ISI UTN ROSARIO

Soporte a la Gestión de Datos con Programación Visual

Seminario
Teórico/Practico

"Análisis de Serie de Tiempo con Pandas"

Alumnos

Chelin, Martin Ezequiel (46287)

Velazquez, Sabrina (40173)

Mails

chelinmartin20@gmail.com

isabri91@gmail.com

Ciclo lectivo

2023

Contenido

Introducción

En la era de la información, los datos son esenciales para la toma de decisiones informadas y estratégicas en cualquier ámbito. Sin embargo, manejar y comprender grandes volúmenes de datos puede ser complicado. Es aquí donde entra en juego la poderosa librería Pandas de Python, que se ha convertido en la herramienta preferida por científicos de datos, analistas y profesionales en todo el mundo para el análisis y manipulación de datos.

A continuación, explicaremos cómo Pandas se ha convertido en la columna vertebral del análisis de datos en Python, proporcionando una infraestructura flexible y fácil de usar para transformar, limpiar y visualizar datos de manera eficiente. Desde la carga de datos hasta el procesamiento y la presentación de resultados, Pandas ofrece un conjunto de herramientas versátiles que nos permiten realizar tareas complejas con apenas unas pocas líneas de código.

A lo largo del seminario, descubriremos cómo utilizar Pandas para trabajar con diversos tipos de datos, así como cómo realizar operaciones esenciales de filtrado, combinación y limpieza. Además, mostraremos algunas técnicas avanzadas de manipulación de datos que nos permitirán extraer información valiosa de conjuntos de datos masivos.

Por último, este seminario también te guiará en la combinación de esta librería con Matplotlib, una herramienta esencial para la visualización de datos en Python. Se mostrarán los resultados en gráficos estadísticos que ayuden a comprender los patrones y tendencias emergentes de tus análisis de manera efectiva.

¿Qué es Pandas?

Pandas es una poderosa biblioteca de código abierto para Python utilizada principalmente para análisis de datos y manipulación de datos estructurados. Fue creada por Wes McKinney en 2008 y se ha convertido en una herramienta esencial para cualquier persona que trabaje con datos en Python. Algunas de las principales características y funcionalidades de Pandas incluyen:

- DataFrames: Tablas bidimensionales con etiquetas en filas y columnas, permitiendo un acceso rápido y flexible a los datos.
- Series: Estructura de datos unidimensional que puede contener datos de cualquier tipo (enteros, cadenas, objetos, etc.).
- Lectura y escritura de datos: Pandas puede leer y escribir datos desde y hacia diferentes formatos de archivos, como CSV, Excel, SQL, JSON, entre otros.
- Selección y filtrado de datos: Permite seleccionar y filtrar datos basados en criterios específicos.
- Agrupación de datos: Facilita la agrupación y resumen de datos mediante operaciones como "groupby".
- Manipulación de datos: Ofrece funciones para unir, combinar y transformar datos.
- Limpieza de datos: Permite manejar valores faltantes y datos inconsistentes.
- Operaciones aritméticas y estadísticas: Proporciona métodos para realizar operaciones matemáticas y estadísticas en los datos.

Pandas es ampliamente utilizado en ciencia de datos, análisis financiero, ingeniería, investigación académica y cualquier otro dominio donde sea necesario trabajar con datos estructurados. Su popularidad se debe a su eficiencia, facilidad de uso y la gran cantidad de funcionalidades que ofrece para el manejo y análisis de datos en Python.

Ventajas en el Análisis de Datos

- Facilidad de uso: Pandas proporciona una sintaxis clara e intuitiva que facilita la manipulación y análisis de datos mejorando productividad y legibilidad.
- Manejo de datos estructurados: Pandas está diseñado específicamente para trabajar con datos estructurados facilitando el almacenamiento y manipulación de datos tabulares.
- Eficiencia en el rendimiento: Rápido y escalable, lo que es crucial para manejar grandes conjuntos de datos.
- Visualización de datos: Se integra fácilmente con otras bibliotecas de visualización populares como Matplotlib y Seaborn, facilitando la generación de gráficos y visualizaciones a partir de los datos.
- Integración con otras bibliotecas de Python: Pandas se integra bien con otras bibliotecas populares de Python utilizadas en análisis de datos, como NumPy, SciPy y scikit-learn. Esto permite un flujo de trabajo más completo y versátil para análisis y modelado de datos.
- Comunidad activa: Pandas tiene una gran comunidad de usuarios y desarrolladores, lo que significa que hay una abundancia de recursos, documentación y soporte disponible en línea.

Estructuras de Datos

Series

Las series son estructuras unidimensionales conteniendo un array de datos (de cualquier tipo soportado por NumPy) y un array de etiquetas que van asociadas a los datos llamado índice.

Características:

- Datos homogéneos o heterogéneos: Una Serie puede contener elementos de un solo tipo de datos (homogéneo) o mezclar diferentes tipos de datos (heterogéneo). Por ejemplo, puedes tener una Serie que contenga solo números enteros o una Serie que contenga una combinación de números, cadenas y fechas.
- Etiquetas (índice): Cada elemento de una Serie está asociado con una etiqueta o índice. El índice permite acceder y etiquetar los elementos de la Serie. Por defecto,

el índice es una secuencia numérica comenzando desde 0, pero también puede ser un conjunto personalizado de etiquetas.

- Operaciones vectorizadas: Las Series en pandas soportan operaciones vectorizadas, lo que significa que puedes realizar operaciones aritméticas y funciones directamente en toda la Serie, lo que simplifica el procesamiento de datos.
- Etiquetado y alineación de datos: Las Series en pandas están diseñadas para alinear automáticamente los datos basándose en las etiquetas del índice. Esto significa que, si realizas operaciones entre dos Series con diferentes etiquetas de índice, pandas alineará los datos correctamente antes de realizar las operaciones.

Index	Data
0	Mark
1	Justin
2	John
3	Vicky

DataFrames

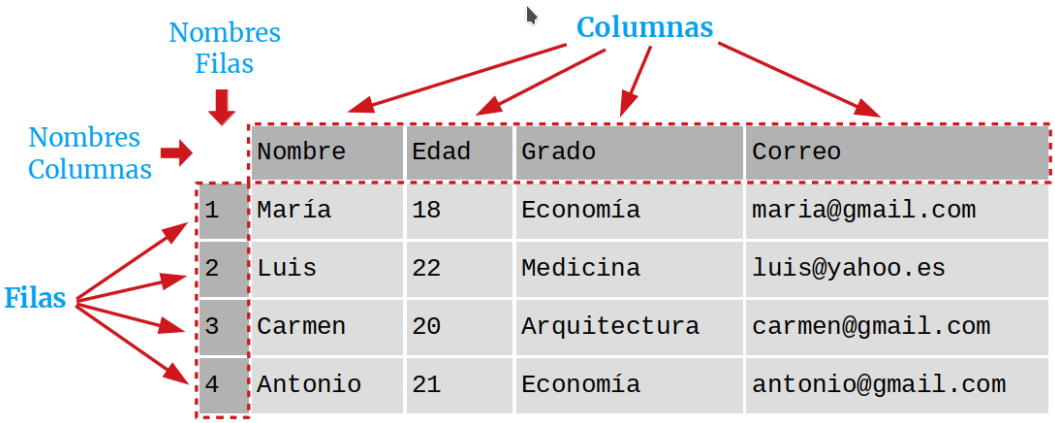
Un DataFrame es una estructura bidimensional compuesta por filas y columnas, se puede decir que es una agrupación de Series unidas bajo los mismos índices dando como resultado estructuras similares a tablas donde representar todo tipo de información.

Características:

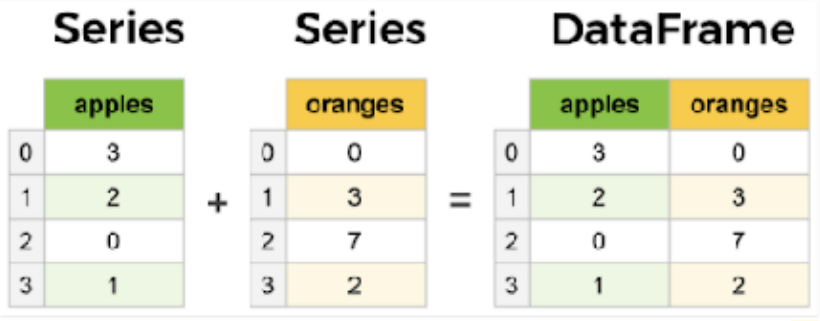
- Tabla bidimensional: Un DataFrame consta de filas y columnas, lo que lo hace adecuado para almacenar y trabajar con datos en formato tabular.
- Datos heterogéneos: Al igual que las Series, los DataFrames pueden contener datos homogéneos o heterogéneos. Esto significa que una columna del DataFrame puede contener elementos de un solo tipo de datos o una combinación de diferentes tipos de datos.
- Etiquetas de índice y columnas: Los DataFrames tienen etiquetas tanto para las filas como para las columnas. El índice de las filas permite etiquetar y acceder a

los datos de forma más intuitiva. Además, puedes asignar nombres a las columnas para identificarlas fácilmente.

- Flexibilidad: Los DataFrames son muy flexibles y pueden manejar diferentes formatos de datos, como listas, diccionarios, arreglos NumPy, otras Series y más. Además, permiten realizar operaciones vectorizadas, filtros, agregaciones y otras operaciones complejas de datos.
- Operaciones de SQL: Los DataFrames en pandas también proporcionan funcionalidades similares a las consultas de SQL, lo que permite realizar consultas, fusiones y operaciones de combinación de datos.



	Nombre	Edad	Grado	Correo
1	María	18	Economía	maria@gmail.com
2	Luis	22	Medicina	luis@yahoo.es
3	Carmen	20	Arquitectura	carmen@gmail.com
4	Antonio	21	Economía	antonio@gmail.com



Lectura y Escritura de Datos

Una de las tareas fundamentales en el análisis de datos es la importación y exportación de información desde y hacia diferentes fuentes. pandas, como una poderosa biblioteca de Python, ofrece herramientas eficientes para leer y escribir datos desde diversas fuentes, como archivos CSV, Excel, JSON y bases de datos, facilitando el proceso de preparación y manipulación de datos.

- Lectura de datos:
  - Leer datos desde un archivo CSV: **"pd.read_csv()"**.
  - Leer datos desde un archivo Excel: **"pd.read_excel()"**.
  - Leer datos desde un archivo JSON: **"pd.read_json()"**.
  - Leer datos desde una base de datos: **"pd.read_sql()"**.
- Escritura de datos:
  - Guardar los datos en un archivo CSV: **"pd.to_csv()"**.
  - Guardar los datos en un archivo Excel: **"pd.to_excel()"**.
  - Guardar los datos en un archivo JSON: **"pd.to_json()"**.

Manipulación y Limpieza de Datos

Una parte fundamental del análisis de datos es la preparación adecuada de los datos antes de comenzar cualquier análisis o modelado. La manipulación y limpieza de datos es una etapa crucial que implica filtrar, transformar y corregir los datos para garantizar que sean coherentes y adecuados para el análisis. pandas, como una biblioteca de Python, ofrece una amplia gama de funciones para llevar a cabo estas tareas de manera eficiente y efectiva.

La manipulación y limpieza de datos son pasos críticos para garantizar que los resultados del análisis sean precisos y confiables. Mediante ejemplos prácticos, abordaremos diversos escenarios y desafíos que se presentan en esta etapa y aprenderemos a enfrentarlos con confianza utilizando las herramientas poderosas que pandas ofrece.

- Filtrado y Selección de Datos:
    - Consultar una columna mediante su nombre:

      “Nombre_DataFrame[“Nombre_Columna”, “…”]”

    - Seleccionar una fila mediante su etiqueta o índice:

      “Nombre_DataFrame.loc[“Nombre_Fila”]”

      “Nombre_DataFrame.iloc[índice]”

    - Seleccionar una celda en particular:

      “Nombre_DataFrame[“Nombre_Fila”, ”Nombre_Columna”]”

    - Selección condicionada:

      “Nombre_DataFrame[Condición]”

- Tratamiento de Valores Faltantes:
    - Comprobar registros nulos:

      “Nombre_DataFrame.isnull()”

    - Descartar filas o columnas con registros nulos:

      “Nombre_DataFrame.dropna()”

      “Nombre_DataFrame.dropna(axis=1)”

    - Rellenar los registros de las filas vacias con un valor:

      “Nombre_DataFrame.fillna(value=0)”

Operaciones en DataFrames

Una de las características más poderosas de pandas es su capacidad para realizar operaciones y cálculos en los DataFrames de manera eficiente y sencilla. Las operaciones en DataFrames nos permiten realizar manipulaciones, agregaciones y transformaciones de datos para obtener información valiosa y respuestas a preguntas clave en el análisis de datos.

- Agregación:
    - Agrupar filas en función de una columna:

      “Nombre_DataFrame.groupby(“Nombre_Columna”)”

    - Calcular media y desviación estándar:

“Nombre_DataFrame.groupby(“Nombre_Columna”).mean()”

“Nombre_DataFrame.groupby(“Nombre_Columna”).std()”

    - Calcular Mínimo y Máximo:

“Nombre_DataFrame.loc[Nombre_DataFrame.groupby[“Columna”].idxmin()]”

“Nombre_DataFrame.loc[Nombre_DataFrame.groupby[“Columna”].idxmax()]”

    - Contar registros de cada columna:

“Nombre_DataFrame.groupby(“Nombre_Columna).count()”

    - Reporte de analíticas descriptivas por empresa:

“Nombre_DataFrame.groupby(“Nombre_Columna).describe()”

    - Transpuesta:

“Nombre_DataFrame.transpose()”

- Operación de Transformación y Cálculos:
    - Primeras y últimas filas:

“Nombre_DataFrame.head()”

“Nombre_DataFrame.tail()”

- Array de valores únicos de una columna:

  “Nombre_DataFrame[“Nombre_Columna”].unique()”

- Contador de valores únicos de una columna:

  “Nombre_DataFrame[“Nombre_Columna”].nunique()”

- Dataframe con los valores únicos y su contador de una columna:

  “Nombre_DataFrame[“Nombre_Columna”].value_counts()”

- Aplicar una función definida:

  “Nombre_DataFrame[“Nombre_Columna”].apply(funcion)”

- Ordenar por columna (inplace=False por defecto):

  “Nombre_DataFrame.sort_values(by=“Columna”,ascending=Boolean]”

- Agregar una columna (inplace=False por defecto):

  “Nombre_DataFrame[Nombre_Columna_Nueva] = Valor”

- Borrar una fila o columna (inplace=False por defecto):

  “Nombre_DataFrame.drop(“Nombre_Columna, axis=1”)”

- Combinaciones:
  - Concatenación (Junta filas, requiere que las dimensiones sean iguales):

    “pd.concat([Nombre_DataFrame_1, Nombre_DataFrame_2, ….])”

  - Fusión (Une tablas a partir de una columna común):

    “pd.merge(Nombre_DataFrame_1, Nombre_DataFrame_2, on=”Columna”)”

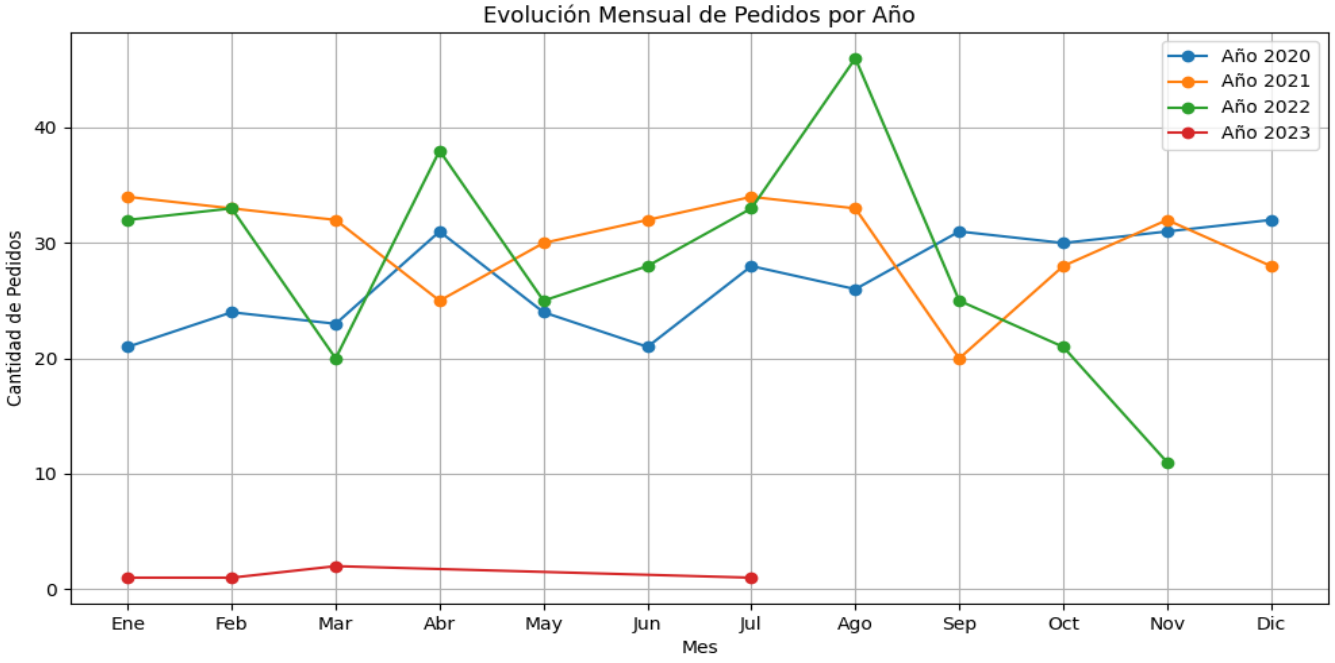
  - Unión (Une Columnas mediante los índices):

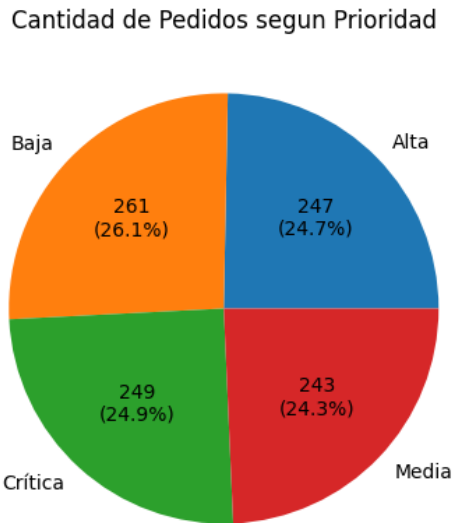
    “Nombre_DataFrame_1.join(Nombre_DataFrame_2)”

Visualización de Datos

La visualización de datos es una herramienta esencial para comprender y comunicar patrones, tendencias y relaciones dentro de nuestros conjuntos de datos. pandas, junto con la biblioteca matplotlib, ofrece capacidades robustas para la creación de gráficos y visualizaciones interactivas que nos permiten explorar y presentar nuestros datos de manera efectiva.

A continuación, se muestran ejemplos de graficas calculadas desde DataFrames con la ayuda de la librería mencionada anteriormente:





Comparativa con otras alternativas

SciPy y Dask son dos bibliotecas populares de Python utilizadas en el análisis de datos y la computación científica junto a Pandas. Aunque pueden tener ciertas similitudes, también tienen diferentes objetivos y funcionalidades. A continuación, compararemos estas dos bibliotecas:

SciPy

- Objetivo: Se enfoca principalmente en la computación científica y matemática, proporcionando funciones avanzadas para tareas como álgebra lineal, optimización, interpolación, integración numérica, estadísticas, procesamiento de señales, entre otras.
- Características principales:
    - Módulos especializados: Proporciona una amplia variedad de módulos especializados para diversas tareas matemáticas y científicas.
    - Integración con NumPy: Se basa en NumPy y utiliza sus arrays para representar datos, lo que facilita la interoperabilidad con otras bibliotecas de análisis de datos y visualización.

Dask

- Objetivo: Se enfoca en la computación paralela y distribuida, y proporciona estructuras de datos y API similares a Pandas y NumPy, pero capaces de trabajar con datos más grandes que no caben en la memoria RAM.
- Características principales:
    - DataFrames y Array distribuidos: Proporciona versiones distribuidas de DataFrames y Array, que permiten realizar operaciones paralelas y distribuidas en clústeres de máquinas.
    - Escalabilidad: Es adecuado para trabajar con grandes conjuntos de datos, ya que divide las operaciones en tareas más pequeñas que se pueden ejecutar en paralelo.
    - Integración con Pandas y NumPy: Se puede utilizar como una extensión de Pandas y NumPy, permitiendo a los usuarios aprovechar la funcionalidad de estas bibliotecas y escalarla a datos más grandes con Dask.

En resumen, Pandas es ideal para el análisis y manipulación de datos estructurados en la memoria RAM, SciPy es excelente para tareas matemáticas y científicas avanzadas, mientras que Dask es una opción cuando se necesita escalar el análisis de datos a grandes conjuntos de datos distribuidos o cuando los datos superan la memoria disponible.

Conclusión

Durante este seminario sobre Pandas en Python, hemos explorado las potentes capacidades de esta biblioteca para el análisis y manipulación de datos. Hemos aprendido cómo Pandas ofrece estructuras de datos flexibles como DataFrames y Series, que permiten trabajar con datos de manera intuitiva y eficiente. Posterior, hemos visto cómo utilizar Pandas para cargar datos, limpiarlos, filtrarlos, agruparlos y realizar diversas operaciones de transformación como manipular datos, crear nuevas columnas, eliminar valores nulos y realizar análisis estadísticos de manera sencilla y efectiva. Además, hemos descubierto cómo Pandas se integra con otras bibliotecas de Python, como NumPy y Matplotlib, lo que nos permite combinar sus funcionalidades para realizar análisis más avanzados y visualizaciones atractivas.

Adicionalmente, se han presentado algunos aspectos de otras bibliotecas relacionadas, como SciPy para cálculos científicos y Dask para computación distribuida, lo que amplía aún más nuestras capacidades para manejar grandes conjuntos de datos y tareas más complejas.

En conclusión, Pandas se ha revelado como una herramienta poderosa y versátil para el análisis de datos en Python. Su facilidad de uso, su capacidad para manejar datos estructurados y su integración con otras bibliotecas hacen de Pandas una opción inestimable para realizar un análisis de forma efectiva y eficiente en trabajar con datos en Python.

Referencias

https://pandas.pydata.org/

https://pandas.pydata.org/docs/reference/api/pandas.Series.html

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html

https://aprendeconalf.es/docencia/python/manual/pandas/

https://matplotlib.org/

https://www.dask.org/

https://scipy.org/

https://pandas.pydata.org/pandas-docs/version/0.13.1/visualization.html

Repositorios

https://github.com/frro-soporte/G2310/tree/master/SEMINARIO