# Essential Machine Learning Models: Comprehensive Study of Decision Trees and Regression Models

**Author: Vigneshwar Rangini**
Organization: Capgemini America, USA

**Abstract:**

Machine learning plays an essential role in modern data-driven decision systems, where accurate prediction, model interpretability, and computational efficiency are critical, reflecting the foundational principles established in statistical learning theory by Vapnik and Mitchell [1], [2]. This paper presents a comprehensive study of two fundamental supervised learning families regression models and decision-tree algorithms along with their practical implementation in the Scikit-Learn framework [14]. The regression component covers linear, polynomial, and regularized models including Ridge, Lasso, and Elastic Net, highlighting their mathematical foundations introduced by Gauss and Legendre [3], [4] and expanded through modern regularization techniques developed by Hoerl and Kennard, Tibshirani, and Zou and Hastie [5],[7]. The decision-tree section examines classification and regression trees through impurity-based splitting mechanisms, such as Gini, entropy, and variance reduction, grounded in the seminal work of Hunt et al. [8], Quinlan's ID3 and C4.5 algorithms [9], [10], and the CART methodology introduced by Breiman et al. [11]. It further evaluates pruning techniques particularly cost-complexity pruning used to address overfitting and improve generalization performance. Comparative analysis illustrates the structural and functional differences between linear models and decision trees, emphasizing their complementary strengths in handling linear trends, nonlinear patterns, and heterogeneous feature interactions, consistent with insights from modern ML literature [16], [17]. Scikit-Learn's unified API is showcased as a versatile platform that streamlines model construction, tuning, and evaluation, supporting reproducible and scalable ML workflows [14]. Overall, this work provides a structured and accessible framework for understanding, comparing, and applying regression and decision-tree models, offering practical insights for developing robust and interpretable machine learning solutions.

**Keywords:**

Machine Learning, Supervised Learning, Regression Models, Decision Tree Algorithms, Ridge, Lasso, and Elastic Net Regression, Impurity Measures, Cost-Complexity Pruning (CCP), Scikit-Learn (sklearn), Model Evaluation and Comparison.

## Introduction:

Machine learning (ML) has emerged as a central component of modern computational systems, enabling automated pattern discovery, predictive modeling, and informed decision-making across diverse application domains, reflecting foundational principles in statistical learning theory described by Vapnik and Mitchell [1], [2]. As organizations increasingly rely on data-driven intelligence, the need for models that are both powerful and interpretable has accelerated the adoption of classical supervised learning techniques. Among these, regression models and decision tree algorithms remain foundational due to their conceptual clarity, analytical strength, and practical versatility, supported by decades of statistical and algorithmic development [3],[11].

Regression models including linear, polynomial, and regularized variants such as Ridge, Lasso, and Elastic Net provide mathematically grounded mechanisms for understanding relationships between variables and predicting continuous outcomes. These models build upon the early work of Gauss and Legendre on least squares estimation [3], [4] and extend through modern regularization techniques developed by Hoerl and Kennard, Tibshirani, and Zou and Hastie [5]–[7]. Their statistical rigor, computational efficiency, and adaptability make them indispensable in fields such as finance, healthcare, environmental modeling, and engineering analytics. However, these models are constrained by assumptions of linearity and global patterns, which may limit performance when data exhibits complex or nonlinear behavior limitations widely discussed in modern machine learning literature [16], [17].

Decision tree models offer a complementary approach by capturing nonlinear relationships, hierarchical feature interactions, and heterogeneous data patterns through a series of interpretable, rule-based splits. Their theoretical foundation originates from early induction research by Hunt et al. [8] and was further formalized through the ID3 and C4.5 algorithms proposed by Quinlan [9], [10], as well as the CART methodology introduced by Breiman et al. [11]. Their ability to model interactions without explicit feature engineering and to operate on mixed data types has led to widespread adoption across classification, regression, risk modeling, fraud detection, and decision support systems. Modern extensions including pruning strategies and ensemble methods such as Random Forests and Gradient Boosted Trees [12], [13] further enhance predictive performance and generalization.

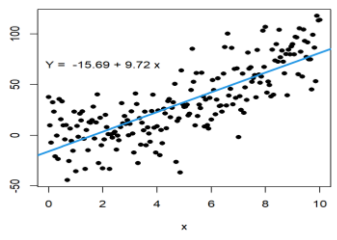
The Scikit-Learn (sklearn) framework provides a unified computational environment that operationalizes both regression and decision-tree methodologies through efficient algorithms, preprocessing utilities, and model-selection tools. Its consistency, modularity, and integration with the broader Python scientific ecosystem make it a leading platform for both academic research and real-world machine learning development, as documented by Pedregosa et al. [14].

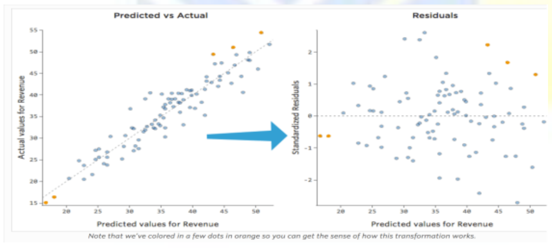
This paper presents a structured examination of regression and decision-tree models, outlining their theoretical foundations, mathematical formulations, splitting criteria, regularization techniques, pruning strategies, and practical implementation using Scikit-Learn. Through comparative analysis, the study highlights the strengths, limitations, and appropriate use cases for each model family, providing practitioners and researchers with a comprehensive framework for selecting and deploying robust, interpretable machine learning solutions an objective aligned with contemporary discussions of model transparency and interpretability [15], [16].

**Literature:**

## 1.Regression Model Overview

Regression is one of the most fundamental supervised learning techniques used to model the relationship between **independent variables** (features) and a **dependent variable** (target). Regression models estimate how changes in input variables influence the output, making them essential for forecasting, trend analysis, and quantitative decision-making in diverse domains such as finance, healthcare, engineering, and social sciences.





At its core, regression attempts to fit a function $f(X)$ that best predicts a continuous outcome y. The most widely used form **Linear Regression** assumes a linear relationship between inputs and outputs.

**1.1. Simple Linear Regression:** Simple linear regression models the relationship between a single independent variable x and a dependent variable y.

**Model Equation**

$y=\beta 0+\beta 1x+\epsilon$

Where: y: dependent variable

x: independent variable
β0: intercept
β1: slope coefficient
ϵ: error term

Goal: Minimize the Sum of Squared Errors (SSE)

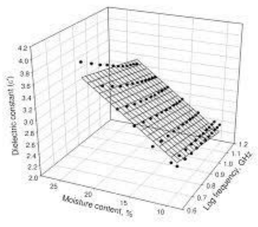
$$\mathrm{SSE} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

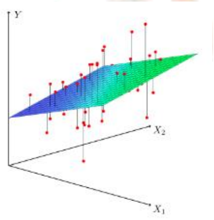
Optimal Coefficients (Closed-form solution)

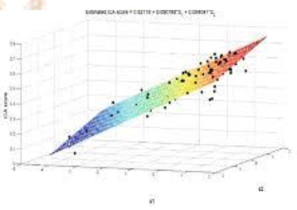
$$\beta_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$\beta_0 = \bar{y} - \beta_1\bar{x}$$

**1.2. Multiple Linear Regression:** Multiple Linear Regression (MLR) is an extension of Simple Linear Regression that models the relationship between a dependent variable and two or more independent variables. It is one of the most widely used statistical and machine learning techniques because real-world outcomes are often influenced by multiple factors rather than a single predictor.







When there are **multiple predictors** $x_1, x_2, \ldots, x_p$, the model generalizes as:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \epsilon$$

Matrix formulation:

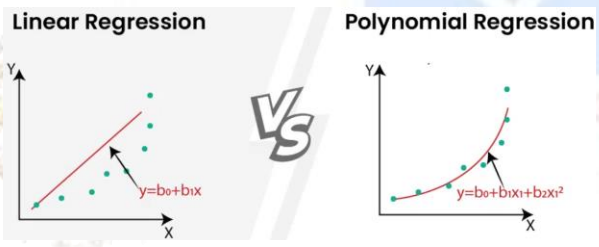
$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

Optimal parameters:

$$\boldsymbol{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

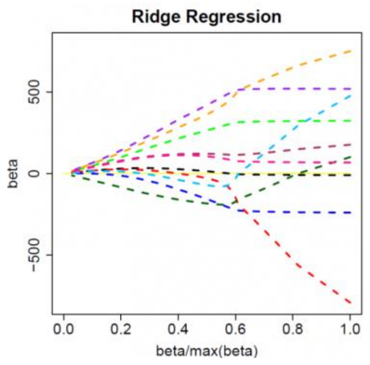
**2. Types of Regression Models**

**2.1. Polynomial Regression:** Polynomial Regression is an extension of Linear Regression used to model **nonlinear relationships** between an independent variable x and a dependent variable y. Although the relationship between variables is nonlinear in behavior, the model remains **linear in parameters**, allowing the use of Ordinary Least Squares (OLS) for estimation. This flexibility makes Polynomial Regression a powerful tool for capturing curves, trends, and complex patterns that simple linear regression cannot model.



**2.2 Ridge Regression (L2 Regularization):** Ridge Regression is a powerful extension of Ordinary Least Squares (OLS) designed to address multicollinearity, model instability, and overfitting. In traditional linear regression, when predictor variables are highly correlated or when the number of predictors is large relative to observations, the OLS solution becomes unstable, resulting in large coefficients and poor generalization. Ridge Regression overcomes this by adding an L2 penalty that shrinks the size of coefficients and stabilizes the model.

Ridge Regression is widely used in high-dimensional prediction problems, such as genomics, finance, image analysis, and machine-learning pipelines where collinearity and variance reduction are critical.



Adds a penalty to reduce overfitting when features are correlated.

$$\hat{\boldsymbol{\beta}} = \arg\min_{\beta} \left( \sum (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right)$$

Penalty term:

$$\lambda \sum \beta_j^2$$

This shrinks coefficients toward zero but never exactly zero.

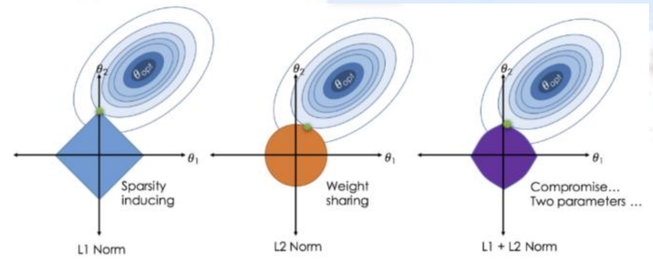
**2.3 Lasso Regression (L1 Regularization):** Lasso Regression (Least Absolute Shrinkage and Selection Operator) is a powerful extension of linear regression that incorporates L1 regularization to enhance model generalization and perform automatic feature selection. By penalizing the absolute size of coefficients, Lasso forces some coefficients to become exactly zero, resulting in sparse models that are easier to interpret and highly effective in high-dimensional datasets where only a subset of features is truly informative.

Useful for **feature selection**, because L1 can shrink coefficients to **exact zero**.

Lasso is widely used in fields such as genomics, finance, medical diagnostics, and machine learning pipelines where interpretability, dimensionality reduction, and model simplicity are essential.

$$\hat{\boldsymbol{\beta}} = \arg\min_{\beta} \left( \sum (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right)$$

**2.4 Elastic Net Regression:** Elastic Net Regression is a powerful regularization technique that combines the strengths of Ridge Regression (L2 penalty) and Lasso Regression (L1 penalty). It addresses key limitations of both methods such as Lasso's instability with highly correlated predictors and Ridge's inability to produce sparse models by integrating both penalties into a unified framework. As a result, Elastic Net performs coefficient shrinkage, feature selection, and stabilization, making it especially effective for high-dimensional and multicollinear datasets. Combines L1 and L2 penalties:



$$\text{Loss} = \sum (y - \hat{y})^2 + \lambda_1 \sum |\beta_j| + \lambda_2 \sum \beta_j^2$$

**3. Model Evaluation Metrics:** Regression model quality is evaluated using metrics such as:

Mean Absolute Error (MAE)

$$MAE = \frac{1}{n}\sum |y_i - \hat{y}_i|$$

Mean Squared Error (MSE)

$$MSE = \frac{1}{n}\sum (y_i - \hat{y}_i)^2$$

Root Mean Squared Error (RMSE)

$$RMSE = \sqrt{MSE}$$

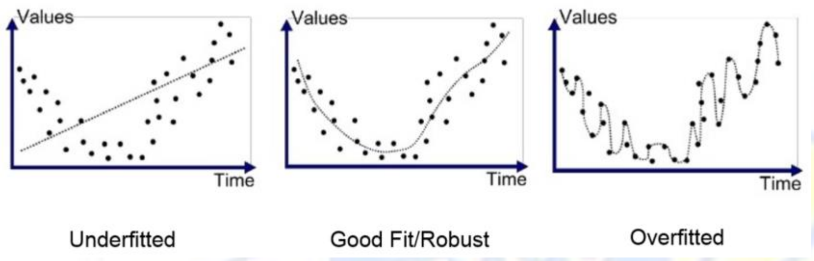
Coefficient of Determination — $R^2$

$$R^2 = 1 - \frac{\sum (y_i - \hat{y})^2}{\sum (y_i - \bar{y})^2}$$

**4. Visual Understanding of Regression Model:** Underfitting occurs when a regression model is too simple to capture the underlying pattern in the data. It typically appears as a nearly straight line that fails to follow the curvature or complexity of the dataset, resulting in high bias and poor performance on both training and test data.

Overfitting arises when a model is excessively complex and adapts too closely to noise or fluctuations in the training data. Visually, the regression curve bends sharply, showing unnecessary oscillations that perfectly fit the training points but generalize poorly to unseen data, leading to high variance.

A well-fit model balances bias and variance, capturing the true trend while avoiding excessive complexity or oversimplification.



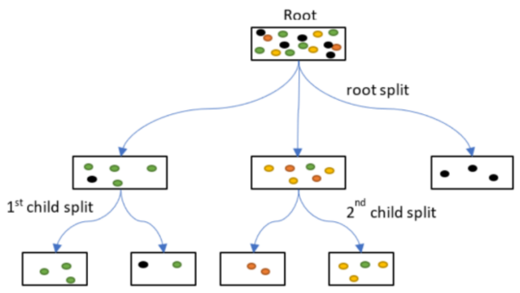
**5. Advantages and Limitations of Regression Model:**
**Advantages**

- Interpretable mathematical structure
- Fast training and prediction
- Works well for linear relationships
- Scikit-Learn implementation is simple and optimized

**Limitations**

- Poor performance on highly nonlinear data
- Sensitive to outliers
- Multicollinearity affects model stability
- Assumes linearity and homoscedasticity

**6. Decision Tree Model Overview (Theoretical + Diagrams)**
A Decision Tree is a supervised machine learning model used for both **classification** and **regression** tasks. It operates by recursively partitioning data into subsets based on feature values, forming a tree-like structure of decision rules. Each interior node represents a test on an attribute, each branch corresponds to an outcome of that test, and each leaf node represents a predicted class or value. Decision trees are valued for their interpretability, ability to model nonlinear relationships, and flexibility with heterogeneous data types.



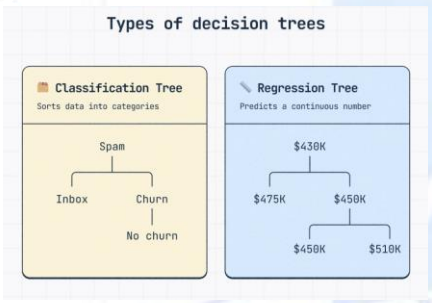
Decision trees are built using recursive binary splitting, guided by impurity measures that quantify how well a feature separates data into homogeneous groups. This greedy, top-down approach is computationally efficient and forms the foundation of modern tree-based algorithms such as CART, Random Forests, and Gradient Boosted Trees.

The training process involves selecting a feature and split point that best partitions the data to reduce uncertainty or impurity. The tree continues to grow until a stopping criterion is reached, such as maximum depth, minimum samples per leaf, or minimal impurity improvement.

**Types of Decision Trees**

**6.1. Classification Trees:** Used when the target variable is categorical. The goal is to maximize class purity at each split.

**6.2. Regression Trees:** Used when predicting continuous outcomes. The goal is to minimize prediction error (variance) within leaf nodes.



**7. Impurity Measures in Decision Trees (Gini, Entropy, and Variance):** Impurity measures quantify the homogeneity or disorder of data subsets within a decision tree. During tree construction, the goal is to select splits that maximize the reduction in impurity, producing child nodes that are more uniform with respect to the target variable. Different impurity metrics are used depending on whether the task is classification or regression.

The three most commonly used impurity measures are Gini Impurity, Entropy, and Variance Reduction. These metrics guide the greedy selection of split points at each internal node of the tree.

**7,1. Gini Impurity (CART Classification):** Gini impurity is a measure of how often a randomly chosen element from a set would be incorrectly labeled if it were assigned a label based on the distribution of labels in the node

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

Where $p_k$ is the probability of class $k$.

Lower Gini → more pure node.

**7.2. Entropy (ID3/C4.5):** Entropy measures the amount of uncertainty or disorder in a dataset. A node with mixed classes has high entropy, while a pure node has low entropy. Used to compute Information Gain, which measures the reduction in entropy after a split.

$$Entropy = -\sum_{k=1}^{K} p_k \log_2(p_k)$$

**7.3. Variance Reduction (Regression Trees):** For regression tasks, the target variable is continuous, so classification impurity measures are not applicable. Instead, regression trees minimize the variance within each node. Splits aim to minimize the weighted variance of child nodes.

$$Var = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2$$

### 7.4. Comparison of Gini, Entropy, and Variance:

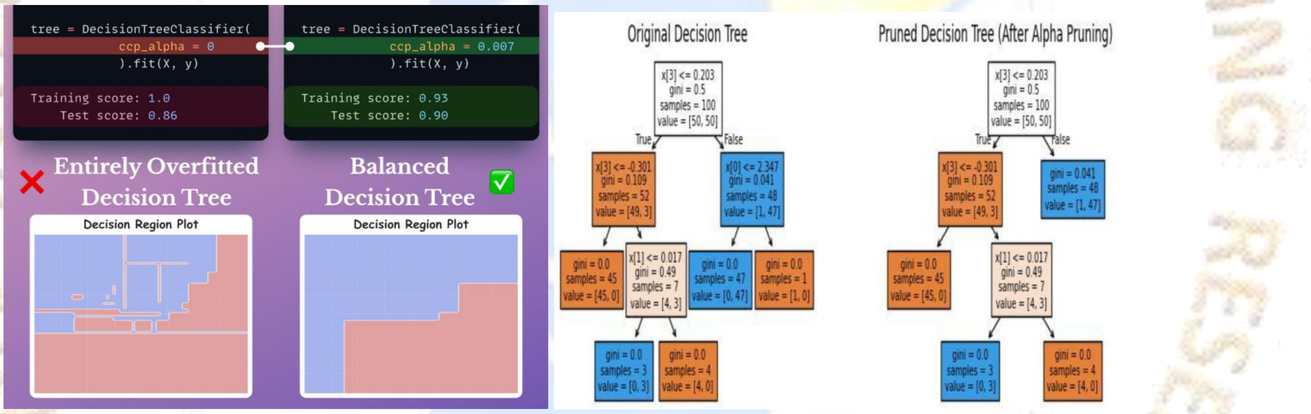
| Impurity Measure | Used For | Sensitive To | Notes |
|---|---|---|---|
| **Gini Impurity** | Classification | Dominant class | Fastest, default in CART/Scikit-Learn |
| **Entropy** | Classification | Rare classes | More theoretically grounded (information theory) |
| **Variance Reduction** | Regression | Spread of numerical values | Minimizes prediction error |

Gini, Entropy, and Variance are core impurity measures that guide how decision trees select the best splits. While Gini and Entropy evaluate classification homogeneity through probability distributions, Variance Reduction minimizes numerical dispersion in regression tasks. Together, these metrics ensure that decision trees produce meaningful, accurate, and interpretable models by reducing uncertainty at each step of recursive partitioning.

## 8. Pruning Methods in Decision Trees: Cost-Complexity Pruning:

Decision trees are highly flexible models capable of capturing complex patterns in data. However, their tendency to grow deep structures with many branches makes them prone to **overfitting**, where the tree memorizes training data instead of generalizing patterns. **Pruning** is a critical technique designed to simplify the tree, reduce variance, and improve predictive performance by removing branches that contribute little to the model's accuracy.

Among pruning approaches, **Cost-Complexity Pruning**, also known as **Minimal Cost-Complexity Pruning (MCCP)**, is the most widely used method implemented in the CART algorithm and adopted in libraries such as Scikit-Learn. MCCP balances model accuracy with structural simplicity by penalizing larger trees.



```
tree = DecisionTreeClassifier(
            ccp_alpha = 0
        ).fit(X, y)

Training score: 1.0
    Test score: 0.86
```

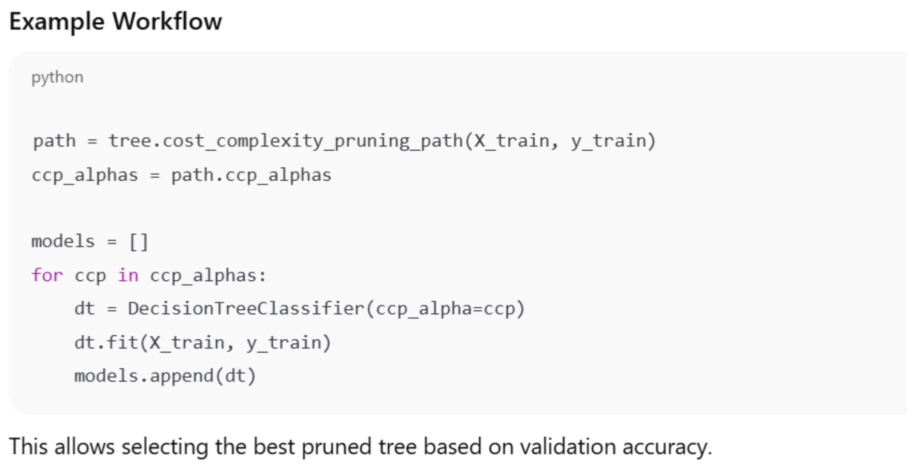
```
tree = DecisionTreeClassifier(
            ccp_alpha = 0.007
        ).fit(X, y)

Training score: 0.93
    Test score: 0.90
```

Entirely Overfitted Decision Tree
Balanced Decision Tree
Decision Region Plot
Decision Region Plot
Original Decision Tree
Pruned Decision Tree (After Alpha Pruning)
x[3] <= 0.203 gini = 0.5 samples = 100 value = [50, 50]
True
False
x[3] <= -0.301 gini = 0.109 samples = 52 value = [49, 3]
x[0] <= 2.347 gini = 0.041 samples = 48 value = [1, 47]
gini = 0.0 samples = 45 value = [45, 0]
x[1] <= 0.017 gini = 0.49 samples = 7 value = [4, 3]
gini = 0.0 samples = 47 value = [0, 47]
gini = 0.0 samples = 1 value = [1, 0]
gini = 0.0 samples = 3 value = [0, 3]
gini = 0.0 samples = 4 value = [4, 0]
gini = 0.041 samples = 48 value = [1, 47]

**Cost-Complexity Pruning in Scikit-Learn:** This pruning removes noisy branches and stabilizes predictions. Pruned trees often outperform fully grown trees on unseen data and provides a smooth, hierarchical pruning path for easy model selection.

Scikit-Learn provides:

- `cost_complexity_pruning_path()` → obtains all effective α values
- `ccp_alpha` parameter → prunes the tree during training

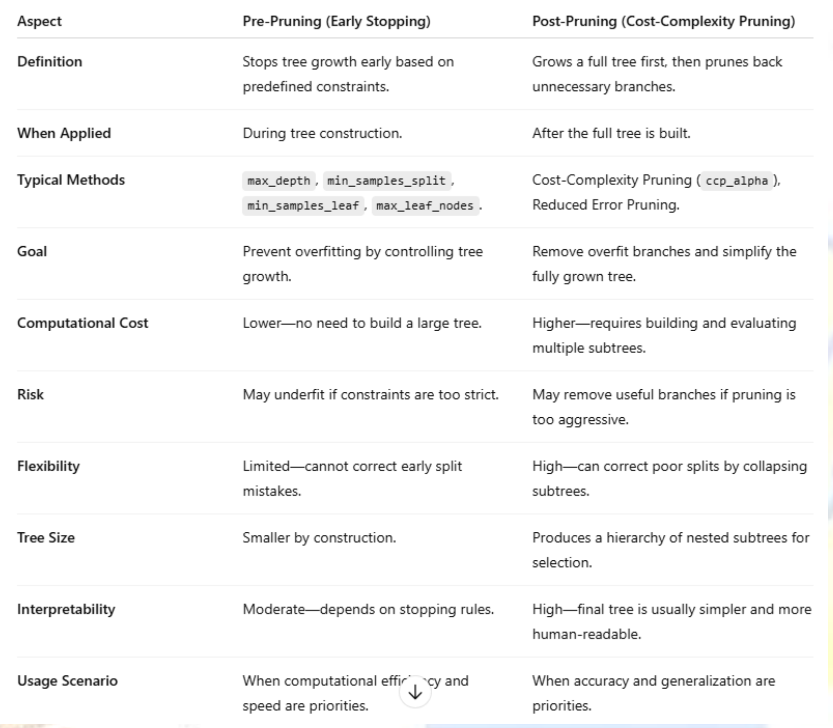


### Example Workflow

```python
path = tree.cost_complexity_pruning_path(X_train, y_train)
ccp_alphas = path.ccp_alphas

models = []
for ccp in ccp_alphas:
    dt = DecisionTreeClassifier(ccp_alpha=ccp)
    dt.fit(X_train, y_train)
    models.append(dt)
```

This allows selecting the best pruned tree based on validation accuracy.

Cost-Complexity Pruning is an essential post-processing technique for decision trees, balancing model accuracy and simplicity by penalizing tree size through a regularization parameter alpha (α). The pruning process generates a hierarchy of nested subtrees, enabling the selection of an optimal model that minimizes overfitting while retaining meaningful structure. As the core pruning approach in CART and Scikit-Learn, MCCP ensures decision trees are interpretable, stable, and generalizable across diverse machine learning applications.



| Aspect | Pre-Pruning (Early Stopping) | Post-Pruning (Cost-Complexity Pruning) |
|---|---|---|
| **Definition** | Stops tree growth early based on predefined constraints. | Grows a full tree first, then prunes back unnecessary branches. |
| **When Applied** | During tree construction. | After the full tree is built. |
| **Typical Methods** | `max_depth`, `min_samples_split`, `min_samples_leaf`, `max_leaf_nodes`. | Cost-Complexity Pruning (`ccp_alpha`), Reduced Error Pruning. |
| **Goal** | Prevent overfitting by controlling tree growth. | Remove overfit branches and simplify the fully grown tree. |
| **Computational Cost** | Lower—no need to build a large tree. | Higher—requires building and evaluating multiple subtrees. |
| **Risk** | May underfit if constraints are too strict. | May remove useful branches if pruning is too aggressive. |
| **Flexibility** | Limited—cannot correct early split mistakes. | High—can correct poor splits by collapsing subtrees. |
| **Tree Size** | Smaller by construction. | Produces a hierarchy of nested subtrees for selection. |
| **Interpretability** | Moderate—depends on stopping rules. | High—final tree is usually simpler and more human-readable. |
| **Usage Scenario** | When computational efficiency and speed are priorities. | When accuracy and generalization are priorities. |

**9. Model Comparison: Decision Trees vs Linear Models:**

Decision Trees and Linear Models represent two fundamentally different approaches to supervised learning. Linear models assume a mathematically defined, often continuous relationship between predictors and the target variable, whereas decision trees learn decision boundaries through hierarchical, rule-based partitioning. Understanding their differences is critical for selecting the appropriate model depending on data characteristics, interpretability needs, and predictive objectives. Linear models excel at capturing **global, linear relationships** and are computationally efficient, while decision trees capture **nonlinear interactions**, hidden structures, and feature interactions without requiring explicit transformations. Each family of models brings unique strengths and trade-offs in terms of accuracy, interpretability, robustness, and generalization.

| Aspect | Linear Models | Decision Trees |
|---|---|---|
| **Assumptions** | Strong assumptions: linearity, independence, normality, homoscedasticity | Non-parametric; no assumptions about data distribution |
| **Relationship Modeled** | Global linear (or polynomial if transformed) | Local, piecewise constant relationships |
| **Handling Nonlinearity** | Requires transformations (polynomial terms, interactions) | Naturally captures nonlinear and interaction effects |
| **Interpretability** | High for linear; moderate for regularized models; coefficients are directly interpretable | High for shallow trees; decreases as tree depth grows |
| **Feature Selection** | Lasso/Elastic Net perform automatic feature selection | Tree splits inherently select features |
| **Handling of Outliers** | Sensitive to outliers, which distort coefficients | More robust; splits reduce outlier influence |
| **Data Preprocessing** | Requires scaling, handling multicollinearity | Minimal preprocessing needed |
| **Overfitting Risk** | Moderate; controlled with regularization | High for unpruned trees; reduced by pruning |
| **Generalization** | Strong, especially with regularization | Can be weaker unless pruned or used in ensembles |
| **Ability to Model Interactions** | Must explicitly create interaction terms | Automatically learns interactions through splits |
| **Stability** | Stable coefficients; small changes in data → small changes in model | Unstable; small data changes may yield different trees |
| **Computational Efficiency** | Fast training and prediction | Efficient, but slower than linear models on large feature spaces |
| **Best Use Cases** | Trend forecasting, risk modeling, econometrics, regression tasks with linear patterns | Classification, segmentation, heterogeneous data, nonlinear relationships |
| **Implementation in Scikit-Learn** | Multiple algorithms (OLS, Ridge, Lasso, ElasticNet) with simple API | DecisionTreeClassifier and DecisionTreeRegressor with impurity-based splitting |

## 10. Conclusion:

This paper presented a comprehensive exploration of foundational machine learning models, focusing on regression techniques, decision-tree methodologies, and their practical implementation through the Scikit-Learn framework. Regression models including linear, polynomial, Ridge, Lasso, and Elastic Net offer robust mathematical structures for modeling continuous outcomes and understanding variable relationships. Their interpretability, computational efficiency, and statistical rigor make them essential tools across scientific and industrial domains. However, their reliance on linear assumptions often limits their flexibility in capturing complex data patterns.

Decision trees, in contrast, provide an intuitive, non-parametric framework capable of modeling nonlinear relationships, variable interactions, and heterogeneous data distributions without extensive preprocessing. Through impurity-driven recursive partitioning and pruning techniques such as cost-complexity pruning, decision-tree models balance expressiveness with generalization, offering transparent decision rules that facilitate interpretability and domain-driven analysis. The comparative analysis highlights that while linear models excel in structured, linearly separable settings, decision trees are better suited for datasets characterized by nonlinear dynamics and categorical attributes.

Scikit-Learn's unified API significantly democratizes the use of these models by providing optimized implementations, preprocessing utilities, hyperparameter tuning tools, and reproducible workflows. Its modular architecture enables seamless experimentation, making it one of the most widely used machine learning libraries in both academia and industry.

Overall, this study emphasizes that no single model universally outperforms others; instead, model selection must be driven by data characteristics, problem requirements, and interpretability needs. Regression models and decision-tree algorithms each offer distinct advantages, and practitioners often benefit from using them in complementary ways such as employing linear models for baseline benchmarking and decision trees for capturing nonlinear structures or serving as base learners in ensemble methods. As machine learning continues to evolve, integrating classical models with modern techniques, automated pipelines, and explainability frameworks will remain essential for developing accurate, transparent, and reliable AI systems.

**References:**

[1] V. Vapnik, *The Nature of Statistical Learning Theory*. New York, NY, USA: Springer-Verlag, 1995.

[2] T. M. Mitchell, *Machine Learning*. New York, NY, USA: McGraw-Hill, 1997.

[3] C. F. Gauss, *Theoria Combinationis Observationum Erroribus Minimis Obnoxiae*. Göttingen: Dieterich, 1821.

[4] A. M. Legendre, *Nouvelles méthodes pour la détermination des orbites des comètes*. Paris: F. Didot, 1805.

[5] A. E. Hoerl and R. W. Kennard, "Ridge Regression: Biased Estimation for Nonorthogonal Problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[6] R. Tibshirani, "Regression Shrinkage and Selection via the Lasso," *Journal of the Royal Statistical Society: Series B*, vol. 58, no. 1, pp. 267–288, 1996.

[7] H. Zou and T. Hastie, "Regularization and Variable Selection via the Elastic Net," *Journal of the Royal Statistical Society: Series B*, vol. 67, no. 2, pp. 301–320, 2005.

[8] E. B. Hunt, J. Marin, and P. J. Stone, *Experiments in Induction*. New York, NY, USA: Academic Press, 1966.

[9] J. R. Quinlan, "Induction of Decision Trees," *Machine Learning*, vol. 1, pp. 81–106, 1986.

[10] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA, USA: Morgan Kaufmann, 1993.

[11] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*. Belmont, CA, USA: Wadsworth International, 1984.

[12] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, pp. 5–32, 2001.

[13] J. H. Friedman, "Greedy Function Approximation: A Gradient Boosting Machine," *Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.

[14] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[15] C. Molnar, *Interpretable Machine Learning*. 2nd ed. Leanpub, 2019.

[16] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York, NY, USA: Springer, 2009.

[17] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical Machine Learning Tools and Techniques*, 4th ed. Burlington, MA, USA: Morgan Kaufmann, 2016.

[18] R. J. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice*. OTexts, 2018.