Distributional Modelling in R

Distributional Trees and Forests

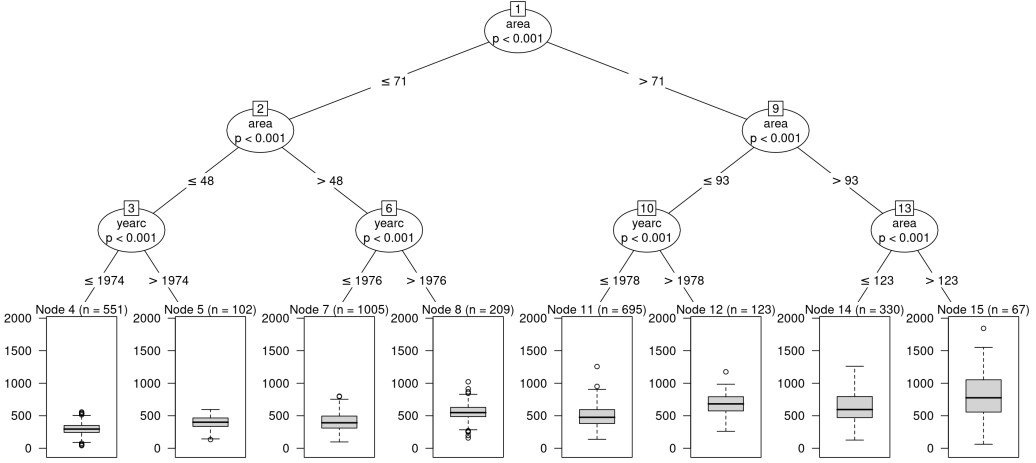
Thomas Kneib, Nikolaus Umlauf

https://nikum.org/dmr.html

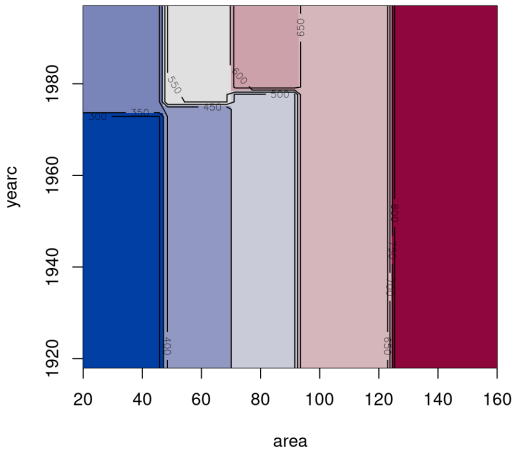
Regression Trees: Overview

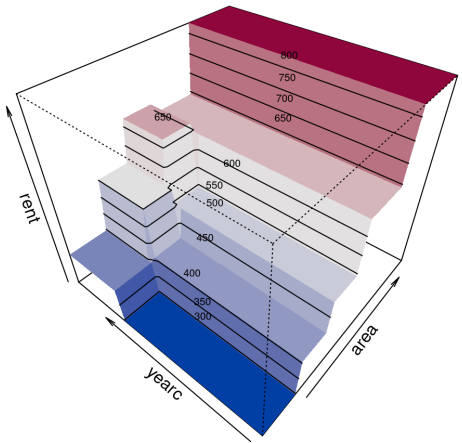
- Regression trees are a versatile and powerful non-parametric method commonly used for regression analysis in the field of machine learning and statistics.
- They provide a flexible framework for modeling the relationship between a set of predictor variables and a continuous target variable.
- At their core, regression trees partition the feature space into disjoint regions, or "leaves", and fit a simple model (usually a constant) within each region.
- The prediction for a new observation is then determined by the mean (or median) of the target variable within the leaf to which it belongs.

Regression Trees: Overview



Regression Trees: Overview





Regression Trees: Construction

The construction of a regression tree involves the following steps:

1. **Feature Selection**: At each node of the tree, a feature and a split point are selected to best divide the data into two subsets.
2. **Recursive Partitioning**: The selected feature and split point are used to partition the data into two subsets. This process is repeated recursively for each subset until a stopping criterion is met (e.g., maximum tree depth or minimum number of samples in each leaf).
3. **Leaf Prediction**: Once the tree structure is determined, a prediction value is assigned to each leaf node. This is typically done by computing the mean or median of the target variable within the leaf.

Estimation of Regression Trees

Tree partitions the feature space into a set of rectangular regions $R_1, R_2, \ldots, R_J$.

The predicted response value for each region R_j, $j = 1 \ldots, J$, is determined by an aggregation function $h_j(\cdot)$ applied to observations in R_j. The tree is given by

$$\hat{y} = f(\mathbf{x}) = \sum_{j=1}^{J} h_j(\{\mathbf{x} : \mathbf{x} \in R_j\}),$$

where

- $\hat{y}$ is the predicted response,
- $\{\mathbf{x} : \mathbf{x} \in R_j\}$ represents the set of observations falling into region R_j,
- $\mathbf{x} = (x_1, \ldots, x_K)^\top$ represents the feature (covariate) vector.

The choice of $h_j(\cdot)$ can vary depending on the application.

Estimation of Regression Trees

Least Squares:

Starting from all data, consider split variable x_k and split point s, define

$$R_1 = \{\mathbf{x}|x_k \leq s\} \quad \text{and} \quad R_2 = \{\mathbf{x}|x_k > s\}.$$

Solve

$$\min_{k,s} \left[\sum_{\mathbf{x}_i \in R_1} (y_i - h_1(\mathbf{x}_i))^2 + \sum_{\mathbf{x}_i \in R_2} (y_i - h_2(\mathbf{x}_i))^2 \right]$$

E.g., if $h_j(\mathbf{x}_i) = \beta_j I(\mathbf{x}_i \in R_j)$, with indicator function $I(\cdot)$, then $\hat{\beta}_j = ave(y_i|\mathbf{x}_i \in R_j)$.

To grow the tree, repeat the splitting process for regions R_1 and R_2, ...

Estimation of Regression Trees

General Loss:

Solve

$$\min_{k,s} \left[\sum_{\mathbf{x}_i \in R_1} L(y_i, h_1(\mathbf{x}_i)) + \sum_{\mathbf{x}_i \in R_2} L(y_i, h_2(\mathbf{x}_i)) \right].$$

E.g., $L(\cdot)$ can be the negative log-likelihood of a logistic model and if $h_j(\mathbf{x}_i) = \beta_j I(\mathbf{x}_i \in R_j)$, then

$$\hat{\beta}_j = \arg\min_{\beta_j} \left[\sum_{\mathbf{x}_i \in R_j} L(y_i, \beta_j I(\mathbf{x}_i \in R_j)) \right].$$

Stopping Criteria for Growing a Tree

- **Maximum Depth**: Limit the maximum depth of the tree to prevent overfitting.
- **Minimum Samples per Leaf**: Specify a minimum number of samples required to be in a leaf node.
- **Minimum Improvement**: Define a threshold for the minimum improvement in, e.g., log-likelihood required for splitting a node.
- **Minimum Samples for Split**: Specify a minimum number of samples required to perform a split on a node.
- **Early Stopping**: Utilize techniques like cross-validation to monitor the performance of the tree on a validation set.

Choice of Split Variable

- **Greedy Algorithm**: Decision trees typically use a greedy algorithm to select the best split variable at each node.
- **Splitting Criteria**: Common criteria include Gini impurity, entropy, and mean squared error.
- **Random Feature Selection**: In ensemble methods like Random Forest, randomly selecting a subset of features at each node can improve generalization.
- **Conditional Inference Tree Splitting**: Uses statistical tests to determine significant splits, avoiding biases and overfitting.

Random Forests

- Random forests combine the power of ensemble learning with decision trees to create robust regression models.
- Start by drawing multiple (bootstrap) samples, $D_l^\star$, from the original dataset D. Each sample has size $n_l \leq n$.
- For each sample, grow a regression tree $f_l(\mathbf{x})$.
- Repeat L times to obtain a collection of trees, $f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_L(\mathbf{x})$.
- The final prediction of the random forest, $f_{rf}^L(\cdot)$, is the aggregated result of all individual trees:

$$f_{rf}(\mathbf{x}) = \sum_{l=1}^{L} f_l(\mathbf{x}).$$

Random forests offer robustness, scalability, and high predictive accuracy.

Trees for GAMLSS I

Reminder: Any parameter of a population distribution $\mathcal{D}_y$ may be modeled by

$$y \sim \mathcal{D}_y \left(h_1(\theta_1) = \eta_1, \ \ldots, \ h_K(\theta_K) = \eta_K \right), \quad k = 1, \ldots, K,$$

and

- $h_k(\cdot)$: Link functions for each distribution parameter.
- η_k: Predictors modeled by covariates.

Now, add to the structurted additive predictor a tree model term

$$\boldsymbol{\eta}_k = f_1(\mathbf{x}_k) + \ldots + f_1(\mathbf{x}_k) + f_{\text{tree}}(\mathbf{x}_k).$$

Trees for GAMLSS I

Example:

```
R> library("gamlss2")
R> data("rent", package = "gamlss.data")
```

Model formula including regression tree.

```
R> f <- R ~  H + loc + tree(~Fl+A) |
+     H + loc + tree(~Fl+A)
```

Estimate model.

```
R> b <- gamlss2(f, data = rent, family = GA)
```

Plot estimated effects.

```
R> plot(b)
```

Residual diagnostics.

```
R> plot(b, which = "resid")
```

Trees for GAMLSS II

- Instead of creating a single tree for each parameter of the distribution, we estimate all parameters of the distribution (intercept only models) within each leaf node.
- To achieve this, we define a split variable and find the maximum likelihood estimate for $\boldsymbol{\theta}$ at each split point s.
- Split variables are chosen using the score function $s(\boldsymbol{\theta}; y_i) = \frac{\partial \ell}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}; y_i)$ and an instability test.
- The data is then split using the variable with the strongest association or instability.
- These steps are repeated recursively to grow the tree.
- Additionally, individual trees can be combined to form a distributional forest.

Trees for GAMLSS II

Example:

```
R> if(!require("disttree")) {
+     install.packages("disttree", repos = "http://R-Forge.R-project.org",
+       dependencies = TRUE)
+  }
R> library("disttree"); library("gamlss.dist")
R> data("rent", package = "gamlss.data")
```

Model formula.

```
R> f <- R ~  H + loc + Fl + A
```

Distributional tree (for forests use the `distforest()` function).

```
R> b <- disttree(f, data = rent, family = GA)
```

Plot tree structure.

```
R> plot(b)
```