Reichman University

Efi Arazi School of Computer Science

M.Sc. Program in Machine Learning and Data Science

Final Project Report

SmartSelect

Yehuda Baharav

Advisor: Ben Galili

June 19, 2025

1 Introduction

In today's data-driven landscape, machine learning models are frequently tasked with analyzing high-dimensional datasets—those containing hundreds, thousands, or even millions of features. This challenge is particularly pronounced in domains such as genomics, cybersecurity, image and speech recognition, and general data science, where data often includes a vast number of variables relative to the number of observations [8].

While having access to a large number of features might seem advantageous, in practice, it often introduces serious complications. Redundant, irrelevant, or noisy features can obscure the underlying patterns in the data, leading to decreased predictive performance. Additionally, they significantly increase the computational burden during model training and inference, which is particularly problematic when working under constraints of time or limited computing resources [5]. In extreme cases, the presence of such features can even mislead the model, causing it to learn spurious correlations instead of meaningful insights.

Feature selection—a critical preprocessing step in the machine learning pipeline—aims to address these challenges. By identifying and retaining only the most relevant and informative features, feature selection can improve model accuracy, reduce overfitting, accelerate training, and enhance interpretability [10]. However, no single feature selection technique works best in all scenarios. Some methods are statistically rigorous but computationally intensive, making them impractical for large-scale datasets. Others are fast and scalable but may compromise on selection quality, potentially overlooking subtle but important variables.

This project introduces a smart ensemble approach to feature selection, which combines the strengths of multiple existing methods to achieve a robust balance between performance (in terms of selection quality and resulting model accuracy) and efficiency (in terms of runtime and resource usage). Ensemble feature selection has been shown to improve stability and accuracy by aggregating results from multiple base methods [7].

A key innovation of this approach lies in its iterative pipeline structure. Instead of performing feature selection in a single pass, the system proceeds through multiple controlled iterations. In each iteration, it evaluates the feature set using the best available model—determined dynamically based on current constraints and performance metrics. These constraints, or regulations, include user-defined parameters such as maximum runtime, memory limits, or a cap on the number of features. The model choice at each step is adaptive: for example, the system might choose a simple and fast model early in the process to rapidly eliminate uninformative features, and then refine the selection using a more complex and accurate model as the feature space becomes smaller and more manageable.

This regulated, adaptive, and iterative process ensures that the pipeline remains both efficient and effective across a wide range of use cases and datasets. It continuously optimizes feature selection quality while respecting practical constraints, striking an optimal trade-off between speed and accuracy.

Furthermore, the implementation offers a high degree of user customization and control. Users can specify a range of parameters according to their needs and constraints, including:

- The target number of features (either as an exact number or a percentage of the original set).

- Maximum allowable runtime.
- Preferred feature selection strategies to include in the ensemble.
- Tolerance levels for correlation or redundancy among selected features, and more.

The approach is also data-agnostic, making no assumptions about the structure, scale, or domain of the input data. This enables it to be applied across disciplines, from healthcare to finance to engineering.

In summary, the proposed system seeks to provide a practical, powerful, and adaptable solution to one of machine learning's most enduring challenges—efficient and effective feature selection in high-dimensional spaces—through a smart, regulated, and iterative ensemble pipeline.

2 Related Works

Ensemble Feature Selection (EFS) is a family of techniques that aim to increase stability and accuracy by aggregating the results of multiple feature selection methods. The motivation is similar to ensemble learning in classification: reducing variance and bias by combining diverse perspectives [15].

2.1 EFS (Ensemble Feature Selection Framework)

A software tool that integrates the results of eight feature selection algorithms (e.g., ReliefF, SVM-RFE, Random Forest importance) by normalizing and aggregating their feature rankings or scores [14]

Pros:

- Increases robustness against noise or data perturbations.
- Outperforms individual methods in accuracy and stability.

Cons:

- Requires careful normalization of output scores.
- Can suffer if constituent methods are too similar or biased.

2.2 EFSIS (Ensemble Feature Selection Integrating Stability)

Combines two strategies:

- Function perturbation: Using various selection algorithms.
- Data perturbation: Applies each method on multiple bootstrapped datasets.

Then it aggregates the frequency or consistency of each feature's selection [18]. Pros:

- Focuses explicitly on stability, a known weakness of many selection algorithms.
- Empirically improves robustness and generalization across datasets.

Cons:

- Higher computational cost (due to multiple runs per method).
- Parameter tuning (e.g., number of resamples) can affect the results.

2.3 FS-MSI (Feature Selection via Multiple Score Integration)

This method assigns each feature a unified score by integrating scores from multiple traditional methods. Instead of raw ranks, it uses weighted averages or voting mechanisms [1]. Pros:

- Captures complementary insights from various methods.
- Demonstrates consistent performance gains in classification tasks.

Cons:

- Weights and scoring strategies must be tuned.
- Lacks model-specific adaptation unless combined with downstream learning.

2.4 Hyb-EFS (Hybrid Ensemble Feature Selection)

Combines homogeneous ensembles (e.g., using the same selection algorithm on different resampled datasets) and heterogeneous ensembles (e.g., different algorithms). Originally proposed for genomics data [4].

Pros:

- High reproducibility across biomedical datasets.
- More resilient to both model variance and data perturbation.

Cons:

- Complex to implement and evaluate.
- May require domain knowledge to tune ensemble composition.

Iterative and Adaptive Selection Techniques:

Recent research also focuses on adaptive and iterative pipelines that refine feature selection over time [6].

2.5 Stability Selection

Introduced by Meinshausen and Bühlmann, this method combines LASSO with subsampling. It selects features that consistently appear across subsamples and penalization strengths [13]

Pros:

- Theoretical guarantees on controlling false positives.
- Robust to overfitting.

Cons:

- Relatively slow.
- May miss weak but important features.

2.6 Boruta

A Random Forest-based wrapper method that compares the importance of real features to that of "shadow features" (randomly permuted copies) [11]. Pros:

- All-relevant feature selection (not just minimal-optimal).
- Works well for high-dimensional data like genomics.

Cons:

- Computationally heavy.
- Tends to retain many correlated features.

2.7 Recursive Feature Addition (RFA)

Starts with an empty set and adds features one-by-one, evaluating model performance (e.g., using cross-validation) at each iteration.
Pros:

- Controlled and explainable process.
- Can terminate early based on runtime or performance thresholds.

Cons:

- Sensitive to early feature choices.
- Slower than filter-based methods.

Table 1: Summary of Trade-Offs

Method Type	Pros	Cons
Filter	Fast, scalable, general	Ignores interactions
Wrapper	Good accuracy, interaction-aware	Slow, overfitting risk [17]
Embedded	Balanced, integrated	Model-specific
Ensemble	Stable, robust, hybrid	Complex, computationally intensive
Iterative	Adaptive, controllable	Slower, requires-tuning

2.8 Relevance to Our Work

This project builds upon the ideas of ensemble and iterative selection, proposing a smart, regulated, and adaptive ensemble pipeline that:

- Combines diverse selection strategies.
- Iteratively refines the feature set using the best available model at each stage.
- Respect user constraints like maximum runtime, number/percentage of features, or memory budget.
- It is fully data-agnostic and adaptable to any tabular dataset.

This hybrid approach is designed to deliver high-quality, explainable feature subsets — balancing interpretability, efficiency, and performance. [7]

3 Data Overview

The SmartSelect framework is designed to be data-agnostic, capable of operating on a wide range of datasets without requiring prior domain knowledge. It is tailored for tabular datasets, where each sample is represented by multiple features (columns) and a single target column [16]. The framework supports both regression and classification problems, as long as they are supervised, meaning that ground-truth labels for the target variable are provided.

In our experiments, we evaluated SmartSelect on multiple datasets exhibiting varying characteristics such as dimensionality, class imbalance, and data sparsity. The datasets ranged in size from small to large-scale, including those with tens of thousands of features and thousands of samples, to simulate realistic high-dimensional learning scenarios.

Due to repository constraints, actual datasets were not included; however, the system includes placeholders and compatible data loaders to allow seamless integration of external tabular datasets from .csv files.

4 Methodology

4.1 User Configuration

At the core of the SmartSelect framework is user flexibility. The user is required to define the maximum computational complexity they are willing to tolerate for both the feature selection phase and for the final benchmark model (described in detail below). This configuration enables SmartSelect to adjust its internal operations to the user's time or resource constraints.

4.2 Determining the Number of Selected Features

Users can explicitly specify the desired number of features to be selected, either as an absolute value or as a percentage of the original feature set. If no specification is provided, SmartSelect defaults to selecting a number of features equal to the square root of the number of data samples.

4.3 Feature Selection Strategy

A predefined list of feature selection methods is constructed, ordered according to their perceived quality. Since the quality of feature selection methods can be subjective and context-dependent [2], SmartSelect provides a default ranking which the user can later modify.

The framework then iteratively evaluates each method in the list against the user-defined computational complexity threshold:

1. If a method is deemed feasible within the given complexity limit, it is executed.
2. The resulting set of selected features is used to filter the dataset.
3. The method is removed from the current iteration list.
4. The system re-evaluates the remaining methods—this time using the reduced dataset—to check whether they now meet the complexity constraint.

If a method exceeds the allowed complexity, the system skips it and proceeds to the next one. If no remaining method is able to run under the current constraints (or none are left), a new iteration begins with the full set of predefined methods. However, this time the input is the reduced dataset output from the previous iteration.

To prevent trivial or ineffective filtering, each feature selection method is required to retain at least:

- 50% of the features it receives as input, and-
- No fewer features than the final target number specified by the user (if defined).

4.4 Stopping Criteria

The iterative selection process stops when any of the following conditions are met:

1. The number of features reaches the user-defined target.
2. The number of features is sufficiently reduced to enable running the benchmark model within the specified complexity limit.
3. None of the methods in the current iteration meet the runtime constraint. In this case, the system will stop and prompt the user to increase the allowed complexity.
4. The feature selection methods no longer perform additional filtering, yet the dimensionality is still too high to run the benchmark model. The system will again prompt the user to relax the complexity constraint of the benchmark model.

4.5 Benchmark Model

The benchmark model is designed to act as an independent, high-quality evaluator of the final feature sets.

Once the feature space is reduced to a level that permits execution of the benchmark model within the user-defined complexity constraint, SmartSelect identifies all feature selection methods that are computationally

feasible at this stage. These methods are then re-executed, each constrained to produce a feature subset whose dimensionality exactly matches the target specified by the user—whether defined as an absolute number or a percentage.

Each selected subset is then evaluated using a strong machine learning model—typically a boosting-based model—trained and tested using a train-test split. The performance of each subset is evaluated using an appropriate metric (e.g., RMSE for regression or F1 score for classification).

The subset that achieves the best score on the test set is selected as the final feature set output by SmartSelect.

[3]

4.6 Feature Selection Workflow Diagram

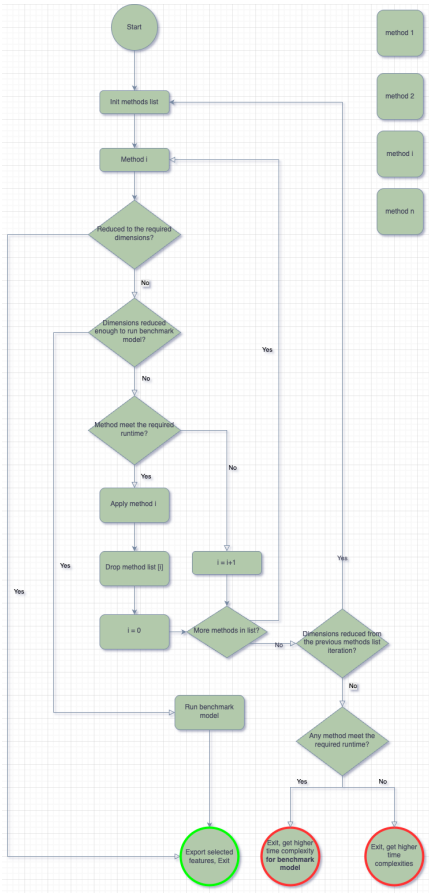

Figure 1: Illustrates the iterative feature selection process employed by SmartSelect, incorporating user-defined constraints and dynamic method evaluation.

5 Results

5.1 Evaluation Procedure

To assess the performance of the SmartSelect framework, we tested it across multiple tabular datasets encompassing both regression and classification problems. Each dataset was divided into training and test sets. For each experiment, we compared the performance of our method against a traditional feature selection technique, applied individually.

For each dataset, an XGBoost model was trained on the training data and evaluated on the test set. The performance metric depended on the task type: F1 score for classification and mean squared error (MSE) for regression.

5.2 Performance Comparison

Across all experiments, SmartSelect consistently outperformed individual feature selection methods in predictive performance on the test set [9], while maintaining reasonable runtime.

The most competitive baseline was observed on the SCANB dataset for a regression task with Lympho column as the label. Over 20 independent runs of SmartSelect, we obtained a mean squared error (MSE) of 0.00784, with a standard deviation of 0.00026 and a maximum value of 0.00833.

In comparison, the Variance Threshold method (It only ran once, since it's a deterministic method) resulted in an MSE of 0.01242.

5.3 Example Results

The figure below presents a representative result comparing SmartSelect with a baseline feature selection method on the SCANB dataset. Target: PAM50 — LumA vs. others.

All evaluation results, including this example and full performance distributions across all datasets, are provided in the Appendix.

SCANB data, Target = PAM50

- (3069 rows × 30,868 columns)
- Binary Classification (Target: PAM50 — LumA vs. others)
- Average running time of SmartSelect: 10 minutes

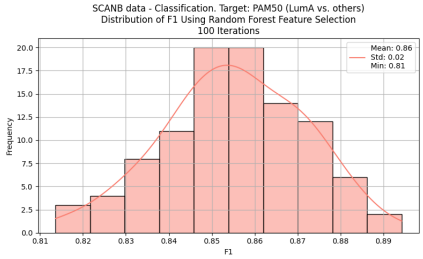

Figure 2: SCANB data, Target = PAM50

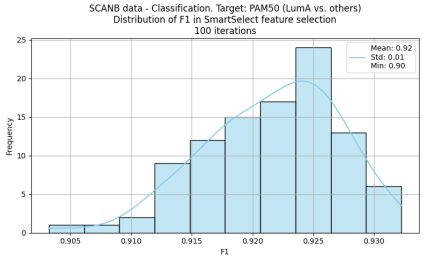

Figure 3: SCANB data, Target = PAM50

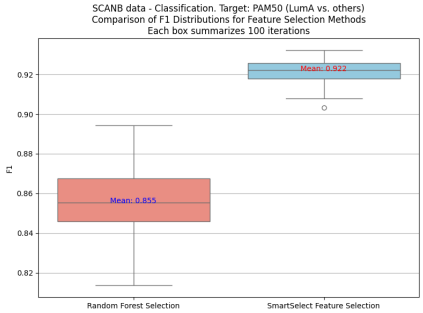

Figure 4: SCANB data, Target = PAM50

6 Discussion and Conclusion

This project introduced SmartSelect, an adaptive and modular framework for feature selection in high-dimensional tabular datasets. By integrating multiple selection techniques in a controlled iterative process,

SmartSelect balances accuracy and computational efficiency while requiring minimal user intervention. The system allows users to define constraints such as the number of desired features and complexity budgets, enabling flexible deployment across various domains.

Experimental results consistently demonstrated that SmartSelect outperforms traditional single-method approaches in both regression and classification tasks. The framework achieved lower mean squared error in regression and higher F1 scores in classification across multiple real-world datasets. These gains were especially notable in high-dimensional scenarios- where conventional methods either failed to scale or yielded suboptimal results [12] — **highlighting SmartSelect's core advantage in handling large feature spaces.**

Another notable benefit of SmartSelect is its increased result stability: across repeated runs, the standard deviation of performance metrics (such as MSE and F1) was significantly lower compared to other methods. This reduced variance suggests greater robustness to stochastic effects in the selection process (e.g., train-test splits or randomized algorithms), which is especially important in real-world pipelines where reproducibility and reliability are critical.

A key advantage of SmartSelect lies in its full automation and configurability. Without tuning internal hyperparameters, the system effectively navigates the trade-off between runtime and performance. The ability to adaptively filter features based on dynamic complexity constraints makes it especially suitable for real-world applications, where resource limitations are common.

A graphical example from a single run of the system is provided below, illustrating the step-by-step feature selection process over multiple iterations. While the time complexities of individual steps may appear high in the plot, it is important to note that the times are normalized relative to the size of the dataset at that point. As the number of features decreases throughout the process, subsequent selection methods run significantly faster, making the overall runtime more efficient than it may initially seem in the graph.

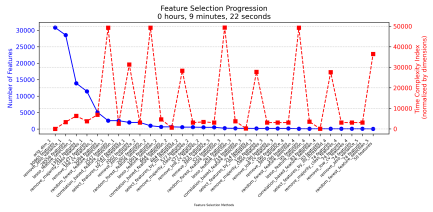

Figure 5: Feature Selection Progression over Iterations - example

Overall, SmartSelect presents a practical, efficient, and scalable solution to the feature selection problem, particularly in environments where high dimensionality and runtime sensitivity pose challenges.

7 Future Work

While SmartSelect demonstrates strong performance across a variety of supervised learning tasks, several important extensions remain for future research and development.

7.1 Support for Unsupervised Learning

Currently, SmartSelect is designed for supervised scenarios, where ground-truth labels allow for clear evaluation metrics and benchmarking. Extending the framework to support unsupervised learning tasks presents a

unique challenge, primarily due to the absence of a definitive ground truth to guide feature evaluation.

Future versions of the system would need to rely on intrinsic evaluation metrics (e.g., silhouette score, reconstruction error) or proxy objectives to assess feature quality in unsupervised contexts. Moreover, the internal selection mechanisms and benchmark model architecture would need to be adapted to operate in unsupervised pipelines.

7.2 Supervised Anomaly Detection

Another future task is adapting SmartSelect for supervised anomaly detection tasks. The current system is optimized for traditional supervised prediction, where average performance across the dataset (e.g., mean squared error or classification accuracy) is the target. In such settings, features that are highly correlated with each other are often considered redundant in classical prediction models and the system is encouraged to retain only one.

However, in supervised anomaly detection, this assumption may not hold. Correlated features might capture subtle, rare variations in the data that are critical for identifying anomalies [19]. For example, two features that are 99% correlated may still exhibit meaningful divergence in the tail of their distribution—precisely the region of interest in anomaly detection.

Supporting this use case would require modifying the selection strategy to preserve redundant but informative patterns, and potentially developing new evaluation criteria that reflect anomaly-specific utility rather than average-case accuracy.

7.3 Multiclass Classification Support

Currently, SmartSelect works in binary classification and regression settings. Extending the framework to better support multiclass classification would involve adjustments in both evaluation metrics (e.g., macro/micro-averaged F1 scores) and in the benchmark model's architecture and selection thresholds.

7.4 Parallel Execution of Selection Methods

The current implementation evaluates feature selection methods sequentially. However, in the benchmark stage—multiple methods could be evaluated in parallel without exceeding runtime constraints. Incorporating parallel execution for compatible methods in the benchmark phase could significantly reduce overall processing time.

7.5 User Control Over Method Parameters and Regularization

In its current form, SmartSelect uses fixed thresholds for method behavior, such as the minimum number of features retained. Providing the user with greater control over parameters and regularization constraints, such as specifying the minimum or maximum percentage of features each method is allowed to retain, would make the system more adaptable to different use cases and domain-specific requirements.

7.6 Support for Non-Numerical Features

SmartSelect currently assumes that all input features have been preprocessed into numerical form. However, the transformation of

categorical features can have a major impact on the relevance and behavior of feature selection methods—especially in contexts like anomaly detection, where rare values can carry critical information.

Future versions of the system could incorporate preprocessing strategies such as adaptive one-hot encoding, guided by feature cardinality or class imbalance, as an integral part of the selection process rather than a preprocessing step external to the system.

References

[1] Subhajit Bhattacharyya. Efs-mi: An ensemble feature selection method for classification. *Complex & Intelligent Systems*, 3(2):105–118, 2017.

[2] Verónica Bolón-Canedo, Natalia Sánchez-Maroño, and Amparo Alonso-Betanzos. Recent advances and emerging challenges of feature selection in the context of big data. *Knowledge-Based Systems*, 86:33–45, 2015.

[3] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016.

[4] Federico Colombelli, Tomasz W Kowalski, and Manuel Recamonde-Mendoza. A hybrid ensemble feature selection design for candidate biomarkers discovery from transcriptome profiles. *arXiv preprint arXiv:2108.00290*, 2021.

[5] Datacamp. The curse of dimensionality in machine learning. `https://www.datacamp.com/blog/curse-of-dimensionality-machine-learning`, 2024. Accessed: 2025-06-18.

[6] Matthias Feurer, Aaron Klein, Matthias Eggensperger, Jost Tobias Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 28, 2015.

[7] Salvador García, Santiago Ramírez-Gallego, Julián Luengo, José M Benítez, and Francisco Herrera. *Big data preprocessing: Enabling smart data*. Springer, 2020.

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

[9] Isabelle Guyon, Stephen Gunn, Mehrdad Nikravesh, and Lofti A Zadeh. *Feature extraction: Foundations and applications*. Springer, 2006.

[10] Ron Kohavi and George H John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

[11] Miron B Kursa and Witold R Rudnicki. Feature selection with the boruta package. *Journal of Statistical Software*, 36(11):1–13, 2010.

[12] Jundong Li, Ke Cheng, Suhang Wang, Fred Morstatter, Ryan P Trevino, Jian Tang, and Huan Liu. Feature selection: A data perspective. *ACM Computing Surveys (CSUR)*, 50(6):94, 2017.

[13] Nicolai Meinshausen and Peter Bühlmann. Stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(4):417–473, 2010.

[14] Uwe Neumann, Nicolas Genze, and Dominik Heider. Efs: An ensemble feature selection tool implemented as r-package and web-application. *BioData Mining*, 10:21, 2017.

[15] Yvan Saeys, Iñaki Inza, and Pedro Larrañaga. A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23(19):2507–2517, 2008.

[16] Wei Shang, Hao Huang, Yiming Zhu, Yan Lin, and Ying Qu. Tabular data feature selection: A practical perspective. *IEEE Access*, 9:124314–124324, 2021.

[17] Jian Tang, Salem Alelyani, and Huan Liu. Feature selection for classification: A review. In *Data classification: Algorithms and applications*, pages 37–64. CRC Press, 2014.

[18] Xiaowei Zhang and Inge Jonassen. Efsis: Ensemble feature selection integrating stability. *arXiv preprint arXiv:1811.07939*, 2018.

[19] Arthur Zimek, Erich Schubert, and Hans-Peter Kriegel. A survey on unsupervised outlier detection in high-dimensional numerical data. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 5(5):363–387, 2012.

8 Appendix

8.1 Full Results Across All Five Datasets

This appendix presents the complete results for all five datasets evaluated in our experiments, including the dataset shown in the main paper.

SCANB data, Target = PAM50

- (3069 rows × 30,868 columns)
- Binary Classification (Target: PAM50 — LumA vs. others)
- Average running time of SmartSelect: 10 minutes

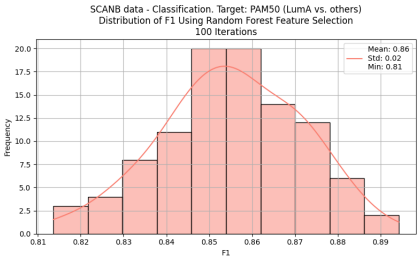

Figure 6: SCANB data, Target = PAM50

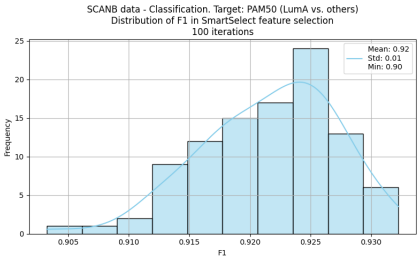

Figure 7: SCANB data, Target = PAM50

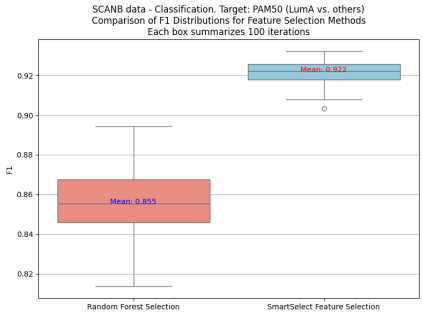

Figure 8: SCANB data, Target = PAM50

SCANB data, Target = ER

- (3069 rows × 30,868 columns)
- Binary classification task (Target: ER)
- Average running time of SmartSelect: 13 minutes

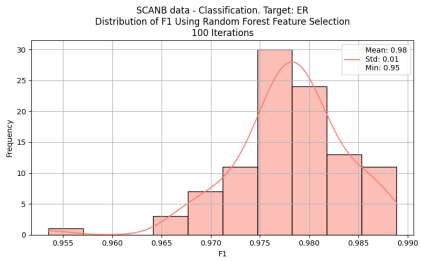

Figure 9: SCANB data, Target = ER

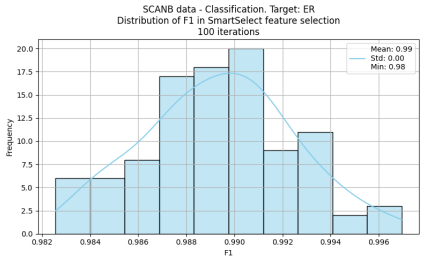

Figure 10: SCANB data, Target = ER

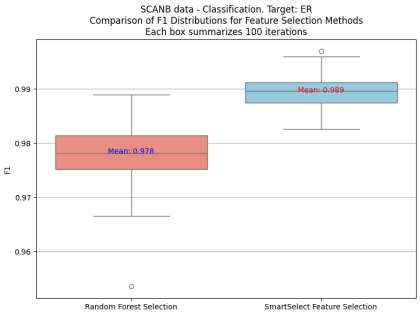

Figure 11: SCANB data, Target = ER

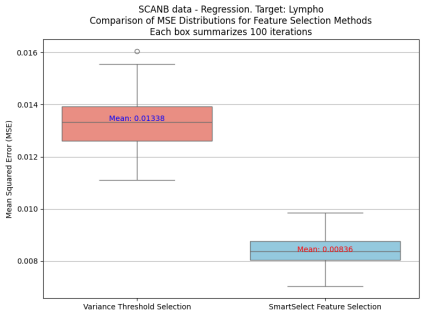

Figure 14: SCANB data, Target = Lympho

SCANB data, Target = Lympho

- (3069 rows × 30,868 columns)
- Regression task (Target: Lympho)
- Average running time of SmartSelect: 20 minutes

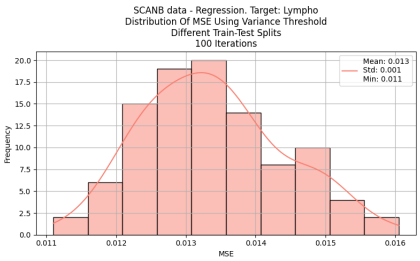

Figure 12: SCANB data, Target = Lympho

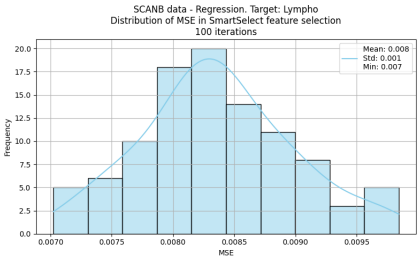

Figure 13: SCANB data, Target = Lympho

MNIST 784

- (14,780 rows × 784 columns)
- Binary Classification
- Average running time of SmartSelect: 0.5 minutes
- **Note: Given the relatively low feature dimensionality of this dataset, performance differences between methods were smaller. The advantages of SmartSelect are more pronounced in high-dimensional settings.**

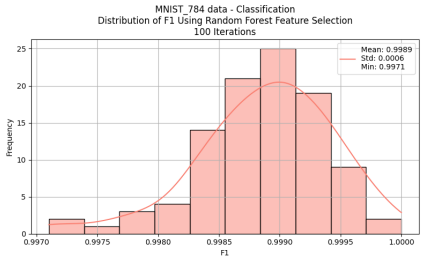

Figure 15: MNIST 784

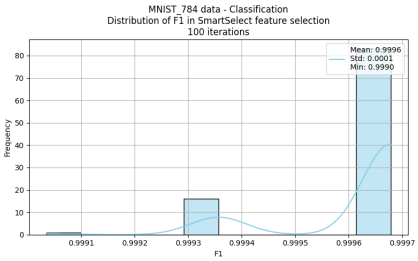

Figure 16: MNIST 784

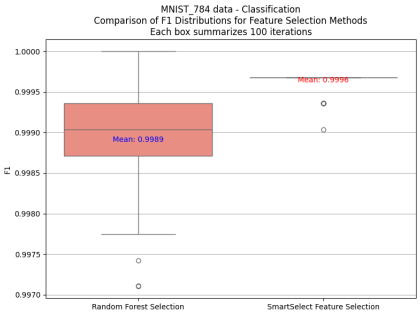

Figure 17: MNIST 784

GSE184773 data

- (24 rows × 24,057 columns)
- Binary Classification
- Average running time of SmartSelect: 5 seconds

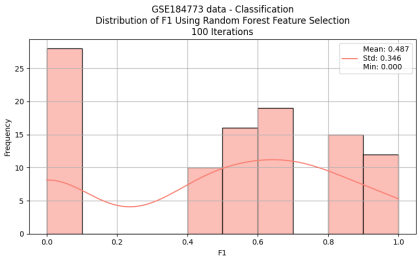

Figure 18: GSE184773 data

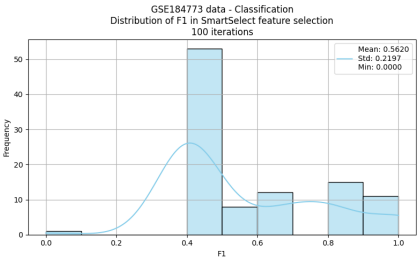

Figure 19: GSE184773 data

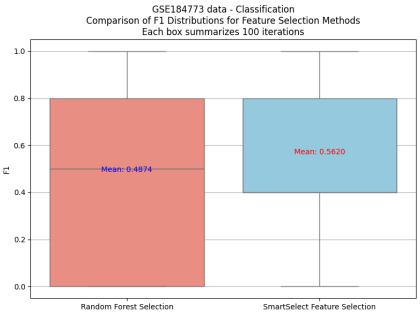

Figure 20: GSE184773 data