Tree-based Models

Zhiyu Quan

University of Connecticut

22 March, 2018

Introduction and motivation

- Nonparametric approach – distribution free
- Interpret by visualizing the tree structure
- Robust to the outliers and missing data
- Partially solves multicollinearity
- Detect non-linear effects and interactions

Introduction and motivation

- Regression trees can fit almost all the traditional statistical models
  - Least-squares, Logistic, Poisson, Proportional Hazards models
  - Quantile, Longitudinal, multiresponse

CART (classification and regression tree)

- Grows a large tree
- Prune the large tree

CART (Grows a large tree)

- Decision tree($T(\mathbf{x}, \Theta)$) algorithm partition the explanatory variable space into disjoint regions M regions $R_1, R_2 \ldots, R_M$; then assign a constant c_m in each region which is the estimated values $\hat{y}_i$ of the response variable y_i in region R_m:

$$\hat{y}_i = f(y_i|\mathbf{x}_i, \Theta) = T(\mathbf{x}_i; \Theta) = \sum_{m=1}^{M} c_m \mathbf{1}_{R_m}(\mathbf{x}_i) \text{ where } \Theta = \{R_m, c_m\}_{m=1}^{M}$$

- Under the sum of square error(SSE), recursive binary splitting, first finds the single numerical explanatory variable $X_{\cdot j}$ which best splits the data into two regions $R_1(j, s) = \{\mathbf{x}_i | X_{\cdot j} < s\}$ and $R_2(j, s) = \{\mathbf{x}_i | X_{\cdot j} > s\}$, i.e. For any j and s,

$$\underset{j,s}{\operatorname{argmin}} \sum_{i:\mathbf{x}_i \in R_1(j,s)} (y_i - \hat{c}_{R_1(j,s)})^2 + \sum_{i:\mathbf{x}_i \in R_(j,s)} (y_i - \hat{c}_{R_2(j,s)})^2$$

CART (Prune the large tree)

- cost-complexity pruning:

$$C_\alpha(T) = \sum_{m=1}^{|T|} \sum_{\mathbf{x}_i \in R_m} (y_i - \hat{c}_m)^2 + \alpha |T|$$

- The tuning parameter $\alpha \geq 0$ governs the tradeoff between tree size and its goodness of fit to the data.

CART (Missing data)

- The CART employees a series of “surrogate” splits
- Splits on alternate explanatory variables that substitute for the best explanatory variable which has a missing value
- Surrogate splits are also used to provide variable importance.

CART Algorithm

1 **Grow** a full tree T_0 on the training data using recursive binary splitting. Stopping cariteria is *minsplit* which is the minimum number of observations that must exist in a node in order for a split to be attempted.
2 **Prune** the full tree T_0 to the subtree T_α using cost-complexity pruning.
3 **Determine** α using K-fold cross-validation, i.e. selecting best *cp* which is numerical value of α. In detail, divide the training set into K folds. For each k = 1,...,K:
4 (a) Repeat Step 1 and 2 on all except for kth fold.
5 (b) Calculate the mean squared prediction error on the hold out kth fold using T_α.
6 Finally, average the results from (b) for each value of α, and pick α to minimized the average prediction error.
7 **Return** the best subtree T_α.

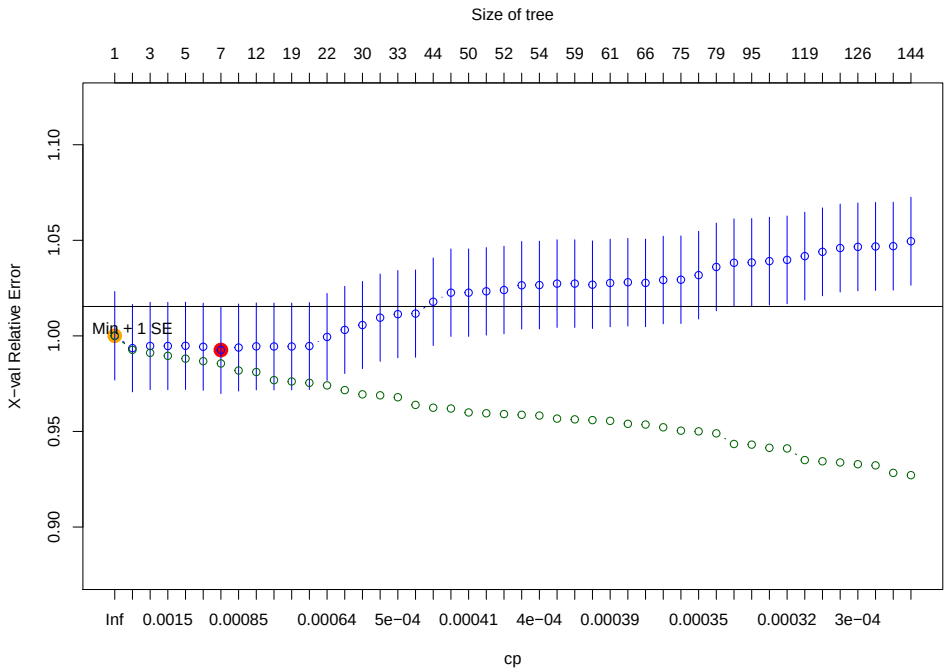
Algorithm 1: CART: R-package-rpart

Regression Tree

```
setwd("/Users/ZhiyuQuan/Downloads/DataSclass/Pricing Game/Pric
set.seed(1)
data_year1 = read.csv("data_year1.csv")  # load data
# paste(names(data_year1), collapse=' + ')
library(rpart)
# grow tree
FitRpart = rpart(log(AMOUNT + 1) ~ RISK_NATURE_1 +
    RISK_NATURE_2 + RISK_NATURE_3 + RISK_FAMILY_STRUCT +
    RISK_ROOMS_NB + RISK_PROT_1 + RISK_PROT_2 + SURF_HOUSE +
    SURF_VER + SURF_OUTB + SURF_GR + OPTION_1 + OPTION_2 +
    OPTION_3 + OPTION_4 + OPTION_5 + OPTION_6 + OPTION_7 +
    OPTION_8 + ZONE_1 + ZONE_2 + ZONE_3, method = "anova",
    data = data_year1, control = rpart.control(minsplit = 13,
        cp = 3e-04))
```

Regression Tree

```
printcp(FitRpart)  # display the results
mvpart::plotcp(FitRpart)  # display the results
```




```
# summary(FitRpart) # detailed summary of splits
```

Regression Tree

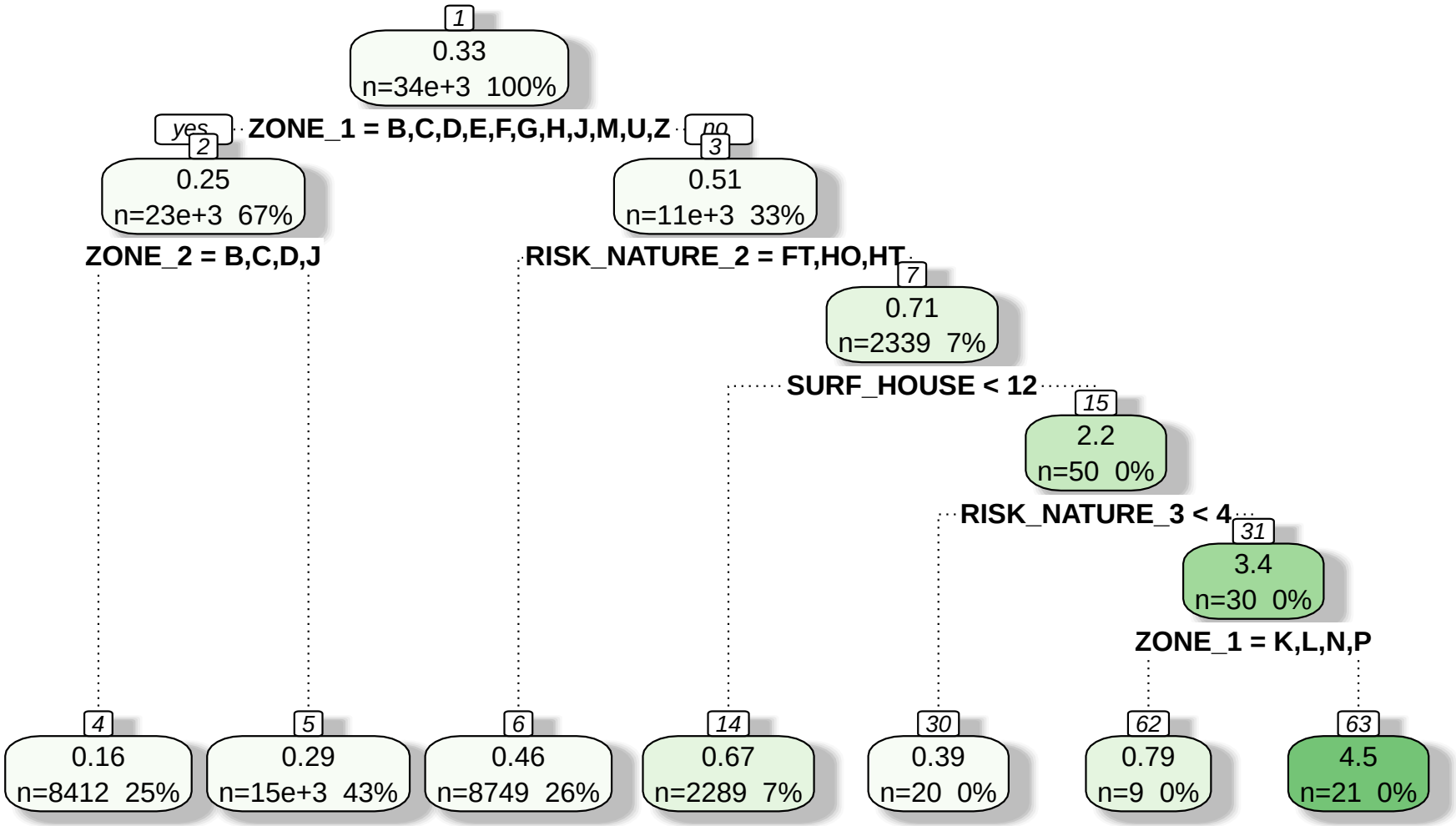


Figure 1: Univariate Regression Tree

Random Forest

- Random forest weakens the dependence among the CART trees by using a random subset of explanatory variables for split selection at each node of a tree.

Random Forest

- Ensemble methods

$$F_B(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B} T(\mathbf{x}; \Theta_b)$$

- Train on bootstrap samplings B
- Random set of explanatory variables
- Decreases the correlation between different trained trees
- As $B \to \infty$, by the Strong Law of Large Numbers,

$$E_{\mathbf{x},\mathbf{y}}(\mathbf{y} - F_B(\mathbf{x}))^2 \to E_{\mathbf{x},\mathbf{y}}(\mathbf{y} - E_\theta T(\mathbf{x}; \Theta))^2 \quad a.s.$$

Random Forest Algorithm

1 **Bootstrap** For b= 1 to B(*ntree*):
2 (a) Draw a bootstrap sample of size *sampsize* from the training data.
3 (b) Grow a full tree $T_b(\mathbf{x}; \Theta_b)$ on the bootstrap sample using recursive binary splitting. In detail, which different from Algorithm 1, select *mtry* variables at random from the p variables and stopping cariteria is *nodesize*.
4 **Return** the ensemble of trees $\{T(\mathbf{x}; \Theta_b), b = 1, 2, ..., B\}$.
5 **Averaging** $F_B(\mathbf{x}) = \frac{1}{B} \sum_{b=1}^{B} T(\mathbf{x}; \Theta_b)$

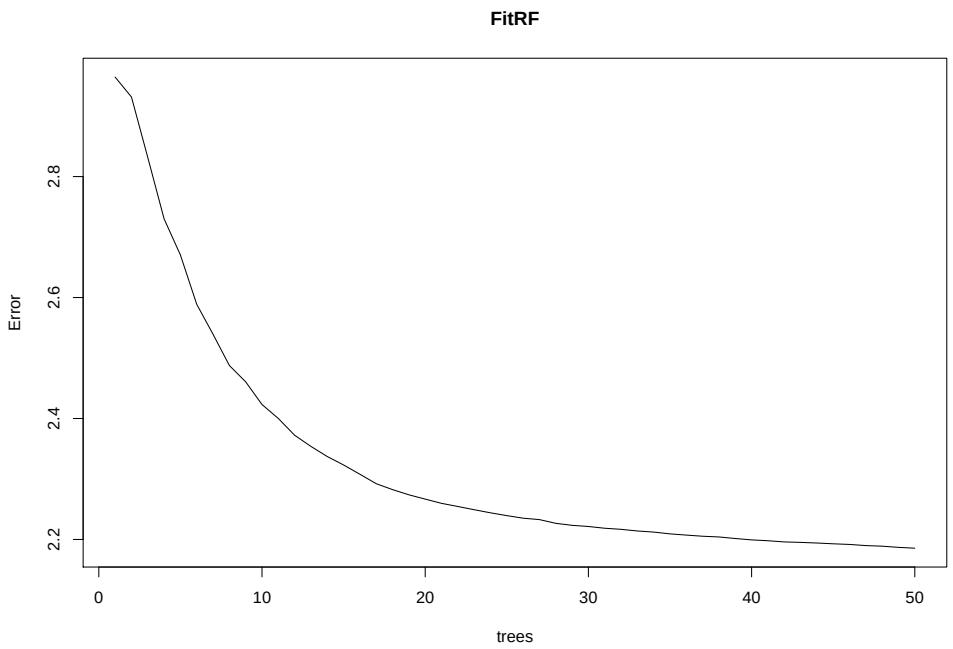
Algorithm 2: randomForest: R-package-randomForest

Random Forest

```
library(randomForest)
FitRF = randomForest(log(AMOUNT + 1) ~ RISK_NATURE_1 +
    RISK_NATURE_2 + RISK_NATURE_3 + RISK_FAMILY_STRUCT +
    RISK_ROOMS_NB + RISK_PROT_1 + RISK_PROT_2 + SURF_HOUSE +
    SURF_VER + SURF_OUTB + SURF_GR + OPTION_1 + OPTION_2 +
    OPTION_3 + OPTION_4 + OPTION_5 + OPTION_6 + OPTION_7 +
    OPTION_8 + ZONE_1 + ZONE_2 + ZONE_3, method = "anova",
    data = data_year1, importance = TRUE, ntree = 50,
    nodesize = 10, mtry = 5)
```

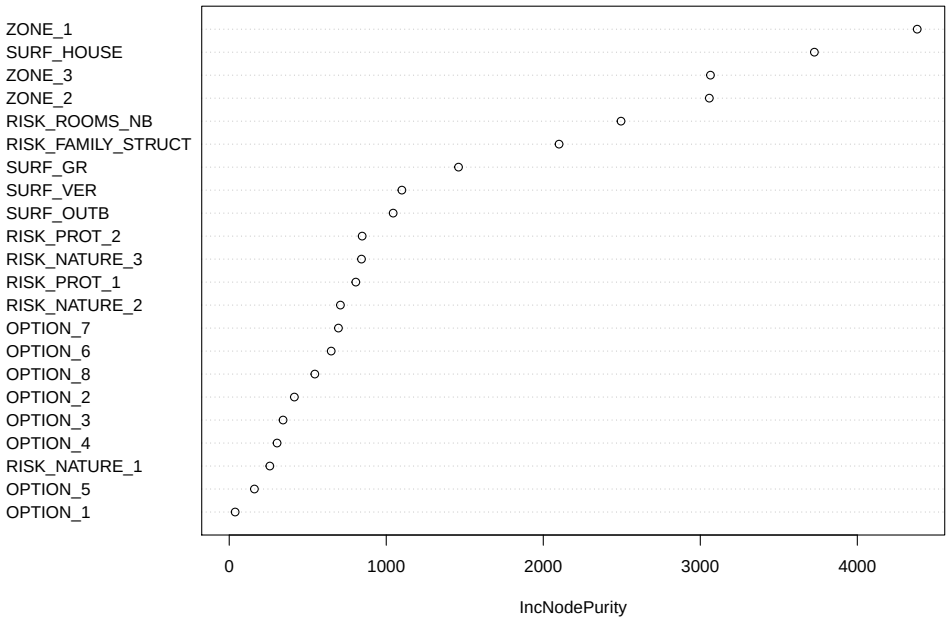
Random Forest

```
plot(FitRF)
```



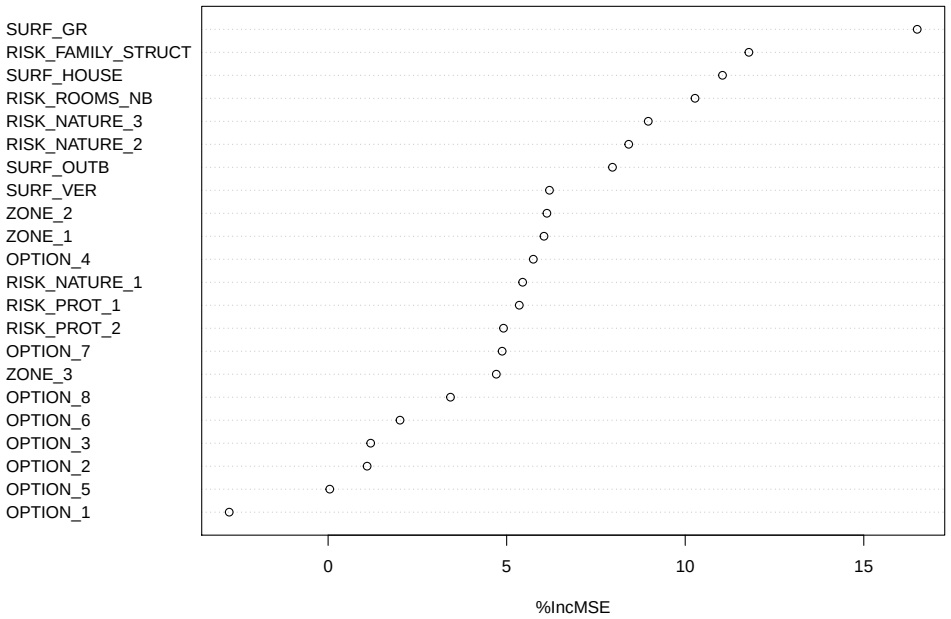
Random Forest

```
varImpPlot(FitRF, type = 2, main = NULL)   # IncNodePurity
```



Random Forest

```r
varImpPlot(FitRF, type = 1, main = NULL)   # IncMSE
```



Bagging

- Bagging uses an ensemble of unpruned CART trees constructed from bootstrap samples of the data.

Gradient Boosted Regression Trees

- Boosting sequentially constructs the trees in the ensemble by putting more weight on the observations residuals in the previous step.

Gradient Boosted Regression Trees

- Builds regression trees sequentially on residuals

$$F_b(\mathbf{x}) = F_{b-1}(\mathbf{x}) + \sum_{m=1}^{M_b} c_{mb} \mathbf{1}_{R_{mb}}(\mathbf{x})$$

$$\hat{c}_{mb} = \arg\min_{c} \sum_{\mathbf{x}_i \in R_{mb}} L(y_i, F_{b-1}(\mathbf{x}_i) + c)$$

$$F_B(\mathbf{x}) = \sum_{b=1}^{B} T_b(\mathbf{x}; \Theta_b)$$

- The tree predictions $T_b(\mathbf{x}; \Theta_b)$ at each steps are analogous to the components of the negative gradient

$$g_{ib} = -\nabla_{F_{b-1}} L(y_i, F_{b-1}(\mathbf{x}_i)) = -\left[\frac{\partial L(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right]_{F(\mathbf{x}_i)=F_{b-1}(\mathbf{x}_i)}$$

Gradient Boosted Regression Trees Algorithm

1 **Initialize** model with a constant value, $F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{N} L(y_i, \gamma)$.

2 **Gradient line search** For b= 1 to B($n.trees$):

3 (a) For i= 1 to $N * p$, p is $bag.fraction$, compute $g_{ib} = -\nabla_{F_{b-1}} L(y_i, F_{b-1}(\mathbf{x}_i))$

4 (b) Fit a regression tree $T_b(\mathbf{x}; \Theta_b)$ to the targets g_{ib} giving terminal regions $R_1, R_2 \ldots, R_{M_b}$. In detail, which different from Algorithm 1, set $interaction.depth = M_b$ and stopping cariteria is $n.minobsinnode$.

5 (c) For m= 1 to M_b compute $\hat{c}_{mb} = \arg\min_{c} \sum_{\mathbf{x}_i \in R_{mb}} L(y_i, F_{b-1}(\mathbf{x}_i) + c)$

6 (d) Update $F_b(\mathbf{x}) = F_{b-1}(\mathbf{x}) + \lambda \sum_{m=1}^{M_b} c_{mb} \mathbf{1}_{R_{mb}}(\mathbf{x})$ here we can add $shrinkage$ λ to reduce the impact of each additional fitted base-learner, regression tree, $T_b(\mathbf{x}; \Theta_b)$.

7 **Return** $F_B(\mathbf{x}) = \sum_{b=1}^{B} \lambda T_b(\mathbf{x}; \Theta_b)$.

Algorithm 3: gbm: R-package-gbm

Multivariate regression trees

- Extended the univariate regression trees
- Multivariate measure

$$L(\mathbf{y}_i, \hat{\mathbf{y}}_i) = \sum_{i=1}^{N} (\mathbf{y}_i - \hat{\mathbf{y}}_i)^T (\mathbf{y}_i - \hat{\mathbf{y}}_i)$$

$$L(\mathbf{y}_i, \hat{\mathbf{y}}_i) = \sum_{i=1}^{N} \sum_{j=1}^{k} |y_{ij} - \tilde{y}_j|$$

- Inherited univariate decision tree advantages
- Robust to the outliers and missing data
- Constrained clustering