Intelligent Automation & Soft Computing

Tech Science Press

DOI: 10.32604/iasc.2024.059429

ARTICLE

Overfitting in Machine Learning: A Comparative Analysis of Decision Trees and Random Forests

Erblin Halabaku and Eliot Bytyçi*

Department of Mathematics, Faculty of Mathematical and Natural Sciences, University of Prishtina, Prishtina, 10000, Kosovo

*Corresponding Author: Eliot Bytyçi. Email: eliot.bytyci@uni-pr.edu

Received: 08 October 2024 Accepted: 10 December 2024 Published: 30 December 2024

ABSTRACT

Machine learning has emerged as a pivotal tool in deciphering and managing this excess of information in an era of abundant data. This paper presents a comprehensive analysis of machine learning algorithms, focusing on the structure and efficacy of random forests in mitigating overfitting—a prevalent issue in decision tree models. It also introduces a novel approach to enhancing decision tree performance through an optimized pruning method called Adaptive Cross-Validated Alpha CCP (ACV-CCP). This method refines traditional cost complexity pruning by streamlining the selection of the alpha parameter, leveraging cross-validation within the pruning process to achieve a reliable, computationally efficient alpha selection that generalizes well to unseen data. By enhancing computational efficiency and balancing model complexity, ACV-CCP allows decision trees to maintain predictive accuracy while minimizing overfitting, effectively narrowing the performance gap between decision trees and random forests. Our findings illustrate how ACV-CCP contributes to the robustness and applicability of decision trees, providing a valuable perspective on achieving computationally efficient and generalized machine learning models.

KEYWORDS

Artificial intelligence; decision tree; random forest; prune; overfitting

1 Introduction

Machine learning, a cornerstone of modern data analysis, represents a significant leap in our ability to make sense of vast and complex datasets [1]. At its core, machine learning leverages algorithms to parse, learn from, and make predictions or decisions based on data and experience [2]. This process is crucial in an era where data generation and collection have escalated exponentially. Following the increasing global competition, fast technology evolution and customer perceptions of product quality have triggered the demand for future strategic plans and advanced manufacturing techniques [3]. This overwhelming data presents unique challenges, not least of which is the phenomenon of overfitting, particularly prevalent in simpler models like decision trees [4] when a model is trained too well on the training data and becomes too specific to that data, leading to a poor generalization of new data. This leads to weak adaptability of new, unseen data, undermining the model's predictive or explanatory

power. Decision trees, with their depth and complexity, are prone to overfitting because they can create particular rules that perfectly describe the training data but fail to apply to the test data [5]. In response, the machine learning community has developed robust algorithms designed to handle the complexity and scale of modern datasets while mitigating the risk of overfitting. One such solution is the random forest algorithm, which integrates the simplicity and interpretability of decision trees, which decomposes a complex decision-making process into a set of more straightforward decisions with ensemble learning techniques [3] by constructing multiple decision tree classifiers where each tree casts a unit vote for the most popular class, random forests aim to maintain the strengths of decision trees of handling high dimensional data without feature selection.

However, despite the effectiveness of random forests in addressing overfitting, this approach has its trade-offs, particularly in terms of increased energy consumption, computational power, and difficulty in visualization. These factors introduce significant challenges, especially in resource-constrained environments or applications where interpretability is crucial. Due to their structure, this paper hypothesizes that decision trees will exhibit a higher degree of overfitting than random forests when applied to the same datasets. The study aims to explore this hypothesis by leveraging machine learning techniques to evaluate and compare the generalization performance of these models in terms of accuracy and usage of resources. By focusing on these trade-offs, the primary objective of this study is to investigate how we can leverage machine learning to balance accuracy with resource efficiency, potentially offering a more cost-effective alternative to the widespread use of random forests. This approach exemplifies the broader strategy in machine learning of developing methodologies that can harness the vast potential of big data without succumbing to its overwhelming complexity. Regarding this vast amount of data, Reference [6] pointed out that a fundamental and unsolved problem is that the working mechanism of deep learning, by extension, machine learning models, could be more understandable. With sufficient interpretability, applying these AI systems in real-life applications would be easier.

2 Research Methodology

Many publications cover classification algorithms, such as Decision Trees and Random Forests, explicitly focusing on their overfitting constraint, as presented in Table 1.

Table 1: Number of publications revised

Database	First pass	Second pass
SpringerLink	22	10
IEEE Xplore	0	0
ScienceDirect	57	32
ACM Digital Library	7	4

To systematically explore this research area, we employed a mapping study approach. This method allows us to comprehensively review and categorize the existing publications, providing a clear overview of the research landscape. The first step was to use the digital libraries to search for related scientific publications: SpringerLink, Science Direct, IEEE Xplore, and ACM Digital Library. The keywords used are “Decision Tree” and “Random Forest”. These search keywords resulted in a huge number of research papers, reaching thousands of them. An extra filtering layer was used to select the latest research and remove all publications older than 5 years.

Nevertheless, this provided some accuracy in our search, but still, it was such a large number of results that it required more filtering. For this reason, some new keywords were presented to narrow the field of study and keep our paper more relevant. The added keywords were "Prune" and "Overfitting". This had such an impact on the results of the search. Nevertheless, it still needed to be filtered more. For this purpose, another filtering round was added because the search yielded all kinds of publications, only to retrieve articles and conference papers. For a more detailed report, see Table 1, which provides detailed information regarding the number of publications. The best results were from Science Direct, which had 57 publications, followed by SpringerLink Library, which had 22 publications, then ACM Digital Library, which had seven publications, and lastly, the IEEE Xplore did not return any publications. In the case of IEEE Xplore, there were a small number of publications to begin with, and they were deemed not relevant for the purpose of this study, and none of them passed the first and second pass. Either the case study was not relevant and very specific to a not-generalized topic, or the title contained the keywords, but they needed to make more sense to be related to the paper. This was a good turnout, but after reading them carefully, removing duplicates, and collecting the ones deemed more valuable, the second pass came to exist with the corresponding number of publications.

Based on Fig. 1, we can see the trend that follows the number of publications over the last 5 years. This 5-year is loosely defined since it also includes the year 2024. This is an exception since some publications were up to date, and it felt wrong to exclude them because the year had just started. We can see an incline from 2018 to 2023, but it is a slow decline from that year on. This decline is because we have just entered the new year, and there is still quite some time before publications can be published.

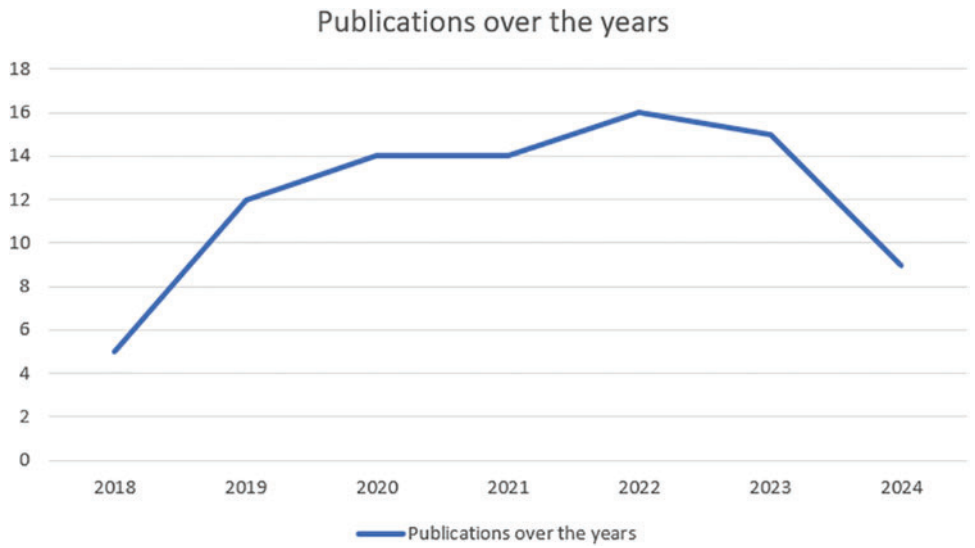


Figure 1: Publications over the time

3 Analysis of the Results

3.1 Theoretical Part

Decision trees are powerful tools in machine learning for classification that uses a tree-structure to model decisions and possible consequences by summarizing a set of classification rules from the data

[7]. Additionally, decision trees are highly effective and popular in data mining applications, where their ability for relatively easy interpretation and efficient computation time proves invaluable [8]. It uses nodes and internodes for the prediction and classification [9], with root nodes classifying the instances with different features by having two or more branches, while the leaf nodes represent classification. The hierarchical structure of decision trees performs a classification task by feeding the data to nodes divided into branches for transferring the input to the most suitable class label [10], which enables the tree to be formed by selecting the nodes as they are divided into branches. However, a significant constraint of decision trees is their tendency to overfit the training data, especially when the tree grows without constraints. Decision trees can become quite tricky when dealing with dirty data, leading to poor estimation of unseen data. This phenomenon occurs because decision trees aim to partition the feature space into regions that are as pure as possible regarding the target variable, often resulting in excessively specific rules that may not generalize well. Additionally, decision trees exhaustively search the entire feature space to find the optimal splits at each node, which can be computationally expensive, particularly for datasets with many features or instances.

A decision tree is fundamentally structured as a series of nodes that guide data to its final classification or prediction. Each decision tree is either a leaf $\lambda(y)$ for some label $y \in Y$ or an internal node $\sigma(f, v, t_l, t_r)$, where $f \in \{1, \ldots, d\}$, for some value d which identifies a feature, $v \in R$ is a threshold for the feature and t_l, t_r are decision trees (left and right child), where study [11] also used $\sigma(f, v)$ to represent an internal node when t_l, t_r are not so important. Decision trees are generally composed of branch nodes B and leaf nodes L. According to [12], a branch node $t \in B$ applies a splitting rule on the samples in that node, dividing data according to specific feature thresholds and directing subsets of data to new branches or leaves. Leaf nodes $l \in L$, on the other hand, act as terminal points in the tree and aggregate samples that satisfy all previous splits along their path. Each leaf node typically assigns a single class label to all samples it contains, often determined by majority voting within the node. This mechanism allows decision trees to provide clear, interpretable decision paths, making them practical for classification tasks where interpretability is critical.

When data exhibits monotonicity—where the target variable consistently increases or decreases with certain features—a decision tree can become overly eager to capture this pattern, often resulting in increasingly specific splits to match the trend. This strict adherence to monotonic relationships may lead the tree to overfit, as it generates complex rules that align closely with the training data but fail to generalize to new samples. Consequently, the tree becomes sensitive to minor fluctuations in the data, capturing noise as meaningful patterns, which ultimately reduces its robustness on unseen data. This was best recognized in [1] where in any given model, defined as a function $M : X_1 \times \ldots \times X_n \rightarrow Y$, where X_i is the value set of features and Y is the set of classes, is considered strongly monotone with respect to a feature i if for any two data instances $x = (x_1, \ldots, x_n)$, $x' = (x'_1, \ldots, x'_n) \in X$ we have $x_i \leq_i x'_i$ implies $M(x) \leq_Y \leq M(x')$. This notation underscores how monotonic trends in data can influence model decisions, often leading to a direct but overly strict mapping that fails to account for variations outside the training set. This tendency was highlighted by [13], where the chance of overfitting is highest if the parameters are associated with equifinality, meaning that the likelihood of different parameter values is similar in a predefined interval.

Machine learning techniques play a preponderant role in dealing with massive amounts of data and are employed in almost every possible domain [14]; therefore, building a high-quality machine learning model is challenging for both the subject matter experts and the machine learning practitioners. This challenge is further amplified by the necessity of effective feature selection, which is a critical step in the model-building process. Many machine learning methods may lead to worse performance because of a large number of redundant features; hence, study [15] puts much emphasis on

feature selection, which reduces overfitting and the number of features and improves the generalization ability of the model. By adopting a standard approach according to [16], we compute the feature importance as the sum of the information gained provided by a feature each time it triggers a split in one of the trees composing the model. Feature selection is the step of choosing the most relevant subset of features from the entire feature space of the original dataset and using them as input into the ML models [4]. This process aims to reduce the dimensionality, costs, information loss, and overfitting of the training model. In addition to eliminating redundancy, feature selection improves the model's interpretability and understanding of the relationship between the explanatory variables and the responses [17]. With the ever-growing technology computational power [18], it has been brought up that several feature selection techniques are used in intrusion detection for feature selection, namely, information gain and feature correlation. However, the usage and type of technique solely depend on the need and type of data we are working on. On the other hand, [19] pointed out that feature selection can be realized through embedded methods that keep and remove features as part of the model construction process, such as Lasso (for least absolute shrinkage and selection operator), performance-influence models, and decision trees.

Ensemble methods, such as random forests, have been developed to address decision trees' overfitting and computational constraints. Random forest is an ensemble of, typically, tens to thousands of base decision tree classifiers [20], where the ensemble operates in parallel and classifies by majority vote.

This pruning process can easily be illustrated in Fig. 2. When provided by a large dataset, random forest will create multiple sub-decision trees, and each sub-decision will provide a vote regarding the selection process. According to [11], pruning leads to a range of simple and intuitive solutions that take advantage of state-of-the-art implementations of existing training algorithms. Those sub-trees whose result has been seen as unfitted will be pruned so they will only take up a little space. This weighted ability to be able to prune them has been determined by the weighted total Gini Impurity IG [20], which is always chosen, and those child nodes are then added to the growing tree by calculating

$$I_G(Q) = \sum_{k=1}^{K} p_k (1 - p_k)$$

where p_k is the proportion of instances with the label p_k in a node Q, and K is the number of classes we consider in a specific dataset. The Gini index as an impurity function [21] state that each new separation, according to a decision rule between the nodes, has been performed via maximal impurity reduction.

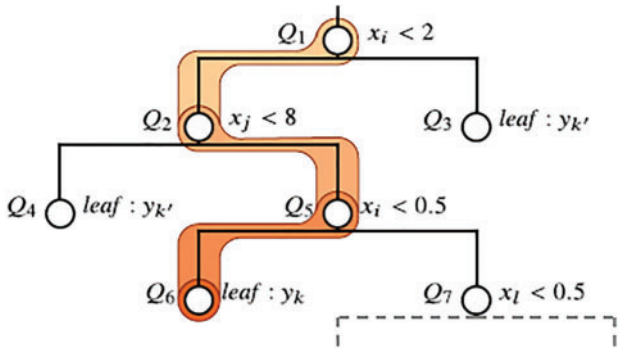


Figure 2: Pruning the decision tree [20]

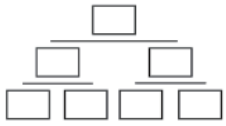
One of the key strategies employed by random forests to mitigate overfitting is the introduction of randomness during the tree-building process. This randomness is manifested by selecting a random subset of features, often called "feature bagging," thereby reducing the likelihood of individual trees relying too heavily on specific features [22]. This randomization introduces diversity among the trees in the forest, making them less correlated and more robust to noise in the data. Therefore, the random forest model overcomes the overfitting issue by constructing individual trees and minimizing their correlation in the classification task [10].

Moreover, random forests also address the computational constraints associated with decision trees by parallelizing the training process and enabling efficient scaling to large datasets. Unlike decision trees, which exhaustively search the entire feature space to find the best split at each node, random forests only consider a random subset of features at each split, significantly reducing the computational burden [23]. Additionally, training individual trees in a random forest can be parallelized, allowing for efficient utilization of multicore processors or distributed computing frameworks. Given the high-dimensional nature of applied data, this classic supervised learning algorithm, such as random forest, commonly accompanies a feature selection step to obtain low-dimensionality representations [24]. This parallelization enables random forests to handle large-scale datasets with millions of instances and thousands of features, which would be infeasible for traditional decision tree algorithms.

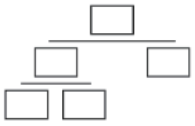
The interpretability of models has become a crucial issue in machine learning because algorithmic decisions have a growing impact on real-world applications [12]. Most classification algorithms consider that features are independent and do not consider the relations between features [2], where due to lack of interpretability are considered as Blackbox models. At its core, a random forest is an ensemble method that combines the predictions of numerous decision trees by employing the process of bagging [25,7] and hence inheriting all the features of a single decision tree classifier. Even though combining multiple trees may provide higher prediction quality than a single one, it sacrifices the interpretability property, resulting in "black box" models [12].

Using random forests introduces another layer of randomness through bootstrap sampling, also known as "bagging," during the construction of each tree. Bootstrap sampling involves random sampling with replacement from the original dataset to create multiple bootstrap samples of the same size as the original dataset.

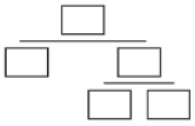
Based on Fig. 3, we can see that from a training dataset, we applied the bootstrapping principle to use our computational resources efficiently. Each bootstrap sample trains a separate decision tree in the forest, leading to variations in the training data fed to each tree. This variation helps to decorate the trees in the ensemble further and reduces the risk of overfitting to specific patterns in the data. We can see from Fig. 3 how the dataset was divided into random chunks of the dataset in hand to create separate test cases. By aggregating the predictions of multiple trees trained on different bootstrap samples, random forests achieve better generalization performance and robustness than individual decision trees.



(a) The first decision tree



(b) The second decision tree



(c) The third decision tree

Figure 3: Various decision trees generated from bootstrap sampling

4 Experiments

To verify our claims, there have been two approaches to our problem. First, in regard to addressing the overfitting of the pruning strategy, the Adaptive Cross-Validated Alpha CCP (ACV-CPP) was proposed as a solution that will disregard those branches of the decision tree that don't align with our interests. The other approach is the usage of random forests to compensate for the letdown of overfitting by generating random subsets of trees.

Eleven different datasets involving classification problems are used to evaluate our models' effectiveness, as shown in Table 2. The datasets are collected from the University of California, Irvine (UCI) machine learning repository [26] and have different numbers of features and instances.

Table 2: Description of the datasets

Dataset	Instances	Features	Classes
Iris	150	4	3
Breast cancer	569	30	2
Wine	178	13	3
Digits	1797	64	10
Diabetes	442	10	214
20 Newsgroups	18,846	173,762	20
Olivetti faces	400	4096	40
COIL20	10,299	561	6
MNIST	70,000	784	10
Covertype	581,012	54	7
SVHN	9298	9298	10

In the first group of results, the performance of the decision tree classifier and the metadata response, including the model's accuracy, tree depth, number of nodes, and number of features on average, were evaluated. By loading the dataset, we iterated the process of classifying and retrieving the said metadata over 50 times and found the average for each case.

Algorithm 1: Decision tree classifier with metrics calculation

Require: Dataset X, Labels y, Number of runs $num\_runs = 50$
Ensure: Average metrics for decision tree classifier

(Continued)

Algorithm 1 (continued)

```
Extract feature X and labels y from the dataset

total_metadata ← 0

for i = 1 to num_runs do
    X_train, X_test, y_train, y_test ← train_test_split
    clf ←DecisionTreeClassifier()
    clf.fit(X_train, y_train)
    y_pred ← clf.predict(X_test)
    accuracy ← accuracy_score(y_test, y_pred)
    total_metadata ← total_metadata + metadata
end for

average_metadata ← total_metadata/num_runs
```

This process is best described in Algorithm 1, which describes the whole process of iterating. In each iteration, we split our training and testing dataset by 70% and 30%, respectively, and ran the decision tree classifier by which we got the wanted metadata and saved each turn. After successfully finishing the iterations, the average metadata is calculated and retrieved, displayed in Table 3. Following this, we started pruning the datasets to achieve the best efficiency level and compare the results. The optimal value that yielded the best results from the decision tree classifier must be found to determine the complexity parameter to prune the dataset. This is best elaborated in Algorithm 2, which shows the steps necessary to find the best value. This process is quite computationally demanding, so it is pretty lengthy.

Algorithm 2: Decision tree classifier with pruning and metrics calculation

Require: Dataset X, Labels y, Number of runs $num\_runs = 50$
Ensure: Average metrics for decision tree classifier

```
Extract feature X and labels y from the dataset

total_metadata ← 0

for ccp_alpha in ccp_alphas do
    clf ← DecisionTreeClassifier()
    clf.fit(X_train, y_train)
    train_scores.append(clf.score(X_train, y_train))
    test_scores.append(cross_val_score(clf, X_test, y_test, cv = 5))
end for
best_alpha ← ccp_alphas[np.argmax(test_scores)]

for i = 1 to num_runs do
    X_train, X_test, y_train, y_test ← train_test_split
    clf ← DecisionTreeClassifier(random_state = 42)
    path ← clf.cost_complexity_pruning_path(X_train, y_train)
```

(Continued)

Algorithm 2 (continued)

```
    ccp_alphas ← path.ccp_alphas
    train_scores ← [ ]
    test_scores ← [ ]

    clf ← DecisionTreeClassifier(best_alpha)
    clf.fit(X_train, y_train)
    y_pred ← clf.predict(X_test)
    total_metadata ← total_metadata + metadata
end for

average_metadata ← total_metadata / num_runs
```

After iterating through each value of the cost complexity index by pruning the dataset running the decision tree classifier, and retrieving the best likelihood of accuracy, we can start the main iterating system to determine the main metadata. This loop is repeated 50 times, and the percentage of testing and training dataset is the same as indicated in the algorithm in Algorithm 2 to keep a coherent approach throughout the experiment. After collecting the results, we display those in Table 3.

The second phase of the experiment was comparing the performance of the random forest classifier with the previous algorithms from Algorithms 1 and 2. The same parameters were applied here as before: the number of iterations, the ratio of training and testing dataset, and metadata collection. This process is elaborated in detail in Algorithm 3. The only thing separating the random forests is the number of sub-trees created from the initial dataset. In our case, we set that number to 500 to achieve the near-optimal distribution of data among all random trees generated.

Algorithm 3: Random forest classifier with metrics calculation

Require: Dataset X, Labels y, Number of runs $num\_runs = 50$, Number of estimators $n\_estimators = 500$

Ensure: Average metrics for random forest classifier

```
Extract feature X and labels y from the dataset

total_metadata ← 0

for i = 1 to num_runs do
    X_train, X_test, y_train, y_test ← train_test_split( )
    rfc ← RandomForestClassifier(n_estimators = n_estimators)
    rfc.fit(X_train, y_train)
    predictions ← rfc.predict(X_test)
    total_metadata ← total_metadata + metadata
end for

average_metadata ← total_metadata / num_runs
```

After successfully completing the classification process, the metadata is collected and stored in Table 3 to better understand our performance.

Table 3: Experimental results

Dataset	Method	Accuracy	Tree depth	Nodes
Iris	DT	100%	5.5	18
	DT-pruned	100%	6	19
	RF	100%	5.3	16.22
Breast cancer	DT	93.10%	7	31
	DT-pruned	94.15%	7	31
	RF	96.4%	7.16	35.5
Wine	DT	95.41%	4	13
	DT-pruned	96.30%	4	13
	RF	100%	4.82	18.32
Digits	DT	84.69%	14	269
	DT-pruned	83.15%	10	73
	RF	97.56%	13.45	334.18
Diabetes	DT	0.81%	22.58	552.08
	DT-pruned	2.26%	22	551
	RF	0.90%	30.65	396.11
20 Newsgroups	DT	63.33%	220.16	5575.64
	DT-pruned	63.12%	203	4561
	RF	86.08%	256.34	11,708
Olivetti faces	DT	48.45%	38.96	121.04
	DT-pruned	53.33%	39	119
	RF	91.00%	31.12	138.22
COIL 20	DT	92.38%	26	449.44
	DT-pruned	92.82%	14	243
	RF	97.71%	19.84	613.59
MNIST	DT	86.87%	40	6824.20
	DT-pruned	87.00%	40	6509
	RF	96.73%	33.45	8930.12
Covertype	DT	93.41%	43	45,001.44
	DT-pruned	93.44%	43	43,915
	RF	95.47%	49.89	84,723.41
SVHN	DT	87.43%	20	808.12
	DT-pruned	85.88%	11	145
	RF	95.57%	19.68	882.04

Table 3 shows that the random forest (RF) model's performance surpasses that of the decision tree (DT) and pruned decision tree (DT-pruned) in terms of accuracy across all datasets.

In the Breast Cancer dataset, the random forest achieves a higher accuracy of 96.4% compared to 93.10% for the DT and 94.15% for the DT-pruned. Despite this improvement in accuracy, the tree depth and the number of nodes increase slightly, indicating that the random forest model introduces more complexity.

Similarly, in the Wine dataset, the random forest achieves 100% accuracy, significantly higher than 95.41% of the DT and 96.30% of the DT-pruned. However, this comes with an increase in tree depth (from 4 to 4.82) and the number of nodes (from 13 to 18.32), illustrating the trade-off between achieving higher accuracy and increasing model complexity.

For more complex datasets like Digits, the random forest shows a substantial improvement in accuracy, reaching 97.56%, compared to 84.69% and 83.15% for the DT and DT-pruned models, respectively. Despite this, the average tree depth for the random forest is 13.45, and the number of nodes is 334.18, compared to the significantly lower numbers in the DT and DT-pruned models.

In the case of large datasets such as the 20 Newsgroups and MNIST, the random forest again shows a notable increase in accuracy, achieving 86.08% and 96.73%, respectively. However, this increase in performance is accompanied by a considerable rise in tree depth and the number of nodes. For the 20 Newsgroups dataset, the tree depth of the random forest is 256.34 with 11,708 nodes, compared to the lower values in the DT and DT-pruned models.

Analyzing the experimental results also provides critical insights into the overfitting behavior of decision trees compared to pruned decision trees and random forests. The identification of overfitting is evident in the unpruned decision trees, particularly in datasets like 20 Newsgroups, MNIST, and Covertype, where the tree depths and node counts are extraordinarily high. This excessive complexity is a clear indicator of overfitting, where the model is learning the underlying patterns and capturing noise and anomalies in the training data. This is particularly problematic in datasets with high dimensionality or those that contain a significant amount of noise. As the depth of the tree increases, the model becomes more specialized to the training data, leading to poor generalization to new, unseen data. The impact of this overfitting is most pronounced in the unpruned decision trees, as seen in the large discrepancies between the number of nodes and tree depth compared to the pruned versions and random forests. For instance, in the 20 Newsgroups dataset, the unpruned decision tree reaches a tree depth of 220.16 with 5757.64 nodes. At the same time, the pruned version significantly reduces this to 203 in-depth and 4561 nodes, albeit with a slight reduction in accuracy. This reduction highlights the effectiveness of pruning as a method to combat overfitting by simplifying the model and enhancing its generalization capability. Random forests, on the other hand, demonstrate a different approach to mitigating overfitting. The ensemble nature of random forests, which involves averaging the results of multiple decision trees, inherently reduces the variance in the model, thereby preventing overfitting. This is evident in their superior performance across all datasets, achieving higher accuracy with a balanced tree depth and node count. For example, in the MNIST dataset, the random forest achieves an accuracy of 96.73%, with a tree depth of 33.45 and 8930.12 nodes, significantly outperforming the unpruned and pruned decision trees in accuracy and generalization.

Overall, the data from Table 3 illustrates that while the random forest model consistently outperforms the decision tree and pruned decision tree models in terms of accuracy, it does so at the cost of increased model complexity. This complexity is reflected in the greater tree depth and the higher number of nodes, highlighting the trade-off between achieving superior predictive performance and managing the interpretability and efficiency of the model.

4.1 Computational Resource Analysis

To complement our comparative analysis of Decision Trees and Random Forests, we conducted an in-depth examination of their computational resource requirements across various datasets. Table 4 provides quantitative insights into the Training Time, Peak Memory Usage, CPU Usage, and Model Size on Disk for both models.

Table 4: Experimental, computational resource analysis

Dataset	Method	Training Time	Peak Memory Usage	CPU Usage	Model Size on Disk
Iris	DT	0.0006 s	180.30 MB	0.01%	0.0024 MB
	RF	0.05 s	181.31 MB	0.12%	0.18 MB
Breast cancer	DT	0.0042 s	180.29 MB	0.01%	0.0037 MB
	RF	0.09 s	181.76 MB	0.26%	0.30 MB
Wine	DT	0.0008 s	179.76 MB	0.01%	0.0024 MB
	RF	0.05 s	181.81 MB	0.15%	0.18 MB
Digits	DT	0.0101 s	182.51 MB	0.01%	0.0372 MB
	RF	0.17 s	197.62 MB	0.50%	4.41 MB
Diabetes	DT	0.0074 s	184.07 MB	0.01%	0.7919 MB
	RF	0.28 s	301.88 MB	0.76%	56.46 MB
20 Newsgroups	DT	15.5339 s	351.48 MB	4.34%	1.1871 MB
	RF	78.27 s	801.70 MB	4.25%	251.16 MB
Olivetti faces	DT	1.3351 s	201.44 MB	2.37%	0.0464 MB
	RF	1.36 s	211.42 MB	2.39%	4.88 MB
COIL20	DT	3.9180 s	335.79 MB	3.33%	0.0545 MB
	RF	9.00 s	349.05 MB	4.03%	6.25 MB
MNIST	DT	9.4066 s	1601.09 MB	3.99%	0.9500 MB
	RF	22.32 s	1849.31 MB	4.03%	116.45 MB
Covertype	DT	4.8188 s	772.59 MB	3.46%	5.1531 MB
	RF	63.53 s	2608.10 MB	3.94%	928.84 MB
SVHN	DT	1.4251 s	246.95 MB	2.44%	0.1180 MB
	RF	5.87 s	269.76 MB	3.97%	11.56 MB

As the data illustrates, Random Forests generally incur higher computational costs than Decision Trees. For instance, the training time for Random Forests is consistently longer, particularly on larger datasets like the Covertype and 20 Newsgroups datasets, where training times exceed those of Decision Trees by significant margins. This increase is due to the ensemble nature of random forests, which build multiple trees simultaneously, thereby increasing the overall duration of training. Additionally, the Peak Memory Usage of Random Forests is notably higher, reflecting the concurrent memory requirements for managing multiple decision trees. Similarly, CPU Usage is markedly higher for Random Forests, especially on complex datasets, indicating a greater demand for processing power. This is crucial for applications in resource-constrained environments where CPU allocation is limited.

Furthermore, the Model Size on Disk is substantially larger for Random Forests, as the ensemble structure produces more complex models. For instance, on the Covertype dataset, the model size of Random Forests reaches over 900 MB, compared to only 5 MB for Decision Trees. These findings underscore the practical trade-offs between achieving higher accuracy and managing resource efficiency. They highlight that while Random Forests can offer superior predictive performance, they do so at a considerable computational cost. This analysis provides a clearer perspective for practitioners on the computational implications of choosing Random Forests over Decision Trees, especially in real-world applications where memory, CPU, and storage constraints are critical considerations.

4.2 Real-Life Applicability and Scalability

To assess the real-life applicability of Decision Trees and Random Forests, we applied these models to four distinct datasets, each representing a real-world scenario: Telco Customer Churn [27], Credit Card Fraud Detection [28], Crime Data from 2020 to Present [29] and Crash Reporting–Drivers Data [30]. Table 5 provides a comparative summary of model accuracy for each dataset, highlighting how each model performs under different conditions. This analysis underscores the practical implications of using these algorithms in production environments, where the accuracy (test and training sets), scalability, and efficiency of model predictions are critical.

Table 5: Real-life experimental results

Dataset	Method	Test accuracy	Train accuracy	Tree depth	Nodes
Telco customer churn	DT	77.30%	99.84%	40.36	1639.08
	DT-pruned	77.47%	99.53%	41.00	779.00
	RF	79.41%	99.84%	117.31	3841.03
Credit card	DT	99.92%	100%	21.00	297.28
	DT-pruned	99.93%	99.93%	21.00	141.00
	RF	99.96%	100%	23.28	299.54
Crime data	DT	79.26%	100%	5.00	27.00
	DT-pruned	78.95%	95.24%	5.00	27.00
	RF	63.16%	100%	6.73	30.52
Crash reporting	DT	56.89%	77.71%	65.00	66,957.08
	DT-pruned	56.85%	77.71%	65.00	65,827.00
	RF	56.62%	77.70%	62.73	54,562.04

4.2.1 Analysis of Crime Data from 2020 to Present

This study's analysis of the *Crime Data from 2020 to the Present [29]* provides insight into the performance and scalability of decision tree-based algorithms when applied to a dataset with real-world complexity. The chosen dataset, a comprehensive repository of crime incidents over a multi-year span, reflects the challenges inherent to large, constantly updating data streams typically encountered in production environments.

The decision tree (DT), pruned decision tree (DT-pruned), and random forest (RF) models were evaluated for accuracy, tree depth, and node count. Based on Table 5, the DT-pruned model achieved the highest accuracy (78.95%), followed by the standard DT model (75.37%), with the RF

model trailing at 63.16%. Despite RF's potential for high accuracy in many applications, the dataset's characteristics—such as temporal dependencies and sparse feature distribution—may have hindered its performance. These observations are crucial when considering model selection for large-scale implementations in crime data analysis.

With its controlled depth and reduced node complexity, the pruned decision tree demonstrates an advantage here, as it maintains a high accuracy with minimal complexity. This makes it a suitable choice for environments where computational resources are limited or where latency is a concern. For larger datasets or scenarios demanding near-real-time processing, deploying a random forest may prove more challenging without optimization techniques. Strategies such as parallelizing the tree-building process or using distributed computing frameworks could enhance the random forest model's applicability. However, this would require balancing model complexity with operational constraints.

4.2.2 Visual Comparison of Decision Tree Structures for Credit Card Fraud Detection

To further illustrate the impact of model complexity on computational efficiency and interpretability, Figs. 4 and 5 provide a visual comparison of the decision tree structure for the *Credit Card Fraud Detection* dataset before and after pruning. These visualizations highlight how pruning reduces tree depth and the number of nodes, simplifying the model while maintaining essential decision paths. Such visual comparisons offer insights into the practical trade-offs between model complexity and accuracy, enhancing our understanding of decision tree behavior in real-life applications.

4.2.3 Interpretability with LIME

While Random Forest models provide high accuracy, they are often criticized for their lack of interpretability. We applied the Local Interpretable Model-agnostic Explanations (LIME) method to the *Crash Reporting–Drivers Data* and *Telco Customer Churn* datasets to address this issue. LIME enables us to generate interpretable, instance-based explanations for model predictions by approximating the Random Forest with more straightforward, locally interpretable models. This approach reveals the influence of individual features on specific predictions, allowing us to understand and trust the model's outputs. Figs. 6 and 7 illustrate LIME explanations for sample instances from each dataset, demonstrating how feature contributions can be visualized and interpreted for end-users, thus mitigating some of the interpretability challenges associated with Random Forests.

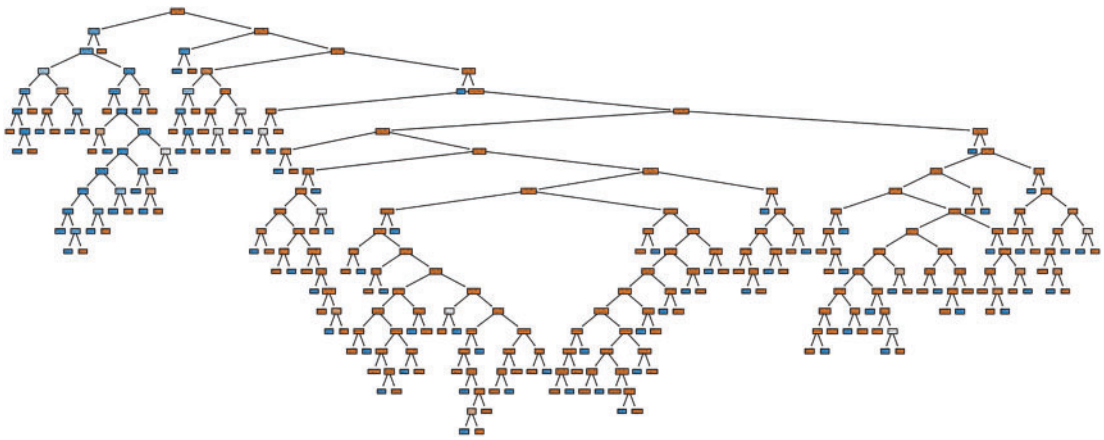


Figure 4: Decision tree before pruning

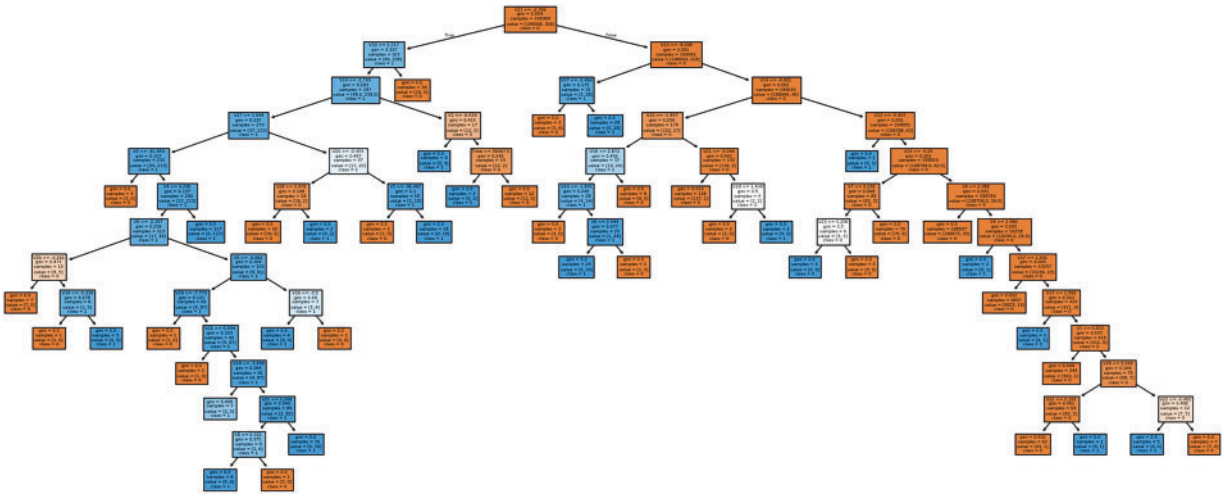


Figure 5: Decision tree after pruning

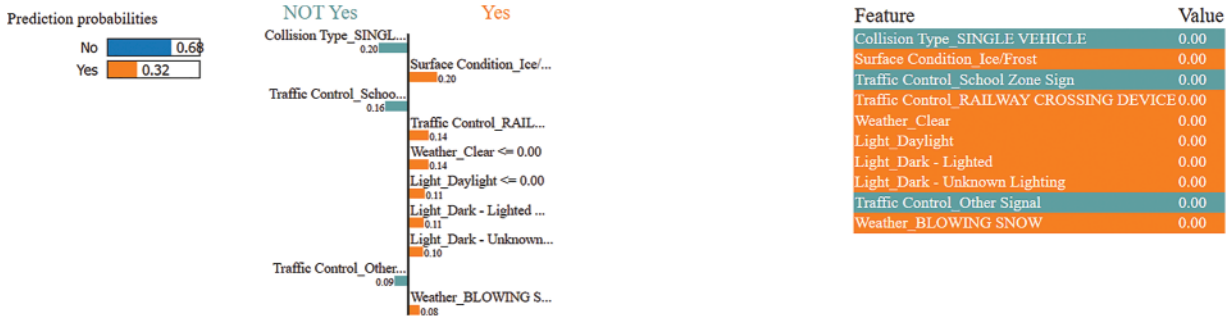


Figure 6: LIME model for crash reporting

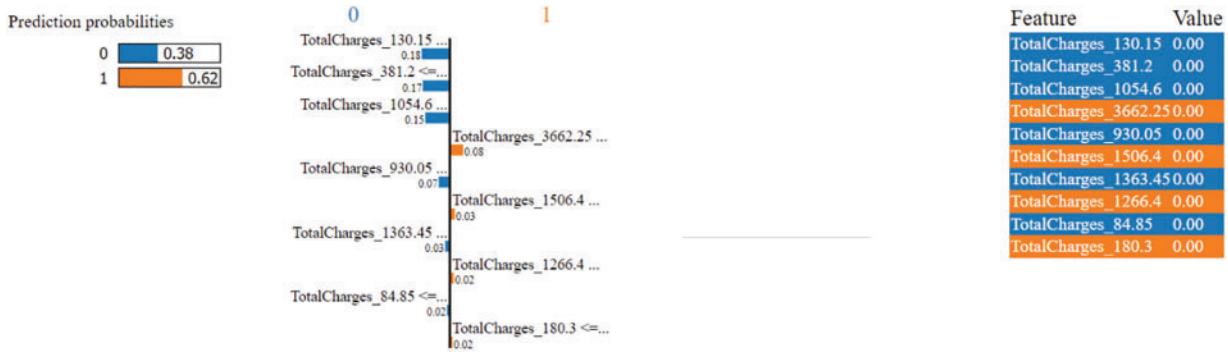


Figure 7: LIME model for telco customer churn

This LIME output provides an interpretation of how various features contribute to a model's prediction, which in this case assigns a 0.68 probability to "No" (likely indicating a low chance of an event, such as a collision or positive outcome) and a 0.32 probability to "Yes" (likely indicating the opposite). The features listed represent various conditions that the model considers in its decision-making process. The feature importance section shows that certain conditions, like *Collision Type_SINGLE*

VEHICLE and *Surface Condition_Ice/Frost*, have the highest influence on the model's prediction, with both weighted at 0.20. Other influential factors include *Traffic Control_School Zone Sign* and *Traffic Control_RAILWAY CROSSING DEVICE*, with weights of 0.16 and 0.14, respectively. These weights indicate that these features are significant in the model's interpretation, and higher weights suggest a more substantial impact on the prediction. In this specific instance, however, the values for each feature (such as collision type, surface condition, and weather) are all recorded as 0, implying that these particular conditions are not present. Despite the lack of these specific values in the instance, the LIME explanation reveals that these features are essential in the model's general decision-making process. The model leans towards predicting "No" (indicating a lower likelihood of "Yes" events) primarily based on these feature weights, which influence its understanding of conditions typically associated with different collision probabilities.

This LIME output explains the impact of *TotalCharges* values on the model's prediction, which has a probability of 0.62 for the class labeled "1" (suggesting a positive outcome) and 0.38 for the class labeled "0" (indicating a negative outcome). The feature importance section highlights specific *TotalCharges* values, such as 130.15, 381.2, and 1054.6, which contribute most to the prediction, with weights of 0.18, 0.17, and 0.15, respectively. These weights reflect the strength of each feature's influence, with higher weights suggesting a more substantial impact. Each *TotalCharges* feature is recorded as 0 for this instance, meaning the actual feature values do not fall within the specified influential ranges. Nonetheless, the LIME explanation reveals that these *TotalCharges* values still play a vital role in the model's overall decision-making, guiding it toward predicting class "1" in similar cases.

4.2.4 Train-Test Performance Comparison and Overfitting Analysis

Based on Table 5, this analysis provides a detailed comparison of training and test accuracies to highlight instances of overfitting across different datasets and models. The table illustrates each model's generalizability by showing the extent to which training accuracy surpasses test accuracy, underscoring the potential limitations in model performance on unseen data.

The results of our experiments provide insights into each model's generalizability and its tendency toward overfitting across four datasets. In the Telco Customer Churn dataset, both Decision Trees (DT) and Random Forest (RF) models display near-perfect accuracy on the training data (99.84%) but significantly lower accuracy on the test data (77%–79%). This pattern suggests that these models may have overfitted, capturing specific patterns within the training set that do not generalize well to unseen data. Overfitting is particularly pronounced in these models because of their complexity and propensity to memorize intricate details in the training data. By contrast, the Credit Card dataset shows high accuracy across both training and test sets with minimal disparity, implying that the models here generalize effectively without sacrificing performance. This outcome suggests that either the dataset characteristics support good generalization or the features are well-suited to the chosen models, reducing the likelihood of overfitting.

Overfitting in the Crime Data dataset becomes more evident, especially with RF, where train accuracy reaches 100% while test accuracy drops to 63.16%. The large gap between train and test performance illustrates how RF, a more complex model, can overfit when faced with potentially noisy or small datasets. Pruned Decision Trees perform slightly better in terms of generalization here, demonstrating the effectiveness of pruning in mitigating overfitting. In contrast, the Crash Reporting dataset yields low accuracies across both train and test sets (77.7% and 56%, respectively), indicating moderate overfitting but suggesting that the models struggle with this dataset overall. This pattern

highlights potential feature separability or data quality limitations, possibly necessitating further feature engineering or alternative model approaches. These comparisons illustrate the importance of balancing model complexity with dataset characteristics and validate the need for techniques that combat overfitting, particularly in scenarios where models must generalize well to new data in production environments.

5 Conclusions, Limitations, and Future Work

In conclusion, this study hypothesized that decision trees, due to their structure, would exhibit a higher degree of overfitting than random forests when applied to the same datasets. The experimental results confirm this hypothesis, demonstrating that unpruned decision trees are particularly prone to overfitting, as evidenced by their excessive tree depths and high node counts, especially in datasets like 20 Newsgroups, MNIST, and Covertype. The structural properties of decision trees, which allow them to create particular rules tailored to the training data, lead to a significant risk of overfitting, particularly in complex or noisy datasets.

The proposed pruning approach, the Adaptive Cross-Validated Alpha CCP (ACV-CCP), has proven to be an effective method for reducing overfitting by simplifying the model, as indicated by the reduction in tree depth and node count in pruned decision trees. Although this slightly decreases accuracy, it enhances the model's generalization ability, making it more applicable to new, unseen data. Even with pruning, decision trees often fall short of the performance and robustness of random forests, which generally outperform them across various datasets. Random forests mitigate overfitting through their ensemble approach, averaging the predictions of multiple decision trees to reduce variance and improve generalization. This is reflected in their superior accuracy and balanced complexity, making them particularly effective for large and complex datasets where overfitting is a significant concern.

In conclusion, while decision trees can be prone to overfitting, this issue can be mitigated through pruning or by employing more advanced ensemble methods like random forests. The model choice should consider the application's specific requirements, balancing the need for accuracy, interpretability, and computational efficiency. Random forests should be the preferred choice when robustness and performance are critical, while pruned decision trees offer a viable alternative when simplicity and interpretability are prioritized.

When we discuss the limitations regarding the usage of the classification algorithms, it is worth mentioning data imbalances in datasets. Random forests may not perform well on imbalanced datasets where some variables are heavily underrepresented. In such cases, the algorithm might be biased towards the majority class, affecting its ability to correctly classify instances of the smaller class. Also, the performance of classification algorithms can be affected by the settings of their hyperparameters. The number of sub-trees and the depth of each tree should be considered at each node. Finding the optimal configuration requires extensive cross-validation, which can be time-consuming and computationally expensive and presents quite a limitation on the usage of the algorithm.

On the other hand, while random forests are safer to overfit than individual decision trees, they can still overfit noisy data. If the noise level in the data is high, the random forest might learn the noise as if it were a pattern, especially with many trees. Addressing these limitations requires a grounded understanding of both the dataset being used and the classification algorithm itself; therefore, addressing these shortcomings becomes a priority.

One significant limitation is accurately measuring power consumption when working within Windows environments. Unlike Linux, where detailed power usage metrics are more accessible,

Windows lacks the same level of granularity for monitoring resource consumption, making it harder to quantify the exact computational cost in terms of energy efficiency. This gap limits the ability to fully assess the model's scalability in environments with energy constraints.

With the evolution of machine learning and the rapid development of classification algorithms, the possibilities for growth are limitless. With this field of study being so current, there is so much capacity for growth and reaching new unseen heights. Future research could focus more on exploring automated and more efficient hyperparameter tuning methods for random forests to reduce the manual effort and expertise required to find the optimal model configuration, possibly by utilizing machine learning techniques like Bayesian optimization; also, with the rise of big data to study the scalability of random forests in the context of big data, focusing on how these models can be adapted or improved to handle huge datasets efficiently.

Acknowledgement: Not applicable.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Erblin Halabaku, Eliot Bytyçi; data collection: Erblin Halabaku; analysis and interpretation of results: Erblin Halabaku; draft manuscript preparation: Erblin Halabaku, Eliot Bytyçi. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data supporting this study's findings are openly available on the Internet in free repositories.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

[1] A. Sharma and H. Wehrheim, "Higher income, larger loan? monotonicity testing of machine learning models," in *ISSTA 2020: Proc. 29th ACM SIGSOFT Int. Symp. Softw. Test. Anal.*, Jul. 2020, pp. 200–201. doi: 10.1145/3395363.3397352.

[2] J. Ma and X. Gao, "Designing genetic programming classifiers with feature selection and feature construction," *Appl. Soft Comput.*, vol. 97, 2020, Art. no. 106826. doi: 10.1016/j.asoc.2020.106826.

[3] Q. Cao *et al.*, "KSPMI: A knowledge-based system for predictive maintenance in Industry 4.0," *Robot. Comput.-Integrat. Manufact.*, vol. 74, 2022, Art. no. 102281. doi: 10.1016/j.rcim.2021.102281.

[4] I. K. Thajeel, K. Samsudin, S. J. Hashim, and F. Hashim, "Machine and deep learning-based XSS detection approaches: A systematic literature review," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 35, Jul. 2023, Art. no. 101628. doi: 10.1016/j.jksuci.2023.101628.

[5] Y. Bengio, A. Lodi, and A. Prouvost, "Machine learning for combinatorial optimization: A methodological tour d'horizon," *Eur. J. Oper. Res.*, vol. 290, no. 2, pp. 405–421, Aug. 2020. doi: 10.1016/j.ejor.2020.07.063.

[6] Z. Yang, A. Zhang, and A. Sudjianto, "GAMI-Net: An explainable neural network based on generalized additive models with structured interactions," *Pattern Recognit.*, vol. 120, Dec. 2021, Art. no. 108192. doi: 10.1016/j.patcog.2021.108192.

[7] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, no. 1, pp. 295–316, Nov. 2020. doi: 10.1016/j.neucom.2020.07.061.

[8] T. Lazebnik and S. Bunimovich-Mendrazitsky, "Decision tree post-pruning without loss of accuracy using the SAT-PP algorithm with an empirical evaluation on clinical data," *Data Knowl. Eng.*, vol. 145, May 2023, Art. no. 102173. doi: 10.1016/j.datak.2023.102173.

[9] D. Sisodia and D. S. Sisodia, "Prediction of diabetes using classification algorithms," *Procedia Comput. Sci.*, vol. 132, pp. 1578–1585, 2018. doi: 10.1016/j.procs.2018.05.122.

[10] A. Degerli, S. Kiranyaz, T. Hamid, R. Mazhar, and M. Gabbouj, "Early myocardial infarction detection over multi-view echocardiography," *Biomed. Signal Process. Control*, vol. 87, Jan. 2024, Art. no. 105448. doi: 10.1016/j.bspc.2023.105448.

[11] S. Calzavara, L. Cazzaro, G. Ermanno Pibiri, and N. Prezza, "Verifiable learning for robust tree ensembles," in *CCS'23: Proc. 2023 ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2023, pp. 1850–1864. doi: 10.1145/3576915.3623100.

[12] G. D. Teodoro, M. Monaci, and L. Palagi, "Unboxing Tree ensembles for interpretability: A hierarchical visualization tool and a multivariate optimal re-built tree," *EURO J. Comput. Optim.*, vol. 12, Jan. 2024, Art. no. 100084. doi: 10.1016/j.ejco.2024.100084.

[13] R. Hollós *et al.*, "Conditional interval reduction method: A possible new direction for the optimization of process based models," *Environm. Mod. Softw.*, vol. 158, Oct. 2022, Art. no. 105556. doi: 10.1016/j.envsoft.2022.105556.

[14] A. Quemy, "Two-stage optimization for machine learning workflow," Jan. 2019. doi: 10.48550/arxiv.1907.00678.

[15] J. Hao, S. Luo, and L. Pan, "Rule extraction from biased random forest and fuzzy support vector machine for early diagnosis of diabetes," *Sci. Rep.*, vol. 12, Jun. 2022, Art. no. 9858. doi: 10.1038/s41598-022-14143-8.

[16] L. Nizzoli, M. Avvenuti, M. Tesconi, and S. Cresci, "Geo-semantic-parsing: AI-powered geoparsing by traversing semantic knowledge graphs," *Decis. Support Syst.*, vol. 136, Jul. 2020, Art. no. 113346. doi: 10.1016/j.dss.2020.113346.

[17] H. Jafarbiglu and A. Pourreza, "A comprehensive review of remote sensing platforms, sensors, and applications in nut crops," *Comput. Electron. Agric.*, vol. 197, Jun. 2022, Art. no. 106844. doi: 10.1016/j.compag.2022.106844.

[18] V. -D. Ngo, T. -C. Vuong, T. Van Luong, and H. Tran, "Machine learning-based intrusion detection: Feature selection versus feature extraction," *Cluster Comput.*, vol. 27, no. 3, pp. 2365–2379, Jul. 2023. doi: 10.1007/s10586-023-04089-5.

[19] H. Martin, M. Acher, J. A. Pereira, and J. -M. Jézéquel, "A comparison of performance specialization learning for configurable systems," in *SPLC'21: Proc. 25th ACM Int. Syst. Softw. Prod. Line Conf.*, Sep. 2021, pp. 46– 57. doi: 10.1145/3461001.3471155.

[20] J. Hatwell, M. M. Gaber, and R. M. A. Azad, "CHIRPS: Explaining random forest classification," *Artif. Intell. Rev.*, vol. 53, no. 8, pp. 5747–5788, Jun. 2020. doi: 10.1007/s10462-020-09833-6.

[21] D. Couespel, J. Tjiputra, K. Johannsen, P. Vaittinada Ayar, and B. Jensen, "Machine learning reveals regime shifts in future ocean carbon dioxide fluxes inter-annual variability," *Commun. Earth Environ.*, vol. 5, Feb. 2024, Art. no. 9. doi: 10.1038/s43247-024-01257-2.

[22] J. -R. Cano, P. A. Gutiérrez, B. Krawczyk, M. Woźniak, and S. García, "Monotonic classification: An overview on algorithms, performance measures and data sets," *Neurocomputing*, vol. 341, no. 1, pp. 168–182, May 2019. doi: 10.1016/j.neucom.2019.02.024.

[23] S. Pal and S. Debanshi, "Machine learning models for wetland habitat vulnerability in mature Ganges delta," *Environ. Sci. Pollut. Res.*, vol. 28, no. 15, pp. 19121–19146, Jan. 2021. doi: 10.1007/s11356-020-11413-8.

[24] Z. S. Chen, P. P. Kulkarni, I. R. Galatzer-Levy, B. Bigio, C. Nasca and Y. Zhang, "Modern views of machine learning for precision psychiatry," *Patterns*, vol. 3, Nov. 2022, Art. no. 100602. doi: 10.1016/j.patter.2022.100602.

[25] M. Szeląg and R. Słowiński, "Explaining and predicting customer churn by monotonic rules induced from ordinal data," *Eur. J. Oper. Res.*, vol. 317, no. 2, pp. 414–424, Oct. 2023. doi: 10.1016/j.ejor.2023.09.028.

[26] "UCI machine learning repository," Accessed: Nov. 01, 2024. [Online]. Available: http://archive.ics.uci.edu/ml
[27] "Predict customer churn using watson machine learning and jupyter notebooks on cloud pak for data," Nov. 29, 2021. Accessed: Nov. 01, 2024. [Online]. Available: https://github.com/IBM/telco-customer-churn-on-icp4d/blob/master/data/Telco-Customer-Churn.csv
[28] Kaggle, "Credit card fraud detection," 2018. Accessed: Nov. 01, 2024. [Online]. Available: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud
[29] "Crime Data from 2020 to Present," Nov. 16, 2022. Accessed: Nov. 01, 2024. [Online]. Available: https://catalog.data.gov/dataset/crime-data-from-2020-to-present
[30] "Crash reporting-drivers data," Apr. 21, 2023. Accessed: Nov. 01, 2024. [Online]. Available: https://catalog.data.gov/dataset/crash-reporting-drivers-data