Machine Learning for Data Science, 2025

Implementation and Evaluation of Classification Trees and Random Forests for Binary Classification

Teodora Taleska

Abstract

This report presents the implementation and evaluation of classification trees and random forests for binary classification tasks, specifically applied to the TKI resistance FTIR spectral data set. The methods utilized include building classification trees using Gini impurity for splitting and random forests with bootstrap sampling and random variable selection. The performance of the models was assessed using misclassification rates and standard errors, and variable importance was analyzed using permutation-based methods. The feature importance approach was extended by assessing combinations of three variables and tree structure-based importance. The findings offer insights into the effectiveness of these models and their performance in predictive tasks.

Introduction

The objective of this project is to implement classification trees and random forests from scratch and evaluate their performance on the TKI resistance FTIR spectral dataset. The dataset consists of spectral measurements across multiple wavelengths for different cell lines, with a binary classification target.

The project is divided into three parts:

1. Building and evaluating classification trees and random forests – Implementing the models, computing misclassification rates, and quantifying uncertainty.
2. Variable importance analysis – Implementing permutation-based feature importance to identify the most relevant spectral variables.
3. Extending feature importance analysis – Assessing variable interactions and analyzing tree structures to derive the most significant combinations of three variables.

This report details the implementation methodology, experimental results, and conclusions drawn from evaluating classification trees and random forests on the dataset.

Part 1: Misclassification Rates and Uncertainty

Methodology

Classification Trees

In this project, **classification trees** were implemented, where the splits were made to minimize the **Gini impurity**. The Gini impurity measures the likelihood of an incorrect classification of a randomly chosen element if it was randomly labeled according to the class distribution in the subset[1]. The formula for Gini impurity is:

$$\text{Gini}(D) = 1 - \sum_{c=1}^{C} \hat{\pi}_c^2$$

where:

- D is the dataset at a node,
- C is the number of classes,
- $\hat{\pi}_c$ is the proportion of data points in D that belong to class c.

The goal of the tree is to recursively partition the data into subsets that are as pure as possible (i.e., containing mostly one class), thereby reducing the Gini impurity at each split.

Random Forests

Random forests are an ensemble learning method designed to reduce the variance of predictions by averaging the outputs of multiple decision trees. Each tree is trained on a different subset of the data, chosen randomly with replacement (a technique known as **bagging**)[1]. The final prediction is computed as:

$$f(x) = \frac{1}{M} \sum_{m=1}^{M} f_m(x),$$

where $f_m(x)$ is the prediction of the m-th tree, and M is the total number of trees in the forest.

To further reduce correlation between trees, random forests introduce randomness in two ways:

- **Random subsets of data**: Each tree is trained on a bootstrap sample of the dataset.
- **Random subsets of features**: At each split, only a random subset of features is considered.

This decorrelation improves the model's predictive accuracy and robustness.

Quantifying Uncertainty

To quantify the uncertainty of the estimates, we compute both the **misclassification rate (MR)** and the **standard error**. The misclassification rate measures the proportion of incorrectly classified samples[1], while the standard error provides an estimate of the uncertainty associated with this rate.

The misclassification rate is calculated as:

$$\text{MR} = 1 - \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(y_i = \hat{y}_i),$$

where:

- y_i is the true label for the i-th sample,
- $\hat{y}_i$ is the predicted label for the i-th sample,
- N is the total number of samples,
- $\mathbb{I}(\cdot)$ is the indicator function, which equals 1 if the condition is true and 0 otherwise.

The standard error of the misclassification rate is calculated as:

$$\text{Standard Error} = \frac{\sigma}{\sqrt{N}},$$

where:

- σ is the standard deviation of the misclassification errors ($y_i \neq \hat{y}_i$),
- N is the total number of samples.

The misclassification rate provides a point estimate of the model's error, while the standard error measures the uncertainty of this estimate. Together, they allow us to assess not only how well the model performs but also how reliable the performance estimate is. For example, a low misclassification rate with a small standard error indicates both high accuracy and high confidence in the estimate. Conversely, a low misclassification rate with a large standard error suggests that the estimate may not be reliable due to high variability.

Results

Decision Tree Misclassification Rates

The decision tree model achieved a **training misclassification rate of** 0.0 ± 0.0, indicating perfect performance on the training data. However, on the test set, the misclassification rate increased to $\mathbf{0.367 \pm 0.062}$. This suggests that the decision tree is overfitting the training data, as it performs well on the training set but poorly on the test set. The relatively high standard error on the test set indicates some variability in the model's performance.

Random Forest Misclassification Rates

The random forest model, built with **100 trees**, achieved a **training misclassification rate of** 0.0 ± 0.0, similar to the decision tree. However, on the test set, the random forest performed significantly better, with a misclassification rate of $\mathbf{0.217 \pm 0.053}$. This demonstrates that the random forest generalizes better to unseen data compared to the decision tree, reducing overfitting and improving test performance. The lower standard error also indicates more consistent performance across different test samples.

Misclassification vs. Number of Trees

The plot below shows how the misclassification rate on the test set changes as the number of trees in the random forest increases. The shaded region represents the **standard error** of the misclassification rate, providing a measure of uncertainty. From the plot, we observe the following:

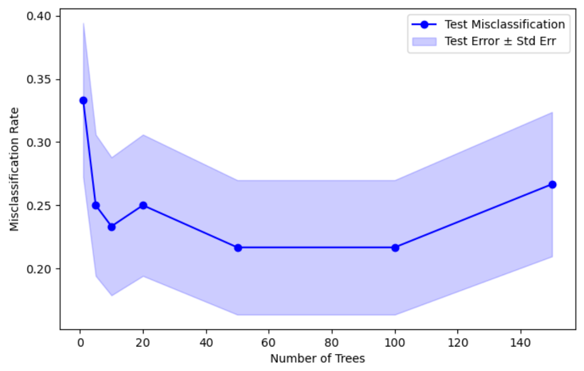

Figure 1. Misclassification Rate vs. Number of Trees in Random Forest

- As the number of trees increases, the misclassification rate on the test set initially decreases, reaching a minimum at around **50 trees**.
- Beyond 50 trees, the misclassification rate stabilizes, with no significant improvement in performance.
- The standard error remains relatively constant, indicating that the model's performance is consistent across different numbers of trees once the forest is sufficiently large.

This behavior aligns with the theory of random forests: increasing the number of trees reduces variance and improves generalization, but beyond a certain point, the gains diminish. The results suggest that **50 trees** are sufficient for this dataset, as adding more trees does not significantly improve performance.

Part 2: Variable Importance Analysis

Methodology

Permutation-Based Feature Importance

Variable importance in random forests can be computed using out-of-bag (OOB) samples. For each tree in the forest, the OOB samples (data not used to train the tree) are passed through the tree, and the prediction accuracy is recorded. To measure the importance of a variable, its values in the OOB samples are randomly permuted, and the accuracy is recomputed. The decrease in accuracy due to this permutation is averaged over all trees and used as the importance measure for the variable. This method effectively simulates the removal of the variable's contribution, providing a robust estimate of its predictive strength[2].

Results

Variable Importance and Root Frequency

To analyze the importance of features in the dataset, we computed the variable importance for a random forest with $n = 100$ trees. For comparison, we also calculated the frequency of features appearing as the root split in 100 non-random trees trained on randomized data. The results are visualized in the plot below:

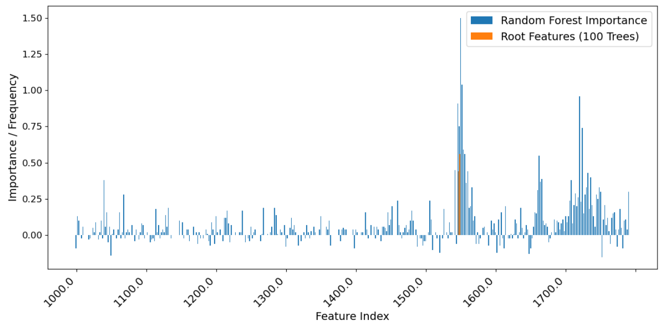

Figure 2. Variable Importance (Random Forest) vs. Root Frequency (Non-Random Trees)

The plot shows the following key observations:

- Only two features appeared as the root split in the non-random trees:
  - **Feature 1548.0**: Ranked as the **5th most important** by the random forest, this feature appeared as the root split in **56%** of the non-random trees.
  - **Feature 1546.0**: Ranked as the **4th most important** by the random forest, this feature appeared as the root split in **44%** of the non-random trees.
- The random forest identified additional important features (e.g., 1550.0, 1552.0, 1720.0) that did not appear as root splits in the non-random trees, highlighting the ability of random forests to capture complex interactions beyond simple root splits.

Part 3: Extended Variable Importance

Extended Variable Importance to Combinations of Three Variables

To extend variable importance to combinations of three variables, we implemented the **importance3()** method. This method evaluates the importance of feature combinations by permuting the values of three features simultaneously and measuring the resulting drop in prediction accuracy. Specifically:

- For each combination of three features, the values of these features are shuffled in the out-of-bag (OOB) samples.
- The decrease in accuracy due to this permutation is averaged over all trees in the forest, providing a measure of the importance of the combination.

This approach captures interactions between features, which are often critical in complex datasets like the TKI resistance data. We applied this method to a random forest with $n = 1000$ trees and compared its performance to the original single-feature importance method.

Tree Structure-Based Importance

In scenarios where the original data is unavailable but a pre-built forest exists, we implemented the **importance3_structure()** method. This method identifies the best combination of three variables by analyzing the structure of the trees in the forest:

- For each tree, we extract the features used in its splits and count how often each combination of three features appears together.
- The combination with the highest frequency across all trees is selected as the most important.

This approach leverages the fact that frequently used feature combinations in the forest are likely to be predictive. By focusing on the structure of the trees, we can identify important feature combinations without access to the original data. This method is computationally efficient and works well because it reflects the forest's inherent reliance on certain feature interactions for making accurate predictions.

Results

Comparison of Classification Trees

We trained classification trees using the top 3 features identified by three different methods:

- **Single-Feature Importance**: The top 3 features were **['1548.0', '1552.0', '1550.0']**, achieving an accuracy of **58.33%**.
- **3-Feature Combination Importance**: The top combination was **['1660.0', '1550.0', '1558.0']**, achieving an accuracy of **63.33%**.
- **Tree Structure-Based Importance**: The top combination was **['1784.0', '1546.0', '1554.0']**, achieving an accuracy of **63.33%**.

Key Observations

- The classification trees built using the top 3-feature combinations (**importance3** and **importance3_structure**) outperformed the tree built using the top 3 single features, achieving an accuracy of **63.33%** compared to **58.33%**.
- Both 3-feature combination methods (**importance3** and **importance3_structure**) performed equally well, suggesting that considering feature interactions is crucial for improving model performance.
- The **importance3_structure** method, which relies on the structure of the pre-built forest, identified a different combination of features (**['1784.0', '1546.0', '1554.0']**) but achieved the same accuracy as **importance3**. This demonstrates that the tree structure-based approach is effective even without access to the original data.

Why the Tree Structure-Based Method Works Well

Since we only have access to the random forest and not the original data, the method analyzes all splits in the trees to extract the features used. By tracking the frequency of 3-feature combinations across the forest, it selects the combination that appears most frequently. This works well because features that are frequently used together in splits are likely to capture important patterns in the data and effectively partition it. In a random forest, features that are frequently used in splits are those that provide the most information gain or Gini impurity reduction. If a combination of features is frequently used together, it suggests that these features collectively contribute to better splits, meaning they capture important patterns in the data.

Why Counting Frequencies at All Levels is Important

Counting frequencies of all features at every level, rather than just high-level features, ensures a comprehensive understanding of the data. High-level splits capture global patterns, while lower-level splits refine these with localized, nuanced interactions. Ignoring lower levels risks missing critical feature combinations that only become apparent deeper in the trees. Additionally, random forests introduce variability across trees, so evaluating all levels ensures robustness and avoids bias. This approach balances global and local importance, capturing hierarchical interactions and leading to more accurate feature selection and data partitioning.

References

[1] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[2] Trevor Hastie, Robert Tibshirani, Jerome Friedma. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*.