Unveiling Data Preprocessing Patterns in Computational Notebooks

Valentina Golendukhina
Computer Science Department
University of Innsbruck
Innsbruck, Austria
valentina.golendukhina@uibk.ac.at

Michael Felderer
German Aerospace Center (DLR)
University of Cologne
Cologne, Germany
michael.felderer@dlr.de

***Abstract*—Data preprocessing, which includes data integration, cleaning, and transformation, is often a time and effort-intensive step due to its fundamental importance. This crucial phase is integral for ensuring the quality and suitability of data for subsequent stages, such as feature engineering and model training. This paper explores the current state of data preprocessing in the context of Machine Learning (ML) and data-driven systems. With a focus on Python-based notebooks, we investigate how prevalent data preparation practices are in computational notebooks focused on ML model development. The paper presents the results of the analysis of 149,048 computational notebooks collected from Kaggle, a platform hosting data science competitions. Despite the crucial role played by data preprocessing in guaranteeing the effectiveness of model performance, our results expose a significant lack of emphasis on data preprocessing activities in the examined notebooks. Notably, users holding the highest rankings tend to skip data preprocessing steps and focus on model-related activities. Although other users exhibit more frequent incorporation of data preprocessing methods, the overall prevalence remains relatively limited. We discovered that data preparation practices such as missing values are present in 20% to 60% of the notebooks depending on a competition, whereas outliers handling are only present in less than 20% of the analyzed scripts. The most frequently and consistently applied practices are the data transformation methods.**

***Index Terms*—Data Quality, Data Cleaning, Machine Learning**

I. Introduction

With the advances and implementation of Machine Learning (ML) and data-driven systems in various industries and domains, the importance of ensuring the quality of input data has become increasingly pronounced. As these technologies enter fields ranging from healthcare to finance, the need for a comprehensive understanding of data preparation becomes imperative. Data preprocessing, also known as data wrangling, is a process of bringing the raw data into an appropriate form for further usage including such steps as data integration, data cleaning, and data transformation [1]. This step, highly important for the final results, is often both time and effort-intensive [2].

The transition from traditional software engineering to the development of data-driven systems, featuring ML models, results in new software referred to as *Software 2.0* [3]. This paradigm shift not only encompasses alterations in the development lifecycle and the libraries employed but also introduces changes in the tools used throughout the process. Notably, computational notebooks have emerged as a tool of choice for data scientists and practitioners engaged in data analysis and model construction, including Jupyter notebooks[1] [4]. The notebooks integrate code, visualizations, and markdown text within the same environment allowing developers to achieve faster feedback and a greater level of process understanding.

Due to a rather model-oriented approach to ML development, many studies have focused on the analysis of ML models and ML development workflows [3], [5], [6], whereas less focus has been on data preprocessing. Studying the data preparation process is essential not only for comprehending the underlying patterns but also for enabling requirements engineers to more effectively formulate data-related specifications concerning data collection, data formats, and data range considerations [7]. Moreover, in a landscape where ML models are increasingly trained on vast open-source datasets including computational notebooks [5], [8], there arises a critical need to assess the quality and characteristics of such data.

In this research, we aim to provide a baseline for further examination of the data preprocessing stage by collecting and providing an extensive list of data preprocessing functions for the keyword-based static code analysis. Furthermore, we applied the keyword-based approach to answer the following research questions:

1) What data preprocessing methods are the most applied for tabular data?
2) Do data preprocessing practices correlate with the experience of the ML developers?
3) What are the variations observed across different competitions on Kaggle?

In our investigation, we prioritize addressing context-independent errors such as duplicates and missing values and aim to analyze data preprocessing as an integral part of the ML development flow, executed within the same file as the model construction. Examining competition notebooks serves as a practical illustration, offering a trade-off between more complex projects tailored for real-world applications and those

This work was supported by the Austrian Research Promotion Agency (FFG) in the frame of the project ConTest [888127]

[1]https://jupyter.org

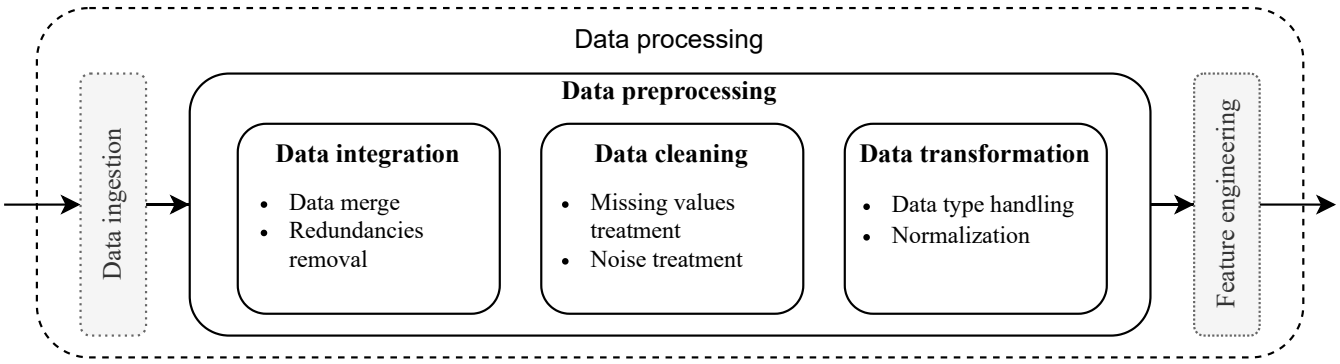

Fig. 1. Typical steps of data preparation workflow with corresponding activities

designed as assignment notebooks or toy projects and provides practices of both experienced ML developers and beginners.

The remainder of this paper is structured as follows. Section II introduces the main concepts and reviews the related work. Section III provides the methodology of data selection, collection, and analysis procedures. Subsequently, Section IV presents the results derived from the analysis. Section V discusses the findings, highlights future work, and identifies potential threats to validity. Finally, Section VI summarizes the results and concludes the paper.

II. Background and Related Work

This section provides background information on key aspects: data cleaning and preparation, computational notebooks such as Jupyter notebooks, and the Kaggle platform (Sections II-A and II-B). Following the background, Section II-C presents an overview of earlier work related to this paper.

A. Data Preparation

Data preprocessing is one of the most time and effort-intensive stages of the data processing pipeline [1]. This stage contains several tasks such as data integration, cleaning, and transformation, and prepares the data for subsequent stages, including feature engineering and model training. Figure 1 shows a typical data processing pipeline. Depending on the type of data, different data preprocessing is required to ensure data quality. For the purposes of this paper, we focus on structured and semi-structured data and do not cover issues associated with labeling.

According to Abedjan et al., there are four types of data errors: outliers, duplicates, rule violations, and pattern violations [9]. Outliers refer to data points that deviate from the established distribution of the dataset, while duplicates involve fully or partially replicated entities representing the same real-world entity. Rule violations are violations of integrity constraints, including such issues as missing values. Pattern violations encompass values that violate syntactic and semantic constraints, such as those related to data types or formatting.

In this study, we focus on context-independent issues including missing values, duplicated values, outliers, and data type issues, thus covering data integration, cleaning, and transformation stages.

B. Computational Notebooks and Kaggle

A computational notebook is a coding environment that offers interactive tools for collaborative data analysis and combines executable scripts, immediate code outputs, visualizations, and written text [10]. One type of computational notebook is Jupyter notebook widely adopted by data scientists and ML engineers [4] due to its interactive and collaborative nature.

Kaggle[2] is a platform for data science competitions where individuals and teams of different experience levels compete and learn to develop predictive models for a given dataset. It provides datasets, a cloud-based coding environment, and a collaboration and knowledge-sharing community of data scientists and machine learning practitioners. The Kaggle platform facilitates the creation and execution of scripts and computational notebooks authored in either R or Python within its work environment encouraging the participants to learn from and reuse computational notebooks.

A ranking system within Kaggle provides an understanding of users' skills and experience. There are currently five tiers: novice, contributor, expert, master, and grandmaster. The progress is calculated based on tournaments and challenges won, submitted notebooks, and their quality.

Mining Jupyter notebooks is a commonplace practice for gaining insights not only into characteristics of computational notebooks [11], [12] but also into prevailing programming practices [5], [6]. By examining Jupyter notebooks, researchers can collect valuable information about data preprocessing, model development, and other aspects of the computational workflow [5], [13], while also uncovering broader trends in coding practices within the programming community.

C. Related Work

Several authors analyzed distinct data preprocessing steps by computational notebook mining. Yan and He [8] crawled through four million Jupyter notebooks to analyze data frame

[2]www.kaggle.com

transformations and data preparation choices of data scientists to create an automated data preparation-oriented recommendation system specifically focusing on seven API calls from Pandas library. Negrini et al. [14] proposed a static analysis method for tracking data transformations focusing on Pandas data frame transformations. By examining and executing public notebooks, Yang et al. [13] investigated data leakage in notebooks. Their analysis included Kaggle notebooks and showed that more than 55% of competition notebooks tend to have preprocessing leakage due to data transformation issues.

Several authors focused on the analysis of ML workflows including the data preparation stage. Dilhara et al. [3] analyzed ML libraries usage, challenges, and trends within the ML development workflow. They identified the most popular ML libraries used for data cleaning (Keras, Sklearn, and Tensorflow) and analyzed the libraries within the development stages by searching for certain API calls. However, the main focus of the research was on the model-related stages, whereas only 12 data-related APIs were identified. Braiek et al. [15] examined the state of ML frameworks integration in open source repositories, the contributions of companies and the ML community, and top-10 company and community-driven frameworks. Ramasamy et al. [5] manually annotated the data science steps of each cell in 470 Jupyter notebooks from GitHub repositories and assigned them to one of the ML development stages including data-related steps such as data exploration and data preprocessing.

Several papers investigated the state of ML development by analyzing computational notebooks. Psallidas et al. [6] analyzed over 8 million notebooks and 2 million enterprise ML pipelines developed within Microsoft to investigate the most used ML libraries, top learners, and transformers. Choetkiertikul et al. [10] mined Jupyter notebooks to understand the characteristics of data science projects and their connection to the ranking of notebook contributors.

To the best of our knowledge, this paper is the first study to comprehensively collect data preprocessing API calls and incorporate all data preprocessing stages in the analysis, aiming to estimate the extent of their application within the domain of ML model development.

III. Methodology

Initially, we analyzed over 200,000 notebooks from Kaggle. In this section, we discuss the data collection procedures and data analysis methods applied in the study.

A. Data Collection

In our analysis, we use publicly available computational notebooks from Kaggle with scripts written in Python, which has currently shown to be the most used language for data science and ML projects [15], and in Jupyter notebooks format, which is the primary environment for data science developers utilizing ML components [4]. To enhance transparency and reproducibility of the results, we used the KGtorrent dataset [16] that comprises computational notebooks and their metadata collected from Kaggle and widely used in academia for computational notebooks mining [17], [18]. For our study, we retrieved information about the users, their experience tier, points, competition titles, script languages, and if a notebook was forked.

TABLE I
Categories of data preprocessing activities

Category	Description
Duplicates	APIs in this category focus on identifying and handling duplicated entries within datasets, ensuring data integrity by addressing redundancy.
Index	Functions that manage and manipulate the index of data frames, including setting, resetting, and modifying index values for efficient data access.
Missing values	Functions designed to detect, impute, or remove missing or null values in the dataset to maintain data completeness and accuracy.
NLP	Functions that preprocess text data.
Outlier	Keywords for 'Outlier' aim to identify lines of code handling outliers - data points that deviate significantly from the expected distribution.
Statistical	This category describes statistical data transformation APIs for scaling and normalization of values.
Transformation	APIs in the 'Transform' category specialize in data transformation processes, enabling the conversion or manipulation of data to meet specific requirements or formats.
Type	APIs categorized under 'Type' address issues related to data types, ensuring consistency and correctness by managing the format and representation of different data types within datasets.
Uniqueness	Functions that ensure that certain fields or records are unique within the dataset.
Other	This category encompasses APIs that can address more than one data issue including such methods as `.replace, .map, .apply`.

To filter toy projects, and tutorials, and avoid copied projects, we established several inclusion criteria:

- The file has an extension `.ipynb`.
- The notebook has at least 10 cells of code to filter out empty and unfinished notebooks.
- The notebook was not forked from another repository to avoid repetitive notebooks.
- Python is a declared programming language.
- Input data is in tabular form.

After analyzing the main packages used in the notebooks, we discovered that Pandas is one of the most frequently used packages (used in more than 90% of all projects) and can be considered as an identifier for projects focusing on tabular data.

After applying inclusion criteria to 250,000 randomly selected notebooks, 184,570 notebooks were left for analysis. For the code analysis extraction, we followed the procedure described in the study by Psallidas et al. [6]. Since Jupyter supports different cell types such as text and code, we parsed the downloaded files and extracted code cells from the `json` files. After the extraction, we could estimate the number of lines of code and code cells in each notebook. To analyze the code patterns, we omitted the output cell and markdown text but included commented code in the code cells.

B. API Calls Collection

The collection of the data preprocessing API calls was performed in two rounds. Firstly, we looked into 20 popular ML libraries identified by Braiek et al. [15] to find data-related APIs. Furthermore, we identified the most popular libraries within the collected notebooks, such as Pandas and Numpy, for further manual labeling. Secondly, we went through 100 randomly selected notebooks with at least 50 lines of code to identify missing APIs. All data cleaning methods were assigned to categories depending on the type of data issue they target resulting in 79 preprocessing functions assigned to 10 categories described in Table I: duplicates, index, missing values, natural language processing (NLP), outliers, statistical, transformation, type, uniqueness, and other. Some of these issues can be addressed by methods provided in libraries, e.g., `.dropna` or `.drop_duplicates` are standard APIs for missing values and duplicate handling. However, other issues such as outliers do not have a well-known standardized solution. Such issues were searched using keywords, for example, outlier handling lines were identified with keywords such as `outlier, quantile, percentile, iqr` and `zscore`. Notably, most of the collected APIs for data preprocessing come from the Pandas library (36).

C. Data Analysis

Our analysis included several steps. Firstly, after the code extraction from computational notebooks, we explored the data and conducted a high-level examination of notebook properties and characteristics to see the general insights, tendencies, and variability. Secondly, we conducted a fine-grained analysis of the preprocessing methods applied and the types of data issues addressed by searching for the identified data preprocessing-related APIs and categories. To see the differences among different datasets, we then compared the results for different competitions. Finally, we analyzed in detail how the missing values are handled in the notebooks. The list of the data preprocessing methods collected for the analysis and the analysis script can be found in the replication package[3].

IV. Results

During the initial analysis of the notebook code, we found that 24% of all notebooks were generated by a Kaggle bot, it automatically provides a distribution and correlation analysis of datasets for quick initial insights. These bot-generated notebooks were excluded from further analysis, leaving us with 138,376 notebooks. Additionally, we discovered that only 62% of the remaining notebooks contained an ML model, while the rest were created solely for data exploration purposes. Consequently, we analyzed these two groups separately to better understand their distinct preprocessing patterns and usage contexts.

[3]https://zenodo.org/records/11396773

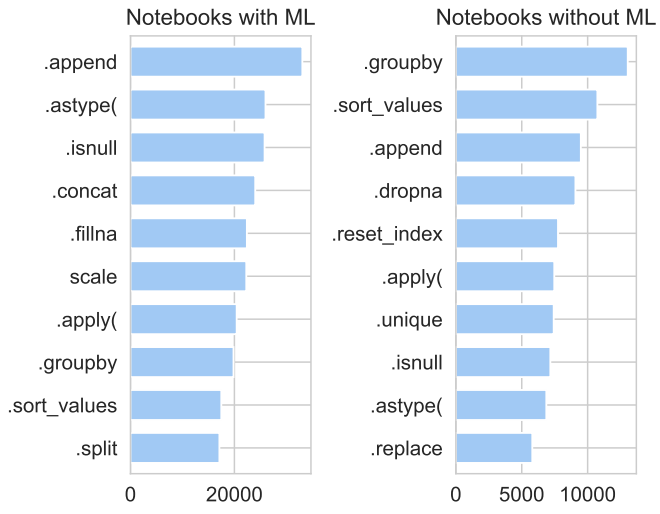

Fig. 2. Top-10 data preprocessing methods used in the computational notebooks with and without ML models

A. RQ1: Data Cleaning Methods

To understand which data issues are being addressed the most frequently and which methods are applied, we searched for data preprocessing-related methods in every collected computational notebook. For the identified methods from different libraries, we used regular expressions to find the methods being implemented. However, when there was no distinct method, we searched for keywords. The top-10 identified methods are presented in Figure 2. We found methods from all categories in the notebooks, except for the statistical and outliers categories where only related keywords were found in the code. Next, we describe the main findings from each category. The complete distribution of methods can be found in the replication package.

1) Transformation: The transformation category stands out as having the largest number of identified methods, showing a higher need for data transformation processes in notebooks. In notebooks oriented towards ML, the top five data transformation methods include `.append` (38%), `.concat` (27%), `.groupby` (23%), and `.split` (10%). In notebooks without ML models, the primary data transformation methods comprise similar methods in a different order with `.groupby` (26%) being the most found, followed by `.sort_values` (21%), `.append` (19%), and `.split` (10%).

2) Type: In ML notebooks, the prominence of the `.astype` method stands out, constituting a substantial proportion at 30%, which is notably larger than other type-related methods including `.to_numeric` (1%), `.to_categorical` (2%), and `.to_datetime` (6%). In notebooks without ML models, the usage of type-related methods is less frequent, with `.astype` indicated in 13% of notebooks and `.to_datetime` found in 9% of notebooks.

A detailed analysis of the `.astype` function revealed that its application is not highly varied. In 46% of all cases, it was used to convert variables into integers. Another 28% of

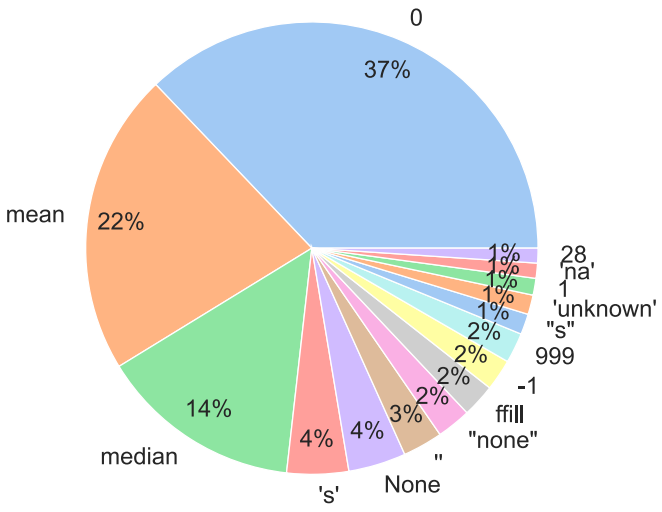

Fig. 3. Top-15 missing values fillers

instances involved converting variables to float values, while 12% of the cases were for converting to strings. Conversions to categorical variables accounted for only 7% of the applications, and boolean conversions were the least common, making up just 2% of the cases.

3) Missing values: For notebooks with ML models, prevalent methods included `.isnull` (29%), `.fillna` (25%), and `.dropna` (12%), indicating that a focus on addressing missing values in the preprocessing stage is present in less than a third of notebooks. Interestingly, notebooks with ML models exhibited higher usage percentages for these methods compared to notebooks without ML models. The latter notebooks demonstrated a lower usage of `.fillna` (11%), but higher application of `.dropna` (18%) method.

We analyzed in more detail how `.fillna` is applied, the top-15 most frequently used values to fill missing values are presented in Figure 3. Among these, 37% of the instances utilized the value 0, and 22% employed the mean value of the column. Additionally, 14% of cases used the median. The remaining cases involved string values such as 's', 'unknown', 'na', dummy values such as 999, or an empty string.

4) Uniqueness: Unlike the duplicates, uniqueness-checking functions are more frequently applied. `.unique` is used in with 18% and 14% in notebooks containing ML models and notebooks without ML models respectively and is applied frequently for the indexes validation.

5) Index: Application of index-related methods signalizes data integration activities such as the unification of disparate datasets based on common indices or keys, dataset alignment and merge, as well as aggregation and grouping. Index operations show similar results for both types of notebooks with 17% and 15% for `reset_index` and similar results for the other APIs such as `.set_index` (6%) and `.reindex` (1%).

6) Statistical: The statistical methods indicate the biggest difference between the types of notebooks. In notebooks with ML models, the prevalent methods include `RobustScaler` (1%), `MinMaxScaler` (4%), and `StandardScaler` (8%). The keywords `normalize` and `scale` are found in (12%) and (25%) of notebooks respectively. Unlike model-oriented notebooks, we did not find any scaling or normalization methods in notebooks without ML models. The keywords `normalize` and `scale` appear in a smaller percentage of notebooks: 4% and 8% which can indicate both less focus on data normalization methods for non-ML model-oriented notebooks or application of custom scale and normalization techniques for the data.

7) Duplicates, NLP, Outliers: These categories belong to the methods the least represented in the code. Duplicates handling functions such as `.drop_duplicates` and `.duplicated` only appear in 1% to 3% of the analyzed notebooks. This group of API calls is the least represented in the notebooks.

The outliers category, where only keywords were identified, is the second least represented category overall. Mentions of outliers appear in only 5% of the notebooks with ML models and 2% of those without.

Logically, NLP functions are not frequently found in the notebooks, which can be explained by the specific nature of data required for these methods; they are designed to work with natural language text, whereas a significant proportion of data on Kaggle is tabular and numerical. Nevertheless, the most frequently used NLP method is `.lower`, appearing in 7% of the notebooks with ML models and 4% of those without.

8) Other: In notebooks with ML models, various other data manipulation methods were identified, including `.where` (6%), `.map` (15%), `.replace` (15%), and `.apply` (23%). In notebooks without ML models, similar methods were found, albeit with lower usage percentages: `.where` (2%), `.map` (8%), `.replace` (11%), and `.apply` (15%). The differences in usage percentages suggest that while certain methods are common across both types of notebooks, ML-focused notebooks may exhibit a higher prevalence of specific techniques such as `.map` and `.apply`, which are often used for complex transformations and feature engineering in ML scenarios.

B. RQ2: Data Handling by Ranking

To understand if there are differences in data handling among the users from different performance tiers, we analyzed notebooks in each group. The result of the analysis is presented in Table II. The provided table presents the percentage of notebooks with specific methods or keywords from data preprocessing categories across different user tiers, ranging from novice (Tier 1) to grandmaster (Tier 5). Overall, the analysis included 46,694 notebooks created by novices, 16,536 notebooks of contributors, 4,202 notebooks of experts, 1,921 masters' notebooks, and 554 grandmaster notebooks which aligns with the increasing number of users in each tier. 79,125 notebooks did not have the tier indicator.

TABLE II
PERCENTAGE OF NOTEBOOKS WITH IDENTIFIED METHODS FROM A CORRESPONDING DATA PROCESSING CATEGORY IN GROUPS OF USERS WITH DIFFERENT RANKINGS WHERE 1 IS NOVICE, 2 IS CONTRIBUTOR, 3 IS EXPERT, 4 IS MASTER, AND 5 IS GRANDMASTER.

Tier	Duplicates	Index	NA	NLP	Other	Outlier	Statistic	Transform	Type	Uniqueness
1	0.045	0.211	0.449	0.089	0.396	0.073	0.272	**0.717**	0.329	0.221
2	0.051	0.248	0.402	0.102	0.404	0.070	0.268	**0.722**	0.334	0.238
3	0.038	0.290	0.362	0.095	0.386	0.067	0.307	**0.726**	0.342	0.235
4	0.040	0.336	0.371	0.098	0.369	0.073	0.290	**0.720**	0.366	0.246
5	0.006	0.093	0.197	0.048	0.186	0.013	0.099	**0.326**	0.130	0.125

Notably, the data preparation behavior of users from the first four tiers does not show large differences, unlike the results of the grandmasters tier. The latter group shows a significantly lower percentage of notebooks containing data-related activities. This applies to the methods from the majority of the categories: duplicates, NLP, other, outliers, statistic, transformation, and uniqueness.

Transformation methods are the most consistently utilized among all notebooks, indicating a common practice irrespective of the user expertise. They were found in the majority of notebooks, more than 70%, for all users except grandmasters, who use transformation methods in 33% of the notebooks.

The index category is the category that shows significantly increasing application of indexing methods among users from tiers 1, 2, 3, and 4, indicating a potential correlation between expertise and index manipulation in data handling.

On the contrary, missing values handling methods show a decreasing trend. In the novice users (Tier 1) group, there is a higher percentage (45%) of notebooks addressing missing values. As expertise increases, this percentage decreases across higher tiers and drops to 20% for grandmasters.

Overall, **the most identified category** across all tiers is *transformation* followed by *missing values, statistical*, and *other*. **The rarest categories** are *outliers, indexes*, and *duplicates*.

Despite observed trending differences in data preprocessing practices among different user tiers in computational notebooks, Spearman's rank correlation analysis indicated no significant correlation larger than 0.35 for the given data. This suggests that the identified trends in data handling and cleaning across user tiers may not exhibit strong linear relationships. Moreover, users from tiers 1-4 show very similar behavior towards data preprocessing methods.

C. RQ3: Differences in Competitions

Lastly, to determine if the low application of certain methods is specific to particular datasets or if it varies across different competitions, we compared data preprocessing category distribution among the sets of notebooks created for specific competitions. We manually selected 30 competitions with structured and semi-structured datasets. For each competition, we calculated the percentage of notebooks containing data preprocessing methods of corresponding categories. The aggregation of the results among 30 projects is shown in Figure 4. The analysis reveals that the application of data preprocessing methods is consistently medium to low across most competitions.

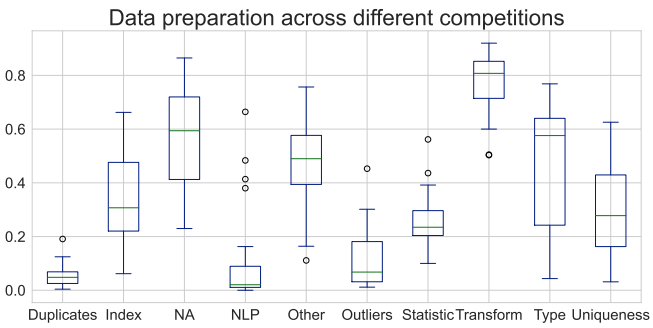

Fig. 4. Distribution of data preprocessing steps among 30 competitions.

The transformation category emerged as the most frequently employed in notebooks in all analyzed competitions, which confirms the emphasis on transforming and reshaping data to meet specific requirements. Notably, outlier detection exhibited consistently low usage across all competitions except for an outlier competition where outlier-related keywords were mentioned in 40% of the notebooks. The handling of indexes, missing values, and type categories exhibited high variance across competitions, suggesting a contextual dependency on the specific requirements and characteristics in each competition.

Strong correlations were observed between certain data preparation categories. Notably, uniqueness and index handling demonstrated a correlation of 0.76, implying that projects containing uniqueness checks also tend to have index handling methods. Similarly, a strong positive correlation of 0.74 was found between index handling and the transformation category, since transformation activities are often followed by reindexing methods.

V. DISCUSSION

In the discussion section, we highlight and discuss the key findings of our research.

Data preprocessing in notebooks: The analysis reveals that the code within computational notebooks on Kaggle often lacks extensive lines dedicated to data preprocessing. This trend suggests that data preparation is a relatively underemphasized aspect of the notebook development process.

After analyzing notebooks created for ML models training and notebooks without ML models, there is little relative difference in the share of data preparation code within the notebooks. However, ML-focused notebooks tend to employ

more sophisticated methods, such as scaling and normalization, prior to model training. This implies a shift towards comprehensive data preprocessing in ML-centric workflows, although data preprocessing steps in both types of notebooks on average do not exceed 10%.

In our analysis, data preprocessing was considered as all lines before model training, which cannot predict all data preprocessing steps since ML workflow is not always a linear process with data preprocessing followed by ML training and earlier stages might repeat later in the code [19]. Thus, our finding shows even lower results than Ramasamy et al. who found that data preprocessing activities take 23.5% of an ML workflow [5].

Notably, duplicates and outliers handling methods are the least applied in the notebooks, while the most frequently encountered APIs focus on dataset transformation, missing values, and data types handling. Moreover, we discovered poor practices while examining the missing values handling including the replacement of missing values with string values and 0, which accounts for more than half of all analyzed code samples. Such replacements, also known as data smells [20], while common, are generally considered poor practices as they can introduce inconsistencies and inaccuracies into the dataset and lead to errors in the data analysis.

The percentages of notebooks featuring duplicates, indexes, missing values, and type handling vary depending on the competition, indicating a contextual dependence on domain and data type. While these findings hint at potential trends, further investigation is required to establish more concrete patterns in data handling across different domains.

User Ranking Impact: Except for grandmaster users, there is no significant difference in data handling among users with different rankings. Grandmasters stand out by applying data preprocessing functions less frequently, proceeding directly to model training. These results confirm the results of the study that showed that both beginners and expert developers are prone to use poor data preprocessing practices leading to data leakage problems [13]. This divergence can be explained in several ways. Experienced users might perform their data exploration and cleaning activities off Kaggle platform and upload only model-related code. On the other hand, it underlines the model-centric approach dominating the Kaggle community.

Another reason, why data handling steps are skipped is the nature of Kaggle competitions and datasets. Some datasets are provided in the preprocessed form, whereas users are encouraged to focus solely on the model building. However, such data processing steps are not documented leaving room for potentially not detected data issues. Additionally, not all provided datasets are curated and an increasing percentage of notebooks applying data preprocessing methods in some competitions implies a need for data cleaning in corresponding datasets. Nevertheless, even in such competitions missing values handling does not exceed 90% of all notebooks (60% on average).

A. Research Implications

There are several practical implications for researchers and practitioners.

While it is a common practice to reference the code of higher-ranked users for learning purposes, our findings indicate a noteworthy observation regarding grandmasters, the highest-ranked users on Kaggle. Contrary to the expectation that top-tier users might exemplify comprehensive data preprocessing practices, our analysis reveals that the codes of grandmasters might often lack essential data integration, cleaning, and transformation steps. While it is common to reference higher-ranked users' code for learning, they may not include all necessary steps for thorough ML training and often place less emphasis on data validation.

Another implication of our findings pertains to the rising popularity of ML models trained on open-source notebooks [8], [18]. As these models learn from the information available on platforms like Kaggle including diverse datasets and code snippets, our findings highlight a deficiency in representing the data preprocessing steps within these open-source datasets. Understanding open-source notebooks may shed some light on the limitations and characteristics of the trained models to improve interpretability and explainability. It is also important to emphasize the need for improved documentation and transparency in the data preprocessing phase of open-source notebooks.

B. Threats to Validity

To mitigate potential sampling bias, an expansive sample of over 200,000 notebooks was analyzed. Nonetheless, it is important to note an inherent imbalance in the distribution of notebooks, with a prevalence of projects by beginners compared to those by higher-level users.

During the API calls collection process, we examined all major libraries and enriched the final list with the manual examination of the notebooks. While acknowledging that our compilation may not include every existing method, our concerted effort aimed to cover the primary and prevalent data preprocessing practices for tabular data.

The results of this study can only be generalized to notebooks aiming at ML model development. The primary focus on developing precise and accurate models on Kaggle contrasts with the multifaceted requirements of ML code intended for real-world products. In a production setting, ML code necessitates additional data preprocessing steps, often tailored to specific domains and encompassing considerations related to larger volumes of data. Furthermore, the characteristics of the Kaggle users might differ from the broader population of data scientists or machine learning practitioners.

C. Future Work

One of the promising directions for future work involves an in-depth exploration of preprocessing activities across diverse domains. Since this study only covered tabular data, different data types such as visual data should be investigated as

well. Moreover, extending the research to target ML in real-world products and production pipelines would be beneficial. This exploration can provide insights into the distinctive preprocessing challenges and practices that arise in practical, applied settings, offering valuable guidance for practitioners navigating the transition from model-centric environments, such as Kaggle, to deployment in real-world scenarios.

Finally, the robustness of different ML models to non-contextual data issues, such as missing values or duplicates, can be further investigated. Conducting experiments to gauge the extent to which models withstand these challenges will contribute to a nuanced understanding of model resilience and guide the development of strategies to enhance robustness in the face of diverse data anomalies.

VI. Conclusion

This paper presents our findings from the examination of data preprocessing practices in computational notebooks from Kaggle. We based our analysis on 138,376 notebooks to understand the trends in the landscape of ML model development and data exploration. To analyze the data preprocessing steps, we performed a keyword-based analysis based on the list of 79 data preprocessing-related API calls.

Despite the foundational role of data preprocessing in ensuring robust model performance, our findings reveal a notable underrepresentation of data preprocessing activities within the analyzed notebooks. Particularly intriguing is the observation that users with the highest rankings tend to bypass data preprocessing steps. While other users demonstrate a higher frequency of implementing data preprocessing methods, the overall prevalence remains modest.

This study underscores the contextual nature of data preprocessing, revealing a dependency on the specific dataset and competition in which users engage. Users tailor their approach based on the distinct characteristics and challenges posed by Kaggle competitions.

As the field of machine learning continues to evolve, our findings prompt further inquiry into the factors influencing the observed patterns and the potential implications for model performance and interpretability. Future research should explore the data preprocessing choices for different data types and investigate how these practices align with real-world ML development scenarios. Ultimately, our study contributes valuable insights into the dynamics of data preprocessing within the Kaggle community, offering a foundation for future investigations aimed at refining and optimizing the data science workflow.

References

[1] H. Foidl, V. Golendukhina, R. Ramler, and M. Felderer, "Data pipeline quality: Influencing factors, root causes of data-related issues, and processing problem areas for developers," *Journal of Systems and Software*, vol. 207, p. 111855, 2024.

[2] M. Steidl, V. Golendukhina, M. Felderer, and R. Ramler, "Automation and development effort in continuous ai development: A practitioners' survey," in *2023 49th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pp. 120–127, IEEE, 2023.

[3] M. Dilhara, A. Ketkar, and D. Dig, "Understanding software-2.0: A study of machine learning library usage and evolution," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 30, no. 4, pp. 1–42, 2021.

[4] J. M. Perkel, "Why jupyter is data scientists' computational notebook of choice," *Nature*, vol. 563, no. 7732, pp. 145–147, 2018.

[5] D. Ramasamy, C. Sarasua, A. Bacchelli, and A. Bernstein, "Workflow analysis of data science code in public github repositories," *Empirical Software Engineering*, vol. 28, no. 1, p. 7, 2023.

[6] F. Psallidas, Y. Zhu, B. Karlas, J. Henkel, M. Interlandi, S. Krishnan, B. Kroth, V. Emani, W. Wu, C. Zhang, *et al.*, "Data science through the looking glass: Analysis of millions of github notebooks and ml. net pipelines," *ACM SIGMOD Record*, vol. 51, no. 2, pp. 30–37, 2022.

[7] N. Nahar, H. Zhang, G. Lewis, S. Zhou, and C. Kästner, "A meta-summary of challenges in building products with ml components–collecting experiences from 4758+ practitioners," *arXiv preprint arXiv:2304.00078*, 2023.

[8] C. Yan and Y. He, "Auto-suggest: Learning-to-recommend data preparation steps using data science notebooks," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pp. 1539–1554, 2020.

[9] Z. Abedjan, X. Chu, D. Deng, R. C. Fernandez, I. F. Ilyas, M. Ouzzani, P. Papotti, M. Stonebraker, and N. Tang, "Detecting data errors: Where are we and what needs to be done?," *Proceedings of the VLDB Endowment*, vol. 9, no. 12, pp. 993–1004, 2016.

[10] M. Choetkiertikul, A. Hoonlor, C. Ragkhitwetsagul, S. Pongpaichet, T. Sunetnanta, T. Settewong, V. Jiravatvanich, and U. Kaewpichai, "Mining the characteristics of jupyter notebooks in data science projects," *arXiv preprint arXiv:2304.05325*, 2023.

[11] J. F. Pimentel, L. Murta, V. Braganholo, and J. Freire, "A large-scale study about quality and reproducibility of jupyter notebooks," in *2019 IEEE/ACM 16th international conference on mining software repositories (MSR)*, pp. 507–517, IEEE, 2019.

[12] A. Rule, A. Tabard, and J. D. Hollan, "Exploration and explanation in computational notebooks," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pp. 1–12, 2018.

[13] C. Yang, R. A. Brower-Sinning, G. Lewis, and C. Kästner, "Data leakage in notebooks: Static detection and better processes," in *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, pp. 1–12, 2022.

[14] L. Negrini, G. Shabadi, and C. Urban, "Static analysis of data transformations in jupyter notebooks," in *Proceedings of the 12th ACM SIGPLAN International Workshop on the State Of the Art in Program Analysis*, pp. 8–13, 2023.

[15] H. B. Braiek, F. Khomh, and B. Adams, "The open-closed principle of modern machine learning frameworks," in *Proceedings of the 15th International Conference on Mining Software Repositories*, pp. 353–363, 2018.

[16] L. Quaranta, F. Calefato, and F. Lanubile, "Kgtorrent: A dataset of python jupyter notebooks from kaggle," in *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*, pp. 550–554, IEEE, 2021.

[17] H. Dong, S. Zhou, J. L. Guo, and C. Kästner, "Splitting, renaming, removing: a study of common cleaning activities in jupyter notebooks," in *2021 36th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW)*, pp. 114–119, IEEE, 2021.

[18] S. Chen, N. Tang, J. Fan, X. Yan, C. Chai, G. Li, and X. Du, "Haipipe: Combining human-generated and machine-generated pipelines for data preparation," *Proceedings of the ACM on Management of Data*, vol. 1, no. 1, pp. 1–26, 2023.

[19] D. Ramasamy, C. Sarasua, A. Bacchelli, and A. Bernstein, "Visualising data science workflows to support third-party notebook comprehension: an empirical study," *Empirical Software Engineering*, vol. 28, no. 3, p. 58, 2023.

[20] H. Foidl, M. Felderer, and R. Ramler, "Data smells: categories, causes and consequences, and detection of suspicious data in ai-based systems," in *Proceedings of the 1st International Conference on AI Engineering: Software Engineering for AI*, pp. 229–239, 2022.