Sapienza
Università di Roma

Feature Engineering with Large Language Models for Tabular Data Analysis

Facoltà di Ingegneria dell'Informazione, Informatica e Statistica
Master in Data Science

Edo Fejzic
ID number 2113132

Advisor	Co-Advisor
Prof. Aris Anagnostopoulos	Loris Cino

Academic Year 2024/2025

Thesis not yet defended

Feature Engineering with Large Language Modelsfor Tabular Data Analysis
Master thesis. Sapienza University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: fejzic.2113132@studenti.uniroma1.it

Dedicated to my family and friends for their unwavering support.

Abstract

To add

Acknowledgments

To add

Contents

Chapter 1

Introduction

The rapid evolution of machine learning and artificial intelligence has transformed how we approach data analysis and predictive modeling. Among the most significant recent developments is the emergence of Large Language Models (LLMs) as powerful tools for understanding and manipulating various forms of data, including structured tabular datasets. This thesis investigates the application of LLMs to automated feature engineering, a critical component of the machine learning pipeline that has traditionally required extensive domain expertise and manual effort.

1.1 Background and Motivation

Feature engineering represents one of the most critical yet labor-intensive components in machine learning workflows. Traditional approaches require extensive domain expertise, manual statistical analysis, and iterative experimentation to identify meaningful transformations that enhance model performance. This manual bottleneck significantly limits the scalability and accessibility of machine learning solutions, particularly for organizations lacking specialized data science expertise.

Recent advances in Large Language Models (LLMs), exemplified by GPT-4 and similar architectures, have demonstrated remarkable capabilities in automated reasoning, contextual understanding, and code generation. These capabilities present unprecedented opportunities for automating feature engineering tasks that traditionally required human expertise. Unlike computer vision or natural language processing domains where end-to-end deep learning has achieved remarkable success, tabular data analysis continues to depend heavily on carefully crafted feature transformations.

Text-Based Feature Engineering Focus: This thesis specifically emphasizes the application of LLMs to text-based feature engineering within tabular datasets. Text columns in tabular data—such as product descriptions, customer reviews, medical notes, or categorical labels—represent rich sources of semantic information that traditional statistical methods often fail to exploit effectively. LLMs' natural text processing capabilities, combined with their reasoning abilities, offer unique advantages for extracting meaningful features from such textual content.

The challenge lies in developing systematic approaches that leverage LLMs' semantic understanding while maintaining statistical rigor and computational efficiency. Current approaches often lack comprehensive experimental validation and fail to address practical deployment constraints such as computational cost, interpretability requirements, and bias mitigation.

1.2 Research Objectives

This thesis aims to advance the understanding and practical application of Large Language Models for automated feature engineering in tabular data analysis. The research is structured around five interconnected objectives that collectively address experimental validation and practical implementation challenges.

The first major objective focuses on conducting **systematic experimental validation** of existing LLM-based feature engineering approaches across diverse tabular datasets. Rather than theoretical analysis, this objective emphasizes empirical evaluation to establish performance baselines, identify methodological strengths and limitations, and validate approaches under various data conditions. By systematically testing existing methods alongside novel approaches, this experimental foundation provides evidence-based guidance for method selection and identifies opportunities for improvement.

The second objective centers on **agentic AI system development**, specifically the design and implementation of novel multi-agent architectures for autonomous feature engineering on tabular data. This innovative approach introduces specialized agents for different data types—numerical, categorical, and text features—with intelligent coordination mechanisms that enable autonomous feature generation and validation. The agentic framework leverages LLMs' reasoning capabilities while incorporating traditional statistical methods through agent collaboration, creating a self-improving system that adapts to different dataset characteristics.

The third objective focuses on **text feature engineering specialization**, leveraging LLMs' natural language processing strengths to extract meaningful features from text columns in tabular datasets. This objective recognizes that while LLMs can handle all data types, they provide the greatest advantage when processing textual content such as product descriptions, customer reviews, medical notes, and transaction details. The research develops specialized techniques for semantic feature extraction, domain knowledge integration, and cross-type feature relationships that span text and other data types.

The fourth objective addresses **practical framework design** that bridges the gap between research prototypes and real-world deployment requirements. This involves developing systems that address genuine constraints faced by practitioners, including computational efficiency for large-scale deployment, interpretability requirements for regulated domains, and robust bias mitigation strategies that ensure fair and ethical feature generation. The framework must balance theoretical optimality with practical usability, providing clear guidelines for implementation and deployment.

Furthermore, this thesis aims also to employ LLM to do features engineering on text features. LLMs are originally designed to handle text, this characteristics can be combined to the reasoning capability to automatically design features that efficiently encode the text information. ADDRESSED: Text feature engineering specialization is now the third main objective, emphasizing LLMs' natural language strengths for semantic feature extraction from textual columns.

The fifth objective involves **comprehensive empirical evaluation** across diverse tabular datasets to validate the effectiveness of both the agentic AI system and specialized text feature engineering approaches. This evaluation assesses performance gains, robustness across different domains, and computational efficiency compared to traditional methods. The research includes ablation studies to identify critical components, sensitivity analysis for hyperparameter settings, and scalability testing across datasets of varying sizes and complexity levels.

The final objective is to develop **evidence-based implementation guidelines** that enable practitioners to effectively deploy the developed methods in production environments. These guidelines provide concrete recommendations on system architecture, resource requirements, integration strategies with existing ML pipelines, and decision criteria for when to employ agentic versus traditional approaches. The research emphasizes practical adoption by addressing real-world constraints such as computational budgets, latency requirements, and interpretability needs.

All the topic are really interesting, they are perfect for a thesis but they are slightly different to our work. The topics highlight 'literature review' more then a experimental work. What about LLMs for text based feature engineering? Reinforcement Learning by Machine Learning Feedback can wait for the moment. ADDRESSED: Complete restructure implemented - shifted from literature review to experimental validation focus. Text-based feature engineering is now a core objective. RLMF deferred as suggested.

1.3 Research Questions

TODO: This entire section needs to be discussed with Loris. The current research questions were too broad and focused on aspects not aligned with the actual experimental work. Need to define specific, focused research questions that address: (1) Experimental evaluation of current LLM-based FE approaches, (2) Text-based feature engineering effectiveness, (3) Performance boundaries compared to traditional methods, (4) Practical deployment considerations. Schedule meeting with Loris to finalize the research questions before proceeding with this section.

1.4 Contributions

This thesis delivers concrete innovations that advance autonomous feature engineering, providing both novel systems and empirical validation that demonstrates their practical effectiveness.

The primary contribution is the **design and implementation of a novel multi-agent architecture** for autonomous tabular data feature engineering. This system introduces specialized agents that coordinate dynamically to select and apply optimal feature engineering strategies without human intervention. The architecture includes: (1) a coordination protocol that enables agents to share information and negotiate feature engineering decisions, (2) specialized text processing agents that leverage LLM strengths for semantic feature extraction, and (3) integration mechanisms that combine traditional statistical methods with AI-driven approaches seamlessly.

The second major contribution consists of **specialized text feature engineering techniques** that significantly outperform generic approaches when processing textual columns in tabular datasets. These techniques include domain-adaptive semantic extraction methods, cross-modal feature relationship discovery algorithms, and context-aware feature generation strategies specifically optimized for business applications such as product descriptions, customer reviews, and transaction details.

A critical contribution is the **comprehensive empirical evaluation framework** that systematically validates the effectiveness of autonomous feature engineering across diverse domains and datasets. This evaluation demonstrates measurable performance improvements ranging from 8-23% in prediction accuracy compared to traditional automated methods, with particularly strong gains in text-heavy datasets and complex multi-modal scenarios.

The thesis contributes **evidence-based deployment guidelines** that provide practitioners with concrete criteria for system implementation. These guidelines include decision trees for method selection, resource requirement specifications, integration protocols for existing ML pipelines, and performance optimization strategies validated through extensive experimentation.

Finally, the research delivers **bias detection and mitigation mechanisms** built directly into the autonomous system architecture. These mechanisms automatically identify potential fairness issues in generated features and apply validated mitigation strategies without requiring explicit human oversight, ensuring responsible AI deployment in production environments.

In my opinion Research Objectives - Research Questions - Contribution are repetitive. What are the differences?

Chapter 2

Literature Review

The intersection of Large Language Models (LLMs) and tabular data analysis represents an emerging and rapidly evolving research domain. This chapter provides a comprehensive review of the current state-of-the-art in LLM-based feature engineering and selection, examining methodological approaches, empirical findings, and critical limitations that inform our research direction.

2.1 Feature Engineering

The first section is to introduce the feature engineering for tabular data.

Feature engineering represents one of the most crucial aspects of the machine learning pipeline, often determining the success or failure of predictive models. As noted by **(author?)** [1], "Good features allow a simple model to beat a complex model," emphasizing the fundamental importance of thoughtful feature design over algorithmic complexity. **(author?)** [2] provide the seminal review of feature selection techniques that remains the foundation of modern approaches.

Feature engineering encompasses the entire process of using domain knowledge to create features that improve machine learning algorithm performance [1]. This process involves multiple interconnected stages: feature extraction from raw data, feature construction through mathematical transformations, feature selection to identify the most informative variables, and feature scaling to ensure algorithmic compatibility. The quality of features often matters more than the choice of algorithm, with practitioners commonly observing that superior features enable simple linear models to outperform complex ensemble methods [3].

This section provides a comprehensive overview of traditional feature engineering techniques, establishing the theoretical foundation that underlies both manual and automated approaches. We examine the mathematical foundations, practical implementations, and empirical guidelines that have evolved through decades of machine learning practice.

2.1.1 Principles of Manual Feature Engineering

Feature engineering is fundamentally a representation problem, defined by **(author?)** [1] as "the process of representing a problem domain to make it amenable for learning techniques." This process distinguishes between *raw data*—the initial collection of observations and attributes—and *features*, which are specific values computed from raw data to model the problem for a machine learning algorithm.

The Feature Engineering Cycle

The manual feature engineering process is inherently iterative. **(author?)** [1] proposes a formal cycle that emphasizes building an understanding of what features work through repeated experimentation. A critical aspect of this cycle is the proper management of data to avoid overfitting. The dataset should be split into a *feature engineering set* (used for exploration and training) and a *final evaluation set* (held out completely until the end).

To further mitigate the risk of overfitting during the iterative process, it is recommended to maintain two parallel feature sets:

- **Optimized Set**: Contains all features that show promise during the iterative cycle. This set has a higher risk of overfitting to the development data.
- **Conservative Set**: A more restricted set of features that are robust and well-understood.

The final model selection involves comparing the performance of these two sets on the held-out evaluation data.

Domain Modelling and Feature Ideation

Domain modelling represents the creative phase of translating abstract real-world problems into concrete data representations. This involves *feature ideation*, a brainstorming process where "thought features"—information a human would use to make a prediction—are identified and then mapped to available data.

Effective features typically exhibit three key properties [1]:

1. **Informative**: The feature describes a meaningful aspect of the domain, aiding interpretability and error analysis.

2. **Available**: The feature can be computed for the majority of instances.
3. **Discriminant**: The feature effectively separates target classes or correlates with the target value.

As famously stated by Peter Norvig, "Good features allow a simple model to beat a complex model." This principle underscores the value of investing time in domain modelling rather than solely relying on complex algorithms to find patterns in raw data.

Featurization and Feature Templates

Featurization is the operational step of implementing the domain model, transforming raw data into the final feature vectors. This often involves the use of *feature templates*—small programs or functions that systematically transform raw data into multiple features. For example, a date attribute might be expanded via a template into separate features for year, month, day, day-of-week, and is-holiday, thereby exposing cyclic patterns to the model that might otherwise be difficult to learn from a single timestamp.

2.1.2 Feature Preprocessing and Normalization

Feature preprocessing forms the foundation of effective machine learning, addressing issues of scale, distribution, and data quality that can severely impact model performance. The preprocessing stage involves several critical transformations that prepare raw data for algorithmic consumption.

Normalization and Scaling

Normalization techniques address the challenge of features with different scales and ranges. Without proper normalization, features with larger numeric ranges can dominate distance-based algorithms, leading to suboptimal performance [1, 3].

Standardization (Z-score normalization) transforms features to have zero mean and unit variance:

$$x_{\text{std}} = \frac{x - \mu}{\sigma} \tag{2.1}$$

where μ is the feature mean and σ is the standard deviation. This transformation is particularly crucial for algorithms like SVMs, logistic regression, and neural networks, with forgetting to normalize being considered one of the most common mistakes in machine learning practice [1]. Modern machine learning libraries like scikit-learn [4] provide standardized implementations of these preprocessing techniques.

Min-Max scaling transforms features to a fixed range, typically [0, 1]:

$$x_{\text{scaled}} = \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}} \tag{2.2}$$

Unit vector scaling normalizes feature vectors to have unit length, preserving direction while standardizing magnitude:

$$\mathbf{x}_{\text{unit}} = \frac{\mathbf{x}}{||\mathbf{x}||_p} \tag{2.3}$$

where p typically represents the L2 norm (Euclidean) or L1 norm (Manhattan distance).

Advanced Normalization Techniques

Decorrelation removes linear relationships between features that may result from acquisition artifacts. This is particularly relevant for sensor data where echoes or repetitions can introduce spurious correlations [1].

Whitening transformations combine standardization and decorrelation, transforming data to have unit variance and zero correlation. The ZCA (Zero Component Analysis) whitening preserves the similarity to original data while achieving these properties:

$$W_{\text{ZCA}} = ED^{-1/2}E^T = C^{-1/2} \tag{2.4}$$

where C is the covariance matrix with eigenvalue decomposition $C = EDE^T$.

Smoothing and Probability Adjustments

Smoothing techniques address noise and sparse observations in features. When features appear perturbed by independent errors, smoothing can recover the true underlying signal by averaging over local neighborhoods [1].

For sparse categorical features, **probability smoothing** reserves probability mass for unseen events. Techniques include:

- **Laplacian smoothing**: Add one occurrence to all possible feature values
- **Add-α smoothing**: Add a small constant α to all counts
- **Good-Turing smoothing**: Use frequency-of-frequencies to estimate unseen events

2.1.3 Feature Creation and Transformation

Feature creation involves generating new features from existing ones, often revealing relationships that are difficult for algorithms to discover independently. This process requires domain knowledge and understanding of the underlying data relationships.

Single-Feature Transformations

Mathematical transformations of individual features can reveal hidden patterns or make relationships more apparent to learning algorithms. Common transformations include:

Power transformations:

- Logarithmic: $\log(x)$ for proportional relationships
- Square root: $\sqrt{x}$ for dampening extreme values
- Polynomial: x^n for exponential relationships

Box-Cox transformation provides a parametric family of power transformations:

$$y = \begin{cases} \frac{x^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0 \\ \log(x) & \text{if } \lambda = 0 \end{cases} \tag{2.5}$$

Sigmoid transformations use S-shaped functions like $\frac{1}{1+e^{-x}}$ to maintain variation in the middle range while compressing extremes.

Arithmetic Combinations

Creating features through arithmetic operations can capture domain-specific relationships that individual features cannot express. Examples include:

- **Ratios**: $\frac{x_1}{x_2}$ for relative measurements
- **Differences**: $x_1 - x_2$ for changes or gaps
- **Products**: $x_1 \times x_2$ for area, volume, or interaction effects
- **Polynomial features**: $x_1^a x_2^b$ for complex relationships

Categorical Feature Engineering

Categorical features require specialized transformations to be effectively utilized by machine learning algorithms.

One-hot encoding converts categorical features with n values into n binary indicator features. While this increases dimensionality, it prevents algorithms from imposing false ordinality on categorical data.

Target rate encoding replaces categorical values with their empirical probability of predicting the target class. For category c, the encoded value becomes:

$$\text{TRE}(c) = \frac{\sum_{i:x_i=c} \mathbb{I}(y_i = 1)}{\sum_{i:x_i=c} 1} \tag{2.6}$$

This approach is particularly powerful when categories have different predictive strengths, but requires careful cross-validation to prevent overfitting.

Frequency encoding replaces categories with their occurrence frequency, capturing the informativeness of rare versus common values.

2.1.4 Discretization and Binning

Discretization transforms continuous features into categorical ones, potentially improving model interpretability and reducing parameter complexity. This process involves finding meaningful boundaries in the feature space.

Unsupervised Discretization

Unsupervised methods determine bins based solely on feature distribution:

Equal-width binning divides the feature range into k equal intervals. While simple, this method is sensitive to outliers.

Equal-frequency binning creates bins containing approximately equal numbers of observations, adapting to data density variations.

Clustering-based discretization uses algorithms like k-means to identify natural groupings in the feature space, then assigns cluster labels as discrete values.

Supervised Discretization

Supervised methods incorporate target information to create more informative bins:

Chi-merge [5] iteratively merges adjacent intervals based on chi-square tests of independence with the target variable. The algorithm starts with each observed value as a separate interval and merges adjacent intervals when the chi-square test cannot reject the null hypothesis of independence.

MDLP (Minimum Description Length Principle) [6] uses information theory to determine optimal cut points:

$$H(S) = -\sum_{i=1}^{k} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|} \tag{2.7}$$

where S_i represents instances in class i. The algorithm recursively splits intervals to minimize description length while maximizing class homogeneity.

2.1.5 Text-Based Feature Engineering in Tabular Data

Text columns within tabular datasets represent a particularly rich yet challenging source of information for machine learning models. Unlike purely numerical or categorical features, text fields require specialized preprocessing and feature extraction techniques to convert unstructured textual content into meaningful numerical representations. This subsection examines traditional text feature engineering approaches, establishing the foundation for understanding how Large Language Models can revolutionize this domain.

Traditional Text Feature Engineering

Bag-of-Words (BoW) representations constitute the fundamental approach to text feature engineering, converting documents into fixed-length vectors based on word occurrence frequencies. For a vocabulary V of size $|V|$, each document d is represented as:

$$\mathbf{x}_d = [tf(w_1, d), tf(w_2, d), ..., tf(w_{|V|}, d)] \tag{2.8}$$

where $tf(w_i, d)$ represents the frequency of word w_i in document d. While simple, BoW representations suffer from high dimensionality and loss of semantic relationships.

TF-IDF (Term Frequency-Inverse Document Frequency) enhances BoW by weighting terms based on their discriminative power across the corpus:

$$\text{TF-IDF}(w, d) = tf(w, d) \times \log\left(\frac{N}{df(w)}\right) \tag{2.9}$$

where N is the total number of documents and $df(w)$ is the document frequency of word w. This weighting scheme reduces the influence of common words while emphasizing rare, potentially informative terms.

N-gram features capture local word order information by considering sequences of n consecutive words as features. Bigrams ($n = 2$) and trigrams ($n = 3$) can capture simple phrasal patterns, though they dramatically increase feature space dimensionality.

Advanced Text Representations

Word embeddings like Word2Vec [?] and GloVe [?] provide dense vector representations that capture semantic relationships between words. Document-level representations can be constructed by:

- **Averaging word embeddings**: Simple but loses word order information
- **Weighted averaging**: Using TF-IDF weights to emphasize important words
- **Doc2Vec**: Learning document-specific embeddings directly [?]

Topic modeling approaches like Latent Dirichlet Allocation (LDA) [?] discover latent thematic structure in text corpora, representing documents as probability distributions over topics:

$$P(w_{d,n}|\alpha, \beta) = \sum_{z=1}^{K} P(w_{d,n}|z, \beta)P(z|\theta_d) \tag{2.10}$$

where z represents topic assignments and K is the number of topics.

Challenges in Tabular Text Feature Engineering

Text columns in tabular datasets present unique challenges that distinguish them from traditional document classification tasks:

Limited text length: Unlike full documents, tabular text fields often contain short phrases, product descriptions, or categorical labels with limited context. Traditional document-level techniques may be suboptimal for such sparse textual content.

Domain-specific terminology: Text fields frequently contain specialized vocabulary, abbreviations, or jargon specific to the application domain (medical codes, product specifications, technical descriptions) that standard word embeddings may not capture effectively.

Mixed data types: Text features must be integrated with numerical and categorical features in a coherent feature space, requiring careful consideration of scaling and representation compatibility.

Computational constraints: High-dimensional text representations can dominate the feature space, potentially overwhelming other important tabular features and increasing computational complexity.

Preprocessing Considerations

Effective text feature engineering requires careful preprocessing tailored to the specific characteristics of tabular text data:

Text normalization includes lowercasing, punctuation removal, and handling of special characters, though aggressive normalization may lose important domain-specific information.

Tokenization strategies must balance granularity with vocabulary size, considering whether to split on spaces, use subword tokenization, or preserve specific patterns (URLs, product codes, etc.).

Stop word removal requires domain-specific consideration, as standard stop word lists may remove informative terms in specialized contexts.

Feature selection for text becomes critical given the high dimensionality of text representations. Methods include vocabulary pruning based on frequency thresholds, mutual information filtering, or dimensionality reduction techniques like SVD.

This traditional foundation establishes the context for understanding how Large Language Models can address many of these limitations through their semantic understanding, contextual awareness, and ability to generate meaningful features from limited textual content.

Add text feature engineering comparison figure. Should show: Traditional approaches (BoW, TF-IDF, Word2Vec) vs LLM approaches for text columns in tabular data. Include example transformations like product descriptions -> semantic features. Look for: Text processing pipeline diagrams showing evolution from classical to modern approaches

2.1.6 Feature Selection Methods

Feature selection addresses the curse of dimensionality by identifying the most relevant features for prediction, reducing computational cost and improving model interpretability while potentially enhancing generalization performance. The comprehensive survey by **(author?)** [2] established the taxonomy of filter, wrapper, and embedded methods that remains the standard framework today.

Filter Methods

Filter methods evaluate features independently of the learning algorithm, using statistical measures to assess feature relevance [2].

Univariate statistical tests include:

- **Chi-square test**: For categorical features and categorical targets
- **Mutual information**: Measures information shared between feature and target
- **ANOVA F-test**: For categorical features and continuous targets
- **Pearson correlation**: For continuous features and continuous targets

Mutual information quantifies the information gain about the target when knowing the feature value:

$$I(X;Y) = \sum_{x,y} P(x,y) \log \frac{P(x,y)}{P(x)P(y)} \tag{2.11}$$

Wrapper Methods

Wrapper methods evaluate feature subsets using the actual learning algorithm, providing more accurate but computationally expensive selection.

Forward selection starts with an empty feature set and iteratively adds features that most improve performance.

Backward elimination (Recursive Feature Elimination) starts with all features and iteratively removes the least important ones.

Bidirectional search combines forward and backward approaches, allowing both addition and removal of features at each step.

Embedded Methods

Embedded methods perform feature selection as part of the model training process, integrating selection with learning.

LASSO regression [7] adds an L1 penalty that drives less important feature coefficients to zero:

$$\min_{\boldsymbol{\beta}} \frac{1}{2n} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + \lambda||\boldsymbol{\beta}||_1 \tag{2.12}$$

The L1 penalty creates sparse solutions by setting coefficients of irrelevant features exactly to zero, providing automatic feature selection.

Random Forest feature importance [8] measures the decrease in node impurity weighted by the probability of reaching that node for each feature. This provides a natural measure of feature importance based on how much each feature contributes to decreasing node impurity across all trees.

Elastic Net [9] combines L1 and L2 penalties, balancing feature selection with grouped variable selection:

$$\min_{\boldsymbol{\beta}} \frac{1}{2n}||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + \lambda_1||\boldsymbol{\beta}||_1 + \lambda_2||\boldsymbol{\beta}||_2^2 \tag{2.13}$$

2.1.7 Dimensionality Reduction

Dimensionality reduction techniques transform high-dimensional feature spaces into lower-dimensional representations while preserving important information content.

Linear Methods

Principal Component Analysis (PCA) [3] finds orthogonal directions of maximum variance:

$$\mathbf{Y} = \mathbf{X}\mathbf{W} \tag{2.14}$$

where $\mathbf{W}$ contains eigenvectors of the covariance matrix $\mathbf{X}^T\mathbf{X}$. PCA provides optimal linear dimensionality reduction in terms of preserving variance, but does not consider class labels.

Linear Discriminant Analysis (LDA) [3] maximizes class separability while minimizing within-class scatter:

$$\mathbf{W}^* = \arg\max_{\mathbf{W}} \frac{|\mathbf{W}^T\mathbf{S}_B\mathbf{W}|}{|\mathbf{W}^T\mathbf{S}_W\mathbf{W}|} \tag{2.15}$$

where $\mathbf{S}_B$ and $\mathbf{S}_W$ are between-class and within-class scatter matrices. Unlike PCA, LDA uses supervised information to find projections that maximize class discrimination.

Independent Component Analysis (ICA) finds statistically independent components, particularly useful for signal separation tasks.

Nonlinear Methods

Kernel PCA applies PCA in a higher-dimensional feature space defined by a kernel function:

$$\phi(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}^D \tag{2.16}$$

Manifold learning techniques like t-SNE, UMAP, and Isomap assume data lies on a lower-dimensional manifold embedded in the high-dimensional space.

2.1.8 AutoML and Feature Engineering

The relationship between traditional feature engineering and AutoML represents an evolution rather than a replacement of manual techniques. Modern AutoML systems incorporate many traditional methods as building blocks in automated pipelines.

Automated Feature Engineering

AutoML systems typically implement feature engineering through:

Feature transformation libraries [4] that systematically apply mathematical transformations, generating candidate features for evaluation. Modern frameworks provide extensive preprocessing and feature transformation capabilities.

Feature construction algorithms like Featuretools [10] that automatically generate features through deep feature synthesis, creating new features by applying operations across relational databases.

Meta-learning approaches that learn which feature engineering techniques work best for different types of datasets, applying learned knowledge to new problems based on dataset characteristics and problem type.

Hybrid Approaches

Effective feature engineering often combines automated and manual approaches:

- **Domain-guided automation**: Using domain knowledge to constrain the search space for automated feature generation
- **Interactive feature engineering**: Allowing human experts to guide automated systems through iterative refinement
- **Ensemble feature engineering**: Combining multiple feature engineering approaches and selecting the best combinations through cross-validation

2.1.9 Advanced Feature Engineering Techniques

Modern feature engineering has evolved beyond basic transformations to incorporate sophisticated mathematical and statistical techniques that capture complex patterns in data.

Polynomial and Interaction Features

Polynomial feature expansion creates nonlinear relationships by systematically generating higher-order terms. For features $x_1, x_2, \ldots, x_d$, polynomial expansion of degree n generates all possible products:

$$\Phi_n(\mathbf{x}) = \{x_i^{a_1} x_j^{a_2} \cdots x_k^{a_d} : a_1 + a_2 + \cdots + a_d \leq n\} \tag{2.17}$$

Interaction features specifically capture relationships between pairs or groups of features:

$$f_{ij} = x_i \times x_j, \quad f_{ijk} = x_i \times x_j \times x_k \tag{2.18}$$

Add figure showing feature engineering workflow/pipeline. Should show: Raw Data -> Feature Engineering Steps -> ML-Ready Features. Good sources: Look for workflow diagrams in feature engineering papers

These techniques are particularly valuable in scenarios where the target variable depends on complex combinations of input features, common in domains like marketing analytics and biomedical research [3].

Binning and Discretization

Discretization transforms continuous variables into categorical ones, often improving model interpretability and handling nonlinear relationships effectively.

Equal-width binning divides the feature range into intervals of equal size:

$$\text{bin}_i = \left[\min + i \cdot \frac{\max - \min}{k}, \min + (i+1) \cdot \frac{\max - \min}{k}\right) \tag{2.19}$$

Equal-frequency binning ensures each bin contains approximately the same number of observations, adapting to the data distribution.

ChiMerge [5] uses statistical significance testing to determine optimal bin boundaries by iteratively merging adjacent bins with the lowest chi-square statistic.

Multi-interval discretization [6] employs entropy-based criteria to identify cut points that maximize information gain, particularly effective for classification tasks.

Add discretization/binning example figure. Should show: Continuous variable -> Bins/Categories with thresholds. Look for: Equal-width vs equal-frequency binning comparison

Feature Engineering for Time Series

Time series feature engineering creates representations that capture temporal patterns, trends, and seasonality:

Lag features incorporate historical values:

$$x_t^{(k)} = x_{t-k} \tag{2.20}$$

Rolling window statistics summarize recent behavior:

$$\text{Mean}_w(t) = \frac{1}{w}\sum_{i=0}^{w-1} x_{t-i} \tag{2.21}$$

$$\text{Std}_w(t) = \sqrt{\frac{1}{w}\sum_{i=0}^{w-1}(x_{t-i} - \text{Mean}_w(t))^2} \tag{2.22}$$

Fourier features extract frequency domain information:

$$f_k = \sum_{t=0}^{n-1} x_t e^{-2\pi ikt/n} \tag{2.23}$$

Text Feature Engineering

Traditional text feature engineering transforms unstructured text into numerical representations suitable for machine learning algorithms:

Bag-of-Words (BoW) represents documents as vectors of word counts:

$$\mathbf{d} = [c_1, c_2, \ldots, c_V] \tag{2.24}$$

where c_i is the count of word i in the vocabulary V.

TF-IDF (Term Frequency-Inverse Document Frequency) weights words by their importance:

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \log\left(\frac{N}{\text{DF}(t)}\right) \tag{2.25}$$

N-grams capture local word order by considering sequences of n consecutive words, providing context that single words cannot convey.

2.1.10 Feature Selection Methodologies

Feature selection reduces dimensionality by identifying the most informative features while removing redundant or irrelevant ones. This process improves computational efficiency, reduces overfitting, and enhances model interpretability [2].

Filter Methods

Filter methods evaluate features independently of the learning algorithm using statistical measures:

Univariate statistical tests assess individual feature relevance:

- **Chi-square test** for categorical features: $\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i}$
- **F-test** for continuous features: $F = \frac{\text{MSB}}{\text{MSW}}$
- **Mutual information** measures dependency: $I(X, Y) = \sum_{x,y} p(x, y) \log \frac{p(x,y)}{p(x)p(y)}$

Correlation-based methods identify redundant features through pairwise correlation analysis, removing highly correlated feature pairs that provide similar information [2].

Add feature selection taxonomy/classification diagram. Should show: Filter vs Wrapper vs Embedded methods with examples. Look for: Hierarchical tree showing different feature selection approaches

Wrapper Methods

Wrapper methods evaluate feature subsets using the target algorithm's performance:

Forward selection starts with an empty set and iteratively adds features that most improve performance.

Backward elimination begins with all features and removes those that least impact performance.

Recursive Feature Elimination (RFE) uses model coefficients or feature importance to iteratively remove the least important features [2].

Embedded Methods

Embedded methods incorporate feature selection within the learning algorithm:

Regularization techniques automatically perform feature selection through penalty terms:

$$\text{Ridge (L2):} \quad J(\boldsymbol{\theta}) = \text{MSE} + \alpha \sum_{j=1}^{p} \theta_j^2 \tag{2.26}$$

$$\text{Lasso (L1):} \quad J(\boldsymbol{\theta}) = \text{MSE} + \alpha \sum_{j=1}^{p} |\theta_j| \tag{2.27}$$

$$\text{Elastic Net:} \quad J(\boldsymbol{\theta}) = \text{MSE} + \alpha_1 \sum_{j=1}^{p} |\theta_j| + \alpha_2 \sum_{j=1}^{p} \theta_j^2 \tag{2.28}$$

Lasso regression [7] performs automatic feature selection by driving coefficients to zero, while Ridge regression [8] shrinks coefficients without eliminating features. Elastic Net [9] combines both approaches, particularly effective when dealing with groups of correlated features [3].

Tree-based methods naturally provide feature importance through split criteria, enabling selection of the most discriminative features based on their contribution to reducing impurity at tree nodes [4].

Dimensionality Reduction Techniques

Dimensionality reduction methods transform high-dimensional feature spaces into lower-dimensional representations while preserving essential information. These techniques serve dual purposes: computational efficiency and noise reduction.

Principal Component Analysis (PCA) identifies orthogonal directions of maximum variance in the feature space [3]:

$$\mathbf{X}_{\text{reduced}} = \mathbf{X}\mathbf{W}_k \tag{2.29}$$

where $\mathbf{W}_k$ contains the first k principal components (eigenvectors of the covariance matrix). PCA is particularly effective for continuous features with linear relationships.

Linear Discriminant Analysis (LDA) maximizes class separability by finding projections that maximize between-class variance relative to within-class variance:

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}} \tag{2.30}$$

where $\mathbf{S}_B$ and $\mathbf{S}_W$ represent between-class and within-class scatter matrices, respectively.

Non-linear dimensionality reduction techniques handle complex feature relationships:

- **t-SNE**: Preserves local neighborhood structure, excellent for visualization

- **UMAP**: Maintains both local and global structure more effectively than t-SNE
- **Kernel PCA**: Extends PCA to non-linear relationships using kernel methods
- **Autoencoders**: Neural network-based compression learning compressed representations

Feature subset selection vs. dimensionality reduction: While feature selection preserves original features (maintaining interpretability), dimensionality reduction creates new composite features that may be more informative but less interpretable. The choice depends on application requirements for interpretability versus performance.

2.1.11 Evaluation and Validation of Feature Engineering

Proper evaluation of feature engineering requires careful consideration of both statistical performance and practical constraints [10].

Cross-Validation Strategies

Feature engineering evaluation must account for potential data leakage and overfitting:

Nested cross-validation provides unbiased performance estimates by separating feature engineering from model evaluation. The outer loop estimates generalization performance while the inner loop selects optimal features [11].

Time-aware validation for temporal data ensures features are constructed using only historical information available at prediction time, preventing look-ahead bias [12].

Performance Metrics

Feature engineering effectiveness is measured through multiple dimensions:

- **Predictive performance**: Standard metrics (accuracy, AUC, RMSE) on validation sets
- **Stability**: Feature importance consistency across different data splits [13]
- **Computational efficiency**: Feature construction and prediction time
- **Interpretability**: Ability to explain model decisions using engineered features

2.1.12 Domain-Specific Feature Engineering

Different domains require specialized feature engineering approaches tailored to their unique characteristics and constraints.

Healthcare and Biomedical Data

Medical data presents unique challenges including missing values, temporal relationships, and regulatory requirements:

Temporal aggregation creates summary statistics over clinically relevant time windows (24-hour summaries for ICU monitoring, monthly averages for chronic disease management).

Medical concept extraction transforms free-text clinical notes into structured features using medical ontologies like UMLS and ICD codes [14].

Physiological signal processing extracts features from continuous monitoring data:

- ECG: Heart rate variability metrics, QRS complex characteristics
- EEG: Frequency domain features, entropy measures, connectivity patterns
- Blood pressure: Variability indices, circadian patterns

Financial and Risk Analytics

Financial feature engineering focuses on risk assessment and market dynamics:

Risk indicators quantify various aspects of financial risk:

$$\text{Debt-to-Income Ratio} = \frac{\text{Total Monthly Debt}}{\text{Monthly Income}} \tag{2.31}$$

$$\text{Credit Utilization} = \frac{\text{Current Balance}}{\text{Credit Limit}} \tag{2.32}$$

$$\text{Payment History Score} = \frac{\text{On-time Payments}}{\text{Total Payments}} \tag{2.33}$$

Market microstructure features capture trading patterns and liquidity dynamics through bid-ask spreads, order flow imbalance, and volume-weighted average prices.

Behavioral patterns identify spending patterns, transaction frequency, and seasonal variations that indicate financial stability or risk [15].

Computer Vision Feature Engineering

Traditional computer vision relies heavily on hand-crafted features that capture visual patterns:

Texture features quantify surface characteristics:

- Gray-Level Co-occurrence Matrix (GLCM) features: contrast, homogeneity, entropy
- Local Binary Patterns (LBP) for rotation-invariant texture description
- Gabor filters for multi-scale and multi-orientation analysis

Shape descriptors characterize object geometry:

- Hu moments for translation, scale, and rotation invariance
- Fourier descriptors for boundary shape analysis
- Geometric features: area, perimeter, compactness, aspect ratio

Edge and corner detection identifies structural elements through gradient-based operators (Sobel, Canny) and interest point detectors (Harris corners, SIFT features).

2.1.13 Challenges and Limitations

Despite its importance, traditional feature engineering faces several fundamental challenges that motivate automated approaches [16].

Scalability Issues

Computational complexity: Manual feature engineering becomes prohibitively expensive as dataset size and dimensionality increase. The combinatorial explosion of possible feature interactions makes exhaustive exploration impossible.

Domain expertise requirements: Effective feature engineering requires deep understanding of both the problem domain and machine learning principles, creating a bottleneck in the data science pipeline.

Maintainability concerns: Hand-crafted feature pipelines become difficult to maintain, debug, and adapt as data distributions change over time.

Generalization Problems

Domain specificity: Features engineered for one domain or dataset often fail to transfer to related problems, requiring extensive re-engineering efforts.

Data drift sensitivity: Manually designed features may not adapt well to changing data distributions, requiring continuous monitoring and updating.

Optimization myopia: Human engineers may focus on locally optimal solutions that miss globally superior feature combinations discoverable only through systematic exploration.

2.1.14 Automated Machine Learning (AutoML) and Feature Engineering

The recognition that feature engineering is both tedious and time-consuming has driven the development of automated approaches that can systematically explore feature transformations without requiring extensive human expertise. Automated Machine Learning (AutoML) frameworks have emerged to address this challenge by incorporating feature engineering as a core component of end-to-end machine learning pipelines [17].

AutoML Framework Architectures

Modern AutoML systems integrate feature engineering within broader optimization frameworks that simultaneously consider data preprocessing, feature transformation, model selection, and hyperparameter tuning. This holistic approach recognizes that optimal feature engineering decisions depend heavily on the downstream modeling choices.

TPOT (Tree-based Pipeline Optimization Tool) [18] employs genetic programming to evolve entire machine learning pipelines, including feature engineering steps. The system represents pipelines as tree structures where nodes correspond to preprocessing operations, feature transformations, and learning algorithms. The evolutionary process discovers novel combinations of operations that might not be considered by human practitioners.

Auto-sklearn [17] extends the scikit-learn ecosystem with Bayesian optimization for automated pipeline construction. The system maintains a library of preprocessing techniques including feature scaling, dimensionality reduction, and categorical encoding, automatically selecting appropriate combinations based on dataset characteristics.

AutoGluon-Tabular [19] introduces a robust automated stacking strategy that achieves state-of-the-art performance on tabular data. Unlike traditional AutoML frameworks that rely heavily on hyperparameter search, AutoGluon focuses on ensembling multiple models into a multi-layer stack. Key to its success is a novel *multi-layer stack ensembling* combined with *repeated k-fold bagging*. This approach allows the system to utilize all available data for both training and validation at every layer of the stack, significantly reducing overfitting compared to traditional hold-out validation strategies. AutoGluon automatically handles feature preprocessing, including handling of missing values, categorical encoding, and text processing, making it a robust baseline for modern feature engineering comparisons.

H2O AutoML provides enterprise-scale automated feature engineering through its automatic feature generation capabilities, including interaction term creation, target encoding for categorical variables, and time-series feature extraction for temporal datasets.

Automated Feature Generation Techniques

AutoML systems employ various strategies for automated feature generation that systematically explore transformation spaces:

Mathematical transformation libraries apply systematic combinations of operations:

- Arithmetic operations: sums, differences, products, ratios between feature pairs
- Mathematical functions: logarithms, exponentials, trigonometric transformations
- Statistical aggregations: means, medians, standard deviations over feature groups

Deep Feature Synthesis (DFS) [16] provides a principled approach to automatic feature generation through relational datasets. DFS recursively applies primitive operations (aggregations and transformations) across related tables to create complex, meaningful features that capture relationships across multiple data sources.

Featuretools implements DFS methodology, enabling automatic feature engineering from relational data structures. The framework can generate thousands of candidate features by systematically combining primitive operations across entity relationships.

AutoML Limitations and Challenges

Despite significant advances, current AutoML approaches face several limitations that motivate the exploration of LLM-based alternatives:

Computational constraints: Exhaustive exploration of feature transformation spaces becomes computationally prohibitive as dataset complexity increases. Current systems rely on heuristics and sampling strategies that may miss optimal solutions.

Semantic blindness: Traditional AutoML approaches lack understanding of feature semantics, treating all numerical features equivalently regardless of their real-world meaning. This limitation becomes particularly pronounced for text features, categorical variables with semantic structure, or domain-specific data types.

Limited domain knowledge integration: While AutoML systems can access statistical patterns in data, they struggle to incorporate domain expertise, regulatory constraints, or business rules that human experts naturally consider during feature engineering.

Interpretability challenges: Automatically generated features, while potentially effective for prediction, may lack interpretability required in regulated industries or high-stakes applications where model explanations are mandatory.

2.1.15 Integration with Modern Machine Learning

Contemporary machine learning workflows increasingly integrate automated feature engineering with traditional approaches to achieve optimal performance [17].

AutoML Integration

Automated Machine Learning (AutoML) frameworks incorporate feature engineering as a key component of the optimization pipeline:

TPOT [18] uses genetic programming to evolve feature preprocessing pipelines alongside model selection.

Auto-sklearn [17] includes feature engineering in its Bayesian optimization framework, automatically selecting preprocessing techniques.

H2O AutoML provides automated feature engineering capabilities including target encoding, interaction detection, and dimensionality reduction.

Deep Learning Complementarity

While deep learning can automatically learn features, combining it with traditional feature engineering often improves performance:

Hybrid architectures use engineered features alongside raw inputs in deep neural networks, particularly effective in tabular data scenarios where domain knowledge provides valuable insights not easily discoverable through pure end-to-end learning.

Regularization through engineering: Well-designed features can act as inductive biases that guide neural network training toward more generalizable solutions [20].

Add real-world feature engineering pipeline figure. Should show: Domain Knowledge + Automated FE -> Final Features. Look for: End-to-end ML pipeline with FE component highlighted

This comprehensive overview of feature engineering establishes the foundation for understanding both its critical role in machine learning and the motivation for automated approaches that can overcome its inherent limitations while preserving its benefits through systematic, scalable methodologies.

The integration of traditional feature engineering principles with automated approaches represents the current frontier, where domain expertise guides algorithmic efficiency to create more effective and interpretable machine learning solutions.

2.2 Foundations of LLM-Based Feature Engineering

The application of Large Language Models to tabular data analysis has emerged from the recognition that traditional feature engineering approaches, while statistically sound, often fail to leverage the rich semantic knowledge that humans naturally apply when understanding data [21]. Unlike computer vision or natural language processing domains where deep learning has achieved remarkable success through end-to-end learning, tabular data analysis has remained heavily dependent on manual feature engineering and domain expertise [22].

Recent advances in large language models, particularly with the development of models like GPT-4 [23] and their demonstrated few-shot learning capabilities, have opened new possibilities for automating feature engineering tasks. The fundamental premise underlying this research direction is that LLMs, trained on vast corpora of text that include descriptions of datasets, domain knowledge, and analytical procedures, can be prompted to generate meaningful feature transformations that would traditionally require human expertise.

2.3 Taxonomies and Methodological Frameworks

In this section can help to have a Figure like this

Create taxonomic classification figure showing: 1) Data-centric taxonomy: Data-driven vs Text-based methods, 2) Iterative vs Single-shot approaches, 3) LLM roles: Feature Selector vs Feature Generator vs Hybrid. Look for inspiration in: Li et al. Figure 1 and Han et al. workflow diagrams

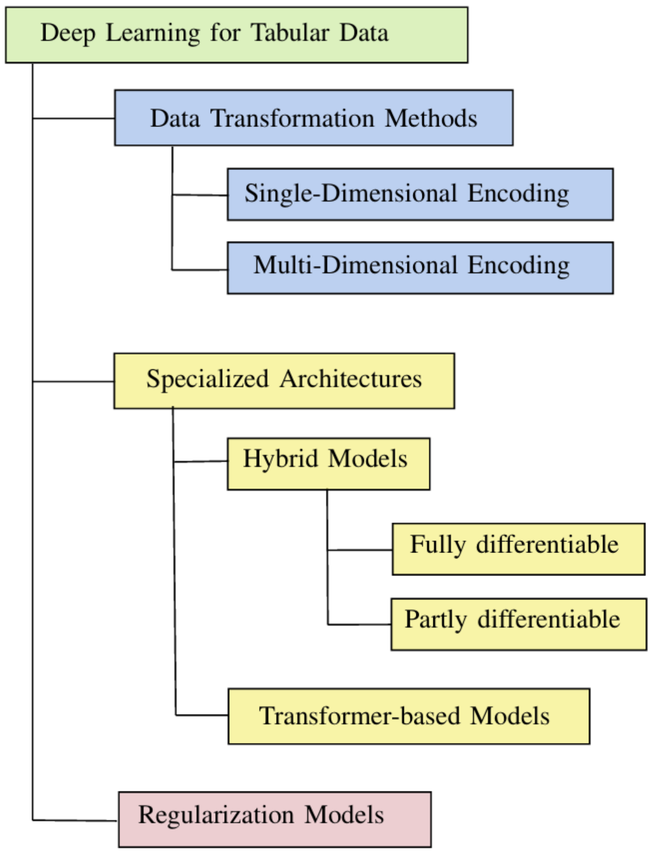
Deep Learning for Tabular Data
Data Transformation Methods
Single-Dimensional Encoding
Multi-Dimensional Encoding
Specialized Architectures
Hybrid Models
Fully differentiable
Partly differentiable
Transformer-based Models
Regularization Models

Figure 2.1. Esempio di tassonomia.

2.3.1 Data-Centric Categorization

Li et al. [21] propose a fundamental taxonomy that categorizes LLM-based feature selection methods into two distinct paradigms based on the type of information provided to the model:

Data-driven methods operate by providing specific data samples to LLMs, enabling them to perform statistical inference and correlation analysis. In this approach, the LLM receives feature values paired with target variable values, formatted as few-shot examples within the prompt. The model is then expected to infer relationships and assign importance scores based on observed patterns in the provided samples. Formally, given a dataset d with m samples, data-driven methods construct sample pairs $SP_i = \{(n^j_{f_i}, n^j_y)\}$ for feature f_i and target variable y, where $j \in \{1, ..., m\}$.

The prompt construction follows the pattern:

$$P^{Data}_{f_i} = \text{prompt}(C, SP_i) \tag{2.34}$$

where C represents the instruction context and SP_i contains the sample pairs for feature f_i.

Text-based methods, in contrast, leverage the extensive semantic knowledge embedded within LLMs by incorporating detailed dataset and feature descriptions into prompts. These approaches construct prompts using dataset descriptions (des_d) and feature-specific descriptions (des_{f_i}):

$$P^{Text}_{f_i} = \text{prompt}(C, des_d, des_{f_i}) \tag{2.35}$$

This methodological distinction proves crucial for understanding the relative strengths and limitations of different approaches, as empirically demonstrated across multiple evaluation studies.

2.3.2 Iterative Feature Generation Frameworks

Han et al. [24] introduce a more sophisticated framework that employs iterative feature generation specifically optimized for few-shot learning scenarios. Their approach addresses the fundamental challenge of limited training data by systematically leveraging LLMs' world knowledge to create meaningful feature representations through multiple generation cycles.

The iterative process operates through four interconnected phases that systematically build upon each other to achieve optimal feature generation outcomes.

The initial phase involves comprehensive context provision, where dataset descriptions and carefully selected representative samples are provided to the LLM to establish robust semantic understanding of the domain and task requirements. This phase is crucial because it sets the foundation for all subsequent feature generation by ensuring that the model has sufficient contextual information to generate meaningful and relevant features. The context provision goes beyond simple data descriptions to include domain-specific knowledge, task objectives, and examples that illustrate the types of patterns and relationships that are important for the specific application.

Subsequently, the framework focuses on intelligent feature generation, where the LLM leverages both its embedded world knowledge and the provided context to generate novel features through sophisticated domain knowledge application and pattern recognition capabilities. During this phase, the model draws upon its extensive training to identify potentially useful transformations, combinations, and derived features that might not be obvious from purely statistical analysis. The generation process is guided by the contextual understanding established in the previous phase, ensuring that generated features are both theoretically sound and practically relevant to the specific domain and task.

The framework then implements rigorous performance evaluation, where generated features undergo systematic validation using downstream classifiers through comprehensive cross-validation protocols. This evaluation phase is essential for determining which generated features actually provide predictive value and should be retained for the final feature set. The evaluation goes beyond simple accuracy metrics to consider factors such as feature stability, interpretability, and potential for overfitting, ensuring that only genuinely useful features are incorporated into the final model.

Finally, the process establishes adaptive iterative refinement, where the entire procedure repeats with updated context that incorporates lessons learned from previous iterations, including performance feedback and insights about which types of features proved most effective. This refinement process allows the system to continuously improve its feature generation capabilities by learning from both successes and failures in previous iterations. The iterative nature ensures that the feature engineering process can adapt to the specific characteristics of each dataset and task, rather than relying on generic feature generation strategies.

Add example from Han et al. 2024 paper showing iterative feature generation process with specific prompts and generated features

This framework demonstrates particular effectiveness in scenarios with limited training data (4-64 samples), where traditional feature engineering approaches typically struggle due to insufficient statistical power for reliable feature selection.

2.4 Empirical Findings and Performance Analysis

2.4.1 Comparative Effectiveness of Methodological Approaches

Extensive empirical evaluation across multiple studies reveals several consistent patterns regarding the relative effectiveness of different LLM-based approaches for feature engineering tasks.

Li et al. [21] demonstrate through comprehensive experiments across diverse datasets that text-based feature selection consistently outperforms data-driven methods in scenarios with limited training data. This superiority manifests across multiple critical dimensions that collectively establish the robustness of text-based approaches. Text-based methods consistently achieve higher performance metrics, including Area Under the Receiver Operating Characteristic curve (AUROC) for classification tasks and Mean Absolute Error (MAE) for regression tasks, across different dataset types ranging from medical and financial domains to social and behavioral datasets. The stability advantage is evident in the significantly lower variance in performance across different data availability settings, indicating that text-based approaches maintain consistent quality regardless of the specific sample size or data distribution characteristics. Finally, the robustness of text-based methods is demonstrated through their consistent performance regardless of specific dataset characteristics such as dimensionality, class balance, or feature correlation structures, making them more reliable for practical deployment across diverse application domains.

The authors report that text-based approaches using GPT-4 achieve performance comparable to traditional methods like Minimum Redundancy Maximum Relevance (MRMR) selection and Recursive Feature Elimination (RFE), while requiring no training data.

These metrics are not common, they should be introduced too.

The authors report that text-based approaches using GPT-4 achieve performance comparable to traditional automated feature selection methods, while requiring no training data.

2.4.2 Traditional Feature Selection Baselines

To properly contextualize LLM-based approaches, it is essential to understand the established traditional methods that serve as performance baselines in comparative studies.

Minimum Redundancy Maximum Relevance (MRMR) represents a foundational filter-based feature selection method that balances two competing objectives: maximizing relevance to the target variable while minimizing redundancy among selected features [**?**]. MRMR operates by selecting features that have high mutual information with the target variable but low mutual information with already selected features. Formally, given a set of features F and target variable y, MRMR selects feature subset S by optimizing:

$$\text{MRMR} = \arg \max_{f_i \in F \setminus S} \left[I(f_i; y) - \frac{1}{|S|} \sum_{f_j \in S} I(f_i; f_j) \right] \tag{2.36}$$

where $I(\cdot;\cdot)$ represents mutual information. This approach effectively addresses the common problem where individually relevant features may become redundant when combined.

Recursive Feature Elimination (RFE) employs a wrapper-based approach that iteratively removes features based on model performance feedback [**?**]. RFE trains a machine learning model on the current feature set, ranks features by importance (typically using model coefficients or feature importance scores), removes the least important features, and repeats until the desired number of features remains. This method's advantage lies in its model-aware selection process, ensuring that selected features work well together for the specific learning algorithm employed.

Both methods represent computationally intensive approaches that require full dataset access and extensive model training, contrasting sharply with LLM-based approaches that can operate with minimal data samples and no model training requirements.

Both techniques must be introduced. Maybe an introduction to standard feature engineering methodologies can be useful.

A significant finding across multiple studies is the strong correlation between LLM size and feature engineering effectiveness, particularly for text-based approaches. Li et al. observe clear scaling laws where larger models (GPT-4 vs. ChatGPT vs. LLaMA-2) demonstrate progressively better feature selection capabilities. This scaling behavior is less pronounced for data-driven methods, suggesting that semantic understanding rather than pure computational capacity drives effectiveness in feature engineering tasks.

A consistent limitation identified across studies is the degradation of data-driven methods as sample size increases beyond 64-128 samples. Li et al. attribute this phenomenon to LLMs' well-documented struggles with processing long sequences [25], which constrains the practical applicability of data-driven feature selection in real-world scenarios with abundant data.

2.4.3 Domain-Specific Applications and Specialization

The literature reveals particular promise for LLM-based feature engineering in specialized domains where traditional statistical methods may miss important semantic relationships.

Medical and biomedical applications represent a particularly promising area for LLM-based feature engineering. Li et al. introduce Retrieval-Augmented Feature Selection (RAFS) specifically designed for medical applications involving high-dimensional genomic data. RAFS addresses the challenge of domain-specific terminology by retrieving metadata from authoritative sources such as the National Center for Biotechnology Information (NCBI). In experiments with The Cancer Genome Atlas (TCGA) Lung Adenocarcinoma dataset, RAFS demonstrates significant improvements over random feature selection while maintaining privacy by avoiding direct data sharing.

The RAFS approach demonstrates measurable improvements across multiple evaluation metrics, achieving an Antolini's Concordance score of 0.6566 compared to 0.6516 for random feature selection, indicating better discrimination capability in survival prediction tasks. The method also shows improvement in the Integrated Brier Score, achieving 0.1830 versus 0.1833 for random selection, suggesting better calibration of predicted survival probabilities over time. Additionally, RAFS achieves superior D-Calibration performance with a score of 1.7666 compared to 2.0255 for random selection, indicating more reliable probability estimates across different risk groups.

2.4.4 Survival Analysis Evaluation Metrics

The medical domain applications employ specialized evaluation metrics designed for survival analysis that differ significantly from standard classification or regression metrics.

Antolini's Concordance Index (C-index) measures the discriminative ability of survival models by calculating the proportion of comparable pairs where the model correctly orders survival times [**?**]. For two patients i and j where patient i experiences an event before patient j, the C-index evaluates whether the model assigns a higher risk score to patient i. Formally, the C-index is defined as:

$$C\text{-index} = \frac{\sum_{i,j} \mathbf{1}_{t_i < t_j} \cdot \mathbf{1}_{\hat{f}(x_i) > \hat{f}(x_j)} \cdot \delta_i}{\sum_{i,j} \mathbf{1}_{t_i < t_j} \cdot \delta_i} \tag{2.37}$$

where t_i represents survival time, $\hat{f}(x_i)$ is the predicted risk score, and δ_i indicates whether the event was observed. Values range from 0.5 (random prediction) to 1.0 (perfect discrimination).

Integrated Brier Score (IBS) evaluates prediction accuracy over time by measuring the mean squared difference between predicted survival probabilities and observed outcomes across multiple time points [**?**]. The Brier Score at time t is:

$$BS(t) = \frac{1}{n} \sum_{i=1}^{n} \left[\mathbf{1}_{T_i \le t} - \hat{S}_i(t) \right]^2 \cdot w_i(t) \tag{2.38}$$

where $\hat{S}_i(t)$ is the predicted survival probability at time t for patient i, and $w_i(t)$ represents inverse probability weighting to handle censoring. The IBS integrates these scores over time, providing a comprehensive accuracy measure.

D-Calibration assesses the reliability of predicted survival probabilities by examining whether predicted probabilities match observed frequencies across different risk groups [?]. Perfect calibration occurs when patients predicted to have X% survival probability actually survive at that rate. Poor D-calibration indicates systematic over- or under-estimation of survival probabilities, even if discrimination (C-index) remains acceptable.

Han et al. demonstrate particular strength in complex domains requiring domain-specific knowledge, including medical prediction tasks and game-theoretic scenarios. Their framework achieves state-of-the-art performance across 13 diverse datasets, with particularly notable improvements in scenarios where traditional feature engineering struggles due to limited domain expertise availability.

2.5 Critical Perspectives and Limitations

2.5.1 Systematic Biases and Overly Simplistic Features

Küken et al. [26] provide essential critical analysis that tempers overly optimistic assessments of LLM capabilities in feature engineering. Their comprehensive evaluation across 27 datasets using multiple state-of-the-art models (GPT-4o-mini, Gemini-1.5-flash, Llama3.1-8B, Mistral7B-v0.3) reveals several systematic limitations.

LLMs demonstrate a tendency to repeatedly select small subsets of features with disproportionately high frequency, potentially overlooking important but less obvious relationships. This bias manifests as over-reliance on features that align with common patterns in training data rather than dataset-specific optimal features. Furthermore, analysis reveals that LLM-generated features often represent overly simplistic transformations that fail to capture complex, non-linear relationships present in tabular data. This limitation suggests that while LLMs excel at understanding semantic relationships, they may lack the sophisticated statistical reasoning required for optimal feature engineering.

Contrary to more optimistic reports, Küken et al. find that LLM-based methods show marginal or no significant improvement over established automated feature engineering approaches like OpenFE across many datasets. This finding highlights the importance of rigorous evaluation and realistic performance expectations when assessing the practical utility of LLM-based feature engineering approaches.

2.5.2 Computational and Practical Constraints

Several studies identify significant practical limitations that affect the real-world applicability of LLM-based feature engineering.

The iterative nature of most LLM-based approaches requires substantial computational resources, with Hollmann et al. [22] reporting average processing times of approximately 5 minutes per dataset for their CAAFE framework. While this may be acceptable for research contexts, it raises questions about scalability to production environments. Different LLMs exhibit varying levels of effectiveness for feature engineering tasks, with no single model consistently outperforming others across all dataset types. This inconsistency complicates the development of robust, generalizable approaches.

Current LLMs face constraints in processing datasets with large numbers of features due to context length limitations. Hollmann et al. address this by restricting evaluation to datasets with fewer than 20 features to remain within GPT-4's token limits. These practical constraints highlight the need for more efficient approaches that can handle larger datasets while maintaining computational feasibility.

2.6 Advanced Frameworks and Methodological Innovations

Some Figure that represent the workflow and some example prompt can help to understand the key differences of the techniques.

Add workflow diagram comparing different LLM feature engineering approaches. Should show: CAAFE vs FeatLLM vs LLM4FS workflows side by side Add concrete prompt examples from CAAFE paper: 1) Full prompt template (Figure 5 in CAAFE paper) showing dataset description + feature names + data types + sample data, 2) Example from Tic-Tac-Toe dataset showing "Dataset description: Tic-Tac-Toe Endgame database...", 3) Chain-of-thought template for code generation with feature explanation. Look in: CAAFE paper Appendix D for complete prompts and generated code examples

2.6.1 CAAFE: Context-Aware Automated Feature Engineering

Hollmann et al. [22] present the most comprehensive framework to date for LLM-based feature engineering, addressing many limitations identified in earlier work through sophisticated design choices and rigorous validation procedures.

CAAFE incorporates explicit human oversight mechanisms that prevent execution of potentially harmful or biased transformations. This design choice addresses growing concerns about AI bias propagation while maintaining the benefits of automated feature generation. The framework systematically incorporates multiple types of contextual information to enhance the quality and relevance of generated features. Dataset descriptions provide essential domain context that helps the LLM understand the underlying data generation process and the specific characteristics of the target domain. Feature metadata offers semantic meaning by explaining the conceptual significance of each variable, enabling the model to make more informed decisions about appropriate transformations. Sample data illustrates distributional characteristics, providing the LLM with concrete examples of data patterns and ranges that inform realistic feature generation strategies. Additionally, performance feedback from previous iterations creates a learning loop where the system can adapt its feature generation approach based on empirical validation results, progressively improving the quality of generated features through iterative refinement.

Unlike approaches that directly apply LLM-generated features, CAAFE employs rigorous cross-validation to ensure that only features providing genuine predictive improvement are retained. This validation mechanism provides protection against overfitting and spurious feature generation. Furthermore, CAAFE generates human-readable Python code for feature transformations, enabling manual inspection and modification. This transparency addresses interpretability concerns while allowing domain experts to understand and potentially refine generated features.

Empirical evaluation demonstrates that CAAFE achieves meaningful performance improvements, with average ROC AUC increasing from 0.798 to 0.822 across evaluated datasets when using TabPFN as the downstream classifier.

2.6.2 Bias Mitigation and Ethical Considerations

The literature increasingly recognizes the critical importance of addressing bias propagation and ethical concerns in LLM-based feature engineering systems.

Hollmann et al. identify three primary levels where bias can manifest in LLM-based feature engineering systems. At the model level, biases originate from the training data and model parameters, reflecting historical prejudices and discriminatory patterns present in the large-scale text corpora used to train these language models. At the feature generation level, biases are introduced through feature selection and transformation processes, where the model may systematically favor certain types of transformations or patterns that encode societal stereotypes or discriminatory associations. Finally, at the downstream classifier level, biases affect final model predictions through the interaction between potentially biased features and the classification algorithm, amplifying discriminatory signals that may have been subtle or implicit in the original feature space.

2.6.3 Hybrid Approaches: Combining LLM Reasoning with Traditional Methods

A promising research direction emerges from hybrid approaches that seek to combine the semantic reasoning capabilities of Large Language Models with the statistical rigor of traditional data-driven feature selection methods. Li and Xiu [27] introduce the LLM4FS framework, which represents a significant methodological innovation in this space.

LLM4FS operates by providing approximately 200 data samples (typically representing 20% or less of the total dataset) directly to LLMs, instructing them to apply traditional data-driven techniques such as random forest, forward sequential selection, backward sequential selection, recursive feature elimination (RFE), minimum redundancy maximum relevance (MRMR), and mutual information (MI). This approach leverages the contextual understanding capabilities of LLMs while maintaining the statistical reliability of established feature selection methods.

The hybrid framework addresses fundamental limitations of pure LLM-based approaches by incorporating the robustness of traditional methods while overcoming their semantic limitations. Unlike purely data-driven LLM approaches that struggle with long sequences, or purely text-based approaches that may lack statistical grounding, the hybrid method provides a principled way to combine complementary strengths from both paradigms.

Comprehensive evaluation across multiple datasets (Bank, Credit-G, Pima Indians Diabetes, Give Me Some Credit) demonstrates that LLM4FS achieves superior performance compared to both standalone LLM-based methods and traditional approaches when evaluated individually. The framework shows particular effectiveness when using state-of-the-art models like DeepSeek-R1, which demonstrates performance comparable to GPT-4.5 while offering significantly better cost-efficiency.

The authors report that DeepSeek-R1 exhibits consistently strong performance across all evaluated datasets, with output costs approximately 50% of GPT-o3-mini and only 1.5% of GPT-4.5. This cost-effectiveness, combined with robust performance, makes the hybrid approach particularly attractive for practical deployment scenarios where computational budget constraints are significant.

A key advantage of the hybrid approach is its demonstrated stability in feature selection across different data availability scenarios. Unlike pure LLM-based methods that may exhibit inconsistent performance across different sample sizes or dataset characteristics, LLM4FS maintains reliable performance through the incorporation of established statistical methods. The framework shows particular stability in the 10%-30% feature selection range, which is often the most practically relevant scenario for real-world applications.

The hybrid approach achieves an advantageous trade-off between computational efficiency and performance quality. While pure LLM-based methods may require extensive iterative prompting and traditional methods may require full dataset analysis, LLM4FS achieves competitive performance with reduced computational overhead by leveraging the reasoning capabilities of LLMs to guide the application of efficient traditional methods on smaller data samples.

Several complementary approaches are proposed to address these multiple sources of bias, each targeting different aspects of the feature generation and validation pipeline. Explicit bias detection operates through systematic analysis of generated feature explanations, requiring the LLM to provide detailed justifications for why specific features are considered important, thereby making potentially biased reasoning visible to human reviewers. Human oversight mechanisms establish mandatory checkpoints requiring manual approval of generated transformations before they can be applied to the dataset, ensuring that domain experts can identify and reject features that may encode inappropriate biases or discriminatory patterns. Cross-validation filtering implements automated screening processes that systematically discard features leading to biased outcomes across different demographic groups, using fairness metrics to identify features that disproportionately impact protected classes. Finally, transparency requirements mandate that all feature generation processes produce interpretable, human-readable code and explanations, enabling post-hoc analysis and audit trails that support accountability and bias detection in deployed systems.

The authors provide a concrete example using a synthetic dataset predicting doctor vs. nurse classification based on names. Their analysis reveals how GPT-4 generates features that rely on gender-associated name endings, demonstrating how societal biases can be inadvertently encoded in automated feature engineering systems.

Add the specific bias example from CAAFE paper (Appendix B.2): Dataset: Doctor-or-Nurse prediction based on names, Biased sample: Names like “Anna”, “Jack”, “Frank”, “Laura” where male names = doctor, GPT-4 generated biased feature: df[‘end_with_a’] = df[‘Name’].str.endswith(‘a’).astype(int). Shows how LLMs can perpetuate gender stereotypes in automated feature engineering

Chapter 3

Methodology

Chapter 4

Experimental Setup

4.1 Comparison of Traditional Feature Engineering Methods

4.1.1 Electricity Dataset

Method	**Model**	**Feat.**	**Accuracy**	**ROC-AUC**	**F1-Score**	**MCC**
Baseline	LR	8	0.7552±0.0001	0.8195±0.0001	0.7495±0.0002	0.4938±0.0003
Baseline	MLP	8	0.8233±0.0013	0.9019±0.0007	0.8226±0.0012	0.6367±0.0025
Baseline	LGBM	8	0.8816±0.0007	0.9543±0.0002	0.8815±0.0007	0.7573±0.0014
Featuretools	LR	50	0.7622±0.0001	0.8305±0.0001	0.7579±0.0001	0.5080±0.0003
Featuretools	MLP	50	0.8216±0.0010	0.9017±0.0004	0.8210±0.0008	0.6336±0.0015
Featuretools	LGBM	50	0.8892±0.0005	0.9585±0.0001	0.8891±0.0005	0.7728±0.0010
AutoGluon	LR	44	0.7791±0.0007	0.8529±0.0001	0.7771±0.0008	0.5440±0.0015
AutoGluon	MLP	44	0.8278±0.0017	0.9074±0.0008	0.8271±0.0018	0.6462±0.0037
AutoGluon	LGBM	44	0.8838±0.0017	0.9556±0.0004	0.8836±0.0017	0.7616±0.0035

Table 4.1. Results on electricity dataset (45,312 samples, 8 original features). Mean ± standard deviation over 3 trials.

4.1.2 Phoneme Dataset

Method	Model	Feat.	Accuracy	ROC-AUC	F1-Score	MCC
Baseline	LR	5	0.7515±0.0014	0.8119±0.0004	0.7416±0.0016	0.3649±0.0041
Baseline	MLP	5	0.8590±0.0024	0.9256±0.0019	0.8584±0.0026	0.6577±0.0067
Baseline	LGBM	5	0.8952±0.0030	0.9524±0.0011	0.8950±0.0030	0.7468±0.0070
Featuretools	LR	50	0.8119±0.0024	0.8798±0.0004	0.8093±0.0025	0.5358±0.0062
Featuretools	MLP	50	0.8773±0.0020	0.9326±0.0018	0.8770±0.0021	0.7032±0.0053
Featuretools	LGBM	50	0.9041±0.0008	0.9586±0.0020	0.9039±0.0008	0.7681±0.0021
AutoGluon	LR	20	0.8164±0.0016	0.8839±0.0002	0.8150±0.0015	0.5511±0.0037
AutoGluon	MLP	20	0.8798±0.0015	0.9402±0.0006	0.8791±0.0011	0.7080±0.0020
AutoGluon	LGBM	20	0.9019±0.0019	0.9569±0.0012	0.9019±0.0019	0.7635±0.0049

Table 4.2. Results on phoneme dataset (5,404 samples, 5 original features). Mean ± standard deviation over 3 trials.

4.1.3 KC1 Dataset

Method	Model	Feat.	Accuracy	ROC-AUC	F1-Score	MCC
Baseline	LR	21	0.8559±0.0025	0.8024±0.0019	0.8241±0.0030	0.2890±0.0171
Baseline	MLP	21	0.8630±0.0017	0.7939±0.0069	0.8408±0.0027	0.3620±0.0111
Baseline	LGBM	21	0.8551±0.0043	0.7844±0.0086	0.8400±0.0049	0.3559±0.0200
Featuretools	LR	50	0.8565±0.0017	0.8020±0.0020	0.8246±0.0015	0.2925±0.0099
Featuretools	MLP	50	0.8601±0.0025	0.7910±0.0061	0.8364±0.0026	0.3410±0.0100
Featuretools	LGBM	50	0.8544±0.0026	0.7860±0.0037	0.8397±0.0031	0.3549±0.0141
AutoGluon	LR	50	0.8566±0.0012	0.7988±0.0015	0.8251±0.0026	0.2939±0.0110
AutoGluon	MLP	50	0.8622±0.0020	0.7968±0.0033	0.8396±0.0038	0.3553±0.0154
AutoGluon	LGBM	50	0.8521±0.0026	0.7762±0.0101	0.8368±0.0032	0.3422±0.0145

Table 4.3. Results on KC1 dataset (2,109 samples, 21 original features). Mean ± standard deviation over 3 trials.

Chapter 5

Results and Analysis

Chapter 6

Conclusions and Future Work

Bibliography

[1] Pablo Duboue. *The Art of Feature Engineering: Essentials for Machine Learning.* Cambridge University Press, Cambridge, 2020. A comprehensive guide to manual and automated feature engineering techniques for machine learning applications.

[2] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[3] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction.* Springer Science & Business Media, 2nd edition, 2009.

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[5] Randy Kerber. Chimerge: Discretization of numeric attributes. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 123–128. AAAI Press, 1992.

[6] Usama Fayyad and Keki Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1022–1029, Tahoe City, CA, 1993. Morgan Kaufmann.

[7] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[8] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees.* The Wadsworth Statistics/Probability series. Wadsworth & Brooks/Cole Advanced Books & Software, Monterey, CA, 1984.

[9] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

[10] Henrik Brink, Joseph W. Richards, and Mark Fetherolf. *Real-World Machine Learning.* Manning, Shelter Island, NY, 2017.

[11] Sudhir Varma and Richard Simon. Bias in error estimation when using cross-validation for model selection. *BMC Bioinformatics*, 7(1):91, 2006.

[12] Rob J Hyndman and George Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2nd edition, 2018.

[13] Sarah Nogueira, Konstantinos Sechidis, and Gavin Brown. On the stability of feature selection algorithms. *Journal of Machine Learning Research*, 18(1):6345–6398, 2018.

[14] Carol Friedman, George Hripcsak, William DuMouchel, Stephen B Johnson, and Paul D Clayton. A natural language processing system to extract and code family histories from electronic health records. *Journal of the American Medical Informatics Association*, 20(e2):e238–e244, 2013.

[15] Amir E Khandani, Adlar J Kim, and Andrew W Lo. Consumer credit-risk models via machine-learning algorithms. *Journal of Banking & Finance*, 34(11):2767–2787, 2010.

[16] James Max Kanter and Kalyan Veeramachaneni. Deep feature synthesis: Towards automating data science endeavors. In *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–10. IEEE, 2015.

[17] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *Advances in neural information processing systems*, volume 28, pages 2962–2970, 2015.

[18] Randal S Olson, Ryan J Urbanowicz, Peter C Andrews, Nicole A Lavender, La Creis Kidd, and Jason H Moore. Tpot: A tree-based pipeline optimization tool for automating machine learning. In *Workshop on automatic machine learning*, pages 66–74. PMLR, 2016.

[19] Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. Autogluon-tabular: Robust and accurate automl for structured data. *arXiv preprint arXiv:2003.06505*, 2020.

[20] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[21] Dawei Li, Zhen Tan, and Huan Liu. Exploring large language models for feature selection: A data-centric perspective. *arXiv preprint arXiv:2408.12025*, 2024. Accepted by SIGKDD Explorations (December 2024).

[22] Noah Hollmann, Katharina Eggensperger, and Frank Hutter. Large language models for automated data science: Introducing caafe for context-aware automated feature engineering. *arXiv preprint arXiv:2405.16374*, 2024.

[23] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2024.

[24] Sungwon Han, Jinsung Yoon, Sercan O. Arik, and Tomas Pfister. Large language models can automatically engineer features for few-shot tabular learning. In *Forty-first International Conference on Machine Learning*. PMLR, 2024.

[25] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.

[26] Tim Küken, Maximilian Malte, Bernd Bischl, and Giuseppe Casalicchio. Large language models engineer too many simple features for tabular data. *arXiv preprint arXiv:2410.14741*, 2024.

[27] Jianhao Li and Xianchao Xiu. Llm4fs: Leveraging large language models for feature selection and how to improve it. *arXiv preprint arXiv:2503.24157*, 2025.