Technical Report - Assignment 2

1. Introduction

The goal of assignment 2, "Metaheuristics," is to explore and deepen our understanding of metaheuristics by applying them with feature selection, aiming to uncover their adaptability and potential in optimizing machine learning models. Through practical implementation and comparison between different models and methods, I seek to discover the strategic nuances between the performance and the computational complexity of different feature selection methods.

2. Feature Selection and Training Model Selection

Before delving into the implementation and outcomes of our approach, it's crucial to provide a concise overview of the feature selection method and machine learning models chosen for this study. Our methodology integrates traditional techniques with advanced metaheuristic algorithms to investigate the impact of feature selection on model performance. This approach not only aligns with the goal of exploring metaheuristics' applicability but also provides a benchmark for evaluating their effectiveness in comparison to conventional methods.

Feature Selection Method Overview

Our exploration begins with Scikit-learn's built-in feature selection functionalities, which serve as a traditional benchmark. These methods, designed to reduce input variable dimensionality, help retain only the most relevant features for the model's performance, thus setting a standard against which the performance of metaheuristic approaches can be assessed. Besides the Sklearn built-in feature selection, Genetic Algorithms and Simulated Annealing were also used for metrics and performance comparison.

- **Genetic Algorithms** (GAs), which simulate the process of natural evolution, applying mechanisms such as selection, crossover, and mutation to evolve a population of feature subsets towards optimality.

- **Simulated Annealing** (SA), a probabilistic technique inspired by the annealing process in metallurgy, known for its ability to approximate the global optimum of a given function by exploring the search space through controlled configurations.

These metaheuristic algorithms are chosen for their distinct approaches to search space exploration and optimization, offering us a rich comparative analysis of traditional versus advanced metaheuristics feature selection methods.

Machine Learning Models Overview

The study employs two distinct machine learning models, both serving dual roles as predictive tools and evaluators of feature fitness within the feature selection process:

- **Random Forest** (RF): The RF model is a collection of decision trees that work together to improve predictions. Each tree in the forest considers a random subset of features when making decisions, which makes RF particularly adept at handling datasets with a high dimensionality and a mix of feature types. Moreover, RF's efficiency shines in computational speed; it is significantly faster compared to more complex models, such as Neural Networks.
- **Neural Network** (NN): NN is a more sophisticated model, excels at capturing and modeling complex, nonlinear relationships hidden within the data through its deep learning architecture. NN is chosen here specifically after considering the high complexity and volume of the independent variable. In our study, the Neural Network not only surpasses the Random Forest in terms of performance but also offers deeper insights into the efficacy of selected features. However, this comes at a cost: the Neural Network requires significantly more time to train.

Through this nuanced exploration, we provide a comprehensive view of how different models can influence the feature selection process, shaping our approach to optimizing machine learning model performance in a way that considers both accuracy and efficiency. Both models are encapsulated within their respective classes using object-oriented programming principles, ensuring reusability and efficiency in the feature selection process.

DataSet Overview

For simplicity and to provide a more nuanced dataset, I assembled all 11 SE-process[1-11] datasets into one large dataset. This aggregated SE-process is used in the whole process.

3. Scikit-learn Built-in FS with metrics

Utilizing Scikit-learn's built-in feature selection functionalities, while exploring this convenient way to optimize training with feature selection, an investigation into the optimal number of features for this dataset was conducted. The findings, illustrated in Figure 1 to Figure 5, highlight a divergence in optimal feature counts between the Random Forest and Neural Network models. For the Random Forest model, selecting a subset of features, rather than utilizing all 90+ features, consistently yielded higher F1 scores, suggesting an efficiency in feature reduction. This phenomenon raises intriguing questions about the model's ability to generalize from reduced feature sets, potentially indicating overfitting with larger numbers of features. See Figure 1 for direct comparison illustration.

Conversely, the Neural Network model exhibited a different trend, with F1 scores for feature sets below 40 generally falling below those achieved using the full feature set. However, a notable improvement was observed when exceeding 50 features, marked by a significant 5% increase in F1 score. This suggests a threshold effect, where a minimum feature set size is necessary for the Neural Network to effectively capture the underlying patterns in the data. See Figure 1 for direct comparison illustration. In terms of performance, the Neural Network model demonstrated superior outcomes compared to the Random Forest, with an average improvement of 5% in F1 score. Specifically, the optimal feature selection for the Random Forest model was identified at 64 features with an F1 score of 0.84, while for the Neural Network, the peak performance was at 44 features, achieving an F1 score of 0.884. This comparison not only underscores the impact of feature selection on model efficacy but also highlights the variable sensitivity of different models to the dimensionality of the input data. See Figure 1 for direct comparison illustration.

Meanwhile, we observed notable differences in training speed between the RF and NN models. Specifically, with Scikit-learn's built-in feature selection, the RF model can be trained in about 0.5 seconds on average, significantly faster than the NN model, which takes approximately 2.5

seconds for each training session. See Figure 2 for comparison illustration. This disparity underscores the RF model's computational efficiency. However, the superior performance metrics of the NN, particularly in handling complex data patterns with optimized feature sets, highlight a trade-off between training speed and predictive accuracy.

In terms of precision, recall, and accuracy, in all three fields, the NN model is consistently outperforming the RF model. The phenomenon where precision exceeds recall for both models suggests they are more successful at correctly predicting positive instances when they do so, but at the expense of missing other positive cases. This indicates a cautious approach to positive predictions, prioritizing the avoidance of false positives over capturing all positives. The discrepancy where accuracy falls below the F1 score highlights that despite a balanced trade-off between precision and recall (as reflected by the F1), the overall correctness of predictions (accuracy) is lower. See direct comparison of precision, recall, and accuracy in Figure 3,4, and 5.

4. Genetic Algorithm and its performance

Our Genetic Algorithm (GA) for feature selection initializes with random binary vectors representing feature subsets from the dataset. Through iterative processes of selection, crossover, and mutation across generations, the GA refines these subsets. Fitness is evaluated by training a chosen model (Random Forest or Neural Network) and assessing F1 score, steering the algorithm towards the most effective feature combinations for model performance optimization.

The GA's efficiency is significantly impacted by the computational intensity of evaluating fitness for each generation, especially when using Neural Networks due to their complex architecture and longer training times. With NN models, finishing one generation of GA will take more than 4 minutes. As a result, completing the GA feature selection process with NN becomes impractical within reasonable time frames. This highlights a critical challenge in applying GAs to feature selection with computationally intensive models, where the balance between exploration and performance evaluation becomes a key bottleneck. Consequently, this limitation has prevented the completion of NN training using GA in my study.

In contrast to the challenges faced with Neural Networks, the efficiency and faster computation times of Random Forest (RF) models enabled the successful completion of the Genetic

Algorithm (GA) feature selection process over 100+ generations. This achievement showcases the practical viability of using GAs with computationally less intensive models like RF for feature optimization. The optimization process reached its peak at round 80, where the GA identified an optimal subset of 53 features that achieved an F1 score of 0.875. This outcome not only illustrates the GA's ability to effectively navigate the search space and refine feature subsets for enhanced model performance but also emphasizes the importance of model selection in GA-based feature selection strategies.

5. Simulated Annealing and its performance

Simulated Annealing (SA) was successfully applied to both Random Forest and Neural Network models for feature selection, owing to its efficient strategy of comparing only neighboring feature selections in each iteration, rather than the entire population (in our case of GA: 50). This neighbor-focused approach significantly reduces computational overhead, making it practical even for the slower NN model. SA's efficiency and adaptability in handling different computational demands demonstrate its effectiveness in optimizing feature sets across various machine learning models. See Figure 7, 8, 9.

The results of applying Simulated Annealing (SA) for feature selection, using Random Forest (RF) to calculate fitness, showed a consistent fitness score of 0.81 across iterations ranging from 20 to 1000. This plateau suggests a convergence of the SA process towards a stable feature set that maximizes fitness for the RF model, indicating that additional iterations do not necessarily contribute to further optimization. The reason behind this consistent fitness level warrants deeper investigation to understand the dynamics of SA's exploration and exploitation balance in relation to the RF model's characteristics.

The performance of the Neural Network when utilizing Simulated Annealing for feature selection strikingly illustrates the model's adeptness at working with highly optimized feature sets. Achieving a peak fitness of 0.92, both in the early stages of SA at 30 iterations and later at 1000 iterations, with only 30 selected features, underscores a significant aspect of the NN's operational dynamics. This consistency in achieving high performance with a minimal set of features suggests that the NN model rapidly identifies and capitalizes on the most predictive attributes, achieving its best performance without the need for extensive iteration.

Achieving a fitness score of 0.92 with only 30 features at both 30 and 1000 iterations in Simulated Annealing, when using NN for fitness evaluation, showcases SA's effectiveness in feature selection. (Figure 7 and 9) This result demonstrates SA's capacity to identify an optimal set of features early in the process, and maintain this optimum across many iterations. It highlights the SA algorithm's efficiency in finding and retaining a feature subset that maximizes the NN model's performance, without the need for extensive iteration. This points to a strength of SA in achieving rapid convergence to a high-performing feature set, suggesting that the process of exploring and refining feature selections can be both swift and stable, even when integrated with computationally intensive models like NNs for fitness calculations.

6. Comparison and Key Takeaway

In comparing the effectiveness of Scikit-learn's built-in feature selection, Genetic Algorithms (GA), and Simulated Annealing (SA) across different computational models, distinct advantages and limitations emerge. Scikit-learn's method offers speed and simplicity but lacks automation in optimizing the number of features for the best F1 score, requiring manual adjustments. For fitness evaluations using Random Forest (RF), GAs outperform SA in achieving higher F1 scores, indicating a superior capability for feature optimization. However, the scenario shifts when applying these metaheuristics with Neural Networks (NN); the computational demand of GAs becomes a significant drawback, making them less practical. In contrast, SA, when paired with NN, excels in identifying optimal feature sets with the highest F1 scores, albeit with the caveat that NN's inherent superiority in modeling may influence the comparative results.

Conclusively, the choice of feature selection method hinges on dataset characteristics and the specific needs of the model in question. For smaller datasets or when a predetermined number of features is targeted, Scikit-learn's built-in functions provide a quick and efficient solution. In contrast, for larger datasets without a predefined feature count, custom feature selection through GAs or SA offers a more tailored approach. GAs are recommended for use with models that allow for rapid fitness evaluation, balancing training time against result quality. For achieving the highest quality feature selection with relatively shorter training durations, especially with complex models, SA combined with NN is advised, underscoring the importance of aligning the feature selection strategy with the model's computational profile and the dataset's scale.

7. Appendix

Figure 1: SK FS with RF and NN

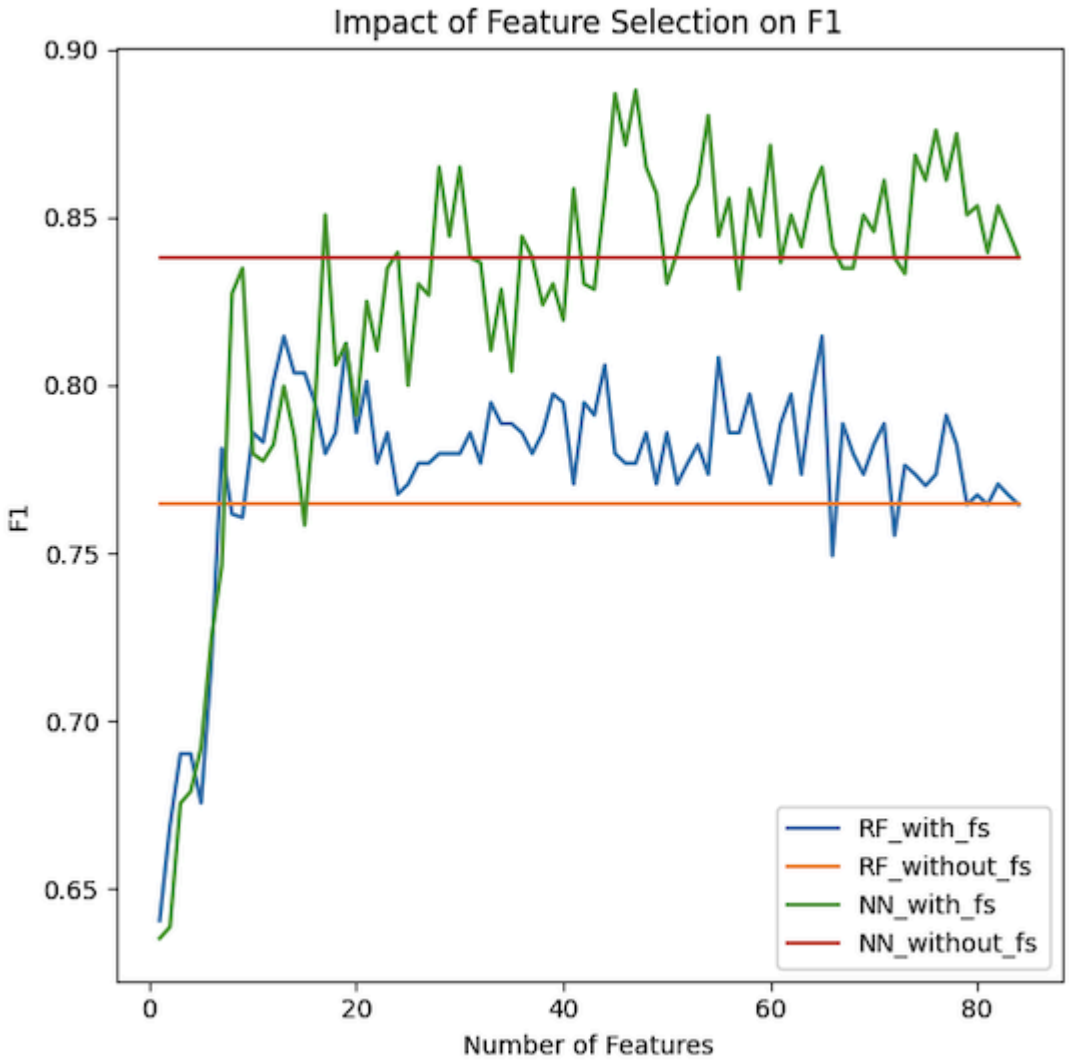

Figure 2: SK FS with RF and NN

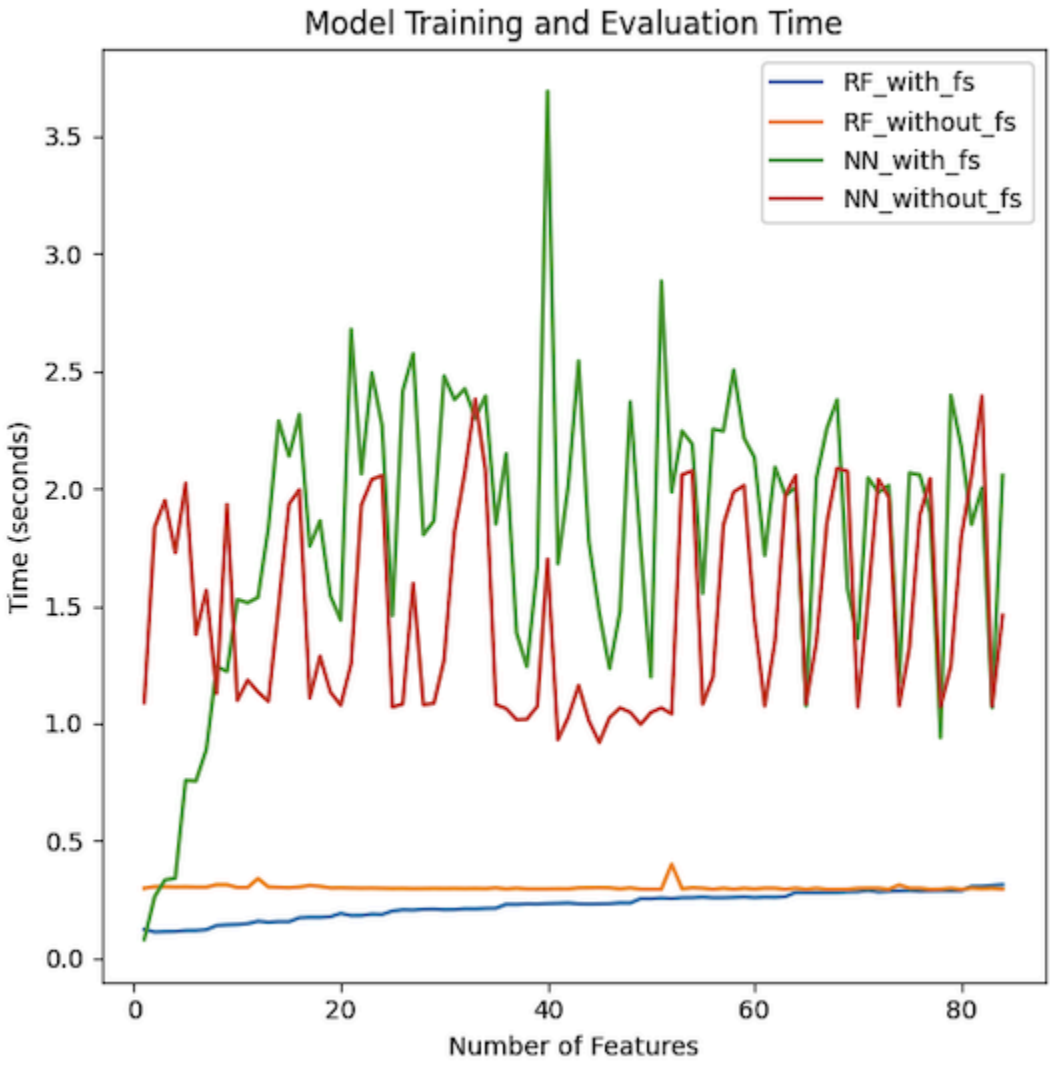

Figure 3: SK FS with RF and NN

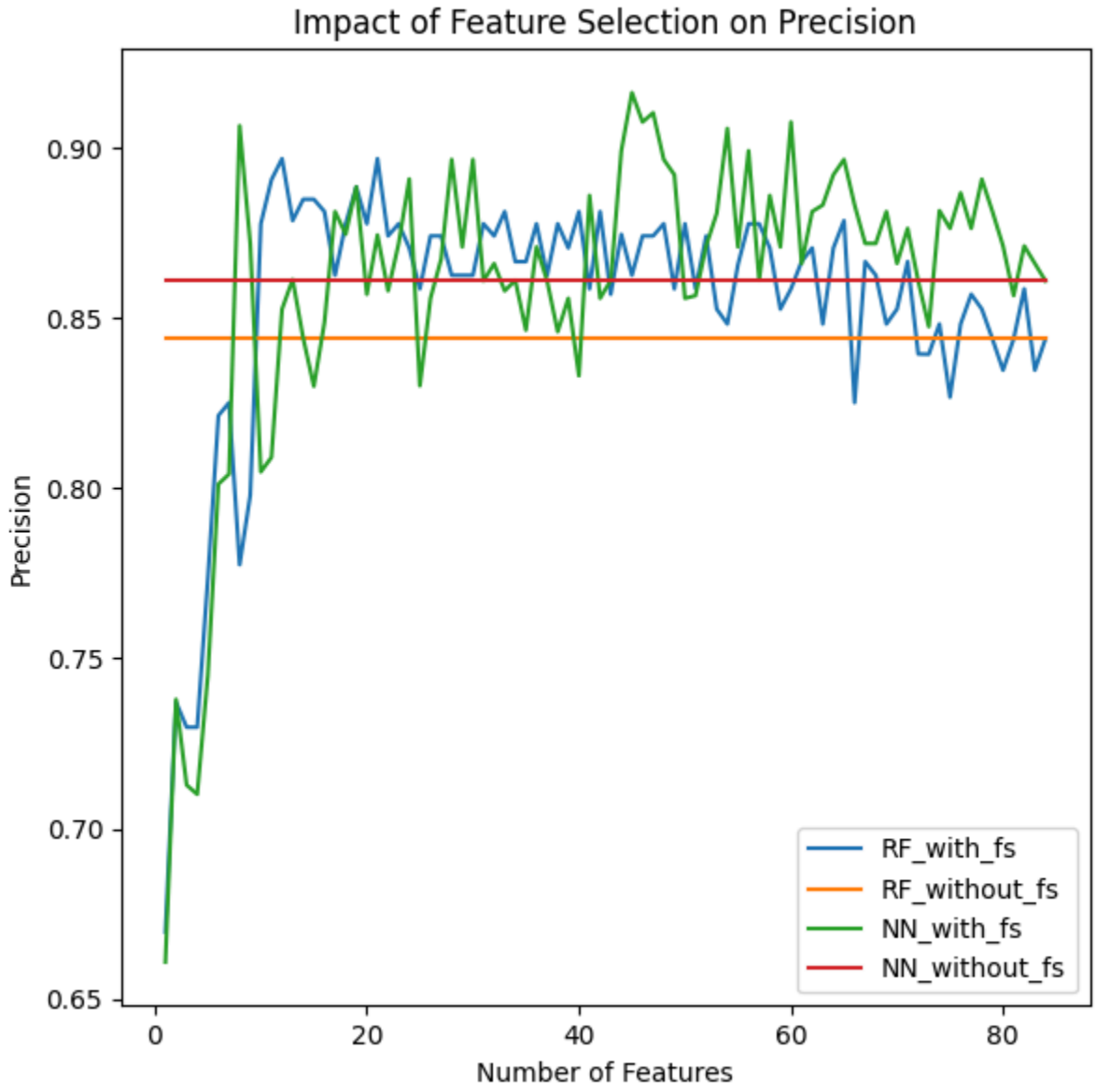

Figure 4: SK FS with RF and NN

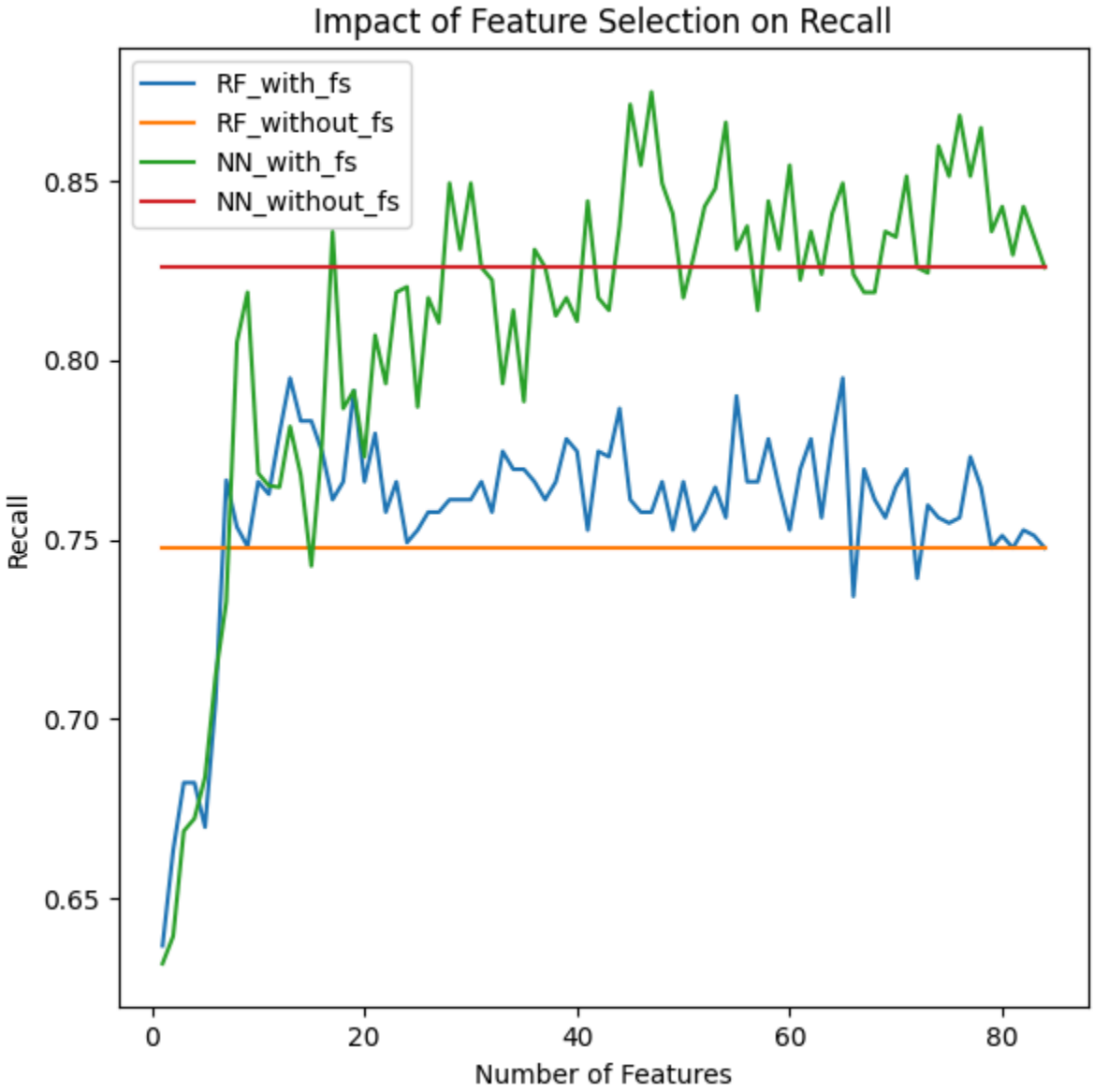

Figure 5: SK FS with RF and NN

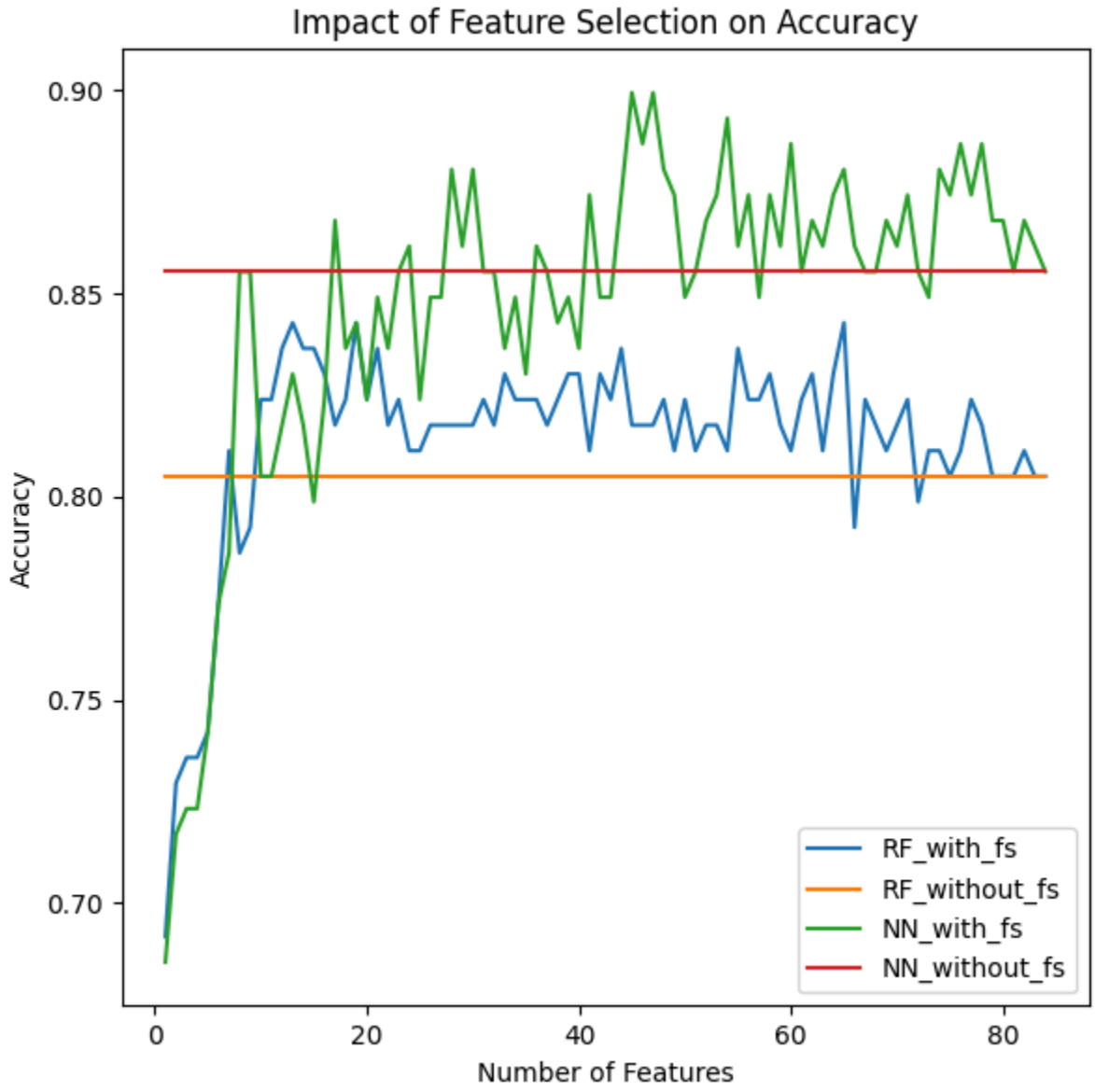

Figure 6: GA with RF

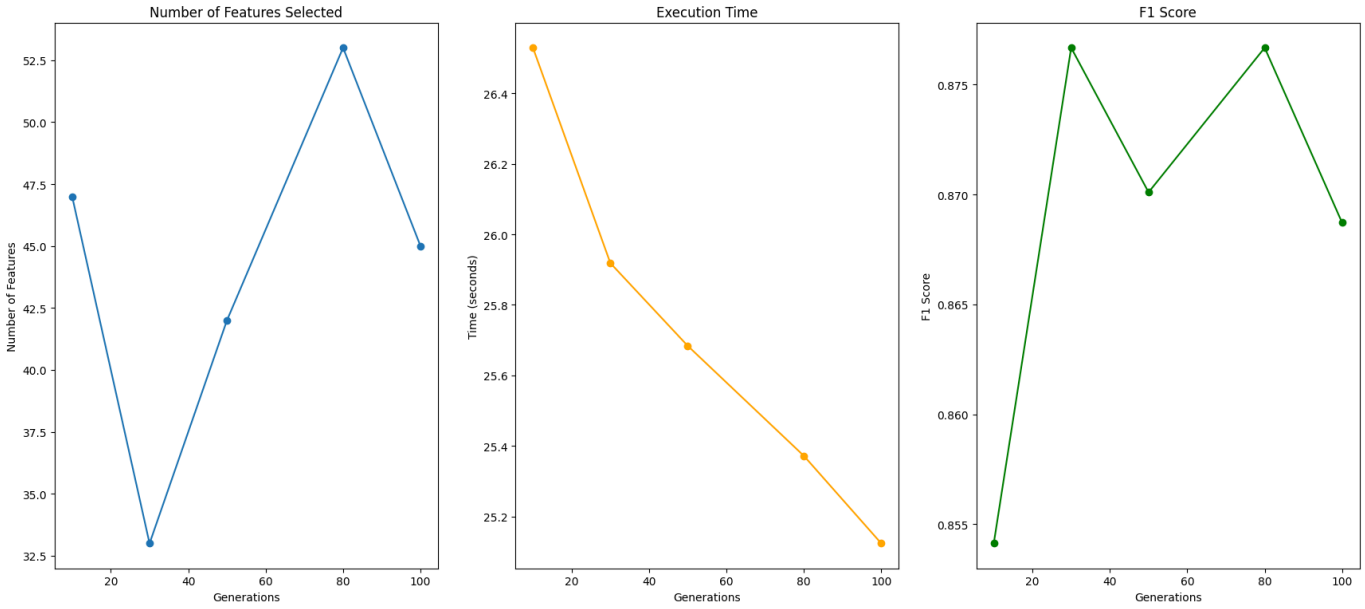

Figure 7: SA With RF and NN

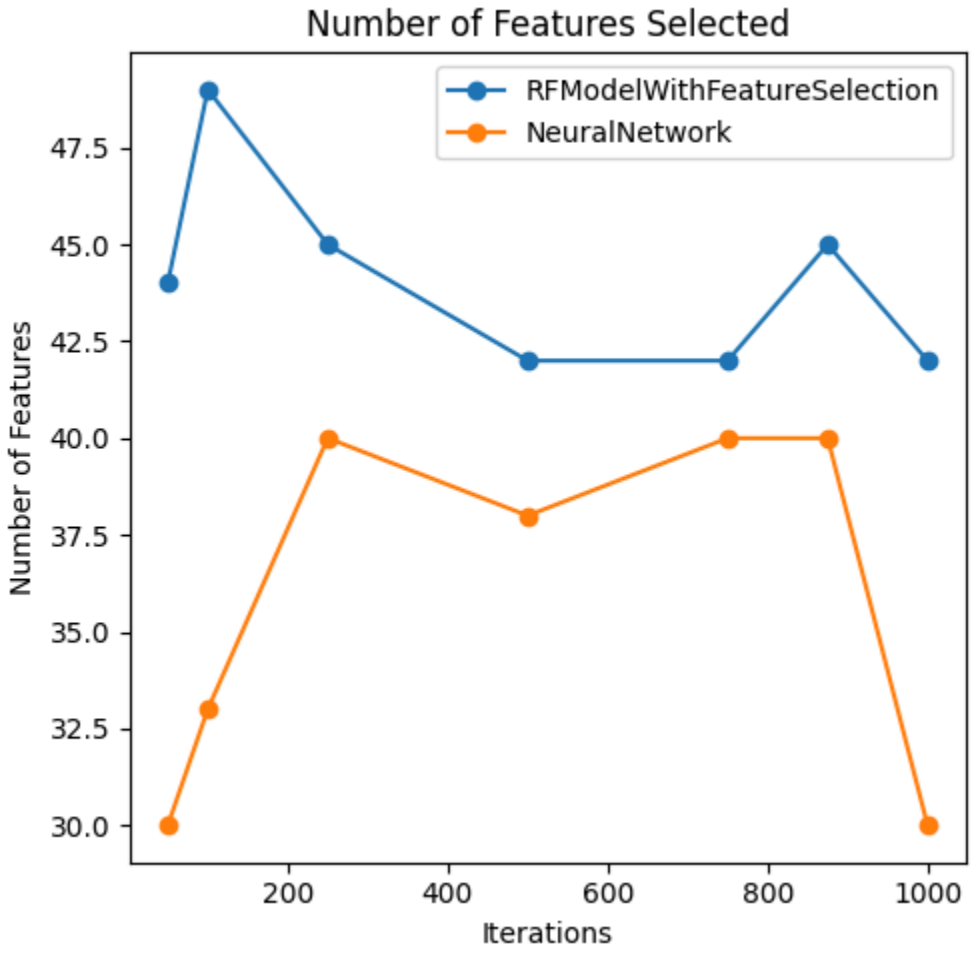

Figure 8: SA With RF and NN

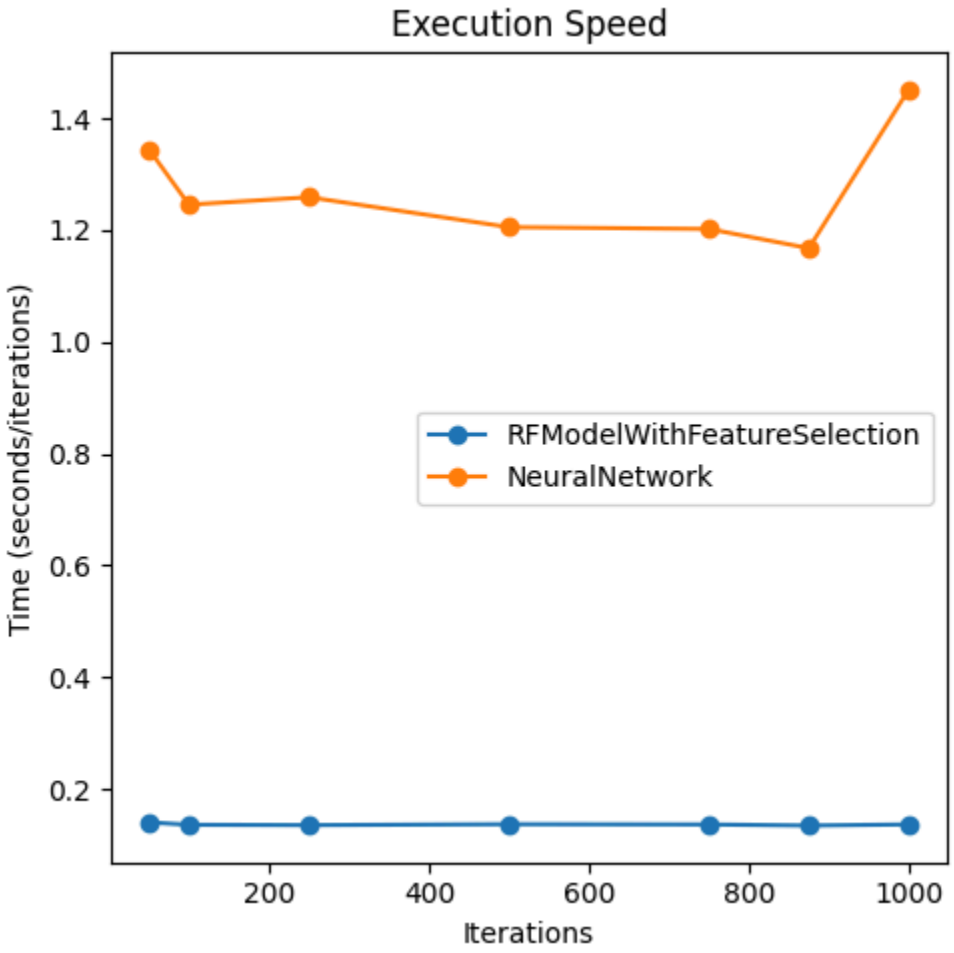

Figure 9: SA With RF and NN