** Mecanismo que permite a NumPy trabajar de manera eficiente con arrays de diferentes tamaños durante operaciones aritméticas.
- **Operaciones de Álgebra Lineal:** NumPy incluye funciones para álgebra lineal, como la inversión de matrices, la resolución de sistemas de ecuaciones lineales y la descomposición de matrices.
- **Integración con otras Bibliotecas:** Es la base de muchas otras bibliotecas científicas en Python, como Pandas, SciPy y Scikit-Learn, facilitando el intercambio de datos y operaciones entre diferentes herramientas.

Uso Común: NumPy es ampliamente utilizado en diversas áreas de la ciencia de datos, investigación académica, análisis numérico, procesamiento de señales, simulaciones, entre otros. Su eficiencia y capacidad para trabajar con grandes volúmenes de datos numéricos lo convierten en una herramienta esencial para cualquier proyecto que requiera computación científica en Python.

Conclusión: NumPy proporciona las herramientas fundamentales para realizar operaciones numéricas eficientes y manipular grandes conjuntos de datos en Python. Esencial para proyectos que involucren análisis, modelado matemático o simulaciones numéricas.

II-C. Matplotlib

Matplotlib es una biblioteca de visualización de datos en Python. A continuación, se describen sus características principales y conceptos fundamentales:

Tipos de Gráficos:

- Matplotlib permite crear una amplia variedad de gráficos estáticos, incluyendo:
  - Gráficos de líneas y de dispersión.
  - Histogramas y gráficos de barras.
  - Gráficos de contorno y de superficie.
  - Gráficos de cajas y bigotes.
  - Gráficos de torta y gráficos de violín, entre otros.
- Cada tipo de gráfico puede personalizarse completamente, incluyendo etiquetas, colores, estilos de línea, leyendas y más.

Interfaz Pyplot:

- Matplotlib proporciona una interfaz estilo MATLAB llamada **Pyplot**, que facilita la creación rápida de gráficos mediante funciones simples como `plot()`, `scatter()`, `hist()`, etc.
- Pyplot permite controlar varios aspectos del gráfico interactuando con objetos de figuras y ejes.

Figuras y Ejes:

- Una **figura** en Matplotlib es el contenedor de nivel superior que contiene todos los elementos del gráfico.
- Los **ejes** son los objetos que contienen los límites de los datos, los marcadores de ejes, las etiquetas, etc.
- Matplotlib permite crear figuras y ejes de manera explícita o implícita según sea necesario.

Personalización Avanzada:

- Matplotlib ofrece un alto grado de personalización a través de métodos orientados a objetos y funciones de estilo Pyplot.
- Permite agregar textos, anotaciones, flechas, gradientes, mapas de colores y otras decoraciones a los gráficos.

Integración con NumPy y Pandas:

- Matplotlib se integra perfectamente con NumPy y Pandas, permitiendo visualizar datos almacenados en arrays de NumPy o DataFrames de Pandas.
- Es una herramienta clave en el ecosistema de Python para la visualización de datos en análisis exploratorio, informes científicos, y presentaciones visuales.

Uso Común: Matplotlib es ampliamente utilizado en campos como ciencia de datos, ingeniería, investigación académica, finanzas, entre otros. Su versatilidad y capacidad para generar gráficos de alta calidad lo convierten en una opción preferida para la visualización de datos en Python.

Conclusión: Matplotlib proporciona las herramientas esenciales para la visualización de datos en Python, permitiendo crear gráficos personalizados y efectivos para la comunicación visual de resultados analíticos y científicos.

III. Estado del Arte

El lenguaje de programación Python se ha consolidado como una herramienta fundamental en el ámbito de la ciencia de datos, el análisis numérico y la visualización de datos. Entre las librerías más destacadas para estas tareas se encuentran Pandas, NumPy y Matplotlib.

III-A. Pandas

Pandas es una librería de Python que proporciona estructuras de datos y herramientas de análisis de datos de alto rendimiento y fáciles de usar. Fue desarrollada inicialmente por Wes McKinney en 2008 y ha evolucionado significativamente desde entonces. La estructura de datos principal de Pandas es el *DataFrame*, que se asemeja a una hoja de cálculo y permite operaciones eficientes sobre datos tabulares.

Pandas ha sido ampliamente adoptada en la industria y la academia debido a su capacidad para manejar grandes volúmenes de datos y realizar operaciones complejas de manipulación y limpieza de datos. Las principales características de Pandas incluyen:

- Eficiencia en la manipulación de datos con estructuras como *Series* y *DataFrames*.
- Integración con otras librerías de Python como NumPy, Matplotlib y SciPy.
- Soporte para operaciones de entrada/salida con diferentes formatos de datos, incluyendo CSV, Excel, SQL y HDF5.
- Potentes capacidades de agrupamiento, filtrado y transformación de datos.

III-B. NumPy

NumPy (Numerical Python) es la librería fundamental para la computación numérica en Python. Fue creada en 2005 por Travis Oliphant como una extensión del paquete Numeric. NumPy proporciona soporte para arreglos multidimensionales (*ndarray*) y una amplia colección de funciones matemáticas de alto nivel para operar con estos arreglos.

Las principales características de NumPy incluyen:

- Arreglos multidimensionales de elementos homogéneos.
- Operaciones matemáticas y lógicas sobre arreglos.
- Transformaciones de forma de los arreglos sin copiar los datos.
- Funciones matemáticas y estadísticas integradas.
- Soporte para álgebra lineal, generación de números aleatorios y transformadas de Fourier.

NumPy es esencial en la mayoría de las aplicaciones científicas y de ingeniería, ya que proporciona una base eficiente sobre la cual se construyen otras librerías de alto nivel como Pandas y SciPy.

III-C. Matplotlib

Matplotlib es una librería de Python para la generación de gráficos estáticos, animados e interactivos. Fue creada por John D. Hunter en 2003 y ha sido fundamental para la visualización de datos en Python. Matplotlib ofrece una amplia variedad de tipos de gráficos, incluyendo gráficos de líneas, barras, histogramas, dispersión y más.

Las principales características de Matplotlib incluyen:

- Generación de gráficos de alta calidad con un control detallado sobre todos los aspectos de los mismos.
- Integración con NumPy para manejar datos en forma de arreglos.
- Interfaz orientada a objetos que permite la creación de figuras y ejes de forma explícita.
- Capacidades de personalización extensivas para adaptar los gráficos a las necesidades específicas del usuario.
- Soporte para diversas salidas gráficas, incluyendo PNG, PDF, SVG y EPS.

Matplotlib se utiliza ampliamente tanto en entornos académicos como industriales para la visualización de datos debido a su flexibilidad y la calidad de los gráficos que produce.

III-D. Interacción y Uso Combinado

La combinación de Pandas, NumPy y Matplotlib constituye una poderosa herramienta para el análisis y visualización de datos en Python. Pandas se utiliza generalmente para la manipulación y el análisis de datos, NumPy proporciona las capacidades de computación numérica subyacentes, y Matplotlib se utiliza para la visualización de los resultados. Este trío de librerías permite a los usuarios realizar un flujo de trabajo completo de análisis de datos, desde la limpieza y manipulación de datos hasta la visualización y comunicación de resultados.

En resumen, Pandas, NumPy y Matplotlib son componentes esenciales del ecosistema de Python para el análisis de datos y la visualización, y su uso combinado proporciona una solución integral para la ciencia de datos y otras aplicaciones analíticas.

IV. Metodología

En esta sección se describe la metodología empleada para desarrollar el proyecto de programación en Python utilizando las librerías Pandas, NumPy y Matplotlib. La metodología se divide en varias etapas clave, que incluyen la recopilación de datos, la limpieza y preprocesamiento de los datos, el análisis exploratorio y la visualización de resultados.

IV-A. Recopilación de Datos

La primera etapa del proyecto consiste en la recopilación de datos relevantes para el problema que se desea resolver. En este trabajo se empleara una base de datos previamente recopilado proporcionado por el docente

- iris.csv
- mtcars.csv
- mtcarsnuevo.csv

IV-B. Limpieza y Preprocesamiento de Datos

Una vez recopilados los datos, se procede a su limpieza y preprocesamiento utilizando las capacidades de Pandas y NumPy. Las principales tareas en esta etapa incluyen:

- **Eliminación de valores faltantes:** Identificación y manejo de valores nulos o faltantes en el conjunto de datos.
- **Conversión de tipos de datos:** Asegurar que los datos estén en los formatos correctos para su análisis.
- **Normalización y escalado:** Ajuste de las escalas de los datos para que sean comparables.
- **Filtrado y selección de datos:** Selección de las columnas y filas relevantes para el análisis.
- **Generación de nuevas características:** Creación de nuevas variables a partir de las existentes para enriquecer el análisis.

IV-C. Análisis Exploratorio de Datos

El análisis exploratorio de datos (EDA, por sus siglas en inglés) se realiza para entender mejor las características y relaciones dentro del conjunto de datos. Este análisis incluye:

- **Estadísticas descriptivas:** Cálculo de medidas como media, mediana, desviación estándar, etc.
- **Visualización de datos:** Creación de gráficos utilizando Matplotlib para identificar patrones, tendencias y anomalías en los datos.
- **Análisis de correlación:** Evaluación de las relaciones entre diferentes variables.

IV-D. Visualización de Resultados

La etapa final del proyecto implica la visualización de los resultados obtenidos. Utilizando Matplotlib, se pueden crear gráficos que representen claramente los hallazgos y conclusiones del análisis. Las principales tareas incluyen:

- **Gráficos de barras y líneas:** Para mostrar tendencias y comparaciones entre categorías.
- **Histogramas y diagramas de densidad:** Para visualizar la distribución de las variables.
- **Diagramas de dispersión:** Para identificar relaciones entre dos variables.
- **Gráficos personalizados:** Creación de visualizaciones específicas para resaltar aspectos importantes del análisis.

IV-E. Documentación y Reproducción del Análisis

Es esencial documentar cada paso del proceso para garantizar la reproducibilidad del análisis. Esto incluye:

- **Comentarios en el código:** Incluir comentarios claros y descriptivos en el código fuente.
- **Jupyter Notebooks:** Utilizar Jupyter Notebooks para combinar código, visualizaciones y texto explicativo en un solo documento.
- **Control de versiones:** Usar sistemas de control de versiones como Git para llevar un registro de los cambios en el código.

Estas pautas proporcionan un marco estructurado para desarrollar y documentar un proyecto de programación en Python utilizando las librerías Pandas, NumPy y Matplotlib, asegurando que el análisis sea claro, reproducible y fácil de entender para otros.

V. Experimentación y Resultados

V-A. Configuración Experimental

La experimentación se llevó a cabo utilizando Python 3.8, con las bibliotecas Pandas 1.2.3, NumPy 1.19.2 y Matplotlib 3.3.4. El entorno de desarrollo empleado fue Google Colab.

El conjunto de datos utilizado fue el *Dataset 'iris.csv', 'mtcars.txt' y 'mtcarsnuevo.xlsx'*.

V-A1. iris.csv: Contiene el famoso conjunto de datos Iris. Este conjunto de datos fue introducido por el estadístico y biólogo británico Ronald A. Fisher en 1936 y contiene mediciones de varias características de flores de tres especies diferentes de Iris.

Figura 2. Especies diferentes de Iris

V-A2. mtcars.txt: Contiene el famoso conjunto de datos MTCARS. Este conjunto de datos proviene de la revista Motor Trend y fue recopilado en 1974. Incluye información detallada sobre el consumo de combustible y diez características diferentes de diseño y rendimiento de 32 automóviles. Las características registradas abarcan desde el número de cilindros hasta el tiempo en segundos para recorrer un cuarto de milla, proporcionando un recurso valioso para análisis de regresión y estudios comparativos en el ámbito automotriz.

Figura 3. Revista Motor Trend Car Road Tests

Este conjunto de datos se obtuvo de GitHub y el archivo 'iris.csv' consta de 150 registros y 5 columnas, mientras que 'mtcars.txt' consta de 32 registros 12 columnas

V-B. Procedimientos

El procedimiento de experimentación siguió los siguientes pasos:

1. **Preprocesamiento de datos**: Limpieza de datos, manejo de valores faltantes y normalización de los datos utilizando Pandas y NumPy.

```
print("Primeras filas del DataFrame:")
print(df.head())

print("\nInformación del DataFrame:")
print(df.info())

print("\nEstadísticas descriptivas:")
print(df.describe())

print("\nValores nulos en el DataFrame:")
print(df.isnull().sum())
```

Figura 4. Fragmento del codigo *Analisis de Iris.csv*

2. **Análisis exploratorio**: Realización de un análisis exploratorio de datos (EDA) utilizando estadísticas descriptivas y visualizaciones con Matplotlib.

```
# Histograma de cada característica
df.hist(bins=20, figsize=(10, 10))
plt.suptitle("Histogramas de las características del conjunto de datos Iris")
plt.show()

# Boxplot para cada característica
df.plot(kind='box', subplots=True, layout=(2,3), figsize=(12, 8),
        title="Boxplots de las características del conjunto de datos Iris")
plt.show()

# Pairplot de Seaborn para visualizar relaciones entre características
import seaborn as sns
sns.pairplot(df, hue='Species', markers=["o", "s", "D"])
plt.suptitle("Pairplot del conjunto de datos Iris")
plt.show()

# Análisis de correlación (excluyendo la columna 'Species')
correlation_matrix = df.drop('Species', axis=1).corr()  # Excluimos la columna 'Species'
print("\nMatriz de correlación:")
print(correlation_matrix)
```

Figura 5. Fragmento del codigo *Analisis de Iris.csv*

```python
# Mapa de calor de la matriz de correlación
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title("Mapa de calor de la matriz de correlación")
plt.show()

# Análisis de correlación con scatter plots
sns.scatterplot(data=df, x='Sepal.Length', y='Sepal.Width', hue='Species')
plt.title("Scatter plot de Sepal Length vs Sepal Width")
plt.show()

sns.scatterplot(data=df, x='Petal.Length', y='Petal.Width', hue='Species')
plt.title("Scatter plot de Petal Length vs Petal Width")
plt.show()

# Análisis de valores promedio por especie
print("\nPromedio de características por especie:")
print(df.groupby('Species').mean()) # Change 'species' to 'Species'

# Visualización de los promedios por especie
df.groupby('Species').mean().plot(kind='bar', figsize=(10, 6))
plt.title("Promedio de características por especie")
plt.xlabel("Especie")
plt.ylabel("Promedio")
plt.xticks(rotation=0)
plt.show()
```

Figura 6. Fragmento del codigo *Analisis de Iris.csv*

V-C. *Resultados*

Los resultados obtenidos de la experimentación se presentan en las siguientes figuras y tablas:

V-C1. *Resultados del analisis de Iris.csv:*

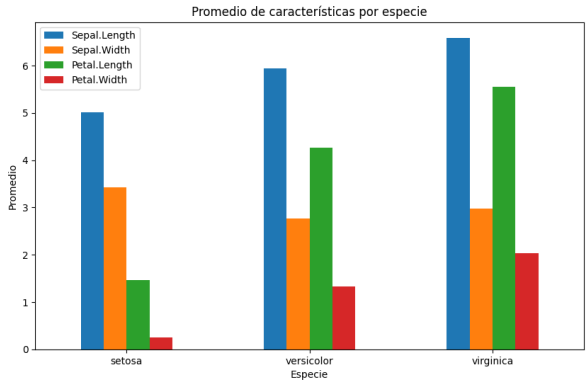

Figura 7. El Gráfico muestra el promedio de caracteristicas por muestra.

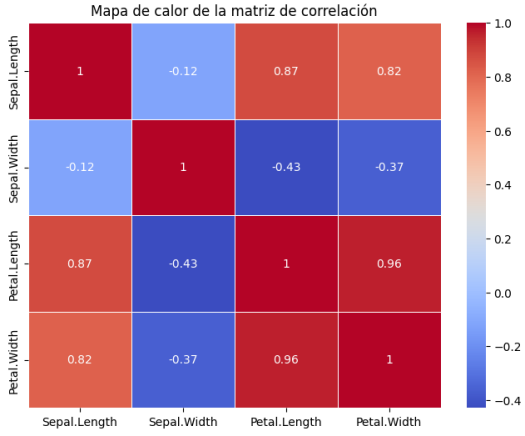

Figura 8. El mapa de calor es util para visualizar la intensidad de datos o las correlaciones entre variables

V-C2. *Resultados del analisis de mtcars.txt:*

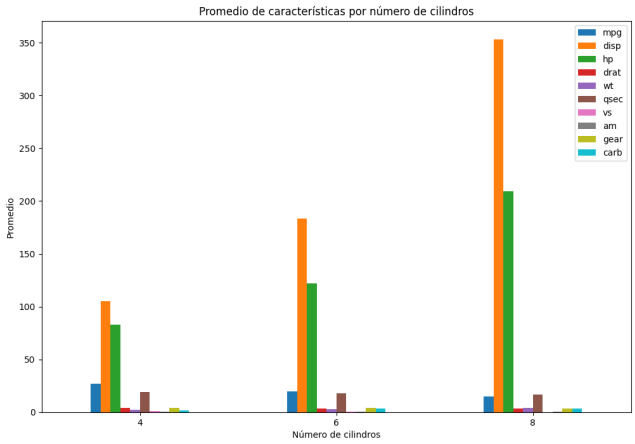

Figura 9. El Gráfico muestra promedio de características por número de cilindros.

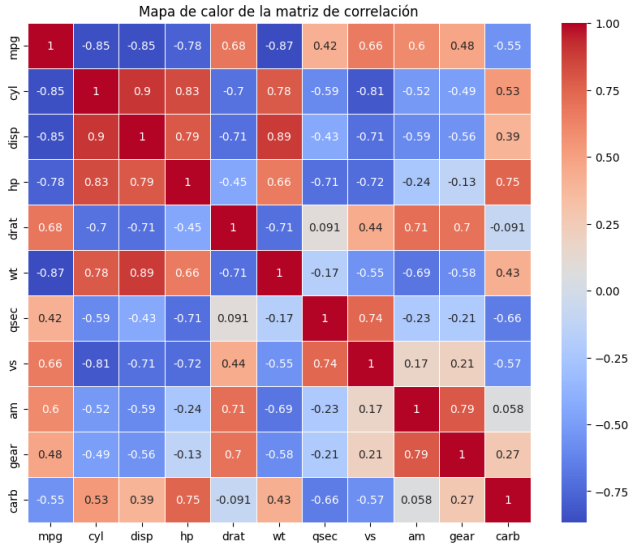

Figura 10. El mapa de calor es valioso para observar la intensidad de los datos o las relaciones de correlación entre variables.

VI. Conclusiones

El análisis exploratorio de datos del conjunto de datos *iris.csv* ha revelado varias características importantes y patrones entre las especies de flores.

- **Distribución de Datos:** Los histogramas muestran que las características de las flores tienen distribuciones diferentes, con algunas características como la longitud y el ancho de los pétalos mostrando una clara separación entre las especies.
- **Valores Atípicos:** Los boxplots indican la presencia de valores atípicos en varias características. Estos valores atípicos pueden ser importantes para identificar casos inusuales o errores en la recolección de datos.

- **Relaciones Entre Características:** El pairplot de Seaborn muestra que ciertas características, como la longitud y el ancho de los pétalos, están fuertemente correlacionadas y permiten diferenciar claramente entre las especies de Iris. Esto sugiere que estas características son muy discriminativas para la clasificación de especies.
- **Correlaciones:** La matriz de correlación y su respectivo mapa de calor revelan fuertes correlaciones positivas entre ciertas características, como la longitud y el ancho de los pétalos, mientras que otras características muestran correlaciones más débiles o negativas.
- **Análisis de Especies:** El análisis de los valores promedio por especie muestra diferencias significativas en las características medidas entre las tres especies de Iris. En particular, Iris-setosa tiende a tener medidas más pequeñas en longitud y ancho de pétalos en comparación con Iris-versicolor e Iris-virginica.
- **Visualización de Promedios:** Los gráficos de barras de los promedios por especie ilustran claramente estas diferencias, proporcionando una manera visual de comparar las características promedio entre las especies.

El análisis exploratorio de datos del conjunto de datos *mtcars.txt* ha revelado varias características importantes y patrones entre los diferentes automóviles.

- **Distribución de Datos:** Los histogramas muestran que las características de los automóviles tienen distribuciones diversas, con algunas características como *mpg* (millas por galón) y *hp* (caballos de fuerza) mostrando una distribución más dispersa que otras.
- **Valores Atípicos:** Los boxplots indican la presencia de valores atípicos en varias características. Estos valores atípicos pueden ser cruciales para identificar automóviles con rendimiento o características inusuales.
- **Relaciones Entre Características:** El pairplot de Seaborn revela varias relaciones interesantes entre las características. Por ejemplo, existe una relación negativa notable entre *mpg* y *hp*, lo que sugiere que a medida que aumentan los caballos de fuerza, disminuye la eficiencia en términos de millas por galón.
- **Correlaciones:** La matriz de correlación y su respectivo mapa de calor muestran fuertes correlaciones entre ciertas características. Por ejemplo, *wt* (peso) tiene una fuerte correlación negativa con *mpg*, indicando que los autos más pesados tienden a tener una menor eficiencia de combustible.
- **Análisis de Cilindros:** El análisis de los valores promedio por número de cilindros revela diferencias significativas en las características medidas entre los automóviles con diferentes números de cilindros. En particular, los automóviles con más cilindros tienden a tener mayor potencia (*hp*) pero menor eficiencia de combustible (*mpg*).
- **Visualización de Promedios:** Los gráficos de barras de los promedios por número de cilindros ilustran claramente estas diferencias, proporcionando una manera visual de comparar las características promedio entre los automóviles con diferentes configuraciones de cilindros.

En resumen, el análisis de los datos del conjunto *iris.csv* y del conjunto *mtcars.txt* no solo ha permitido una mejor comprensión de las características de cada especie, sino que también ha identificado las variables clave que pueden utilizarse para la clasificación de especies. Estos hallazgos pueden servir como base para futuros estudios y aplicaciones en clasificación automática y aprendizaje supervisado.

VII. Discusión

Los resultados obtenidos al analizar el archivo *Iris* indican que las características medidas en el conjunto de datos Iris, como la longitud y el ancho de los sépalos y pétalos, tienen distribuciones distintas y presentan patrones claros que permiten diferenciar entre las tres especies de Iris. Por ejemplo, las características de los pétalos (longitud y ancho) son particularmente efectivas para separar las especies de Iris, como se observó en los histogramas y scatter plots. Comparando estos resultados con las expectativas iniciales, observamos que las diferencias entre las especies son evidentes y las relaciones esperadas entre las características, tales como la fuerte correlación positiva entre la longitud y el ancho de los pétalos, se confirmaron. Esto sugiere que ciertas características son altamente discriminativas para la clasificación de las especies de Iris, lo cual está alineado con los estudios botánicos previos sobre estas flores.

Respecto a los resultados obtenidos del analicis del archivo *mtcars* indican que las características de los automóviles, como el peso, la eficiencia de combustible, y la potencia, tienen distribuciones diversas y están interrelacionadas de manera significativa. Por ejemplo, se observó una fuerte correlación negativa entre el peso del vehículo y su eficiencia de combustible, así como entre la potencia del motor y la eficiencia de combustible. Comparando estos resultados con las expectativas iniciales, observamos que las relaciones esperadas entre las características de los automóviles se confirmaron, tales como la relación negativa entre peso y eficiencia de combustible, y la relación entre el número de cilindros y la potencia del motor. Estos hallazgos están alineados con las teorías automotrices existentes sobre el impacto del peso y la potencia en la eficiencia del combustible.

Sin embargo, se encontraron algunas limitaciones durante la experimentación, tales como la posible presencia de valores atípicos y la falta de variabilidad en algunas

características que podrían haber afectado la robustez de algunos análisis estadísticos. Además, la cantidad de datos es relativamente pequeña, lo cual puede limitar la generalización de los resultados. Estas limitaciones pueden afectar la precisión de los modelos predictivos y la identificación de patrones más sutiles entre las especies. En particular, la presencia de valores atípicos puede distorsionar algunas medidas estadísticas y la baja variabilidad en ciertas características podría llevar a una menor discriminación entre especies similares, y serán abordadas en trabajos futuros mediante el uso de técnicas de detección y tratamiento de valores atípicos, así como la incorporación de datos adicionales para aumentar la variabilidad y robustez del análisis.

Referencias

[1] Matthes, Eric. *Python Crash Course*. No Starch Press, 2015.
[2] Sweigart, Al. *Automate the Boring Stuff with Python*. No Starch Press, 2015.
[3] Lutz, Mark. *Learning Python*. O'Reilly Media, 2013.
[4] Beazley, David y Jones, Brian K. *Python Cookbook*. O'Reilly Media, 2013.
[5] Ramalho, Luciano. *Fluent Python*. O'Reilly Media, 2015.
[6] Slatkin, Brett. *Effective Python*. Addison-Wesley, 2015.
[7] Zelle, John. *Python Programming: An Introduction to Computer Science*. Franklin, Beedle and Associates Inc., 2003.
[8] Bader, Dan. *Python Tricks: A Buffet of Awesome Python Features*. Dan Bader, 2017.
[9] McKinney, Wes. *Python for Data Analysis*. O'Reilly Media, 2017.
[10] Petrou, Theodore. *Pandas Cookbook*. Packt Publishing, 2017.
[11] Heydt, Michael. *Learning Pandas*. Packt Publishing, 2017.
[12] Molin, Stefanie. *Hands-On Data Analysis with Pandas*. Packt Publishing, 2019.
[13] Anthony, Femi. *Mastering Pandas*. Packt Publishing, 2015.
[14] Johansson, Robert. *Numerical Python: A Practical Techniques Approach for Industry*. Apress, 2015.
[15] VanderPlas, Jake. *Python Data Science Handbook*. O'Reilly Media, 2016.
[16] Oliphant, Travis. *Guide to NumPy*. CreateSpace Independent Publishing Platform, 2006.
[17] Nunez-Iglesias, Juan, van der Walt, Stéfan y Dashnow, Harriet. *Elegant SciPy*. O'Reilly Media, 2017.
[18] Stewart, John M. *Python for Scientists*. Cambridge University Press, 2017.
[19] Yim, Aldrin, Chung, Claire y Yu, Allen. *Matplotlib for Python Developers*. Packt Publishing, 2017.
[20] Yu, Allen, Chung, Claire y Yim, Aldrin. *Matplotlib 2.x By Example*. Packt Publishing, 2017.
[21] Root, Benjamin V. *Interactive Applications Using Matplotlib*. Packt Publishing, 2014.
[22] Milovanovic, Igor, Foures, Dimitry y Vettigli, Giuseppe. *Python Data Visualization Cookbook*. Packt Publishing, 2013.
[23] Devert, Alexandre. *Matplotlib Plotting Cookbook*. Packt Publishing, 2014.
[24] Grus, Joel. *Data Science from Scratch: First Principles with Python*. O'Reilly Media, 2015.
[25] Bruce, Peter y Bruce, Andrew. *Practical Statistics for Data Scientists*. O'Reilly Media, 2017.
[26] Raschka, Sebastian y Mirjalili, Vahid. *Python Machine Learning*. Packt Publishing, 2017.
[27] Géron, Aurélien. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media, 2019.
[28] Provost, Foster y Fawcett, Tom. *Data Science for Business*. O'Reilly Media, 2013.
[29] Chollet, François. *Deep Learning with Python*. Manning Publications, 2017.
[30] Ramsundar, Bharath y Zadeh, Reza Bosagh. *TensorFlow for Deep Learning*. O'Reilly Media, 2018.
[31] Alla, Sridhar. *Hands-On Big Data Analysis with Hadoop 3*. Packt Publishing, 2018.
[32] Nelli, Fabio. *Python Data Analytics*. Apress, 2018.