
KAGGLE CHRONICLES: 15 YEARS OF COMPETITIONS, COMMUNITY, AND DATA SCIENCE INNOVATION

Kevin THEITCROW Bönisch
Goethe-University Frankfurt, Germany
k.boenisch@outlook.com
kaggle.com/kevinbnisch

Brando BWANDOWANDO Losaria
Pasig City, Philippines
brando.losaria@gmail.com
kaggle.com/bwandowando

July 21, 2025

Created as part of the Meta Kaggle Hackathon



Team BIERBINGKA

ABSTRACT

Since 2010, Kaggle has been a platform where data scientists from around the world come together to compete, collaborate, and push the boundaries of Data Science. Over these 15 years, it has grown from a purely competition-focused site into a broader ecosystem with forums, notebooks, models, datasets, and more. With the release of the Kaggle Meta Code and Kaggle Meta Datasets, we now have a unique opportunity to explore these competitions, technologies, and real-world applications of Machine Learning and AI. And so in this study, we take a closer look at 15 years of data science on Kaggle—through metadata, shared code, community discussions, and the competitions themselves. We explore Kaggle’s growth, its impact on the data science community, uncover hidden technological trends, analyse competition winners, how Kagglers approach problems in general, and more. We do this by analyzing millions of kernels and discussion threads to perform both longitudinal trend analysis and standard exploratory data analysis. Our findings show that Kaggle is a steadily growing platform with increasingly diverse use cases, and that Kagglers are quick to adapt to new trends and apply them to real-world challenges, while producing—on average—models with solid generalization capabilities. We also offer a snapshot of the platform as a whole, highlighting its history and technological evolution.

Contents

1	Introduction	3
2	Kaggle User Growth	4
2.1	Early Years (2010-2015)	4
2.2	Expansion and Pre-Pandemic Acceleration (2015-2020)	4
2.3	COVID Pandemic era to Present (2020-present)	4
2.4	Anomaly Detection Using Sliding Window Z-test	6
2.5	Identified Events and its effects on User Registration	6
2.6	Conclusion	8
3	Investigating Meta Kaggle Code	10
3.1	R or Python?	10
3.2	What Packages & Methods do Kagglers use in General?	11
3.2.1	Over Time	13
3.2.2	By Category	14
3.2.3	torch vs tensorflow	16
3.3	Competition Trends & Insights	16
3.3.1	What Packages are Kagglers using in Competitions?	17
3.3.2	Public vs. Private Leaderboard Scores	17
3.4	What Technologies do (winning) Kagglers use?	22
3.4.1	Technology Diversity per Competition over Time	24
3.4.2	Competition & Technologies Network	26
3.5	Conclusion	26
4	Kaggle Competition Topics	28
4.1	What is Topic Modeling?	28
4.2	Topic Modeling Kaggle Competitions	28
4.3	Topic Modeling Specific Competitions	31
4.4	Conclusion	31

1 Introduction

TL;DR

We begin each section with a TL;DR (*Too long; didn't read*) box, which provides a rough summary of the insights presented in that section. These summaries are not intended to replace the full content, but rather to serve as a skim guide for identifying the most relevant sections.

Kaggle was founded in 2010 by Anthony Goldbloom and Ben Hamner with the vision of creating a platform where data scientists from around the world could compete to develop the best possible solutions—measured by a specified metric—to a given problem within a limited timeframe, driven by the motivation of earning medals, recognition, and prize money. Over time, Kaggle has evolved far beyond these competition-focused roots. With the introduction of dedicated discussion forums, integrated notebooks, public datasets, and most recently, model repositories, it has transformed into a comprehensive data science ecosystem. This growth encouraged a continually expanding community, making Kaggle not only a place for competition but also a hub for learning, collaboration, and innovation in data science. When Google acquired Kaggle in 2017, this development was further accelerated through greater integration with the Google ecosystem and infrastructure.

Meanwhile, the field of data science has undergone massive transformation. Machine learning and AI exploded in popularity across industry, daily life, and academic research [12, 2]. At the same time, the demand for computational resources has skyrocketed—particularly with the rise of (large) language models, which continue to grow in size and complexity [13]. For example, recent studies show a dramatic increase in AI-generated content online, with a 57,3% rise on mainstream websites and an alarming 474% rise on misinformation sites [10], alongside growing concerns around plagiarism [3, 8, 14]. This trend is not limited to language technologies—AI adoption is also accelerating across healthcare, finance, entertainment, and transportation, powered by machine learning algorithms, vision models, and time series forecasting [5].

Critically reflecting on this brings us to a central—yet often overlooked—necessity: one of the major challenges in managing a technological boom such as AI is the task of **accurately measuring and testing its capabilities**. That is, translating research and potential breakthroughs into rigorous, real-world application to answer questions such as:

How effective is AI for specific tasks? What are its boundaries—not just in theory, but in practice? And what real-world problems can it meaningfully and reliably solve?

Blindly deploying state-of-the-art AI technologies in sensitive or high-stakes domains—such as financial forecasting, education, information retrieval, or medical diagnostics—without understanding their actual capabilities can lead to unintended and potentially harmful outcomes.

To mitigate this, researchers have turned to benchmark development [15, 11], controlled experiments, and systematic evaluations across domains [16, 6]. Yet within this landscape, Kaggle stands out as a uniquely valuable case study: over the past fifteen years, it has gathered a global community that advances AI through open competitions, shared notebooks, public discussion, and collaborative experimentation. With the release of Kaggle's very own META-DATASETS—Meta Kaggle and Meta Kaggle Code—we are now equipped to analyze Kaggle itself, providing a rare opportunity to reflect on how the platform and its community have contributed to the evolution of AI and Data Science as a whole.

And so in this study, we take a data-driven look at Kaggle's development, examining how its users, forums, notebooks, and competitions have collectively helped shape the data science landscape—and how this insight can inform the future of empirically-driven AI development. We follow a general research paper structure, but do not fully adhere to its conventions. Throughout this work, we frequently cite non-traditional sources—such as Kaggle notebooks (often enough created by us for this specific purpose, and cited like: [1]) or other forms of evidence—to support our claims.

It is also important to note that this work was conducted within the scope of the Meta Kaggle Hackathon, under a limited time and resource frame. As such, we encourage readers to explore the cited resources directly to validate, extend, or build upon our findings and to acknowledge, that this is not a peer-reviewed scientific paper.

2 Kaggle User Growth

TL;DR

We've analyzed the growth of users in Kaggle over the years and have identified spikes in the daily counts in the registration of users. By identifying the spikes in new users registering, and not just the organic growth over time, we can gauge how influential Kaggle is in the datascience space and how effective is its democratization through associated events, competitions, and collaborations with other platforms.

Kaggle was launched in April 2010 as a small platform hosting data science competitions, aiming to connect organizations with skilled individuals to address real-world problems and promote the democratization of data science. Over time, it evolved into a global community where users, regardless of background, can access datasets, share code, and learn from one another. Today, Kaggle plays a significant role in making data science more open and accessible by supporting skill development, career growth, and collaborative problem-solving. The platform also provides free access to high-performance tools such as GPUs, enabling users to run advanced models and experiments with ease.

By examining spikes in new user registrations, beyond the platform's steady organic growth, we can assess Kaggle's influence within the data science community and evaluate how effectively it promotes democratization through key events, competitions, and partnerships with other platforms.

Some questions that we would like to answer with the analysis.

1. **When did key Kaggle membership milestones occur (e.g., 1M, 5M, 10M members)?**
2. **What specific dates show significant spikes in user registrations?**
3. **Which events, both within Kaggle and externally, contributed to these spikes?**
4. **How can the impact of these events be quantified by analyzing the number of users who registered during the associated periods?**

2.1 Early Years (2010-2015)

Kaggle began in April 2010 as a platform for data science competitions. It started small but quickly gained attention by allowing data science and machine learning practitioners to have a platform to discuss real-world problems, exchange ideas, and to learn from each other. Figure 1 shows that from 2010 to 2015, user registration remained relatively modest, with occasional spikes corresponding to competitions or media exposure. During this period, Kaggle was still building features, but already attracting attention from academia and early adopters. In general, this phase was defined by gradual organic growth driven by competition-based participation.

2.2 Expansion and Pre-Pandemic Acceleration (2015-2020)

Figure 2 shows that between 2015 and 2020, daily user registration began increasing more sharply during this period. A major milestone was the registration of the **one-millionth user in June 2017**, marking rapid adoption by learners, educators, and practitioners. Spikes in user registration during this phase indicate moments of high activity, likely tied to popular competitions, educational partnerships, and the launch of **Kaggle Learn**. By late 2019, daily user growth had significantly increased, as Kaggle slowly became mainstream, and offer free tools for data science education and collaboration.

2.3 COVID Pandemic era to Present (2020-present)

Last March 2020, the **COVID-19 pandemic was announced by the World Health Organization**. During the pandemic (highlighted in green on the graph), there was a surge in new user registrations. In a short span of just three years, from Mar 2020 to May 2023, the user count tripled from 4.2M to 13.3M. During this period, people around the world were likely stuck at home, looking for online learning opportunities and ways to upskill. Kaggle, being a platform for learning data science and machine learning, became a natural choice for these people. This COVID-era boom pushed Kaggle past 13M registered users, and even after the pandemic officially ended last May 2023, the growth trend remained strong.

Following the COVID-19 pandemic, Kaggle sustained its strong organic user growth. Starting in 2023, key events, such as the release of Gemma models in Kaggle, various newly introduced Large Language Model competitions, and hosting of Google's Gen AI intensive courses, coincided with spikes in daily user registration. With Kaggle's current growth, it

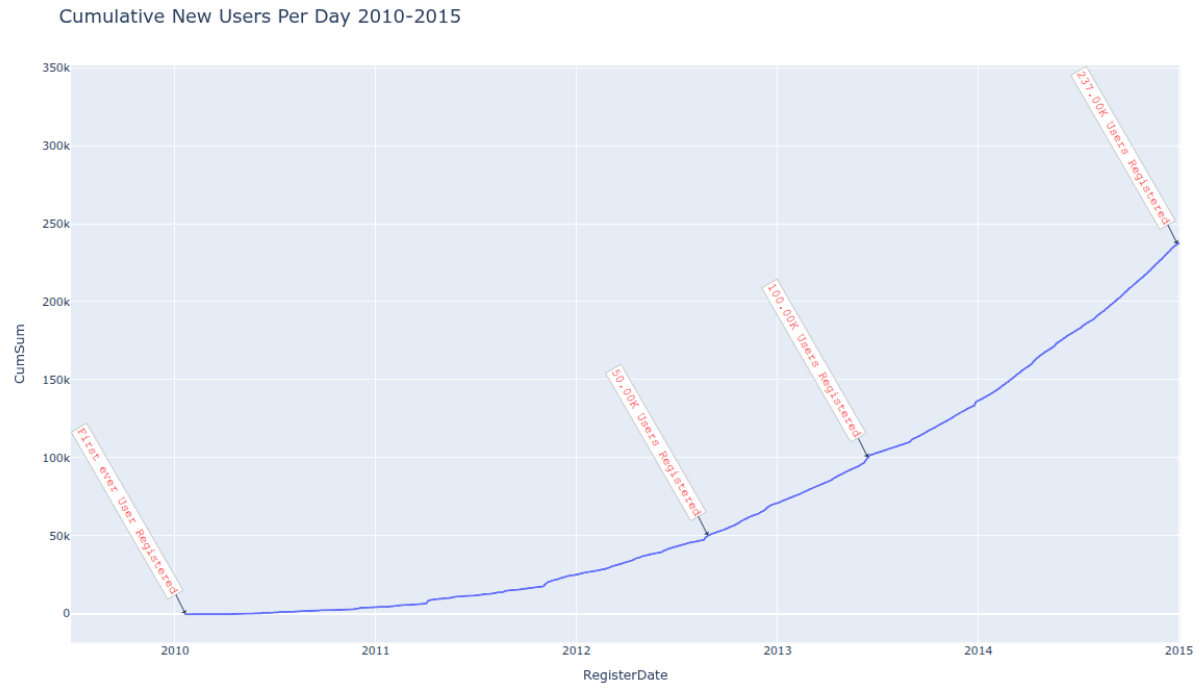


Figure 1: Kaggle Cumulative Growth 2010-2015 [3]

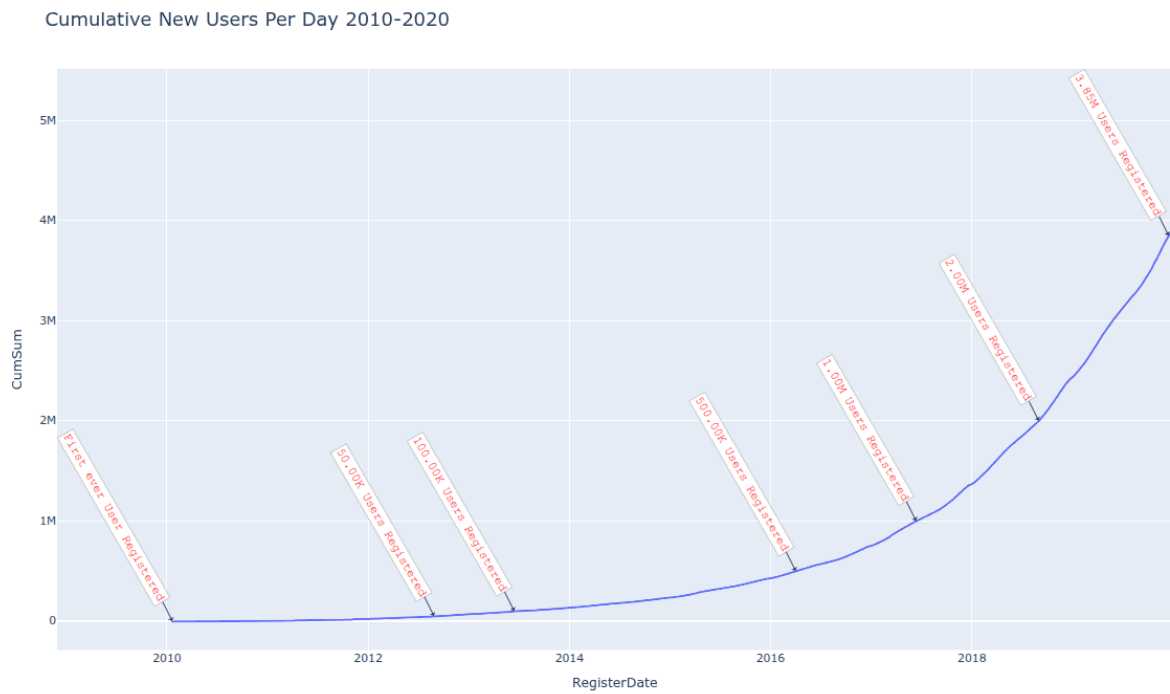


Figure 2: Kaggle Cumulative Growth 2010-2020 [5]

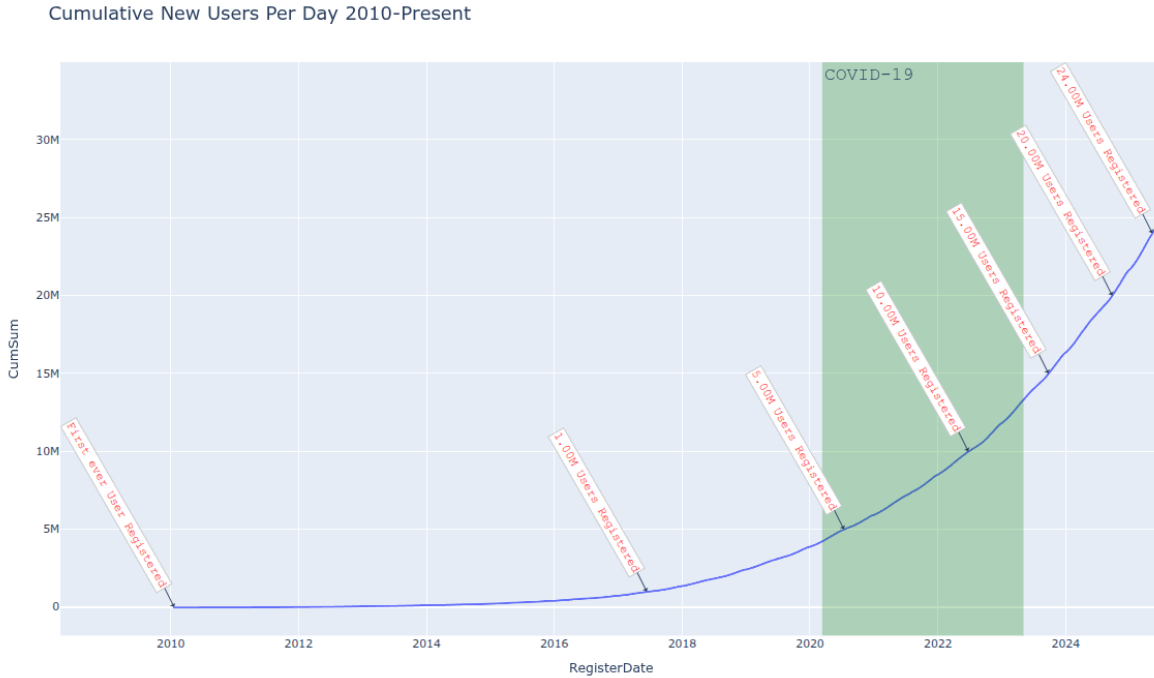


Figure 3: Kaggle Cumulative Growth 2010- Present [7]

is expected that by Q2 2026, Kaggle will reach **30 million registered users**, highlighting its continued relevance as a go-to platform for AI and data science innovation in the post-COVID digital landscape.

Figure 3 shows the COVID pandemic era and post-pandemic daily user registration counts of Kaggle.

2.4 Anomaly Detection Using Sliding Window Z-test

Anomaly detection using a sliding window Z-test is a statistically valid technique for identifying unusual spikes or drops in time series data, particularly in contexts like daily user registrations on platforms such as Kaggle. This method works by continuously analyzing recent historical data within a fixed-size window (we used 90 days), calculating the mean and standard deviation, and then assessing whether the next data point deviates significantly from the expected range using a Z-score. The formula can be seen below.

$$Z_{score} = \frac{X_{RegisteredUsersOnObservationDate} - \mu_{RegisteredUsersNinetyDaysSlidingWindow}}{\sigma_{RegisteredUsersNinetyDaysSlidingWindow}} \quad (1)$$

If a registration count falls far outside the typical variation observed in the sliding window, it is flagged as an anomaly, indicating potential external events, campaigns, or platform changes. This approach is especially relevant to Kaggle, where user activity can exhibit sharp, event-driven spikes in user registration due to competitions, events, and collaboration(s) with other platforms such as Coursera.

2.5 Identified Events and its effects on User Registration

Although Kaggle's overall growth has been largely organic, Figure 4 reveals notable spikes in daily user registrations. We've investigated and extensively looked into the MetaKaggle dataset, with the aid of Google search and ChatGPT, and validated the events that we've associated these spikes with. The dataset can be found here. We have associated events that coincided with these dates if an event was held within +/- 2 days of these spikes in user registration.

Having calculated the Z-scores of the daily registration counts (see Table 1), we can see multiple dates that have a Z-score of more than 2.5, suggesting that the effect of these events within and outside Kaggle was not due to random chance.

A higher Z-score indicates a day with significantly more registrations than typical. In Figure 5, the mean **Z-score: 5.7787** reflects the overall intensity of these registration surges, suggesting that, on average, these events deviate

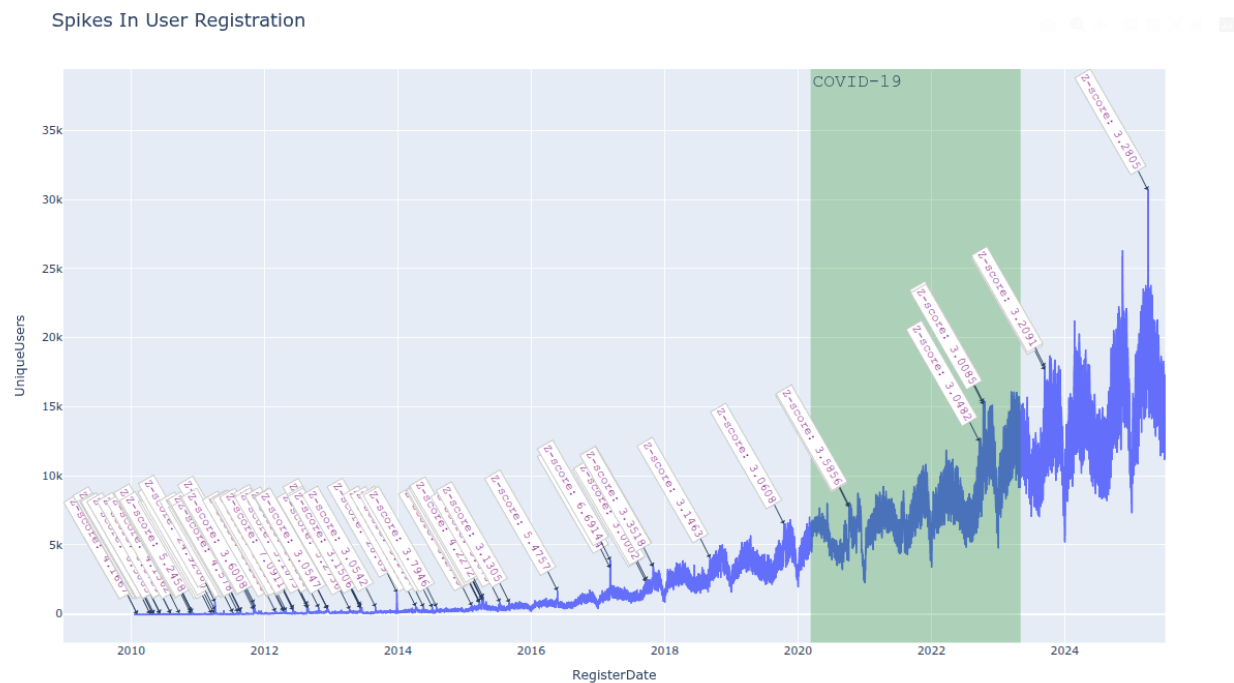


Figure 4: Spikes in Daily User Registration [9]

Table 1: Kaggle events with the highest Z-scores

Rank	Date	Event	Registered Users	90-day Rolling Window Daily Average	Z-score
1	2013-12-26	Packing Santa's Sleigh	1578	192.177778	28.6900
2	2011-04-05	Heritage Health Prize	889	30.133333	24.9206
3	2011-11-04	Kaggle Raises \$11M in Series A Funding	803	51.977778	15.5391
4	2012-08-22	Predict Closed Questions on Stack Overflow	670	103.066667	8.3101
5	2017-03-08	Kaggle is joining Google Cloud!	3353	1215.111111	6.5344
6	2014-04-08	Johns Hopkins Specialization in Data Science Through Coursera	580	245.833333	4.9852
7	2015-04-08	March Machine Learning Mania 2015	984	469.977778	3.3669
8	2022-10-11	Google Universal Image Embedding	15361	8473.200000	3.3053
9	2025-03-31	5-Day Gen AI Intensive Course with Google (Mar 2025)	30701	17176.144444	3.2805
10	2010-11-23	RTA Freeway Travel Time Prediction	43	12.888889	3.1082
11	2024-02-22	Fine-tune Gemma models in Keras using LoRA	21223	12483.444444	2.8890
12	2024-11-11	5-Day Gen AI Intensive Course with Google (Nov 2024)	26313	15929.644444	2.7808
13	2020-11-23	Jane Street Market Prediction	8741	5793.444444	2.4862
14	2023-10-10	AI Village Capture the Flag @ DEFCON31	18685	12046.111111	2.3507
15	2018-04-10	2018 Data Science Bowl	3875	2760.411111	2.0961
16	2021-07-31	30 Days of ML	8915	6705.400000	2.0519
17	2019-11-19	Link Prediction Data Challenge 2019	6823	4620.333333	2.0212
18	2016-11-14	The Nature Conservancy Fisheries Monitoring	1574	1067.311111	2.0017
19	2018-11-12	Inclusive Images Challenge	5027	3453.133333	1.9391
20	2016-05-24	Can we predict voting outcomes?	1197	849.455556	1.8972

Let's get the mean and median Z-scores of these Kaggle events

We can see below how far these Kaggle events deviate from the typical registration day

```
print(f"Average Z-score: {round(df_kaggle_events['ZScore'].mean(),4)}")
print(f"Median Z-score: {round(df_kaggle_events['ZScore'].median(),4)}")
```

Average Z-score: 5.7787

Median Z-score: 2.8349

Figure 5: Mean and Median Z-scores [11]

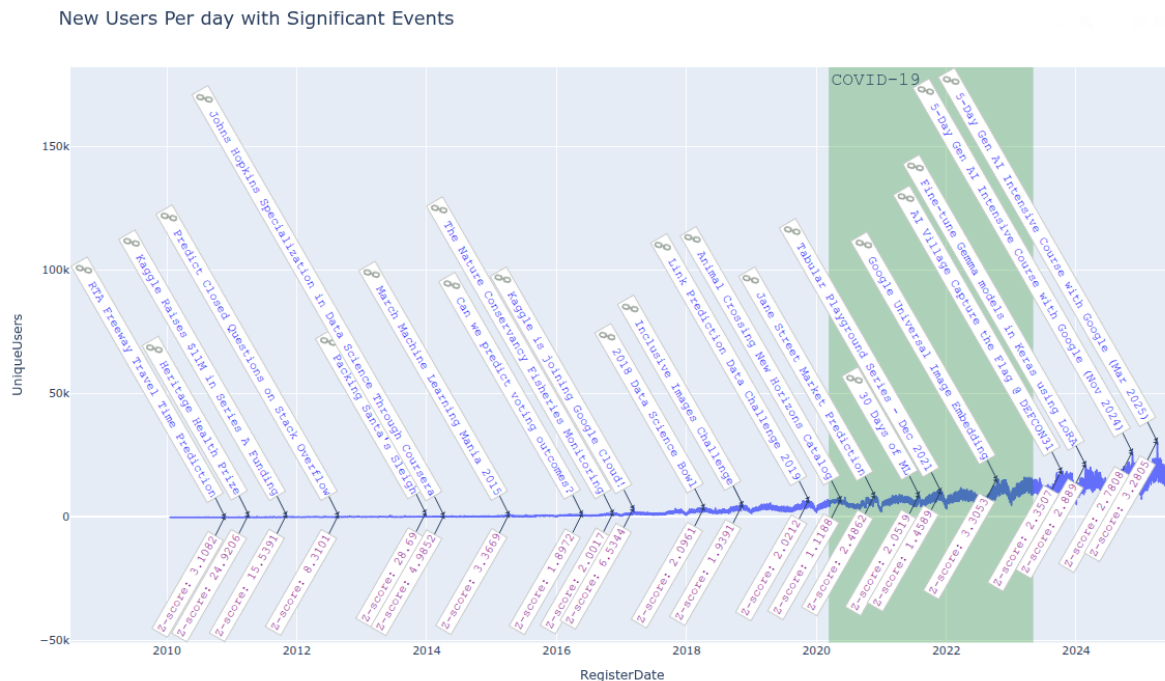


Figure 6: Registration Spikes, with Events and Z-scores [13]

substantially from normal activity. The median **Z-score: 2.8349**, being slightly lower, implies that while some events are extreme outliers, the majority are more moderately above the norm.

In Figure 6 we've super-imposed the Z-values against the spikes in registration over time would yield the next graph, and with this graph, we can now clearly see the effect of these events on registrations.

2.6 Conclusion

Using simple statistical checks, whenever there is an official Kaggle event or collaboration with another platform, there were significant increases in new user sign-ups on Kaggle. We can also quantify the effect and reach of Kaggle during these events through counts in new users! These events are often linked to competitions, online courses, or collaborations with schools and companies, **with the 5-Day Gen AI Intensive Course with Google attracting the most number of new users in single day with 30K users!** Figure7.

Since Kaggle's mission includes spreading data science knowledge to a wider audience, it's not surprising that such events draw more people to the platform. This pattern shows that the public responds well to opportunities to learn or take part in challenges. The spikes in registrations are a clear sign that Kaggle events help attract and involve more users, helping to make data science more accessible to everyone.

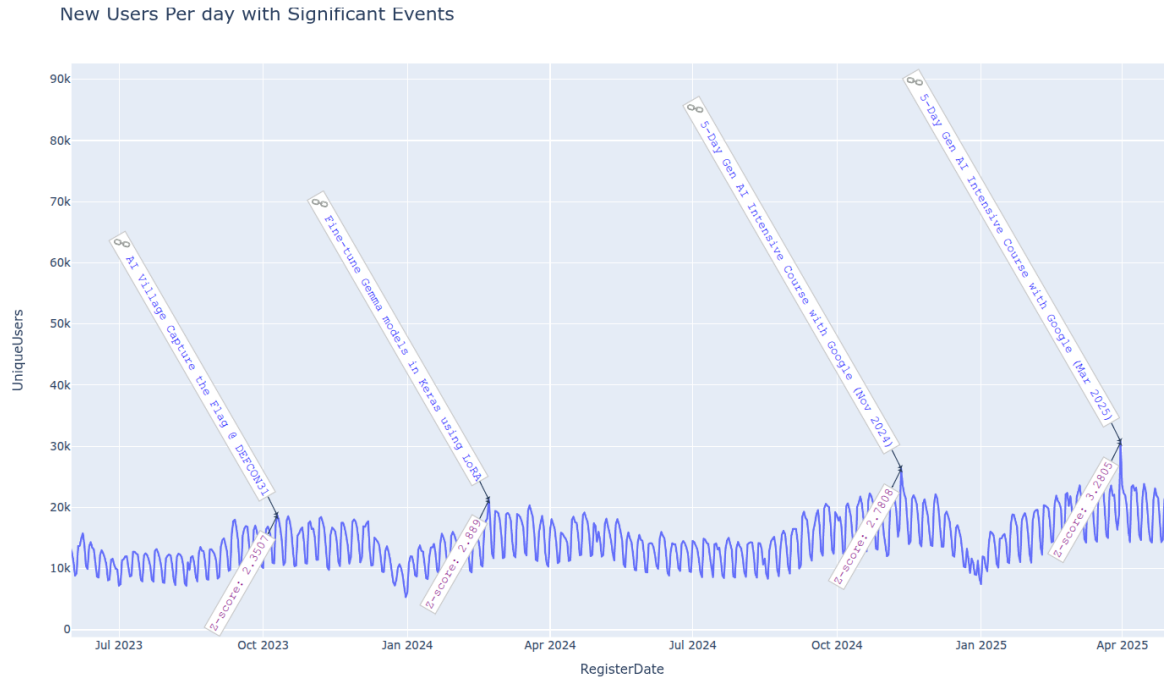


Figure 7: 5-Day Gen AI Intensive Course with Google (Mar 2025) [15]

We also saw a clear rise in new Kaggle users during the COVID-19 period, when many people were staying at home. With more free time and a need to stay productive, many turned to learning new skills. Kaggle became a go-to place for this. Kaggle offered not just the tools and data, but also the community and the platform for people to grow their data science knowledge and connect with other practitioners around the world.

These findings highlight how far Kaggle's influence has grown within the global data science community. These insights underscore Kaggle's significant role and evolving influence in the data science domain. In its early years, Kaggle began primarily as a competition platform, over time, it has grown into a comprehensive platform that supports not only competitions but also learning, collaboration, and career development. Its ability to engage users, whether through partnerships, educational programs, or timely challenges, demonstrates its effectiveness in making data science more approachable and inclusive. Its continued growth reflects not just its technical offerings, but also its success in democratizing access to data science and empowering individuals to advance in this rapidly evolving field.

3 Investigating Meta Kaggle Code

TL;DR

Kaggle’s kernels are predominantly written in Python, with occasional usage of R. We investigate package imports and method calls, and find a diverse range of packages and technologies being used, both in out-of-competition and competition kernels. The entropy and overall diversity of these technologies have been increasing year by year. Specifically in competitions, we observe a rapid adaptation to new technologies, some of which even dominate the winners’ write-ups.

The META KAGGLE CODE dataset provides an extensive archive of publicly shared notebooks (*kernels*) created by users on the Kaggle platform. It contains approximately 5 900 000 notebooks, each accompanied by metadata such as author information, timestamps, competition status, and the full source code. This dataset covers a time span from 2015 to the present, amounting to roughly 300 gigabytes of data. Its size and diversity make it a valuable resource for understanding trends in data science practice, particularly in the context of real-world, user-generated content. Unlike many benchmark datasets that focus solely on outputs or results, Meta Kaggle Code enables the analysis of actual codebases at scale - something that is often neglected. This opens the door to exploring how data scientists structure their workflows, which tools and methods they prefer, and how coding behavior evolves over time.

So in this section, we take it further than a regular metadata analysis and examine the code itself. We do this by parsing import statements, method calls, and library usage patterns; we identify technological trends and behavioral patterns in the Kaggle community. We distinguish between code written for competitions and that created for general experimentation or sharing, enabling us to compare these two modes of engagement. This approach allows us to address several key research questions:

1. **Which packages and libraries are most frequently used on Kaggle?** Are there dominant tools, and how do their usage patterns evolve over time?
2. **Which technologies are most prominent in competitions?** Do certain frameworks or methods correlate with higher performance or winning submissions?
3. **How diverse is the Kaggle ecosystem in terms of programming technologies?** Can we quantify technological diversity or detect shifts toward particular toolchains?
4. **How fast do kagglers adapt?** Are they working with cutting-edge technology? If so, how effective? And consequently: which tools stand the test of time and stay?
5. **What technologies do winning Kagglers rely on?** Do winners converge around certain best practices, or do they innovate with lesser-known tools?

Each of the conducted data analyses has been made publicly available by us in the form of notebooks and source code, for your convenience and to ensure proper citation. All of the facts and insights presented in this work were generated based on data available as of June 2025. Accordingly, statements referring to the current state of the platform, such as “as of now” should be understood in the context of that time frame.

3.1 R or Python?

TL;DR

Since 2016, the use of the R programming language on Kaggle has been steadily declining, while Python has seen continuous growth. As of now, approximately 95% of all Kaggle kernels are written in Python [16].

Out of the total of 1 487 483 kernels, approximately 5% are written in R, while the remaining 95% are written in Python. The earliest kernel in the dataset dates back to 2015-03-25, and—as shown by Figure 8—Python usage has steadily increased in popularity since then.

The most upvoted R kernel belongs to a user named *chebotinaa*. Their R kernel, paradoxically created in 2021, has received a total of 728 upvotes. The user with the highest number of R kernels is *willianoliveiragibin*, with an impressive total of 332 R kernels.

In comparison, *alexisbcook* authored the Python kernel with the highest number of upvotes: 17 847. Unsurprisingly, this kernel is part of the well-known Titanic Tutorial series, serving as an introductory notebook for newcomers to Kaggle.

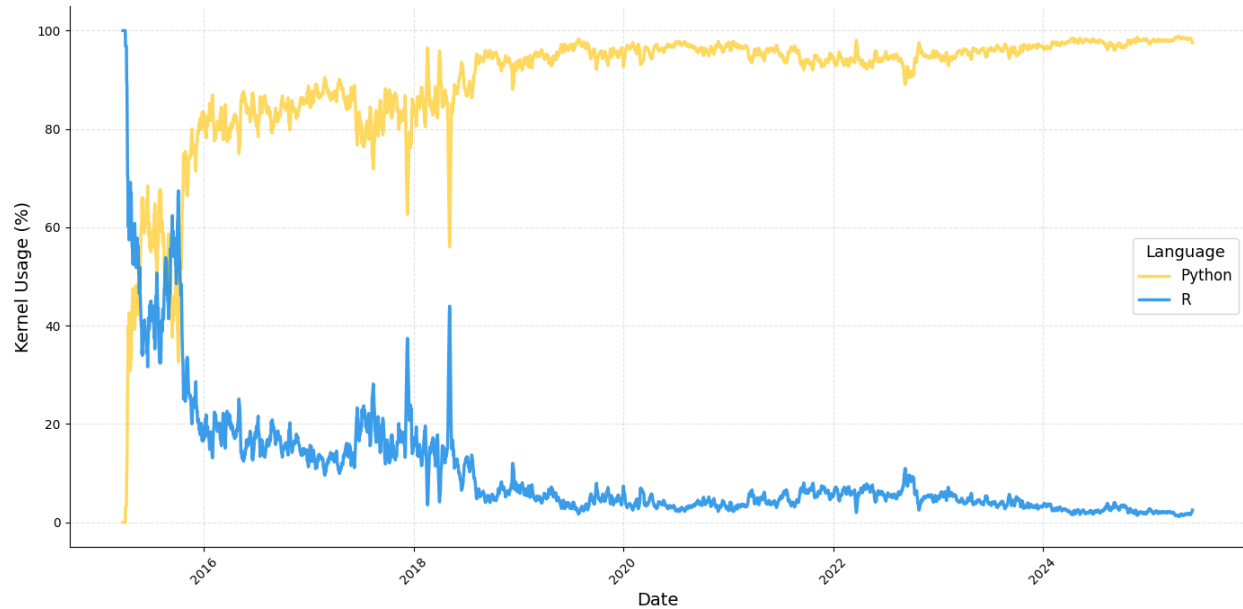


Figure 8: The development of R versus Python kernels on Kaggle reveals a clear and consistent trend toward Python since 2016.

3.2 What Packages & Methods do Kagglers use in General?

TL;DR

The most frequently imported packages on Kaggle are `pandas`, `numpy`, and `matplotlib`, particularly the `matplotlib.pyplot` module. Among R kernels, the most commonly imported package is `ggplot2`, with over 100 000 imports. These prominent packages often appear together, indicating a strong co-dependency and forming a stable foundation for many Kaggle workflows. When combined with the observed method usage distribution, we can infer how data science is practiced on Kaggle: a pragmatic, modeling-driven approach, yet with a focus on visualization and frequent basic data cleaning.

We analyzed all kernel versions available in the META CODE dataset and extracted package imports and method calls from 5 357 315 kernels by using regex commands [17]. Since this computation is resource-intensive, we make the resulting data publicly available as a dataset for your convenience [18].

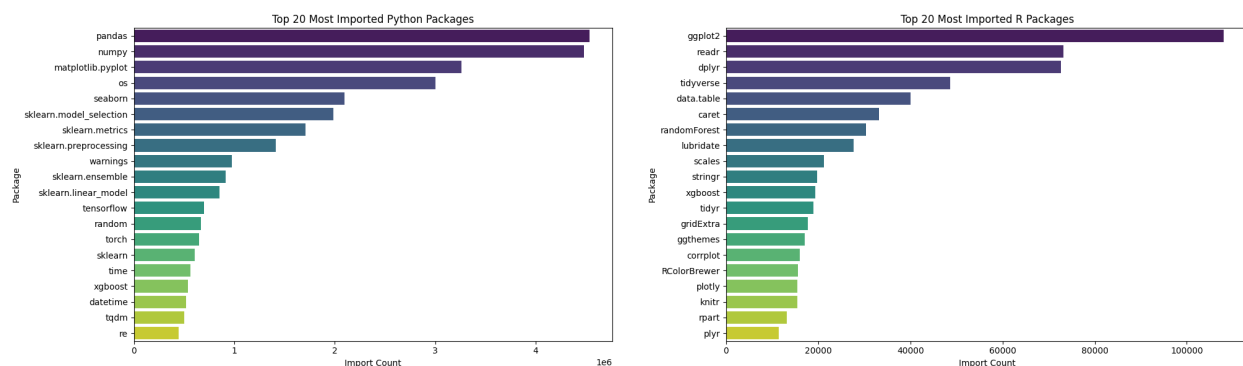


Figure 9: The top 20 overall most imported packages, left for Python and right for R [20].

When looking at Figure 9, the total number of Python package imports is nearly two orders of magnitude higher than that of R packages (as a consequence of Python's dominance on the platform). It is no surprise that `pandas` and `numpy` are the most frequently imported packages, as they are commonly included at the start of nearly every

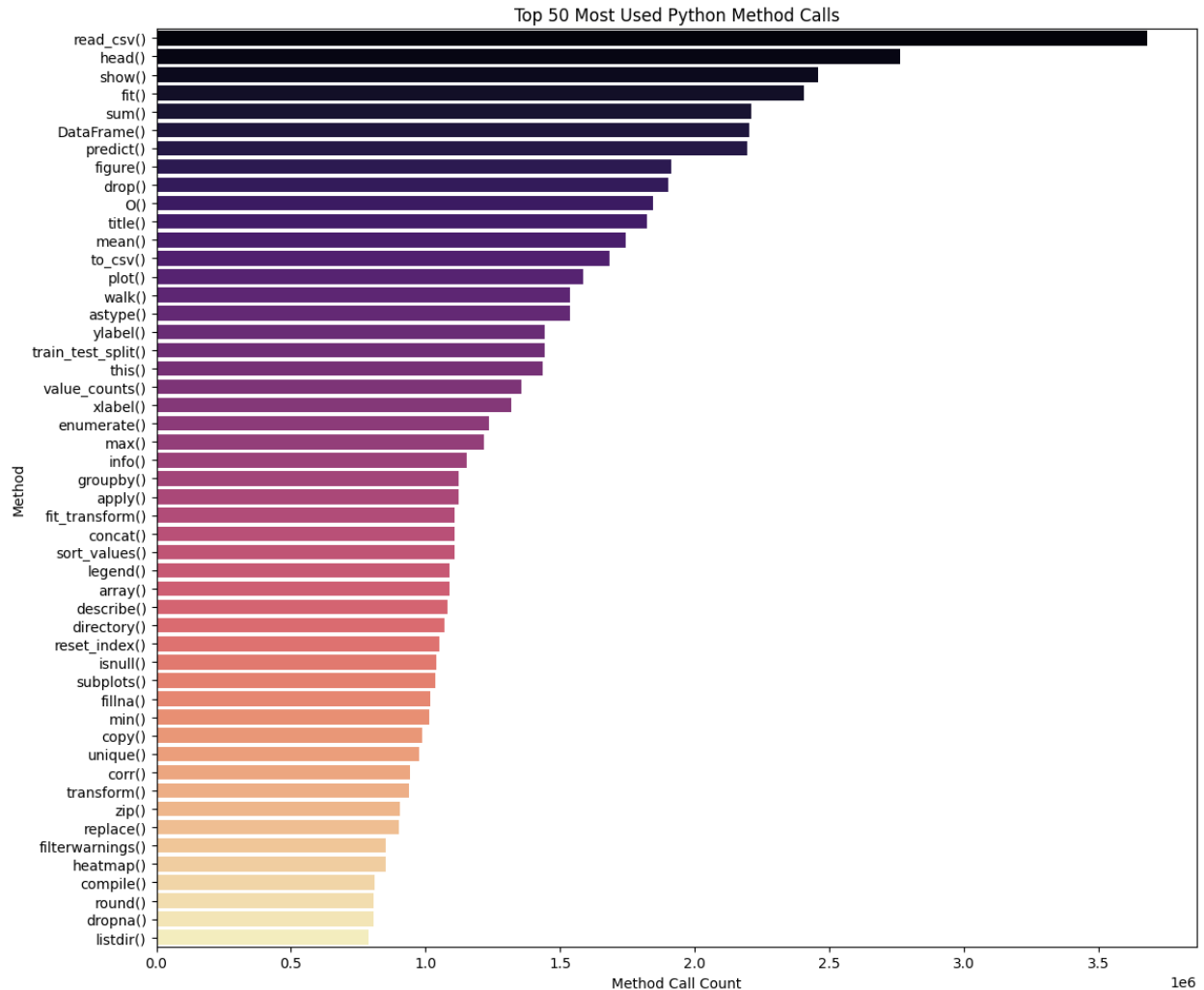


Figure 10: The top 50 most called methods in all Python Kernels on Kaggle [22].

Python kernel. Perhaps surprisingly, the third most imported package is `matplotlib.pyplot`, rather than the base namespace `matplotlib`. This is likely because most of Matplotlib’s functionality being concentrated within the `pyplot` submodule, which is typically imported using the alias `plt`. This submodule provides commonly used functions such as the widely used `plot()` method. On the R side, the most prominent packages are `ggplot2`, `readr`, and `dplyr`.

For Python kernels, we also extracted and analyzed method calls. Figure 10 shows the 50 most frequently used methods. As expected, the list is headed by `read_csv()`, the typical entry point for most data science workflows. This method is provided by `pandas`, as well as by other libraries such as `polars`. It is followed by `head()`, commonly used to preview a dataframe, and `show()`, typically used to render plots or outputs. Interestingly, model-related method calls such as `fit()` and `predict()` appear more frequently than classic preprocessing methods like `drop()`, `apply()`, or `fillna()`, suggesting a strong focus on modeling possibly even at the early stages of many kernels—maybe even at the risk of working with less clean data?

This modeling-centric trend is further supported by the prominent appearance of `train_test_split()` and `fit_transform()`, both core elements of machine learning workflows. It suggests that a significant portion of Kaggle users quickly transition from data loading to model training, possibly due to the competition-driven, results-oriented nature of the platform.

At the same time, plotting functions such as `plot()`, `figure()`, `title()`, `xlabel()`, `ylabel()`, and `legend()` are also highly represented, indicating that visualization plays a crucial role in EDA and communication of results. This

underlines our assumption that the typical Kaggle kernel is not just about modeling, but also about presenting findings in a visually interpretable way.

The inclusion of `dropna()`, `fillna()`, and `isnull()` highlights the importance of dealing with missing values, though their relatively lower rank compared to modeling methods might suggest that data cleaning is sometimes treated as secondary—or at least performed using fewer unique calls. This would align with our previous findings.

Lastly, the presence of `filterwarnings()` implies a degree of attention to notebook readability and presentation quality. Users may suppress warnings to beautify output, especially when preparing kernels for public sharing or competition submission.

Overall, the method usage distribution offers an interesting look into how data science is practiced in Python on Kaggle: **a pragmatic, modeling-driven approach, supported by visualization and basic data cleaning.**

3.2.1 Over Time

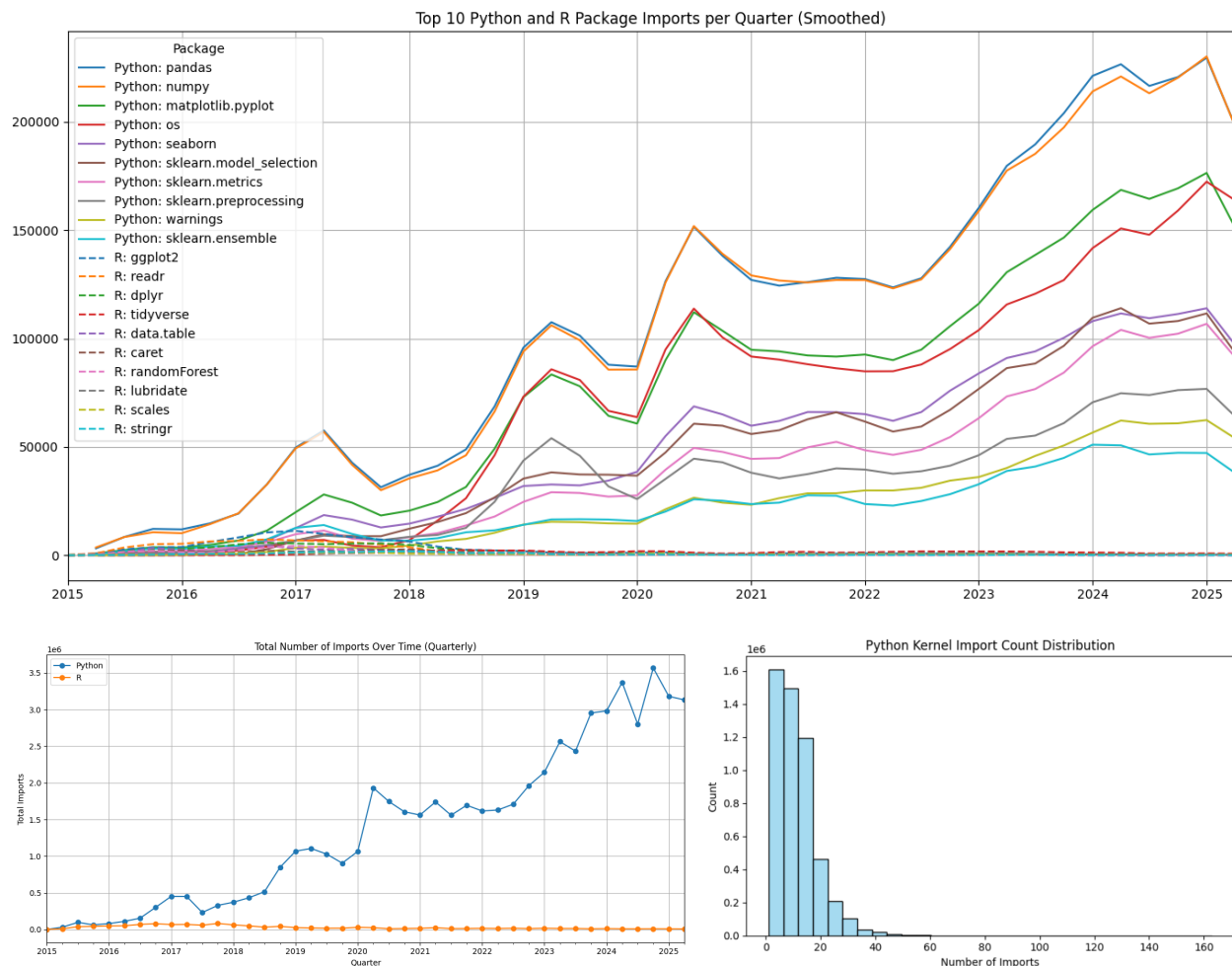


Figure 11: The top 10 most imported packages overall, for both Python and R. Below that, the total number of package imports over time is shown for both languages (left), along with the distribution of the number of imported packages per Python kernel (right) [24].

We now turn to an analysis of package usage over time, as presented in Figure 11. At first glance, the top chart reveals two key observations:

1. The most frequently used packages—particularly in Python—show strong similar usage patterns, suggesting an interdependency. This pattern likely reflects a default toolbox commonly utilized by Kaggle users and data scientists for a wide variety of tasks.

- Secondly, the total number of Python package imports increases steadily over time, while R imports gradually decline. This trend is also visible in the bottom-left plot (showcasing the total imports over time) and can be attributed to the overall rise in Kaggle’s popularity and user engagement (see Section 4), which has led to an increasing number of submitted kernels.

The bottom-right plot shows the distribution of the number of imports per kernel. Most kernels include approximately 2 to 5 imports, with the frequency gradually decreasing in a roughly Gaussian fashion toward higher counts, reaching up to 40 or more imports. This distribution suggests that Kaggle users typically rely on a minimal set of tools when writing kernels, often importing only the essential libraries required for data loading, manipulation, and visualization. The tapering tail toward 40 or more imports indicates the presence of more complex workflows, although these remain relatively rare. Overall, this indicates a tendency toward concise and narrowly scoped kernels, which supports simplicity, ease of understanding, and reproducibility—especially when kernels are forked or reused by other users.

3.2.2 By Category

To obtain a more fine-grained view of the broader range of packages used on Kaggle, we categorized all packages into four groups: **Visualization**, **Training**, **Data Science**, and **ML/LLM Models**. To construct these categories, we compiled lists of commonly used packages for each category by consulting various online sources (e.g., Medium articles, Kaggle discussions) and by querying large language models such as ChatGPT and Gemini to further expand the lists [25]. For each category, we assembled a set of 50 representative packages and then assigned package imports from Kaggle kernels to the appropriate category based on these lists, as outlined by Figure 12.

The x-axis of all four charts uses a logarithmic scale to represent the total number of imports across all kernels. As such, bars that appear to be of similar length may still differ by several orders of magnitude in actual import frequency.

For **visualization**, `seaborn` seemingly leads the way, followed closely by `matplotlib` and `plotly`—the core plotting tools in the Python data science stack. However, this instance is somewhat skewed: since we consider only base imports, it excludes submodules such as `matplotlib.pyplot`, which—as shown in Figure 9—is used more frequently than `seaborn`. This should be kept in mind when interpreting the results. Nevertheless, each can serve a distinct role: `matplotlib` for low-level control, `seaborn` for statistical plots, and `plotly` for interactive visuals.

Other Python libraries like `bokeh`, `altair`, and `holoviews` appear less often, likely due to being less known or specialized use cases. R packages such as `ggplot2`, `lattice`, and `shiny` are present but much less common, as to be expected given Python’s dominance.

Geospatial libraries—including `folium`, `leaflet`, `geopandas`, `cartopy`, and `mapview`—show up in the long tail, mainly tied to location-based tasks. Packages like `missingno` and `heatmaply` highlight how visualization also supports data quality checks and exploration. This might help explain why traditional cleaning methods appear less frequently—many preprocessing steps happen through such specialized tools with custom method calls.

For **training**, the top spots go to the familiar trio: `xgboost`, `catboost`, and `lightgbm`, probably favored by kagglers for their speed, accuracy, and ease of use with tabular data. They’re followed by `sklearn` for classical ML and `tensorflow`, which—along with `keras` and `pytorch`—has long been a core deep learning framework. The strong presence of both `keras` and `pytorch` shows that while gradient boosting is still dominant, neural networks are just as popular, especially for image, text, or time series tasks. Newer tools like `jax`, `fastai`, and `torchvision` are gaining ground, though they still trail the main libraries.

In the mid-range, R packages like `caret`, `mlr3`, and `randomForest` appear alongside Python tools such as `h2o`, `spark`, and `tidymodels`—suggesting the obvious: Python clearly leads in model training.

A long tail of specialized tools—including `optuna`, `ray.tune`, `hyperopt`, and `flaml`—points to growing interest in AutoML and hyperparameter tuning, especially in competitive settings (see Section 3.3).

For **data science**, the top libraries once again reflect the core Python stack: `pandas` and `numpy` remain essential for data manipulation and numerical computing, with the addition of `scipy`, providing fundamental algorithms. The strong showing of `polars`, a faster alternative to `pandas`, suggests more interest in high-performance tools, possibly due to the rise of larger datasets. Further down, libraries like `dask`, `statsmodels`, and `pyspark` point to the need for scalable and statistical workflows. On the R side, `data.table`, `tidyr`, `dplyr`, and `readr` reflect continued use of tidy data principles.

Many mid-ranked packages are tied to natural language processing—including `nltk`, `text2vec`, `textblob`, `spacy`, `langdetect`, and `wordcloud`—highlighting the trend of text analysis. The presence of `re` and `regex` underlines the ongoing role of regular expressions in preprocessing. Tools like `janitor`, `skmr`, `forcats`, and `reshape2` stress the importance of data cleaning, while packages such as `fuzzywuzzy` and `pandarallel` address string matching

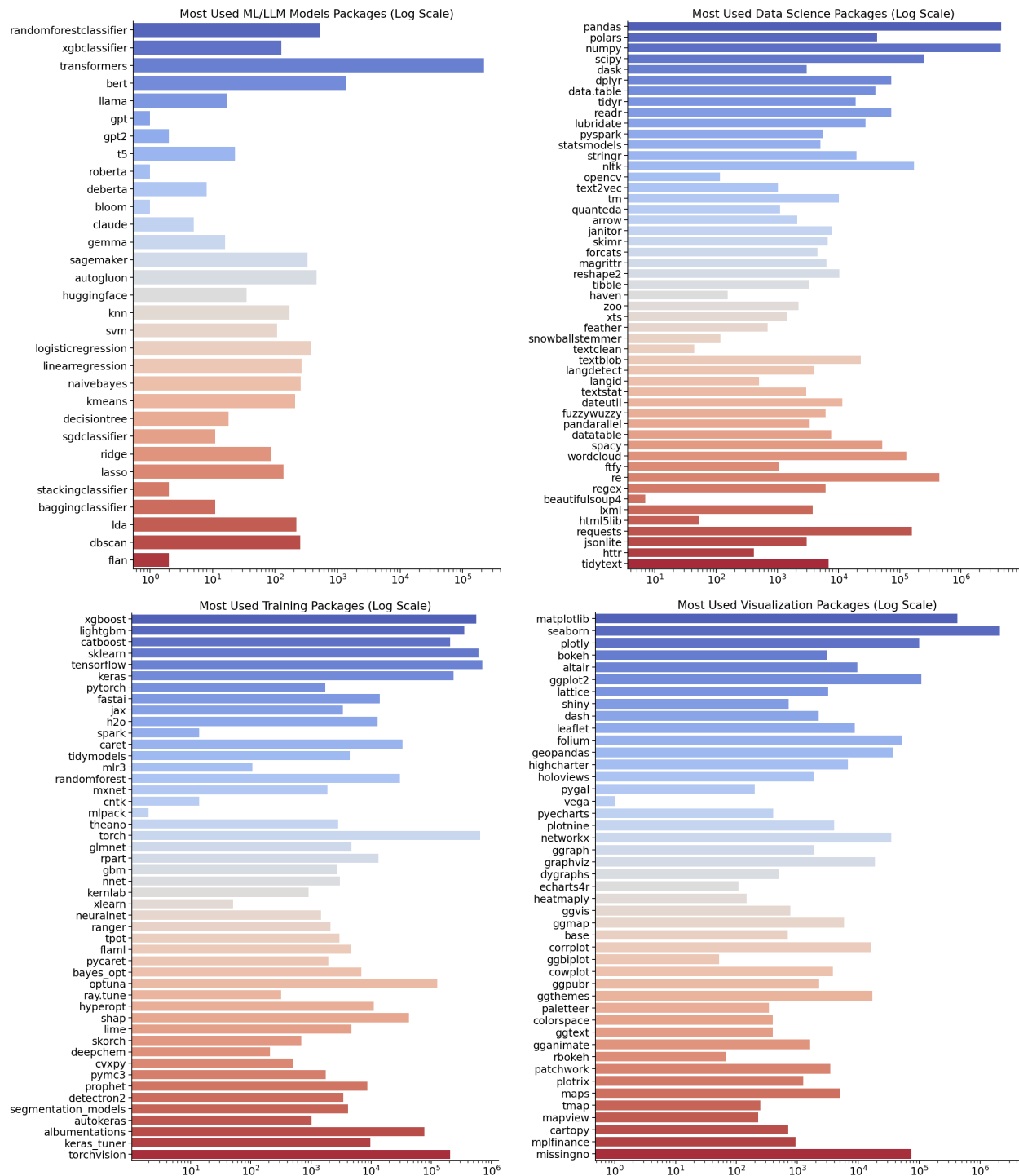


Figure 12: Distributions of imported packages across four categories in Kaggle kernels: Visualization, Training, Data Science, and ML/LLM Models. The x-axis represents the total imports, displayed on a logarithmic scale [27].

and parallel processing. Finally, libraries like `html5lib`, `beautifulsoup4`, `requests`, and `httr` suggest that web scraping and API-based data collection are somewhat common in Kaggle workflows.

For **ML–LLM Models**, we see a blend of classic machine learning and modern transformer-based approaches. Leading the chart by a lot (remember the log scale) is `transformers`, the go-to Hugging Face library for working with pretrained language models—showcasing the growing role of LLMs in applied workflows, even within Kaggle notebooks, which was to be expected. Close behind are `bert` and `xgboostclassifier`, representing two key paradigms: transformers for text-heavy tasks, and gradient boosting for structured data. `randomforestclassifier` also ranks high, continuing the popularity of ensemble methods in traditional ML.

There is also growing interest in newer LLMs such as `llama`, `t5`, `gpt2`, and `deberta`. Mentions of models like `gemma`, `bloom`, `flan`, and `claude` suggest that open-weight models are being adopted soon after their release, although their usage remains limited—likely due to hardware or platform constraints, but also because of their novelty in comparison to other technologies.

Classic `scikit-learn` models such as `logisticregression`, `svm`, `knn`, and `naivebayes` seem to remain steady, possibly valued for their simplicity and interpretability. Tree-based models like `decisiontree`, `baggingclassifier`, and `stackingclassifier` also appear, though they’re less prominent than ensembles like `xgboostclassifier`. The presence of tools like `autogluon`, `huggingface`, and `sagemaker` points to a shift toward more automated or cloud-friendly workflows that streamline model development and deployment.

3.2.3 torch vs tensorflow

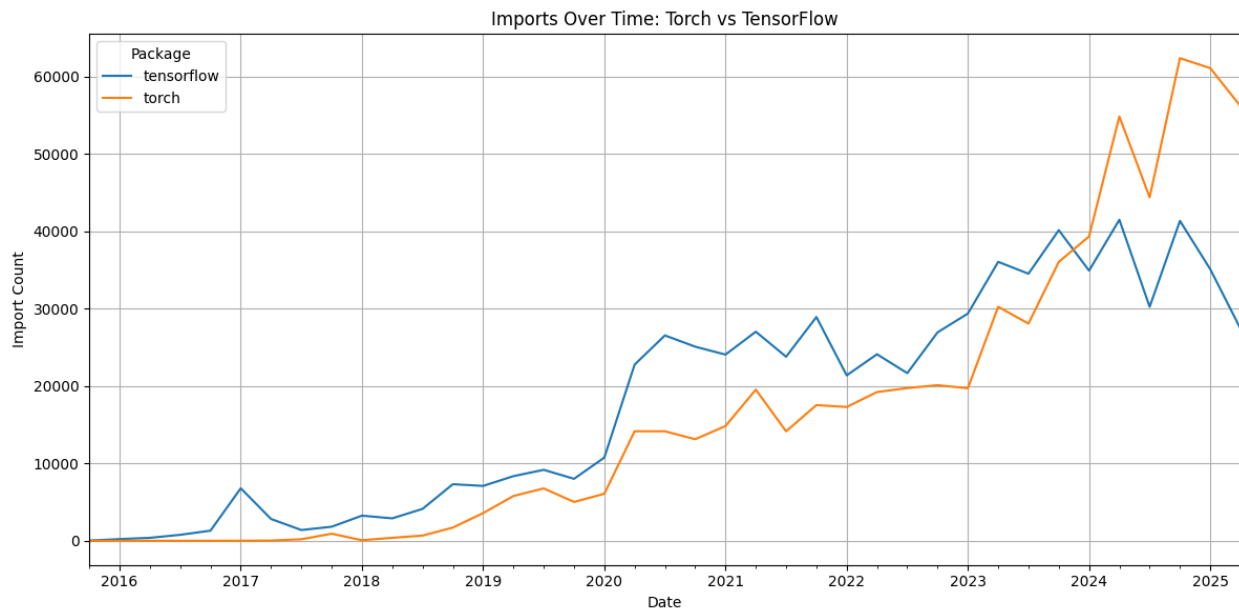


Figure 13: torch vs tensorflow over time. We can see a rise in torch users [29].

The difference in usage between `torch` and `tensorflow` caught our attention, as the latter appears to dominate the former in overall import counts. However, based on our own experience and broader trends within the machine learning community, we have observed a strong shift toward `torch` in recent years. When plotting the usage of both libraries over time, this intuition is confirmed: `torch` has been overtaking `tensorflow` since late 2023 (see Figure 13).

3.3 Competition Trends & Insights

TL;DR

First, we examine the evolution of imported packages in competition kernels: `xgboost` dominated early years, later replaced by `lightgbm`, `tensorflow`, and `transformers`, reflecting a broader community shift from classical ML to deep learning and NLP. Packages like `sklearn` remain foundational, while newer entries such as `autogluon`, `optuna`, and `openai` indicate growing interest in AutoML and generative models. Explainability libraries (`shap`, `lime`) are also increasingly common.

Secondly, we investigate discrepancies between public and private leaderboard scores, outlining consistent gaps that highlight the risks of leaderboard overfitting. Using visualizations and metric-based analyses, we show how real-world generalizability can differ massively from public leaderboard performance—as seen in competitions like *Optiver* and *M5 Forecasting*. However, our analysis also shows that, on average, Kagglers produce only a 5–10% discrepancy between private and public leaderboard scores.

Since 2010, Kaggle has been hosting data science competitions that offer medals and prize money for the best solutions. To truly participate in a regular competition, a Kaggle must submit a kernel that produces a specified output, which is then evaluated against the competition’s defined metric. Thanks to the KAGGLE META CODE dataset, we have access to these competition kernels from 2015 onward, along with their method calls and imported packages (which we extracted in a previous Section [30]), but also meta data such as public and private leaderboard scores.

3.3.1 What Packages are Kagglers using in Competitions?

As a first step, we analyze the evolution of imported packages in competition kernels over time [31]. To do this, we first normalize all sub-imports to their base namespace (e.g., `xgboost.core` becomes `xgboost`). We then filter out non-essential packages—that is, packages not directly related to model building, prediction, or competitive data science [32] (e.g., `os`, `warnings`) and plot the results. Figure 14 presents each major technology and its usage in competition kernels over time, visualized as ridge plots.

A few clear trends emerge. Notably, the usage of `xgboost` was dominant in the early years (2015–2017), reflecting the popularity of gradient boosting methods in structured data competitions during that period. However, its usage tapered off significantly after 2018, giving way to deep learning frameworks, but also similar techniques such as `lightgbm`. The plot even indicates a seamless swap from `xgboost` to `lightgbm`.

The rise of `tensorflow`, `pytorch`, and `transformers` is evident starting around 2018, with especially strong growth for `transformers` from 2020 onward. This aligns with the widespread adoption of transformer-based models (such as BERT and GPT) in both research and applied machine learning tasks. The HuggingFace ecosystem, including `datasets`, `tokenizers`, and `textattack`, shows increasing relevance, emphasizing the growing focus on NLP competitions. `sentencepiece` and `sentence-transformers` show usage spikes coinciding with the adoption of transformer models and multilingual NLP pipelines, which fits the overall transformer-based narrative.

Libraries like `scikit-learn` (`sklearn`) maintain a steady baseline, highlighting their foundational role in preprocessing, model evaluation, and classical ML methods. Interestingly, `shap` and `lime` show noticeable presence in recent years, suggesting a rising awareness around explainability in model development, even in competitive settings.

In recent years (2022–2025), newer frameworks such as `openai`, `diffusers`, and especially `autogluon` have emerged, indicating a strong shift towards generative models and AutoML pipelines. The increasing presence of `optuna` points to a growing focus on automated hyperparameter optimization. Conversely, packages like `mxnet`, `chainer`, and `h2o` have largely diminished, indicative of a transition in community preference and tool support. Similarly, `fastai` shows a peak followed by a decline.

Finally, the long tail of niche or specialized packages (e.g., `faiss`, `cv2`, `mmcv`, `darts`, and `evaluate`) highlights the variety of tools employed in specific types of competitions such as computer vision, time series, or hardware-constrained tasks.

Overall, the evolution of Kaggle competitions show a shift from tree-based methods to deep learning and transformer-centric approaches, with increasing attention to interpretability, reproducibility, and automation.

3.3.2 Public vs. Private Leaderboard Scores

In a typical Kaggle competition, both a public and a private leaderboard exist. The public leaderboard is, as the name suggests, publicly visible and displays the rankings of participants based on their submitted kernels. Each Kaggle is represented on the public leaderboard by the submission with their **highest public leaderboard** score.

What remains hidden during the competition is the private leaderboard. This leaderboard is computed using a different test set than the one used for the public leaderboard. For every submission evaluated, a corresponding **private leaderboard score** is calculated as well—but not shown. After the competition concludes, the private leaderboard is revealed, and it determines the final rankings and official winners. This means that a participant who ranks first on the public leaderboard for an entire month could theoretically end up far lower—or even last—once the private leaderboard is revealed.



Figure 14: Ridge plots illustrating packages used in competitions over time [34].

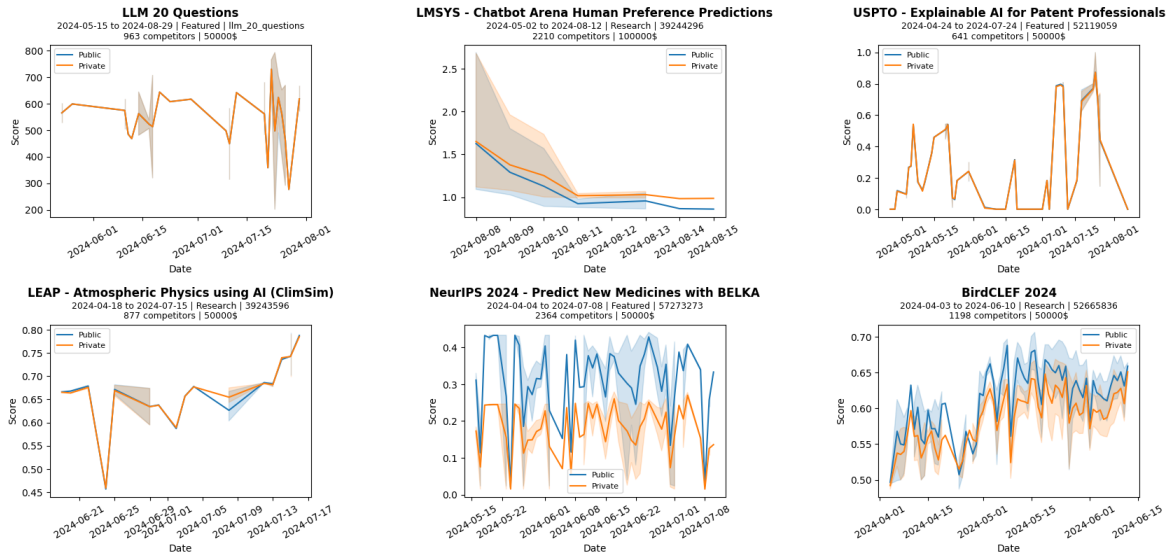


Figure 15: Recent six competitions and their private vs. public leaderboard scoring [36].

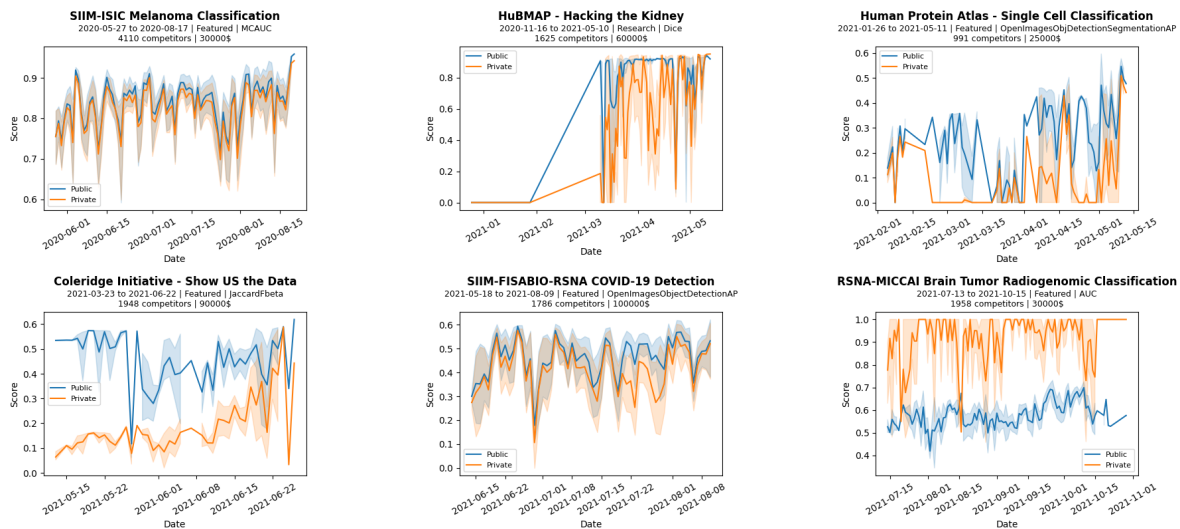


Figure 16: Competitions with biggest average private/public leaderboard score difference [38].

Kaggle implements this mechanism to mitigate overfitting to a single public leaderboard dataset. When participants fine-tune their models purely to improve public leaderboard performance, they risk overfitting the specific test set used for public evaluation. This reduces the generalizability and real-world applicability of the model. The private leaderboard thus acts as a safeguard to ensure that winning models perform well on unseen data. In this sense, the public leaderboard is merely an indication—not the final outcome. Fortunately, we have access to both the public and private leaderboard scores for all submitted competition kernels.

Figure 16 displays a selection of competitions (for more competitions, refer to [39]) along with the evolution of public and private leaderboard scores over time. The top row shows recent competitions, sorted by date, while the bottom row highlights competitions with the highest average discrepancy between public and private leaderboard scores throughout the duration of the competition. In an ideal scenario, the public and private leaderboard scores should align closely, indicating that submitted models generalize well across both evaluation sets.

Generally, public and private leaderboard scores tend to evolve in a parallel fashion over time, which is desirable, although both can exhibit considerable variance. In most cases, the two scores are not identical, and in some extreme

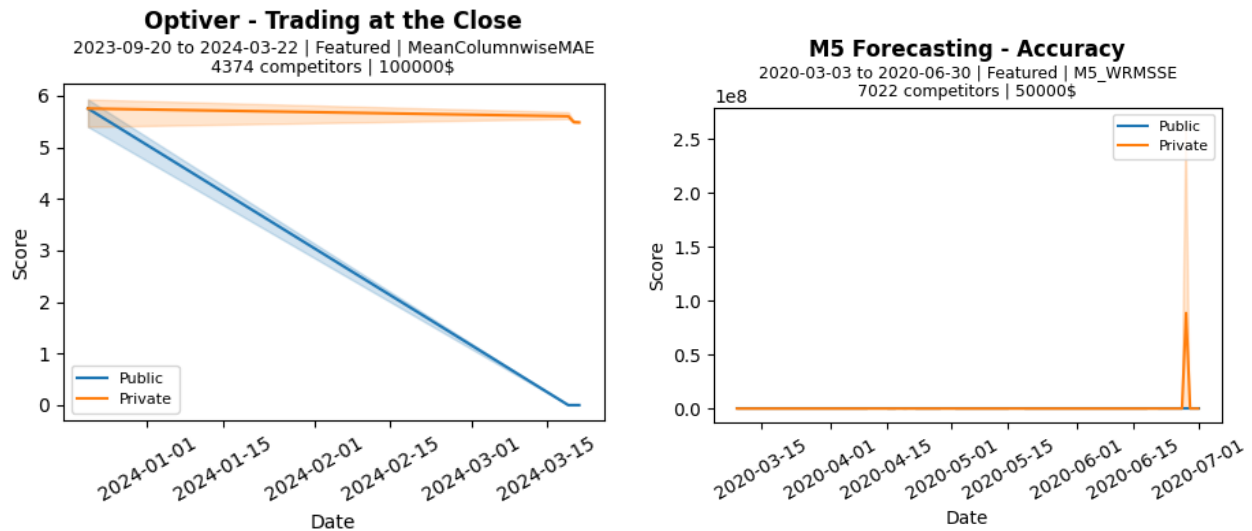


Figure 17: Two exemplary competitions, on the left the *Optiver* competition and on the right the *M5 Forecasting* competition [41].

instances—such as the *RSNA-MICCAI Brain Tumor* competition—they operate on entirely different scales, indicating substantial divergence between the public and private evaluation sets.

The plots also show numerous spikes in both score variance and average progression over time. Since the visualizations include all submissions from all teams across the full duration of each competition, such fluctuations are expected. They highlight the wide range of modeling approaches employed—some more successful than others—and show the experimental nature of many competition strategies.

The key takeaway, of course, is that optimizing for a single leaderboard—particularly the public one—does not guarantee good generalizability or robust modeling performance. These plots, and Kaggle competitions more broadly, illustrate this phenomenon clearly.

We now take a closer look at two specific competitions outlined in Figure 17: the *Optiver* and *M5 Forecasting* competitions.

The *Optiver* competition serves as a good example of the contrast between real-world modeling and controlled test scenarios. In this competition, lower scores indicate better performance according to the official evaluation metric. As shown in the plot, public leaderboard scores steadily decrease over time, approaching near-perfect values. In contrast, private leaderboard scores show more variance and only a very gradual improvement.

In the competition, the timeline was divided into two distinct phases: a training phase and a forecasting phase. During the training phase, participants were allowed to train, test, and submit their models as usual. Once this phase concluded, no further submissions were accepted. In the subsequent forecasting phase, real-world data was used to evaluate the submitted models by having them predict the closing prices of hundreds of Nasdaq-listed stocks. All results from this phase were reflected exclusively in the private leaderboard scores. As a result, the private leaderboard scores more accurately reflect the true capabilities of the models, while the public leaderboard can be easily adapted to—and overfitted—highlighting the risks of relying on static test sets and controlled evaluation scenarios.

Furthermore, the *M5 Forecasting* competition is a classic case of drastic public vs. private leaderboard shifts. Participants forecasted 28 daily Walmart item sales across two phases: a public phase with live leaderboard feedback, and a private phase revealed only after the deadline.

Many top teams tuned models to short-term trends seen in the public period, notably applying “magic multipliers” (e.g., $\times 1.02$ – 1.05) to inflate forecasts, assuming June sales would rise as in prior years. However, the expected trend broke in June 2016, causing such models to overshoot and rank poorly on the private leaderboard.

This led to a massive shake-up: some public top-100 teams plummeted, while others who avoided these tweaks rose sharply (1 000 - 5 000 places up). Robust models relying on cross-validation without magic numbers generalized better, showing that leaderboard-driven tuning can heavily backfire when trends shift. To summarize in the words of Christophe Nicault, a participant of the competition:

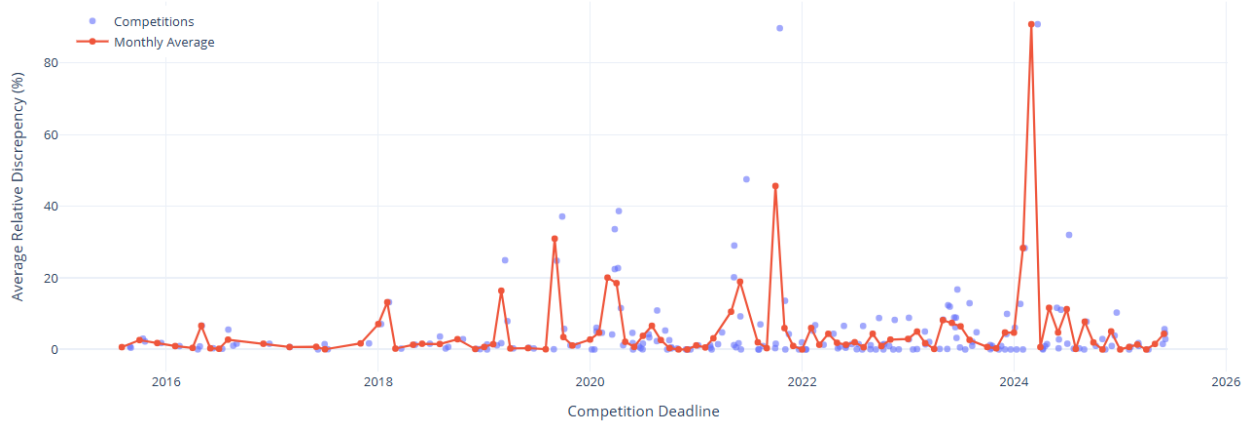


Figure 18: Showcases the average normalized discrepancy percentage between public and private leaderboard scores over time [43].

“There was a huge shake-up for this competition with different results between the public leaderboard and the private leaderboard. Part of it can be explained by the difference for this period compare to the previous years. For the 5 previous year, the sales in June have always increased compared to the sales in May. The trend was up during the previous month, and most of the model were being optimistic on the volume of sales. For the validation period, many teams used a multiplier coefficient to reflect the trend in the forecast, leading to a better score for the public data, but working only for that period.”

Average Public/Private Score Discrepancy Over Time

We are also interested in the *average normalized discrepancy percentage* between public and private leaderboards across all Kaggle competitions over time. So for each competition, we compute the average *relative change* between public and private scores for every participant, normalized by the range of public scores and outline it through Figure 18. The formula is:

$$\text{avg_rel_diff_pct} = \left(\frac{1}{N} \sum_{i=1}^N \left| \frac{\text{public}_i - \text{private}_i}{\max(\text{public}) - \min(\text{public})} \right| \right) \cdot 100 \quad (2)$$

where:

N = number of participants,
 public_i = public score of participant i ,
 private_i = private score of participant i

Each value (purple dots) reflects the *average percentage discrepancy* per competition. For example, a value of 15% means that on average, each participant’s score changed by 15% of the public leaderboard range when switching to the private leaderboard. Finally, we aggregate these per-competition discrepancy values by the month of the competition deadline, to observe how leaderboard volatility has evolved over time (red line).

The high spike observed in early 2024 is attributed to the aforementioned *Optiver* competition, which exhibited a significant discrepancy between public and private leaderboard scores.

A smaller spike, occurring at the end of 2021, is primarily due to the *RSNA-MICCAI Brain Tumor Radiogenomic Classification* competition (indicated by the purple dot at the top). In this competition, participants were tasked with predicting the genetic subtype of glioblastoma using MRI (magnetic resonance imaging) scans to detect the presence of MGMT promoter methylation. However, the challenge was hindered by the limited number of available samples—approximately 310—which resulted in leaderboard scores that were nearly indistinguishable from random submissions. As a consequence, no strong models could be developed, leading to an unusually large gap between public and private leaderboard performance.

Aside from these anomalies, the graph shows an overall average discrepancy of approximately 5–10% between private and public submission scores, which shows the importance of real-world evaluation and testing. Nevertheless, within this context, Kaggle participants generally do a commendable job of minimizing this gap.

3.4 What Technologies do (winning) Kagglers use?

TL;DR

We extract technologies from 4 419 solution write-ups using LLM-based parsing, showing that winning Kaggle solutions most frequently mention `efficientnet`, `lightgbm`, `data augmentation`, and ensemble methods. Both classical and deep learning approaches co-exist, with strong emphasis on training techniques such as `cross-validation`, `label smoothing`, and `mixup`. We also demonstrate high entropy and technological diversity in competition write-ups—both per competition and per year—with a steady increase observed over time.

After a competition concludes, Kaggle participants are encouraged to publish a *solution write-up*—a forum post in which they explain their approach, the technologies used, and any noteworthy modeling decisions. This practice is especially encouraged for top-performing teams, who are often incentivized with additional monetary rewards or recognition. This serves as another highly valuable resource for analyzing the technological landscape within the competition ecosystem. Consequently, we collected all available solution write-ups from the competition discussion sections and analyzed them [44]. We used a large language model (specifically, `gpt-4o-mini`) to extract all mentions of technologies from the full text and structure them into a list. In addition, we prompted the model to generate concise summaries of each write-up. To group and standardize different references to the same technology (e.g., `AdamW`, `AdamW Optimizer`, or `AdamW Optim`), we applied a simple fuzzy-matching algorithm to group equivalent mentions. As a result, we obtained a structured dataset consisting of 4 419 competition write-ups—primarily from top-ranking and winning teams—along with extracted technologies, techniques, and generated summaries. This dataset has been made publicly available on Kaggle [45].

Figure 19 displays the top technologies mentioned in Kaggle solution write-ups. It is important to note that the mere mention of a technology does not necessarily indicate that it was a key contributor to the team’s success. In many cases, teams also describe methods and approaches that did not perform well, although the primary focus of most write-ups is typically on what ultimately worked.

Firstly, the plot reveals a clear dominance of widely adopted machine learning frameworks and techniques. `efficientnet`, `lightgbm`, and `data augmentation` stand out as the most frequently referenced tools. This is particularly noteworthy in the case of `efficientnet`, which was only introduced by Google in 2019, yet has rapidly become a staple in many—supposedly winning—solutions. The prominence of `data augmentation` is also significant, as it reflects a focus on generating additional training data beyond the fixed public dataset. This suggests that enhancing model robustness through data augmentation remains a highly effective and commonly adopted approach in competitive settings.

Ensemble-based methods such as `xgboost`, `cnn`, and general ensemble learning also appear prominently, highlighting a preference for robust model combination approaches.

Interestingly, both classical and deep learning methods are well represented. Traditional models like logistic regression, `randomforest`, and `svm` appear alongside deep learning frameworks such as `pytorch`, `tensorflow`, and architectures like `resnet` and `bert`. This suggests that Kaggle practitioners often blend modern and traditional approaches depending on the nature of the task.

Furthermore, a variety of optimization strategies and training techniques such as the `adam optimizer`, `cross-validation`, `label smoothing`, and `mixup` are mentioned, highlighting the importance of fine-tuning and robust evaluation in achieving competitive results. Tools used for experimentation and tuning, like `optuna`, also feature, suggesting an emphasis on hyperparameter optimization.

Lower-ranked entries like `llm`, `xception`, and `gradient accumulation` indicate niche or emerging techniques that may either be less suitable for typical Kaggle problems or are still gaining traction among practitioners.

Overall, the plot highlights a wide range of technologies and approaches employed by participants, while also revealing the strong influence of Google-developed tools—particularly the Transformer architecture and `efficientnet`.

In this context, we provide two more visualizations:

1. Figures 31 and 32 within the Appendix show the most frequently mentioned technologies in competition write-ups over time, highlighting trends in tool adoption across different years.
2. Figure 20 illustrates the distribution of mentioned technologies clustered by competition and frequency.

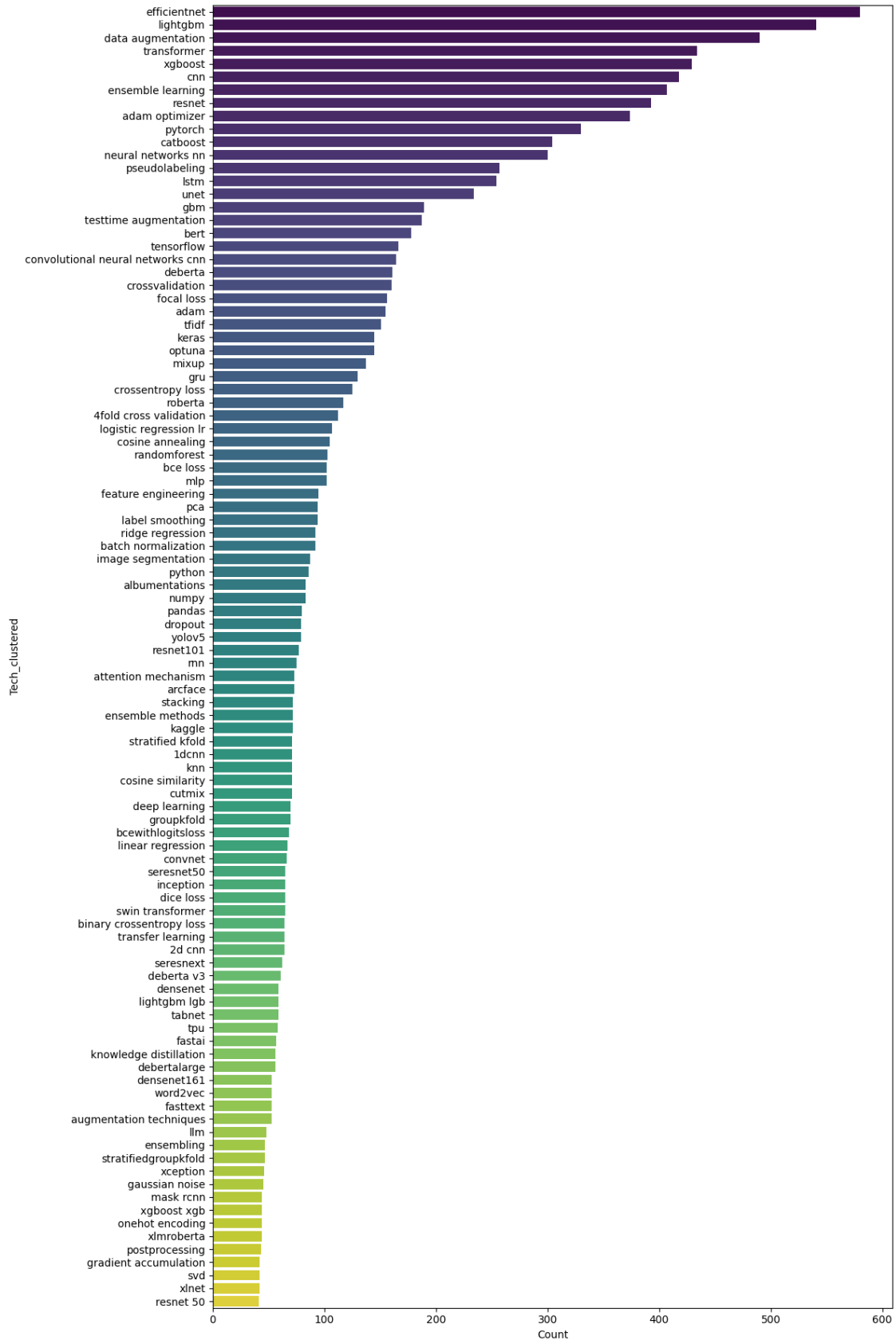


Figure 19: The top technologies mentioned in solution write-ups [47].



Figure 20: Technologies mentioned within writeups, clustered by their competition and frequency. Refer to [49] for the full interactive version.

3.4.1 Technology Diversity per Competition over Time

As we already discussed, each competition typically involves several teams or participants whose write-ups reference various technologies. Herein, the diversity of technologies mentioned can vary significantly:

- Some competitions may consistently use only a single dominant technology → **low entropy**
- Others may involve a broad and evenly distributed set of technologies → **high entropy**

To quantify this, we use *normalized Shannon entropy*, which measures how evenly technologies are distributed across a competition's write-ups and rescales the entropy to the $[0, 1]$ range for comparability.

Normalization is necessary because competitions vary in the number of mentioned technologies, the number of participating teams, and the underlying technological landscape, making direct comparisons across competitions otherwise unreliable.

1. Normalized Entropy per Competition

Figure 21 shows the *normalized entropy* for each competition launched since 2010. Each bar represents how evenly technologies were mentioned in this competition's write-ups. In this context, a value close to 0 indicates that a single technology dominated the write-ups for that competition, whereas a value near 1 suggests that multiple technologies were mentioned with similar frequency. Here, we only present a zoomed-out, high-level view of the figure to illustrate that the average entropy remains consistently close to 1 (for the complete interactive chart, refer to [50]). This

indicates that within competition write-ups, no single technology overwhelmingly dominates. Instead, a general diversity of tools and approaches is consistently present across competition writeups.

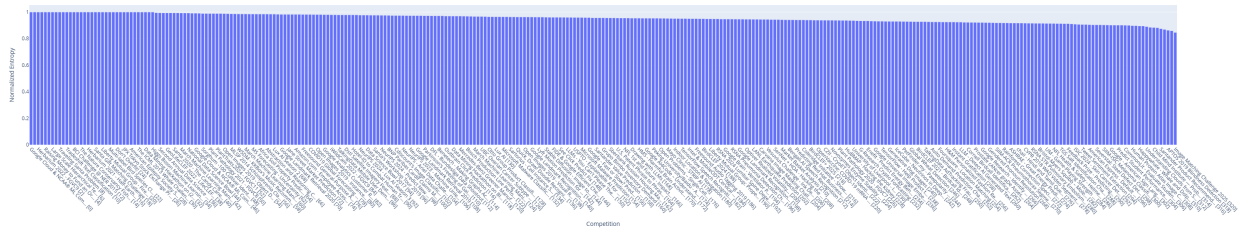


Figure 21: Normalized Entropy per Competition [52]

2. Effective Number of Technologies per Competition

Since the normalized entropy can be abstract and unitless, we also compute the effective number of technologies, defined as:

$$\text{Effective Tech Count} = 2^{H(X)}$$

where $H(X)$ refers to the unnormalized Shannon entropy (in bits). This yields a count-like metric, where—for example—if the technology usage is equivalent to evenly using six different technologies, the effective technology count is 6. This interpretation makes diversity comparisons more intuitive, especially in cases where normalized entropy values are tightly clustered. Figure 22 shows the results.

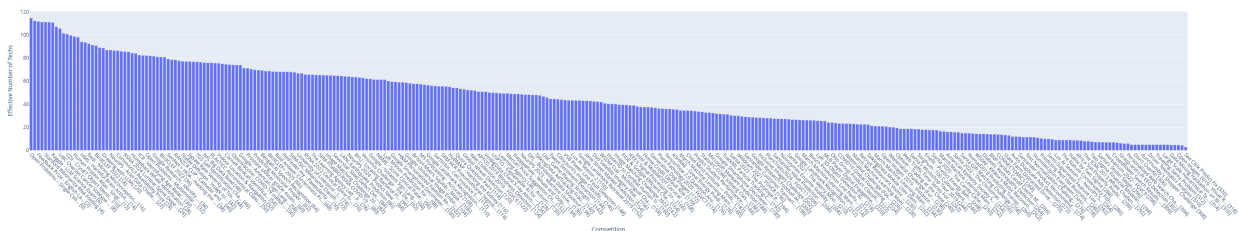


Figure 22: Effective Number of Technologies per Competition [54]

We see a different chart now—one that reveals not every competition utilizes an actually broad technology stack, contrary to earlier impressions. However, before interpreting these results in detail, we first examine technological diversity aggregated by year, rather than by individual competitions.

3. Technology Diversity Trends by Year

To analyze how technological diversity has evolved over time, we compare two annual metrics: normalized entropy and the effective number of technologies and plot them through Figure 23. The normalized entropy per year shows how evenly different technologies were used across competitions in a given year. If the value goes up over time, it means the range of technologies is becoming more diverse. If it stays flat or goes down, it might indicate that fewer technologies are being used more consistently—possibly due to standardization or consolidation. To again make this easier to interpret, we also show the effective number of technologies per year. This translates the entropy value into a more intuitive number, similar to a count. For instance, an effective count of about 500 in 2021 means that the mix of technologies used that year was as diverse as if 500 different technologies were each used equally. This helps us easily spot the years when technological diversity increased or decreased.

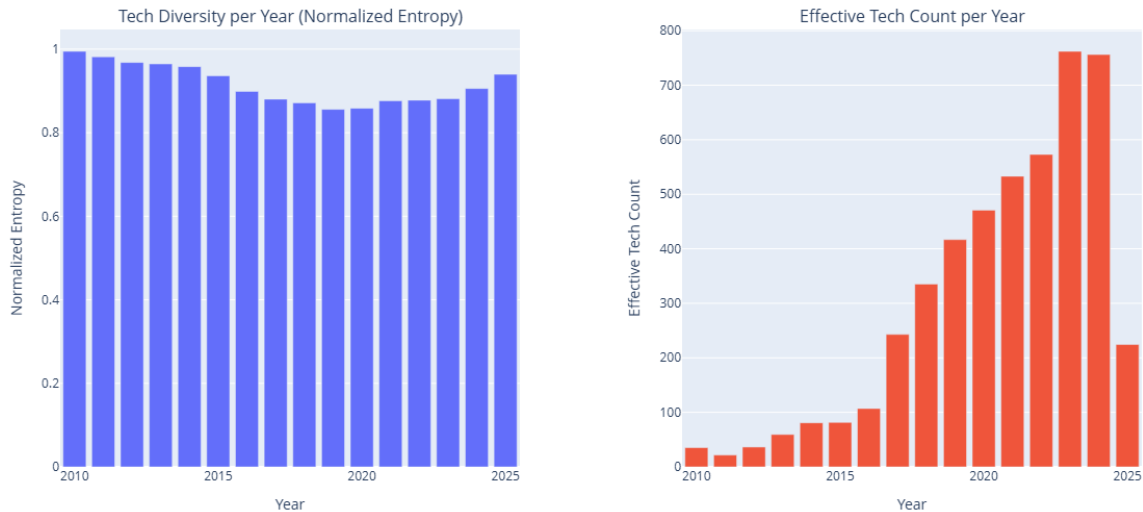


Figure 23: Technology Diversity Trends by Year: Effective vs. Normalized [56]

Why Normalized Entropy and Effective Tech Count Look Different

The two plots tell different stories because they measure different aspects of technology use. Normalized entropy shows how evenly technologies are used in a given year. It's a score between 0 and 1. Once usage becomes fairly balanced, the score levels off—even if new technologies are introduced. Effective tech count, on the other hand, reflects how many technologies were actually used. As more technologies are adopted over time, this number keeps increasing. In short, the entropy plot tells us that tech usage has stayed balanced, while the effective tech count reveals that the overall variety of technologies has grown significantly.

Overall, we observe that (a) the effective count of technological diversity has been steadily increasing year over year, and (b) the normalized entropy remains consistently high. This suggests that Kagglers are not converging on a single dominant technology—although many rely on a common core toolbox—but instead experiment with a variety of approaches and tools. Such diversity, often achieved by combining multiple technologies, indicates a broad modeling strategy and aligns well with real-world experimentation and solution development.

3.4.2 Competition & Technologies Network

Finally, for each year, we constructed bipartite graphs in which blue-colored nodes represent competitions and yellow-colored nodes represent technologies. An edge between a competition and a technology indicates that the technology was either mentioned or used in the corresponding competition. Edge thickness and color intensity reflect the frequency of mentions or usage. Since these graphs are highly interactive, we include one example graph here as Figure 24—corresponding to the year 2014. For all other yearly graphs and their interactive versions, we refer readers to the accompanying notebook [57].

3.5 Conclusion

Our analysis of Kaggle kernels reveals clear trends in programming language preferences, technology adoption, and data science practices within the community. Python has become the dominant language, now used in approximately 95% of all kernels, while the use of R has steadily declined since 2016.

The most frequently imported packages—such as `pandas`, `numpy`, and `matplotlib`—form the backbone of many workflows, underscoring a pragmatic, modeling-driven approach that also values visualization and basic data cleaning.

Over time, we observe increasing entropy and diversity in both package imports and method calls, particularly in competition settings where adaptation to new tools is quick. Early dominance by `xgboost` has shifted toward modern libraries like `lightgbm`, `transformers`, and `autogluon`, with growing interest in AutoML, deep learning, and generative models.

Despite concerns over leaderboard overfitting, our metric-based evaluation across competitions shows that public-private score discrepancies typically range between 5–10%, suggesting solid generalization in most cases.

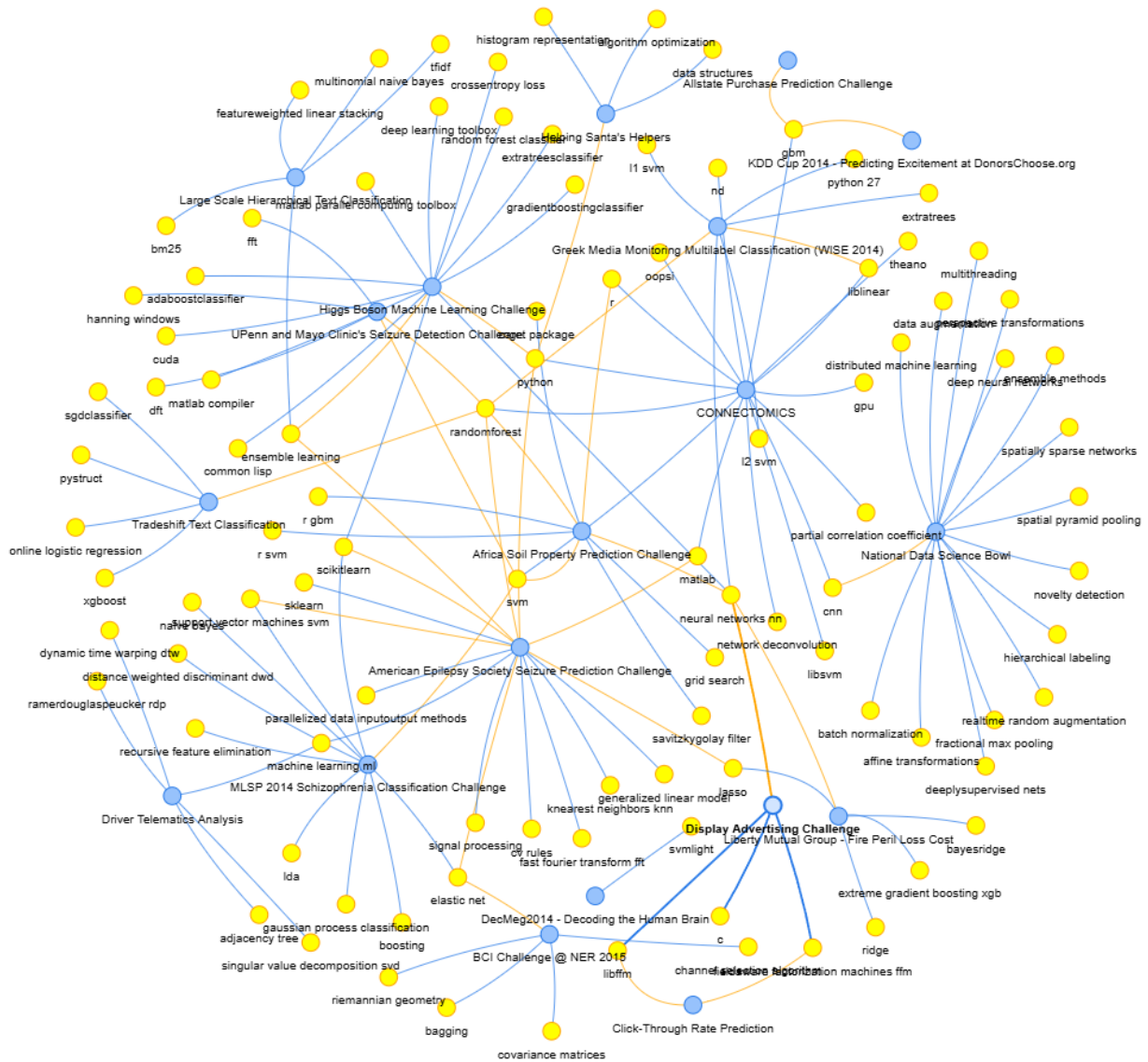


Figure 24: Competition–Technology network for the year 2014. Blue nodes represent competitions, while yellow nodes represent technologies. An edge between a competition and a technology indicates that the technology was mentioned—and therefore used—in that competition’s solution write-ups. The color intensity and thickness of each edge reflect the frequency of mentions [59].

4 Kaggle Competition Topics

TL;DR

Kaggle competitions have consistently focused on core machine learning topics like model evaluation, loss functions, and data preprocessing. Over the years, discussions have expanded to include specialized areas such as image generation, autonomous systems, and fraud detection. Despite evolving themes, the community's focus on practical, performance-driven strategies has remained strong and steadily grown.

In the previous section, we analyzed the META KAGGLE CODE dataset to uncover key insights, trends, and highlight the most significant developments. Within this section, we shift our focus to identifying recurring themes of topics in Kaggle competition forum discussions. As of this writing, approximately 10,000 competitions have been hosted. To minimize our scope we've restricted our analysis to *Featured* and *Research* type competitions with prize pools of at least \$10,000 (or equivalent), yielding a smaller dataset of 418 competitions. **Topic modeling** is then used to extract the main themes from these discussions.

4.1 What is Topic Modeling?

Topic modeling is a **Natural Language Processing** (NLP) technique used to identify underlying themes within a large collection of text by grouping frequently co-occurring words into meaningful topics [1]. These clusters of terms naturally form patterns that reveal the main topics or themes of topics being discussed. While traditional approaches like **Latent Dirichlet Allocation (LDA)** [4] and **Latent Semantic Analysis (LSA)** [7] have been widely used. Recent advancements in technology and processing power have paved way for newer, more sophisticated techniques. In this section, we will be utilizing **BERTopic** [9], which, under-the-hood, uses pre-trained transformer models such as BERT to generate contextual embeddings for improved topic extraction.

Compared to classical methods, **BERTopic** offers greater accuracy and a deeper understanding of context, particularly when working with short or complex texts. Its flexibility and interpretability make it especially valuable for large-scale text analysis across various domains, including business intelligence, social media monitoring, and customer feedback analysis.

4.2 Topic Modeling Kaggle Competitions

Based on the **BERTopic** model applied to Kaggle *Featured* and *Research* competitions with prize pools of at least \$10,000, we observe several key topics that are being discussed by participants in multiple competitions as shown in Figure 25. Across all competitions, the most dominant topic, **Topic 0**, centers around model evaluation and feature engineering. Terms like “cross validation”, “public leaderboard”, “private leaderboard” and “feature engineering” highlight how participants prioritize understanding of how their models are scored, generalize to unseen data, and rank on leaderboards. This reflects the competitive nature of Kaggle participants, where effective validation strategies and well-engineered features are critical for achieving top rankings.

Following this, **Topic 1** focuses on loss functions and model performance, with key terms such as “log loss”, “mean average precision”, and “winning rate” indicating discussions around optimizing evaluation metrics specific to competition goals. **Topic 2** reflects team collaboration, with phrases like “team member”, “looking team” and “team work” showing the importance of group effort and shared insights. **Topic 3** is centered on optimization problems, involving terms like “optimal solution”, “lower bound” and “optimization problem” suggesting that participants often discuss theoretical and practical approaches to improving model performance. **Topic 4** is largely about missing data and imputation, with high scores for words like “missing values” “impute” and “NA values” indicating a strong focus on data preprocessing as a foundational step in many competitions.

Beyond the top five, the remaining topics cover a wide range of themes relevant to machine learning competitions. **Topic 6** addresses “class imbalance”, a common challenge in classification tasks. **Topics 7 and 27** discuss “autonomous systems” and “image generation”, respectively, pointing to the presence of specialized domains such as computer vision and autonomous driving. **Topics 10 and 11** focus on “competition logistics” and “platform features”, reflecting discussions about how competitions are run and participant engagement.

In general, these topics show that while core modeling and data strategies occur the most, participants also engage deeply with on competition-specific discussions.

The graph displayed in Figure 26 contains the same set of topics as outlined before, but presented in a 2D visualization of topic clusters. We generated this graph the built in function `visualize_document_datamap()` of a **BERTopic**



Figure 25: Competition Topics and Associated Most Important Terms [61]

Documents and Topics

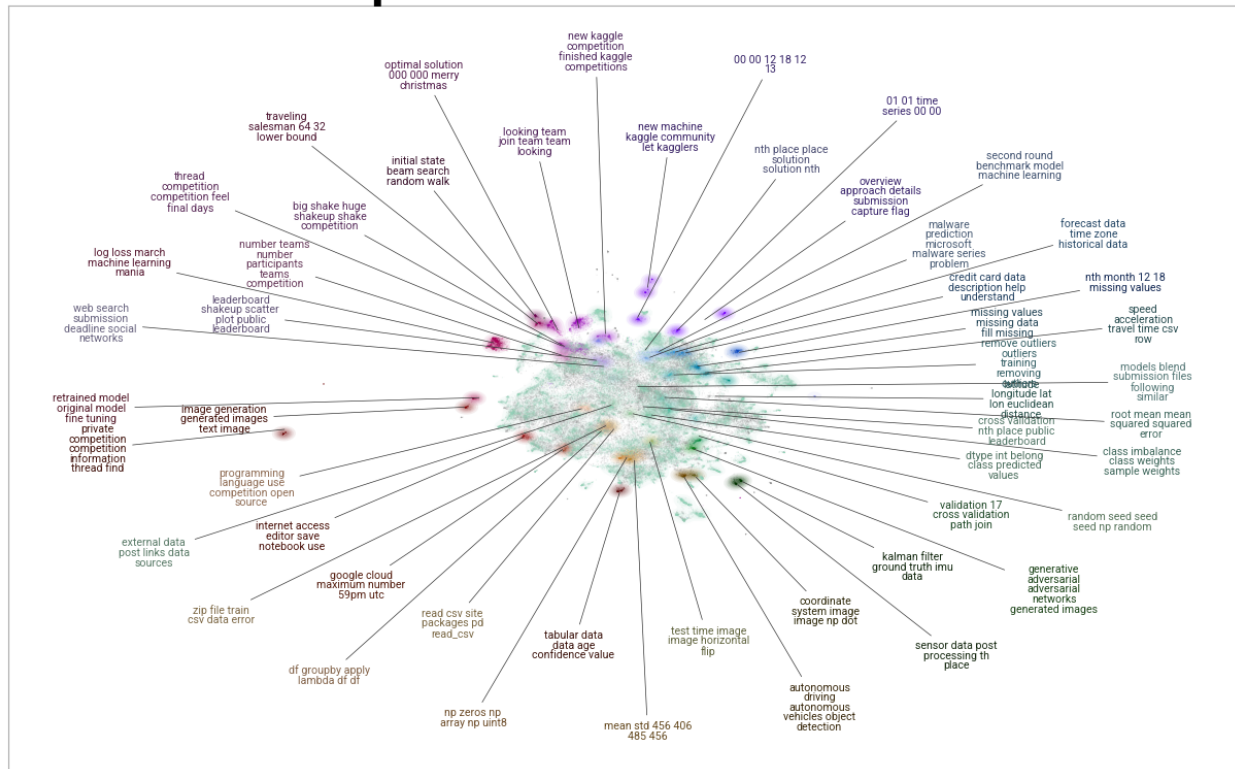


Figure 26: Competition Topics Data Map [\[63\]](#)

instance. Each point represents a document or post from the dataset, and points that are closer together share higher semantic similarity, meaning they discuss similar ideas or themes.

Topics are shown as color-coded clusters, and their spatial distribution reflects how near or distant the topics are from one another. For example, closely grouped labels such as “log loss”, “public leaderboard” and “cross validation” occupy the central area, indicating these themes are core to most discussions and overlap significantly across many competitions. In contrast, more specialized topics like “image generation”, “autonomous driving” and “credit card fraud” appear on the outer edges, representing niche domains or task-specific discussions.

The visualization helps us understand not only what topics are discussed, but how interconnected they are. General machine learning principles such as model evaluation, data preprocessing, and leaderboard strategies form a dense central cluster, while more technical or domain-specific topics spread outward, illustrating the breadth of conversations in high-stakes Kaggle competitions.

Figure 27 shows how often different topics were discussed in Kaggle competitions from 2011 to 2025. It highlights that while new topics have appeared over the years, many key themes have stayed relevant and consistent. The steady growth of most topics over the years suggests that interest in both general machine learning concepts and specific problem areas has continued to grow over time. Also, the number of members that are joining Kaggle and participating in competitions is growing as well.

From early years through to the most recent competitions, the diversity of discussions has broadened, indicating an expanding scope of challenges and methodologies in the Kaggle ecosystem. Some topics gained momentum quickly and sustained high relevance, while others appeared later but grew rapidly, reflecting evolving interests in areas such as deep learning, specialized domains, and practical implementation issues. Overall, the chart highlights the evolving, yet stable nature of knowledge exchanges in discussions within Kaggle competition forums.

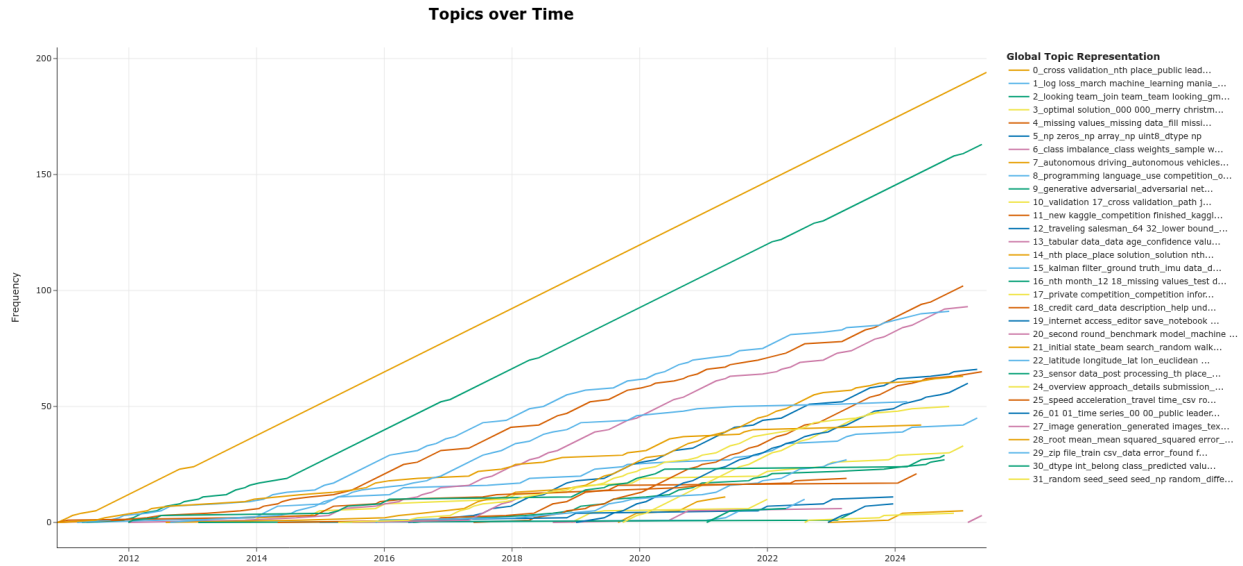


Figure 27: Competition Topics Over Time [65]

4.3 Topic Modeling Specific Competitions

In the previous section, we identified recurring themes across multiple Kaggle competitions. To explore whether these patterns also emerge at the individual competition level, we selected one of Kaggle's competitions with the highest number of participants. Generally, competitions with a larger number of participants tend to generate more forum activity, resulting in a greater volume and variety of discussion threads.

The Santander Customer Transaction Prediction competition challenged participants to build models that predict which customers will make a specific future transaction, focusing on a binary classification task. With 25,500 entrants, over 104,000 submissions, and a \$65,000 prize pool, it became one of Kaggle's most participated competitions. Hosted by Santander Bank, the competition offered anonymized, real-world-like data to encourage innovative solutions from the global data science community.

We extracted all the topics that were created during the competition was active and up to a month after the competition has closed. Figure 28 is a topic barchart of the competition showing most important terms (words) associated with each topic, ranked by their c-TF-IDF scores.

Figure 29 visualizes the clustering of the topics' embeddings and we can definitely see that most of the discussions revolve around training and test data, cross-validation strategies and the metric being used in the competition. And based on the discussion, we get to know that the competition's chosen metric is AUC metric.

Figure 30 visualizes the evolution of discussion topics in the Kaggle forum for the "Santander Customer Transaction Prediction" competition, which began in February 2019. The graph tracks cumulative frequencies of four main topic clusters over time, with the competition period shaded in green and the post-competition period in red. During the competition, the most dominant topic was related to model training and cross-validation strategies, reflecting participants' focus on improving performance.

4.4 Conclusion

The results confirm that the same key topics identified across all competitions, model evaluation, leaderboard strategies, and feature engineering, are also consistently discussed within individual competitions. These topics discuss the fundamentals which participants need to adhere to become effective when competing in Kaggle competitions.

The analysis of Kaggle competitions reveals consistent engagement with fundamental machine learning topics and techniques such as model evaluation, leaderboard strategies, and data preprocessing. These themes have remained central across the years, reflecting their importance in achieving competitive results. At the same time, the growth of domain-specific topics including image generation, autonomous systems, and fraud detection, demonstrates that both

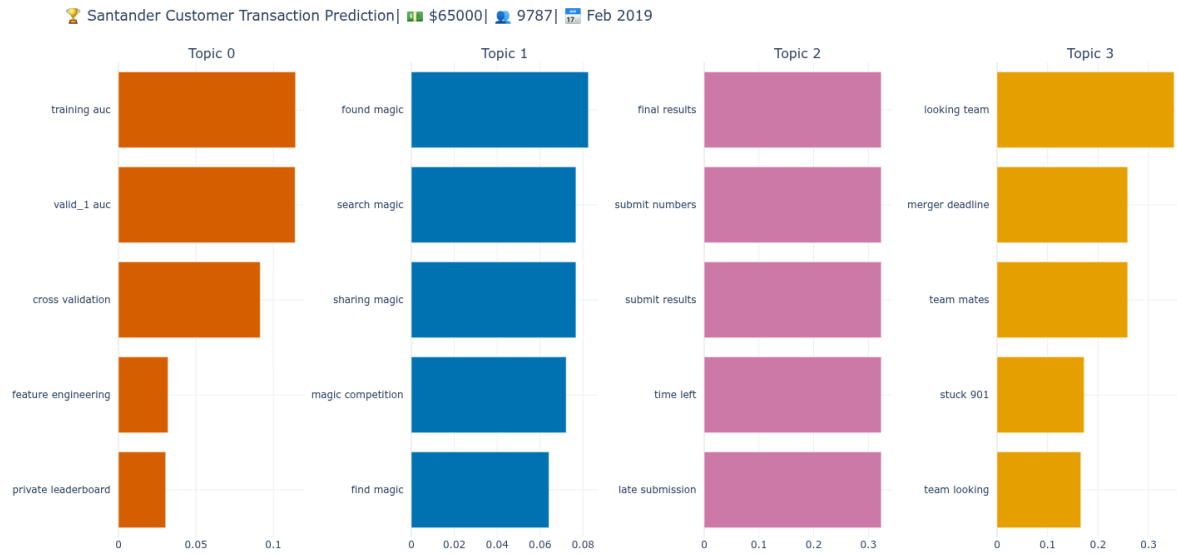


Figure 28: Santander Topics and Associated Most Important Terms [67]

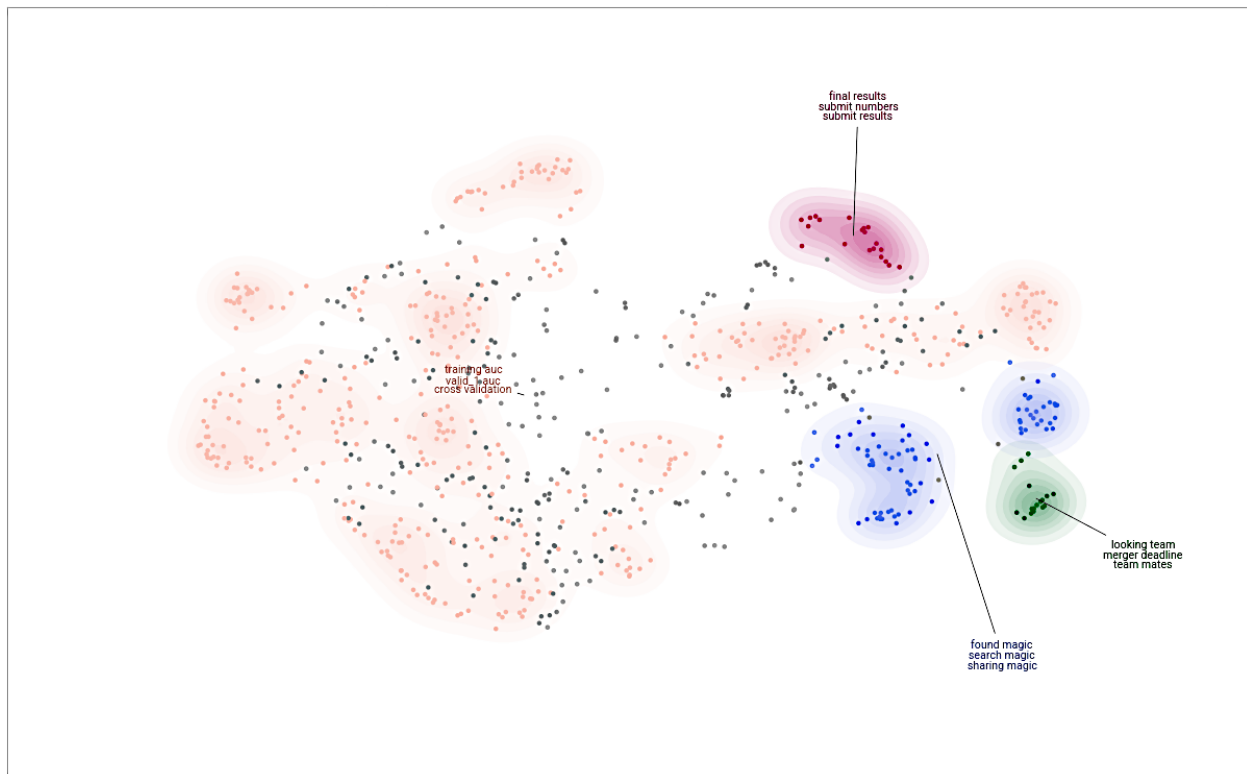


Figure 29: Santander Topics Data Map [69]

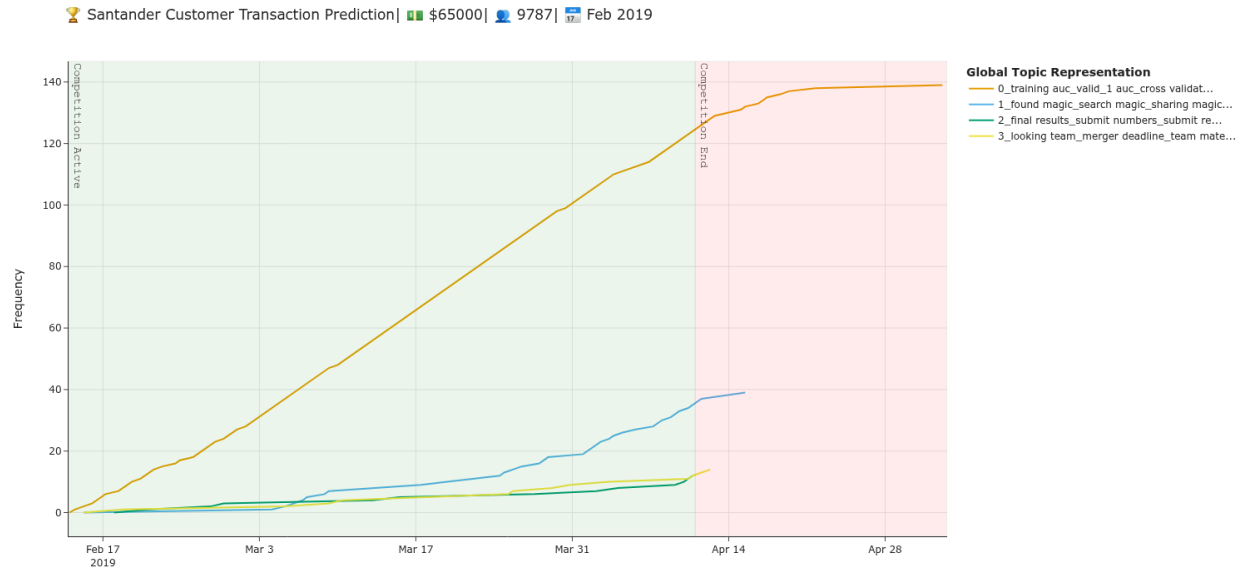


Figure 30: Santander Topics over Time [71]

the platform and community evolves and adapts to real-world challenges and industry trends. The visual distribution of topics shows clear clustering around core technical discussions, with more specialized topics forming distinct but related areas. Our analysis further highlights the sustained and increasing interest in both general and niche topics.

In general, the findings underscore the maturity and depth of the Kaggle community, as well as its ongoing collective effort to pursue high-quality, solution-driven collaboration in data science.

References

- [1] Aly Abdelrazek, Yomna Eid, Eman Gawish, Walaa Medhat, and Ahmed Hassan. Topic modeling algorithms and applications: A survey. *Information Systems*, 112:102131, 2023.
- [2] Tania Babina, Anastassia Fedyk, Alex He, and James Hodson. Artificial intelligence, firm growth, and product innovation. *Journal of Financial Economics*, 151:103745, 2024.
- [3] Théophile Bisi, Anthony Risser, Philippe Clavert, Henri Migaud, and Julien Dartus. What is the rate of text generated by artificial intelligence over a year of publication in orthopedics & traumatology: Surgery & research? analysis of 425 articles before versus after the launch of chatgpt in november 2022. *Orthopaedics & Traumatology: Surgery & Research*, 109(8):103694, 2023.
- [4] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.
- [5] Venkata Rama Padmaja Chinimilli and Lakshminarayana Sadasivuni. The rise of ai: a comprehensive research review. *IAES International Journal of Artificial Intelligence (IJ-AI)*, 13:2226, 06 2024.
- [6] Manas Dave and Neil Patel. Artificial intelligence in healthcare and education. *British Dental Journal*, 234(10):761–764, 2023.
- [7] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.
- [8] Faisal R. Elali and Leena N. Rachid. Ai-generated research paper fabrication and plagiarism in the scientific community. *Patterns*, 4(3):100706, 2023.
- [9] Maarten Grootendorst. Bertopic: Neural topic modeling with a class-based tf-idf procedure, 2022.
- [10] Hans W. A. Hanley and Zakir Durumeric. Machine-made media: Monitoring the mobilization of machine-generated articles on misinformation and mainstream news websites, 2024.
- [11] Seungone Kim, Juyoung Suk, Ji Yong Cho, Shayne Longpre, Chaeun Kim, Dongkeun Yoon, Guijin Son, Yejin Cho, Sheikh Shafayat, Jinheon Baek, Sue Hyun Park, Hyeonbin Hwang, Jinkyung Jo, Hyowon Cho, Haebin Shin, Seongyun Lee, Hanseok Oh, Noah Lee, Namgyu Ho, Se June Joo, Miyoung Ko, Yoonjoo Lee, Hyungjoo Chae, Jamin Shin, Joel Jang, Seonghyeon Ye, Bill Yuchen Lin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. The biggen bench: A principled benchmark for fine-grained evaluation of language models with language models, 2025.
- [12] Weixin Liang, Yaohui Zhang, Zhengxuan Wu, Haley Lepp, Wenlong Ji, Xuandong Zhao, Hancheng Cao, Sheng Liu, Siyu He, Zhi Huang, Diyi Yang, Christopher Potts, Christopher D Manning, and James Y. Zou. Mapping the increasing use of llms in scientific papers, 2024.
- [13] Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. Large language models: A survey, 2025.
- [14] Shushanta Pudasaini, Luis Miralles-Pechuán, David Lillis, and Marisa Llorens Salvador. Survey on plagiarism detection in large language models: The impact of chatgpt and gemini on academic integrity, 2024.
- [15] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, Agnieszka Kluska, Aitor Lewkowycz, Akshat Agarwal, Alethea Power, Alex Ray, Alex Warstadt, Alexander W. Kocurek, Ali Safaya, Ali Tazarv, Alice Xiang, Alicia Parrish, Allen Nie, Aman Hussain, Amanda Askell, Amanda Dsouza, Ambrose Slone, Ameet Rahane, Anantharaman S. Iyer, Anders Andreassen, Andrea Madotto, Andrea Santilli, Andreas Stuhlmüller, Andrew Dai, Andrew La, Andrew Lampinen, Andy Zou, Angela Jiang, Angelica Chen, Anh Vuong, Animesh Gupta, Anna Gottardi, Antonio Norelli, Anu Venkatesh, Arash Gholamidavoodi, Arfa Tabassum, Arul Menezes, Arun Kirubakaran, Asher Mullokandov, Ashish Sabharwal, Austin Herrick, Avia Efrat, Aykut Erdem, Ayla Karakaş, B. Ryan Roberts, Bao Sheng Loe, Barret Zoph, Bartłomiej Bojanowski, Batuhan Özyurt, Behnam Hedayatnia, Behnam Neyshabur, Benjamin Indén, Benno Stein, Berk Ekmekci, Bill Yuchen Lin, Blake Howald, Bryan Orinon, Cameron Diao, Cameron Dour, Catherine Stinson, Cedrick Argueta, César Ferri Ramírez, Chandan Singh, Charles Rathkopf, Chenlin Meng, Chitta Baral, Chiyu Wu, Chris Callison-Burch, Chris Waites, Christian Voigt, Christopher D. Manning, Christopher Potts, Cindy Ramirez, Clara E. Rivera, Clemencia Siro, Colin Raffel, Courtney Ashcraft, Cristina Garbacea, Damien Sileo, Dan Garrette, Dan Hendrycks, Dan Kilman, Dan Roth, Daniel Freeman, Daniel Khashabi, Daniel Levy, Daniel Moseguí González, Danielle Perszyk, Danny Hernandez, Danqi Chen, Daphne Ippolito, Dar Gilboa, David Dohan, David Drakard, David Jurgens, Debajyoti Datta, Deep Ganguli, Denis Emelin, Denis Kleyko, Deniz Yuret, Derek Chen, Derek Tam, Dieuwke Hupkes, Diganta Misra, Dilyar Buzan, Dimitri Coelho Mollo, Diyi Yang, Dong-Ho Lee, Dylan Schrader, Ekaterina Shutova, Ekin Dogus

Cubuk, Elad Segal, Eleanor Hagerman, Elizabeth Barnes, Elizabeth Donoway, Ellie Pavlick, Emanuele Rodola, Emma Lam, Eric Chu, Eric Tang, Erkut Erdem, Ernie Chang, Ethan A. Chi, Ethan Dyer, Ethan Jerzak, Ethan Kim, Eunice Engefu Manyasi, Evgenii Zheltonozhskii, Fanyue Xia, Fatemeh Siar, Fernando Martínez-Plumed, Francesca Happé, Francois Chollet, Frieda Rong, Gaurav Mishra, Genta Indra Winata, Gerard de Melo, Germán Kruszewski, Giambattista Parascandolo, Giorgio Mariani, Gloria Wang, Gonzalo Jaimovitch-López, Gregor Betz, Guy Gur-Ari, Hana Galijasevic, Hannah Kim, Hannah Rashkin, Hannaneh Hajishirzi, Harsh Mehta, Hayden Bogar, Henry Shevlin, Hinrich Schütze, Hiromu Yakura, Hongming Zhang, Hugh Mee Wong, Ian Ng, Isaac Noble, Jaap Jumelet, Jack Geissinger, Jackson Kernion, Jacob Hilton, Jaehoon Lee, Jaime Fernández Fisac, James B. Simon, James Koppel, James Zheng, James Zou, Jan Kocoń, Jana Thompson, Janelle Wingfield, Jared Kaplan, Jarema Radom, Jascha Sohl-Dickstein, Jason Phang, Jason Wei, Jason Yosinski, Jekaterina Novikova, Jelle Bosscher, Jennifer Marsh, Jeremy Kim, Jeroen Taal, Jesse Engel, Jesujoba Alabi, Jiacheng Xu, Jiaming Song, Jillian Tang, Joan Waweru, John Burden, John Miller, John U. Balis, Jonathan Batchelder, Jonathan Berant, Jörg Froberg, Jos Rozen, Jose Hernandez-Orallo, Joseph Boudeman, Joseph Guerr, Joseph Jones, Joshua B. Tenenbaum, Joshua S. Rule, Joyce Chua, Kamil Kanclerz, Karen Livescu, Karl Krauth, Karthik Gopalakrishnan, Katerina Ignatyeva, Katja Markert, Kaustubh D. Dhole, Kevin Gimpel, Kevin Omondi, Kory Mathewson, Kristen Chiafullo, Ksenia Shkaruta, Kumar Shridhar, Kyle McDonell, Kyle Richardson, Laria Reynolds, Leo Gao, Li Zhang, Liam Dugan, Lianhui Qin, Lidia Contreras-Ochando, Louis-Philippe Morency, Luca Moschella, Lucas Lam, Lucy Noble, Ludwig Schmidt, Luheng He, Luis Oliveros Colón, Luke Metz, Lütfi Kerem Şenel, Maarten Bosma, Maarten Sap, Maartje ter Hoeve, Maheen Farooqi, Manaal Faruqi, Mantas Mazeika, Marco Baturan, Marco Marelli, Marco Maru, Maria Jose Ramírez Quintana, Marie Tolkiehn, Mario Giulianelli, Martha Lewis, Martin Potthast, Matthew L. Leavitt, Matthias Hagen, Mátyás Schubert, Medina Orduna Baitemirova, Melody Arnaud, Melvin McElrath, Michael A. Yee, Michael Cohen, Michael Gu, Michael Ivanitskiy, Michael Starritt, Michael Strube, Michał Śwędrowski, Michele Bevilacqua, Michihiro Yasunaga, Mihir Kale, Mike Cain, Mimeo Xu, Mirac Suzgun, Mitch Walker, Mo Tiwari, Mohit Bansal, Moin Aminnaseri, Mor Geva, Mozhdah Gheini, Mukund Varma T, Nanyun Peng, Nathan A. Chi, Nayeon Lee, Neta Gur-Ari Krakover, Nicholas Cameron, Nicholas Roberts, Nick Doiron, Nicole Martinez, Nikita Nangia, Niklas Deckers, Niklas Muennighoff, Nitish Shirish Keskar, Niveditha S. Iyer, Noah Constant, Noah Fiedel, Nuan Wen, Oliver Zhang, Omar Agha, Omar Elbaghdadi, Omer Levy, Owain Evans, Pablo Antonio Moreno Casares, Parth Doshi, Pascale Fung, Paul Pu Liang, Paul Vicol, Pegah Alipoormolabashi, Peiyuan Liao, Percy Liang, Peter Chang, Peter Eckersley, Phu Mon Htut, Pinyu Hwang, Piotr Miłkowski, Piyush Patil, Pouya Pezeshkpour, Priti Oli, Qiaozhu Mei, Qing Lyu, Qinlang Chen, Rabin Banjade, Rachel Etta Rudolph, Raefer Gabriel, Rahel Habacker, Ramon Risco, Raphaël Millière, Rhythm Garg, Richard Barnes, Rif A. Saurous, Riku Arakawa, Robbe Raymaekers, Robert Frank, Rohan Sikand, Roman Novak, Roman Sitelew, Ronan LeBras, Rosanne Liu, Rowan Jacobs, Rui Zhang, Ruslan Salakhutdinov, Ryan Chi, Ryan Lee, Ryan Stovall, Ryan Teehan, Rylan Yang, Sahib Singh, Saif M. Mohammad, Sajant Anand, Sam Dillavou, Sam Shleifer, Sam Wiseman, Samuel Gruetter, Samuel R. Bowman, Samuel S. Schoenholz, Sanghyun Han, Sanjeev Kwatra, Sarah A. Rous, Sarik Ghazarian, Sayan Ghosh, Sean Casey, Sebastian Bischoff, Sebastian Gehrmann, Sebastian Schuster, Sepideh Sadeghi, Shadi Hamdan, Sharon Zhou, Shashank Srivastava, Sherry Shi, Shikhar Singh, Shima Asaadi, Shixiang Shane Gu, Shubh Pachchigar, Shubham Toshniwal, Shyam Upadhyay, Shyamolima, Debnath, Siamak Shakeri, Simon Thormeyer, Simone Melzi, Siva Reddy, Sneha Priscilla Makini, Soo-Hwan Lee, Spencer Torene, Sriharsha Hatwar, Stanislas Dehaene, Stefan Divic, Stefano Ermon, Stella Biderman, Stephanie Lin, Stephen Prasad, Steven T. Piantadosi, Stuart M. Shieber, Summer Misherghi, Svetlana Kiritchenko, Swaroop Mishra, Tal Linzen, Tal Schuster, Tao Li, Tao Yu, Tariq Ali, Tatsu Hashimoto, Te-Lin Wu, Théo Desbordes, Theodore Rothschild, Thomas Phan, Tianle Wang, Tiberius Nkinyili, Timo Schick, Timofei Kornev, Titus Tunduny, Tobias Gerstenberg, Trenton Chang, Trishala Neeraj, Tushar Khot, Tyler Shultz, Uri Shaham, Vedant Misra, Vera Demberg, Victoria Nyamai, Vikas Raunak, Vinay Ramasesh, Vinay Uday Prabhu, Vishakh Padmakumar, Vivek Srikumar, William Fedus, William Saunders, William Zhang, Wout Vossen, Xiang Ren, Xiaoyu Tong, Xinran Zhao, Xinyi Wu, Xudong Shen, Yadollah Yaghoobzadeh, Yair Lakretz, Yangqiu Song, Yasaman Bahri, Yejin Choi, Yichi Yang, Yiding Hao, Yifu Chen, Yonatan Belinkov, Yu Hou, Yufang Hou, Yuntao Bai, Zachary Seid, Zhuoye Zhao, Zijian Wang, Zijie J. Wang, Zirui Wang, and Ziyi Wu. *Beyond the imitation game: Quantifying and extrapolating the capabilities of language models*, 2023.

- [16] Haonan Wang, James Zou, Michael Mozer, Anirudh Goyal, Alex Lamb, Linjun Zhang, Weijie J Su, Zhun Deng, Michael Qizhe Xie, Hannah Brown, and Kenji Kawaguchi. Can ai be as creative as humans?, 2024.

Appendix

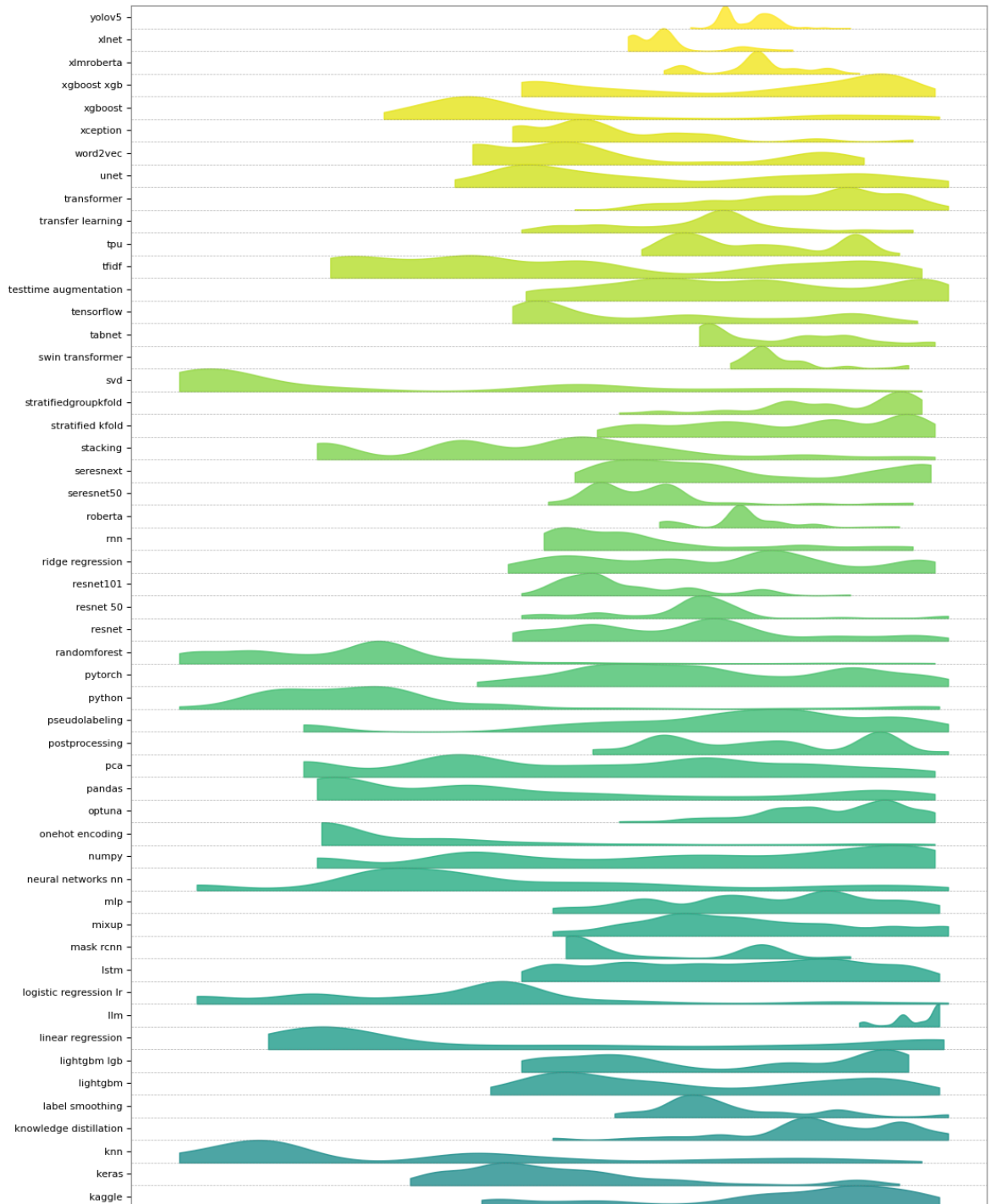


Figure 31: The top technologies mentioned in solution writeups over time (part 1/2) [73].

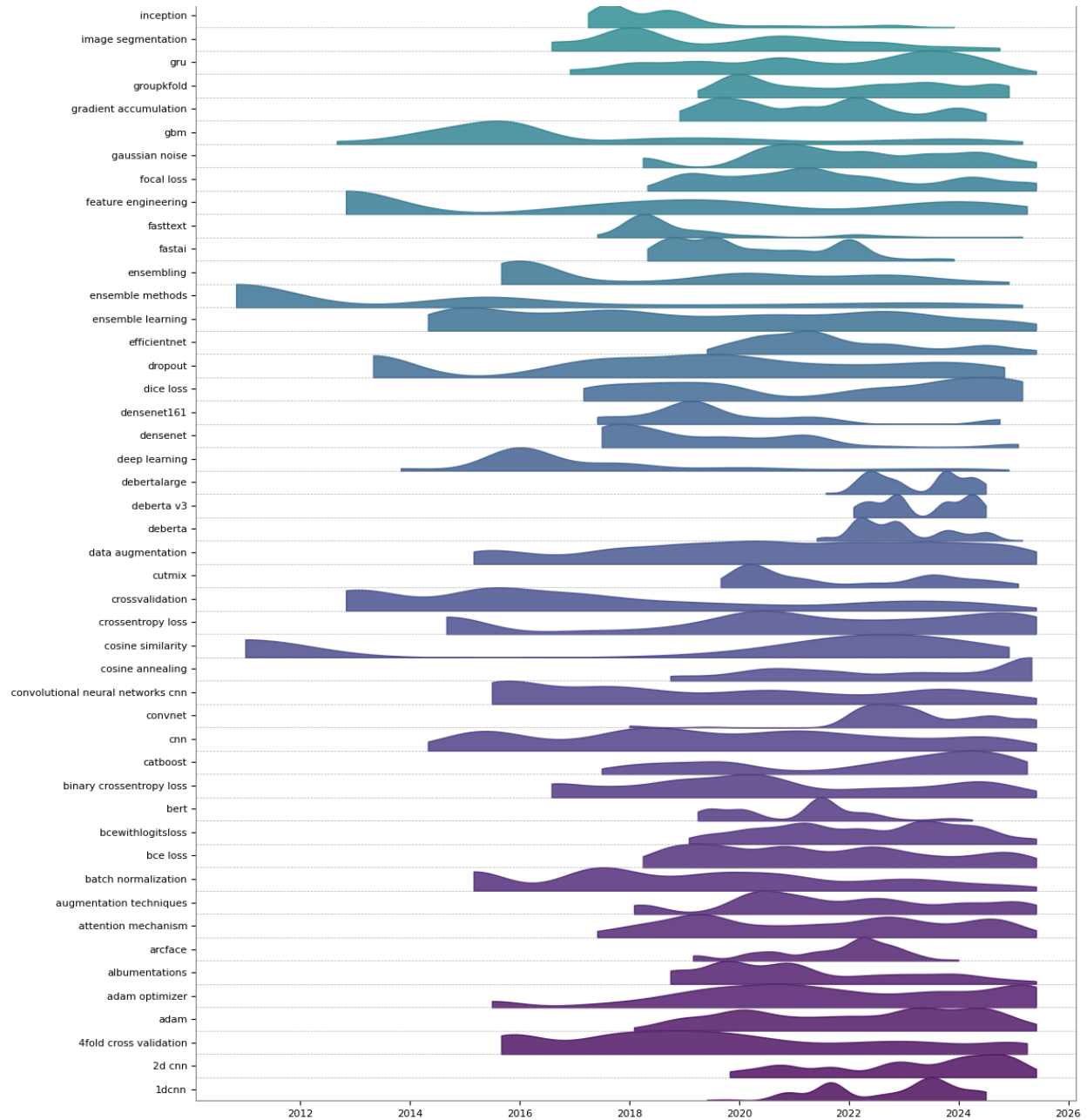


Figure 32: The top technologies mentioned in solution writeups over time (part 2/2) [\[75\]](#).