Books2Scrape

# Objetivo

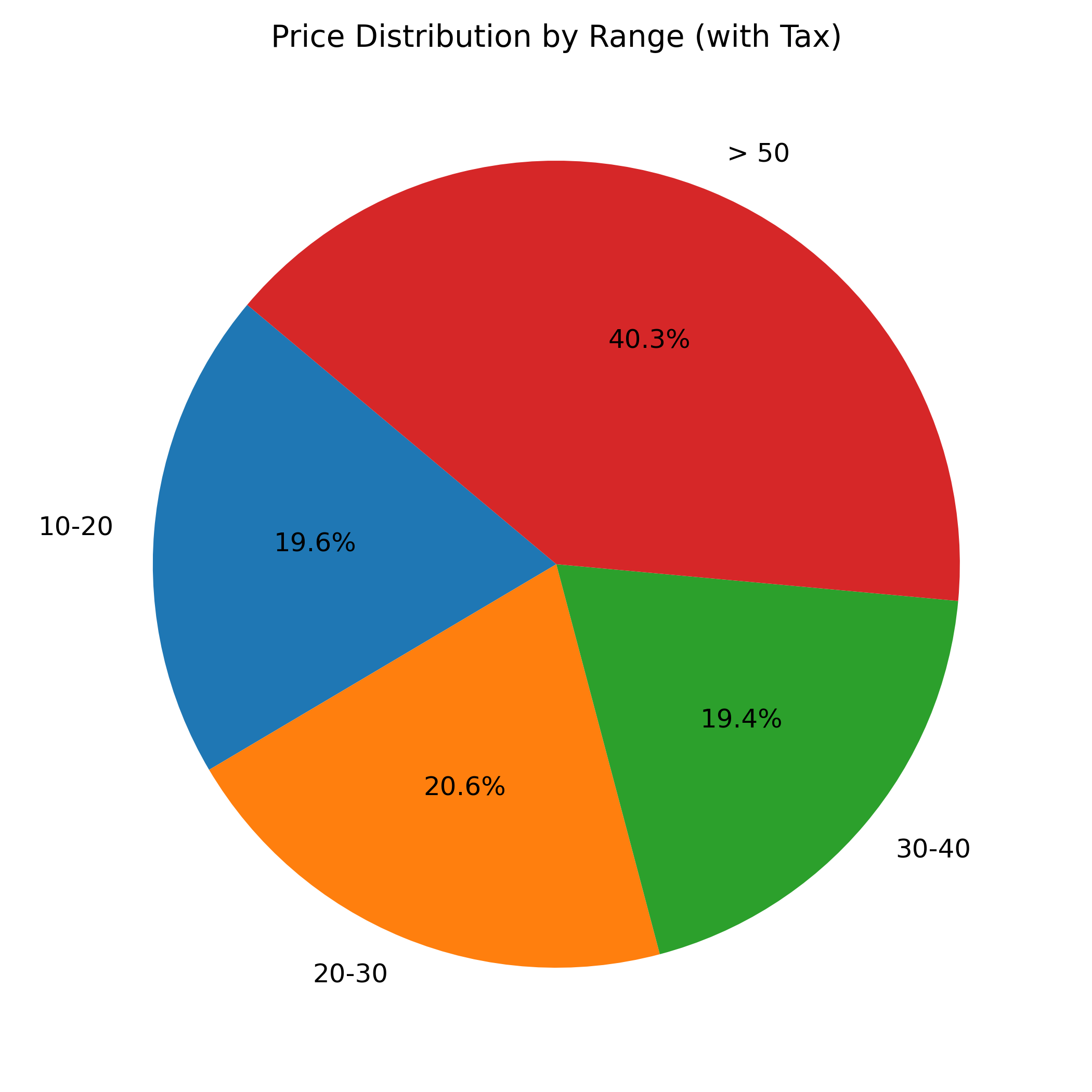
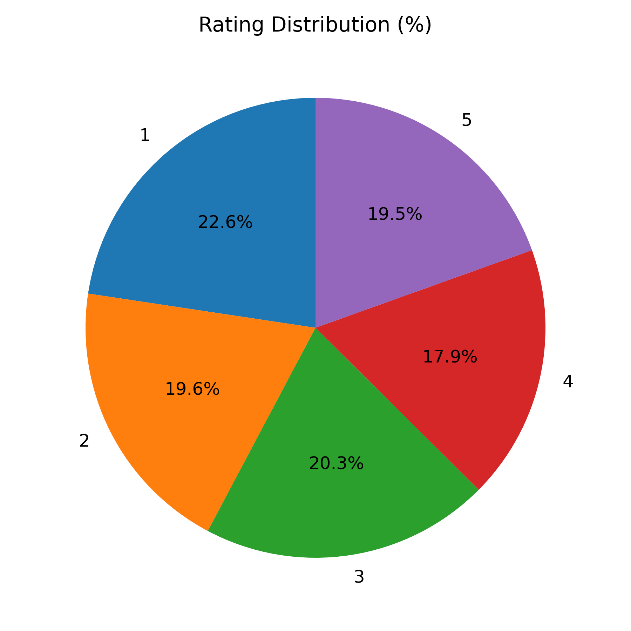
Desarrollar una aplicación que realice web scraping de la página http://books.toscrape.com/ con el fin de procesar los datos obtenidos para generar informes y gráficos. Para la creación de esta herramienta, se tuvieron en cuenta gran parte de las funcionalidades mencionadas en el enunciado del ejercicio. A continuación, se detalla cada una:

* Webscraping: La información fue obtenida mediante técnicas de extracción de datos web, utilizando principalmente la librería Scrapy para recolectar la mayoría de los datos de los libros, y BeautifulSoup4 para extraer las categorías. Se eligió Scrapy por su velocidad, escalabilidad y conjunto de funcionalidades. Además, el sitio web presenta pocos elementos visuales y de backend, por lo que su eficiencia es aún mayor. En cuanto a las categorías, se utilizó BeautifulSoup4, ya que más simple para obtener esa parte de los datos.
* Limpieza y transformación: Se utilizó Pandas para la limpieza y transformación de los datos, debido a su simplicidad y eficiencia en el manejo de estructuras. Pandas es una librería rápida, flexible, con gran legibilidad y capacidades de autocorrección.
* Reportes y agregación: Para la creación de gráficos se empleó Matplotlib, ya que permite generar visualizaciones directamente desde el código, de forma flexible y personalizable. Para la generación de reportes y tablas dinámicas (pivot tables), se utilizó nuevamente Pandas, por las razones mencionadas anteriormente.
* Estrategia contra bloqueos de datos: Si bien la página no contaba con un antibot, se tomaron ciertas medidas para no ser bloqueado, entre ellas:
  + User-agents: cada pedido de información a la página puede usar aleatoriamente uno de tres agentes declarados, además del propio.
  + Proxies: cada pedido de información a la página puede usar aleatoriamente uno de los multiples proxies declarados, además del propio.
  + Intentos: Cada URL tiene 2 intentos, lo que le permite hacer dos combinaciones de proxy + user-agent
  + Retardo: Tiene seteado por default entre 0.1 y 0.2 segundos, además de esto tiene activado la configuración “AUTOTHROTLE” lo que le permite elegir el retraso conveniente entre un pedido y otro (configurado entre 0.1 y 1).
  + Optimización: con todos los ajustes de seguridad anteriormente nombrados, el proceso se hizo exponencialmente más lento, pasando de ~1 minuto, a más de 1 hora, por lo que se habilitaron, configuración de multiproceso de pedidos, y pedidos en paralelo.
* Desafíos: El mayor desafío fue aprender a utilizar Scrapy, al ser distinta a las tecnologías a las que se acostumbran a trabajar en las área de data, fue necesario comprender rápidamente sus fundamentos y estructura interna, tanto de sus clases, como sus configuraciones. En cuanto a los datos, se pudieron procesar de manera adecuada, realizándoles algunas modificaciones a su diseño original, para que los mismos sean más útiles. Se realizó una segunda fase de scraping accediendo a la URL de libro, lo que permitió generar un archivo final más completo.
* Capacidad de procesamiento y escalabilidad: Para un mejor rendimiento se evitó el uso de bucles innecesarios, para procesar información se utilizó código vectorizado, para evitar crear archivos de 0 se creó una función que agrega las nuevas líneas al archivo final, utilizando el campo “UPC”. El código este habilitado para ser escalable al usar rutas relativas, código modularizado, un proceso de instalación sencillo, y ajustes optimizados para una correcta extracción, permiten que el código se ejecute relativamente rápido, y que además sea fácilmente tweakable.
* Se aplicaron las siguientes buenas prácticas:
  + Código modularizado
  + Manejo de errores
  + Uso de rutas relativas
  + Parámetros de configuración personalizables
  + Limpieza de datos
  + Estructura clara de clases
  + Comentarios breves y útiles
  + Eliminación de código repetido
* Documentación: Se creo un repositorio en GitHub con instrucciones sobre como descargar, instalar dependencias, y ejecutar el código. Además de este archivo.

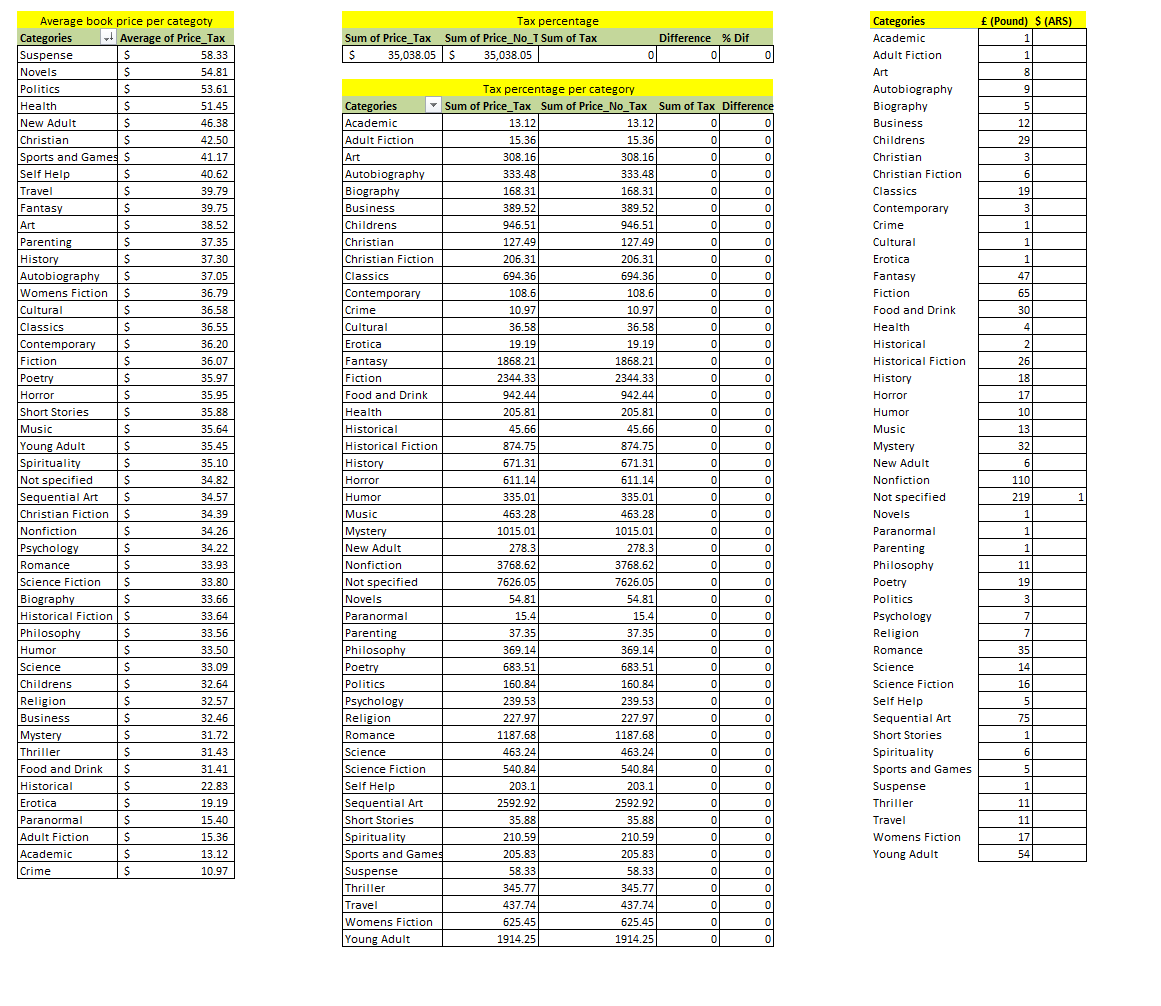
# Resultados

Al realizar la extracción, limpieza y preparación de datos obtuvo un archivo crudo final, el cual posee los siguientes campos: UPC, Category, Title, Currency, Price\_Tax ,Price\_No\_Tax, Tax, Stock\_availability, Stock\_quantity y URL. Con estos se crearon múltiples reportes entre ellos:

* Promedio de precio por categoría
* Mejores libros económicos
* Mejores libros caros
* Mejores libros con poco stock
* Mejores categorías con poco stock
* Cantidad de libros por categoría
* Dos graficas que se adjuntan a continuación:



Nota: Si se hubiese tenido información más variada, se podrían haber generado más reportes, entre ellos los siguientes ejemplos



# Mejoras y siguientes pasos

Dado el tiempo limitado con el que se contó, considero que hay algunos aspectos del proyecto que podrían mejorarse. Entre ellos:

* Implementación de una base de datos: Sería conveniente incorporar algún tipo de base de datos para el almacenamiento de los datos procesados. En caso de tratarse de un proyecto de gran volumen, se recomienda utilizar tecnologías como Spark u otras herramientas orientadas al procesamiento de datos.
* Optimización del procesamiento: Podria considerarse el uso de Polars o NumPy para mejorar la eficiencia en el tratamiento de grandes volúmenes de datos.
* Mayor manejo de errores: Se podría mejorar el tratamiento de errores tanto en los procesos automatizados mediante Airflow, como en el código ejecutado en Jupyter Notebook.
* Automatización con Power BI: En lugar de generar reportes en Excel, una mejor opción, sería automatizar la actualización de los datos mediante una API de Power BI, permitiendo así mostrar los reportes desde una nube, donde todas las personas (que se quiera) puedan ver.
* Verificación de cambios en los datos: Sería útil incorporar una función que detecte si hubo modificaciones en los datos, para actualizar la base de datos únicamente cuando sea necesario, optimizando así los recursos.
* Generación de pruebas unitarias: La implementación de tests unitarios permitiría validar el comportamiento esperado del código, facilitando el mantenimiento y la escalabilidad del proyecto.