
Stability-Weighted Ensemble Feature Importance for Financial Applications

Hamid Arian^a · Alireza Mousavizade^b · Luis Seco^c

^b York University, 4700 Keele St, Toronto, Ontario, M3J 1P3, Canada; E-mail: harian@yorku.ca

^a Sharif University of Technology, GSME, Azadi Street, Iran; E-mail: mousavizade@risklab.ai

^d University of Toronto, 40 St George St, Toronto, Ontario, M5S 2E4, Canada; E-mail: luis.seco@utoronto.ca

Received: date / Accepted: date

Abstract Robust feature importance is crucial in financial applications to enhance model interpretability and performance. Traditional methods often falter when applied to noisy, non-stationary financial data, leading to inconsistent and unreliable feature importance scores. This paper introduces the stability-weighted ensemble Feature Importance (SWEFI) method, which combines stability selection with ensemble learning to address these challenges. SWEFI aims to deliver stable and interpretable feature importance scores, thereby improving model reliability in high-stakes financial environments. The methodology integrates multiple feature importance techniques and learning models, assessing feature relevance through a comprehensive and systematic approach. Extensive experiments using synthetic and real-world financial datasets demonstrate SWEFI's superior stability and accuracy compared to existing methods. This research underscores the importance of stable feature importance measures, facilitating better decision-making and enhancing trust in machine learning models.

Keywords Machine Learning · Feature Importance · Model Uncertainty · Model Stability · Feature Importance


1 Introduction

Feature importance techniques have become crucial in the era of big data and complex models, particularly in financial applications. These techniques aid in identifying the most relevant features within a dataset, enhancing model interpretability and performance (Dhal and Azad [2022]). With advancements in machine learning, ensemble methods have gained popularity due to their ability to improve model accuracy and robustness by combining the predictions of multiple models. Ensemble approaches have also extended to feature importance, where combining feature importance scores from various models can provide a more robust understanding of feature relevance. Stability selection focuses on the consistency of feature importance scores across different data subsets and model variations, addressing the issue of sensitivity to data perturbations and model choice (Khaire and Dhanalakshmi [2022]).

Despite the availability of various feature importance techniques, several pressing issues and financial gaps necessitate further research into feature importance. Data is often noisy in

Supporting Material:

 Code Library: github.com/RiskLabAI

 Reproducible Results: github.com/RiskLabAI/Notebooks.py/tree/main/features/SWEFI

financial applications and can exhibit non-stationary behavior, making it challenging to derive consistent and meaningful feature importance scores using conventional methods (Schmitt et al. [2013]). The financial sector requires methods that robustly handle such data characteristics and provide stable feature importance rankings. Additionally, while ensemble methods improve model performance by leveraging multiple models, integrating stability selection into these methods remains underexplored. There is a gap in research addressing how to systematically combine the stability of feature importance scores with the robustness of ensemble methods.

The American Statistical Association cautions against misinterpreting p-values in feature importance. p-values alone do not imply importance; holistic evaluation of effect size and context is crucial for reliable inference (Association et al. [2016]). The misuse of p-values is so widespread that the American Statistical Association discourages their application going forward (Wasserstein et al. [2019]). Machine learning can play a valuable role in addressing ASA concerns. For example, Breiman’s Random Forests introduced a robust way to compute feature importance by measuring the decrease in impurity or the average gain of the criteria used to split nodes (Breiman [2001]). This tree-based feature importance concept, introduced by Breiman, is known as mean decrease impurity. Another case is permutation importance, which assesses feature importance by measuring how much a model’s performance drops when feature values are randomly shuffled. Introduced by (Breiman [2001]), this is a versatile, model-agnostic method applicable across different machine learning algorithms for feature selection and understanding model behavior based on data behavior. Gradient Boosting Machines (GBMs), such as those proposed by Friedman, compute feature importance similarly to Random Forests but aggregate the importance across multiple boosting iterations (Friedman [2001]).

Two prominent explainable methods for assessing feature importance are SHAP values and LIME. SHAP (SHapley Additive exPlanations) values are based on cooperative game theory. They provide a unified measure of feature importance by considering the contribution of each feature across all possible combinations, ensuring a fair distribution of importance (Štrumbelj and Kononenko [2014]). LIME (Local Interpretable Model-agnostic Explanations) is used for feature selection by training a local surrogate model around a prediction. It assesses feature importance by perturbing features and observing changes in predictions, ranking them by the impact on model outcome (Ribeiro et al. [2016]).

De Prado [2018] outlines three methods for obtaining importance scores. First, the Mean Decrease Impurity (MDI) score, derived from tree-based classifiers, uses in-sample (IS) performance to estimate feature importance. Second, the Mean Decrease Accuracy (MDA) method, applied to any classifier, relies on out-of-sample (OOS) performance for feature importance estimation. Lastly, the Single Feature Importance (SFI) method addresses the substitution effects seen in MDI and MDA, where highly correlated features might be deemed redundant. SFI, as an OOS feature importance estimator, evaluates each feature separately, thus avoiding these substitution effects.

Current methods for determining feature importance, including MDI, MDA, and SFI, suffer from significant instability. This instability undermines the reliability and interpretability of these methods, which are crucial for high-stakes applications like finance. Traditional feature importance techniques, while intuitive and straightforward, often fall short when applied to complex ensemble models, leading to fluctuating importance scores that can misguide decision-making processes. This issue is further compounded in high-dimensional financial datasets characterized by intricate patterns, temporal dynamics, and non-linear correlations, which pose significant challenges in extracting reliable insights and making accurate predictions. Conventional statistical techniques often fail to address these complexities, necessitating advanced machine-learning approaches. A crucial challenge in this domain is feature selection. Identifying and selecting relevant features from an array of options is a complex task, compounded by the risk of including irrelevant features, leading to overfitting or omitting valuable

features, thereby diminishing model efficacy. The temporal dynamics of financial time series data introduce an additional layer of complexity, making it essential to use bootstrapping methods to capture the data's intrinsic variability and uncertainty (Subbiah and Chinnappan [2021]). The lack of stable feature importance measures poses a critical challenge in machine learning: how can we ensure that the identified important features are both reliable and robust across different data subsets and model configurations?

The primary goal of this research is to develop and evaluate the Stability Weighted Ensemble Feature Importance (SWEFI) method to enhance the reliability and interpretability of feature importance in machine learning models. The study aims to address the instability and inconsistency of existing feature-importance methods, which undermines their reliability in high-stakes applications such as finance. To achieve this, we are focusing on several key objectives. Firstly, we are developing SWEFI by creating a novel approach that integrates stability selection with ensemble learning. This integration is designed to produce feature importance scores that are both stable and interpretable, addressing the common issues of instability and inconsistency in traditional methods. Next, we will evaluate the performance of SWEFI through comprehensive experiments. These experiments will compare SWEFI's accuracy, stability, and robustness against existing feature importance methods. We will use a variety of datasets, including noisy and complex financial data, to ensure that our evaluation is thorough and representative of real-world scenarios. Furthermore, we aim to analyze the impact of improved stability on model interpretability and trustworthiness. By investigating how stable feature importance scores affect these critical aspects, we hope to demonstrate the essential role of stability in reliable decision-making. Finally, we seek to address existing gaps in the literature by exploring the integration of stability selection with ensemble learning. This involves examining the trade-offs between stability and accuracy and providing insights that could guide future research and practical applications. By achieving these objectives, the study aims to advance machine learning by offering a robust, reliable, and interpretable feature importance approach, facilitating better decision-making and increased trust in machine learning models.

This paper introduces the SWEFI method. The primary contributions of this study are multifaceted. Firstly, the SWEFI method represents a novel approach by combining stability selection techniques with ensemble learning methods. This integration addresses the instability and inconsistency issues commonly encountered with traditional feature importance methods, particularly in complex and noisy data environments. By leveraging the strengths of both stability selection and ensemble methods, SWEFI produces feature importance scores that are more stable and interpretable. This stability ensures that the feature importance rankings are reliable and consistent across different subsets of data and model configurations. The paper conducts extensive experiments to evaluate SWEFI's performance, comparing its stability against existing methods. The evaluation includes a variety of datasets, demonstrating SWEFI's practical applicability and effectiveness in real-world scenarios. Additionally, the study investigates how the improved stability of feature importance scores impacts model interpretability and trustworthiness. By providing stable and reliable importance scores, SWEFI enhances the transparency and trustworthiness of machine learning models, facilitating better decision-making processes in applications. Finally, the research fills significant gaps in the existing literature by exploring the trade-offs between stability and accuracy in feature importance measures and demonstrating how stability selection can be effectively integrated with ensemble learning methods.

2 Methodology

Our method results from an integrated approach, merging the robustness of ensemble learning techniques with the rigor of stability selection methodologies. The primary objective of SWEFI is to provide a consistent and reliable measure of feature significance. It critically analyzes

outcomes across diverse datasets and model architectures to ensure a robust understanding of features' performance.

This systematic approach underscores the resilience of the selected features, confirming their ability to remain consistent, even in the face of data variations often encountered in financial scenarios. As financial data is susceptible to minor fluctuations, which can have significant implications, the SWEFI framework's emphasis on stability and consistency provides an essential layer of reliability.

2.1 An Overview of SWEFI

Algorithm 1 Stability Weighted Ensemble Feature Importance (SWEFI)

Require: $\mathcal{D} = (X, y)$ where $X : \mathcal{F} = \{f_i\}_{i=1}^m, \forall 1 \leq i \leq m : f_i \in \mathbb{R}^n$ and $y \in \mathbb{R}^n$, Input Dataset

Require: $\mathcal{FI}_{1 \leq i \leq k} \subset \{\text{MDA}, \text{MDI}, \dots\}$, Feature Importance Methods

Require: $\mathcal{M}_{1 \leq i \leq k} \subset \{\text{SVM}, \text{LDA}, \dots\}$, Learning Models

Require: resampling $\in \{\text{cv}, \text{bootstrap}, \dots\}$, Resampling method.

Require: k , The top $k\%$ of the features.

Require: τ , Threshold for feature selection using cumulative sum.

Ensure: $\mathcal{F}_s \subset \mathcal{F}_{1 \leq i \leq m}$, Selected features

```

1:  $X^* \leftarrow \text{preprocess}(X)$ 
2:  $\mathcal{M}_i^* \leftarrow \text{hpo}(\mathcal{M}_i)$ 
3:  $\mathcal{S}_{1 \leq i \leq r} \leftarrow \text{resampling}(\mathcal{D})$ 
4: for  $S_i \in \mathcal{S}_{1 \leq i \leq r}$  do
5:   for  $(\mathcal{M}_j, \mathcal{FI}_j) \in (\mathcal{M}, \mathcal{FI})$  do
6:     for  $f \in \mathcal{F}$  do
7:        $I_{i,j}(f) \leftarrow \mathcal{FI}_j(S_i, \mathcal{M}_j)$ 
8:     end for
9:   end for
10: end for
11:  $T \leftarrow k \times |\mathcal{F}|/100$ 
12: for  $f \in \mathcal{F}$  do
13:    $SS(f) = \frac{1}{|\mathcal{M}| |\mathcal{S}|} \sum \mathbb{I}(\text{rank}(f, I_{i,j}) \leq T)$ 
14: end for
15: normalize  $SS$  to 1.
16: for  $f \in \mathcal{F}$  do
17:    $I : EI(f) \leftarrow \frac{\sum_{i,j} SS(f) \cdot I_{i,j}(f)}{\sum_{i,j} SS(f)}$ 
18: end for
19:  $\mathcal{F}_s \leftarrow \mathcal{F}[\text{select}(I, th)]$  (select Algorithm 2)
   return  $\mathcal{F}_s$ , Return selected features.
```

As outlined in Algorithm 1, the overall process of SWEFI involves several steps. Given a dataset $\mathcal{D} = (X, y)$, where X consists of features $\mathcal{F} = \{f_i\}_{i=1}^m$ with $f_i \in \mathbb{R}^n$ and $y \in \mathbb{R}^n$, the SWEFI algorithm aims to identify a subset of important features $\mathcal{F}_s \subset \mathcal{F}$. The algorithm utilizes multiple feature importance methods $\mathcal{FI}_{1 \leq i \leq k}$ (e.g., MDA, MDI) and various learning models $\mathcal{M}_{1 \leq i \leq k}$ (e.g., Support Vector Machine, Linear Discriminant Analysis). The method also incorporates a resampling technique (e.g., cross-validation (CV), bootstrap), and requires parameters k (the top percentage used in stability score computation) and τ (the threshold for feature selection based on cumulative sum).

The SWEFI algorithm begins by optionally pre-processing the dataset X to potentially reduce its dimensions. Following this, hyper-parameter optimization is performed on each learning model \mathcal{M}_i . The dataset \mathcal{D} is then resampled r times using the specified resampling method, resulting in r subsets $\mathcal{S}_{1 \leq i \leq r}$. For each subset S_i , feature importance is calculated using various combinations of learning models \mathcal{M}_j and feature importance methods \mathcal{FI}_j . Specifically, for each feature f , an importance score $I_{i,j}(f)$ is computed. Next, a stability score $SS(f)$ is determined for each feature based on the frequency with which it ranks within the top $k\%$ across all these combinations. This score reflects the consistency of a feature's

importance. The ensemble importance $EI(f)$ for each feature is then calculated by weighting its individual importance scores $I_{i,j}(f)$ with the stability score $SS(f)$. Finally, features are selected based on their ensemble importance scores using a specified threshold τ , resulting in the subset of selected features \mathcal{F}_s . This process ensures that the selected features are not only important according to multiple methods and models but also stable across subsets.

2.2 Feature Pre-processing and Dimension Reduction

To optimize the feature set and enhance the efficiency of subsequent analyses, it is often necessary to apply pre-processing techniques. One key aspect of feature pre-processing is dimension reduction, which aims to simplify the complexity and computational burden associated with high-dimensional data. By employing dimension reduction techniques, such as Principal Component Analysis (PCA) and manifold learning methods, we can extract the most informative aspects of the data. These algorithms transform the original feature space into a lower-dimensional representation, retaining essential information while mitigating issues like the curse of dimensionality and discarding redundant or less informative features.

2.3 Diversified Model Selection

In our study, we aimed to enhance pattern spotting and improve our methods by employing a diverse set of machine learning models. Our approach included Linear Models, which excel at identifying straightforward connections in data and perform optimally when the data points follow a clear linear trend. Additionally, we utilized Tree-based Models, such as Decision Trees and Random Forests, which are adept at handling varied types of data and identifying complex relationships between variables. These models are particularly useful when the data exhibits intricate patterns and interactions. Support Vector Machines were also part of our methodology. They are effective in data-rich environments due to their ability to filter out noise and focus on essential patterns. This makes them especially valuable when dealing with large datasets. Finally, we incorporated Neural Networks, particularly those leveraging Deep Learning techniques. These networks are proficient in analyzing layered data structures and uncovering hidden patterns within extensive datasets. By integrating these diverse models, we did not rely solely on the strengths of any single approach. Instead, this comprehensive strategy allowed us to mitigate the limitations of individual models and create a more robust and versatile system. Combining insights from different models provided a clearer picture of the data and facilitated thorough cross-checking and accurate error measurement. This multifaceted approach is particularly advantageous in complex domains such as finance, ensuring our methods remain adaptable and robust.

2.4 Feature Importance Calculation

In this step of the SWEFI algorithm, the importance of each feature is calculated for each combination of the subsets of datasets and learning models - feature importance methods, using various feature importance methods. This approach ensures that feature importance is evaluated across multiple contexts, thereby increasing the robustness of the final selection. Specifically, for each subset $S_i \in \mathcal{S}_{1 \leq i \leq r}$, where S_i represents one of the r resampled versions of the original dataset \mathcal{D} , the algorithm iterates over each pair of learning model \mathcal{M}_j and feature importance method \mathcal{FI}_j . The learning model \mathcal{M}_j is used to fit the subset S_i , and the feature importance method \mathcal{FI}_j evaluates the importance of features based on the fitted model. These importance of features are notated as follows:

$$I_{i,j}(f) = \mathcal{FI}_j(S_i, \mathcal{M}_j) \quad (1)$$

For each feature $f \in \mathcal{F}$, the algorithm computes the importance score $I_{i,j}(f)$ (Eqn. (1)) using the feature importance method \mathcal{FI}_j applied to the model \mathcal{M}_j trained on the subset S_i . This

involves first training the learning model \mathcal{M}_j with the subset S_i , which requires fitting the model to the data in S_i to learn the relationships between features and the target variable y . Subsequently, the feature importance method \mathcal{FI}_j is applied to the trained model to assess the significance of each feature f in making accurate predictions. The calculated importance score $I_{i,j}(f)$ (Eqn. (1)) for feature f is then stored, representing the importance of feature f according to the combination of the subset S_i , learning model \mathcal{M}_j , and feature importance method \mathcal{FI}_j .

It is important to note that, in order to make the importance scores comparable, for each $\mathcal{FI}_j(S_i, \mathcal{M}_j)$ run, the calculated importance scores $I_{i,j}(f)$ (Eqn. (1)) must be normalized to the $[0,1]$ interval using Min-Max normalization [Patro and Sahu \[2015\]](#).

By the end of this step, the algorithm will have a comprehensive set of importance scores $I_{i,j}(f)$ (Eqn. (1)) for each feature f , derived from multiple models and feature importance methods across different subsets of the dataset. These scores form the basis for calculating stability scores and ensemble importance in the subsequent steps.

In addition to model-specific feature importance methods, model-free feature importance methods, utilized for univariate analysis, such as Mutual Information, ANOVA F-scores, and various correlation metrics, can also be used. Similarly to model-specific methods, these model-free methods can be applied to different S_i subsets to obtain model-free feature importance scores ([Vergara and Estévez \[2014\]](#), [St et al. \[1989\]](#) and [Hall \[1999\]](#)).

2.5 Stability Score for Feature Importance

Understanding how a model works and ensuring its accuracy largely depends on pinpointing key data points or features, which can be thought of as the main ingredients in a recipe. Just as some ingredients are crucial to a dish, some features are essential for a model's success. To identify these essential features, we use the "stability score". This score evaluates how consistently a feature plays a crucial role across different datasets and learning models.

The stability score calculation is a critical component of the SWEFI algorithm. This process assesses the consistency of each feature's importance across various subsets and learning models. Specifically, for each feature $f \in \mathcal{F}$, we calculate its stability score as follows: the dataset \mathcal{D} is resampled r times, generating subsets $S_{1 \leq i \leq r}$. For each subset S_i , feature importance scores are computed using multiple combinations of learning models \mathcal{M}_j and feature importance methods \mathcal{FI}_j . This produces a collection of importance scores $I_{i,j}(f)$ (Eqn. (1)) for each feature f .

In each combination of subset S_i and learning model \mathcal{M}_j - feature importance method \mathcal{FI}_j , features are ranked based on their importance scores $I_{i,j}(f)$ (Eqn. (1)). The rank of feature f in this context is denoted as $\text{rank}(f, I_{i,j})$. A threshold is defined as the top $k\%$ of features. If there are $|\mathcal{F}|$ features, the threshold T is calculated as $T = k \times |\mathcal{F}|/100$. For each feature f , an indicator function \mathbb{I} determines whether the rank of f is within the top T features in each combination of S_i , \mathcal{M}_j , and \mathcal{FI}_j , returning 1 if it is within the top T and 0 otherwise.

The stability score is calculated by averaging the values of the indicator function across all combinations of subsets and learning models - feature importance methods. Mathematically, this is represented as:

$$SS(f) = \frac{1}{|\mathcal{M}| \cdot |\mathcal{S}|} \sum_{i=1}^{|\mathcal{S}|} \sum_{j=1}^{|\mathcal{M}|} \mathbb{I}(\text{rank}(f, I_{i,j}) \leq T) \quad (2)$$

where $|\mathcal{M}|$ is the number of learning model - feature importance method pairs and $|\mathcal{S}|$ is the number of subsets and $T = k \times |\mathcal{F}|/100$ as described. Dividing by the total number of $S_i - (\mathcal{M}_j, \mathcal{FI}_j)$ combinations ensures that the stability score value always falls within the $[0,1]$ interval.

The stability score $SS(f)$ (Eqn. (2)) reflects how consistently a feature f ranks among the top $k\%$ features across different subsets and learning model - feature importance method combinations. A higher stability score indicates that the feature is frequently deemed important, suggesting robust importance. By computing the stability score $SS(f)$ (Eqn. (2)), the SWEFI algorithm identifies features with both high importance and consistent significance across multiple combinations, ensuring that the selected features are reliable and stable.

2.6 Ensemble Importance

In our study, we've devised a method to determine the combined importance of features by blending scores from different models and data groupings to identify which features consistently stand out. Specifically, we aim to compute the ensemble importance for each feature f by aggregating its importance scores from various combinations, weighted by stability scores.

To achieve this, the inputs include the importance scores $I_{i,j}(f)$ (Eqn. (1)) and the stability scores $SS(f)$ (Eqn. (2)). The importance scores $I_{i,j}(f)$ are derived from the j -th model using the j -th feature importance method on the i -th subset, while the stability scores $SS(f)$ measure the consistency of feature importance rankings across different combinations. Higher stability scores indicate that the feature ranks consistently within the top $k\%$ across different settings.

To compute the ensemble importance, we use a weighted average of the feature importance scores $I_{i,j}(f)$ (Eqn. (1)), with the stability scores $SS(f)$ (Eqn. (2)) serving as weights. This ensures that features consistently important across various combinations have a greater influence on the final ensemble importance. Mathematically, the ensemble importance is computed as follows:

$$EI(f) = \frac{SS(f)}{\sum_f SS(f)} \cdot \sum_{i,j} I_{i,j}(f). \quad (3)$$

The rationale behind using stability scores $SS(f)$ (Eqn. (2)) as weights is to emphasize features that are consistently important across different learning models. This approach reduces the influence of features that may be significant in only a few models or subsets, leading to a more robust and reliable feature selection process.

This process ensures that selected features are not only important according to individual models and methods but also stable and reliable across multiple iterations, contributing to a more robust feature selection. The emphasis on stability scores in Eqn. (3) means that features consistently deemed important, regardless of data or model variations, receive a higher overall importance score, while features whose importance fluctuates with changes in data or models receive a lower score.

2.7 Feature Selection

The final step of the SWEFI algorithm focuses on selecting a subset of features \mathcal{F}_s based on their ensemble importance scores $EI(f)$. These scores, computed in the previous step, reflect the aggregated importance of each feature across different learning model - feature importance method and subsampling iterations, weighted by their stability scores $SS(f)$ ((Eqn. (2))). The feature selection process involves setting a threshold τ on $EI(f)$ to determine which features should be included in \mathcal{F}_s . Features with higher ensemble importance score $EI(f)$, Eqn. (3) are prioritized, ensuring that selected features are not only individually significant across various models and methods but also exhibit stability across different data subsets and modeling techniques. By selecting features based on their ensemble importance, the algorithm aims to optimize the feature set for subsequent modeling tasks, potentially improving model performance and interpretability. This approach enhances the robustness of feature selection by focusing on features that consistently demonstrate high importance across diverse modeling conditions.

Feature selection based on ensemble importance that sums up to 1 typically involves identifying and retaining the most important features while discarding the less important ones. One effective strategy for this purpose is selection based on Cumulative Importance. Cumulative importance for feature selection involves retaining features that collectively contribute to a significant portion of the total feature importance. This method is based on the idea that a small subset of features often carries most of the predictive power of a model, allowing for a reduction in dimensionality without significant loss of information. For instance, one might retain features that contribute to cumulative importance above a threshold $\tau = 0.9$ of the total importance. Algorithm 2 demonstrates the feature selection based on cumulative importance.

Algorithm 2 Feature Selection Using Cumulative Importance

Require: $\tau \in (0, 1]$, Filtration threshold

Require: $I \in \mathbb{R}^m$, Ensemble Importances,

Ensure: Selected feature indices i_s

- 1: $I \leftarrow f(D)$ where $f \in \{\text{SWEFI, MDA, MDI, ...}\}$
 - 2: $I \leftarrow I / \sum I_i$ (Normalize importances)
 - 3: $i \leftarrow \text{argsort}(I)[::-1]$ (Sort descendingly)
 - 4: $\text{CI} \leftarrow \text{cumsum}(I[i])$
 - 5: $k \leftarrow \min\{j \mid \text{CI}[j] \geq \tau\} + 1$
 - 6: $i_s \leftarrow i_s[:k]$ **return** i_s
-

2.8 Algorithm Stability

Stability refers to the consistency of feature importance scores across different datasets, model configurations, and random initializations. A stable algorithm consistently identifies the same features as significant, regardless of minor changes in data or model parameters.

Various studies have delved into the reasons behind this instability (Alelyani et al. [2013]). identified that a small sample-to-feature ratio is a primary cause, as it leads to high variance in the selected features. Noise in the data is another major contributor, as highlighted by Shanab et al. [2012], which can obscure the true relationships between features and the target variable, resulting in inconsistent feature selection. Additionally, Awada et al. [2012] pointed out that an imbalanced target variable exacerbates selection instability, as the model becomes biased towards the majority class, affecting the feature importance rankings. Furthermore, feature redundancy, discussed by Somol and Novovicova [2010], can lead to instability since redundant features provide overlapping information, causing the selection algorithm to arbitrarily choose among them, leading to variability in the feature sets across different runs.

Stable feature importance algorithms are vital for several reasons. First, they ensure reliability in decision-making. Decision-makers rely on machine learning models to make informed choices, and if feature importance scores fluctuate significantly, it becomes difficult to trust the model's recommendations. Stable scores provide a reliable basis for decision-making, ensuring that identified features are genuinely influential in the model's predictions. Second, stability enhances model interpretability. Interpretability is crucial in regulated industries where understanding the model's behavior is necessary for compliance and trust. Stable feature importance scores consistently highlight the same important features, making it easier for stakeholders to understand and validate the model's decisions. Lastly, stability contributes to the robustness of noisy data. Real-world data is often noisy and variable, and a stable feature importance algorithm can identify significant features despite noise, leading to more resilient and reliable models. This robustness is especially important in fields like finance, where data can be highly volatile and unpredictable.

Pearson's Correlation

A widely-used stability measure based on weights calculates the average correlation between feature weights across different selection runs:

$$\phi_{\text{pears}} = \frac{2}{M(M-1)} \sum_{i=1}^{M-1} \sum_{j=i+1}^M \rho_{i,j}, \quad (4)$$

where

$$\rho_{i,j} = \frac{\sum_{f=1}^d (w_{f,i} - \mu_i)(w_{f,j} - \mu_j)}{\sqrt{\sum_{f=1}^d (w_{f,i} - \mu_i)^2 \sum_{f=1}^d (w_{f,j} - \mu_j)^2}}. \quad (5)$$

In these equations, $w_{f,i}$ represents the score assigned to feature f in selection run i , and μ_i denotes the average weight of the features in that run. According to [Nogueira and Brown \[2016\]](#), if the feature weights are binary (0 or 1, indicating whether the feature is selected) and the number of non-zero weights remains constant across selection runs, then ϕ_{pears} (Eqn. (4)) is equivalent to the Kuncheva index.

The measure ϕ_{pears} (Eqn. (4)) evaluates the consistency of feature selection and weighting across multiple runs. By calculating the average Pearson correlation coefficient between feature weights from different runs, this measure provides insight into how reliably features are selected across different iterations. A high ϕ_{pears} (close to 1) indicates high stability, meaning that the feature selection process is consistent across runs. Conversely, a low ϕ_{pears} (close to 0 or negative) suggests low stability, reflecting significant variability in feature selection between runs. Thus, ϕ_{pears} offers valuable information about the robustness and reliability of the feature selection process.

Kuncheva Index

The Kuncheva Index (KI) is a stability measure for feature selection methods ([Kuncheva \[2007\]](#)) and is used to assess how consistently features are selected across different runs. The equation calculates the similarity between feature subsets F_i and F_j over all possible pairs of subsets, as given by:

$$KI = \frac{2}{M(M-1)} \sum_{i=1}^{M-1} \sum_{j=i+1}^M \frac{|F_i \cap F_j| - \frac{k^2}{d}}{k - \frac{k^2}{d}}, \quad (6)$$

where $|F_i \cap F_j|$ is the number of features that subset F_i and subset F_j have in common, k is the number of selected features (assumed to be constant), M is the number of selected subsets, and d is the total number of features. The Kuncheva Index (KI) measures the stability of feature selection. A high KI value indicates that the features selected are consistent across different runs, thus providing confidence in their importance. Conversely, a low or negative KI suggests that the feature selection process may not be very reliable, as it indicates variability in the features selected across different iterations.

Importance Weighted Kuncheva Index

The Importance Weighted KI (ϕ_{iw}) is a stability measure proposed by Hamer and Dupnot [Hamer and Dupont \[2021\]](#), which weights the contributions of the features in the stability by their relative importance in the associated runs. Specifically, they define ϕ_{iw} as

$$\phi_{\text{iw}} = \frac{2}{M(M-1)} \sum_{i=1}^M \sum_{j=i+1}^M \frac{|\mathcal{F}_i \cap \mathcal{F}_j|_{\text{iw}} - |\mathcal{F}_i \cap \mathcal{F}_j|_{\text{iw}}^{\text{rand}}}{\bar{k} - C}, \quad (7)$$

where

$$|\mathcal{F}_i \cap \mathcal{F}_j|_{\text{iw}} = \sum_{f \in \mathcal{F}_i \cap \mathcal{F}_j} \min(I_{f,i}, I_{f,j}), \quad (8)$$

$$|\mathcal{F}_i \cap \mathcal{F}_j|_{\text{iw}}^{\text{rand}} = \frac{1}{d} \sum_{f \in \mathcal{F}_i, f' \in \mathcal{F}_j} \min(I_{f,i}, I_{f',j}), \quad (9)$$

$$\bar{k} = \sum_f I_{f,i}, \quad \forall 1 \leq i \leq M, \quad (10)$$

and

$$C = \frac{2}{M(M-1)} \sum_{i=1}^M \sum_{j=i+1}^M |\mathcal{F}_i \cap \mathcal{F}_j|_{\text{iw}}^{\text{rand}}. \quad (11)$$

Here, $I_{f,i}$ represents the importance of the selected feature f in predictive model number i . Essentially, the similarity between two selection runs is defined as the sum of the common importance that selected features have between both decision models. The overall stability value is then the average of the pairwise similarities, normalized and corrected for chance.

Hamer and Dupnot also show that ϕ_{pears} is capable of identifying instability correctly. However, ϕ_{pears} has some important limitations and may behave inadequately in certain scenarios, particularly when the feature space consists of highly correlated feature groups. Algorithm 3 demonstrates how to compute accord stability.

Algorithm 3 Stability Measure Computation

Require: Set of Datasets $\{D_i : (X_i, y_i)\}_{i=1}^S$ with n samples

Ensure: Stability measure Φ

```

1: for each  $D_i$  do
2:    $I_i \leftarrow \mathcal{FI}(D_i)$  where  $\mathcal{FI} \in \{\text{swefi}, \text{mda}, \text{mdi}, \dots\}$ 
3: end for
4:  $\Phi \leftarrow \phi(\{I_i\}_{i=1}^S)$  where  $\phi \in \{\phi_{\text{iw}}, \phi_{\text{pears}}, \dots\}$  return  $\Phi$ 

```

3 Empirical Analysis

3.1 Dataset Generation

The dataset generation process involves creating synthetic time series and cross-sectional data tailored for binary classification tasks. The features are categorized into informative, redundant, and noise subsets. Informative features are directly linked to the labels, while redundant features are variations of informative ones with added Gaussian noise. Noise features carry no useful predictive information.

Synthetic Time Series Generation

This section details the approach for generating synthetic time series curated for binary classification tasks. Given a dataset, the primary objective is to assign one of two possible labels based on the data features. We introduce F as the comprehensive feature set, which is systematically partitioned into three discrete subsets: I for informative features, R for redundant features, and N for noise features.

Informative features are those from which the label can be determined absolutely. Redundant features are generated by adding Gaussian noise with a standard deviation of σ to the individual informative features. A smaller value of σ^2 indicates that the redundant feature

closely resembles its originating informative feature. In contrast, noise features are irrelevant to the label and do not contain any valuable information for predicting the label.

All features are time series generated by Autoregressive Moving-Average (ARMA) processes. An ARMA process combines autoregression (AR) and moving average (MA). The AR component models the time series based on its past values, characterized by the order p , which specifies the number of lagged observations included. Mathematically, an AR process of order p (AR(p)) is represented as:

$$X_t = c + \sum_{i=1}^p \phi_i X_{t-i} + \epsilon_t, \quad (12)$$

where X_t is the time series value at time t , c is a constant, ϕ_i are the parameters of the model, and ϵ_t is white noise.

The MA component models the time series based on past forecast errors, characterized by the order q , which specifies the number of lagged forecast errors included. An MA process of order q (MA(q)) is represented as:

$$X_t = \mu + \sum_{j=1}^q \theta_j \epsilon_{t-j} + \epsilon_t, \quad (13)$$

where μ is the mean of the series, θ_j are the parameters of the model, and ϵ_t is white noise.

Combining both AR and MA components, an ARMA(p, q) process is formulated as:

$$X_t = c + \sum_{i=1}^p \phi_i X_{t-i} + \sum_{j=1}^q \theta_j \epsilon_{t-j} + \epsilon_t. \quad (14)$$

Algorithm 4 generates ARMA time series with specified parameters (Φ_i , Θ_j and σ_ϵ). By following these steps, we can generate time series data that exhibit the statistical properties governed by the ARMA process parameters.

Finally, all individual features are combined into a feature matrix X , and a label vector y is formed based on the mean of the informative features across time steps. The outputs of this process are the matrix X and the vector y .

Algorithm 4 Generate Time Series using ARMA Process

Require: Define the orders p and q for the AR and MA components, respectively.

Require: Specify the parameters ϕ_i and θ_j .

```

1: Generate a series of white noise  $\epsilon_t$ .
2: for  $t = 1$  to  $T$  do
3:   if  $t \leq \max(p, q)$  then
4:     Initialize  $X_t$  with appropriate initial values ( $=0$ ).
5:   else
6:     Compute  $X_t$  using the ARMA equation:
7:      $X_t = c + \sum_{i=1}^p \phi_i X_{t-i} + \sum_{j=1}^q \theta_j \epsilon_{t-j} + \epsilon_t$ ,
8:     (where  $c$  is a constant)
9:   end if
10: end for
```

Synthetic Cross Section Generation

The dataset for the challenging enough classification problem can be generated using the snippet code provided by De Prado [2018].

3.2 Data Subsampling

We use time-series bootstrapping methods for data resampling, which differ significantly from ordinary bootstrapping in treating temporal dependencies during resampling. Time-series bootstrap methods are designed to preserve the time structure and autocorrelation of the original data, making them particularly suitable for time-series analysis. In contrast, ordinary bootstrapping assumes independence among observations and is a more general-purpose technique. While it can be applied to a wide range of data types, it may be inappropriate for time series data due to its inability to account for temporal dependencies. To address this, we divide the dataset into several blocks using the stationary bootstrapping method [Politis and Romano \[1994\]](#), which effectively maintains the original temporal relationships and other unique features of financial time series data.

3.3 Pre Hyperparameter Tuning

Optimizing machine learning models necessitates careful hyperparameter tuning to determine the optimal values for parameters not inherently learned during training, such as regularization coefficients, kernel choices, and learning rates. This process is crucial for achieving the best model performance and ensuring that the model generalizes well to unseen data. Validation during hyperparameter tuning is essential to prevent issues like overfitting or underfitting. Cross-validation is a common technique used for this purpose, where the dataset is divided into multiple subsets or 'folds.' The model is then trained and evaluated iteratively on different combinations of these folds, leading to an averaged performance estimate over all iterations.

Cross-validation is vital in preventing sub-optimal hyperparameter choices that could negatively impact model performance and help determine their optimal values, thereby ensuring the model performs at its best. Cross-validation supports hyperparameter tuning and is integral to achieving high-performing machine-learning models. This process enhances predictive accuracy and strengthens the model's decision-making capabilities ([Gareth et al. \[2013\]](#)).

For hyperparameter tuning, the Bayesian approach is preferred over common approaches like grid search and random search. Bayesian hyperparameter tuning provides a more advanced approach than traditional grid and random search methods by utilizing probabilistic models to guide the search process efficiently. Unlike grid search, which methodically examines a predefined set of hyperparameters, and random search, which samples randomly, Bayesian optimization employs a surrogate model (often a Gaussian process) to approximate the objective function mapping hyperparameters to performance. This model is iteratively refined with the results of previously tested hyperparameters to make informed predictions about new hyperparameters. An acquisition function, such as Expected Improvement or Upper Confidence Bound, directs the next evaluations by balancing exploration and exploitation. The key advantages of Bayesian optimization include increased efficiency, requiring fewer iterations to find optimal hyperparameters; improved performance, as it systematically enhances the surrogate model; and greater adaptability, as it adjusts its strategy based on observed outcomes. This makes Bayesian optimization particularly advantageous for computationally intensive training processes and complex hyperparameter spaces ([Shahriari et al. \[2015\]](#)).

3.4 Comparison with Existing Literature

Previous works introduce the MDI and MDA feature importance methods, which assign importance to features of an input financial dataset. While these methods are innovative, they have limitations, as discussed below. Specifically, we encountered scenarios where the MDA method failed to assign appropriate feature importance to dataset features generated with the parameters listed in [Tables 1 and 2](#). This failure occurred for both time series and cross-sectional data using SVM as the base learning model for MDA. As shown in [Figures 1\(a\) and 2\(a\)](#), informative features were computed as less important than noise features,

Parameter	Value
Dataset Parameters	
I : Number of informative features	5
R : Number of redundant features	20
N : Number of noise features	5
t: Number of time steps	2
n: Number of samples	1000
σ : Standard Deviation of noise added to $\{R\}_{i=1}^r$	0.25
ARMA Parameters	
ϕ_{-1} : Autoregressive parameters	0.5
θ_{-1} : Moving average parameters	0.5
σ_ϵ : Standard Deviation of ARMA noise	1

Table 1: Parameter values for dataset and ARMA model

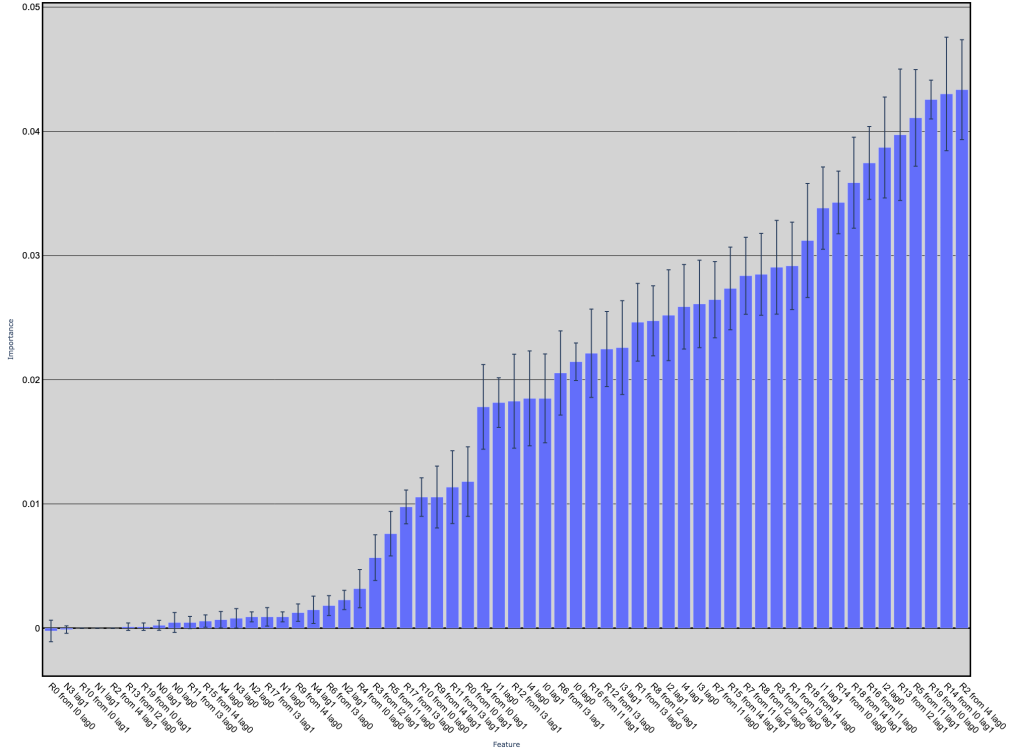
Parameter	Value
Dataset Parameters	
I : Number of informative features	5
R : Number of redundant features	20
N : Number of noise features	5
n: Number of samples	1000
σ : Standard Deviation of noise added to $\{R\}_{i=1}^r$	0.1

Table 2: Parameter values for the dataset

contrary to the expected higher importance values for informative features based on the dataset generation process.

To address this, we applied the SWEFI method to the previously generated dataset. The results, illustrated in Figures 1(b) and 2(b), demonstrate the correct ranking of informative and redundant features compared to noise features. The significant gap between the importance values of informative and redundant features versus noise features simplifies filtering out noise features, presenting an advantage of the SWEFI algorithm. Additionally, the lower contribution of noise and redundant features compared to informative ones, as shown in Figures 1 and 2, is an evident benefit of the SWEFI method over MDA.

A comparison of SWEFI and MDA results reveals that MDA’s importance ranking is incorrect, while SWEFI provides the correct ranking. In MDA feature importance, noise features receive higher importance than informative features, contradicting the dataset generation logic. Replacing the base learner with Linear Discriminant Analysis, Logistic Regression, or Naive Bayes did not rectify the MDA results.



(a) MDA results (Base Learner Linear SVM(C=0.1))

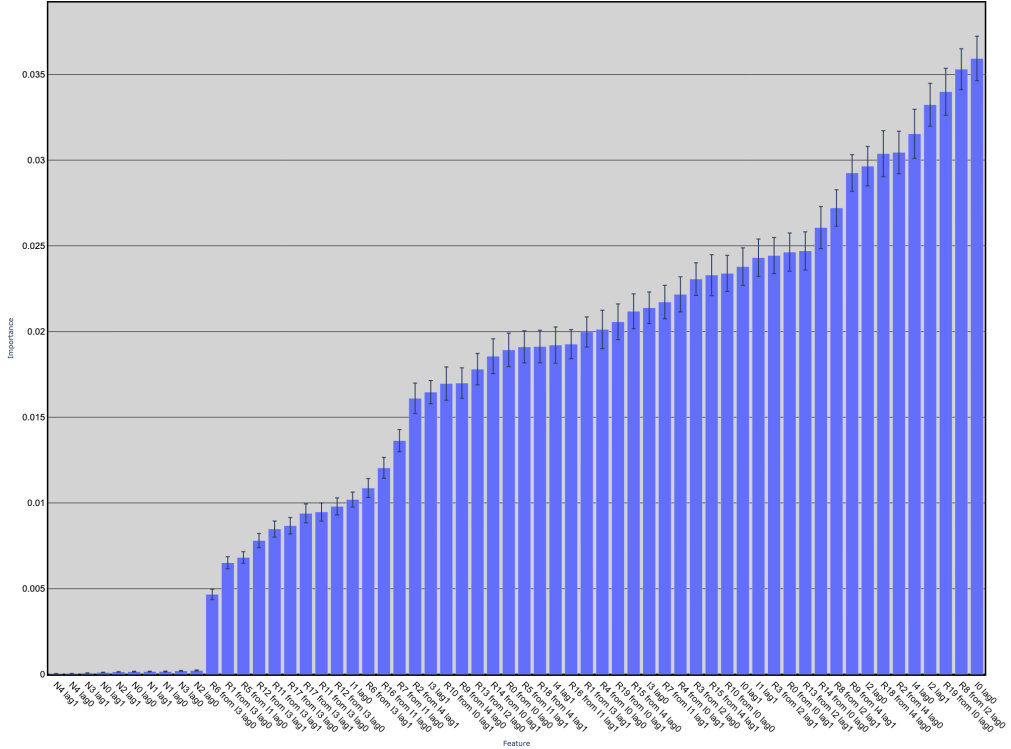
(b) SWEFI results ($k = 0.5$)

Fig. 1: (Set $\tau = 1$: No Selection) Comparison between feature importance results of SWEFI ($k = 50\%$) and MDA show the informative (and redundant) features being computed as less important than noise features. The experiment run on the synthetic time series data generated based on Table 1 parameters.

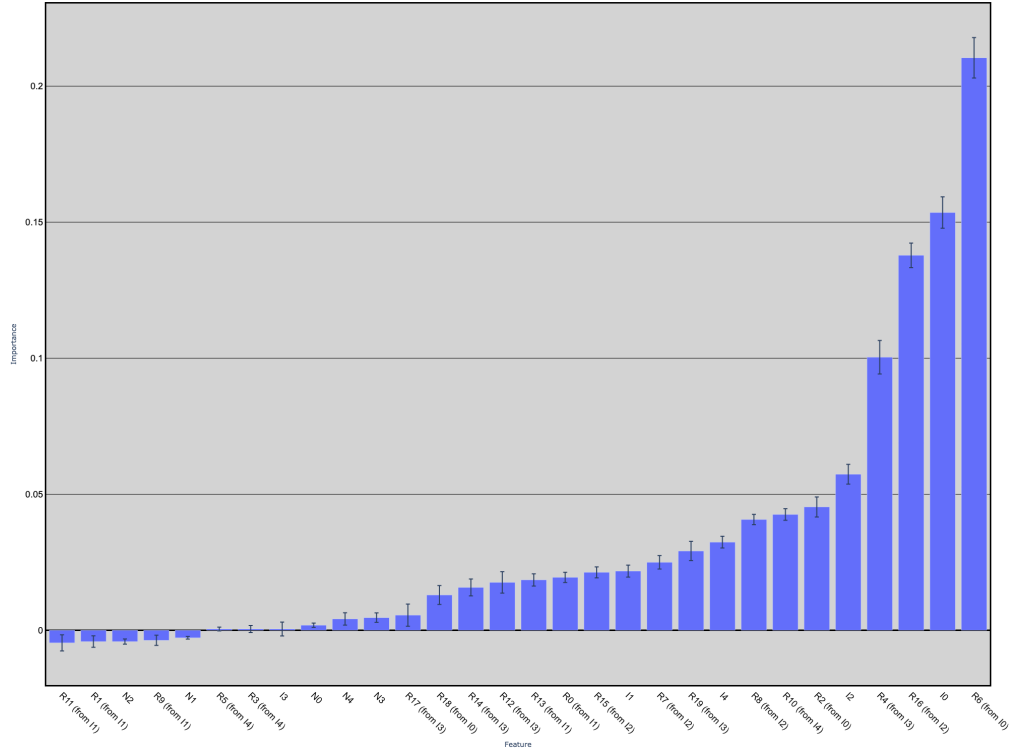
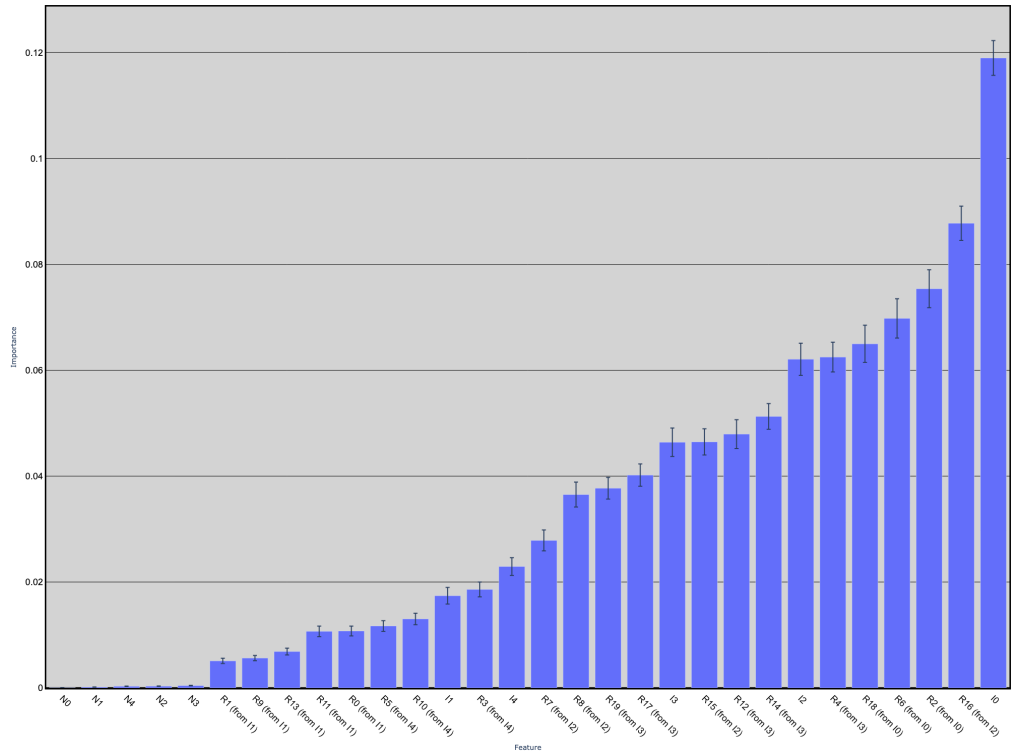
(a) MDA results (Base Learner Linear SVM($C=0.1$))(b) SWEFI results ($k = 0.5$)

Fig. 2: (Set $\tau = 1$: No selection) Comparison between feature importance results of SWEFI ($k = 50\%$) and MDA show the informative (and redundant) features being computed as less important than noise features. The experiment run on the synthetic cross-sectional data generated based on Table 2 parameters.

Parameter	Value
Dataset Parameters	
$\ I\ $: Num. of informative features	5
$ R $: Num. of redundant features	20
$ N $: Num. of noise features	5
t : Num. of time steps	2
n : Num. of samples	12500
σ : Standard Deviation of noise added to $\{R\}_{i=1}^r$	0.25
Time Series Parameters	
ϕ_{-1} : Autoregressive parameters	0.5
θ_{-1} : Moving average parameters	0.5
σ_ϵ : Std of ARMA noise	1
Subsets Parameters	
S : (Num. of Subsets)	5
n_s : Num. of Subset Samples	2500
Resampling Method	Stratified CV

Table 3: Parameters for Dataset, Time Series, and Subsets

3.5 Algorithm Stability Comparison

Stability is another crucial aspect of feature-importance methods. It refers to the consistency of importance rankings across different samples from the same data distribution. High stability indicates that feature importance rankings do not change significantly with different samples from the same distribution, reflecting the method’s ability to produce reproducible results. Therefore, high stability is as crucial as high classification accuracy when assessing a feature importance algorithm.

To estimate the stability of the SWEFI method compared to methods introduced by De Prado, we generated subsets by subsampling with the configurations listed in Table 3 using Algorithm 4. By computing feature importances for each dataset, we determined the stability measure as described in Algorithm 3. Figures 3(a), 3(b), 3(c), and 3(d) illustrate that SWEFI demonstrates greater stability compared to previous methods.

In feature selection, balancing performance and stability is crucial. Stability pertains to the consistency of selected features across various datasets or iterations, while predictive accuracy indicates the effectiveness of these features in making predictions. Perturbing a dataset to create multiple variations can lead to instability, complicating feature interpretation, and diminishing trust, especially in financial machine learning, where reproducibility is paramount. While optimizing for stability alone is straightforward, a meaningful approach must balance stability and predictive accuracy. As Figure 3(a) shows, SWEFI selects the most important informative features, ensuring predictive performance based on the dataset generation process, where the target variable is determined by informative features.

In contrast, Figures 3(b), 3(c), and 3(d) reveal that methods with less stability, such as MDA, include many redundant and noise features, which decrease the predictive performance of models trained on these features.

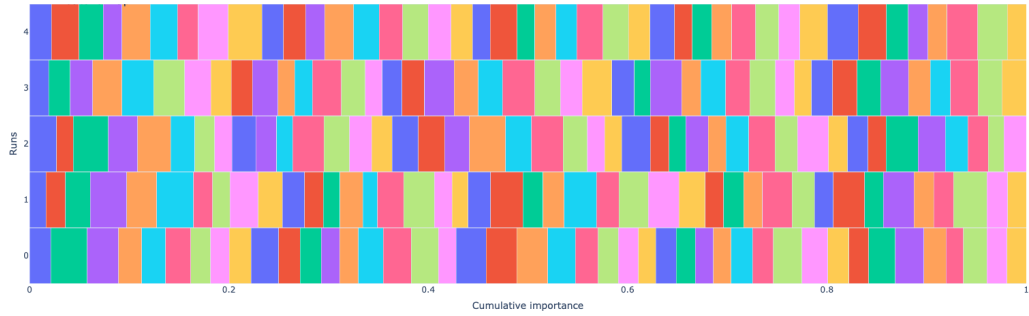
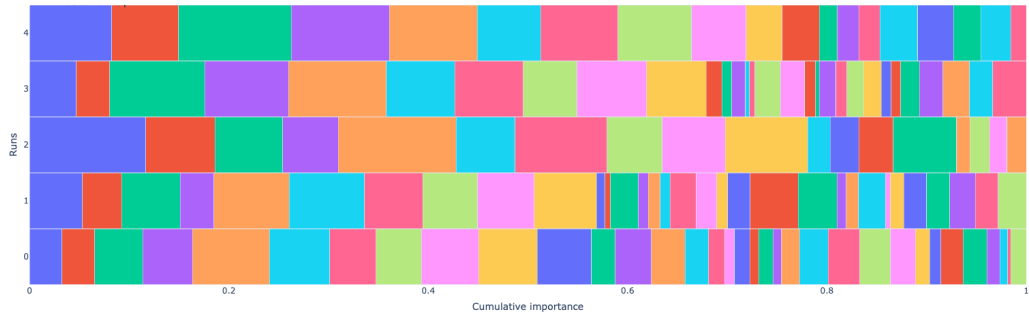
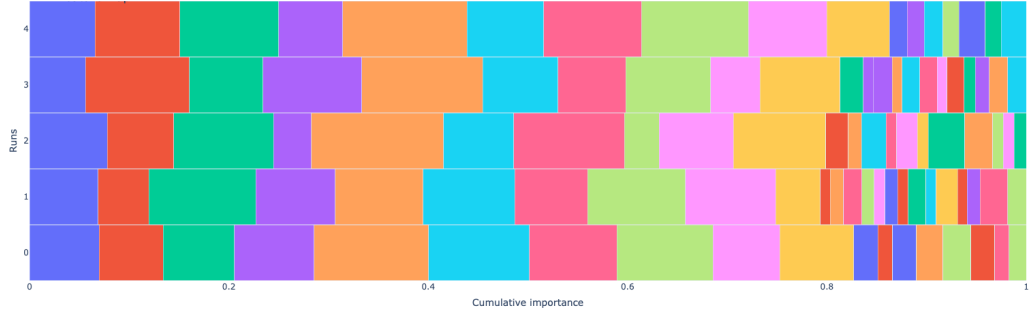
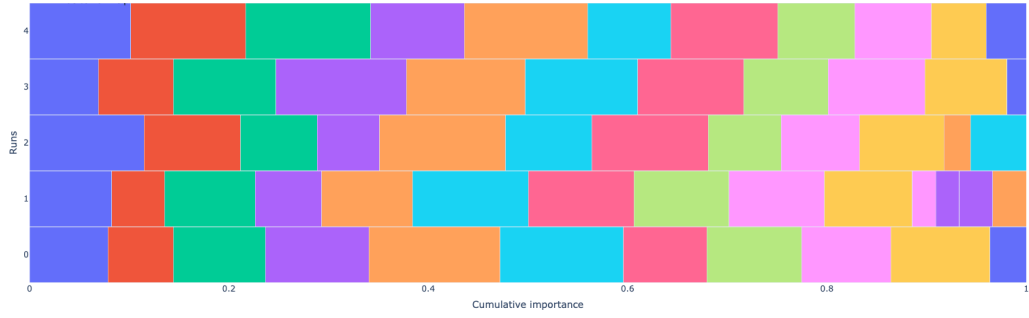


Fig. 3: (Set $\tau = 0.9$ for selection step) Comparison of feature maps across subsets indicates that there is a significant gap in stability between SWEFI and previous methods, with SWEFI demonstrating greater stability.

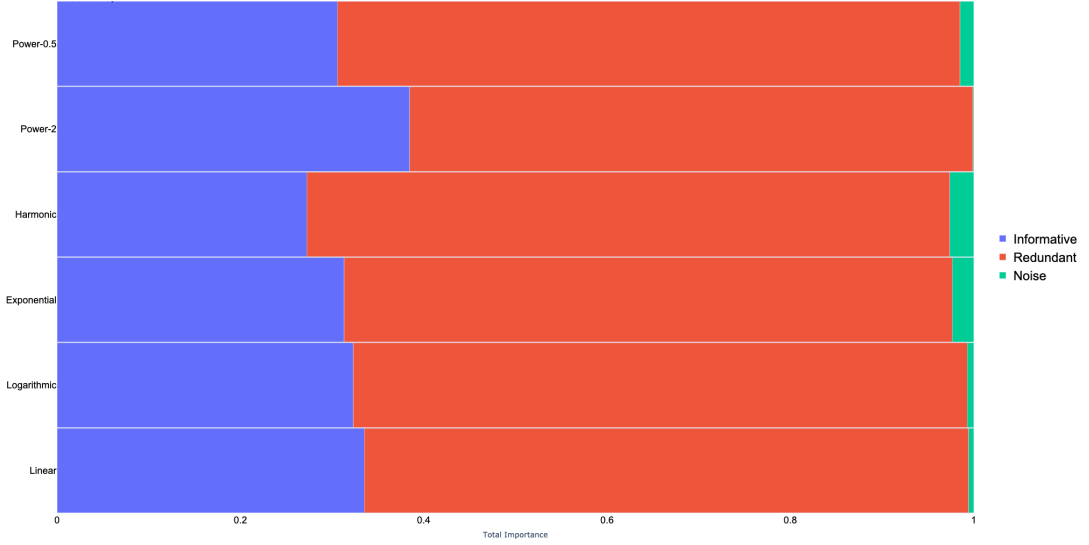


Fig. 4: Comparison between total sum of scores for the informative, redundant and noise features on different weighting strategies.

3.6 Stability Weighting Strategy

Given the stability scores $SS(f)$ (Eqn. 2) and the feature importance scores $I_{i,j}(f)$ (Eqn. 1), Instead of employing a simple linear weighting strategy, the ensemble importance can be computed using various weighting strategies as follows:

$$\text{Exponential Weighting} := \frac{e^{SS(f)}}{\sum_f e^{SS(f)}} \cdot \sum_{i,j} I_{i,j}(f) \quad (15)$$

$$\text{Power Weighting} := \frac{SS(f)^\alpha}{\sum_f SS(f)^\alpha} \cdot \sum_{i,j} I_{i,j}(f) \quad (16)$$

$$\text{Harmonic Mean Weighting} := \sum_{i,j} \left(\frac{2 \cdot SS(f) \cdot I_{i,j}(f)}{SS(f) + I_{i,j}(f)} \right) \quad (17)$$

$$\text{Logarithmic Weighting} := \frac{\log(1 + SS(f))}{\sum_f \log(1 + SS(f))} \cdot \sum_{i,j} I_{i,j}(f) \quad (18)$$

Selecting the appropriate method for computing ensemble importance depends on the context of the problem and specific goals. For simplicity, simple weighting is ideal (as used in Eqn. 3). If the emphasis is on achieving high stability scores, Exponential (Eqn. 15) and Power (Eqn. 16) weighting strategies are more suitable. To balance influence, the Harmonic Mean (Eqn. 17) weighting strategy should be considered. For smooth scaling, the Logarithmic (Eqn. 18) weighting strategy works best.

The results of the SWEFI algorithm on different weighting strategies applied to the dataset with a configuration similar to Table 2, but with σ replaced by 0.5 to increase the complexity of the classification task, are shown in Figures 4 and 5. In these experiments, it can be seen that the Power weighting strategy (with $\alpha > 1$) achieves a higher total sum of scores on the informative features due to its emphasis on higher stability scores. This strategy expects higher relative stability scores on the informative features compared to the redundant and, specifically, the noise features.

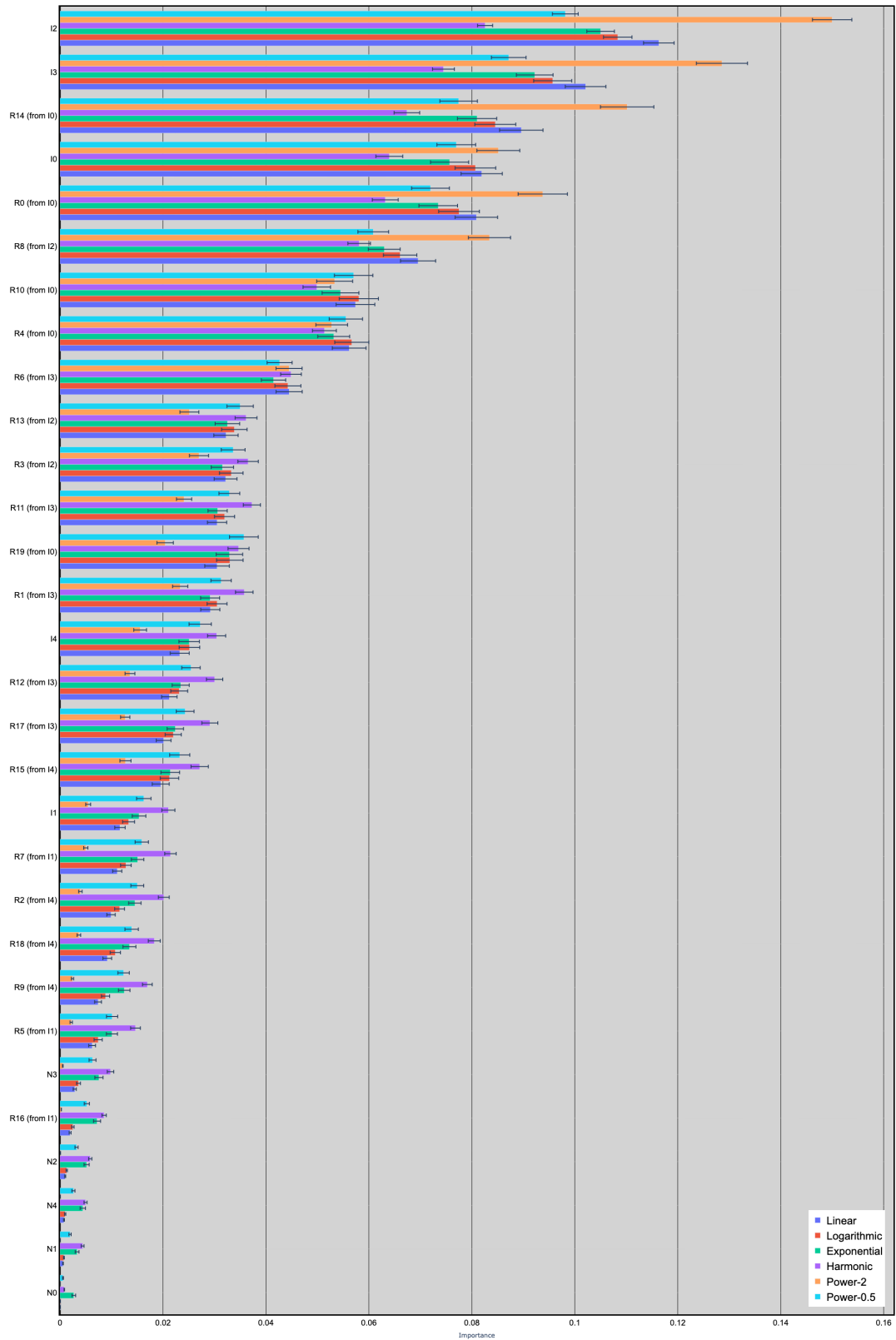


Fig. 5: Comparison between the computed scores of each individual feature on different weight-
ing strategies.

4 Conclusion and Future Research

The research introduces the SWEFI method, which addresses critical challenges in feature selection for high-dimensional financial datasets. SWEFI uniquely integrates stability selection with ensemble learning, significantly enhancing the stability and interpretability of feature importance scores. This approach effectively mitigates the instability and inconsistency issues prevalent in traditional methods, particularly when dealing with complex and noisy financial data. The extensive experiments demonstrate SWEFI's superior accuracy, stability, and robustness compared to existing methods, highlighting its practical applicability in real-world financial scenarios. The results emphasize SWEFI's capability to improve model trustworthiness and decision-making processes, which is crucial for high-stakes applications.

Moreover, the research explores the trade-offs between stability and accuracy, providing valuable insights for future research and practical implementations. The study underscores the importance of stability in feature importance measures, suggesting that stable and interpretable feature importance scores are essential for reliable machine learning models in finance and beyond. Future research directions include further refinement of the SWEFI method, exploring its applicability to other domains, and investigating additional techniques to enhance stability and interpretability in machine learning models.

The discussion on substitution effects and the orthogonalization of features highlights a fundamental challenge in feature importance analysis: the handling of multicollinearity. Substitution effects arise when predictive information is shared between features, leading to biased feature importance results. Traditional methods like MDI and MDA can misjudge the importance of highly codependent features. Orthogonalization, such as applying Principal Component Analysis (PCA), partially addresses this issue but introduces new complexities, including the loss of intuitive explanations and potential misalignment with out-of-sample performance optimization. Feature clustering mitigates the impact of substitution effects in feature importance models such as MDI and MDA. The substitution effect refers to the influence of substituting one input variable for another on the results produced by feature importance methods (Belsley et al. [2005], Hill and Adkins [2001]). In the case of MDI, when features are highly correlated, the substitution effect can cause the importance of redundant features to be split among them. This can lead to an underestimation of the true importance of these features. For instance, if two features provide similar information, the model might use one feature in some trees and the other feature in different trees, diluting the apparent importance of both features. Similarly, in the case of MDA, when features are highly correlated, the substitution effect can distort importance measures. Permuting one of the correlated features can have a minimal impact on accuracy because the model can still rely on the other correlated features, leading to an underestimation of their individual importance.

Clustered Feature Importance (CFI) offers a promising alternative that maintains the interpretability of results while mitigating substitution effects. By clustering similar features and evaluating their importance at the cluster level, CFI reduces the impact of multicollinearity without requiring a change of basis. This approach preserves the original feature space, making the results more intuitive and applicable in practical scenarios. Investigating the empirical comparison of CFI with traditional feature importance methods (MDI, MDA) and orthogonalization techniques (like PCA) could provide valuable insights (De Prado [2020]). This includes evaluating their performance in terms of stability, interpretability, and predictive power across different datasets and domains. Examining how CFI affects overall model performance, particularly in high-dimensional and noisy datasets, involves analyzing whether clustering similar features before importance evaluation can lead to better generalization and reduced overfitting. Assessing the interpretability and usability of CFI results in practical applications is also crucial. This includes investigating how domain experts perceive the clustered feature importance results and how these insights influence decision-making processes. Exploring the application of CFI in different domains, such as finance, healthcare, and mar-

keting, is essential. Each domain has unique characteristics, and understanding how CFI performs across these contexts can provide a broader perspective on its utility.

Building on the SWEFI framework, future research can extend this method to incorporate clustered feature importance, creating a Stability Weighted Ensemble 'Clustered' Feature Importance (SWE-CFI) approach. This extension aims to combine the benefits of stability weighting, ensemble learning, and clustered feature importance, offering a robust and interpretable feature selection method. Key research areas to develop and evaluate SWE-CFI include designing and implementing the SWE-CFI algorithm, which integrates stability selection, ensemble learning, and clustered feature importance. This involves clustering features using algorithms such as hierarchical clustering or k-means to group similar features and applying stability weighting and ensemble learning techniques to the clustered features to evaluate their importance at the cluster level.

References

1. Salem Alelyani, Jie Tang, and Huan Liu. The effect of the characteristics of the data on the stability of feature selection. In *Proceedings of the 2013 IEEE International Conference on Data Mining Workshops*, pages 556–563. IEEE, 2013.
2. American Statistical Association et al. Statement on statistical significance and p-values. *Am. Stat.*, 70:129–133, 2016.
3. Wafaa Awada, Taghi M. Khoshgoftaar, Donald J. Dittman, and Randy Wald. Prediction of imbalanced data with hybrid model of undersampling and evolutionary algorithm. *Proceedings of the IEEE 13th International Conference on Information Reuse & Integration (IRI)*, pages 180–185, 2012.
4. David A Belsley, Edwin Kuh, and Roy E Welsch. *Regression diagnostics: Identifying influential data and sources of collinearity*. John Wiley & Sons, 2005.
5. Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.
6. Marcos Lopez De Prado. *Advances in financial machine learning*. John Wiley & Sons, 2018.
7. Marcos M López De Prado. *Machine learning for asset managers*. Cambridge University Press, 2020.
8. Pradip Dhal and Chandrashekhar Azad. A comprehensive survey on feature selection in the various fields of machine learning. *Applied Intelligence*, 52(4):4543–4581, 2022.
9. Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.
10. James Gareth, Witten Daniela, Hastie Trevor, and Tibshirani Robert. *An introduction to statistical learning: with applications in R*. Springer, 2013.
11. Mark A Hall. *Correlation-based feature selection for machine learning*. PhD thesis, The University of Waikato, 1999.
12. Victor Hamer and Pierre Dupont. An importance weighted feature selection stability measure. *The Journal of Machine Learning Research*, 22(1):5153–5209, 2021.
13. R Carter Hill and Lee C Adkins. Collinearity. *A companion to theoretical econometrics*, pages 257–78, 2001.
14. Utkarsh Mahadeo Khaire and R Dhanalakshmi. Stability of feature selection algorithm: A review. *Journal of King Saud University-Computer and Information Sciences*, 34(4): 1060–1073, 2022.
15. Ludmila I Kuncheva. A stability index for feature selection. In *Artificial intelligence and applications*, pages 421–427. Citeseer, 2007.
16. Sarah Nogueira and Gavin Brown. Measuring the stability of feature selection. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part II 16*, pages 442–457. Springer, 2016.

17. SGOPAL Patro and Kishore Kumar Sahu. Normalization: A preprocessing stage. *arXiv preprint arXiv:1503.06462*, 2015.
18. Dimitris N Politis and Joseph P Romano. The stationary bootstrap. *Journal of the American Statistical association*, 89(428):1303–1313, 1994.
19. Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.
20. Thilo A Schmitt, Desislava Chetalova, Rudi Schäfer, and Thomas Guhr. Non-stationarity in financial time series: Generic features and tail behavior. *Europhysics Letters*, 103(5): 58003, 2013.
21. Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.
22. Mohamed E. Shanab, Mohamed H. Gadallah, and Magdy Z. Rashad. Noise detection in datasets using genetic programming and clustering. *International Journal of Computer Applications*, 59(2), 2012.
23. Petr Somol and Jana Novovicova. Evaluating the stability of feature selectors and their efficiency in feature selection stability problem. *Pattern Recognition*, 43(11):3817–3829, 2010.
24. Lars St, Svante Wold, et al. Analysis of variance (anova). *Chemometrics and intelligent laboratory systems*, 6(4):259–272, 1989.
25. Erik Štrumbelj and Igor Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and information systems*, 41:647–665, 2014.
26. Siva Sankari Subbiah and Jayakumar Chinnappan. Opportunities and challenges of feature selection methods for high dimensional data: A review. *Ingénierie des Systèmes d’Information*, 26(1), 2021.
27. Jorge R Vergara and Pablo A Estévez. A review of feature selection methods based on mutual information. *Neural computing and applications*, 24:175–186, 2014.
28. Ronald L Wasserstein, Allen L Schirm, and Nicole A Lazar. Moving to a world beyond $p < 0.05$, 2019.