Statistical Test for Auto Feature Engineering by Selective Inference

Tatsuya Matsukawa* Nagoya University

Tomohiro Shiraishi* Nagoya University

Shuichi Nishino Nagoya University RIKEN

Teruyuki Katsuoka Nagoya University

Ichiro Takeuchi Nagoya University RIKEN

Abstract

Auto Feature Engineering (AFE) plays a crucial role in developing practical machine learning pipelines by automating the transformation of raw data into meaningful features that enhance model performance. By generating features in a data-driven manner, AFE enables the discovery of important features that may not be apparent through human experience or intuition. On the other hand, since AFE generates features based on data, there is a risk that these features may be overly adapted to the data, making it essential to assess their reliability appropriately. Unfortunately, because most AFE problems are formulated as combinatorial search problems and solved by heuristic algorithms, it has been challenging to theoretically quantify the reliability of generated features. To address this issue, we propose a new statistical test for generated features by AFE algorithms based on a framework called selective inference. As a proof of concept, we consider a simple class of tree search-based heuristic AFE algorithms, and consider the problem of testing the generated features when they are used in a linear model. The proposed test can quantify the statistical significance of the generated features in the form of p-values, enabling theoretically guaranteed control of the risk of false findings.

Proceedings of the 28[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258.

1 Introduction

Feature engineering (FE), which is used to extract important features from raw data, plays a crucial role in many practical machine learning tasks. While FE is often performed through trial and error based on experts' domain knowledge, data-driven approaches known as *auto feature engineering (AFE)* are also promising and are being actively researched. By generating features in a data-driven manner, AFE enables the discovery of important features that may not be apparent through human experience or intuition. However, since AFE generates features based on data, there is a risk that these features may be overly adapted to the data, making it essential to appropriately assess their reliability. In this study, we explore how to quantify the statistical reliability of features automatically generated by AFEs.

The AFE problem is essentially formulated as a combinatorial optimization problem because features are generated by recursively applying various operations to raw features and previously generated features. Figure 1 illustrates an example of feature engineering, where features such as $x_1 \sin(x_3) + \ln(|x_5|)$, $\exp(x_4) + \sin(x_2 x_3)$, and $x_4/x_2 + \cos(x_1)$ are generated by recursively applying various operations to the original raw features $x_1, x_2, \ldots, x_5$, and previously generated features. Therefore, most AFE algorithms are developed with the aim of finding approximate solutions to combinatorial search problems using heuristic search methods. Since the properties of approximate solutions identified through such heuristic searches are difficult to characterize, quantifying the statistical uncertainties of the generated features is highly challenging. As a result, to the best of our knowledge, no research has been conducted on statistical inference for the features generated by AFE algorithms.

*Equal contribution.

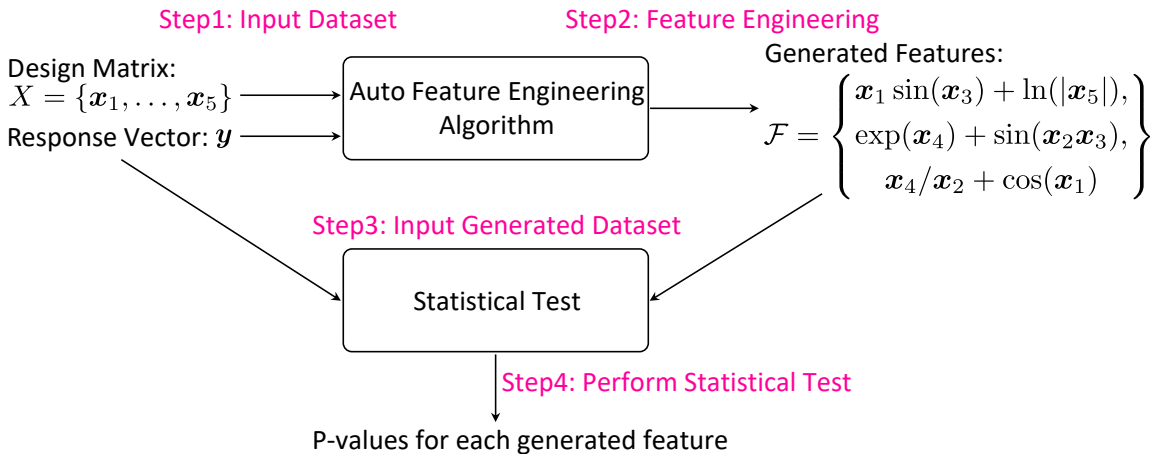


Figure 1: Overview of the proposed framework for evaluating the significance of features generated by an AFE algorithm. The input dataset $(X, \boldsymbol{y})$ is processed by an AFE algorithm to generate a set of features $\mathcal{F}$. The significance of each generated feature is then evaluated using a proposed selective inference method. Note that directly applying traditional statistical tests to generated features can lead to inflated type I error rates, as these features are data-dependent.

To address this issue, we propose a new statistical test for automatically generated features by heuristic AFE algorithms, based on a framework called selective inference (SI). SI is a statistical framework that allows for valid inference after data-dependent hypothesis selection procedures. The basic idea of our proposed test is to conduct a statistical test conditional on the fact that the feature is generated by an AFE algorithm, which might otherwise lead to overfitting and inflated type I error rates. By conditioning on the feature generation event, our proposed test adjusts for the bias introduced by the feature generation process, providing valid statistical inference for the generated features. Our proposed test provides p-values for the automatically generated features, allowing users to assess their statistical significance and control the risk of falsely identifying irrelevant features. This ensures that features passing the our proposed test, even those generated by heuristic AFE algorithms, can be confidently utilized in subsequent analyses.

In this paper, as a proof of concept, we consider a simple class of tree search-based heuristic AFE algorithms and address the problem of testing the generated features when they are used in a linear model. We validate the effectiveness of our proposed method through experiments on both synthetic and real-world datasets. The results demonstrate that our proposed test provides valid p-values for features generated by AFE algorithms. For reproducibility, our implementation is available at `https://github.com/Takeuchi-Lab-SI-Group/statistical_test_for_auto_feature_engineering`.

Related Work. Various heuristic AFE algorithms have been proposed in the literature. For example, *Data Science Machine* (Kanter and Veeramachaneni, 2015) uses singular value decomposition to select generated features, while *autofeat* (Horn et al., 2020) performs feature generation based on L_1 regularization. The limitation of all these heuristic strategies is that calculating all possible candidate features is computationally expensive. To address this computational issue, many studies have adopted strategies that recursively generate more and more complex features (Kaul et al., 2017; Piramuthu and Sikora, 2009; Shi et al., 2020; Zhang et al., 2022; Khurana et al., 2016; Markovitch and Rosenstein, 2002). Additionally, methods using popular meta-heuristics such as genetic algorithms have also been explored (Smith and Bull, 2003; Shi and Saad, 2023; Khan and Byun, 2020). However, to the best of our knowledge, none of these approaches have focused on statistical reliability of generated features. One common heuristic adopted by many AFE methods is known as the *Expand-Reduce* strategy (Kanter and Veeramachaneni, 2015; Horn et al., 2020; Lam et al., 2021). This strategy consists of an *expansion step*, where feature candidates are generated from the original raw features or previously generated features, and a *reduction step*, where features that contribute to the model's performance are selected. In this study, as an example of commonly used heuristics in AFE, we address the problem of statistical testing for features generated by simple class of expand-reduce strategy.

SI has gained attention as a method of statistical in-

ference for feature selection in linear models (Taylor and Tibshirani, 2015; Fithian et al., 2015). SI for various feature selection algorithms, such as marginal screening (Lee and Taylor, 2014), stepwise FS (Tibshirani et al., 2016), and Lasso (Lee et al., 2016), has been explored and extended to more complex feature selection methods (Yang et al., 2016; Suzumura et al., 2017; Hyun et al., 2018; Rügamer and Greven, 2020; Das et al., 2021). SI proves valuable not only for FS in linear models but also for statistical inference across various data-driven hypotheses, including unsupervised learning tasks such as OD (Chen and Bien, 2020; Tsukurimichi et al., 2021), segmentation (Tanizaki et al., 2020; Duy et al., 2022; Le Duy et al., 2024), clustering (Lee et al., 2015; Gao et al., 2022), change-point detection (Duy et al., 2020; Jewell et al., 2022). Moreover, SI is being applied not only to linear models but also to more complex models, such as kernel models (Yamada et al., 2018), tree-structured models (Neufeld et al., 2022), Neural Networks (Duy et al., 2022; Miwa et al., 2023; Shiraishi et al., 2024b), and data analysis pipelines (Shiraishi et al., 2024a). However, existing SI studies have focused only on the inference of features selected from predefined candidates. To our knowledge, there are no existing SI studies that perform inference on newly generated features as in AFE.

2 PRELIMINARIES

Auto feature engineering (AFE) is the process of selecting, transforming, and creating relevant features from raw dataset to improve the performance of machine learning models. In this study, we propose a framework for evaluating the significance of features generated by an AFE algorithm using statistical hypothesis testing. Figure 1 illustrates the overview of the proposed framework. The AFE algorithm considered in this study will be discussed in detail in §3.

Problem Setup. Let us consider AFE for regression problem with n instances and m features. We denote the dataset $(X, \boldsymbol{y})$, where $X = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m) \in \mathbb{R}^{n\times m}$ is the fixed design matrix and $\boldsymbol{y} \in \mathbb{R}^n$ is the response vector. We assume that the observed response vector $\boldsymbol{y}$ is a random realization of the following random response vector

$$\boldsymbol{Y} = \boldsymbol{\mu}(X) + \boldsymbol{\varepsilon},\ \boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{0}, \Sigma) \tag{1}$$

where $\boldsymbol{\mu}(X) \in \mathbb{R}^n$ is the unknown true value function, while $\boldsymbol{\varepsilon} \in \mathbb{R}^n$ is normally distributed noise with covariance matrix Σ which is known or estimable from an independent dataset[1]. Although we do not pose any functional form on the true value function $\boldsymbol{\mu}(X)$ for theoretical justification, we consider a scenario where the true values $\boldsymbol{\mu}(X)$ can be well approximated with a linear model by adding a feature set $\mathcal{F} \in \mathbb{R}^{n\times k}$, generated from the dataset $(X, \boldsymbol{y})$, to the design matrix X, where k is the number of generated features. In this case, the true value function can be approximated as $\boldsymbol{\mu}(X) \approx [X, \mathcal{F}]\boldsymbol{\beta}$ with $(m+k)$-dimensional coefficient vector $\boldsymbol{\beta}$. This is a common setting in the field of SI, referred to as the *saturated model* setting. Using the above notations, an AFE algorithm is represented as a mapping:

$$f_{\mathrm{AFE}} \colon \mathbb{R}^{n\times m} \times \mathbb{R}^n \ni (X, \boldsymbol{y}) \mapsto \mathcal{F} \in \mathbb{R}^{n\times \mathbb{N}}, \tag{2}$$

whose specific procedure are detailed in §3.

Statistical Test for Generated Features. Given the generated features by the AFE algorithm in (2), the statistical significance of the generated features can be quantified based on the coefficients of the linear model fitted with the concatenated design matrix $[X, \mathcal{F}]$. To formalize this, we denote the design matrix after adding the generated features as $X_{\mathcal{F}} = [X, \mathcal{F}] \in \mathbb{R}^{n\times(m+k)}$ for simplicity. Using these notations, the least-squares solution of the linear model after adding the generated features is expressed as

$$\hat{\boldsymbol{\beta}} = \left(X_{\mathcal{F}}^\top X_{\mathcal{F}}\right)^{-1} X_{\mathcal{F}}^\top \boldsymbol{y}.$$

Similarly, we consider the population least-square solution for the unobservable true value vector $\boldsymbol{\mu}(X)$ in (1), which is defined as

$$\boldsymbol{\beta}^* = \left(X_{\mathcal{F}}^\top X_{\mathcal{F}}\right)^{-1} X_{\mathcal{F}}^\top \boldsymbol{\mu}(X).$$

To quantify the statistical significance of the generated features, we consider the following null hypothesis and the alternative hypothesis:

$$\mathrm{H}_{0,j} \colon \beta_j^* = 0 \text{ v.s. } \mathrm{H}_{1,j} : \beta_j^* \neq 0,\ j \in \{m+1, \ldots, m+k\}. \tag{3}$$

In this testing framework, we focus on testing the coefficient of the generated feature itself, thus we let $j \in \{m+1, \ldots, m+k\}$. Note that if we were to consider $j \in [m]$, the test would assess the coefficients of the original features within a linear model augmented with the generated features. The subsequent discussion applies in exactly the same way to the both cases. Therefore, for simplicity, we assume $j \in [m+k]$ for the remainder of this paper.

Selective Inference (SI). For the statistical test in (3), it is reasonable to use $\hat{\beta}_j, j \in [m+k]$ as the test statistic. An important point when addressing this

[1]We discuss the robustness of the proposed method when the covariance matrix is unknown and the noise deviates from the Gaussian distribution in §6.

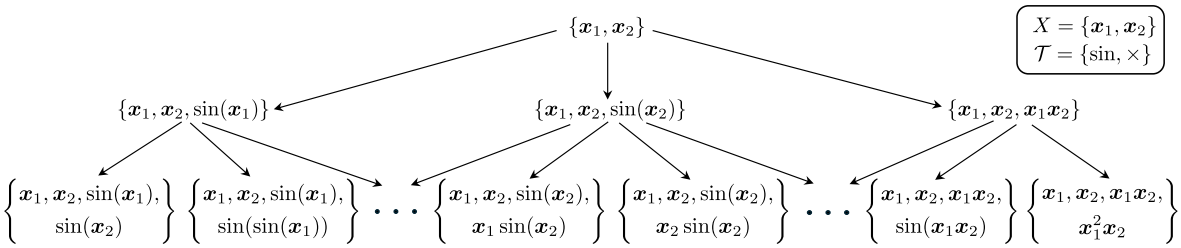


Figure 2: Schematic illustration of the feature generation process as a directed tree search.

statistical test within the SI approach is that the test statistic is represented as a linear function of the observed response vector as $\hat{\beta}_j = \boldsymbol{\eta}_j^\top \boldsymbol{y}, j \in [m+k]$, where $\boldsymbol{\eta}_j \in \mathbb{R}^n,\ j \in [m+k]$ is a vector that depends on $\boldsymbol{y}$ only through the generated features $\mathcal{F}$. In SI, this property is utilized to perform statistical inference based on the sampling distribution of the test statistic conditional on $\mathcal{F}$. More specifically, since $\boldsymbol{y}$ follows a normal distribution, it can be derived that the sampling distribution of the test statistic $\hat{\beta}_j = \boldsymbol{\eta}_j^\top \boldsymbol{y},\ j \in [m+k]$ conditional on $\mathcal{F}$ and the sufficient statistic of the nuisance parameters follows a truncated normal distribution. By computing p-values based on this conditional sampling distribution represented as a truncated normal distribution, it is ensured that the type I error can be controlled even in finite samples. For more details on SI, please refer to the explanations in §4 or literatures such as Taylor and Tibshirani (2015); Fithian et al. (2015); Lee and Taylor (2014).

3 AUTO FEATURE ENGINEERING

In this section, we describe the AFE algorithm f_{AFE} in (2) used in this study. Note that the novelty of this study is to propose a method for quantifying the statistical significance of the generated features by the AFE algorithm, not the AFE algorithm itself.

3.1 Directed Tree Search Algorithm

As a simple AFE algorithm, we consider a procedure that iteratively applies a predefined set of transformations $\mathcal{T} = \{t_1, \ldots, t_{|\mathcal{T}|}\}$ to the original features $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}$, and selects the resulting features based on their fitness (or relevance) to the response vector $\boldsymbol{y}$. Examples of such transformations include elementary functions (e.g., sine, cosine, exponential) and arithmetic operations (e.g., multiplication and division).

To formalize this AFE algorithm, we employ a directed tree structure to represent its process. In this tree structure, each node represents a set of features (note that the root node corresponds to the original set of features $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}$), and each directed edge signifies the addition of a new feature, which is generated by applying a transformation from $\mathcal{T}$ to one feature in the parent node. Each path from the root to any other node represents a sequence of transformations applied to the original features. Thus, the AFE algorithm can be viewed as a search within this directed tree for a node containing a suitable set of features, evaluated based on their fitness to the response vector $\boldsymbol{y}$. A schematic illustration of the feature generation process is shown in Figure 2.

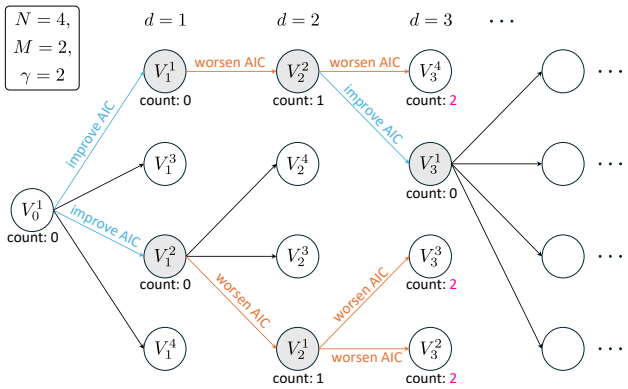


Figure 3: Schematic illustration of the AFE algorithm. The search proceeds roughly as follows: for each increase in depth, $N(=4)$ nodes are randomly generated (each node has a feature set that adds one new feature to the parent node). Here, V_i^j denotes a node, with the subscript i denoting the depth of the node and the superscript j denoting the rank of the node within the same depth, ordered by the AIC. Then, $M(=2)$ nodes with the best AIC among the N generated nodes are selected as parents of the next depth nodes (nodes filled in gray). This operation is repeated until the maximum depth D is reached. Furthermore, to handle exceptions, we check whether the AIC has improved compared to the best AIC at the previous depth and record the number of consecutive times the AIC has not improved for each node (note that it is reset to 0 at V_3^1). The AFE algorithm prevents the node with this counter value greater than or equal to $\gamma(=2)$ from becoming parent node. In this figure, at depth 3, all nodes except V_3^1 cannot become parents, then the next feature generation is performed only from V_3^1.

Algorithm 1 Overall of the AFE Algorithm

Require: Set of original features $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}$, response vector $\boldsymbol{y}$.

Initialize the set of nodes at depth 0 as $\mathcal{V}_0 = \{X\}$
Cache the best AIC at depth 0 as $\texttt{best\_aic}[0] = \mathrm{AIC}(X, \boldsymbol{y})$
Cache the consecutive non-improvement count for node X as $\texttt{no\_improve}[X] = 0$
for each depth $d \in \{0, \ldots, D-1\}$ **do**
  Initialize the set of nodes at depth $d+1$ as $\mathcal{V}_{d+1} = \emptyset$
  while $|\mathcal{V}_{d+1}| < N$ and new feature $\boldsymbol{f}$ can be generated from $\mathcal{V}_d$ and $\mathcal{T}$ **do**
    Randomly select a node $V \in \mathcal{V}_d$ to expand
    Generate new feature $\boldsymbol{f} = t(\boldsymbol{x})$, where $(t, \boldsymbol{x})$ are randomly selected from $\mathcal{T} \times V$
    if feature $\boldsymbol{f}$ do not exhibit multicollinearity with V **then**
      Add the node representing feature set $V \cup \{\boldsymbol{f}\}$ to $\mathcal{V}_{d+1}$
    end if
  end while
  Reduce nodes as $\mathcal{V}_{d+1} = \texttt{ReducingNode}(\mathcal{V}_{d+1})$ based on the AIC (see Algorithm 2)
  if $\mathcal{V}_{d+1}$ is empty **then**
    Decrement the depth d and break the for loop
  end if
end for

Ensure: Generated features in the node with the best AIC in $\mathcal{V}_{d+1}$, i.e., $\arg\min_{V \in \mathcal{V}_{d+1}} \mathrm{AIC}(V, \boldsymbol{y}) \setminus X$

3.2 Details of AFE Algorithm

The search space of the directed tree, as described above, is exponentially large, making it computationally infeasible to explore all possible nodes exhaustively. To address this issue, we propose a simple directed tree search algorithm governed by four hyperparameters: the maximum depth D, the maximum number of nodes to be generated N at each depth, the maximum number of nodes used for generation M at each depth, and the tolerance parameter γ for cases where increasing the depth does not lead to an improvement in fitness. To evaluate the fitness of a node to the response vector $\boldsymbol{y}$, we adopt the Akaike information criterion (AIC) in this study. The AIC is a widely used criterion for model selection, balancing the goodness of fit and the complexity of the model. In our context, it favors features that fit the response vector $\boldsymbol{y}$ well while penalizing excessive feature generation. A schematic illustration of the AFE algorithm as a directed tree search is shown in Figure 3, and the overall algorithm is summarized in Algorithm 1.

The AFE algorithm, evident from its procedure, is significantly influenced by its hyperparameters. For instance, 1) setting large D and γ, a small N, and $M = 1$ results in the generation of a few parent node candidates and exploration deep into the graph without fear of AIC deterioration, which is similar to the behavior of depth-first search. 2) setting small D and γ, large N and M leads to exploring a greater number of potential nodes at each depth without going too deep, which is similar to the behavior of breadth-first search.

Algorithm 2 Reducing Node

Require: Set of nodes $\mathcal{V}_{d+1}$ at depth $d+1$

Number each node of $\mathcal{V}_{d+1}$ in ascending order of the AIC as $\mathcal{V}_{d+1} = \{V_{d+1}^1, \ldots, V_{d+1}^{|\mathcal{V}_{d+1}|}\}$
Cache the best AIC at depth $d+1$ as $\texttt{best\_aic}[\texttt{d}+1] = \mathrm{AIC}(V_{d+1}^1, \boldsymbol{y})$
Initialize set of nodes to be returned as $\mathcal{V}'_{d+1} = \emptyset$
for each node $V \in \mathcal{V}_{d+1}$ **do**
  Cache count $\texttt{no\_improve}[\texttt{V}]$ as 0 if $\mathrm{AIC}(V, \boldsymbol{y}) < \texttt{best\_aic}[\texttt{d}]$, else $\texttt{no\_improve}[\texttt{pa(V)}] + 1$
  if $\texttt{no\_improve}[\texttt{V}] < \gamma$ **then**
    Add node V to $\mathcal{V}'_{d+1}$
  end if
  if $|\mathcal{V}'_{d+1}| = M$ **then**
    Break the for loop
  end if
end for

Ensure: $\mathcal{V}'_{d+1}$

4 SELECTIVE INFERENCE FOR GENERATED FEATURES

To perform statistical hypothesis testing for generated features, it is necessary to consider how the data influenced the finally generated features through the whole process of the AFE algorithm. We address this challenge by utilizing the SI framework. In the SI framework, statistical test is performed based on the sampling distribution conditional on the process by which the features are generated from the data[2], thereby incorporating the influence of how data is used to generate the features.

4.1 Concept of Selective Inference

In SI, p-values are computed based on the null distribution conditional on an event that a certain hypothesis is generated. The goal of SI is to compute a p-value

[2]This can also be interpreted as the process by which the data selects the generated features from the set of all potentially generated features. In the context of SI, this interpretation as a selection from the range of the algorithm based on the data is more natural.

such that

$$\mathbb{P}_{\mathrm{H}_0}\left(p \leq \alpha \mid \mathcal{F}_{\boldsymbol{Y}} = \mathcal{F}\right) = \alpha,\ \forall \alpha \in (0,1), \tag{4}$$

where $\mathcal{F}_{\boldsymbol{Y}}$ indicates the random set of generated features given the random response vector $\boldsymbol{Y}$ and the fixed design matrix X, thereby making the p-value is a random variable. Here, the condition part $\mathcal{F}_{\boldsymbol{Y}} = \mathcal{F}$ in (4) indicates that we only consider response vectors $\boldsymbol{Y}$ from which a certain set of features are generated. If the conditional type I error rate can be controlled as in (4) for all possible hypotheses $\mathcal{F} \in \mathcal{F}(X)$ where $\mathcal{F}(X)$ represents the set of all possible generated features from the fixed design matrix X, then, by the law of total probability, the marginal type I error rate can also be controlled for all $\alpha \in (0,1)$ because

$$\mathbb{P}_{\mathrm{H}_0}(p \leq \alpha) = \sum_{\mathcal{F} \in \mathcal{F}(X)} \mathbb{P}_{\mathrm{H}_0}(\mathcal{F}) \mathbb{P}_{\mathrm{H}_0}\left(p \leq \alpha \mid \mathcal{F}_{\boldsymbol{Y}} = \mathcal{F}\right) = \alpha.$$

Therefore, in order to perform valid statistical test, we can employ p-values conditional on the hypothesis selection event. To compute a p-value that satisfies (4), we need to derive the sampling distribution of the test statistic

$$\boldsymbol{\eta}_j^\top \boldsymbol{Y} \mid \{\mathcal{F}_{\boldsymbol{Y}} = \mathcal{F}_{\boldsymbol{y}}\}. \tag{5}$$

4.2 Selective p-value

To perform statistical test based on the conditional sampling distribution in (5), we introduce an additional condition on the sufficient statistic of the nuisance parameter $\mathcal{Q}_{\boldsymbol{Y}}$, defined as

$$\mathcal{Q}_{\boldsymbol{Y}} = \left(I_n - \frac{\Sigma \boldsymbol{\eta} \boldsymbol{\eta}^\top}{\boldsymbol{\eta}^\top \Sigma \boldsymbol{\eta}}\right) \boldsymbol{Y}. \tag{6}$$

This additional conditioning on $\mathcal{Q}_{\boldsymbol{Y}}$ is a standard practice in the SI literature required for computational tractability[3]. Based on the additional conditioning on $\mathcal{Q}_{\boldsymbol{Y}}$, the following theorem tells that the conditional p-value that satisfies (4) can be derived by using a truncated normal distribution.

Theorem 4.1. *Consider a fixed design matrix X, a random response vector $\boldsymbol{Y} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$ and an observed response vector $\boldsymbol{y}$. Let $\mathcal{F}_{\boldsymbol{Y}}$ and $\mathcal{F}_{\boldsymbol{y}}$ be the obtained set of generated features, by applying an auto feature engineering algorithm in the form of* (2) *to $(X, \boldsymbol{Y})$ and $(X, \boldsymbol{y})$, respectively. Let $\boldsymbol{\eta} \in \mathbb{R}^n$ be a vector depending on $\mathcal{F}_{\boldsymbol{y}}$, and consider a test statistic in the form of $T(\boldsymbol{Y}) = \boldsymbol{\eta}^\top \boldsymbol{Y}$. Furthermore, define the nuisance parameter $\mathcal{Q}_{\boldsymbol{Y}}$ as in* (6)*.*

[3]The nuisance component $\mathcal{Q}_{\boldsymbol{Y}}$ corresponds to the component $\boldsymbol{z}$ in the seminal paper Lee et al. (2016) (see Sec. 5, Eq. (5.2), and Theorem 5.2) and is used in almost all the SI-related works that we cited.

Then, the conditional distribution

$$T(\boldsymbol{Y}) \mid \{\mathcal{F}_{\boldsymbol{Y}} = \mathcal{F}_{\boldsymbol{y}}, \mathcal{Q}_{\boldsymbol{Y}} = \mathcal{Q}_{\boldsymbol{y}}\}$$

is a truncated normal distribution $\mathrm{TN}(\boldsymbol{\eta}^\top \boldsymbol{\mu}, \boldsymbol{\eta}^\top \Sigma \boldsymbol{\eta}, \mathcal{Z})$ *with the mean $\boldsymbol{\eta}^\top \boldsymbol{\mu}$, the variance $\boldsymbol{\eta}^\top \Sigma \boldsymbol{\eta}$, and the truncation intervals $\mathcal{Z}$. The truncation intervals $\mathcal{Z}$ is defined as*

$$\mathcal{Z} = \left\{z \in \mathbb{R} \mid \mathcal{F}_{\boldsymbol{a}+\boldsymbol{b}z} = \mathcal{F}_{\boldsymbol{y}}\right\},\ \boldsymbol{a} = \mathcal{Q}_{\boldsymbol{y}},\ \boldsymbol{b} = \boldsymbol{\Sigma\eta}/\boldsymbol{\eta}^\top \Sigma \boldsymbol{\eta}. \tag{7}$$

The proof of Theorem 4.1 is deferred to Appendix A.1. By using the sampling distribution of the test statistic $T(\boldsymbol{Y})$ conditional on $\mathcal{F}_{\boldsymbol{Y}} = \mathcal{F}_{\boldsymbol{y}}$ and $\mathcal{Q}_{\boldsymbol{Y}} = \mathcal{Q}_{\boldsymbol{y}}$ in Theorem 4.1, we can define the selective p-value as

$$p_{\text{selective}} = \mathbb{P}_{\mathrm{H}_0}\left(|T(\boldsymbol{Y})| \geq |T(\boldsymbol{y})| \mid \mathcal{F}_{\boldsymbol{Y}} = \mathcal{F}_{\boldsymbol{y}}, \mathcal{Q}_{\boldsymbol{Y}} = \mathcal{Q}_{\boldsymbol{y}}\right). \tag{8}$$

Theorem 4.2. *The selective p-value defined in* (8) *satisfies the property in* (4)*, i.e.,*

$$\mathbb{P}_{\mathrm{H}_0}\left(p_{\text{selective}} \leq \alpha \mid \mathcal{F}_{\boldsymbol{Y}} = \mathcal{F}_{\boldsymbol{y}}\right) = \alpha,\ \forall \alpha \in (0,1).$$

Then, the selective p-value also satisfies the following property of a valid p-value:

$$\mathbb{P}_{\mathrm{H}_0}(p_{\text{selective}} \leq \alpha) = \alpha,\ \forall \alpha \in (0,1).$$

The proof of Theorem 4.2 is deferred to Appendix A.2. This theorem guarantees that the selective p-value is uniformly distributed under the null hypothesis H_0, and thus can be used to conduct the valid statistical inference in (3). Once the truncation intervals $\mathcal{Z}$ is identified, the selective p-value in (8) can be easily computed by Theorem 4.1. Thus, the remaining task is reduced to identifying the truncation intervals $\mathcal{Z}$.

5 COMPUTATIONS OF TRUNCATION INTERVALS

From the discussion in §4, it is suffice to identify the one-dimensional subset $\mathcal{Z}$ in (7) to conduct the statistical test. In this section, we propose a novel line search method to efficiently identify the $\mathcal{Z}$.

5.1 Overview of the Line Search

The difficulty in identifying the $\mathcal{Z}$ arises from the exponential number of possible paths that the AFE algorithm can traverse in the tree structure during feature generation. To address this challenge, we propose an efficient search method based on parametric-programming. The core idea is to compute the interval within which the nodes traversed by the AFE algorithm remain unchanged in the tree structure, ensuring that the same set of features is generated.

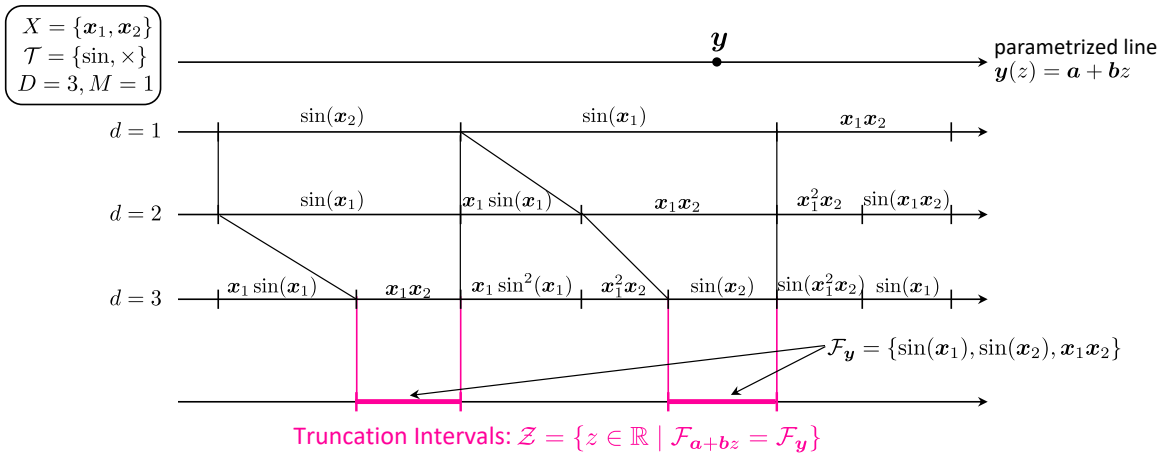


Figure 4: Schematic illustration of the proposed line search method for identifying the truncation intervals $\mathcal{Z}$. Initially, we compute the interval within which the generated features by the AFE algorithm remain unchanged. These intervals, depicted as the intervals on the line at $d = 3$ in the figure, are computed by sequentially identifying the intervals within which the features generated up to that point remain unchanged as the AFE algorithm progresses (i.e., the depth d increases). This computation process, illustrated in the figure by the narrowing intervals from $d = 1$ to $d = 3$, is detailed in §5.3. Finally, we identify the truncation intervals $\mathcal{Z}$ by taking the union of some intervals based on parametric-programming technique, as detailed in §5.2.

In the following, we present an overview of the proposed line search method to identify the truncation intervals $\mathcal{Z}$ using parametric-programming, leveraging this interval computation. Subsequently, we will elaborate on the details of this interval computation procedure. An overview of the proposed line search method is illustrated in Figure 4.

5.2 Parametric-Programming

To identify the truncation intervals $\mathcal{Z}$, we assume that we have a procedure to compute the interval $[L_z, U_z]$ for any $z \in \mathbb{R}$ which satisfies: for any $r \in [L_z, U_z]$, the outputs of the AFE algorithm from $(X, \boldsymbol{a} + \boldsymbol{b}r)$ and $(X, \boldsymbol{a} + \boldsymbol{b}z)$ are the same, i.e.,

$$\forall r \in [L_z, U_z],\ \mathcal{F}_{\boldsymbol{a}+\boldsymbol{b}r} = \mathcal{F}_{\boldsymbol{a}+\boldsymbol{b}z}.$$

Then, the truncation intervals $\mathcal{Z}$ can be obtained by the union of the intervals $[L_z, U_z]$ as

$$\mathcal{Z} = \bigcup_{z \in \mathbb{R} \mid \mathcal{F}_{\boldsymbol{a}+\boldsymbol{b}z} = \mathcal{F}_{\boldsymbol{y}}} [L_z, U_z]. \tag{9}$$

The procedure in (9) is commonly referred to as parametric-programming. We discuss the details of the procedure to compute the interval $[L_z, U_z]$ by considering the nodes traversed by the AFE algorithm.

5.3 Interval Computation

We subsequently elaborate on the computational procedure for obtaining the interval $[L_z, U_z]$ for any $z \in \mathbb{R}$, as referenced in §5.2. To compute $[L_z, U_z]$, it is sufficient to consider the conditions under which applying the AFE algorithm to $(X, \boldsymbol{a} + \boldsymbol{b}r)$ leads to traversing the exact same nodes as when applied to $(X, \boldsymbol{a} + \boldsymbol{b}z)$, irrespective of the response vector $\boldsymbol{a} + \boldsymbol{b}r$. Note that, to ensure determinism in the AFE algorithm, we adopt the convention of fixing the random seed to a constant value at the algorithm's initiation.

Upon reviewing Algorithms 1 and 2, it becomes apparent that their procedures depend on the response vector solely at two specific points within Algorithm 2: the AIC-based sorting in the first line and the AIC improvement assessment in the fifth line. Thus, we can deduce that the nodes traversed by the AFE algorithm remain unchanged if all AIC comparisons maintain the same order relations as when the algorithm is applied to $(X, \boldsymbol{a} + \boldsymbol{b}z)$. This condition can be readily translated into multiple inequalities, each taking the form $\{r \in \mathbb{R} \mid \mathrm{AIC}(V_i, \boldsymbol{a} + \boldsymbol{b}r) \leq \mathrm{AIC}(V_j, \boldsymbol{a} + \boldsymbol{b}r)\}$, where V_i and V_j represent the sets of features.

Finally, our objective is reduced to solving the inequality $\mathrm{AIC}(V_i, \boldsymbol{a} + \boldsymbol{b}r) \leq \mathrm{AIC}(V_j, \boldsymbol{a} + \boldsymbol{b}r)$ for r. Here, the AIC value of $(V, \boldsymbol{a} + \boldsymbol{b}r)$ can be expressed as:

$$\mathrm{AIC}(V, \boldsymbol{a} + \boldsymbol{b}r) = (\boldsymbol{a} + \boldsymbol{b}r)^\top \Lambda (\boldsymbol{a} + \boldsymbol{b}r) + 2|V|,$$

yielding a quadratic function of r, where $\Lambda = \Sigma^{-1} - \Sigma^{-1}V(V^\top\Sigma^{-1}V)^{-1}V^\top\Sigma^{-1}$. Therefore, the condition is represented by the intersection of multiple quadratic inequalities, which can be analytically solved to obtain a union of multiple intervals. For notational simplicity, we denote only the interval containing z as $[L_z, U_z]$.

6 NUMERICAL EXPERIMENTS

Methods for Comparison. In our experiments, we compare the proposed method (`proposed`) in terms of type I error rate and power with the following three methods:

- `w/o-pp`: An ablation study variant that omits the parametric-programming technique described in §5.2. This variant is implemented by replacing the truncation intervals $\mathcal{Z}$ in (9) with a single interval $[L_z, U_z]$ that encompasses the observed test statistic $T(\boldsymbol{y})$.
- `ds`: A data splitting method that divides the dataset into two parts: one for feature generation and the other for hypothesis testing on the generated features.
- `naive`: A classical z-test without any conditioning. The naive p-value is computed as $p_{\text{naive}} = \mathbb{P}_{\text{H}_0}(|T(\boldsymbol{Y})| \geq |T(\boldsymbol{y})|)$.

Experimental Setup. In all experiments, we set the significance level $\alpha = 0.05$. For the configuration of the AFE algorithm, we employed the transformation set $\mathcal{T} = \{\sin(\cdot), \exp(\min\{5, \cdot\}), \sqrt{|\cdot|}, \text{mul}\colon (a, b) \mapsto ab\}$ and set the maximum depth $D = 5$, the maximum number of nodes to be generated $N = 2$, the maximum number of nodes used for generation $M = 1$, and the tolerance parameter $\gamma = 2$. We considered two types of covariance matrices: $\Sigma = I_n \in \mathbb{R}^{n \times n}$ (independence) and $\Sigma = (0.5^{|i-j|})_{ij} \in \mathbb{R}^{n \times n}$ (correlation).

For the experiments to see the type I error rate, we changed the number of samples $n \in \{100, 150, 200\}$ and set the number of features m to 4. For each configuration, we generated 10,000 null datasets $(X, \boldsymbol{y})$, where $X_{ij} \sim \mathcal{N}(0, 1)$ for all $(i, j) \in [n] \times [m]$ and $\boldsymbol{y} \sim \mathcal{N}(\mathbf{0}, \Sigma)$. To investigate the power, we set $n = 150$ and $m = 4$ and generated dataset $(X, \boldsymbol{y})$, where $X_{ij} \sim \mathcal{N}(0, 1)$ for all $(i, j) \in [n] \times [m]$ and $\boldsymbol{y} = \Delta(\exp(\boldsymbol{x}_2) + \boldsymbol{x}_4 \exp(\boldsymbol{x}_2) + \sin(\exp(\boldsymbol{x}_2)) + \sqrt{|\boldsymbol{x}_4|} \exp(\boldsymbol{x}_2)) + \boldsymbol{\epsilon}$. The error term $\boldsymbol{\epsilon}$ followed a normal distribution $\mathcal{N}(\mathbf{0}, \Sigma)$, and we changed the true coefficient $\Delta \in \{0.2, 0.4, 0.6, 0.8\}$. For power evaluation, hypothesis testing was conducted only when the AFE algorithm generated at least one true feature (i.e., the features used to construct the response vector $\boldsymbol{y}$), resulting in a total of 10,000 tests. In addition, we provide an analysis of the computational time of the proposed method in Appendix B.1. See Appendix B.2 for the computer resources used in the experiments.

Results. The results of type I error rate are presented on the left side of Figure 5. The `proposed`, `w/o-pp`, and `ds` can successfully controlled the type I error rate below the significance level across all settings and for both covariance matrices, whereas the `naive` could not. Because the `naive` failed to control the type I error rate, we no longer consider its power. The results of power are presented on the right side of Figure 5, and we confirmed that the `proposed` has the highest power among those that successfully controlled the type I error rate, across all settings and for both covariance matrices. Comparing the `w/o-pp` and `ds`, the `w/o-pp` generally demonstrated higher power, although this varied depending on the specific settings and covariance matrices.

Robust Experiments. Additionally, we assess the robustness of the `proposed` in terms of type I error rate control by applying it to two additional scenarios: 1) when the variance is estimated from the same data used for testing, and 2) when the noise distribution is non-Gaussian. In the first scenario, we conducted hypothesis testing using the covariance matrix of $\hat{\sigma}^2 I_n$, where $\hat{\sigma}^2$ is estimated from the residuals of a multiple linear regression model fitted to the same data used for testing. This experiment mirrored the settings of the type I error rate experiment. In the second scenario, we considered five distribution families: `skewnorm`, `exponnorm`, `gennormsteep`, `gennormflat`, and `t`. These distribution families include the Gaussian distribution as a special case. Within each family, we evaluated the type I error rate by increasing the 1-Wasserstein distance from the Gaussian distribution, thereby increasing the non-Gaussianity of the noise distribution. See Appendix B.3 for more details. The results are presented in Figure 6, and we confirmed that the `proposed` can robustly control the type I error rate in both scenarios.

Real-World Data Experiments. We compare the `proposed` with the `ds` in terms of power and AIC, using seven real-world datasets from the UCI Machine Learning Repository (all licensed under the CC BY 4.0; see Appendix B.4 for details). A comparison with the `w/o-pp` is presented separately in Appendix B.5. The discussion and results presented there closely mirror those of the comparison with the `ds`. The power comparison aims to evaluate the superiority of each method as a statistical hypothesis test, under the implicit assumption that the features generated by the AFE algorithm are truly relevant. This assumption is reasonable because both the `proposed` and `ds` have been shown to control the type I error rate. In contrast, the AIC comparison assess the performance of the AFE algorithm within each method by comparing the AIC values of the features generated by the AFE algorithm. In this context, the AIC serves as

Table 1: Power and AIC on the real datasets when changing the number of samples. Each cell indicates two powers or AIC values: `proposed` vs. `ds`, which are separated by a slash. For power, the higher one is better and is boldfaced; for AIC value, conversely, lower one is better and is boldfaced. Our proposed method exhibits higher power and lower AIC value than the data splitting across all datasets and sample sizes $n \in \{100, 150, 200\}$. Note that the standard errors of the AIC values are all less than 0.005 and are therefore omitted from the table.

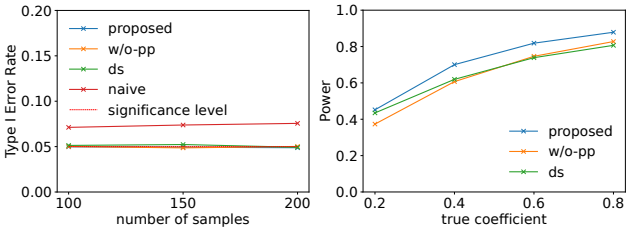
	n	Data1	Data2	Data3	Data4	Data5	Data6	Data7
Power	100	**.23**/.19	**.35**/.29	**.38**/.33	**.35**/.29	**.31**/.25	**.17**/.12	**.14**/.10
	150	**.28**/.24	**.42**/.37	**.46**/.41	**.41**/.36	**.40**/.34	**.21**/.16	**.19**/.13
	200	**.31**/.29	**.46**/.41	**.52**/.47	**.47**/.42	**.45**/.40	**.26**/.19	**.22**/.15
$\frac{\text{AIC}}{100}$	100	**1.01**/1.02	**0.92**/0.96	**0.91**/0.95	**0.97**/1.01	**1.00**/1.02	**1.08**/1.10	**1.09**/1.10
	150	**1.49**/1.50	**1.32**/1.37	**1.27**/1.33	**1.37**/1.42	**1.45**/1.48	**1.58**/1.59	**1.58**/1.60
	200	**1.96**/1.98	**1.72**/1.77	**1.63**/1.69	**1.77**/1.82	**1.90**/1.94	**2.07**/2.09	**2.07**/2.08

a measure of the goodness of fit of the final generated features to the response vector. This is consistent with the design of the AFE algorithm, as described in §3.2, which functions as a heuristic optimization procedure aimed at optimizing the AIC.

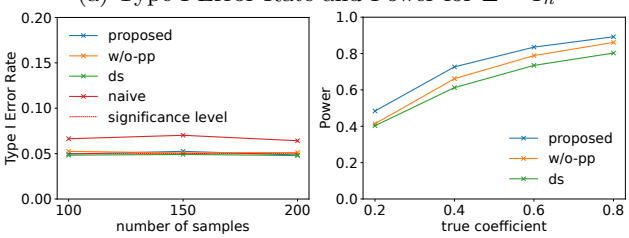
For these comparisons, we randomly generated 10,000 sub-sampled datasets with size of $n \in \{100, 150, 200\}$ from each of the original datasets. Both the `proposed` and `ds` were then applied to each sub-sampled dataset to asses their respective power and AIC values. The results, which are summarized in Table 1, demonstrate that the `proposed` exhibits higher (better) power and lower (better) AIC values compared to the `ds` across all datasets and sample sizes. This result further indicates that the `proposed` method not only provides a superior statistical hypothesis test with high statistical power, but also avoids the performance degradation of the AFE algorithm caused by the sample size reduction inherent in the `ds`. Moreover, both the power and AIC of the `proposed` improve as the sample size increases.

7 CONCLUSION

In this paper, we developed a statistical method to evaluate the reliability of features generated by an AFE algorithm. AFE can uncover important features that human intuition might miss, making it valuable for data-driven studies. Since AFE algorithms rely on heuristics due to their exponential complexity, assessing their reproducibility has been challenging so far. The proposed statistical test presented in this paper enables an unbiased evaluation of feature reproducibility based on SI framework. While this study focuses on a simple tree search-based heuristic AFE algorithm, we believe it represents an important step toward providing theoretical guarantees for AFE. In future work, we aim to extend this framework to offer reproducibility guarantees for various practical AFE algorithms.

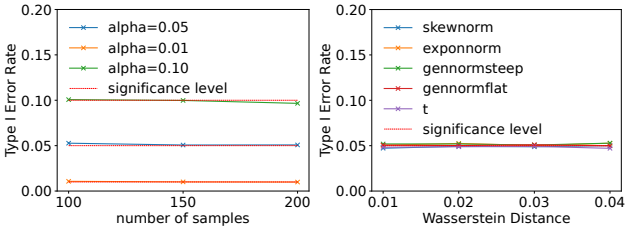


(a) Type I Error Rate and Power for $\Sigma = I_n$



(b) Type I Error Rate and Power for $\Sigma = (0.5^{|i-j|})_{ij}$

Figure 5: Type I Error Rate when changing the number of samples n (left side) and Power when changing the true coefficient (right side). All methods except for the naive method can successfully control the type I error rate. However, among these methods, our proposed method has the highest power across all settings and for both covariance matrices.



(a) Estimated Variance (b) Non-Gaussian Noise

Figure 6: Robustness of Type I Error Rate Control. Our proposed method can robustly control the type I error rate for the two scenarios: (a) when the variance is estimated from the same data used for testing and (b) when the noise distribution is non-Gaussian.

Acknowledgments

This work was partially supported by MEXT KAKENHI (20H00601), JST CREST (JPMJCR21D3, JPMJCR22N2), JST Moonshot R&D (JPMJMS2033-05), JST AIP Acceleration Research (JPMJCR21U2), NEDO (JPNP18002, JPNP20006) and RIKEN Center for Advanced Intelligence Project.

References

Brooks, T., Pope, D., and Marcolini, M. (1989). Airfoil Self-Noise. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5VW2C.

Chen, S. and Bien, J. (2020). Valid inference corrected for outlier removal. *Journal of Computational and Graphical Statistics*, 29(2):323–334.

Cortez, P., Cerdeira, A., Almeida, F., Matos, T., and Reis, J. (2009). Wine Quality. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C56S3T.

Das, D., Duy, V. N. L., Hanada, H., Tsuda, K., and Takeuchi, I. (2021). Fast and more powerful selective inference for sparse high-order interaction model. *arXiv preprint arXiv:2106.04929*.

Duy, V. N. L., Iwazaki, S., and Takeuchi, I. (2022). Quantifying statistical significance of neural network-based image segmentation by selective inference. *Advances in Neural Information Processing Systems*, 35:31627–31639.

Duy, V. N. L., Toda, H., Sugiyama, R., and Takeuchi, I. (2020). Computing valid p-value for optimal changepoint by selective inference using dynamic programming. In *Advances in Neural Information Processing Systems*.

Fithian, W., Taylor, J., Tibshirani, R., and Tibshirani, R. (2015). Selective sequential model selection. *arXiv preprint arXiv:1512.02565*.

Gao, L. L., Bien, J., and Witten, D. (2022). Selective inference for hierarchical clustering. *Journal of the American Statistical Association*, pages 1–11.

Horn, F., Pack, R., and Rieger, M. (2020). The autofeat python library for automated feature engineering and selection. In *Machine Learning and Knowledge Discovery in Databases: International Workshops of ECML PKDD 2019, Würzburg, Germany, September 16–20, 2019, Proceedings, Part I*, pages 111–120. Springer.

Hyun, S., G'sell, M., and Tibshirani, R. J. (2018). Exact post-selection inference for the generalized lasso path. *Electronic Journal of Statistics*, 12(1):1053–1097.

Jewell, S., Fearnhead, P., and Witten, D. (2022). Testing for a change in mean after changepoint detection. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 84(4):1082–1104.

Kanter, J. M. and Veeramachaneni, K. (2015). Deep feature synthesis: Towards automating data science endeavors. In *2015 IEEE international conference on data science and advanced analytics (DSAA)*, pages 1–10. IEEE.

Kaul, A., Maheshwary, S., and Pudi, V. (2017). Autolearn—automated feature generation and selection. In *2017 IEEE International Conference on data mining (ICDM)*, pages 217–226. IEEE.

Khan, P. W. and Byun, Y.-C. (2020). Genetic algorithm based optimized feature engineering and hybrid machine learning for effective energy consumption prediction. *Ieee Access*, 8:196274–196286.

Khurana, U., Turaga, D., Samulowitz, H., and Parthasrathy, S. (2016). Cognito: Automated feature engineering for supervised learning. In *2016 IEEE 16th international conference on data mining workshops (ICDMW)*, pages 1304–1307. IEEE.

Lam, H. T., Buesser, B., Min, H., Minh, T. N., Wistuba, M., Khurana, U., Bramble, G., Salonidis, T., Wang, D., and Samulowitz, H. (2021). Automated data science for relational data. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 2689–2692. IEEE.

Le Duy, V. N., Lin, H.-T., and Takeuchi, I. (2024). Cad-da: Controllable anomaly detection after domain adaptation by statistical inference. In *International Conference on Artificial Intelligence and Statistics*, pages 1828–1836. PMLR.

Lee, J. D., Sun, D. L., Sun, Y., and Taylor, J. E. (2016). Exact post-selection inference, with application to the lasso. *The Annals of Statistics*, 44(3):907–927.

Lee, J. D., Sun, Y., and Taylor, J. E. (2015). Evaluating the statistical significance of biclusters. *Advances in neural information processing systems*, 28.

Lee, J. D. and Taylor, J. E. (2014). Exact post model selection inference for marginal screening. *Advances in neural information processing systems*, 27.

Markovitch, S. and Rosenstein, D. (2002). Feature generation using general constructor functions. *Machine Learning*, 49:59–98.

Miwa, D., Le, D. V. N., and Takeuchi, I. (2023). Valid p-value for deep learning-driven salient region. In *Proceedings of the 11th International Conference on Learning Representation*.

Neufeld, A. C., Gao, L. L., and Witten, D. M. (2022). Tree-values: selective inference for regression trees. *Journal of Machine Learning Research*, 23(305):1–43.

Piramuthu, S. and Sikora, R. T. (2009). Iterative feature construction for improving inductive learning algorithms. *Expert Systems with Applications*, 36(2):3401–3406.

Rügamer, D. and Greven, S. (2020). Inference for l 2-boosting. *Statistics and computing*, 30(2):279–289.

Shi, K. and Saad, S. (2023). Automated feature engineering for automl using genetic algorithms. In *SECRYPT*, pages 450–459.

Shi, Q., Zhang, Y.-L., Li, L., Yang, X., Li, M., and Zhou, J. (2020). Safe: Scalable automatic feature engineering framework for industrial tasks. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 1645–1656. IEEE.

Shiraishi, T., Matsukawa, T., Nishino, S., and Takeuchi, I. (2024a). Statistical test for data analysis pipeline by selective inference. *arXiv preprint arXiv:2406.18902*.

Shiraishi, T., Miwa, D., Katsuoka, T., Duy, V. N. L., Taji, K., and Takeuchi, I. (2024b). Statistical test for attention maps in vision transformers. *International Conference on Machine Learning*.

Smith, M. G. and Bull, L. (2003). Feature construction and selection using genetic programming and a genetic algorithm. In *European conference on genetic programming*, pages 229–237. Springer.

Suzumura, S., Nakagawa, K., Umezu, Y., Tsuda, K., and Takeuchi, I. (2017). Selective inference for sparse high-order interaction models. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3338–3347. JMLR. org.

Tanizaki, K., Hashimoto, N., Inatsu, Y., Hontani, H., and Takeuchi, I. (2020). Computing valid p-values for image segmentation by selective inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9553–9562.

Taylor, J. and Tibshirani, R. J. (2015). Statistical learning and selective inference. *Proceedings of the National Academy of Sciences*, 112(25):7629–7634.

Tibshirani, R. J., Taylor, J., Lockhart, R., and Tibshirani, R. (2016). Exact post-selection inference for sequential regression procedures. *Journal of the American Statistical Association*, 111(514):600–620.

Tsanas, A. and Xifara, A. (2012). Energy Efficiency. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C51307.

Tsukurimichi, T., Inatsu, Y., Duy, V. N. L., and Takeuchi, I. (2021). Conditional selective inference for robust regression and outlier detection using piecewise-linear homotopy continuation. *arXiv preprint arXiv:2104.10840*.

Yamada, M., Umezu, Y., Fukumizu, K., and Takeuchi, I. (2018). Post selection inference with kernels. In *International conference on artificial intelligence and statistics*, pages 152–160. PMLR.

Yang, F., Barber, R. F., Jain, P., and Lafferty, J. (2016). Selective inference for group-sparse linear models. In *Advances in Neural Information Processing Systems*, pages 2469–2477.

Yeh, I.-C. (1998). Concrete Compressive Strength. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5PK67.

Yeh, I.-C. (2018). Real Estate Valuation. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5J30W.

Zhang, T., Zhang, Z., Luo, H., Liu, F., Cao, W., and Li, J. (2022). Openfe: Automated feature generation beyond expert-level performance. *arXiv preprint arXiv:2211.12507*.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. Yes, we provide a clear description in §2, §3 and §4.
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. Yes, we provide an analysis of the computation time in Appendix B.1.

(c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries.
Yes, we provide the source code for the reproducibility of the experiments in the supplemental file, keeping in mind the anonymity of the authors.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results.
   Yes, we state the full set of assumptions in all of our theorem statements (i.e., Theorem 4.1 and Theorem 4.2).
   (b) Complete proofs of all theoretical results.
   Yes, we provide the complete proofs of all our theorem statements in Appendix A.
   (c) Clear explanations of any assumptions.
   Yes, we state the assumptions clearly in all of our theorem statements.

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL).
   Yes, we provide the source code for the reproducibility of the experiments in the supplemental file.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen).
   Not Applicable, we do not include any training process of the model.
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times).
   Yes, we always provide the number of iterations and the evaluation metric in all of our experiments (§6 and Appendix B.1).
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider).
   Yes, we provide the computing infrastructure used in the experiments in Appendix B.2.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets.
   Yes, we cite the creator of the datasets we used in Appendix B.4.
   (b) The license information of the assets, if applicable.
   Yes, we clarify that all datasets we used are licensed under the CC BY 4.0 license in Appendix B.4.
   (c) New assets either in the supplemental material or as a URL, if applicable.
   Not Applicable.
   (d) Information about consent from data providers/curators.
   Not Applicable.
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content.
   Not Applicable.

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots.
   Not Applicable.
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable.
   Not Applicable.
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation.
   Not Applicable.

A Proofs

A.1 Proof of Theorem 4.1

According to the conditioning on $\mathcal{Q}_{\boldsymbol{Y}} = \mathcal{Q}_{\boldsymbol{y}}$, we have

$$\mathcal{Q}_{\boldsymbol{Y}} = \mathcal{Q}_{\boldsymbol{y}} \Leftrightarrow \left(I_n - \frac{\Sigma \boldsymbol{\eta} \boldsymbol{\eta}^\top}{\boldsymbol{\eta}^\top \Sigma \boldsymbol{\eta}} \right) \boldsymbol{Y} = \mathcal{Q}_{\boldsymbol{y}} \Leftrightarrow \boldsymbol{Y} = \boldsymbol{a} + \boldsymbol{b} z,$$

where $z = T(\boldsymbol{Y}) = \boldsymbol{\eta}^\top \boldsymbol{Y} \in \mathbb{R}$. Then, we have

$$\begin{aligned}
&\{\boldsymbol{Y} \in \mathbb{R}^n \mid \mathcal{F}_{\boldsymbol{Y}} = \mathcal{F}_{\boldsymbol{y}}, \mathcal{Q}_{\boldsymbol{Y}} = \mathcal{Q}_{\boldsymbol{y}}\} \\
=&\{\boldsymbol{Y} \in \mathbb{R}^n \mid \mathcal{F}_{\boldsymbol{Y}} = \mathcal{F}_{\boldsymbol{y}}, \boldsymbol{Y} = \boldsymbol{a} + \boldsymbol{b} z, z \in \mathbb{R}\} \\
=&\{\boldsymbol{a} + \boldsymbol{b} z \in \mathbb{R}^n \mid \mathcal{F}_{\boldsymbol{a}+\boldsymbol{b}z} = \mathcal{F}_{\boldsymbol{y}}, z \in \mathbb{R}\} \\
=&\{\boldsymbol{a} + \boldsymbol{b} z \in \mathbb{R}^n \mid z \in \mathcal{Z}\}.
\end{aligned}$$

Therefore, we obtain

$$T(\boldsymbol{Y}) \mid \{\mathcal{F}_{\boldsymbol{Y}} = \mathcal{F}_{\boldsymbol{y}}, \mathcal{Q}_{\boldsymbol{Y}} = \mathcal{Q}_{\boldsymbol{y}}\} \sim \mathrm{TN}(\boldsymbol{\eta}^\top \boldsymbol{\mu}, \boldsymbol{\eta}^\top \Sigma \boldsymbol{\eta}, \mathcal{Z}).$$

A.2 Proof of Theorem 4.2

By probability integral transformation, under the null hypothesis, we have

$$p_{\mathrm{selective}} \mid \{\mathcal{F}_{\boldsymbol{Y}} = \mathcal{F}_{\boldsymbol{y}}, \mathcal{Q}_{\boldsymbol{Y}} = \mathcal{Q}_{\boldsymbol{y}}\} \sim \mathrm{Unif}(0, 1),$$

which leads to

$$\mathbb{P}_{\mathrm{H}_0} \left(p_{\mathrm{selective}} \leq \alpha \mid \mathcal{F}_{\boldsymbol{Y}} = \mathcal{F}_{\boldsymbol{y}}, \mathcal{Q}_{\boldsymbol{Y}} = \mathcal{Q}_{\boldsymbol{y}} \right) = \alpha, \ \forall \alpha \in (0, 1).$$

For any $\alpha \in (0, 1)$, by marginalizing over all the values of the nuisance parameters, we obtain

$$\begin{aligned}
&\mathbb{P}_{\mathrm{H}_0} \left(p_{\mathrm{selective}} \leq \alpha \mid \mathcal{F}_{\boldsymbol{Y}} = \mathcal{F}_{\boldsymbol{y}}, \right) \\
=&\int_{\mathbb{R}^n} \mathbb{P}_{\mathrm{H}_0} \left(p_{\mathrm{selective}} \leq \alpha \mid \mathcal{F}_{\boldsymbol{Y}} = \mathcal{F}_{\boldsymbol{y}}, \mathcal{Q}_{\boldsymbol{Y}} = \mathcal{Q}_{\boldsymbol{y}} \right) \\
&\qquad \mathbb{P}_{\mathrm{H}_0} \left(\mathcal{Q}_{\boldsymbol{Y}} = \mathcal{Q}_{\boldsymbol{y}} \mid \mathcal{F}_{\boldsymbol{Y}} = \mathcal{F}_{\boldsymbol{y}}, \right) d\mathcal{Q}_{\boldsymbol{y}} \\
=&\alpha \int_{\mathbb{R}^n} \mathbb{P}_{\mathrm{H}_0} \left(\mathcal{Q}_{\boldsymbol{Y}} = \mathcal{Q}_{\boldsymbol{y}} \mid \mathcal{F}_{\boldsymbol{Y}} = \mathcal{F}_{\boldsymbol{y}}, \right) d\mathcal{Q}_{\boldsymbol{y}} = \alpha.
\end{aligned}$$

Therefore, we also obtain

$$\begin{aligned}
\mathbb{P}_{\mathrm{H}_0}(p_{\mathrm{selective}} \leq \alpha) =& \sum_{\mathcal{F}_{\boldsymbol{y}} \in \mathcal{F}(X)} \mathbb{P}_{\mathrm{H}_0}(\mathcal{F}_{\boldsymbol{y}}) \mathbb{P}_{\mathrm{H}_0} \left(p_{\mathrm{selective}} \leq \alpha \mid \mathcal{F}_{\boldsymbol{Y}} = \mathcal{F}_{\boldsymbol{y}} \right) \\
=&\alpha \sum_{\mathcal{F}_{\boldsymbol{y}} \in \mathcal{F}(X)} \mathbb{P}_{\mathrm{H}_0}(\mathcal{F}_{\boldsymbol{y}}) = \alpha.
\end{aligned}$$

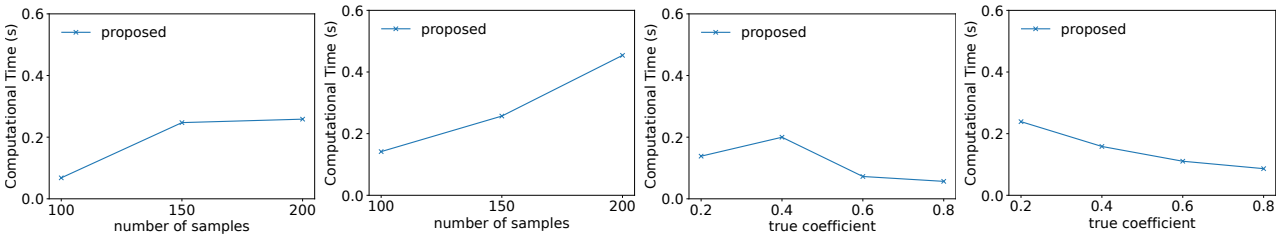
B Details of the Experiments

B.1 Computational Time of the Proposed Method

We analyzed the computational time of our proposed method in the run of our experiments on synthetic datasets in §6. The results are shown in Figure 7.

B.2 Computer Resources

All numerical experiments were conducted on a computer with a 96-core 3.60GHz CPU and 512GB of memory.



(a) $\Sigma = I_n$ (b) $\Sigma = (0.5^{|i-j|})_{ij}$ (c) $\Sigma = I_n$ (d) $\Sigma = (0.5^{|i-j|})_{ij}$

Figure 7: Computational Time when evaluating type I error rate (left two panels) and Power (right two panels). The results show that the computation time increases moderately with the number of samples. Additionally, the computational time tends to decrease as the true coefficient increases. This may be attributed to the increased clarity of the hypothesis testing results with larger true coefficients.

B.3 Experimental Setup for the Robust Experiments

In the robustness experiments, we assess the robustness of the proposed method in terms of type I error rate control by applying it to two additional scenarios: 1) when the variance is estimated from the same data used for testing, and 2) when the noise distribution is non-Gaussian.

In the first scenario, we change the number of samples $n \in \{100, 150, 200\}$ and set the number of features m to 4. For each configuration, we generated 10,000 null datasets $(X, \boldsymbol{y})$, where $X_{ij} \sim \mathcal{N}(0, 1)$ for all $(i, j) \in [n] \times [m]$ and $\boldsymbol{y} \sim \mathcal{N}(\mathbf{0}, I_n)$. We conducted hypothesis testing on these testing using the covariance matrix of $\hat{\sigma}^2 I_n$, where $\hat{\sigma}^2$ is estimated from the residuals of a multiple linear regression model fitted to the same data used for testing, i.e.,

$$\hat{\sigma}^2 = \frac{1}{n-m} \|\boldsymbol{y} - X(X^\top X)^{-1} X^\top \boldsymbol{y}\|_2^2.$$

In the second scenario, we set the number of samples n to 150 and the number of features m to 4. As non-Gaussian distributions, we consider the following five distribution families:

- `skewnorm`: Skew normal distribution family.
- `exponnorm`: Exponentially modified normal distribution family.
- `gennormsteep`: Generalized normal distribution family (limit the shape parameter β to be steeper than the normal distribution, i.e., $\beta < 2$).
- `gennormflat`: Generalized normal distribution family (limit the shape parameter β to be flatter than the normal distribution, i.e., $\beta > 2$).
- `t`: Student's t distribution family.

These distribution families include the Gaussian distribution as a special case and are standardized in the experiments. Within each distribution family, we initially obtained a distribution such that the 1-Wasserstein distance from the standard Gaussian distribution was $l \in \{0.01, 0.02, 0.03, 0.04\}$. For each obtained distribution, we generated 10,000 null datasets $(X, \boldsymbol{y})$, where $X_{ij} \sim \mathcal{N}(0, 1)$ for all $(i, j) \in [n] \times [m]$ and y_i was generated from the distribution for all $i \in [n]$.

B.4 Details of the Real-World Datasets

We used the following seven real-world datasets from the UCI Machine Learning Repository. All datasets are licensed under the CC BY 4.0 license.

- Airfoil Self-Noise (Brooks et al., 1989) for Data1
- Concrete Compressive Strength (Yeh, 1998) for Data2

- Energy Efficiency (Tsanas and Xifara, 2012) for Data3 (heating load) and Data4 (cooling load)
- Real Estate Valuation (Yeh, 2018) for Data5
- Wine Quality (Cortez et al., 2009) for Data6 (red wine) and Data7 (white wine)

B.5 Comparison of Proposed Method and Ablation Study

We compared the proposed method with the ablation study (`w/o-pp`) using the same seven real-world datasets as in §6. Note that the difference between these two methods is only in the part that compute the p-value, and the generated feature sets by the AFE algorithm are identical, so there is no need to compare the AIC values. The results of power comparison are summarized in Table 2, and demonstrate that the proposed method (`proposed`) exhibits higher power compared to the ablation study variant (`w/o-pp`) across all datasets and sample sizes.

Table 2: Power on the real datasets when changing the number of samples. Each cell indicates two powers: `proposed` vs. `w/o-pp`, which are separated by a slash. The higher one is better and is boldfaced. Our proposed method has higher power than the ablation study variant across all datasets and sample sizes $n \in \{100, 150, 200\}$.

	n	Data1	Data2	Data3	Data4	Data5	Data6	Data7
Power	100	**0.23**/0.21	**0.35**/0.31	**0.38**/0.34	**0.35**/0.31	**0.31**/0.27	**0.17**/0.16	**0.14**/0.13
	150	**0.28**/0.26	**0.42**/0.37	**0.46**/0.41	**0.41**/0.38	**0.40**/0.35	**0.21**/0.20	**0.19**/0.18
	200	**0.31**/0.29	**0.46**/0.42	**0.52**/0.48	**0.47**/0.43	**0.45**/0.39	**0.26**/0.24	**0.22**/0.21