PDS Challenge 4:
Trees and Forests

SR+ITL

November 24, 2023

Abstract

In this challenge, we explore the fundamental concepts of decision trees and random forests in the context of predictive modeling using the Space Titanic dataset. Decision trees, with their intuitive hierarchical structure, serve as building blocks for more advanced ensemble methods like random forests. We investigate how these algorithms can effectively handle classification tasks and reveal insights into the factors influencing space travelers' survival on the Titanic. By delving into the intricacies of decision trees and the collective power of random forests, participants will gain hands-on experience in feature selection, model interpretation, and ensemble learning. The Space Titanic dataset serves as a compelling backdrop, allowing you to apply these techniques to real-world scenarios.

1 Introduction to Decision Trees

Decision trees are versatile and interpretable machine learning models widely used for classification and regression tasks. They operate by recursively partitioning the input space based on feature conditions, creating a tree-like structure where each internal node represents a decision based on a specific feature, and each leaf node corresponds to the predicted outcome.

1.1 Decision Tree Structure

The structure of a decision tree can be represented by a set of rules in the form of "if-then" conditions. Given a dataset with features $\mathbf{X}$ and labels $\mathbf{y}$, a decision tree recursively splits the dataset into subsets based on feature conditions to minimize impurity or maximize information gain. The decision at each node is determined by optimizing a splitting criterion, often measured by Gini impurity or entropy.

Let D be a dataset, and X_i be a feature. The decision tree recursively partitions D into subsets D_j based on a binary condition:

$$D_j = \{(\mathbf{x}, y) \in D : x_i \leq \text{threshold}\} \quad \text{or} \quad D_j = \{(\mathbf{x}, y) \in D : x_i > \text{threshold}\} \tag{1}$$

where threshold is determined to optimize the splitting criterion.

1.2 Decision Tree Algorithm

The construction of a decision tree involves selecting the best features and thresholds at each node to maximize the homogeneity of the resulting subsets. The algorithm continues recursively until a stopping criterion is met, such as reaching a maximum depth or a minimum number of samples in a leaf node.

Mathematically, the decision tree algorithm aims to find the optimal splitting parameters by minimizing a cost function, which is often defined as:

$$J(D, X_i, \text{threshold}) = \frac{|D_L|}{|D|} \cdot \text{impurity}(D_L) + \frac{|D_R|}{|D|} \cdot \text{impurity}(D_R) \tag{2}$$

where D_L and D_R are the subsets resulting from the split.

In the upcoming sections, we will delve into the practical application of decision trees, exploring their strengths, limitations, and extensions to ensemble methods such as random forests.

2 Random Forests: Ensemble of Decision Trees

While decision trees offer interpretable models, they are susceptible to overfitting and can be sensitive to variations in the training data. Random Forests, an ensemble learning method, address these issues by aggregating the predictions of multiple decision trees.

2.1 Ensemble Learning

The key idea behind ensemble learning is to combine the predictions of multiple models to create a more robust and accurate predictor. In the context of Random Forests, this involves training a collection of decision trees on different subsets of the training data and averaging their predictions (for regression tasks) or holding a vote (for classification tasks).

2.2 Bootstrap Aggregating (Bagging)

Random Forests employ a technique known as Bootstrap Aggregating, or Bagging, to train each decision tree on a random subset of the training data. Given a dataset D with N samples, Bagging involves creating multiple bootstrap samples D_i, each of size N, by randomly sampling with replacement from D. Each decision tree in the Random Forest is then trained on one of these bootstrap samples.

2.3 Random Feature Selection

To further decorrelate the individual trees and promote diversity, Random Forests introduce randomness in feature selection at each split. Instead of considering all features when making a split decision, only a random subset of features is considered. This ensures that each tree is trained on a different subset of features, reducing the risk of overfitting to specific features in the dataset.

Mathematically, at each node of a decision tree, Random Forests select a random subset of features m out of the total M features. The best split is then determined based on this subset.

2.4 Bootstrapping in Random Forests

Let D_i be a bootstrap sample, and X_i be a random subset of features. The predicted output of a Random Forest, denoted as $\hat{y}_{\text{RF}}(\mathbf{x})$, is the average (for regression) or the majority vote (for classification) of individual tree predictions:

$$\hat{y}_{\text{RF}}(\mathbf{x}) = \frac{1}{T} \sum_{t=1}^{T} \hat{y}_t(\mathbf{x}) \tag{3}$$

where T is the total number of trees in the Random Forest, and $\hat{y}_t(\mathbf{x})$ is the prediction of the t-th tree.

Random Forests, by aggregating the predictions of diverse and randomly trained decision trees, enhance generalization performance and improve robustness, making them a powerful tool for various machine learning tasks.

3 Spice: Interaction Terms and Random Forests

In traditional linear models, the inclusion of interaction terms is often crucial to capture non-linear relationships and address potential violations of independence assumptions between the response variable Y and predictors $X_1, \ldots, X_n$. Interaction terms, denoted as $X_i \cdot X_j$ or higher-order combinations, enable the model to account for complex relationships that may not be captured by a simple linear relationship.

However, the paradigm shifts when we consider Random Forests. The strength of Random Forests lies in their ability to automatically handle complex interactions and non-linear relationships without explicitly requiring the inclusion of interaction terms.

3.1 Mathematical Formulation

Consider a dataset D with predictors $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$ and the response variable Y. Let T be the total number of trees in the Random Forest. The prediction $\hat{y}_{\text{RF}}(\mathbf{x})$ of the Random Forest for an input vector $\mathbf{x}$ is given by:

$$\hat{y}_{\text{RF}}(\mathbf{x}) = \frac{1}{T} \sum_{t=1}^{T} \hat{y}_t(\mathbf{x}) \tag{4}$$

where $\hat{y}_t(\mathbf{x})$ is the prediction of the t-th tree. The inherent randomness in the tree-building process introduces diversity, capturing different facets of the interaction space.

For a single decision tree, the prediction $\hat{y}_t(\mathbf{x})$ is constructed based on the rules of each node j:

$$\hat{y}_t(\mathbf{x}) = \sum_{j=1}^{J} c_j I(\mathbf{x} \in R_j) \tag{5}$$

where c_j is the predicted value at node j, R_j is the region defined by the conditions along the path to node j, and $I(\mathbf{x} \in R_j)$ is an indicator function.

3.2 Random Forests' Flexibility

Random Forests are inherently non-parametric and robust, capable of capturing intricate relationships between the response variable and predictors. The ensemble nature of Random Forests, comprising multiple decision trees trained on different subsets of data, naturally allows them to learn and adapt to complex patterns, including non-linearities and interactions.

3.3 Implicit Interaction Capture

The randomness introduced through bootstrap sampling and random feature selection during tree construction inherently incorporates interaction effects. By training each tree on a different subset of data and features, Random Forests implicitly capture a diverse range of interactions. The averaging or voting process during prediction across the ensemble further combines these interactions to provide a comprehensive model.

3.4 Parameter Tuning

While Random Forests can inherently capture interactions, parameter tuning is still important. Adjusting parameters such as the maximum depth of trees ($d_{\max}$) and the number of features considered at each split (m) allows for fine-tuning the model's performance. The algorithm considers a random subset of m features at each split, promoting diversity across trees.

4 Project Overview

For this challenge, your tasks will involve the following key objectives:

1. Make use of Decision Trees and Random Forests to make predictions.
2. Compare the performance of Decision Trees and Random Forests.
3. Participate in the Kaggle competition, where you will have the opportunity to apply your insights and techniques developed during this challenge to solve real-world problems. Link: Spaceship Titanic

4.1 Deliverables

As with the past challenges, you are to deliver on E3 platform:

1. Code: as either notebook or python file. Make sure to add meaningful comments.
2. Report: a PDF file detailing what you did, and why. Delve into the mathematics and the reasoning behind your model.
3. Video: I suggest adding a link to your PDF report to your video. You can upload your video to the platform of your preference.

Please name your files as follows: *yourgroupid_report-or-code.file-extension*
For example, if your group ID is "group10," and you wrote your code on a notebook, then your submitted code should be named *group10_code.ipynb*.