Iteración Eficiente con Pandas

José Luis Rodríguez, Gonzalo Martínez y Alexandre Muñoz

November 2023

1 Experiencias pasadas con el aprendizaje automático y el uso de Pandas.

Hemos tenido bastantes experiencias pasadas en las que hemos usado Pandas en diferentes proyectos.

Sí, hemos creado funciones combinando columnas en conjuntos de datos utilizando la biblioteca de Python llamada Pandas. Pandas es una herramienta poderosa y ampliamente utilizada en la manipulación y análisis de datos, especialmente cuando trabajamos con conjuntos de datos tabulares, como hojas de cálculo o bases de datos.

Algunos ejemplos de funciones que hemos creado al combinar columnas en conjuntos de datos incluyen:

1. **Cálculo de Medianas**: Podemos combinar varias columnas numéricas para calcular la mediana de un conjunto de datos. Esto es útil cuando necesitamos resumir datos numéricos y entender la tendencia central de un conjunto de datos.

2. **Cálculo de Promedios Ponderados**: A veces, es necesario calcular promedios ponderados en función de los valores de dos o más columnas. Esto es especialmente útil en situaciones donde ciertos valores tienen un peso mayor que otros en el cálculo del promedio.

3. **Creación de Nuevas Categorías**: También podemos combinar columnas para crear nuevas categorías o características derivadas. Por ejemplo, podemos combinar las columnas de "fecha" y "hora" para crear una nueva columna de "marca de tiempo" que sea más fácil de analizar o filtrar.

4. **Manipulación de Cadenas de Texto**: Pandas nos permite combinar columnas de cadenas de texto y realizar operaciones de manipulación, como concatenación, división, extracción de subcadenas, entre otros. Esto es útil para limpiar y transformar datos de texto.

También hemos enfrentado problemas de rendimiento debido a procesar grandes volúmenes de datos. A medida que el tamaño del conjunto de datos aumenta, las operaciones de lectura, escritura y manipulación de datos pueden volverse considerablemente más lentas. Las operaciones que son eficientes en conjuntos de datos pequeños pueden requerir mucho más tiempo en grandes conjuntos de datos.

En lugar de procesar el conjunto de datos completo, a veces hemos tenido que aplicar técnicas de muestreo y filtrado para trabajar con subconjuntos más pequeños de datos, lo que puede mejorar significativamente el rendimiento ya que, Pandas proporciona herramientas para realizar muestreo y filtrado de datos de manera eficiente.

En resumen, Pandas es una herramienta esencial para trabajar con datos, y su flexibilidad para combinar y manipular columnas de datos es fundamental para realizar análisis de datos efectivos y extraer información útil de conjuntos de datos complejos.

2 Técnicas para Iterar Filas en Pandas

Iterar sobre filas en Pandas es una tarea común al realizar análisis de datos. A continuación, se presentan algunas técnicas para iterar sobre filas en un DataFrame de Pandas que hemos usado en algunos proyectos, cada una con sus ventajas y desventajas:

2.1 Usando `iterrows()`

Ventajas:

- Proporciona un índice de fila y una Serie de Pandas para cada fila, lo que facilita el acceso a elementos específicos.
- Fácil de entender y usar.

Desventajas:

- Puede ser lento en DataFrames grandes debido a la naturaleza iterativa de la función.
- No es eficiente si se planea realizar modificaciones en el DataFrame original.

Ejemplo:

```
for index, row in df.iterrows():
    print(row['Column1'], row['Column2'])
```

2.2 Usando `itertuples()`

Ventajas:

- Más rápido que `iterrows()` ya que devuelve una tupla nativa en lugar de una Serie de Pandas.
- Adecuado para acceder a valores, pero no para modificaciones.

Desventajas:

- No es la mejor opción si necesita realizar modificaciones en el DataFrame original.

Ejemplo:

```
for row in df.itertuples():
    print(row.Index, row.Column1, row.Column2)
```

2.3 Usando `apply()` con `axis=1`

Ventajas:

- Permite aplicar una función a cada fila y es más eficiente que las dos opciones anteriores.
- Se puede utilizar para realizar modificaciones en el DataFrame.

Desventajas:

- Puede ser un poco más complejo de usar en comparación con `iterrows()` e `itertuples()`.

Ejemplo:

```
def process_row(row):
    return row['Column1'] + row['Column2']

df['NewColumn'] = df.apply(process_row, axis=1)
```

2.4 Usando Vectorización

La manera más eficiente de iterar sobre filas en un DataFrame de Pandas es utilizando la vectorización. Esto implica aplicar operaciones directamente a las columnas o filas sin necesidad de bucles explícitos. La vectorización es altamente eficiente y está optimizada para aprovechar la velocidad de las operaciones en matrices y arreglos NumPy, que son la base de Pandas.

Detrás de escena, la vectorización aprovecha la optimización de bajo nivel proporcionada por las bibliotecas subyacentes, lo que hace que las operaciones sean más rápidas y eficientes en comparación con las técnicas de iteración tradicionales. Además, la vectorización es más legible y fácil de entender, ya que permite expresar las operaciones de manera más concisa y declarativa.

Por ejemplo, en lugar de utilizar un bucle para sumar dos columnas en cada fila, la vectorización permite realizar la suma directamente utilizando una expresión como:

$$df['NuevaColumna'] = df['Columna1'] + df['Columna2']$$

Esta técnica es altamente eficiente y suele ser la preferida en tareas de procesamiento de datos en Pandas. Sin embargo, es importante comprender las operaciones vectorizadas y cuándo aplicarlas, ya que no son aplicables a todas las situaciones. La elección de la técnica depende de las necesidades específicas de la tarea y del tamaño de los datos.

2.5 Ventajas e Inconvenientes de la Vectorización

2.5.1 Ventajas de la Vectorización:

1. **Eficiencia**: La vectorización es altamente eficiente y rápida en comparación con otras técnicas de iteración, como `iterrows()` o `itertuples()`. Aprovecha las optimizaciones de bajo nivel de las bibliotecas subyacentes, como NumPy.

2. **Legibilidad**: La vectorización permite expresar las operaciones de manera más concisa y legible. Esto hace que el código sea más fácil de entender y mantener.

3. **Optimización automática**: Pandas y NumPy realizan automáticamente optimizaciones en las operaciones vectorizadas para mejorar el rendimiento.

4. **Facilidad de uso**: No requiere bucles explícitos ni un conocimiento profundo de Python, lo que la hace accesible incluso para usuarios con menos experiencia.

2.5.2 Desventajas de la Vectorización:

1. **Limitaciones**: No todas las operaciones pueden ser vectorizadas. Algunas tareas complejas pueden requerir bucles o enfoques personalizados.

2. **Uso de memoria**: En algunas situaciones, la vectorización puede consumir más memoria que otros enfoques, especialmente cuando se crean DataFrames muy grandes.

3. **Requiere conocimiento**: Aunque es más fácil de usar en comparación con bucles, aún se requiere un buen conocimiento de las operaciones vectorizadas y de Pandas para aplicarlas correctamente.

En resumen, la vectorización es la técnica más eficiente para iterar sobre filas en Pandas y es preferible en la mayoría de los casos. Sin embargo, es importante tener en cuenta las limitaciones y comprender cuándo y cómo aplicarla adecuadamente en función de las necesidades específicas del análisis de datos. Ya que, en algunos casos específicos donde la velocidad no es una preocupación principal y se necesita un acceso detallado a cada fila, `iterrows()` puede ser adecuado. La elección de la técnica depende de las necesidades específicas de su tarea y del tamaño de sus datos.

3 Reflexión

Nuestra perspectiva sobre la iteración en Pandas ha evolucionado a medida que hemos adquirido experiencia en el campo de la ciencia de datos. A lo largo de nuestra carrera, hemos enfrentado diversas situaciones que nos han llevado a aprender lecciones valiosas en relación con la iteración en Pandas.

En nuestros primeros encuentros con Pandas, solíamos recurrir a bucles explícitos para iterar sobre filas y columnas de DataFrames. Aunque esta aproximación funcionaba, pronto nos dimos cuenta de que tenía limitaciones en términos de eficiencia y legibilidad del código. Con el tiempo, descubrimos la vectorización, una técnica que cambió completamente nuestro enfoque y mejoró la eficiencia de nuestro trabajo.

La vectorización nos permitió realizar operaciones en DataFrames de manera más eficiente y legible. En lugar de utilizar bucles, podíamos aplicar operaciones directamente a columnas o filas, lo que mejoró la velocidad de procesamiento y facilitó la comprensión del código. Esto cambió nuestra perspectiva al hacer que comprendiéramos la importancia de la optimización y la legibilidad del código en el análisis de datos.

Además, aprendimos que la elección de la técnica de iteración adecuada depende de la tarea y del tamaño de los datos. A veces, la vectorización es la mejor opción, pero en situaciones más complejas, los bucles pueden ser necesarios. Esta flexibilidad nos permitió abordar una variedad de desafíos en proyectos de análisis de datos.

Para nuestro futuro como científicos de datos, estas lecciones son invaluables. La eficiencia en el procesamiento de datos es esencial, especialmente al lidiar con grandes conjuntos de datos en el mundo real. La vectorización es una técnica que optimiza el rendimiento y mejora la legibilidad del código, lo que nos permite ser más productivos y efectivos en nuestro trabajo.

También aprendimos que la exploración y la experimentación son cruciales para adquirir habilidades en Pandas. La práctica constante y la resolución de problemas del mundo real son las mejores formas de dominar la iteración en Pandas y mejorar nuestras capacidades como científicos de datos.

En resumen, nuestra perspectiva sobre la iteración en Pandas ha evolucionado hacia un enfoque más eficiente y optimizado. Hemos aprendido a valorar la importancia de la vectorización y la legibilidad del código, lo que será fundamental para nuestro futuro como científicos de datos. La práctica constante y la adaptabilidad son claves para el éxito en este campo en constante evolución.