

Enhanced Hybrid Search for LLM Hyperparameter Optimization

Stanford CS224N Default Project

Avery Voss

Department of Computer Science
Stanford University
averyv@stanford.edu

CJ Indart

Department of Computer Science
Stanford University
cjindart@stanford.edu

Aaron Sequeira

Department of Computer Science
Stanford University
aarons01@stanford.edu

Abstract

This project investigates the optimization of pre-trained Transformer architectures when transitioned from general-language objectives to highly constrained downstream tasks. While Large Language Models (LLMs) demonstrate remarkable zero-shot capabilities, their performance on specific domains is critically sensitive to the hyperparameter configuration selected during fine-tuning. A fundamental tension exists during this process: over-tuning leads to a loss of basic logic, while under-tuning results in generic, low-utility outputs. This project focuses on identifying optimal hyperparameter configurations within a restricted computational window. We propose a method that leverages the efficiency of Random Search on a narrowed sampling window to improve model accuracy while maintaining a comparable search time to baseline approaches. By evaluating 30 distinct hyperparameter combinations, we demonstrate that marginal adjustments in key variables yield large impacts on performance, achieving impactful results without exceeding a reasonable computation budget.

1 Key Information to include

- Mentor: Luke Bailey
- External Collaborators (if you have any): N/A
- Sharing project: N/A
- Late Days (if you are using any): N/A

2 Introduction

The rise of Large Language Models (LLMs) has shifted the focus of Natural Language Processing from task-specific architectures to the fine-tuning of versatile, multi-purpose models. However, finding the optimal training parameters for multiple tasks remains a significant challenge in balancing these multi-task models. In this project, we explore this tension by fine-tuning a GPT-style Transformer on two distinct objectives: paraphrase detection (using the Quora dataset) and sonnet generation (using a Shakespearean dataset).

While the Transformer architecture provides the necessary capacity for multitask learning, identifying the optimal hyperparameter configuration remains a significant bottleneck. This necessity is driven by

the extreme sensitivity of Transformers to their optimization environment. Even a minor misalignment in the learning rate can lead to gradient instability or overshooting a possible optima. Within a computation window restricted by time and cost, these challenges are amplified. Traditional Grid Search methods often fail because they suffer from the curse of dimensionality, wasting expensive cycles on unimportant hyperparameters while under sampling critical ones. On the other hand, while Random Search is statistically more efficient, it can struggle to converge on the absolute peak of performance when the high-accuracy range of the loss function is exceptionally narrow. There is, therefore, a specific need to combine the efficiency of Random Search with a more targeted, informed starting region to ensure the model settles into a stable and high-performing local minimum, without relying on a random trial to yield a "good enough" configuration.

To navigate these challenges, we implemented an Enhanced Hybrid Search strategy. This method is a unique approach in that it combines two leading methods for hyperparameter searching into one. We first run 20 trials using the Grid Search approach to find promising regions for specific weight decay and learning rate values. Using this promising region, we conduct another 10 trials using Random Search on a narrowed region centered around these values we got from Grid Search. We first utilized this method on the paraphrase detection task. This two-phase pipeline allowed us to discover a high performing configuration: $7.02e-5$ learning rate and 0.052 weight decay, which achieved a peak development accuracy of 0.897 for paraphrase detection, an increase from the 0.895 score of our standard Random Search. We then tested our method on the sonnet generation task, ultimately settling on a hyperparameter configuration of: $1.037e-4$ learning rate, $4.97e-4$ weight decay, 0.9377 temperature, and 0.7885 Top-P, yielding a chrF score of 41.91, bettering the baseline score of 41.46. Our results prove that a systematic, data-driven approach to hyperparameter selection can overcome the inherent trade-offs in multitask fine-tuning, even under strict resource constraints.

3 Related Work

The main inspiration for our project direction was a paper written by Bergstra and Bengio that introduced Random Search as a faster and equally effective method for hyperparameter optimization to the then industry-standard Grid Search [1]. The problem Bergstra and Bengio identified is that Grid Search suffers from the curse of dimensionality. In most high-dimensional spaces, only a small subset of hyperparameters truly dominated the impact of that configuration on the model. Grid Search wastes significant computational budget by repeatedly testing the same values for these important hyperparameters while varying the unimportant ones.

Bergstra and Bengio proposed a method called Random Search, which sampled randomly from a set range for each hyperparameter. In their paper, they rigorously demonstrate how Random Search is a superior alternative to Grid Search for hyperparameter optimization. Most importantly, their findings provide results to illustrate the viability of Random Search. Through extensive experiments on Neural Networks and Deep Belief Networks (DBNs), they showed that Random Search finds better (or equal) models in a fraction of the time required by Grid Search. On top of these findings, Bergstra and Bengio showed that Random Search explores the important dimensions more densely. Since Random Search does not rely on predefined values that often spend too much time on less important hyperparameters, Random Search is likely to test many different values for the important hyperparameters, allowing it to find close to ideal values.

More modern hyperparameter optimization techniques have shifted from manual tuning to automated methods like Bayesian Optimization [2] and Metagradient Descent [3]. Bayesian Optimization uses probabilistic models to predict which hyperparameter combinations will likely yield the best results. On the other hand, Metagradient Descent treats hyperparameters like learnable parameters, using the gradient of the loss function to update hyperparameters, like learning rate, automatically during training. While these methods are very powerful and state of the art for large scale pipelines, they usually require significant compute and complex implementations that can be difficult to get right. On top of that, these methods are extremely well researched. So, a more transparent hybrid approach like the one we suggest remains both more practical to implement and makes the results we obtain interpretable and novel in the space of hyperparameter tuning.

The paper that proposes an approach closest to our approach is a 2024 paper by Abhay Kumar Pathak, Mrityunjay Chaubey, and Manjari Gupta [4]. In this paper, the authors introduce a hybrid method for hyperparameter tuning that first starts with Random Search and then performs Grid Search on the

high performing areas. This paper shows that Randomized-Grid Search significantly outperforms traditional methods in cardiovascular disease classification tasks. Not only does the method improve performance and accuracy, but it also is computationally more efficient.

While Pathak, Chaubey, and Gupta demonstrate the efficiency of a Random-to-Grid hybrid in medical classification, our work provides a promising analysis of whether this optimization logic generalizes to other fine-tuning tasks. We identify a key limitation in the Random-to-Grid sequence: if the initial Random Search fails to sample near a narrow, high-performance window, the subsequent Grid Search will be anchored to a sub-optimal region, potentially missing the global optimum entirely.

Our method offers a systematic next step by inverting this sequence. By performing an initial coarse Grid Search, we establish a reliable base of the loss landscape, ensuring that our subsequent Random Search is focused on a verified region of stability. Testing this on two distinct tasks, paraphrase detection and sonnet generation, allows us to evaluate the robustness of our new hybrid approach across very different tasks, moving beyond the single-task constraints of previous literature.

4 Approach

Our approach for this project implements a hybrid framework for hyperparameter tuning by combining Grid Search and Random Search to efficiently find a strong hyperparameter configuration. First, we defined 20 different configurations of hyperparameters that span a wide yet reasonable range of values that could lead to good models. From there, we took the best configuration and created a narrow range around each hyperparameter value. Then, we ran 10 random search trials on the new, tighter intervals, which yielded a configuration of hyperparameters that performs well across both sonnet generation and paraphrase detection. We note that the number of trials we chose heavily relies on the number of hyperparameters we are trying to optimize, along with our computational resources and time. In traditional Grid Search, the number of trials would scale exponentially with the number of hyperparameters that require optimization. We only need to optimize two hyperparameters: learning rate and weight decay. Therefore, our choice of 20 trials is more justified.

We devised the main logic for this method ourselves. We define the bounds for our hyperparameters and loop through every iteration, sampling a new configuration each time, which we then use to call the train model function implemented for earlier parts of this project. We call the train model function by running the respective file (either `paraphrase_detection.py` or `sonnet_generation.py`) using a Python package called `subprocess`. This way, we can run 30 different commands with one simple master file without having to monitor the results. Detailed information about each run, including learning rate, weight decay, loss, and accuracy/chrF, is stored in a CSV file so that we can easily compare results and determine the best configuration.

The Random Search algorithm was chosen as the baseline for our project. Random Search is a well-known algorithm which is highly effective at finding optimal hyperparameter configurations. This makes it the ideal baseline for our hybrid framework to outscore. The approach for random search is quite simple: set a number of trials to execute (30 in our case), give defined ranges for the hyperparameters, then sample and record new hyperparameter values for each new trial.

Although hybrid approaches are heavily researched, including the near identical Random Search to Hybrid Search method described above [4], our approach to the project is entirely novel. Based on existing literature, and the narrowed window of our Random Search, we believe that combining Grid Search and then Random Search would lead to compelling results that are improved over our baseline.

5 Experiments

5.1 Data

For our data, we utilized the provided subsets of the Quora and Sonnet datasets, which are given to us for previous parts of the project. The Quora dataset has 141,506 training, 20,215 dev, and 40,431 test examples, which we derived binary labels from to train a classifier. The Sonnet Dataset has 143 training and 12 test sonnets, which were used to train a sonnet generator.

5.2 Evaluation method

The main sources of numerical evaluation for the detection task are training loss, training accuracy, and most importantly, dev accuracy. We tested these after every epoch of training, along with testing dev accuracy with our best model in order to determine the success of our experiments. The main sources of numerical evaluation for the sonnet generation task are training loss, and chrF score. The loss is again calculated after every epoch and the chrF score is calculated after training is complete, when we compare the sonnets our model generated to a held-out set.

One other main source of evaluation we use is time to complete. We want to see if our hybrid approach is better than the random search baseline, both in accuracy and time to complete. Overall, these evaluation metrics give us a complete evaluation of our methods and the models they produce.

5.3 Experimental details

For our preliminary baseline methods, we took the initial GPT2 framework we completed and verified using the sentiment analysis task earlier in the project, using the sentiment analysis task. We then ran 30 training loops using random configurations of hyperparameters. To do this, we used the script described above to randomly sample the desired hyperparameters, train the model, and evaluate the model 30 times. The following table lists the ranges each hyperparameter was sampled with by task:

Task	Epochs	Learning Rate	Weight Decay	Batch Size	Temperature	Top-P
Sonnet	1 to 10	1e-6 to 1e-3	1e-4 to 0.1	4, 8, or 16	0.5 to 1.2	0.7 to 1.0
Paraphrase	2 to 4	1e-6 to 1e-3	1e-4 to 0.1	16 and 32	N/A	N/A

n.b.: learning rate and weight decay are sampled from a log-uniform distribution as discussed by Bergstra and Bengio [1].

With these ranges identified, we evaluated our framework on both the paraphrase detection and sonnet generation tasks. For the paraphrase detection task, we ran the random search over a 15% subset of the full dataset (train, dev, and test) in order to conserve compute and reduce trial times. We acknowledge that this introduces some uncertainty, as the subset may not fully represent the entire training distribution. However, we reasoned that 15% of the data, roughly 15,000 examples, was sufficient for identifying configurations that generalize well to the full dataset. After determining the optimal configuration through this process, we retrained using the complete dataset.

To account for the inherent uncertainty in our subset-based optimization, we calculated the confidence bounds for our experimental results. While the Random Search baseline produced a high singular peak of 0.842, it exhibited a wider performance spread with a standard deviation of 0.0185—indicating that its top results were potentially driven by sampling luck. On the other hand, the Hybrid Search demonstrated significantly greater stability, yielding a mean development accuracy of 0.810 ± 0.008 at a 95% confidence level with a tighter standard deviation of 0.0129. This reduced variance suggests that our hybrid approach successfully found a high performing region that consistently leads to promising results, rather than relying on random sampling to find a high-performing region. These results also show that our method of testing configurations on small subsets of the data is a viable method, as the variance is relatively small.

The experiments conducted using our hybrid method followed a very similar approach. First, we ran grid search on a combination of different predetermined hyperparameter values. Because learning rate and weight decay are the "significant" hyperparameters related to improving model performance, we chose to grid search over just these values. This allowed us to sample more permutations of these in our limited sample space, while not getting hung up on hyperparameters that do not make a big impact on the model's ability to learn. The following table gives the set values we used for our grid tasks:

Task	Learning Rate	Weight Decay
Sonnet	[1e-4, 3e-4, 5e-4, 7e-4, 1e-3]	[0.0005, 0.001, 0.01, 0.1]
Paraphrase	[1e-5, 4e-5, 8e-5, 1e-4, 3e-4]	[0.005, 0.01, 0.1, 0.5]

We ran each of these permutations over 3 epochs for each task. From there, we took the best performing configurations for each task over the Grid Search and centered our new learning rate and weight decay sampling around those values. For all other hyperparameter configurations, we kept the

sampling the same as the random search baseline. The table below shows the winning configuration for each task:

Task	Learning Rate	Weight Decay	Final Loss	Final Accuracy/chrF
Sonnet	3.0e-4	0.001	3.082	41.395
Paraphrase	8.0e-5	0.1	0.359	0.823

With these results, we were able to center on these new, narrowed Random Search ranges for our final 10 sampled configurations:

Task	Epochs	Learning Rate	Weight Decay	Batch Size	Temperature	Top-P
Sonnet	1 to 10	1e-4 to 7e-4	1e-4 to 0.01	4, 8, or 16	0.5 to 1.2	0.7 to 1.0
Paraphrase	2 to 4	5e-5 to 1e-4	0.05 to 0.2	16 and 32	N/A	N/A

Finally, we run our same Random Search algorithm over 10 iterations, using these new ranges. The winner of these trials will be our output configuration from our hybrid optimization method.

5.4 Results

We first ran our baseline Random Search algorithm over our two tasks. For the paraphrase detection task, our winning configuration (the configuration with both the highest dev accuracy and lowest final loss) had the following values:

Epochs	Learning Rate	Weight Decay	Batch Size	Dev accuracy	Final Loss
3	4.28e-5	0.044	16	0.842	0.277

From here, we took this configuration and ran it over the entire dataset. The resulting dev accuracy was 0.895, with a dev f1 score of 0.888 and final loss of 0.193. Overall, the random search took 2 hours and 14.39 minutes to complete, while running our best configuration over the whole dataset took 2 hours and 5.1 minutes to complete.

For random search on the sonnet generation task, we saw a wide range of performances, as expected. Ultimately, we settled on three configurations out of the 30 that had a final training loss below 2. The following table gives the configurations for the aforementioned 3 runs:

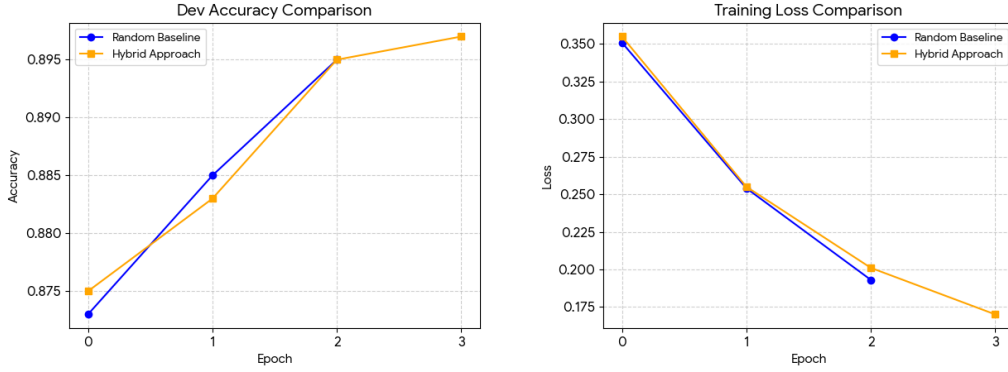
Epochs	Learning Rate	Weight Decay	Batch Size	Temperature	Top-P	Final Loss
7	4.12e-4	2.5e-4	4	0.52	0.93	0.56
5	3.2e-4	0.028	4	0.52	0.86	1.653
7	9.6e-5	5.3e-3	4	0.69	0.90	1.727

We finally evaluated these three configurations on the dev set and calculated chrF scores for each. We got scores of 41.46, 41.42, and 41.44 respectively for the configurations above. The total runtime for this task was 1 hour and 8.614 minutes.

Once these baseline configurations were evaluated, we proceeded with our hybrid approach. On the paraphrase detection task, we also had a run that had both the lowest final loss and highest dev accuracy. The following table gives this configuration:

Epochs	Learning Rate	Weight Decay	Batch Size	Dev accuracy	Final Loss
4	7.02e-5	0.052	32	0.831	0.311

Next, we ran this configuration over our whole dataset to get the final results for our hybrid approach. This configuration resulted in a dev accuracy of 0.897 (which has been submitted to the leaderboard), dev f1 score of 0.890, and final loss of 0.170. In total, this hybrid search took 2 hours and 29.232 minutes, which is a fairly similar runtime to the 2 hours and 14 minute runtime of the baseline. We can also see the trends of both the losses and dev accuracies of the random baseline and hybrid search methods in the figures below.

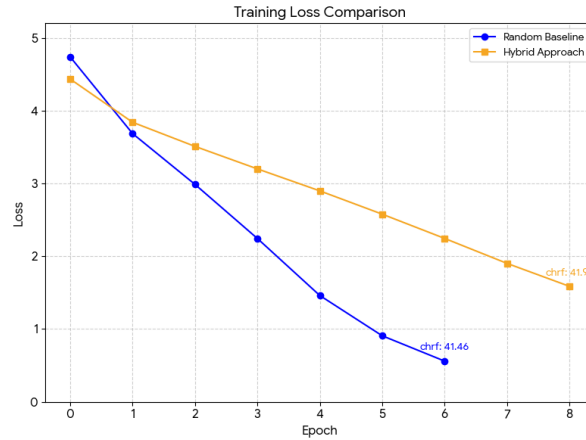


These figures help show the legitimacy and impact of our hybrid method on the paraphrase detection task, as it achieved a lower loss and higher accuracy than the random baseline, along with having a higher F1 score. These suggest that our model is correctly classifying paraphrases since it understands what a paraphrase is as opposed to memorization.

Finally, for the sonnet generation task, we executed on the same plan as for the paraphrase task. We had one clear winning configuration that had a higher loss than other configurations, but had a higher chrF than the others.

Epochs	Learning Rate	Weight Decay	Batch Size	Temperature	Top-P	Final Loss
9	1.037e-4	4.97e-4	4	0.9377	0.7885	1.587

The final chrF score for this run was 41.91 (which has been submitted to the leaderboard). The total time for this process was 1 hour and 26.98 minutes, again a comparable runtime to the 1 hour and 8 minute runtime of the baseline. Just as we had with the hybrid performance on paraphrase detection, we can see that our hybrid approach performed favorably to the baseline approach on this task. While the loss was higher, the chrF score was also higher than the baseline, likely hinting at our baseline model overfitting with such a low loss. We can see this trend in the figure below.



As the figure shows, the baseline achieved a lower loss than the hybrid approach, but had a chrF score of 41.46, compared to the higher chrF of our hybrid approach of 41.91. As mentioned above, this hints at our baseline model overfitting and memorizing the training data, resulting in a low loss with a low chrF score.

6 Analysis

The success of the Enhanced Hybrid Search over the standard Random Search baseline suggests that for Transformer fine-tuning, where you search is just as important as how you search. Our analysis

reveals that the hybrid approach did not just find a higher accuracy, but it found a more promising region of the loss landscape.

For paraphrase detection, we can explain the slim improvement with a couple observations. The primary reasoning for this improvement is because this task is easier to learn; paraphrases can very generally be defined as having the same topic and sentiment in a sentence. This makes achieving meaningful gains in accuracy difficult, because even suboptimal configurations can still decently learn these simple patterns.

Another reason for the close scores may be due to Random Search luckily sampling a configuration that works really well, something that cannot be guaranteed every time Random Search is used. From our results, we can see that the sampled weight decay is very close to the weight decay of our novel hybrid approach on the paraphrase task, while learning rates are almost identical.

Lastly, we observe that our hybrid approach navigated the loss landscape more effectively by identifying a high-performance region. Even though the random search baseline performed well in general, our method’s structured optimization allowed it to continue learning efficiently, even sampling over a critical final epoch, ultimately achieving a superior accuracy of 0.897. This trajectory suggests that the hybrid model converged upon a more robust local minimum, representing a more thorough solution for the paraphrase detection task.

For sonnet generation, the hybrid approach similarly demonstrated its value in navigating a more constrained loss landscape. The baseline using Random Search, while producing three good chrF scores, converges to a max of 41.46. This is a good result, but hinted at possible overfitting with an unusually low loss of 0.56 relative to the chrF score that was produced. This disconnect between loss and chrF score suggests that the model memorized training patterns rather than learning the sonnet style generally.

The hybrid approach directly addressed this issue. Our hybrid approach locked our hyperparameters in a range that was created by our grid search and narrowed the sampling window, guiding the model toward configurations that balanced training loss with generalization. Our winning configuration with 9 epochs, learning rate $1.037e-4$, weight decay $4.97e-4$, temperature 0.9377, Top-P 0.7885, achieved a higher chrF score of 41.91 while producing a higher loss of 1.587. This is a meaningful result, showing that a higher chrF with a higher loss points to a model that was able to generalize well to the held-out sonnet rather than overfitting.

The temperature and Top-P values are also important to note. The high temperature (0.9377) and moderate Top-P (0.7885) encourages diverse tokens while constraining to high probability outputs. Looking at the random baseline, it used a much lower temperature (0.52), which likely lead to its overfitting and memorization behavior.

These results reinforce the mission of our hybrid approach. For both tasks, our hybrid approach consistently found higher-performing configurations to the random baselines while finishing at comparable times. More specifically, the sonnet generation improvement is compelling, since the dataset contained a relatively small amount of data (143 training sonnets), which makes the possibility of overfitting to the data more likely. Although we acknowledge that Random Search can occasionally find higher performing configurations by chance, our methods remove this reliance on luck by systematically narrowing the search area and checking performance throughout. The consistency of our method is what makes our Hybrid Search a promising direction for future hyperparameter optimization work.

7 Conclusion

This project demonstrates that multitask LLM fine-tuning can be effectively managed through a structured, two-phase optimization strategy. By reversing the order of the previously researched Randomized-Grid Search, we achieved a peak dev accuracy of 0.897 and very stable test accuracy of 0.878 for the paraphrase detection task and a peak chrF score of 41.909 on the dev set and 41.714 on the test set for the sonnet generation task.

The key finding of our project is that the area of the loss function to be minimized is the primary driver of performance in fine-tuning. While standard Random Search is broad, it is often too noisy and random to settle into the narrow, high-performance windows that bring out the best results in

these tasks. Our novel hybrid method successfully found a promising region of the loss function, and then searched effectively for the optimal values, resulting in a model that generalizes more effectively to unseen data while avoiding the issues with overfitting experienced by many models.

Despite these successes, our work is limited by the computation resource and accuracy balance. Because we utilized a 15% subset for the initial search phases, there is a risk that some hyperparameter configurations that work well on small data do not scale perfectly to the full 400,000-row Quora dataset. Although we analyzed the confidence bounds and found a small variance over these tests, there are still risks and possible unknowns in searching over small subsets of the data. However, for the sonnet generation task, we are fairly confident that our methods were not impacted by this trade off, as we sampled, trained, and tested over the entire, small dataset.

Future work could include expanding our Grid Search portion to more hyperparameters, rather than just learning rate and weight decay. Rather than Random Search, our Hybrid Search method can be compared to more studied and robust methods such as the ones mentioned prior, like Bayesian Optimization, and even combining our hybrid approach with these more modern methods to perform well on more complex tasks. Lastly, our project shows that even with limited GPU resources, a systematic and grounded approach to hyperparameter tuning can yield results that rival both previous and current industry standard pipelines.

Team contributions (Required for multi-person team)

- Avery: Avery helped implement the initial architecture and the random and grid search algorithms. Avery also helped run various experiments, both for the baselines and for the extension. Finally, Avery contributed to writing the Proposal, Milestone, and Final reports.
- Aaron: Aaron helped implement the initial architecture and helped with running experiments for the baseline and extension to validate results. Aaron also contributed with the project deliverables, especially in creating and formatting all material and results for the poster, as well as in writing the Proposal and Final reports.
- CJ: CJ helped implement the initial architecture and improve the baseline models through grid and random search. Specifically, CJ used class Modal credits and our hybrid search to find hyper parameters for sonnet generation. CJ also helped with the class writeups (proposal, milestone, final), especially the Final report.

References

- [1] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. In *Journal of Machine Learning Research*, 2012.
- [2] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. In *arxiv:1206.2944*, 2012.
- [3] Logan Engstrom, Andrew Ilyas, Benjamin Chen, Axel Feldmann, William S. Moses, and Aleksander Madry. Optimizing ml training with metagradient descent. In *arxiv:2503.13751*, 2025.
- [4] Abhay Kumar Pathak, Mrityunjay Chaubey, and Manjari Gupta. Randomized-grid search for hyperparameter tuning in decision tree model to improve performance of cardiovascular disease classification. In *arxiv:2411.18234*, 2024.