THE BRIDGE

Lucas Perez Barone

Project Break I:
Análisis Exploratório de Datos (EDA)

Data Science Repository Trends on GitHub: An Insightful EDA

Índice

1. Introducción

1.1. Panorama

La Ciencia de Datos es un campo que ha experimentado un crecimiento y desarrollo acelerados en los últimos años. Aproximadamente hace 12 años, la revista Harvard Business Magazine publicó un artículo en el que afirmaba categóricamente que el Científico de Datos era la profesión más "atractiva" del siglo XXI. En ese momento, el tema era poco discutido. Sin embargo, poco después, la demanda de profesionales en esta área aumentó considerablemente, a pesar de que el número de especialistas era limitado. De manera rápida, el campo fue expandiéndose, atrayendo cada vez más interés, hasta el punto de que ciertos sectores ya se encuentran algo saturados.

Más de 10 años después, hemos sido testigos de varias revoluciones en la Ciencia de Datos, impulsadas por la rapidez con la que se desarrollan nuevas técnicas y tecnologías innovadoras. El Aprendizaje Automático (Machine Learning), el Aprendizaje Profundo (Deep Learning), el Procesamiento de Lenguaje Natural (Natural Language Processing) y la Inteligencia Artificial Generativa (IA Generativa) son solo algunos ejemplos de tecnologías que han transformado la profesión, los profesionales y las corporaciones que las implementan.

Analytics And Data Science

Data Scientist: The Sexiest Job of the 21st Century

Meet the people who can coax treasure out of messy, unstructured data. by Thomas H. Davenport and DJ Patil

From the Magazine (October 2012)

2. Objetivos

2.1. Hipótesis

Considerando el contexto descrito, la hipótesis formulada y probada en este proyecto es que existen subáreas dentro de la ciencia de datos que actualmente están en ascenso y que se espera que tengan un impacto significativo en el mercado en un futuro cercano. Específicamente, la hipótesis es que los repositorios de determinados temas emergentes tendrán una repercusión más elevada en comparación con aquellos que se aparten de dichos temas.

2.2. Global

El presente proyecto tiene como objetivo principal observar la evolución del campo de la Ciencia de Datos y la emergencia de nuevas tendencias en el área, con el fin de generar insights a partir de estas.

2.3. Objetivos específicos

- Visión general sobre el aumento del número de técnicas y subáreas dentro de la Ciencia de Datos a lo largo de los años;
- Identificar cuáles de estas técnicas y subáreas han aumentado su relevancia en los últimos años y cuáles han perdido utilidad;
- Buscar temas específicos dentro de la Ciencia de Datos que estén relacionados con la mayor popularidad y el mayor nivel de compromiso en los repositorios;
- Generar insights sobre los resultados obtenidos.

3. Métodos

3.1. Obtención de los Datos

Todos los datos utilizados en este proyecto fueron obtenidos a través de la REST API de GitHub mediante la creación de un token personal. Se utilizó el endpoint para repositorios, ya que estos eran los objetos de análisis. El enlace a la documentación oficial de la API se presenta a continuación:

https://docs.github.com/en/rest?apiVersion=2022-11-28

La tabla siguiente resume todos los filtros que pueden especificarse en una solicitud de la API, de los cuales se utilizó un rango de fechas (created_at = range(2010, 2025)), filtrando la variable topic por 'data-science', lo que permitió recopilar información de todos los repositorios etiquetados con data-science desde 2010 (primer repositorio con dicha etiqueta) hasta el 12 de diciembre de 2024.

Parámetro	Descripción	Valores/Ejemplos
topic:	Filtra repositorios por tema (data-science, machine-learning, etc.)	`data-science`, `machine-learning`, etc.
stars:>=	Filtra repositorios con al menos un número determinado de estrellas.	`>=100`, `>=50`, etc.
fork:	Filtra repositorios que son bifurcaciones de otros repositorios.	`true`, `false`
language:	Filtra repositorios por lenguaje de programación.	`python`, `javascript`, `ruby`, etc.
created:	Filtra repositorios creados en un rango de fechas específicas (inicio..fim)	`year(4d)-month(2d)-day(2d)` / `2022-05-15`, etc.
updated:	Filtra repositorios actualizados hasta una fecha específica.	`year(4d)-month(2d)-day(2d)` / `2022-05-01`, etc.
is:	Filtra repositorios públicos o privados	`public`, `private`
sort:	Ordena los resultados por un campo específico.	`stars`, `updated`, `forks`, `help-wanted-issues`
order:	Define el orden de clasificación.	`asc`, `desc`

Para más detalles, consulte el Jupyter Notebook disponible en el enlace a continuación, que proporciona una descripción detallada de todo el proceso seguido.

https://github.com/neural-insights/EDA_GitHub_Repositories/blob/master/src/notebooks/getting_GH_datasets.ipynb

Se obtuvo un conjunto de 48,774 datos y 81 variables.

4. Análisis Exploratorio de Datos

Se hará un breve resumen de los siguientes temas, ya que son demasiado extensos. Para más información, consulte el Jupyter Notebook disponible en el siguiente enlace:

https://github.com/neural-insights/EDA_GitHub_Repositories/blob/master/main.ipynb

1.1. Previsualización del DataFrame y datos

Tras una primera visualización de los datos, se optó por mantener únicamente las columnas listadas en la tabla a continuación para continuar con la etapa de tratamiento y limpieza de los datos.

Columna	Descripción	dtype
`id`	El identificador único del repositorio.	int64
`name`	El nombre del repositorio.	object (str)
`full_name`	El nombre completo del repositorio, incluyendo el nombre del propietario.	object (str)
`owner`	Un objeto que contiene información sobre el propietario del repositorio	object (dict)
`stargazers_count`	El número de estrellas (stars) que el repositorio ha recibido.	int64
`forks_count`	El número de forks (copias) realizadas a partir del repositorio.	int64
`watchers_count`	El número de observadores que están siguiendo las actividades del repositorio.	int64
`open_issues_count`	El número de problemas (issues) abiertos en el repositorio.	int64
`topics`	Una lista de temas y palabras clave asociados al repositorio.	object (list)
`created_at`	La fecha y hora en que el repositorio fue creado.	datetime64[ns, UTC]
`language`	El principal lenguaje de programación utilizado en el repositorio.	object(str)
`has_issues`	Valor que indica si el repositorio tiene issues (problemas) habilitados.	bool
`has_projects`	Valor que indica si el repositorio tiene proyectos habilitados.	bool
`has_downloads`	Valor que indica si el repositorio permite la descarga de archivos.	bool
`has_discussions`	Valor que indica si el repositorio tiene discusiones habilitadas.	bool
`has_wiki`	Valor que indica si el repositorio tiene una wiki habilitada.	bool
`license`	El tipo de licencia bajo la cual el repositorio está disponible.	object (dict)
`score`	Un valor numérico (0-1) que representa la relevancia de un repositorio en los resultados de una búsqueda.	int64

4.2. Limpieza y preparación de los datos para análisis;

En esta etapa, se realizó el tratamiento de los valores nulos de diversas maneras; algunos datos fueron eliminados y otros imputados manualmente. Dado que la API de GitHub ya proporciona los datos de manera muy bien estructurada, no fue necesario dedicar mucha atención a los valores nulos.

También se trataron los valores duplicados, manteniendo solo una versión de cada uno.

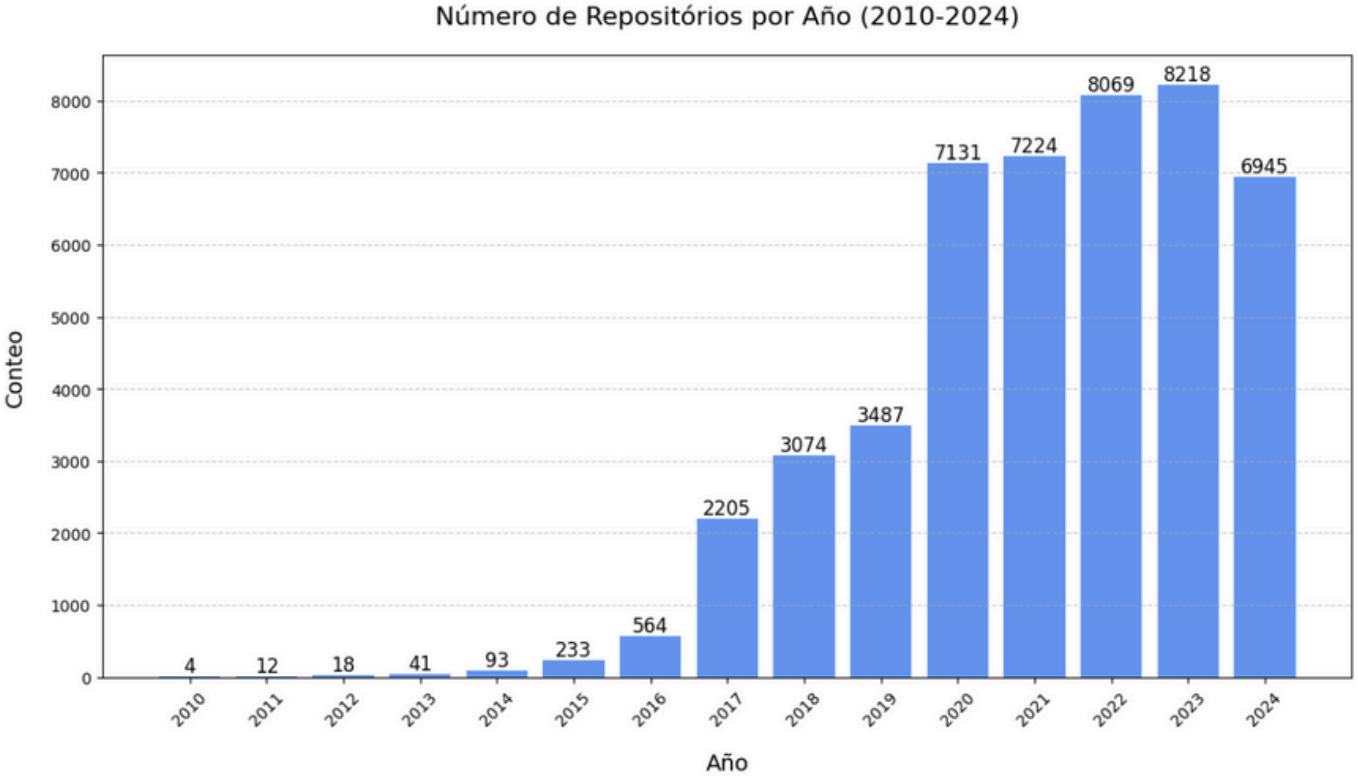
Nuestra columna de interés, topics, presentaba algunos problemas para los objetivos específicos de nuestro análisis. Existían decenas e incluso cientos de tópicos redundantes y equivalentes, como por ejemplo 'machine-learning', 'machine-learning-algorithms', 'machine-learning-for-business', 'ml', entre otros. Para resolver esto, se utilizó una función denominada posteriormente 'plot_string_with_normalization', que empleó expresiones regulares para identificar las cadenas equivalentes y reemplazarlas por un único valor, unificándolas.

La tabla siguiente resume los dataframes generados tras la limpieza.

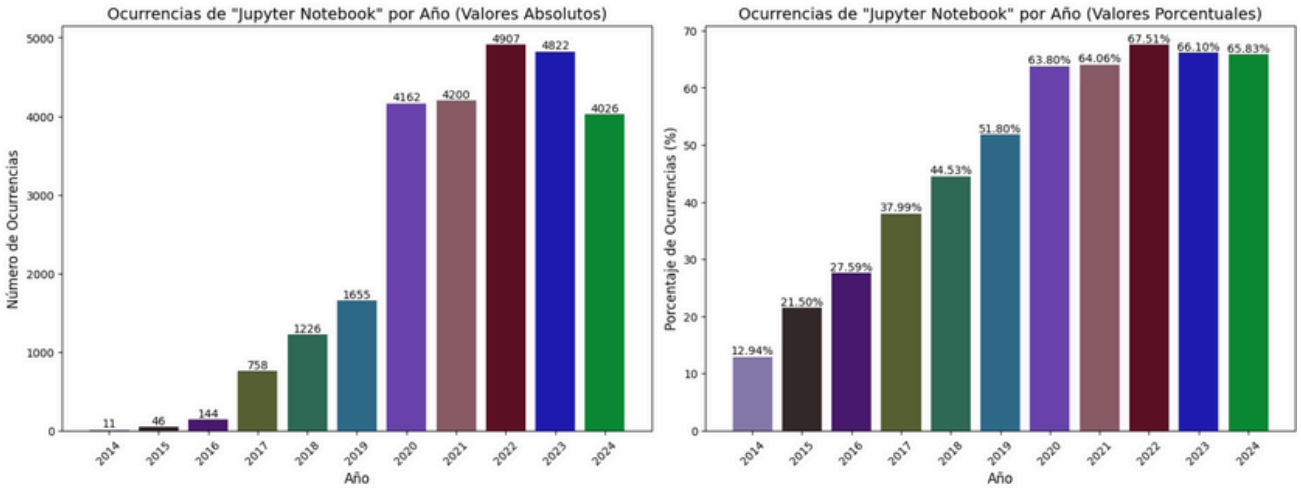
DataFrame	Descripción
`df_2010_2024`	df que incluye todos los datos de 2010 a 2024 sin tratamiento
`df_clean`	df con imputación manual de valores faltantes en la columna 'topics'
`df_check_duplicate`	df con eliminación de valores duplicados
`df`	df preparado para análisis, sin nulos ni duplicados
`df_language`	df preparado para análisis que utilizen el procesamiento de la variable 'language'

4.3. Análisis global y visualización;

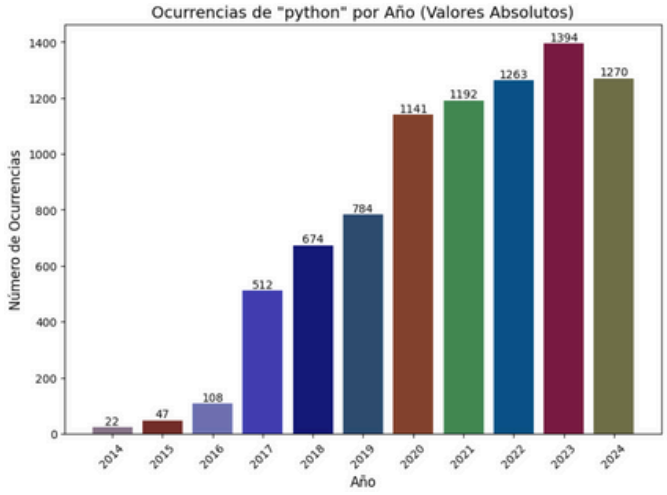
El gráfico a continuación evidencia el aumento en el número total de repositorios públicos en GitHub con la etiqueta 'data-science'. Se observa un aumento acelerado en tan solo 14 años, especialmente entre 2016-2017 y 2019-2020, con incrementos de 290.9% y 104.5%, respectivamente.

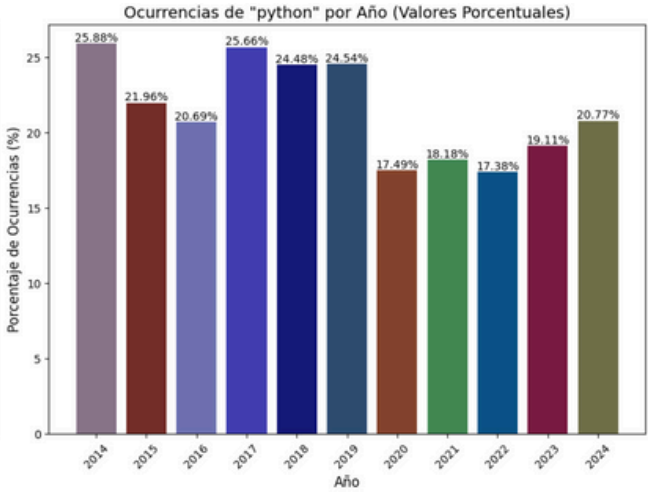


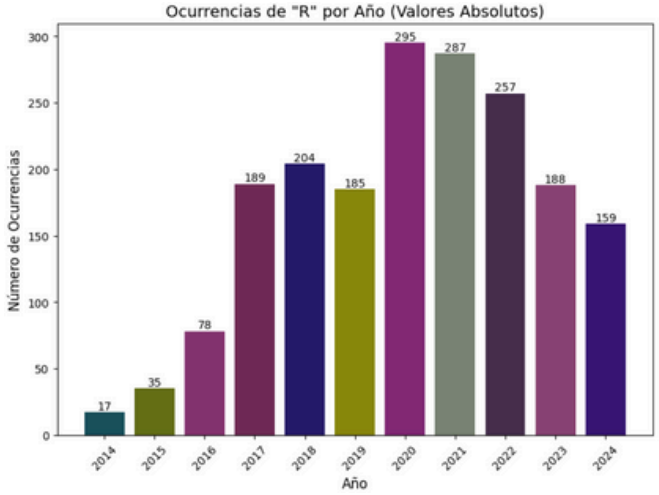
El gráfico siguiente muestra el crecimiento de la herramienta Jupyter Notebook tanto en valores absolutos (izquierda) como en porcentajes (derecha), normalizados por el total de repositorios en el año.

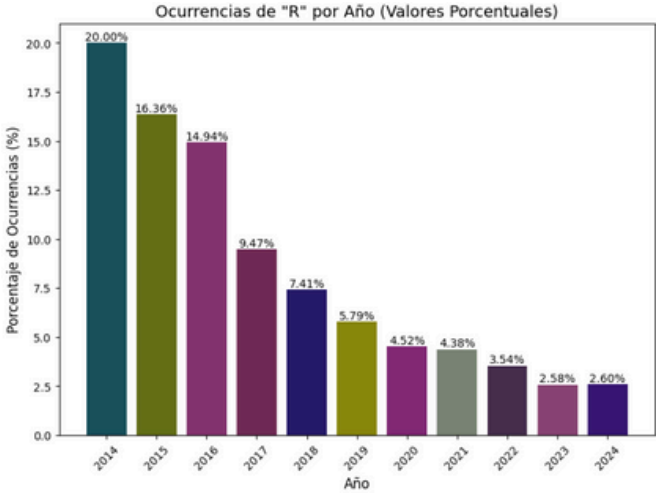


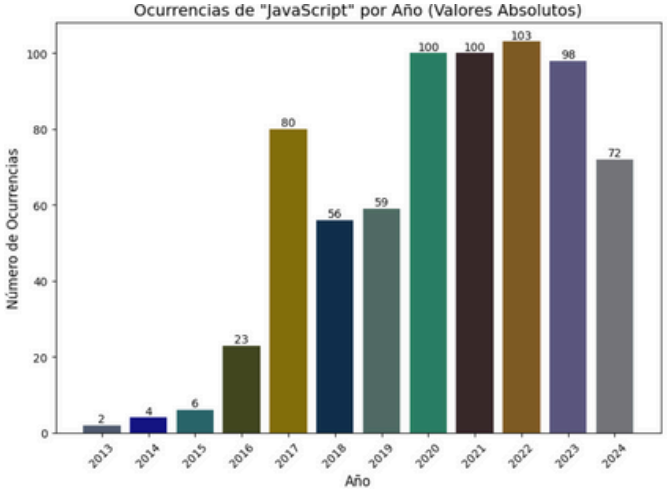
El lenguaje Python parece tener una participación constante y consistente, aunque presenta algunas oscilaciones. Por otro lado, lenguajes clásicos como R y JavaScript muestran un claro y progresivo decrecimiento.

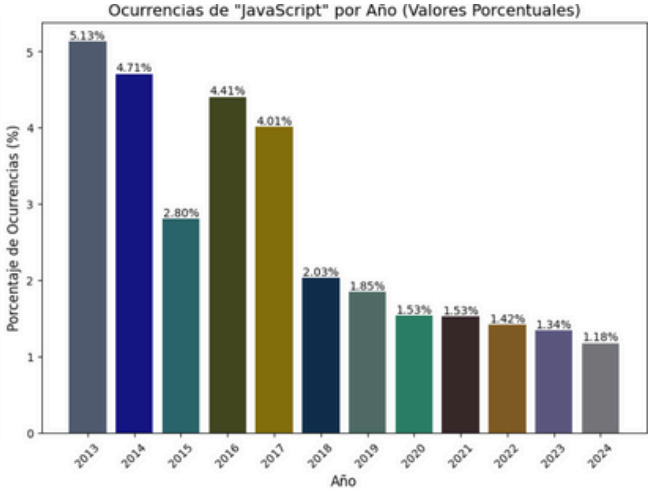




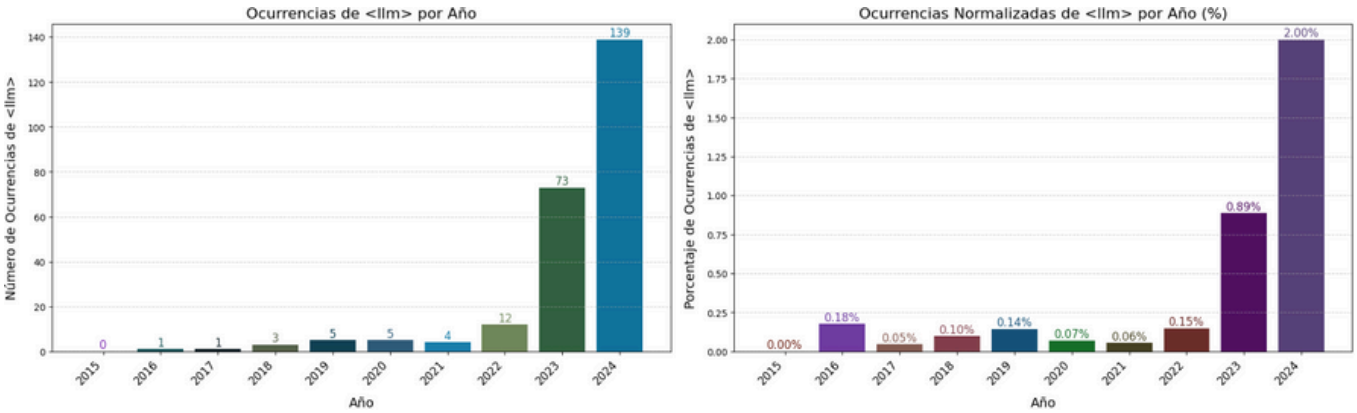


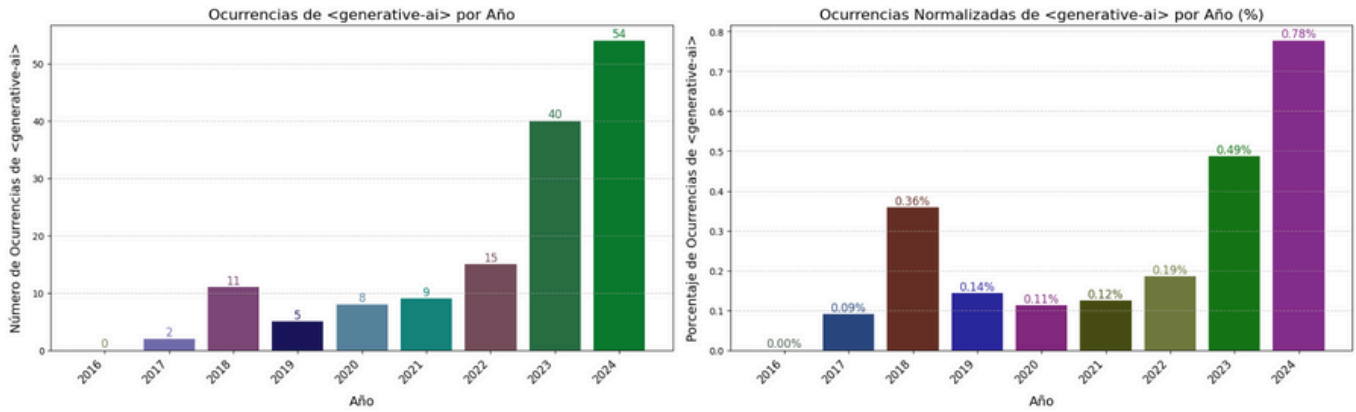


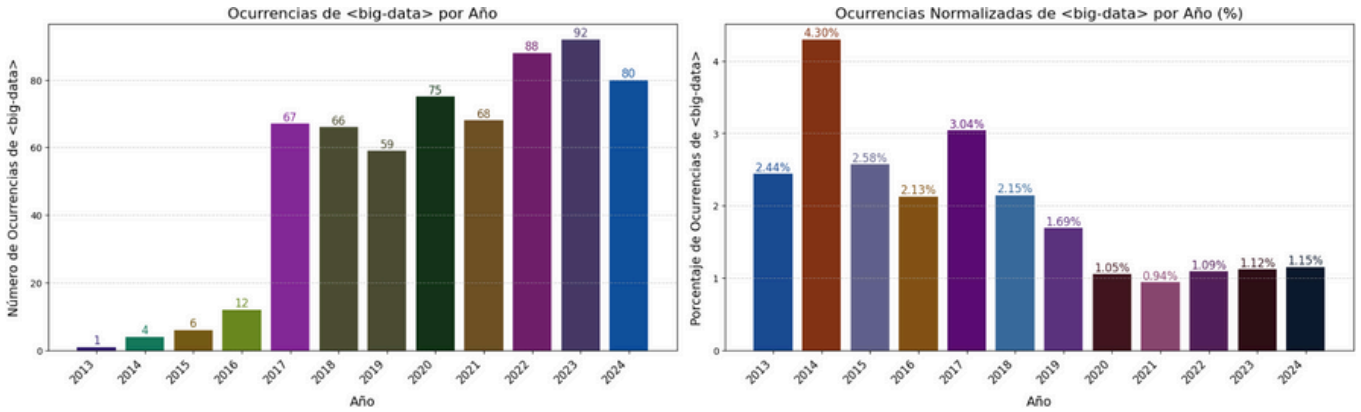




Los gráficos a continuación muestran los valores absolutos y relativos de los repositorios filtrados por diferentes temas, tras el tratamiento de las cadenas redundantes y repetidas mencionadas en el ítem 4.2. Esta prueba se realizó para numerosos temas; a continuación, se presentan algunos ejemplos:

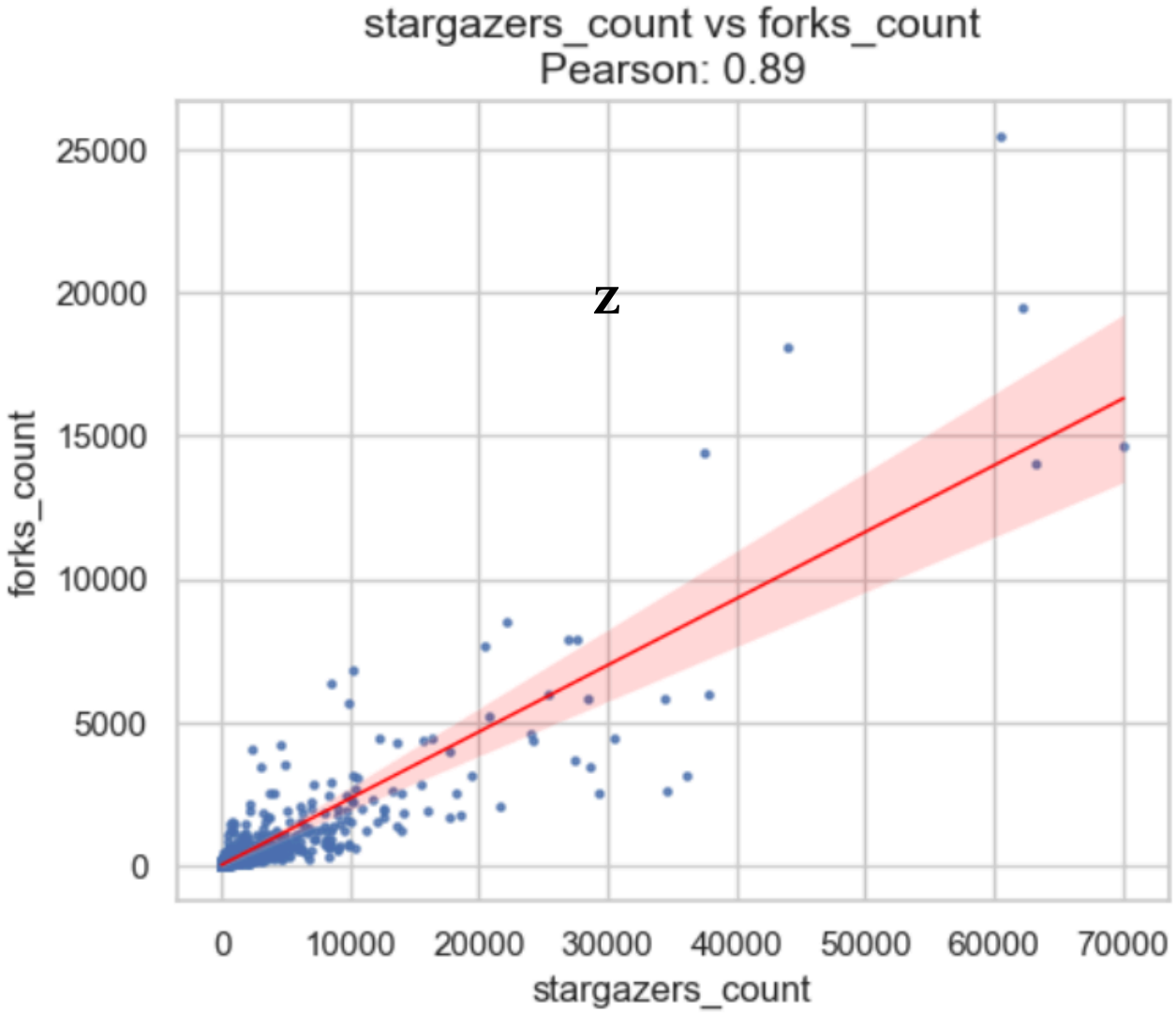






Los modelos de Large Language Models (LLMs) han sido, con diferencia, los que han mostrado el crecimiento más abrupto en los últimos dos años, y por ello serán el foco de los análisis siguientes. Generative AI también presenta un crecimiento, aunque más moderado. Por otro lado, Big Data ha venido perdiendo participación.

Hasta el momento, lo único que se ha hecho ha sido observar el número de repositorios creados con cada tema, pero esto no es un buen indicador del impacto en la comunidad y en el mercado. Por ello, se estableció una métrica de repercusión e impacto de los repositorios basada en el número de stargazers_count (estrellas) y forks_count (ramas). Las estrellas reflejan cuántas personas han "marcado" el repositorio como interesante y se utilizarán como medida de popularidad. Las ramas reflejan cuántos repositorios derivados se han creado a partir de este repositorio, lo que constituye un indicador claro de compromiso activo.



El gráfico anterior muestra que las variables de estrellas (medidor de popularidad) y ramas (compromiso activo) presentan una fuerte correlación positiva (Pearson: 0,89). Por ello, se utilizarán conjuntamente como métrica para estimar la repercusión de un repositorio.

A partir de este punto, se tuvo que ejecutar una serie de nuevas limpiezas y transformaciones para garantizar la credibilidad de las pruebas de hipótesis. El conjunto de datos en cuestión presenta una distribución completamente aberrante y asimétrica, con más de la mitad de los repositorios teniendo un valor de 0 en estrellas y ramas, mientras que los repositorios más populares superan los 5000.

```
count    47318.000000
mean        60.673676
std        981.126563
min          0.000000
25%          0.000000
50%          0.000000
75%          2.000000
95%         39.000000
max      70101.000000
Name: stargazers_count, dtype: float64

Mediana: 0.0

Número de datos com stargazers_count = 0: 25389
Número de datos en todo el DataFrame 47318
Más de: 0.54 % de los datos son iguales a 0
```

Esta distribución completamente desbalanceada puede generar desconcierto en un primer momento y levantar sospechas sobre la confiabilidad de los datos. Sin embargo, parece que todo está en orden con ellos, y lamentablemente, este fenómeno sería esperado. GitHub, al igual que en redes sociales como Instagram, presenta un escenario donde unos pocos repositorios acaparan la mayor parte de la atención debido a factores como la visibilidad de sus creadores, la viralidad, la calidad de la documentación o la resolución de problemas comunes. Este fenómeno sigue la **Ley de Potencia**, en la que pocos repositorios son muy populares, mientras que la mayoría permanece en la oscuridad. Los repositorios menos conocidos carecen de los mismos recursos de marketing y difusión, lo que limita su visibilidad y compromiso. Así, se genera un círculo de retroalimentación positiva para los repositorios más populares, aumentando aún más la desigualdad de visibilidad en la plataforma.

Los datos fueron sometidos a una normalización temporal. Esto es relevante porque, cuanto más antiguo es un repositorio, mayor tiempo de exposición ha tenido para ser encontrado por los usuarios. Naturalmente, los repositorios más recientes pueden no tener valores absolutos tan altos, pero sí un mayor impacto. Por ejemplo, un repositorio de 2024 con un forks_count de 5 podría ser más relevante en términos de compromiso que uno de 2014 con 13 forks. Así, el objetivo de la normalización es "penalizar" los repositorios más antiguos y aumentar el peso de los más recientes.

Posteriormente, se generaron nuevos dataframes filtrados para contener únicamente los valores positivos superiores a 1,5 veces el IQR, es decir, el 1% más alto. Dado que el objetivo aquí es evaluar la repercusión, no tendría sentido utilizar datos con valor 0.

A continuación, se utilizaron técnicas de transformación logarítmica para intentar reducir aún más la asimetría de la distribución, que seguía desbalanceada.