svmodt: An R Package for Linear SVM-Based Oblique Decision Trees

by Aneesh Agarwal, Jack Jewson, and Erik Sverdrup

Abstract Decision trees are widely used for classification tasks due to their interpretability, but traditional axis-aligned splits often require deep trees to approximate complex decision boundaries. We introduce svmodt, an R package implementing two an SVM-based oblique decision tree (SVMODT) that uses linear Support Vector Machines at each node to create multivariate splits. The package supports dynamic feature selection strategies, node-specific class weighting for handling imbalanced data, and feature diversity mechanisms through penalization. We demonstrate that SVMODT achieves competitive accuracy with standard SVMs while maintaining the interpretability advantages of tree structures. The package includes comprehensive visualization tools, detailed documentation, and is freely available on GitHub.

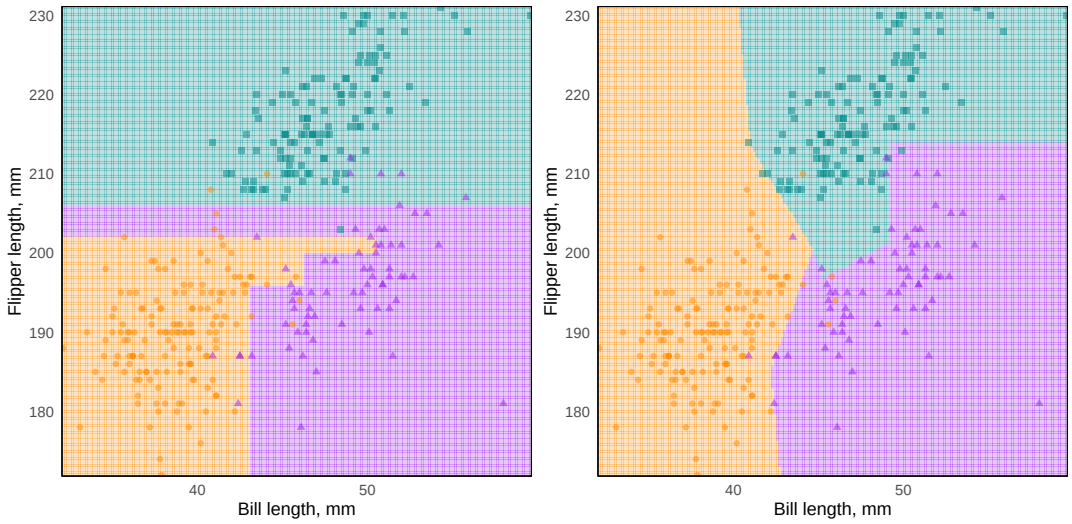
1 Introduction

Decision trees remain widely used in machine learning because they are interpretable, straightforward to apply, and can accommodate both categorical and numerical predictors with minimal preprocessing. Classical algorithms such as CART (Breiman et al., 1984) and C4.5 (Quinlan, 2014) employ axis-aligned (univariate) splits that partition the feature space along coordinate axes, thereby preserving transparency in the resulting decision rules. In R, traditional decision trees are readily implemented via packages such as **rpart**, **tree**, and **party**. Although axis-parallel decision trees offer high interpretability, they suffer representational limitations: they frequently grow unnecessarily large as they approximate complex relationships via multiple axis-aligned partitions, which can lead to replicated subtrees and repeated testing of the same feature along different paths, reducing efficiency and impairing generalization (Pagallo and Haussler, 1990). Oblique decision trees mitigate these limitations by permitting splits on linear combinations of features; for example, a single oblique test of the form $w_1x_1 + w_2x_2 < \theta$ can often replace multiple axis-aligned tests, producing more compact and geometrically simpler partitions of the feature space. Oblique trees are available in R through packages such as **ODRF**, **aorsf**, and **oblique.tree**. Empirical comparisons across diverse data sets indicate that oblique decision trees typically attain higher accuracy and smaller tree sizes than their axis-parallel counterparts, albeit at the expense of some interpretability. Importantly, oblique decision trees preserve key advantages of standard trees—sequential split evaluation and transparent decision procedures—while providing greater modeling flexibility for complex data sets (Koziol and Wozniak, 2009; Friedl and Brodley, 1997; Liu and Setiono, 1998; Cañete et al., 2021).

Figure 1 contrasts an axis-parallel decision tree with an axis-oblique decision tree. The axis-parallel tree displays (left panel) clear overfitting and a high rate of training misclassification, reflecting its inability to capture interactions among predictors that determine class membership. In contrast, the oblique tree (right panel) yields geometrically simpler decision boundaries and substantially fewer misclassifications on the training set, indicating a better fit to the underlying multivariate relationships. Together, these panels illustrate how oblique splits can more effectively represent linear combinations of features and thereby reduce spurious complexity and error in tree-based classifiers.

Support Vector Machines (SVMs), introduced by Cortes and Vapnik (1995), provide a principled way to obtain oblique splits by finding margin-maximizing hyperplanes. The idea of fitting SVMs at tree nodes was formalized by Bennett and Blue (1998) and extended to multiclass settings and other refinements in subsequent work (Takahashi and Abe, 2002; Bala and Agrawal, 2011; Ganaie et al., 2022).

Python's STree implements SVM-node oblique trees using scikit-learn's SVC (Montañana et al., 2021; Pedregosa et al., 2011), while Java tools such as Weka offer components that can be adapted for hierarchical SVM trees (Menkovski et al., 2008); however, fully featured, native R implementations remain limited. To address this gap we make two contributions. First, we provide an R implementation of an STree-style oblique tree that does not call Python, delivering a production-ready API and efficient, vectorized operations for R users. Second, we propose SVM-ODT, an enhanced SVM-based oblique decision tree that extends the STree algorithm with practical improvements which include feature subsetting, feature selection, feature penalization, node-specific class weighting to improve performance and robustness in both binary and multiclass settings. Together these contributions aim to bring the representational advantages of SVM-based oblique splits to R practitioners while addressing key computational and methodological limitations of existing approaches. We are exclusively going to focus on building a supervised learning algorithm capable of predicitng y from d-dimensional



(a) axis-parallel split **(b)** axis-oblique split

Figure 1: Comparison of Axis-parallel and Axis-oblique splits on Palmerpenguins dataset.

predictors x.

To address this methodological gap, we advance two primary contributions.
First, we introduce a fully native R implementation of an STree-style oblique decision tree that operates independently of Python. This implementation provides a production-ready API and leverages vectorized computation to ensure efficiency and seamless integration within the R ecosystem.

Second, we develop **SVM-ODT**, an enhanced SVM-based oblique decision tree that extends the original STree framework through several practical innovations. These include feature sub-setting, embedded feature selection, feature penalization, and node-specific class weighting. Collectively, these enhancements are designed to improve predictive performance, interpretability, and robustness across both binary and multiclass classification settings.

Together, these contributions aim to make the representational strengths of SVM-driven oblique splits accessible to R programmers while simultaneously addressing key computational and methodological limitations present in existing approaches. Throughout this work, we focus exclusively on the supervised learning problem of predicting y from d-dimensional predictors x, where the goal is to learn a decision function that maps $x \in \mathbb{R}^d \longrightarrow y$.

2 Background

Decision Trees

Decision Trees (DTs) are interpretable classification models that represent their decision-making process through a hierarchical, tree-like structure. This structure comprises of internal nodes containing splitting criteria and terminal (leaf) nodes corresponding to predicted outcomes (class labels and probabilities). The nodes are connected by directed edges, each representing a possible outcome of a splitting criterion. Formally, a DT can be expressed as a rooted, directed tree $T = (G(V, E), v_1)$, where V denotes the set of nodes, E represents the set of edges linking these nodes, and v_1 is the root node.

If the tree T has m nodes, then for any $j \in \{1, \ldots, m\}$, the set of child nodes of $v_j \in V$ can be defined as:

$$N^+(v_j) = \{v_k \in V \mid k \in \{1, \ldots, m\},\ k \neq j,\ (v_j, v_k) \in E\}.$$

Here, $N^+(v_j)$ denotes the set of nodes that are directly connected to v_j through outgoing edges, representing two subsequent child nodes that can be reached from v_j within the tree structure (Rivera-Lopez and Canul-Reich, 2018).

Decision tree algorithms can be categorized based on whether the same type of test is applied at all internal nodes. **Homogeneous trees** employ a single algorithm throughout (e.g., univariate or multivariate splits), whereas **hybrid trees** allow different algorithms such as linear discriminant functions, k-nearest neighbors, or univariate splits that can be used in different sub-trees (Brodley and

Utgoff, 1995). Hybrid trees exploit the principle of *selective superiority*, allowing subsets of the data to be modeled by the most appropriate classifier, thereby improving flexibility and accuracy.

Axis-Parallel Decision Trees

Axis-Parallel Decision trees represent hyper-planes dividing the instance space into several disjoint axis-parallel regions. Axis-parallel decision trees, such as CART and C4.5, represent two of the most widely used algorithms for classification tasks. The **CART (Classification and Regression Trees)** algorithm employs a binary recursive partitioning procedure at capable of handling both continuous and categorical variables as predictors or targets. At each node, the algorithm systematically evaluates every available variable and its potential split points to determine the optimal partition. In R, traditional decision trees are readily implemented via packages such as **rpart**, **tree**, and **party**.

In contrast, **C4.5**, an extension of the earlier **ID3** algorithm (Quinlan, 1986), utilizing information theory measures such as **information gain** and **gain ratio** to select the most informative attribute for each split (Quinlan, 2014). C4.5 also includes mechanisms to handle missing attribute values by weighting instances according to the proportion of known data and employs an **error-based pruning** method to reduce overfitting. Although these techniques are effective across diverse data sets, studies have shown that the choice of pruning strategy and stopping criteria can significantly affect model performance across different domains (Mingers, 1989; Schaffer, 1992).

While axis-parallel decision tree are highly interpretability, they have several representational limitations. Such trees often grow unnecessarily large, as they must approximate complex relationship between features through multiple axis-aligned partitions. This can result in the replication of sub-trees and repeated testing of the same feature along different paths, both of which reduce efficiency and hinder generalization performance (Pagallo and Haussler, 1990).

Axis-Oblique Decision Trees

Axis-oblique decision trees extends axis-parallel decision trees by allowing each internal node to perform splits based on linear or nonlinear combinations (Chabbouh et al., 2025) of multiple features. This flexibility enables the tree to form oblique decision boundaries that more accurately partition the instance space. For example, a single multivariate test such as $x + y < 8$ can replace multiple univariate splits needed to approximate the same boundary. Oblique trees are available in R through packages such as **ODRF**, **aorsf**, and **oblique.tree**.

The construction of Axis-oblique decision trees introduces several design considerations, including how to represent multivariate tests, determine their coefficients, select features to include, handle symbolic and missing data, and prune to avoid over-fitting (Brodley and Utgoff, 1995). Various optimization algorithms such as recursive least squares (Young, 1984), the pocket algorithm (Gallant, 1986), or thermal training (Frean, 1990) may be used to estimate the weights. However, Axis-oblique decision trees trade interpretability for representational power and often require additional mechanisms for **local feature selection**, such as *sequential forward selection* (SFS) or *sequential backward elimination* (SBE) (Kittler, 1986).

Empirical comparisons across multiple data sets demonstrate that multivariate trees generally achieve higher accuracy and smaller tree sizes than their univariate counterparts, though this comes at the cost of reduced interpretability. Moreover, MDTs retain key advantages of standard decision trees such as sequential split criteria evaluation and transparent decision procedures while offering improved modeling flexibility for complex data sets (Koziol and Wozniak, 2009; Friedl and Brodley, 1997; Liu and Setiono, 1998; Cañete et al., 2021).

Support Vector Machines (SVMs)

Support Vector Machines (SVMs) are powerful supervised learning models used for classification and regression tasks. They aim to determine an optimal separating hyperplane that maximizes the margin between binary classes in the data. This margin-based approach enhances the generalization ability of the model, making SVMs robust and effective for many real-world problems (Cristianini, 2000). A simplest **linear SVMs** construct a separating hyperplane in an p-dimensional space such that the margin between the classes is maximized. Given a training dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, where $\mathbf{x}_i \in \mathbb{R}^p$ and $y_i \in \{-1, +1\}$, the decision function is defined as $f(\mathbf{x}) = \text{sign}(\mathbf{w}^\top \mathbf{x} + b)$. The optimal hyperplane is the one that maximizes the distance between the closest points of each class (the **support vectors**) and the hyperplane itself (Cortes and Vapnik, 1995). While linear classifiers provide useful insights, they are often inadequate for real-world data sets, where classes are not linearly separable. In such cases, SVMs can be extended to create **nonlinear decision boundaries** by mapping the input vectors

into a higher-dimensional **feature space** using a nonlinear transformation $\phi : \mathbb{R}^n \to \mathcal{F}$. The linear transformation is then achieved in the transformed space using $f(\mathbf{x}) = \text{sign}(\mathbf{w}^\top \phi(\mathbf{x}) + b)$.

Although SVMs exhibit strong theoretical foundations and robust generalization capabilities, they present several practical limitations. Model performance is highly dependent on the appropriate selection of hyperparameters such as the regularization term (C) and kernel parameters (e.g., γ), which govern the trade-off between margin maximization and misclassification tolerance (Nanda et al., 2018). Training an SVM requires solving a quadratic programming (QP) optimization problem involving an $n \times n$ kernel matrix, where n denotes the number of training samples, leading to quadratic growth in both computational time and memory usage (Dong et al., 2005). This makes SVMs computationally expensive for large-scale data sets. Moreover, SVMs are inherently designed for binary classification, necessitating decomposition strategies such as One-vs-One and One-vs-All for multi-class problems (Hsu and Lin, 2002). Their performance also tends to degrade in imbalanced data settings, where the decision boundary becomes biased toward the majority class (Cervantes et al., 2020).

Projection Pursuit Trees

Projection pursuit methods (Friedman and Tukey, 2006; Kruskal, 1969) have been used to uncover low-dimensional projections that reveal structure in high-dimensional data; the **PPtree** extends this idea into a recursive partitioning framework by optimizing a class-sensitive projection index at each node and then applying conventional split criteria on the resulting one-dimensional projection. This hybrid design allows the model to exploit multivariate correlations that axis-aligned trees often miss, while preserving the interpretability and hierarchical structure of tree models. In addition to improved separation in many settings, the PPtree produces node-specific projection coefficients that serve as transparent indicators of which variables drive each split, making it useful both for classification and for feature selection in multivariate problems.

Hybrid Decision Trees with Support Vector Machines

Support Vector Machine–based decision trees (DTSVMs) were first formalized by Bennett and Blue (1998), who adapted Statistical Learning Theory and Structural Risk Minimization to construct binary decision trees in which each internal node is an SVM that partitions the data by an optimal hyperplane. This formulation enabled multivariate, margin-based decisions at every split. Takahashi and Abe (2002) extended the idea to multiclass settings by using a recursive partitioning strategy: the root node isolates the most separable class or classes from the remainder, and the procedure recurses until each leaf contains a single class, thereby covering the entire feature space.

Scalability and optimization advances

Subsequent studies have built upon this foundation to address scalability, optimization, and generalization issues. Optimal Decision Tree SVM (ODT-SVM) (Bala and Agrawal, 2011) introduced split-selection criteria based on Gini index, information gain, and scatter matrix separability to balance tree interpretability and margin-based precision.

Oblique and Rotation-Based Extensions

Recent research generalizes DTSVMs through oblique splits and rotation-based ensemble strategies. Oblique Double Random Forests with MPSVM (MPDRaF) (Ganaie et al., 2022) implement multivariate (oblique) node splits using Multi-Plane SVM (MPSVM) formulations (Mangasarian and Wild, 2006), increasing the geometric flexibility of decision boundaries. These methods incorporate regularization variants (Tikhonov, axis-parallel, null-space) to mitigate small-sample issues at deeper nodes and improve generalization. Rotation-based Double Random Forests (DRaF) (Ganaie et al., 2022) apply PCA and LDA transforms at non-terminal nodes to generate diverse subspaces, enhancing ensemble diversity and stabilizing classification performance.

Modern multi-class integration

More recent approaches embed multiple SVM classifiers directly within each split to handle multi-class problems more efficiently. The Oblique Decision Tree Ensemble (ODTE) (Montañana et al., 2025) places one-vs-one or one-vs-rest SVMs at splits and dynamically selects the classifier that minimizes class impurity, enabling n-ary decisions within a single tree and addressing scalability and class-imbalance limitations inherent to binary SVM tree extensions.

Synthesis and implications for future work

Across these developments, two recurring themes emerge: (1) leveraging multivariate, oblique decision boundaries to capture complex class geometry, and (2) combining transformation or ensemble strategies (rotations, multiple SVMs) to improve diversity and robustness. These trends point toward hybrid architectures that balance interpretability, margin-based generalization, and computational tractability—an agenda that motivates continued exploration of regularization schemes, split-selection heuristics, and efficient multi-class integration in SVM-based tree models.

3 STree Alogrithm

STree (Montañana et al., 2021) is an oblique multi-class decision tree algorithm that integrates support vector machines (SVMs) to construct single margin-based splits capable of handling multiclass problems. Unlike traditional multi-class tree methods that rely on clustering or ensembles of binary models, STree builds a single decision tree. Its central innovation lies in using SVM-derived hyperplanes at each internal node, enabling more expressive splits than axis-aligned trees while also being interpretable and computationally simple.

Methodology

The algorithm generates a binary tree recursively (1) . At each node:

1) Stopping conditions are evaluated to check the depth of the tree and the class purity. If the stopping conditions are met, a leaf node is created, labelled with the most frequent class in the node.
2) If the stopping conditions haven't been met the algorithm then selects the best hyperplane to split the data into two partitions: T^+ (positive side) and T^- (negative side).

When all instances at a node belong to more than two classes ($k' > 2$), the approach generates multiple candidate binary splits: For each class label y_i, it constructs a one-vs-rest binary problem: Class y_i vs. all other classes. It trains an SVM for each of these k' problems, resulting in k' hyperplanes H_i. Each hyperplane partitions the data into T_i^+ and T_i^-. The algorithm computes the impurity (using Shannon entropy) of the class distribution within each partition. When only two classes remain, STree trains a single SVM to obtain the maximum-margin hyperplane. Instances are then routed left or right based on the sign of their distance to this hyperplane. Stree's performance depends heavily on hyperparameter tuning, including kernel choice (linear, polynomial or Gaussian), gamma, polynomial degree, regularization parameter C, and max iteration.

Algorithm 1 STree (Multi-class SVM-based oblique decision tree)

Input: $\mathcal{D}' = \{(\vec{x}_i, y_i)\}_{i=1}^{l'}$: Data
Output: *tree*: the root node of an SVM-based oblique decision tree
function `STree`($\mathcal{D}'$)**:**

```
if stopping_condition({y_i}_{i=1}^t) then
    return create_leaf_node(mode({y_i}_{i=1}^t))                    // Leaf node
k' ← num_different_labels({y_i}_{i=1}^t)
    r ← k'(k'−1)/2
    Y' ← {y'_1, ..., y'_{k'}}                                       // set of labels
I_all ← I({y_i}_{i=1}^t)                          // I(·) is an information theory measure
if (OvO) then
    n_models ← r
else
    n_models ← k'                                                   // OvR
for j = 1 to n_models do
    if (j = k' = 2) then
        break                                          // Two labels, one SVM is enough
    if (OvO) then
        Let c_a and c_b the pair of labels corresponding to index j
        models[j] ← SVM(D'↓c_a ∪ D'↓c_b)
    else
        models[j] ← SVM(D') using binary class y'_j (+) vs rest (–)  // OvR
D'+, D'− ← models[j](x⃗_j)   ∀(x⃗) ∈ D'
I_j ← (|D'+|/|D'|) I({y_i}_{i=1}^{|D'+|}) + (|D'−|/|D'|) I({y_i}_{i=1}^{|D'−|})
IG_j ← I_all − I_j

b* = arg max_{j=1,...,n_models} IG_j

if IG_{b*} > 0 then
    node ← create_node(models[b*])
    node.left ← STree(D'+)
    node.right ← STree(D'−)
else
    return create_leaf_node(model({y_i}_{i=1}^t))                   // No split gain
return node
```

4 Experiment

Table 1

5 Acknowledgements

The authors would like to thank Professor **Natalia Da Silva** and Professor **Dianne Helen Cook** for their constructive discussions and valuable insights regarding oblique decision trees and their implementations in R. The authors also acknowledges the use of OpenAI's GPT-5 model (ChatGPT) for editorial assistance and technical writing support during the preparation of this manuscript. All conceptual development, analysis, data processing, coding, and interpretation of results were performed independently by the author. The AI tool was used solely to refine language clarity, improve structure, and ensure stylistic consistency in the documentation.

Table 1: Comparing Mean Prediction Accuracy with default arguments - STree(R), STree(Python), SVMODT

Dataset	Stree(R)	STree(Python)	SVMODT
WDBC Diagnosis	**0.970 ± 0.004**	**0.970 ± 0.003**	**0.970 ± 0.004**
Iris	**0.966 ± 0.006**	0.965 ± 0.010	0.965 ± 0.004
Echocardiogram	**0.846 ± 0.004**	0.845 ± 0.008	0.845 ± 0.007
Fertility	0.876 ± 0.011	0.874 ± 0.010	**0.881 ± 0.000**
Wine	**0.978 ± 0.008**	0.959 ± 0.008	0.975 ± 0.011
Cardiotography-3	0.901 ± 0.003	**0.903 ± 0.003**	0.898 ± 0.003
Cardiotography-10	0.763 ± 0.005	**0.795 ± 0.018**	0.760 ± 0.008
Ionosphere	**0.896 ± 0.009**	0.892 ± 0.015	**0.896 ± 0.007**
Dermatology	**0.972 ± 0.006**	0.959 ± 0.004	0.967 ± 0.006
Statlog Australian Credit	**0.678 ± 0.000**	**0.678 ± 0.000**	**0.678 ± 0.000**

Bibliography

M. Bala and R. Agrawal. Optimal decision tree based multi-class support vector machine. *Informatica*, 35(2), 2011. [p1, 4]

K. Bennett and J. Blue. A support vector machine approach to decision trees. In *1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98CH36227)*, volume 3, pages 2396–2401 vol.3, 1998. doi: 10.1109/IJCNN.1998.687237. [p1, 4]

L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Wadsworth, 1984. URL http://lyle.smu.edu/~mhd/8331f06/cart.pdf. [p1]

C. E. Brodley and P. E. Utgoff. Multivariate decision trees. *Machine Learning*, 19:45–77, 1995. URL https://api.semanticscholar.org/CorpusID:16836572. [p2, 3]

L. Cañete, R. Monroy, and M. Medina-Pérez. A review and experimental comparison of multivariate decision trees. *IEEE Access*, PP:1–1, 08 2021. doi: 10.1109/ACCESS.2021.3102239. [p1, 3]

J. Cervantes, F. Garcia-Lamont, L. Rodríguez-Mazahua, and A. Lopez. A comprehensive survey on support vector machine classification: Applications, challenges and trends. *Neurocomputing*, 408:189–215, 2020. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2019.10.118. URL https://www.sciencedirect.com/science/article/pii/S0925231220307153. [p4]

M. Chabbouh, S. Bechikh, E. Mezura-Montes, and L. Ben Said. Evolutionary optimization of the area under precision-recall curve for classifying imbalanced multi-class data. *Journal of Heuristics*, 31, 01 2025. doi: 10.1007/s10732-024-09544-z. [p3]

C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995. [p1, 3]

N. Cristianini. An introduction to support vector machines and other kernel-based learning methods, 2000. [p3]

J.-x. Dong, A. Krzyzak, and C. Y. Suen. Fast svm training algorithm with decomposition on very large data sets. *IEEE transactions on pattern analysis and machine intelligence*, 27(4):603–618, 2005. [p4]

M. R. Frean. *Small nets and short paths: Optimising neural computation*. PhD thesis, 1990. [p3]

M. Friedl and C. Brodley. Decision tree classification of land cover from remotely sensed data. *Remote Sensing of Environment*, 61(3):399–409, 1997. ISSN 0034-4257. doi: https://doi.org/10.1016/S0034-4257(97)00049-7. URL https://www.sciencedirect.com/science/article/pii/S0034425797000497. [p1, 3]

J. H. Friedman and J. W. Tukey. A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on computers*, 100(9):881–890, 2006. [p4]

S. I. Gallant. Optimal linear discriminants. *Eighth International Conference on Pattern Recognition*, pages 849–852, 1986. URL https://cir.nii.ac.jp/crid/1573668924476144256. [p3]

M. Ganaie, M. Tanveer, P. Suganthan, and V. Snasel. Oblique and rotation double random forest. *Neural Networks*, 153:496–517, 2022. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2022.06.012. URL https://www.sciencedirect.com/science/article/pii/S0893608022002258. [p1, 4]

C.-W. Hsu and C.-J. Lin. A comparison of methods for multiclass support vector machines. *IEEE transactions on Neural Networks*, 13(2):415–425, 2002. [p4]

J. Kittler. Feature selection and extraction. *Handbook of Pattern Recognition and Image Processing*, pages 59–83, 1986. URL https://cir.nii.ac.jp/crid/1573950400671608448. [p3]

M. Koziol and M. Wozniak. *Multivariate Decision Trees vs. Univariate Ones*, pages 275–284. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-540-93905-4. doi: 10.1007/978-3-540-93905-4_33. URL https://doi.org/10.1007/978-3-540-93905-4_33. [p1, 3]

J. B. Kruskal. Toward a practical method which helps uncover the structure of a set of multivariate observations by finding the linear transformation which optimizes a new "index of condensation". In *Statistical computation*, pages 427–440. Elsevier, 1969. [p4]

H. Liu and R. Setiono. Feature transformation and multivariate decision tree induction. In S. Arikawa and H. Motoda, editors, *Discovey Science*, pages 279–291, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg. ISBN 978-3-540-49292-4. [p1, 3]

O. Mangasarian and E. Wild. Multisurface proximal support vector machine classification via generalized eigenvalues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(1):69–74, 2006. doi: 10.1109/TPAMI.2006.17. [p4]

V. Menkovski, I. T. Christou, and S. Efremidis. Oblique decision trees using embedded support vector machines in classifier ensembles. In *2008 7th IEEE International Conference on Cybernetic Intelligent Systems*, pages 1–6, 2008. doi: 10.1109/UKRICIS.2008.4798937. [p1]

J. Mingers. An empirical comparison of pruning methods for decision tree induction. *Machine Learning*, 4:227–243, 01 1989. doi: 10.1023/A:1022604100933. [p3]

R. Montañana, J. A. Gámez, and J. M. Puerta. Stree: A single multi-class oblique decision tree based on support vector machines. In E. Alba, G. Luque, F. Chicano, C. Cotta, D. Camacho, M. Ojeda-Aciego, S. Montes, A. Troncoso, J. Riquelme, and R. Gil-Merino, editors, *Advances in Artificial Intelligence*, pages 54–64, Cham, 2021. Springer International Publishing. ISBN 978-3-030-85713-4. [p5]

R. Montañana, J. A. Gámez, and J. M. Puerta. Stree: a single multi-class oblique decision tree based on support vector machines, 2021. [p1]

R. Montañana, J. A. Gámez, and J. M. Puerta. Odte—an ensemble of multi-class svm-based oblique decision trees. *Expert Systems with Applications*, 273:126833, 2025. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2025.126833. URL https://www.sciencedirect.com/science/article/pii/S0957417425004555. [p4]

M. Nanda, K. Seminar, D. Nandika, and A. Maddu. A comparison study of kernel functions in the support vector machine and a comparison study of kernel functions in the support vector machine and its application for termite detection, 2018. [p4]

G. Pagallo and D. Haussler. Boolean feature discovery in empirical learning. *Machine Learning*, 5:71–99, 1990. URL https://api.semanticscholar.org/CorpusID:5661437. [p1, 3]

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. [p1]

J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986. URL https://api.semanticscholar.org/CorpusID:189902138. [p3]

J. R. Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014. [p1, 3]

R. Rivera-Lopez and J. Canul-Reich. Construction of near-optimal axis-parallel decision trees using a differential-evolution-based approach. *IEEE Access*, PP:1–1, 01 2018. doi: 10.1109/ACCESS.2017.2788700. [p2]

C. Schaffer. Deconstructing the digit recognition problem. In D. Sleeman and P. Edwards, editors, *Machine Learning Proceedings 1992*, pages 394–399. Morgan Kaufmann, San Francisco (CA), 1992. ISBN 978-1-55860-247-2. doi: https://doi.org/10.1016/B978-1-55860-247-2.50056-5. URL https://www.sciencedirect.com/science/article/pii/B9781558602472500565. [p3]

F. Takahashi and S. Abe. Decision-tree-based multiclass support vector machines. volume 3, pages 1418 – 1422 vol.3, 12 2002. ISBN 981-04-7524-1. doi: 10.1109/ICONIP.2002.1202854. [p1, 4]

P. C. Young. Recursive estimation and time-series analysis: An introduction. 1984. URL https://api.semanticscholar.org/CorpusID:60181335. [p3]

Aneesh Agarwal
Monash University

aaga0022@student.monash.edu

Jack Jewson
Monash University
Department of Econometrics and Business Statistics, Monash University, Australia
Jack.Jewson@monash.edu

Erik Sverdrup
Monash University
Department of Econometrics and Business Statistics, Monash University, Australia
Erik.Sverdrup@monash.edu