# A Dynamic Semi-Random Approach to Decision Tree Forests

Nikhil Alladi, Jacob Dipasupil, Petr Kisselev

# 1 Abstract

The Random Forest algorithm [7] is a versatile and often effective machine learning algorithm, and in this paper we introduce Dynamic Semi-Random Forest (DSRF) to improve upon Random Forest's shortcomings. Random Forest's stochastic nature can result in major variation between runs, making it difficult to interpret. This stochastic nature comes from both bootstrapping and random feature selection in the tree creation process. Rather than using random feature selection, DSRF instead uses a weighted random feature selection to create a forest. By using a weighted random feature selection, the random forest will weed out any uncorrelated features, place a heavier value on highly correlated features, but will keep random forest's resistance to overfitting because of the randomness. Features are weighted based on performance measures of decision trees created using said features in order to generate forests valuing highly correlated features, pruning uncorrelated features along the way. Due to the additional performance demands to the already demanding random forest algorithm, we focus on high dimensional datasets with many noisy features. In practice, real-world data is very noisy and it is not always clear which features are uncorrelated to the class for preprocessing. Our model works to limit the influence of low-correlated features which may not always be obvious to us. DSRF produces noticeable improvement upon Random Forest in terms of accuracy.

# 2 Introduction

Due to the stochastic nature of the Random Forests (RFs), there is often high run-to-run variance. Although this can be offset with a larger sample size, increasing the number of samples and generated trees leads to unnecessary computational bloat for little improvement. Additionally, obtaining or creating more data can be time-consuming and expensive, which calls for an approach that involves taking full advantage of the data we have. While RFs generally provide an overall performance uplift in comparison to single-tree models, we believe performance suffers as a result of the uniformly distributed random selection of model features. This can especially be a problem for high dimensionality datasets, or any dataset with features that have low correlation to the class. The uniform random attribute selection will not accurately represent the relative importance of each attribute for predicting the class. For these reasons, we intend to improve the RF model by taking a dynamic approach to feature selection which builds upon itself while building the model. To evaluate our approach, we will test a low sample, high dimensionality dataset with a traditional RF, decision trees, and our DSRF, and compare how they perform. We will also vary the train-test-split distribution in order to observe how our model performance changes with varying amounts of data.

# 3 Related Work

One implementation, Dynamic Random Forests, similarly focuses on the fact that traditional RF models create decision trees independent from one another. Dynamic Random Forests works to have the tree making process dependent on the ensemble being created by resampling the training data and reweight the random feature selection [2]. Another implementation also focuses on feature extraction by combining random forests and Ant Colony Optimization and splitting up the algorithm into two phases: one for feature selection and one for creating an isolated forest for classification [4]. Other implementations use accuracy and correlation measurements of decision trees to prune decision trees [5], recursive feature selection

[6], or even infinite feature selection for weighted feature selection in order to improve upon the random forest algorithm [1]. Many of these implementations show improvements by tapping into the assumption of random forest that all features have equal correlation to the class, when that is rarely the case. We work to improve upon the feature selection process as well and adopt the two phase system–with the initial phase for choosing features and the second phase for creating the forest–as well as weighted feature selection based on performance metrics of decision trees.

# 4 Dataset and Features

Internet Advertisements comprises 3,279 instances and 1,558 features representing possible advertisement images on the Internet [3]. Features include information on phrases occurring in the URL, the geometry of the image, the anchor text, and more. The objective is to predict whether an image is an advertisement (“ad”) or not (“nonad”). As for the class distribution, there are 2821 nomads and 458 ads. Internet Advertisements has three continuous variables for geometric information about the images: height, width, and aspect ratio. Because height and width are directly related to aspect ratio, we chose to remove the height and width features, and discretize aspect ratio. Aspect ratio was discretized into three bins of varying ranges: ratio less than or equal to 4, between 4 and 10, and greater than 10. Afterwards, we replaced missing values in aspect ratio with the mode. We tested our performance with an 80/20 test split. Among the 80%, we also tested three different splits for our feature selection validation: 30/70, 20/80, and 10/90.

# 5 Methods

## 5.1 Random Forest

The random forest algorithm is an ensemble learning algorithm that aggregates predictions from multiple decision trees. Random forest uses bagging and random feature selection to create a forest of decision trees–much less susceptible to overfitting compared to standalone decision trees. The random forests’ decision trees are created with bootstrapped samples and randomly selected subsets of features and ultimately aggregated to make a prediction.

Deviating from RF’s random feature selection approach, the DSRF will comprise two phases: the dynamic feature selection phase, and generating the forest.

During the first phase, the algorithm will start like a typical random forest–randomly selecting a sample of root(n) attributes from the total attribute list of n attributes to create a decision tree on the validation set. Then, the decision tree will be evaluated on accuracy with a validation set. For each attribute in the tree, the decision tree’s overall accuracy percentage [scaled to 0-1, then raised to the 5th power to help differentiate higher scoring attributes] will be stored in a table. To allow each attribute to be seen a reasonable amount of times, this process will repeat sqrt(N) times, and the external table will be updated with the adjusted average of each feature across all the created decision trees. By doing this, the best features will “float to the top” due to repeated high performance, while poor features will lose ranking, the more epochs are trained.

Following this, choosing the k lowest adjusted accuracy features, move to the training set to create a forest of decision trees. Rather than the traditional random attribute selection, the

algorithm will use a semi-random weighted feature selection. Each feature's adjusted accuracy will be scaled to a proportion of the total adjusted accuracy of the remaining k features and used for a weighted random feature selection when generating the forest. In the rare case that a feature has no probability associated with it yet due to not being hit by the earlier passes of feature selection, it will be set to have average probability to avoid skewing the feature selection.The rest of the algorithm will be the same as random forest and will classify with the aggregate prediction. A pseudo code representation of the algorithm can be seen in Figure 1.

```
Define performance table
Repeat N times:
      Sample root(n) attributes without replacement
      Create decision tree with attributes
      Evaluate decision tree accuracy on validation set
      For each attribute in decision tree:
            Add accuracy^5 to performance table, average

Construct random forest from k best attributes in performance table
```

*Figure 1: Pseudocode of DSRF algorithm*

# 6 Experiments

## 6.1 Procedures

When choosing hyperparameters for our subforests that the DSRF constructs, we experimented with a variety of possible hyperparameters to see what would work best for this dataset. For the amount of attributes, we used the standard amount of attributes for random forest choosing (specifically, sqrt(n) for a dataset with n attributes). For the others (how many trees we would create per epoch, how deep those trees would go) we experimented with different values and evaluated them with pure accuracy. This eventually settled to a batch size of 30 trees per epoch (this showed a constant increase per batch size, but we chose 30 for efficiency reasons) and a depth of 15. Our search for optimal hyperparameters was limited by the computational complexity of the algorithm and high runtime. As a result, we had to limit our search to a smaller range of values, being forced to in some way prioritise computational speed over accuracy to yield timely results.

## 6.2 Results

To evaluate our model, we used three metrics: accuracy, precision, and recall. Given that our dataset is about identifying ads, precision and recall are vitally important to both the user experience.

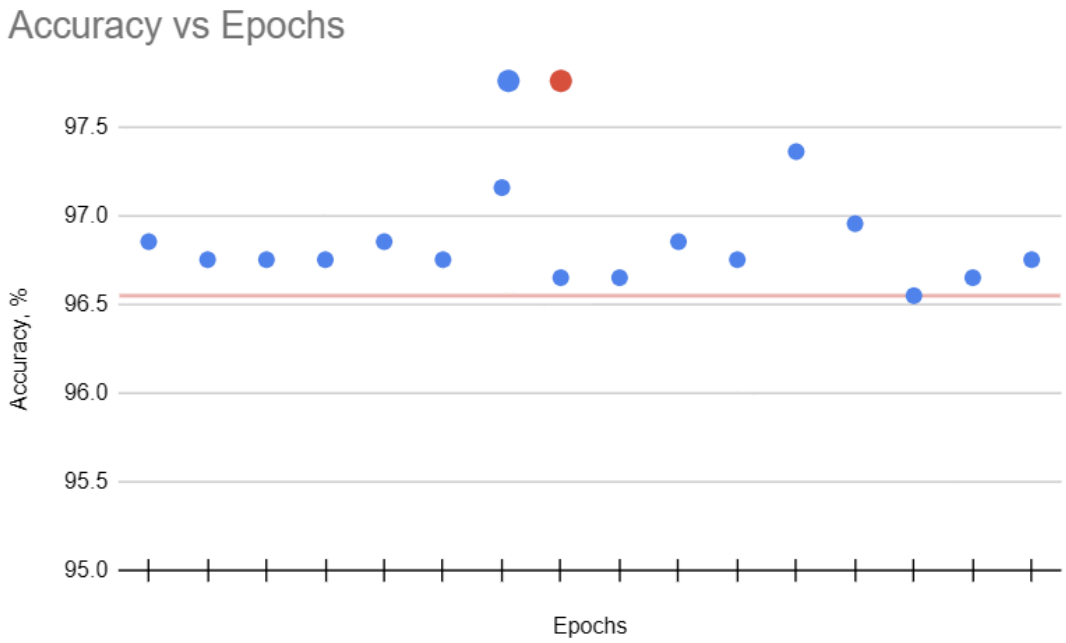


*Figure 2: Graph of Accuracy vs Epoch*

Accuracy held relatively consistent throughout epoch changes (note the scale of the Y axis starting at 95%), but peaked at around epoch 11, as seen in Figure 2. The red line in the figure is the accuracy of our control random forest, to compare the effectiveness of the two solutions alongside each other.

The confusion matrix at epoch 11, which we empirically picked as the random forest to use due to its max accuracy, is displayed in Table 1.

| | | Predicted | |
|---|---|---|---|
| Actual | | 121 | 17 |
| | | 15 | 832 |

*Table 1: Graph of Accuracy vs Epoch*

The top left of Table 1 represents a predicted ad that ended up being an ad. The precision, tp / (tp + fp), is 121 / (121 + 17) = **0.877**, and the recall, tp / (tp + fn), is 121 / (121 + 15) = **0.89**. As we can see, recall is very high, which means that most of the necessary ads were flagged, which means that the user is much less likely to come across an ad if this was to be used in an application such as ad blocking / removal. It can be said that the recall metric is more important than precision in this case.

## 6.3 Discussion

An immediate observation is that our model does not perform significantly better than the standard random forest on this dataset. There are a couple of reasons for this: namely, the already high accuracy of the standard random forest model, and model imperfections. The performance of a normal random forest model on this dataset already approaches 97%- that is a very high bar to set and incredibly hard to crack. The DSRF seeing a near-percent improvement on that could be considered a very impressive achievement in the sense that there only really is 3.5% to improve, so seeing that improvement is already nearly 25% of the possible improvement there is to be had. Besides that, the model not improving much over epochs signifies that our model isn't working exactly as we intended it to. We believe that a possible point of failure was that we would weight an attribute's pick chance based on its success in algorithms. While this was the intention of the algorithm, we did it every pick as opposed to in moderation, which would lead to immediately good attributes always being picked and not letting other attributes shine; essentially, it found a local maximum. While we attempted to remedy this by giving unpicked attributes a baseline probability to appear, it was likely not enough. Furthermore, our attempt to remedy this issue through the use of batching to delay updating the weighting values until a certain point still did not wholly fix the issue. A possible remedy might be to store average values instead of absolute values, with the downside being that they will possibly take more iterations to adjust to the feature's "true" importance.

# 7 Conclusion and Future Work

While we believe that the motivation for the creation of the DSRF model has basis, as the completely random nature of feature selection within the Random Forest model leaves performance on the table, our implementation was only able to achieve a very meager improvement. There are a number of factors at play that contributed to this apparent weak result. First, the initial concept of the algorithm would require a very highly dimensional dataset that traditional Random Forest approaches would struggle with, and we were unable to find a dataset that matched these criteria perfectly, as although the datasets have a very high dimension, they were fit very well by traditional random forest approaches, not leaving much room for improvement. If we had more time to conduct a more thorough search for datasets, we could find a dataset that would be more suitable for our algorithm. However, the DSRF model *was* able to find a small performance uplift over RF, at the cost of computational complexity. The repetitive nature of DSRF means that there is a lot of recomputation and wasted time recomputing slightly different variations of the same tree. However, because of the value of the core idea, we believe that several modifications to our approach could have resulted in significant improvement.

An example of a direction to take future work would be to integrate improvements of the decision tree architecture as well as the random forest selection architecture. Attributes could be better evaluated if decision trees could store the loss function results at each point in tree and derive the relative importance of each attribute in the tree. Another improvement we could make with more time, is the modification of the metrics by which the features are weighted, to include some combination of recall, precision, and accuracy as opposed to only being able to work with accuracy values. This could bring substantia uplift in datasets with high accuracy, as the recall or precision becomes a better judging factor for the performance of the model. In summary, we

believe that the fundamental principles behind our algorithm, while sound, would need additional structures and more complex analysis of feature importance to fully leverage and gain access to the performance left on the table by Random Forest.

# 8 References

[1] Abdellatif, A., Abdellatef, H., Kanesan, J., Chow, C.-O., Chuah, J. H., & Gheni, H. M. (2022). Improving the heart disease detection and patients' survival using supervised infinite feature selection and improved weighted random forest. *IEEE Access*, *10*, 67363-67372. https://doi.org/10.1109/access.2022.3185129

[2] Bernard, S., Adam, S., & Heutte, L. (2012). Dynamic random forests. *Pattern Recognition Letters*, *33*(12), 1580-1586. https://doi.org/10.1016/j.patrec.2012.04.003

[3] Kushmerick, N. (n.d.). Internet advertisements. *UC Urvine Machine Learning Repository*. https://archive.ics.uci.edu/dataset/51/internet+advertisements

[4] Lifandali, O., Abghour, N., & Chiba, Z. (2023). Feature selection using a combination of ant colony optimization and random forest algorithms applied to isolation forest based intrusion detection system. *Procedia Computer Science*, *220*, 796-805. https://doi.org/10.1016/j.procs.2023.03.106

[5] Sun, Z., Wang, G., Li, P., Wang, H., Zhang, M., & Liang, X. (2024). An improved random forest based on the classification accuracy and correlation measurement of decision trees. *Expert Systems With Applications*, *237*. https://doi.org/10.1016/j.eswa.2023.121549

[6] Wang, Y., Xu, Y., Yang, Z., Liu, X., & Dai, Q. (2021). Using recursive feature selection with random forest to improve protein structural class prediction for low-similarity sequences. *Computational and Mathematical Methods in Medicine*, *2021*, 1-9. https://doi.org/10.1155/2021/5529389

[7] Tin Kam Ho (1995). Random decision forests. *Proceedings of 3rd International Conference on Document Analysis and Recognition*, *1995, pp. 278-282 vol.1*, https://doi.org/10.1109/ICDAR.1995.598994