**Tree-Based Models in Machine Learning**

[[Nathan Rosidi](https://medium.com/@nathanrosidi?source=post_page-----55295681f13c--------------------------------)](https://medium.com/@nathanrosidi?source=post_page-----55295681f13c--------------------------------)

[[MLearning.ai](https://medium.com/mlearning-ai?source=post_page-----55295681f13c--------------------------------)](https://medium.com/mlearning-ai?source=post_page-----55295681f13c--------------------------------)

[Nathan Rosidi](https://medium.com/@nathanrosidi?source=post_page-----55295681f13c--------------------------------)

·

[Follow](https://medium.com/m/signin?actionUrl=https%3A%2F%2Fmedium.com%2F_%2Fsubscribe%2Fuser%2Fab636cbf3611&operation=register&redirect=https%3A%2F%2Fmedium.com%2Fmlearning-ai%2Ftree-based-models-in-machine-learning-55295681f13c&user=Nathan+Rosidi&userId=ab636cbf3611&source=post_page-ab636cbf3611----55295681f13c---------------------post_header-----------)

Published in

[MLearning.ai](https://medium.com/mlearning-ai?source=post_page-----55295681f13c--------------------------------)

·

9 min read

·

Dec 1

69

1

*Mastering Tree-Based Models in Machine Learning: A Practical Guide to Decision Trees, Random Forests, and GBMs*

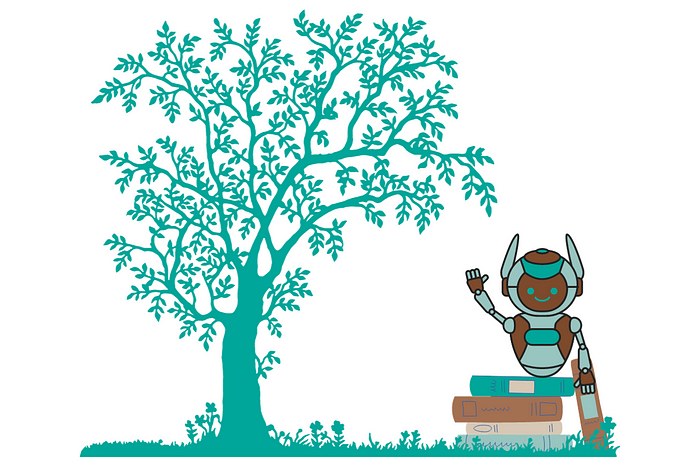


Image created by the author on [Canva](https://www.canva.com/)

Ever wondered how machines make complex decisions? Just like a tree branches out, tree-based models in machine learning do something similar. They’re the backbone of decision-making processes in AI.

In the next few sections, we’ll explore different types of tree-based models. We start with the basics: decision trees. Then we branch out to random forests and gradient boosting machines. Each has its unique way of processing data and making decisions.

But how do these models apply to real-world scenarios? From financial analysis to healthcare predictions, tree-based models have a significant impact. At the end, we also will see the roadblocks and their solutions. Let’s get started!

**Types of Tree-Based Models**

In this section, we explore tree-based models in machine learning. We start with decision trees, understanding their basics and functionality. Next, we discuss random forests and their benefits over single decision trees.

Finally, we examine gradient boosting machines (GBMs), focusing on their unique aspects and comparison with random forests. This overview will introduce you to the key tree-based models.

So buckle up!

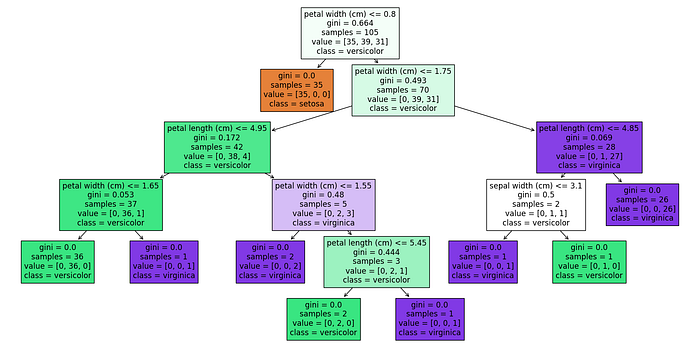
**Decision Trees**

Decision trees in machine learning are like flowcharts, making decisions based on data. They’re particularly useful in scenarios requiring clear, logical decisions.

For our Python example, we’ll use the famous Iris dataset from scikit-learn, which includes measurements of iris flowers and their species. We’ll predict the species based on these measurements. We’ll also visualize the tree to understand how decisions are made.

from sklearn.datasets import load\_iris  
from sklearn.tree import DecisionTreeClassifier, plot\_tree  
from sklearn.model\_selection import train\_test\_split  
import matplotlib.pyplot as plt  
from sklearn.datasets import load\_iris  
from sklearn.model\_selection import train\_test\_split  
from sklearn.tree import DecisionTreeClassifier, plot\_tree  
import matplotlib.pyplot as plt  
  
# Load the Iris dataset  
iris = load\_iris()  
X = iris.data  
y = iris.target  
  
# Split the data  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3)  
  
# Create and train the decision tree  
model = DecisionTreeClassifier()  
model.fit(X\_train, y\_train)  
  
# Visualize the decision tree  
plt.figure(figsize=(20,10))  
plot\_tree(model, filled=True, feature\_names=iris.feature\_names, class\_names=list(iris.target\_names))  
plt.show()

Let’s see the output.



You see the decision tree that we draw.

* **Splitting Criteria**: The tree splits based on petal length and width, indicating these are key features for classification.
* **Purity of Nodes**: Nodes with a Gini index of 0 are perfectly pure, meaning all samples at that node belong to one class.
* **Class Distribution**: Each leaf node shows how many samples are classified into each class. For instance, the leftmost leaf has 36 samples all classified as versicolor.

**Random Forests**

Random forests are an ensemble learning method, where multiple decision trees come together to make more accurate predictions.

Think of it as a team of experts where each member provides their opinion, and the final decision is made based on the majority vote. This method is often more accurate than a single decision tree.

Here’s how we can implement a random forest in Python using the scikit-learn library:

from sklearn.datasets import load\_iris  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import accuracy\_score  
  
# Load Iris dataset  
iris = load\_iris()  
X = iris.data  
y = iris.target  
  
# Splitting dataset  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3)  
  
# Creating and training the Random Forest model  
random\_forest\_model = RandomForestClassifier(n\_estimators=100)  
random\_forest\_model.fit(X\_train, y\_train)  
  
# Making predictions  
predictions = random\_forest\_model.predict(X\_test)  
  
# Evaluating the model  
print("Accuracy:", accuracy\_score(y\_test, predictions))

In this code, we use the same Iris dataset. However, instead of a single decision tree, we use a random forest with 100 trees (n\_estimators=100).

After training, we predict the species of the iris flowers in the test set. The accuracy of these predictions typically surpasses that of a single decision tree, showcasing the strength of random forests in handling complex data sets in data science.

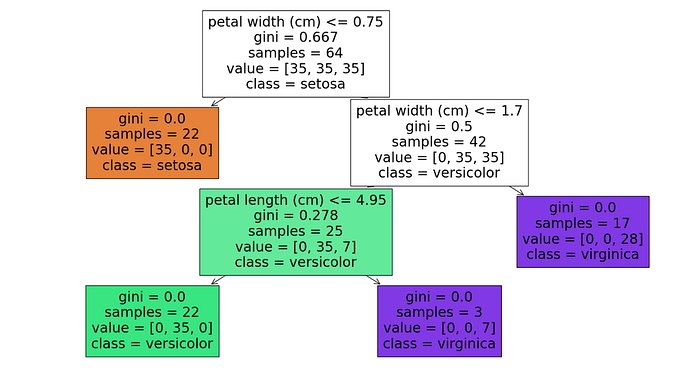
Here is the output.

Random forests in Tree Based Models

Now let’s visualize this.

# Visualize one of the trees in the forest  
plt.figure(figsize=(20,10))  
tree\_index = 0 # Choose the index of the tree you want to visualize  
plot\_tree(random\_forest\_model.estimators\_[tree\_index], filled=True, feature\_names=iris.feature\_names, class\_names=list(iris.target\_names))  
plt.show()

Here is the output.



This is a single decision tree from a Random Forest model, specifically the first tree given that tree\_index is set to 0., that successfully separates iris species with nodes showing a Gini index of 0, indicating pure classifications.

It starts by classifying based on petal width and further refines with petal length, demonstrating the model’s ability to distinguish species, especially the setosa and virginica.

**Gradient Boosting Machines (GBMs)**

Gradient Boosting Machines (GBMs) are another type of ensemble learning technique, but they work differently from random forests. While random forests build trees independently, GBMs build them sequentially.

Each new tree helps correct errors made by the previous ones. It’s like each tree in the sequence is learning from the mistakes of its predecessors, making the overall model more accurate.

Let’s implement a GBM using Python’s scikit-learn library:

from sklearn.datasets import load\_iris  
from sklearn.ensemble import GradientBoostingClassifier  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import accuracy\_score  
  
# Load Iris dataset  
iris = load\_iris()  
X = iris.data  
y = iris.target  
  
# Splitting the dataset  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3)  
  
# Creating and training the GBM model  
gbm\_model = GradientBoostingClassifier(n\_estimators=100)  
gbm\_model.fit(X\_train, y\_train)  
  
# Making predictions and evaluating the model  
predictions = gbm\_model.predict(X\_test)  
print("Accuracy:", accuracy\_score(y\_test, predictions))

Here is the output.

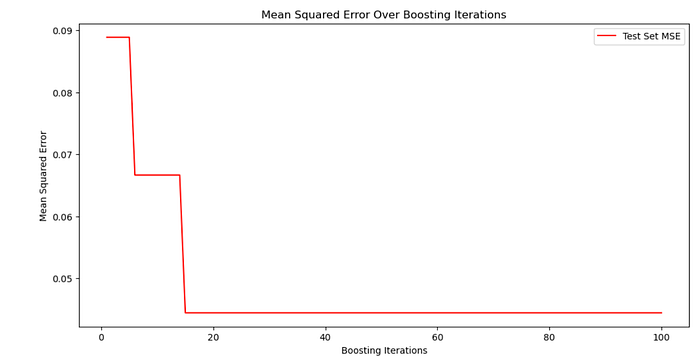
Gradient Boosting Machines in Tree Based Models

In this example, we use the GradientBoostingClassifier to predict the species of iris flowers. The model builds 100 trees (n\_estimators=100), each learning from the errors of the previous one. This improvement often results in high accuracy, making GBMs a powerful tool in data science for solving complex problems.

Let’s visualize this.

# Plotting the Mean Squared Error  
test\_score = np.zeros((100,), dtype=np.float64)  
for i, y\_pred in enumerate(gbm\_model.staged\_predict(X\_test)):  
 test\_score[i] = mean\_squared\_error(y\_test, y\_pred)  
  
plt.figure(figsize=(12, 6))  
plt.title('Mean Squared Error Over Boosting Iterations')  
plt.plot(np.arange(100) + 1, test\_score, 'r-', label='Test Set MSE')  
plt.legend(loc='upper right')  
plt.xlabel('Boosting Iterations')  
plt.ylabel('Mean Squared Error')  
plt.show()

Here is the output.



This graph above shows the mean squared error (MSE) of a Gradient Boosting model’s predictions on the test set over 100 boosting iterations. The MSE drops sharply in the initial iterations, indicating significant improvements in the model’s performance.

After around 20 iterations, the MSE stabilizes, suggesting that additional iterations do not contribute much to further reducing the error. This could mean that the model has reached its optimal performance early, and additional iterations are not adding value, which is a good sign of model convergence.

**Decision Trees & Random Forests & Gradient Boosting Machines**

Now, let’s discuss the differences between decision trees, random forests, and GBMs:

* **Decision Trees**: In our example, we used a single decision tree to classify iris species. Decision trees are straightforward and easy to interpret but can be prone to overfitting. They work well for simple tasks but might not perform as well in more complex scenarios.
* **Random Forests**: The random forest model we implemented uses **multiple decision trees** to make more accurate predictions. Unlike a single decision tree, random forests reduce the risk of overfitting by averaging the results of many trees. This makes them more robust and accurate, especially in more complex datasets.
* **Gradient Boosting Machines (GBMs)**: GBMs, as shown in our example, build trees one at a time, where each new tree helps to correct the errors made by previous trees. This sequential building process can lead to a highly accurate model. However, GBMs might be more sensitive to overfitting if the data is noisy and can be more challenging to tune than random forests.

Each of these methods has its strengths and weaknesses, and the choice between them depends on the specific requirements of your data and problem.

**Roadblocks in Applying Decision Trees**

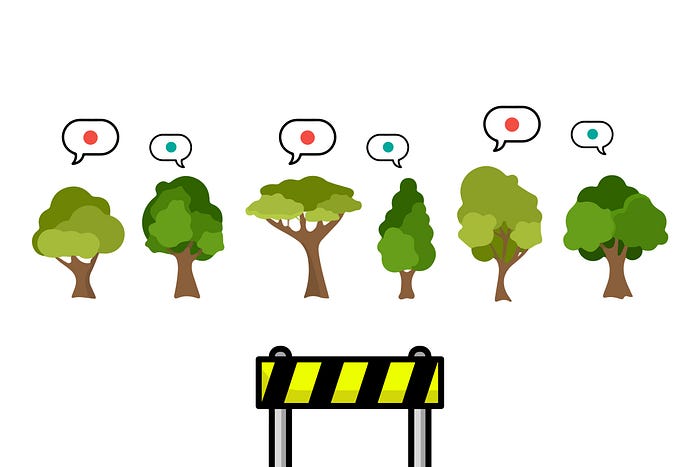


Image created by the author on [Canva](https://www.canva.com/)

When applying decision trees in machine learning, several challenges or roadblocks can arise. Understanding these challenges is key to effectively using decision trees in data science projects.

1. **Overfitting:** One of the most common issues with decision trees is overfitting, where the model learns the training data too well, including its noise and outliers. This causes to poor performance on unseen data. Overfitting happens especially when the tree is too deep, capturing too much detail and complexity.

* **Solution**: Implement pruning techniques to limit the depth of the tree, and use cross-validation to ensure the model generalizes well to unseen data.

**2. Handling Continuous Variables:** Decision trees can struggle with continuous variables. They split these variables at various points, but finding the optimal split point can be tricky, especially when the continuous variable doesn’t have a clear, discrete boundary.

* **Solution**: Use discretization methods to convert continuous variables into categorical ones, simplifying the decision-making process.

**3. Biased Trees with Imbalanced Data:** If the training data is imbalanced (i.e., some classes are underrepresented), the decision tree might become biased towards the dominant class. This bias can skew the predictions, favoring the majority class at the expense of the minority class.

* **Solution**: Apply techniques like SMOTE (Synthetic Minority Over-sampling Technique) or adjust class weights to balance the dataset before training.

**4. Complexity with Large Datasets:** While decision trees handle small to medium-sized datasets efficiently, their performance can decrease with very large datasets. The time to train the tree increases significantly, and the tree structure can become overly complex, making it harder to interpret.

* **Solution**: Use dimensionality reduction techniques or feature selection to reduce the dataset size and complexity, making the tree more manageable.

**5. Limited to Linear Decision Boundaries:** Decision trees primarily create linear decision boundaries, which can be limiting in scenarios where the relationship between features and the target variable is more complex or non-linear.

* **Solution**: Combine decision trees with other algorithms, like SVMs (Support Vector Machines), that can capture non-linear relationships.

**6. Sensitivity to Small Changes in Data:** Decision trees can be quite sensitive to small changes in the training data. A slight change can lead to a significantly different tree structure. This lack of stability can be a concern in dynamic environments where data changes frequently.

* **Solution**: Use ensemble methods like Random Forest or Gradient Boosting which are less sensitive to small data variations and offer more stability.

If you want to know more about machine learning algorithms, here is a good source for you → [machine learning algorithms you must know](https://www.stratascratch.com/blog/machine-learning-algorithms-you-should-know-for-data-science/?utm_source=blog&utm_medium=click&utm_campaign=ML.ai+tree+based+models).

**Final Thoughts**

We went through the world of tree-based models in machine learning, uncovering the nuances of decision trees, random forests, and GBMs. Using the Iris dataset, we saw these models in action, from simple visualizations to tackling complex data challenges. We also navigated through common roadblocks, offering practical solutions to enhance model performance.

Practice is key in data science. Experiment with these models, tweak them, and see how they perform with different data. Each challenge you encounter and overcome sharpens your skills further.

Join our platform to take this learning further. Engage with real-world data projects and prepare for your career in data science. Visit us, apply your knowledge, and continue your growth as a budding data scientist. Your journey is just starting, and there’s much more to explore and achieve!

*Originally published at*[*https://www.stratascratch.com*](https://www.stratascratch.com/blog/tree-based-models-in-machine-learning/?utm_source=blog&utm_medium=click&utm_campaign=ML.ai+tree+based+models)*.*

[**WRITER at MLearning.ai**](https://mlearning.substack.com/about)**/**[**48K+ GPTs**](https://open.substack.com/pub/mlearning/p/top-best-custom-gpts-top-ai-tools-chatgpt-openai?r=z7zu8&utm_campaign=post&utm_medium=web)**/**[**Free TURBO AI Art Tools**](https://open.substack.com/pub/mlearning/p/free-turbo-ai-art-tools-no-limits?r=z7zu8&utm_campaign=post&utm_medium=web)