Feature Engineering in Machine Learning: Current Trends, Challenges, and Best Practices

Vijayalakshmi S Abbigeri [1], Geetha D Devanagavi [2]

School of Computing and Information Technology, REVA University, Bengaluru, India

Abstract

Feature engineering plays a critical role in the machine learning pipeline, profoundly impacting the performance of predictive models. This survey provides a comprehensive overview of the latest advancements in feature engineering, including its techniques, challenges, and best practices. It examines various methods for feature representation, selection, and extraction, outlining their strengths, limitations, and applications across different domains. Emphasis is placed on the importance of data quality and collection, as these are foundational to effective feature engineering. Additionally, the paper explores the emerging concept of "Software 2.0," which reimagines feature engineering as an integral aspect of modern software development.

Keywords: Feature engineering, feature representation, selection, extraction, and machine learning

1. Introduction

Machine learning (ML) has emerged as a transformative tool for tackling a wide range of challenges, from image recognition to natural language processing. A critical factor in the success of ML models is the quality and relevance of the input features [1]. Feature engineering—the process of creating, selecting, and transforming features from raw data, as illustrated in Fig. 1—is a foundational step in the ML pipeline. When executed effectively, feature engineering improves a model's capability to identify hidden patterns and relationships in data, ultimately enhancing its overall performance. [2]. Conversely, poorly engineered features can hinder model performance and lead to suboptimal results.

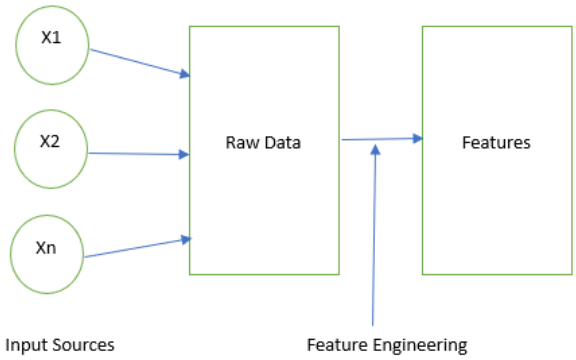

Fig 1: Feature Engineering

As datasets grow increasingly complex and diverse, the importance of feature engineering has become more pronounced. This survey provides a comprehensive overview of the current state of feature engineering research, discussing a range of techniques, challenges, and best practices. We explore various approaches to feature representation, selection, and extraction, highlighting their strengths, limitations, and applications across multiple domains. Additionally, the paper addresses the challenges associated with data collection and quality, emphasizing their critical role in ensuring effective feature engineering and optimal ML model performance.

Finally, we examine the emerging paradigm of "Software 2.0," which redefines feature engineering as a specialized software engineering discipline. This perspective emphasizes the need for dedicated tools,

frameworks, and methodologies to streamline the feature engineering process and uphold the quality of ML models in increasingly complex environments.

2. Feature Representation and Selection

2.1 Feature Representation

Feature representation involves converting raw data into a structured format that can be effectively utilized by machine learning models [3]. The techniques used for this process vary based on the type of data:

- **Numerical Data:** Techniques such as vectorization are commonly applied to represent numerical values effectively.
- **Textual Data:** Methods like tokenization, stemming, and lemmatization are employed to preprocess and structure textual information.
- **Visual Data:** Approaches such as edge detection, color histograms, and deep learning-based feature extraction are utilized to capture and represent visual patterns.

  Feature Representation is illustrated in Fig 2. It illustrates how different types of raw data (Numerical, Textual, and Visual) are processed using specific techniques to generate structured features, which are then used by the ML model.

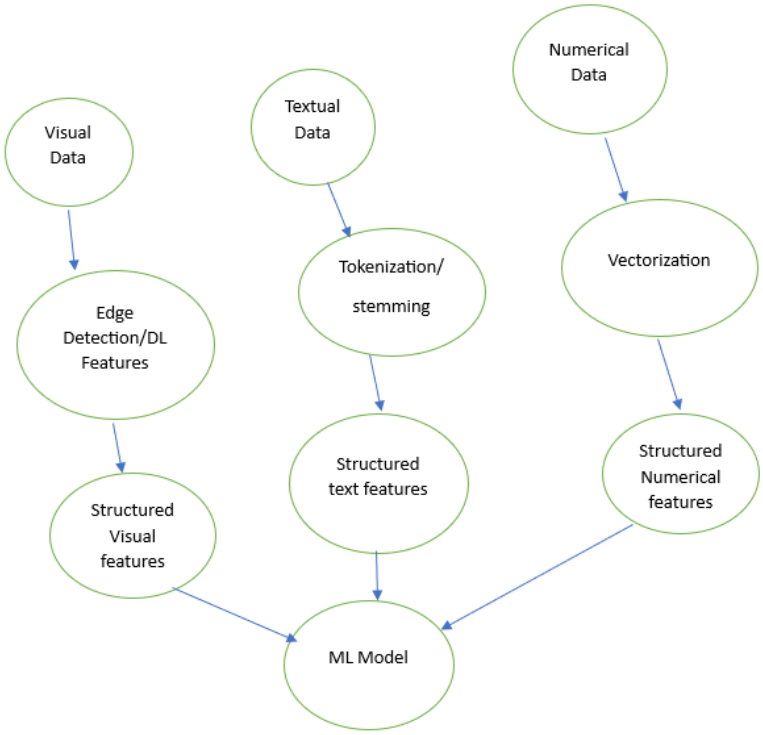

Fig 2: A diagram representing Feature Representation.

2.2 Feature Selection

Feature selection identifies the most relevant features for a specific task [4]. This step offers several key advantages:

- **Dimensionality Reduction:** Simplifies the model by reducing the number of features.
- **Improved Training Time:** With fewer features, models can be trained more efficiently.
- **Enhanced Generalization:** Minimizing irrelevant features reduces overfitting, enhancing performance on unseen data.

- **Increased Predictive Accuracy:** Relevant features lead to more accurate predictions.
- **Better Interpretability:** Models with fewer features are easier to analyze and understand.

Feature Selection is illustrated in Fig 3. It illustrates how raw features go through a selection process, resulting in a reduced feature set that enhances model performance by improving accuracy, training time, and interpretability.

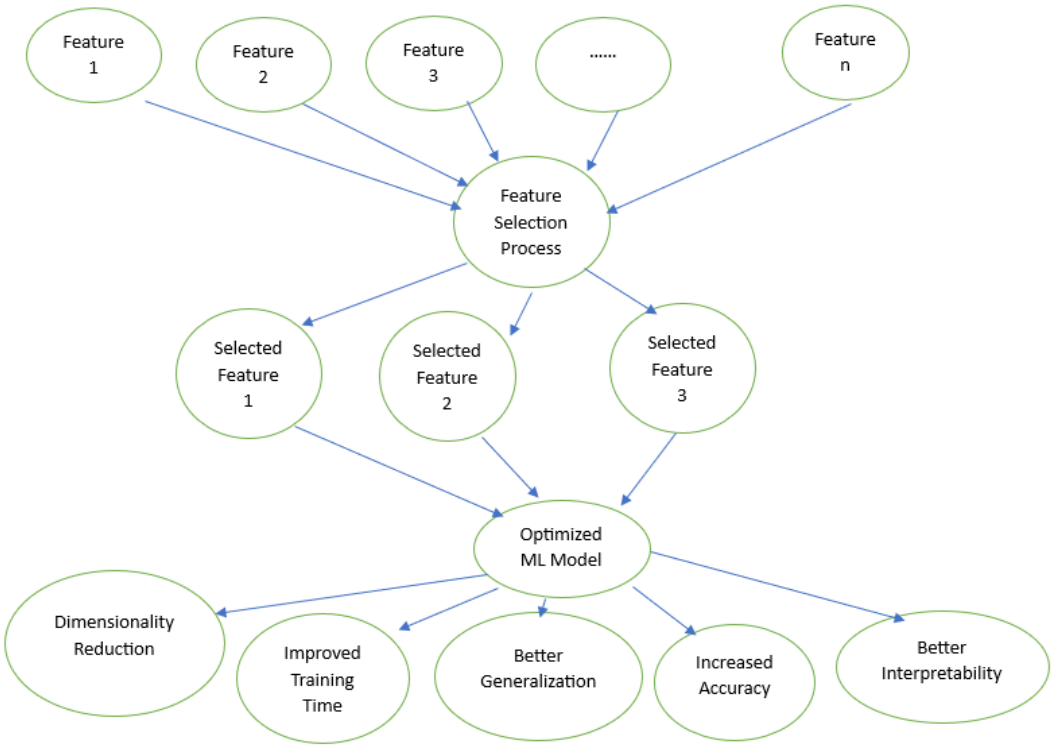

Fig 3: A diagram representing Feature Selection.

2.3 Feature Selection Techniques

- **Filter Methods:** Filter methods assess the significance of features by applying statistical measures such as correlation coefficients or mutual information, without relying on a specific machine learning model [5]. These techniques rank features based on their individual importance, without taking into account the relationships between features or the specific needs of the model. While filter methods are computationally efficient and capable of handling high-dimensional feature spaces, making them useful for initial feature selection [6], they may fail to identify complex dependencies or interactions between features that could be crucial for improving model performance.

- **Wrapper Methods:** Wrapper methods evaluate subsets of features by training and testing a machine learning model while accounting for feature interactions [7]. These techniques use the model's performance as the evaluation metric for feature selection, essentially "wrapping" around the model. While wrapper methods are more computationally intensive than filter methods, they are better at identifying the feature combinations that most enhance the model's performance [8]. Unlike filter methods, which evaluate features independently, wrapper methods account for how features work together to impact the model's predictive accuracy.

  Embedded Methods: Embedded methods integrate feature selection within the model training process, enabling the model to identify the most important features automatically [9]. These techniques combine feature selection with model optimization, enabling the model to identify which features are most useful for the task. By capturing complex feature interactions and dependencies, embedded methods offer a more effective and robust approach to feature selection compared to methods that rank features independently [10]. Since feature selection is

part of the training process, embedded methods can pinpoint the optimal feature subset that maximizes model performance, often leading to better accuracy and more relevant features than filter and wrapper methods. In Fig 4 A well-structured table summarizing the feature selection techniques are shown.

Feature Selection Technique	Description	Advantages	Disadvantages
Filter Methods	Assess feature significance using statistical measures (e.g., correlation, mutual information) without relying on a specific model.	Computationally efficient, suitable for high-dimensional data, independent of model choice.	Does not consider feature interactions, may miss complex dependencies.
Wrapper Methods	Evaluate feature subsets by training and testing a model, considering feature interactions.	Identifies optimal feature combinations, improves model performance.	Computationally expensive, especially for large datasets.
Embedded Methods	Integrate feature selection into model training, selecting features automatically.	Captures feature interactions, optimizes selection during training, improves accuracy.	Dependent on the chosen model, may require more tuning.

Fig 4: A table summarizing the feature selection techniques

2.4. Challenges and Recent Advancements

One of the key challenges in feature selection is ensuring that the chosen features generalize well across different datasets and are robust to variations in the data. Selecting a subset of features that performs well on the training set but fails to adapt to unseen data is a common pitfall. To tackle this, recent research has focused on optimizing the continuous embedding space for feature selection. This approach aims to refine the selection process by considering not just the relevance of individual features, but also how they interact in a more dynamic, high-dimensional space. By doing so, it seeks to identify feature subsets that are not only accurate but also more stable and adaptable to diverse conditions, ultimately leading to better generalization and robustness.

3. Feature Extraction and Engineering

3.1 Feature Extraction

Feature extraction is the process of transforming raw data into a set of meaningful features that can be used for machine learning. This often involves leveraging domain knowledge or unsupervised methods to identify patterns and relationships that are not immediately apparent in the original data[[11]. The goal is to extract more informative and relevant features, which can enhance the performance of machine learning models by providing them with a clearer representation of the underlying structure of the data.

3.2 Feature Engineering

Feature engineering is the process of designing and transforming features to improve the performance of machine learning models. It involves a combination of technical skill and creativity, requiring a deep understanding of the problem domain to represent the data in a way that best supports model learning. This process often includes modifying existing features, creating new ones, or selecting the most relevant features, all with the goal of enhancing the model's ability to make accurate predictions[2].

3.3 Techniques in Feature Engineering

Polynomial Feature Generation: This technique generates new features by capturing the nonlinear relationships between the original variables. It involves creating higher-order polynomial terms, such as squares, cubes, and cross-products, which can help the machine learning model better represent and capture the non-linear patterns in the data [12]. By expanding the feature space with these additional polynomial features, the model gains the ability to uncover intricate relationships that may not be immediately evident in the original feature set, leading to improved model performance. Polynomial Feature Generation is illustrated in Fig 5. It shows how original features undergo polynomial transformation to create higher-order terms, which are then used by the ML model for improved learning.

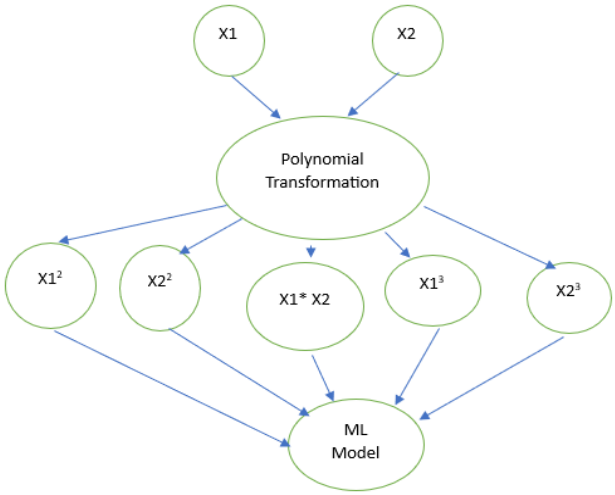

Fig 5: A diagram representing Polynomial Feature Generation.

Interaction Feature Design: This technique focuses on creating new features that capture the complex relationships and interdependencies between multiple input variables. By constructing interaction terms—such as products, ratios, or other mathematical combinations of the original features—the model can learn to identify and utilize the intricate interactions that may exist within the data [13]. This approach can significantly enhance predictive performance, as it enables the model to better capture the subtle patterns and dependencies underlying the problem. Interaction feature design is a key component of feature engineering, as it provides the model with a richer and more informative representation of the data, often improving accuracy, generalization, and interpretability. Interaction Feature Design is illustrated in Fig 6. It visually demonstrates how original features interact to create new features, which are then used in the machine learning model.

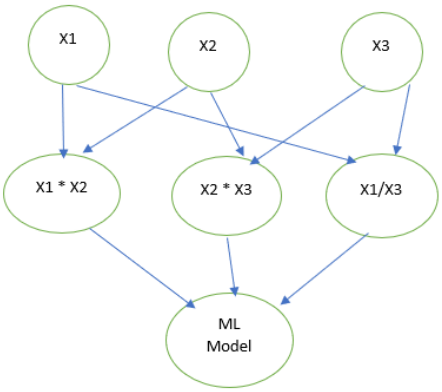

Fig 6: A diagram representing Interaction Feature Design.

Temporal Feature Incorporation: This technique focuses on creating features that capture time-dependent patterns and trends in the data. By incorporating temporal information, such as the timing or sequence of events, the model can learn to recognize the dynamic relationships and evolving patterns over time[14]. Examples of temporal features include variables like time of day, day of the week, time since the last event, and time-series lags or differences. By modeling these time-based dependencies, the model can better account for the temporal nature of the data, leading to improved predictive performance and the ability to learn more robust and generalizable patterns. Temporal Feature Incorporation is illustrated in Fig 7. It shows how time-related features are extracted from events and used to improve a machine learning model's performance.

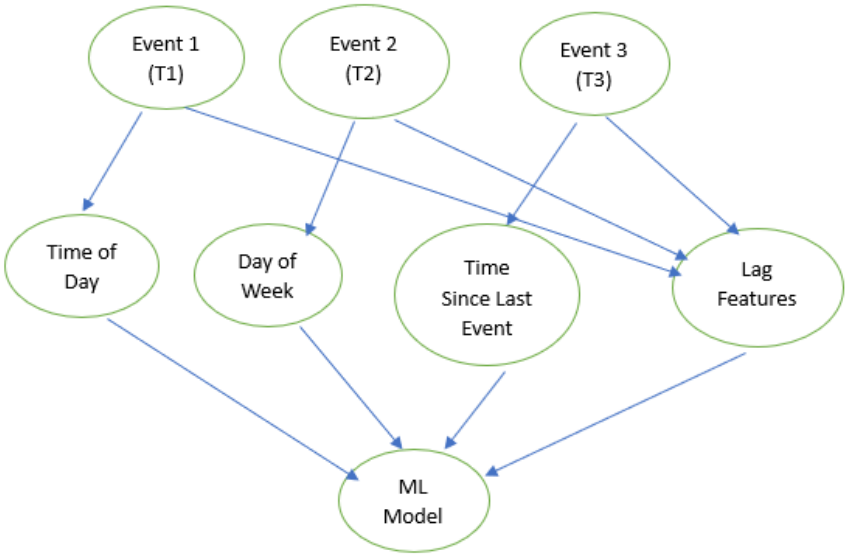

Fig 7: A diagram representing Temporal Feature Incorporation.

These techniques allow models to uncover complex relationships and underlying structures in the data, resulting in more accurate, robust, and interpretable outcomes. Successful feature engineering relies on a deep understanding of the domain, creative design of features, and the ability to leverage domain knowledge to identify the most informative and discriminative features. This combination of expertise and innovation is key to improving model performance and making better predictions.

3.4 Deep Learning in Feature Engineering

Deep learning provides advanced methods for feature engineering, offering new ways to select and transform features for improved model performance.

Deep Feature Selection: Utilizes deep learning techniques to automatically select the most relevant and informative features from the available data. By harnessing the powerful representational capabilities of deep neural networks, these methods can learn to pinpoint the most discriminative features that are essential for the specific machine learning task[5]. Deep feature selection goes beyond the limitations of traditional approaches by capturing complex, non-linear relationships and interactions between input features. As a result, these methods often outperform conventional feature selection techniques—such as filter, wrapper, and embedded methods—by identifying a more compact, effective feature set that maximizes predictive performance. Deep Feature Selection is illustrated in Fig 8. It illustrates how deep learning layers process input features to extract the most relevant ones for improved machine learning performance.

Fig 8: A diagram representing Deep Feature Selection.

Deep Feature Extraction: Employs deep learning techniques to automatically learn and extract informative, high-level features directly from raw data. By leveraging the powerful representation learning capabilities of deep neural networks, these methods can uncover complex, non-linear patterns and relationships within the input data. This leads to more discriminative and meaningful features for machine learning tasks[15]. Deep feature extraction models are able to learn rich, hierarchical representations that capture intricate structures and semantics in the data, often outperforming traditional feature engineering methods in terms of performance and generalization. This automated process significantly reduces the time and effort needed for manual feature engineering, making deep learning a highly promising approach for scalable and adaptive feature extraction. Deep Feature Extraction is illustrated in Fig 9. It illustrates how deep learning layers process raw data (such as images, text, or signals) to extract high-level features, which are then used for machine learning tasks.

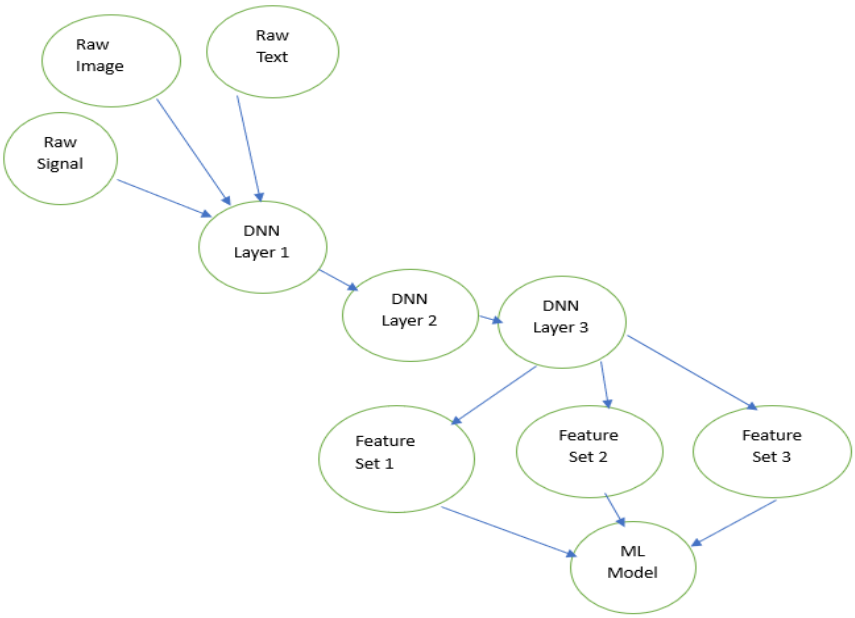

Fig 9: A diagram representing Deep Feature Extraction.

These methods often outperform traditional techniques by learning more informative and representative features directly from the data.

4. Data Quality and Collection

The quality and relevance of the input data are crucial for successful feature engineering. Poor data quality can severely limit the effectiveness of feature engineering, resulting in suboptimal model performance. High-quality input data is essential for deriving meaningful and discriminative features that capture the underlying patterns and relationships within the data[16]. The below Fig 10 shows a simple graph illustrating the relationship between data quality and model performance.

Fig 10: A simple graph illustrating the relationship between data quality and model performance.

Data collection plays a fundamental role in feature engineering, as it directly impacts the availability and quality of the input data. Careful attention must be given to the data collection process, including selecting appropriate data sources, using the right data formats, and employing efficient data acquisition methods. The decisions made during this phase can significantly influence the success of feature engineering efforts and, ultimately, the performance of machine learning models. [17]

In addition, data preprocessing is a critical step in ensuring the quality of the input data for feature engineering. This involves addressing issues such as missing values, outliers, and noise in the data. Applying proper data cleaning techniques—like imputing missing values, removing outliers, and filtering out noise—can enhance the reliability and consistency of the data. Furthermore, normalization techniques such as standardization or min-max scaling help ensure the data is on a consistent scale, which improves the effectiveness of the feature engineering process. By addressing issues with the input data, these preprocessing steps can lead to more informative and discriminative features, ultimately improving the performance of the machine learning models. [18]

In conclusion, feature engineering is a vital component of machine learning, focusing on identifying, selecting, and transforming the most relevant features from raw data. When done effectively, feature engineering leads to more accurate, robust, and interpretable models, making it a key step in building successful AI systems.

5. The Evolving Paradigm of "Software 2.0"

As feature engineering has become increasingly important in machine learning, a new software engineering paradigm has emerged, known as "Software 2.0." In this paradigm, feature engineering is recognized as a central discipline within software engineering. Here, machine learning models are the primary output, and the process of feature engineering takes center stage in software development[19].

This shift in perspective has spurred the development of numerous open-source and proprietary tools and frameworks designed to streamline and automate the feature engineering process. These tools offer a variety of capabilities, including automated feature generation, feature selection, and feature transformation, to help data scientists and machine learning engineers more efficiently create informative features from raw data[5].

Moreover, the growing awareness of the critical role data quality plays in machine learning has led to the rise of MLOps practices. MLOps focuses on the management, monitoring, and continuous improvement of data and model quality throughout the entire machine learning lifecycle[20]. These practices ensure that high-quality data and features are consistently available, supporting the development and deployment of robust and reliable machine learning models. The below Fig 11 highlights the key aspects of Software 2.0 in a structured way.

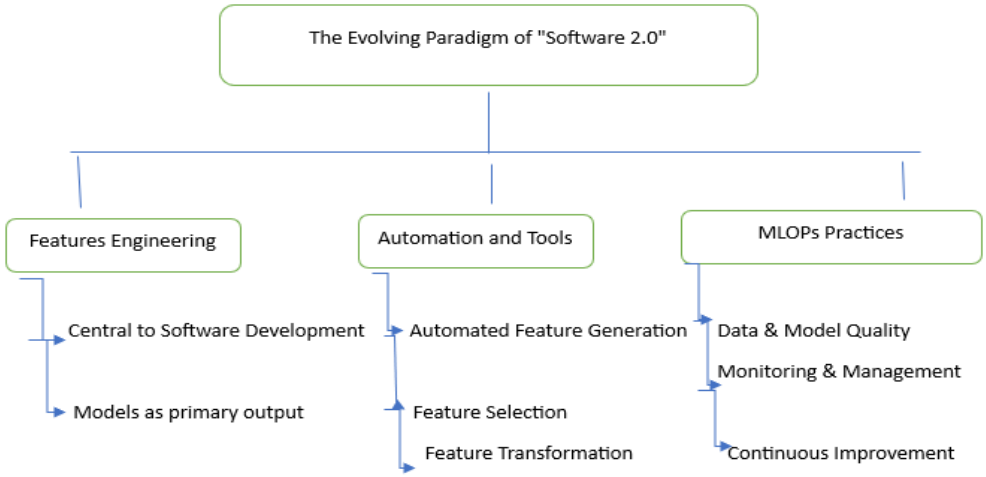

Fig 11: A diagram illustrating the evolving paradigm of "Software 2.0"

6. The Challenges of Feature Engineering

While feature engineering is a powerful method for improving machine learning performance, it comes with several challenges. One of the main obstacles is its labor-intensive and time-consuming nature, often requiring deep domain expertise and significant creativity. As highlighted in the literature, "feature engineering is important but labor-intensive and underscores the limitation of current learning algorithms: their inability to autonomously extract and organize discriminative information from data"[3]. Another challenge is the potential risk of introducing bias and overfitting. If feature engineering is not done carefully, poorly designed features or those that lack generalization can cause models to perform well on training data but struggle to generalize to new, unseen data[9]. Additionally, the exponential growth of data and the increasing complexity of machine learning tasks have made traditional feature engineering methods less scalable and adaptable. To address these issues, researchers are turning to deep learning techniques for feature engineering, which can automatically learn more informative and representative features from raw data in a more scalable and efficient way[21]. The Fig 12 summarizes the key challenges in a structured way.

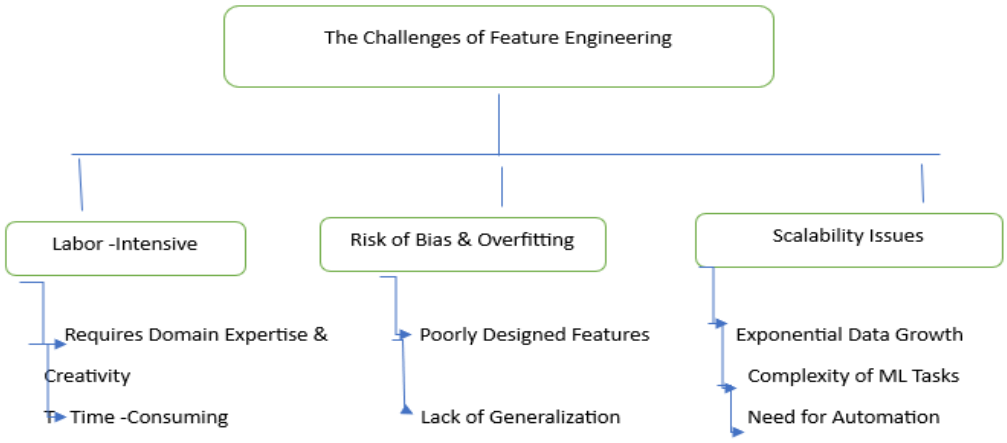

Fig 12: A diagram illustrating the challenges of feature engineering

7. Emerging Trends in Feature Engineering

Emerging trends in feature engineering emphasize the growing role of deep learning techniques in automating feature extraction and selection. Methods like deep representation learning and end-to-end feature extraction are gaining prominence because they can automatically learn more informative and discriminative features directly from raw data. These deep learning-based approaches often outperform traditional manual feature engineering methods in terms of both accuracy and efficiency[15].

Another important trend is the increasing integration of feature engineering into the broader machine learning lifecycle, often referred to as MLOps. This approach seeks to streamline the feature engineering process, ensuring the consistent availability of high-quality features for model training and deployment, while aligning feature engineering practices with the overall goals and requirements of the machine learning system [22].

Additionally, there is a growing emphasis on the interpretability and explainability of features, especially as machine learning models are being deployed in high-stakes fields like healthcare and finance. Techniques such as feature importance analysis, feature visualization, and feature interaction analysis are becoming more critical for understanding how models make decisions and ensuring that they are trustworthy and accountable[23]. The below Fig 13 provides a clear, high-level view of the three major trends in feature engineering.

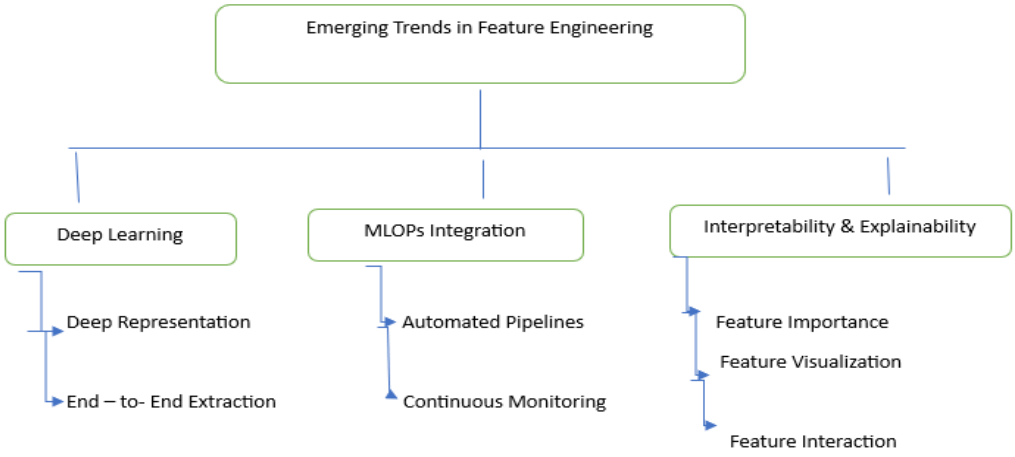

Fig 13: A diagram representing major trends in feature engineering.

8. Conclusion

In conclusion, feature engineering is a critical aspect of machine learning, playing a key role in developing effective and reliable artificial intelligence systems. While traditional feature engineering methods have been effective, the increasing complexity of data and the challenges in machine learning have highlighted the need for more automated and scalable techniques. Deep learning-based approaches to feature engineering are emerging as powerful solutions, as they can automatically learn informative and discriminative features directly from raw data, often surpassing conventional manual methods in performance.

Moreover, there is a growing focus on integrating feature engineering into the broader machine learning lifecycle through MLOps practices. This integration ensures that high-quality features are consistently available for both model training and deployment, aligning feature engineering with the overall goals of the system.

Additionally, the importance of interpretability and explainability, particularly in high-stakes fields such as healthcare and finance, has spurred the development of techniques to analyze feature importance, visualize feature interactions, and understand how machine learning models make decisions. These advancements are crucial for building trustworthy and accountable AI systems, ensuring that they are both effective and reliable in real-world applications.

References

[1] M. Kalinowski et al., "Identifying Challenges in Machine Learning-Enabled Systems Engineering," *arXiv preprint*, May 2024. [Online]. Available: https://doi.org/10.48550/arxiv.2406.04359.

[2] J. Heaton, "A Study on Feature Engineering in Predictive Modeling," *IEEE International Workshop on Software Engineering for Data Science*, Mar. 2016. [Online]. Available: https://doi.org/10.1109/secon.2016.7506650.

[3] Y. Bengio, A. Courville, and P. Vincent, "A Review on Representation Learning and Its Perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798-1828, Aug. 2013. [Online]. Available: https://doi.org/10.1109/tpami.2013.50.

[4] K. Kowsari et al., "Survey of Text Classification Techniques," *Information*, vol. 10, no. 4, p. 150, Apr. 2019. [Online]. Available: https://doi.org/10.3390/info10040150.

[5] J. Li et al., "Feature Selection from a Data Perspective," *ACM Transactions on Knowledge Discovery from Data*, vol. 50, no. 6, Dec. 2017. [Online]. Available: http://arxiv.org/abs/1601.07996.

[6] A. Jović, K. Brkić, and N. Bogunović, "Overview of Feature Selection Approaches and Applications," *Proceedings of the International Convention on ICT, Electronics, and Microelectronics (MIPRO)*, pp. 716-721, May 2015. [Online]. Available: https://doi.org/10.1109/mipro.2015.7160458.

[7] R. Kohavi and G. H. John, "Wrapper Methods for Selecting Feature Subsets," *Artificial Intelligence*, vol. 97, no. 1-2, pp. 273-324, Dec. 1997. [Online]. Available: https://doi.org/10.1016/s0004-3702(97)00043-x.

[8] I. Guyon and A. Elisseeff, "Understanding Variable and Feature Selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157-1182, Mar. 2003. [Online]. Available: https://doi.org/10.5555/944919.944968.

[9] B. Butcher and B. A. Smith, "Practical Methods for Feature Engineering and Selection," *Statistical Modelling*, vol. 74, no. 3, pp. 308-309, Jul. 2020. [Online]. Available: https://doi.org/10.1080/00031305.2020.1790217.

[10] M. Wojtas and K. Chen, "Ranking Feature Importance in Deep Learning Models," *arXiv preprint*, Jan. 2020. [Online]. Available: https://doi.org/10.48550/arxiv.2010.08973.

[11] M. S. Uddin et al., "Developing an Improved Model for Feature Engineering and Selection in ML," *Applied Sciences*, vol. 8, no. 4, p. 646, Apr. 2018. [Online]. Available: https://doi.org/10.3390/app8040646.

[12] X. Cheng et al., "Exploring Polynomial Regression as an Alternative to Neural Networks," *arXiv preprint*, Jan. 2018. [Online]. Available: https://doi.org/10.48550/arxiv.1806.06850.

[13] Z. Zhao and H. Liu, "Identifying Interacting Features in Machine Learning," *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1156-1161, Jan. 2007. [Online]. Available: https://ijcai.org/papers07/Papers/IJCAI07-187.pdf.

[14] L. Qiu, V. M. Chinchilli, and L. Lin, "Understanding Deep Representation Learning for Multi-View Temporal Data," *arXiv preprint*, Jan. 2020. [Online]. Available: https://doi.org/10.48550/arXiv.2005.05485.

[15] Y. LeCun, Y. Bengio, and G. Hinton, "Foundational Concepts in Deep Learning," *Nature*, vol. 521, no. 7553, pp. 436-444, May 2015. [Online]. Available: https://doi.org/10.1038/nature14539.

[16] Y. Roh, G. Heo, and S. E. Whang, "Review on Data Collection Strategies for ML and Big Data," *arXiv preprint*, Jan. 2018. [Online]. Available: https://doi.org/10.48550/arxiv.1811.03402.

[17] J. Qiu et al., "Overview of ML Techniques for Processing Big Data," *Big Data Research*, vol. 3, no. 1, pp. 14-22, May 2016. [Online]. Available: https://doi.org/10.1186/s13634-016-0355-x.

[18] S. García, J. Luengo, and F. Herrera, *Data Preprocessing Techniques in Data Mining*. Springer Nature, Jan. 2015. [Online]. Available: https://doi.org/10.1007/978-3-319-10247-4.

[19] C. Watson et al., "Deep Learning Applications in Software Engineering: A Systematic Review," *arXiv preprint*, Jan. 2020. [Online]. Available: https://doi.org/10.48550/arxiv.2009.06520.

[20] I. Karamitsos, S. Albarhami, and C. Apostolopoulos, "Integration of Continuous Automation in Machine Learning Using DevOps," *Information*, vol. 11, no. 7, p. 363, Jul. 2020. [Online]. Available: https://doi.org/10.3390/info11070363.

[21] L. Hong et al., "Deep Learning-Based Methods for Extracting Text Features: A Review," *Journal of Ambient Intelligence and Humanized Computing*, vol. 2017, no. 1, Dec. 2017. [Online]. Available: https://doi.org/10.1186/s13638-017-0993-1.

[22] D. Kreuzberger et al., "Machine Learning Operations (MLOps): Concepts and Architecture," *arXiv preprint*, Jan. 2022. [Online]. Available: https://doi.org/10.48550/arxiv.2205.02302.

[23] A. Chatzimparmpas et al., "Building Trust in ML Models Through Visualization Techniques," *Computer Graphics Forum*, vol. 39, no. 3, pp. 713-756, Jun. 2020. [Online]. Available: https://doi.org/10.1111/cgf.14034.