ELF-Gym: Evaluating Large Language Models Generated Features for Tabular Prediction

Yanlin Zhang
Fudan University
Shanghai, China
21210720043@m.fudan.edu.cn

Ning Li
Shanghai Jiao Tong University
Shanghai, China
lining01@sjtu.edu.cn

Quan Gan
Amazon Shanghai AI Lab
Shanghai, China
quagan@amazon.com

Weinan Zhang
Shanghai Jiao Tong University
Shanghai, China
wnzhang@sjtu.edu.cn

David Wipf
Amazon Shanghai AI Lab
Shanghai, China
daviwipf@amazon.com

Minjie Wang
Amazon Shanghai AI Lab
Shanghai, China
minjiw@amazon.com

Abstract

Crafting effective features is a crucial yet labor-intensive and domain-specific task within machine learning pipelines. Fortunately, recent advancements in Large Language Models (LLMs) have shown promise in automating various data science tasks, including feature engineering. But despite this potential, evaluations thus far are primarily based on the end performance of a complete ML pipeline, providing limited insight into precisely how LLMs behave relative to human experts in feature engineering. To address this gap, we propose ELF-Gym, a framework for Evaluating LLM-generated Features. We curated a new dataset from historical Kaggle competitions, including 251 "golden" features used by top-performing teams. ELF-Gym then quantitatively evaluates LLM-generated features by measuring their impact on downstream model performance as well as their alignment with expert-crafted features through semantic and functional similarity assessments. This approach provides a more comprehensive evaluation of disparities between LLMs and human experts, while offering valuable insights into specific areas where LLMs may have room for improvement. For example, using ELF-Gym we empirically demonstrate that, in the best-case scenario, LLMs can semantically capture approximately 56% of the golden features, but at the more demanding implementation level this overlap drops to 13%. Moreover, in other cases LLMs may fail completely, particularly on datasets that require complex features, indicating broad potential pathways for improvement.

CCS Concepts

• **Computing methodologies** → **Artificial intelligence**.

Keywords

Large Language Models, Feature Engineering, Data Science

CIKM '24, October 21–25, 2024, Boise, ID, USA

ACM ISBN 979-8-4007-0436-9/24/10
https://doi.org/10.1145/3627673.3679153

ACM Reference Format:
Yanlin Zhang, Ning Li, Quan Gan, Weinan Zhang, David Wipf, and Minjie Wang. 2024. ELF-Gym: Evaluating Large Language Models Generated Features for Tabular Prediction. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM '24), October 21–25, 2024, Boise, ID, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3627673.3679153

1 Introduction

Feature engineering is a crucial step in the machine learning pipeline, transforming raw data into meaningful features that improve model performance and interpretability. Effective feature engineering can significantly enhance the predictive power of models, making it a vital component in various data-driven applications. This importance is particularly evident in competitive data science environments like Kaggle [23], where top-performing models often rely heavily on sophisticated feature engineering techniques. For instance, in one interview [1], the winners of the Grupo Bimbo Inventory Prediction competition reported that 95% of their time was on feature engineering while only 5% was on modeling.

Despite its importance, traditional feature engineering is labor-intensive and requires extensive domain knowledge. Automated tools like AutoFeat [18], OpenFE [36], SAFE [32], and Deep Feature Synthesis (DFS) [25] have emerged to streamline this process. AutoFeat automates feature selection and generation using statistical methods and heuristics but suffers from high feature generation costs. OpenFE and SAFE mitigate these costs by optimizing the feature selection phase using feedback from model evaluation. DFS extends feature engineering to multi-table scenarios by utilizing data relationships to generate features. Despite their effectiveness in reducing manual effort, these tools often fall short in leveraging the domain knowledge that human experts typically rely on for crafting relevant features.

The advent of Large Language Models (LLMs) such as GPTs [1] has opened new possibilities for automating various data science tasks. LLMs have demonstrated remarkable capabilities in natural language understanding [10, 28], text generation [27, 33], summarization [6, 13, 35], and even code synthesis [5, 7, 34]. Their ability to process and generate human-like text makes them particularly well-suited for tasks that require semantic understanding and contextual reasoning. Of particular relevance to data science, LLMs

[1] https://medium.com/kaggle-blog/grupo-bimbo-inventory-demand-winners-interview-clustifier-alex-andrey-1e3b6cec8a20

have shown potential in automating tasks such as data cleaning, feature generation, and model selection.

For example, recent work such as CAAFE [17], DS-Agent [14] and FeatLLM [16] have explored the application of LLMs to feature engineering. In brief, CAAFE leverages LLMs to generate additional features based on dataset descriptions, iteratively improving model performance through semantic understanding. Meanwhile DS-Agent employs a case-based reasoning approach, combining LLMs with expert knowledge from Kaggle competitions to automate the entire data science workflow. Finally, FeatLLM utilizes LLMs to engineer binary features through rule generation and rule parsing, significantly improving down-stream tabular prediction tasks. Despite the potential, their actual evaluations thus far are primarily based on the end performance of a complete machine learning pipeline, providing limited insight into the reason behind the distinction between the solutions built by LLMs and human experts especially with respect to feature engineering. As LLM-based data science is increasingly becoming an active research area, this calls for more transparent and insightful evaluation tools and benchmarks to effectively assess and improve these systems.

To make strides in this direction, we proposed ELF-Gym, a framework for Evaluating LLM-generated Features in machine learning pipelines. We curate a new dataset specifically designed for evaluating LLMs in feature engineering tasks, using historical data from Kaggle competitions. This dataset includes 251 so-called "golden" features used by top-5 teams in 8 Kaggle competitions. Using those features as references, we then develop toolkits to quantitatively evaluate the quality of features generated by LLMs. In addition to measuring the impact of LLM-generated features on downstream model performance, ELF-Gym also evaluates their alignment with expert-crafted features by assessing semantic and functional similarity. This provides a more direct measurement of the gap between LLMs and data science experts, offering valuable insights into specific areas where LLMs need improvement.

Leveraging the proposed toolkit, we conducted a pioneer study involving multiple popular LLMs [1, 2, 4, 21] to highlight the key differences and performance gaps between LLM-generated and human-engineered features. In doing so we focused on two main research questions:

- **RQ1**: *Can LLMs discover golden features by reasoning from data descriptions and schemas?*
- **RQ2**: *What are common golden feature patterns that LLMs excel at generating, or conversely, struggle to produce?*

For RQ1, we find that LLMs can capture at best approximately 56% of the "golden" features by description, but only 13% at best when coming to implementation, and may fail completely on datasets that require complex features. For RQ2, we found that while LLMs are capable of generating features based on simpler patterns such as feature interactions or simple group-by operations, they struggle to discover patterns involving multi-variable functions, custom aggregations and complex table joins. These new findings again highlight the necessity of a standardized benchmark and robust evaluation tools, and how the proposed new evaluation metrics against human experts can derive more insights into the area for LLMs to improve.

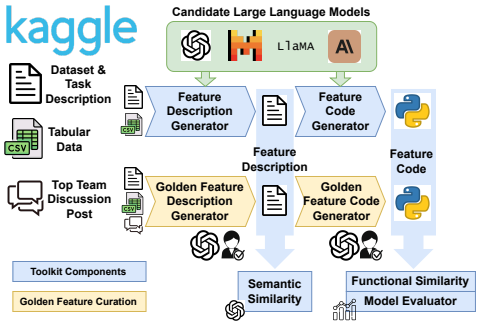

Figure 1: ELF-Gym evaluation framework overview. Blue components are part of the evaluation pipeline while the yellow ones are only triggered during golden feature curation.

2 ELF-Gym Design

2.1 Dataset and Golden Feature Collection

Kaggle is a widely used platform in the data science community, providing a rich source of datasets for developing and testing LLM-based methods. It hosts numerous competitions where data science experts compete to build the best models, often employing sophisticated feature engineering techniques. This makes Kaggle an ideal source for comparing features discovered by experts with those generated by LLMs.

However, despite competition hosts inviting or enforcing winner teams to share their code and insights in discussion posts, the code is often hard to consume as it was written in R, C++, Python, etc., well before scientific computing packages like Pandas [30] became popular. As per the insights, the discussion posts are typically written casually and are often entangled with unrelated discussions, making it difficult to extract relevant information directly.

To address this, we employ an LLM-assisted strategy to convert discussion posts into features that can be calculated from actual data. Similar to [16], we adopt a two-step approach to ensure reliability. For each Kaggle dataset, we provide GPT-4o with the dataset description, including table and column names, column labels, and the intended task (i.e., the column to predict), and prompt GPT-4o to extract feature descriptions from the discussion posts of the top-5 teams in a specific format. Once we have the feature descriptions, we further prompt GPT-4o to generate code that computes actual feature given the original table. GPT-4o can generate correct code approximately 80% of the time, significantly reducing the time of manual checking. We then manually sanitize the remaining cases where the code fails to run.

To validate the effectiveness of these extracted golden features, we test if these features can enhance model performance. This is done by executing the generated code, adding the resulting features to the original tabular data, and using AutoGluon [12] for prediction. We compare the results with predictions made using only the raw features. With this approach, we successfully curated 251 golden features from 8 Kaggle datasets with details in Table 1.

2.2 Generating Features Using LLMs

Similar to how golden features are curated, the LLM-generated features also take two steps (Figure 1). First, the candidate LLM receives the dataset description and prediction target and is asked

Table 1: Dataset statistics. Asterisks(*) indicate that the licenses prohibit use of data for academic publications.

Name	Task Description	# Golden Feats
Avito [15]	Click-through-rate Prediction	30
Instacart [20]	Product Prediction	24
Outbrain [29]	Click-through-rate Prediction	30
IEEE-CIS [19]	Fraud Detection	27
AirBnB* [3]	Destination Prediction	52
Facebook* [11]	Bot Detection	32
WestNile* [24]	Virus Prediction	25
TalkingData* [9]	Demographic Prediction	31

to generate a list of feature descriptions. The feature descriptions together with the dataset and task descriptions are then passed to the second stage for code generation. Our toolkit provides two components called **Feature Description Generator** and **Feature Code Generator** where users can plug in their LLMs for evaluation. Both the feature descriptions and code generated by LLMs are compared with the golden feature descriptions and code for evaluation, as we will explain next.

2.3 Evaluation Protocol

To measure the alignment between LLM-engineered features and golden features, ELF-Gym employs a recall metric. Given two feature sets, F_{LLM} and F_{golden}, the recall metric uses a measurement function M that returns a binary flag (1 or 0), indicating whether two features are similar. The recall metric is then defined as the proportion of golden features for which the M returns 1 when compared to features in F_{LLM}:

$$\text{Recall}(F_{LLM}, F_{golden}) = \frac{\sum_{f \in F_{golden}} \max_{f' \in F_{LLM}} M(f, f')}{|F_{golden}|}$$

ELF-Gym uses two measurement functions M. M_{sem} measures the semantic similarity of two feature descriptions by prompting a GPT-4o to assess and score the similarity. M_{func} checks if two features are functionally equivalent by comparing the outputs of feature functions applied to the input data. This process can be costly due to the slow performance of LLM-generated codes on large datasets, so we employ strategies such as representative downsampling to mitigate issues like false positives. For instance, downsampling must be done carefully to ensure that key patterns in the data are preserved, especially for features involving aggregations by specific IDs. The use of two M functions is important because, for example, if a feature is hit by M_{sem} but not M_{func}, it indicates that the LLM can identify relevant features but needs to improve its code generation capability for feature engineering tasks. We remark that while a precision metric can be defined in a similar fashion, we omit it here since the usefulness of a feature crafted by LLMs, but not overlapping with human experts, requires downstream model evaluation.

In addition to these alignment measures, ELF-Gym also evaluates the impact of LLM-generated features on downstream model performance. This involves incorporating the generated features into the original dataset and using them to train a model, then comparing the model's performance to that of a model trained on raw features. This step provides a direct measure of the utility of the

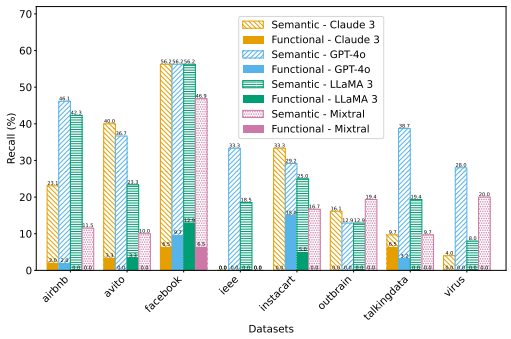

Figure 2: Recall metrics of different LLMs hitting golden features measured by semantic and functional similarity.

LLM-engineered features. ELF-Gym also supports scenarios where table data is unavailable due to restricted licenses, allowing for the evaluation of LLM feature generation capabilities based solely on data descriptions.

3 Initial Results

We select four LLMs to evaluate: GPT-4o [1], Claude 3 Sonnet [4], LLaMA3-70B-Instruct [2], and Mixtral-8x7B-Instruct [21]; by design though, ELF-Gym can also be readily adapted to handle other LLMs (see Section 2.2). We invoke GPT-4o with OpenAI's official SDK and the rest three with Amazon Bedrock[2]. For feature description generation, we explicitly prompt the LLMs to generate as many features as possible. For code generation, we gave each LLM three chances to write runnable code, each time feeding in the error message from Python interpreter as the next round of conversation. Our implementation is available at https://github.com/Lilyzhangyanlin/ELF-Gym.

3.1 RQ1: Can LLMs Discover Golden Features?

Figure 2 shows the alignment results between LLM-engineered features and golden features, measured by recall metrics using both M_{sem} and M_{func}. Overall, the recall percentages are higher for M_{sem} compared to M_{func}, indicating that while LLMs can generate features that are semantically similar to the golden features, they struggle to ensure these features are functionally equivalent. GPT generally leads in both metrics, highlighting its effectiveness in feature generation.

For the Facebook dataset, all models perform well semantically (56.25%) and functionally, with LLaMA3 leading at 12.90%, followed by GPT-4o (9.68%). This indicates that the features required for Facebook are easier for LLMs to generate both semantically and functionally. Conversely, datasets like IEEE-CIS, Outbrain, and Virus present significant challenges, with low or zero recall scores across all models for both metrics. Notable golden features in these datasets include frequency counting, feature interaction, and features grouped by multiple columns, all of which are difficult for existing LLMs (see Sec. 3.2 for deeper analysis). Additionally, IEEE-CIS has particularly strong features created by grouping by the "card1" ID column, but LLMs tend to group by other columns. No single model consistently

[2]https://aws.amazon.com/bedrock/

Table 2: Downstream model performance with different features. Top-1 and top-2 scores are bold-faced and underlined respectively.

Features Metric	Avito LogLoss ↓	IEEE-CIS AUC ↑	Outbrain AUC ↑	Instacart AUC ↑
Raw Feat.	0.0337	0.9108	0.5321	0.4977
GPT-4o	0.0355	0.9210	0.5194	0.7620
Claude 3	<u>0.0328</u>	0.9100	0.5298	<u>0.8437</u>
LLaMA 3	**0.0321**	0.9058	<u>0.6013</u>	0.5000
Mixtral	0.0340	<u>0.9213</u>	0.5971	0.5345
Golden Feat.	0.0331	**0.9236**	**0.6164**	**0.8526**

outperforms across all datasets. While GPT-4o shows strong overall performance, Claude3 excels in specific cases such as Avito, with a 40% recall for M_{sem} but struggles with M_{func}. In conclusion, these recall comparisons highlight areas where LLMs may be improvable, particularly in generating functionally equivalent features.

Table 2 compares the performance of models using LLM-generated features with those using golden features and raw data alone. The performance metrics vary by dataset, with LogLoss used for Avito and AUCROC used for IEEE-CIS, OutBrain, and Instacart. We found that golden features generally lead to the best model performance across these datasets, demonstrating the superior quality of expert-crafted features. LLM-generated features are generally useful, as they consistently improve model performance over using raw features alone. Although GPT-4o performs well in feature recall metrics, its downstream model performance surprisingly lags behind. This discrepancy is likely because GPT-4o tends to generate more features than other models, which can introduce more noise and complexity, leading to overfitting and reduced generalization. For instance, in the Instacart dataset, GPT-4o generates 35 features but achieves an AUCROC of 0.7620, whereas Claude3, with only 11 features, achieves a higher AUCROC of 0.8437. Notably, LLaMA3 and Claude3 show potential in outperforming human baselines in specific cases. For example, in the Avito dataset, Claude3 (0.0328) and LLaMA3 (0.0321) achieve better LogLoss than the human-engineered features (0.0331), indicating that LLM-generated features can sometimes surpass expert-crafted ones, especially when they effectively balance feature quantity and quality.

3.2 RQ2: Patterns LLMs Excel or Struggle with

To study which kind of features LLMs are better generating, we further break down the golden features into two categories:

- **Feature transforms**, including
  - **Unary transforms**: a unary function of a column.
  - **N-ary transforms**: a function involving multiple columns.
  - **Time-lagged features**: features based on time differences between events (rows), e.g. "finding the difference between the number of calls from the same each day with respect to the day before".
- **Joins and aggregations**, including
  - **Simple joins**: columns merged from another table.
  - **Single-column group-by**: statistics of columns grouped by a single column.

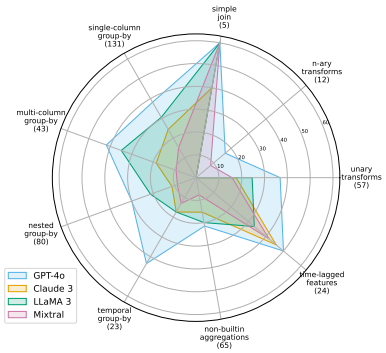

Figure 3: Recall scores (measured by semantic similarity) grouped by feature category. The numbers in parentheses represent the number of golden features within that category.

  - **Multi-column group-by**: statistics of columns grouped by a combination of columns.
  - **Nested group-by**: statistics from group-by operations within another group-by operations, e.g. "finding the minimum average of call durations per device over all mobile devices for each user".
  - **Temporal group-by**: statistics of columns grouped by some column, but only over rows whose timestamp columns are before its own timestamp, e.g. "counting the number of calls from the same device before the current call".
  - **Non-builtin aggregations**: statistics with custom aggregation functions unavailable in Pandas, e.g. entropy.

The statistics of such features, as well as the category-wise recall by semantic similarity, are shown in Figure 3. A majority of golden features are **joins and aggregations**. This is expected as the downstream models contestants used, such as gradient-boosted decision trees [8, 26] or factorization machines [22, 31], inherently incorporate feature transformations and interactions to some extent. Notably, GPT-4o outperformed the other LLMs in all categories. However, none of the LLMs achieved 50% recall in any categories, with the exception of **time-lagged features** and **simple joins** — categories that are simpler and have fewer golden features (5 and 24, respectively). The figure also highlights that LLMs struggle in generating complex features like **n-ary transforms**, **nested group-by** and **non-builtin aggregations**. These categories require designing and implementing highly specialized functions that often demand deep domain-specific knowledge, which goes beyond general common sense.

4 Conclusion

This paper proposes ELF-Gym, a framework for evaluating the feature engineering capability of LLMs, by curating a dataset with 251 human-engineered features from 8 Kaggle competitions. Initial experiments on 4 LLMs reveal varying gaps relative to human-level feature engineering expertise; in particular, we observe that while LLMs in may excel generating simple features, they struggle at times to generate features involving complex functions, aggregations, and table joins. We believe that addressing these challenges will be crucial for realizing the full potential of LLMs in automating data science tasks.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

[2] AI@Meta. 2024. Llama 3 Model Card. (2024). https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md

[3] alokgupta, Anna Montoya, LizSellier, Meghan O'Connell, and Wendy Kan. 2015. Airbnb New User Bookings. https://kaggle.com/competitions/airbnb-recruiting-new-user-bookings

[4] AI Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku. *Claude-3 Model Card* (2024).

[5] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732* (2021).

[6] Lochan Basyal and Mihir Sanghvi. 2023. Text Summarization Using Large Language Models: A Comparative Study of MPT-7b-instruct, Falcon-7b-instruct, and OpenAI Chat-GPT Models. *arXiv preprint arXiv:2310.10449* (2023).

[7] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374* (2021).

[8] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.

[9] DannyBickson, Freedom, Guy Rapaport, HanZhu, Ibrahim, RossWang, Wendy Kan, Yangyang, and Yao Lu. 2016. TalkingData Mobile User Demographics. https://kaggle.com/competitions/talkingdata-mobile-user-demographics

[10] Mengnan Du, Fengxiang He, Na Zou, Dacheng Tao, and Xia Hu. 2023. Shortcut learning of large language models in natural language understanding. *Commun. ACM* 67, 1 (2023), 110–120.

[11] Jim Dullaghan, John P. Costella, John_W, Meghan O'Connell, Rafael, Ruchi, Ruchi Varshney, Sergey, Sofus Macskassy, and Wendy Kan. 2015. Facebook Recruiting IV: Human or Robot? https://kaggle.com/competitions/facebook-recruiting-iv-human-or-bot

[12] Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. 2020. AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data. *arXiv preprint arXiv:2003.06505* (2020).

[13] Tanya Goyal, Junyi Jessy Li, and Greg Durrett. 2022. News summarization and evaluation in the era of gpt-3. *arXiv preprint arXiv:2209.12356* (2022).

[14] Siyuan Guo, Cheng Deng, Ying Wen, Hechang Chen, Yi Chang, and Jun Wang. 2024. DS-Agent: Automated Data Science by Empowering Large Language Models with Case-Based Reasoning. *arXiv preprint arXiv:2402.17453* (2024).

[15] Ivan Guz, night_bat, and Wendy Kan. 2015. Avito Context Ad Clicks. https://kaggle.com/competitions/avito-context-ad-clicks

[16] Sungwon Han, Jinsung Yoon, Sercan O Arik, and Tomas Pfister. 2024. Large Language Models Can Automatically Engineer Features for Few-Shot Tabular Learning. *arXiv preprint arXiv:2404.09491* (2024).

[17] Noah Hollmann, Samuel Müller, and Frank Hutter. 2024. Large language models for automated data science: Introducing caafe for context-aware automated feature engineering. *Advances in Neural Information Processing Systems* 36 (2024).

[18] Franziska Horn, Robert Pack, and Michael Rieger. 2020. The autofeat python library for automated feature engineering and selection. In *Machine Learning and Knowledge Discovery in Databases: International Workshops of ECML PKDD 2019, Würzburg, Germany, September 16–20, 2019, Proceedings, Part I*. Springer, 111–120.

[19] Addison Howard, Bernadette Bouchon-Meunier, IEEE CIS, inversion, John Lei, Lynn@Vesta, Marcus2010, and Prof. Hussein Abbass. 2019. IEEE-CIS Fraud Detection. https://kaggle.com/competitions/ieee-fraud-detection

[20] jeremy stanley, Meg Risdal, sharathrao, and Will Cukierski. 2017. Instacart Market Basket Analysis. https://kaggle.com/competitions/instacart-market-basket-analysis

[21] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088* (2024).

[22] Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. 2016. Field-aware factorization machines for CTR prediction. In *Proceedings of the 10th ACM conference on recommender systems*. 43–50.

[23] Kaggle. [n. d.]. Kaggle. https://www.kaggle.com

[24] Wendy Kan. 2015. West Nile Virus Prediction. https://kaggle.com/competitions/predict-west-nile-virus

[25] James Max Kanter and Kalyan Veeramachaneni. 2015. Deep feature synthesis: Towards automating data science endeavors. In *2015 IEEE international conference on data science and advanced analytics (DSAA)*. IEEE, 1–10.

[26] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems* 30 (2017).

[27] Albert Lu, Hongxin Zhang, Yanzhe Zhang, Xuezhi Wang, and Diyi Yang. 2023. Bounding the capabilities of large language models in open text generation with prompt constraints. *arXiv preprint arXiv:2302.09185* (2023).

[28] Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heintz, and Dan Roth. 2023. Recent advances in natural language processing via large pre-trained language models: A survey. *Comput. Surveys* 56, 2 (2023), 1–40.

[29] mjkistler, Ran Locar, Ronny Lempel, RoySassonOB, Rwagner, and Will Cukierski. 2016. Outbrain Click Prediction. https://kaggle.com/competitions/outbrain-click-prediction

[30] The pandas development team. 2020. *pandas-dev/pandas: Pandas*. https://doi.org/10.5281/zenodo.3509134

[31] Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International conference on data mining*. IEEE, 995–1000.

[32] Qitao Shi, Ya-Lin Zhang, Longfei Li, Xinxing Yang, Meng Li, and Jun Zhou. 2020. Safe: Scalable automatic feature engineering framework for industrial tasks. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 1645–1656.

[33] Ann Yuan, Andy Coenen, Emily Reif, and Daphne Ippolito. 2022. Wordcraft: story writing with large language models. In *27th International Conference on Intelligent User Interfaces*. 841–852.

[34] Shun Zhang, Zhenfang Chen, Yikang Shen, Mingyu Ding, Joshua B Tenenbaum, and Chuang Gan. 2023. Planning with large language models for code generation. *arXiv preprint arXiv:2303.05510* (2023).

[35] Tianyi Zhang, Faisal Ladhak, Esin Durmus, Percy Liang, Kathleen McKeown, and Tatsunori B Hashimoto. 2024. Benchmarking large language models for news summarization. *Transactions of the Association for Computational Linguistics* 12 (2024), 39–57.

[36] Tianping Zhang, Zheyu Zhang, Haoyan Luo, Fengyuan Liu, Wei Cao, and Jian Li. 2022. Openfe: Automated feature generation beyond expert-level performance. (2022).