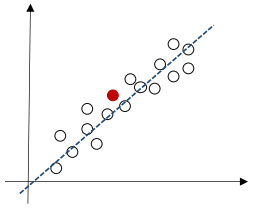
Trustworthy AI for Business and Society

Ilker Birbil

Glass Box Methods

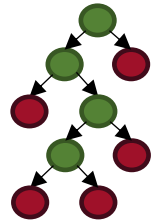
Why Glass Box?

"Additive" Models

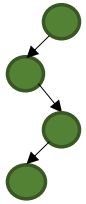


$$\hat{y} = 1.0 + 2.0x_1 - 3.0x_2$$

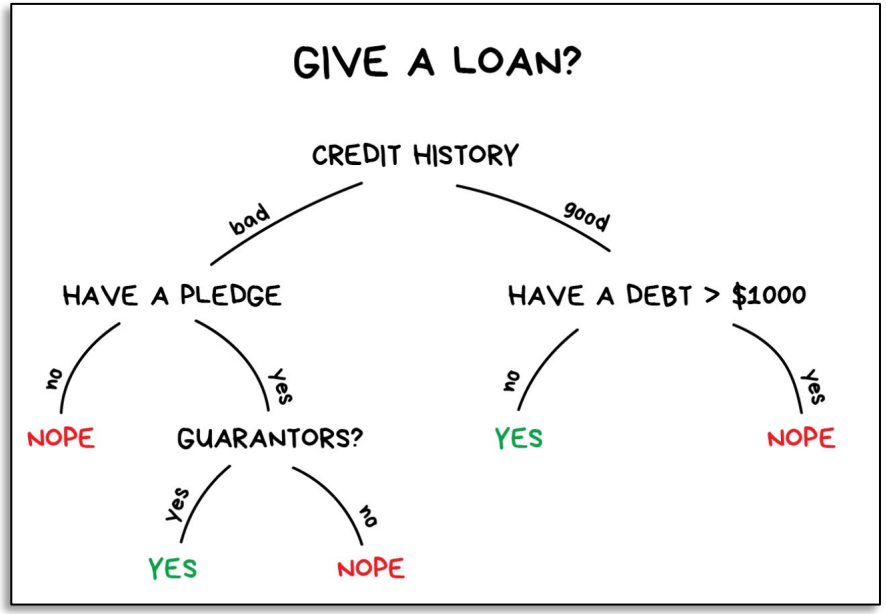
Trees



Rules



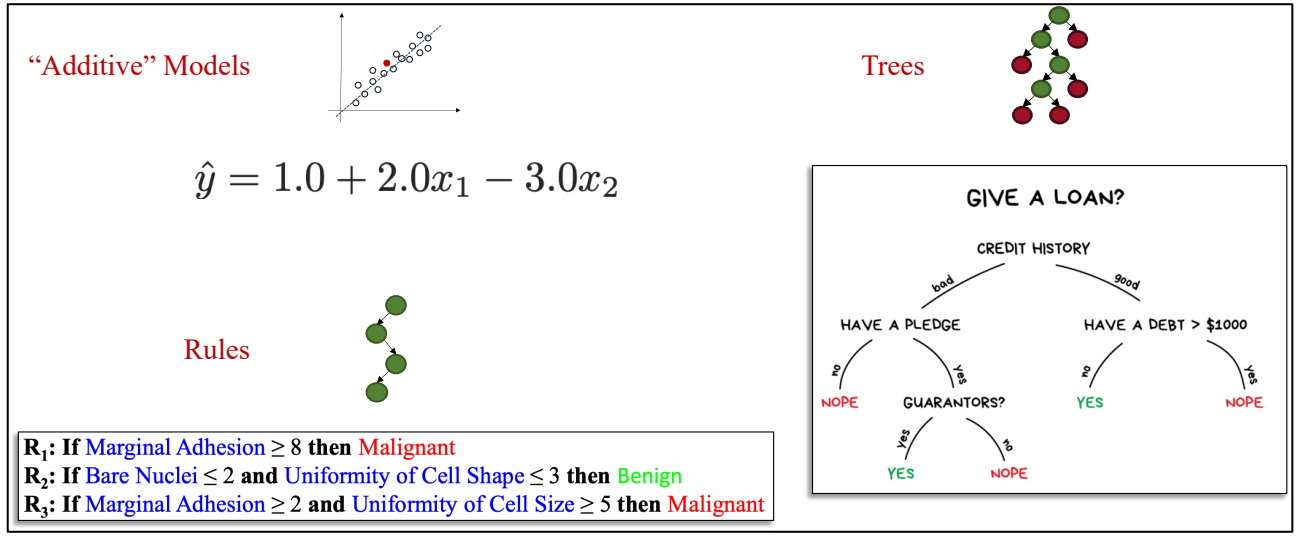
R_1: If Marginal Adhesion ≥ 8 **then** Malignant
R_2: If Bare Nuclei ≤ 2 **and** Uniformity of Cell Shape ≤ 3 **then** Benign
R_3: If Marginal Adhesion ≥ 2 **and** Uniformity of Cell Size ≥ 5 **then** Malignant



(link)

Big Picture

- Linear Models
- Generalized Additive Models
- Explainable Boosting Machines
- Decision Trees
- Rule-based Approaches



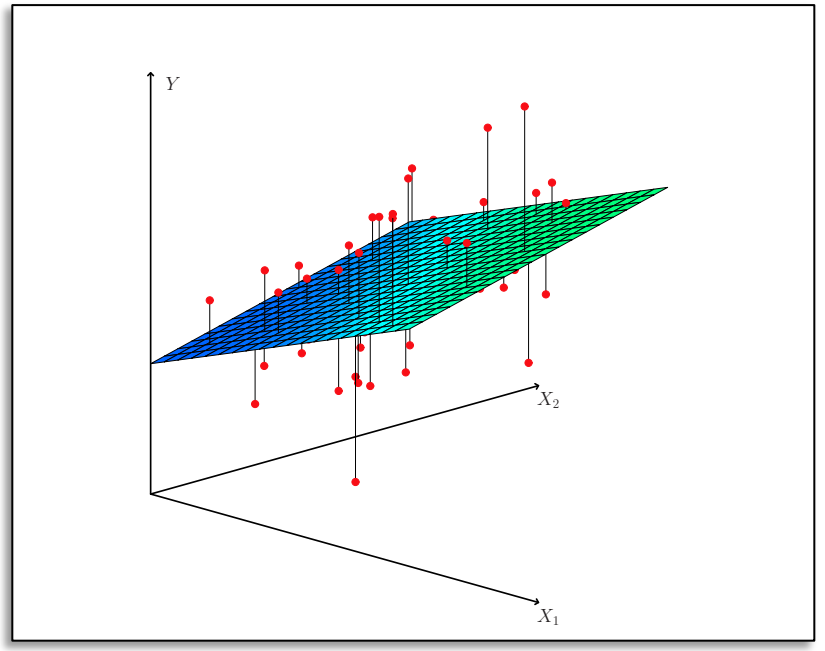
Linear Models

$$\{(x_i, y_i) : i = 1, \ldots, n\}$$
$$x_i \in \mathbb{R}^p, y_i \in \mathbb{R}$$

$$Y \approx \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$$

$$y_i \approx \underbrace{\hat{\beta}_0 + \hat{\beta}_1 x_{i1} + \hat{\beta}_2 x_{i2} + \cdots + \hat{\beta}_p x_{ip}}_{\hat{y}_i}, \quad i = 1, \ldots, n$$

prediction



(link)

Least Squares Method

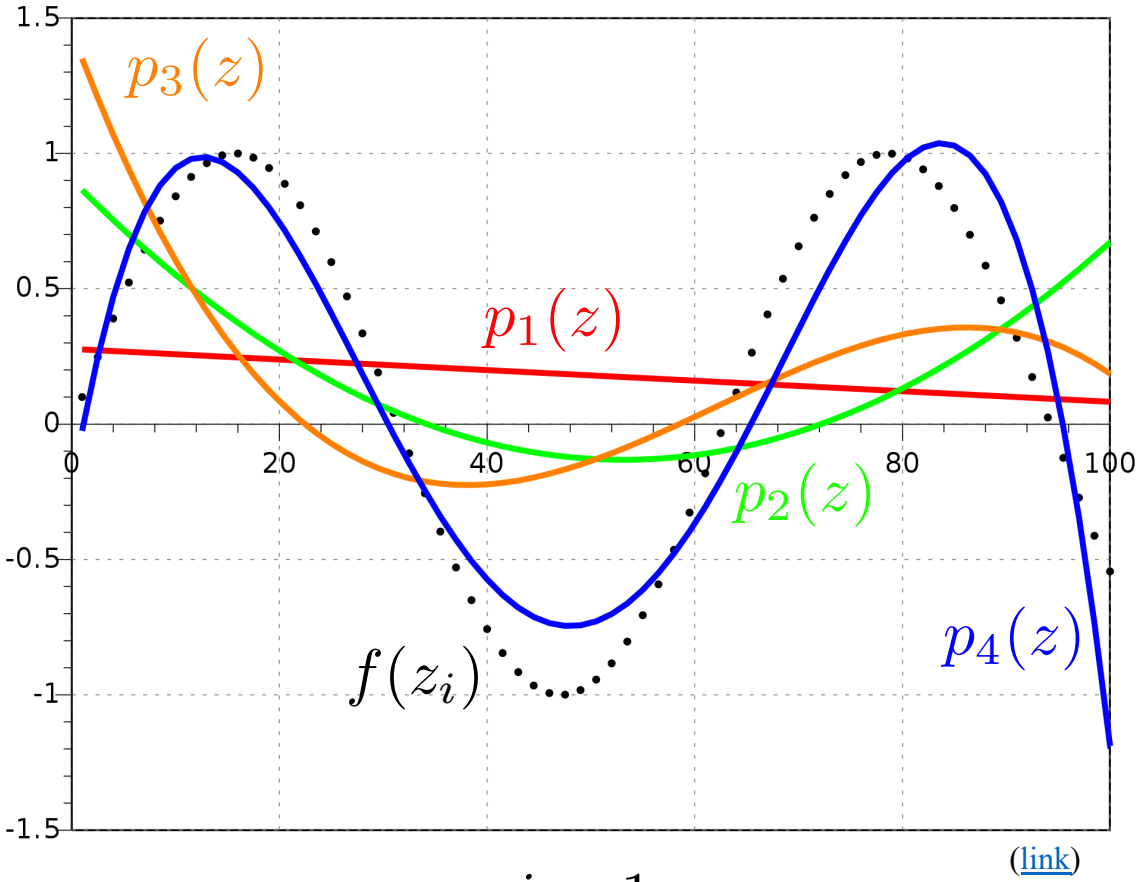
$$\min_{\beta_0, \beta} \frac{1}{2n} \sum_{i=1}^{n} \left(y_i - \beta_0 - x_i^\intercal \beta\right)^2$$

$$\beta = (\beta_1, \ldots, \beta_p)^\intercal$$

$$\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p$$

Linear Models

$$Y \approx \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$$



(link)

$z_i, \quad i = 1, \ldots, n$

$$p_4(z_i) = \hat{\beta}_0 + \hat{\beta}_1 z_i + \hat{\beta}_2 z_i^2 + \hat{\beta}_3 z_i^3 + \hat{\beta}_4 z_i^4$$

$$x_i = (z_i, z_i^2, z_i^3, z_i^4)^\intercal \in \mathbb{R}^4, \quad i = 1, \ldots, n$$

$$\hat{y}_i = p_4(z_i)$$

Training Data $\{(x_i, \underbrace{f(z_i)}_{y_i}) : i = 1, \ldots, n\}$

Linear Regression and Regularization

Linear

$$\min_{\beta_0,\beta} \frac{1}{2n}\sum_{i=1}^{n}\left(y_i-\beta_0-x_i^\intercal\beta\right)^2$$

Ridge

$\lambda \geq 0$

$$\min_{\beta_0,\beta} \frac{1}{2n}\sum_{i=1}^{n}\left(y_i-\beta_0-x_i^\intercal\beta\right)^2+\frac{\lambda}{2}\|\beta\|_2^2$$

Lasso

$\lambda \geq 0$

$$\min_{\beta_0,\beta} \frac{1}{2n}\sum_{i=1}^{n}\left(y_i-\beta_0-x_i^\intercal\beta\right)^2+\lambda\|\beta\|_1$$

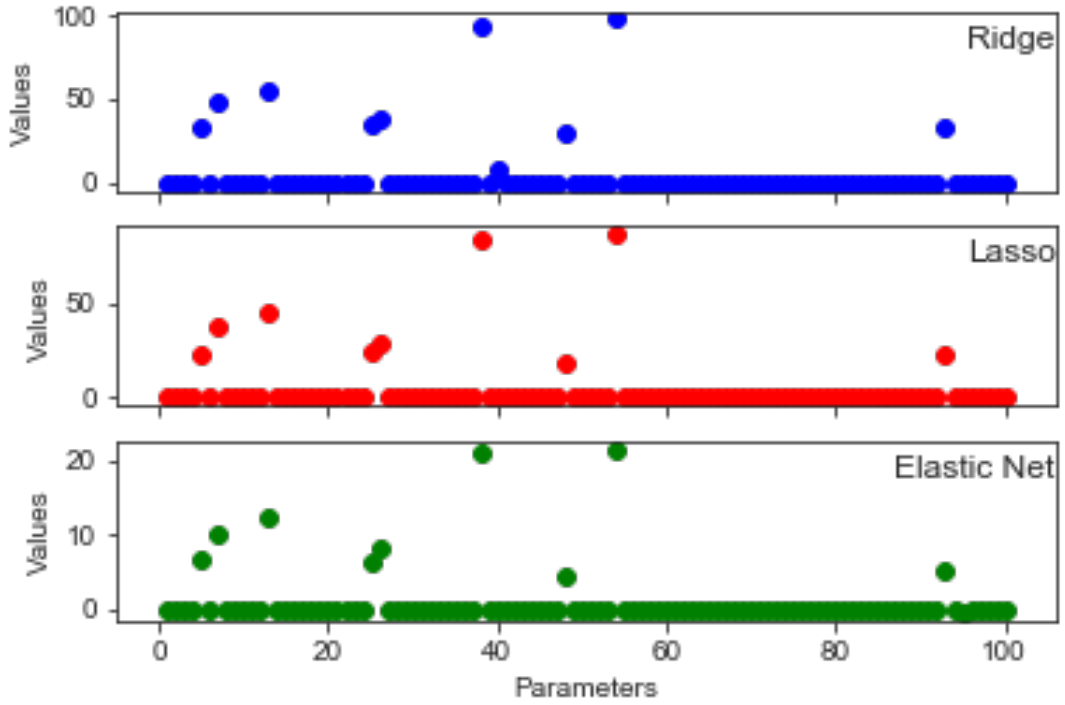
Elastic net

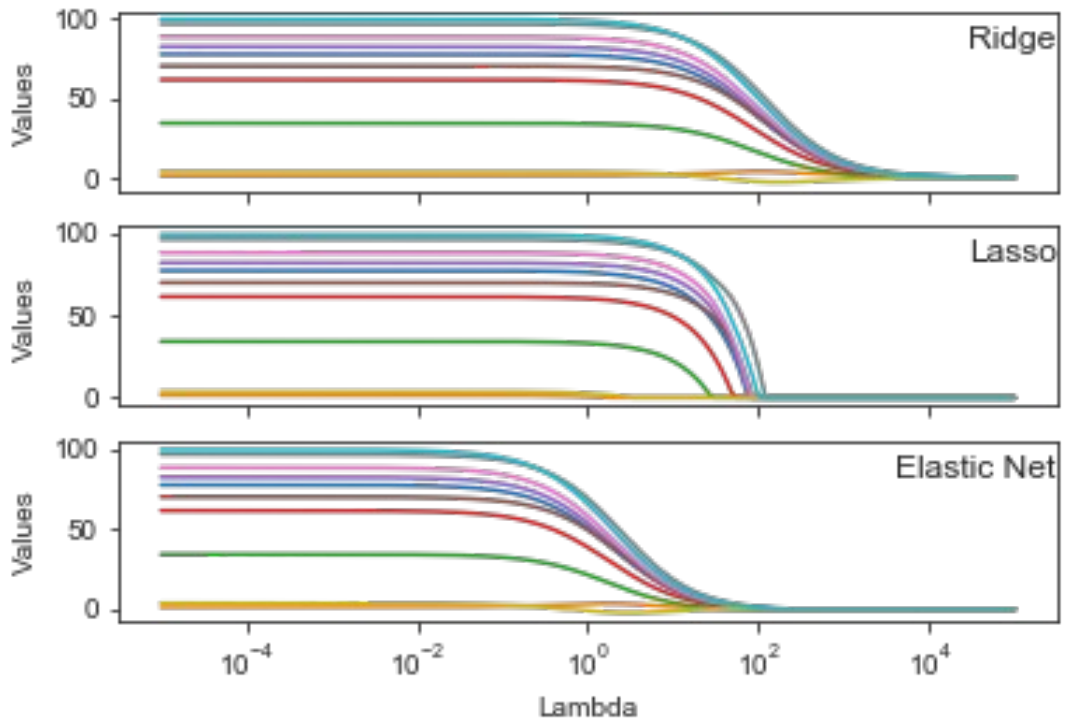
$\lambda \geq 0,\ \alpha \in [0,1]$

$$\min_{\beta_0,\beta} \frac{1}{2n}\sum_{i=1}^{n}\left(y_i-\beta_0-x_i^\intercal\beta\right)^2+\lambda\left(\frac{1-\alpha}{2}\|\beta\|_2^2+\alpha\|\beta\|_1\right)$$

λ, α hyperparameters

Regularization





Linear Regression and Regularization

β_0 dropped
(output centered, input standardized)

$$X = \begin{bmatrix} x_1^\intercal \\ x_2^\intercal \\ \vdots \\ x_n^\intercal \end{bmatrix} \in \mathbb{R}^{n \times p}$$

$$y = (y_1, \ldots, y_n)^\intercal$$

$$\beta = (\beta_1, \ldots, \beta_p)^\intercal$$

Linear $\quad \min_\beta \ \frac{1}{2n} \|y - X\beta\|_2^2$

Ridge $\quad \min_\beta \ \frac{1}{2n} \|y - X\beta\|_2^2 + \frac{\lambda}{2} \|\beta\|_2^2$

Lasso $\quad \min_\beta \ \frac{1}{2n} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1$

Elastic net $\quad \min_\beta \ \frac{1}{2n} \|y - X\beta\|_2^2 + \lambda \left(\frac{1-\alpha}{2} \|\beta\|_2^2 + \alpha \|\beta\|_1 \right)$

Regularization - Maximum a Posteriori Estimation

$$y_i \sim \mathcal{N}(\hat{y}_i, \sigma^2), i = 1, \ldots, n \text{ and } i.i.d$$

$$\mathbb{P}(y|\beta) = \prod_{i=1}^{n} \mathbb{P}(y_i|\beta)$$

β_0 dropped
(output centered, input standardized)

$$X = \begin{bmatrix} x_1^\intercal \\ x_2^\intercal \\ \vdots \\ x_n^\intercal \end{bmatrix} \in \mathbb{R}^{n \times p}$$

$$y = (y_1, \ldots, y_n)^\intercal$$

$$\beta = (\beta_1, \ldots, \beta_p)^\intercal$$

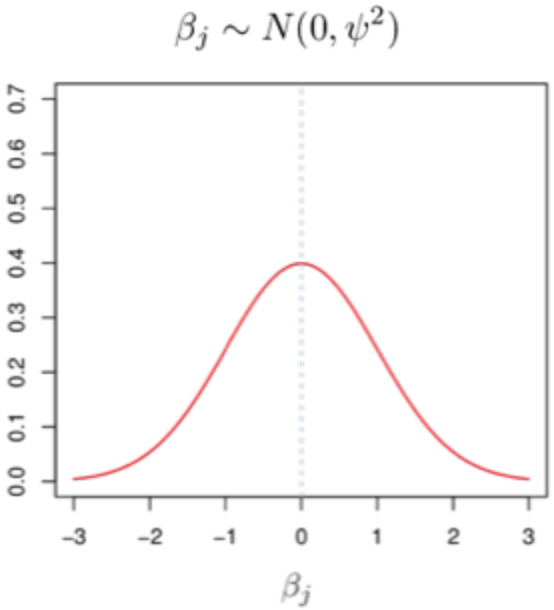
$$\mathbb{P}(\beta|y) \propto \mathbb{P}(y|\beta)\,\mathbb{P}(\beta)$$

$$\begin{aligned} \hat{\beta} &= \arg\max_{\beta} \; \mathbb{P}(y|\beta)\,\mathbb{P}(\beta) \\ &= \arg\max_{\beta} \; \log \mathbb{P}(y|\beta) + \log \mathbb{P}(\beta) \\ &= \arg\max_{\beta} \; \sum_{i=1}^{n} \log \mathbb{P}(y_i|\beta) + \log \mathbb{P}(\beta) \end{aligned}$$

Regularization - Maximum a Posteriori Estimation

prior, $\mathbb{P}(\beta)$

$$\hat{\beta} = \arg\max_{\beta} \sum_{i=1}^{n} \log \mathbb{P}(y_i|\beta) + \log \mathbb{P}(\beta)$$

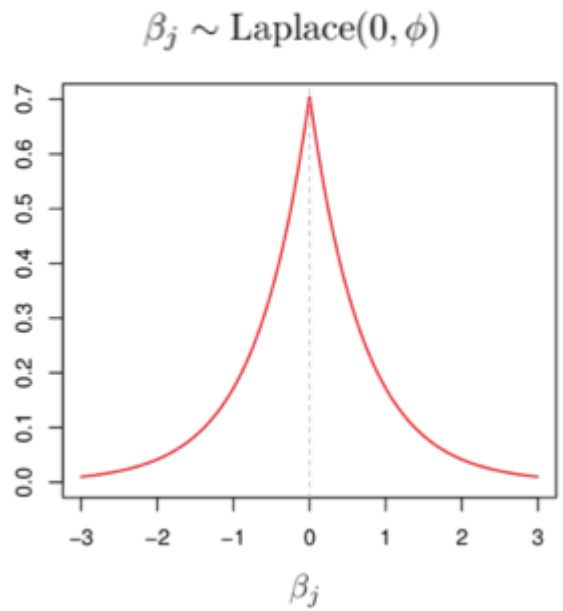


constant prior

Linear

$$\beta_j \sim \mathcal{N}(0, \psi^2),\ j = 1, \ldots, p \text{ and } i.i.d^*$$

Ridge



$$\beta_j \sim \text{Laplace}(0, \phi),\ j = 1, \ldots, p \text{ and } i.i.d^*$$

Lasso

* $\mathbb{P}(\beta) = \prod_{j=1}^{p} \mathbb{P}(\beta_j)$

Classification and Regularization

$$\{(x_i, y_i) : i = 1, \ldots, n\}$$
$$x_i \in \mathbb{R}^p, \; y_i \in \{0, 1\}$$

$$\mathbb{P}(Y = 1 | X = x_i) = \frac{1}{1 + e^{-(\beta_0 + x_i^\intercal \beta)}} = p(x_i)$$
$$\mathbb{P}(Y = 0 | X = x_i) = \frac{1}{1 + e^{(\beta_0 + x_i^\intercal \beta)}} = 1 - p(x_i)$$

mean of binary response

$$\mu(X) = \mathbb{P}(Y = 1 | X)$$

Log Odds $\log\left(\frac{\mu(X)}{1 - \mu(X)}\right) = \beta_0 + X^\intercal \beta$

Maximizing Log Likelihood

$$\max_{\beta_0, \beta} \frac{1}{n} \sum_{i=1}^{n} \underbrace{(y_i \log p(x_i) + (1 - y_i) \log(1 - p(x_i))}_{y_i(\beta_0 + x_i^\intercal \beta) - \log(1 + e^{(\beta_0 + x_i^\intercal \beta)})}$$

Classification and Regularization

Logistic Regression

$$\min_{\beta_0,\beta} \frac{1}{n}\sum_{i=1}^{n}\left(\log(1+e^{(\beta_0+x_i^\intercal\beta)})-y_i(\beta_0+x_i^\intercal\beta)\right)$$

Logistic Regression with Regularization

$$\min_{\beta_0,\beta} \frac{1}{n}\sum_{i=1}^{n}\left(\log(1+e^{(\beta_0+x_i^\intercal\beta)})-y_i(\beta_0+x_i^\intercal\beta)\right)+\lambda\left(\tfrac{1-\alpha}{2}\|\beta\|_2^2+\alpha\|\beta\|_1\right)$$

$\lambda \geq 0,\ \alpha \in [0,1]$ hyperparameters

Interpretation

Features
balance
income
student (1) or not (0)

Output
default (1) or not (0)

$p = 1$

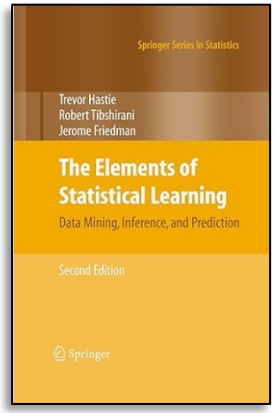
	Coefficient	Std. Error	*Z*-statistics	*p-value*
Intercept	-3.5041	0.0707	-49.55	< 0.0001
Student (1)	0.4049	0.1150	3.52	0.0004

$p > 1$

	Coefficient	Std. Error	*Z*-statistics	*p-value*
Intercept	-10.8690	0.4923	-22.08	< 0.0001
Balance	0.0057	0.0002	24.74	< 0.0001
Income	0.0030	0.0082	0.37	0.7115
Student (1)	-0.6468	0.2362	-2.74	0.0062

Confounding Effect

Interpretation



(link)

TABLE 4.2. *Results from a logistic regression fit to the South African heart disease data.*

	Coefficient	Std. Error	Z Score
(Intercept)	−4.130	0.964	−4.285
sbp	0.006	0.006	1.023
tobacco	0.080	0.026	3.034
ldl	0.185	0.057	3.219
famhist	0.939	0.225	4.178
obesity	-0.035	0.029	−1.187
alcohol	0.001	0.004	0.136
age	0.043	0.010	4.184

TABLE 4.3. *Results from stepwise logistic regression fit to South African heart disease data.*

	Coefficient	Std. Error	Z score
(Intercept)	−4.204	0.498	−8.45
tobacco	0.081	0.026	3.16
ldl	0.168	0.054	3.09
famhist	0.924	0.223	4.14
age	0.044	0.010	4.52

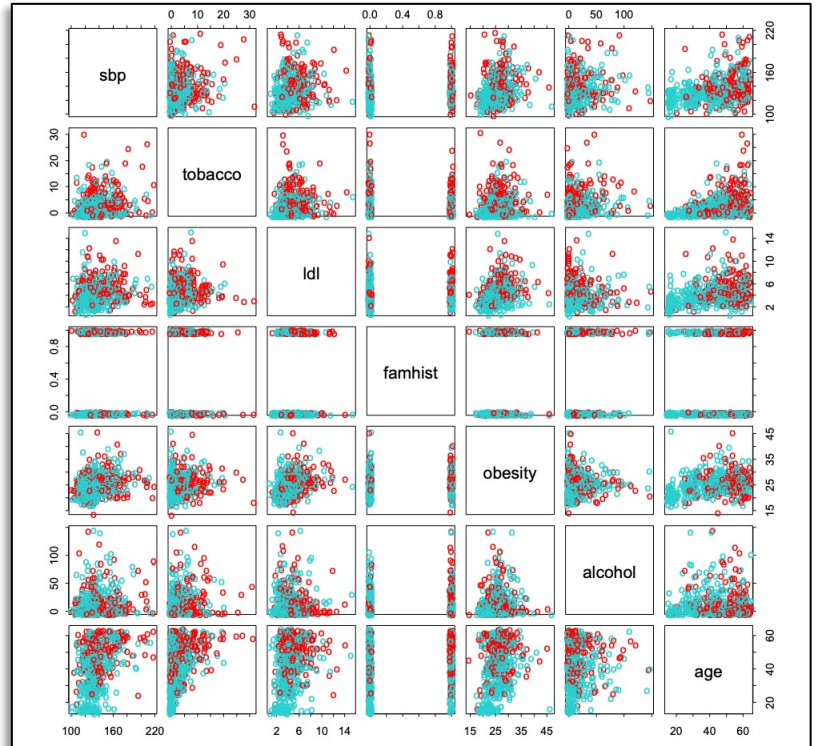


FIGURE 4.12. *A scatterplot matrix of the South African heart disease data. Each plot shows a pair of risk factors, and the cases and controls are color coded (red is a case). The variable* family history of heart disease (famhist) *is binary (yes or no).*

Interpretation

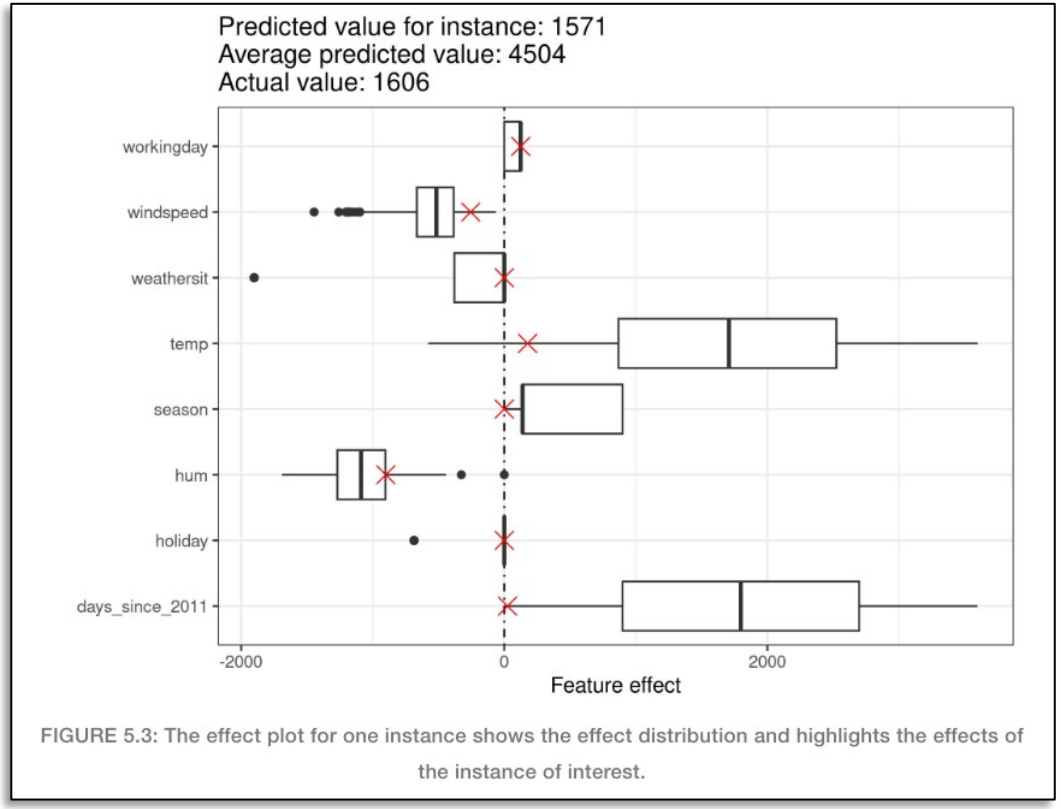


FIGURE 5.3: The effect plot for one instance shows the effect distribution and highlights the effects of the instance of interest.

(link)

 Anything missing? Please send feedback to christoph.molnar.ai@gmail.com version 1.0

Logistic Regression – Interpretation Cheat Sheet

Summary

Logistic regression is a statistical model used to predict the probability of an outcome. *Example*: Predict if a customer will default on their credit card loan (Y=1) or not (Y=0) based on income, remaining balance, and student status.

Probability

$$\pi = P(Y = 1|X) = \frac{1}{1 + exp(-(\beta_0 + \beta_1 X_1 + ... + \beta_p X_p))}$$

π is the probability of an outcome ($Y = 1$) given covariates $X_1, ..., X_p$ and coefficients $\beta_0, \beta_1, ..., \beta_p$. *Example*: Probability of loan default as a function of income, credit balance, and student status. Unfortunately, coefficients can't be interpreted on the level of probabilities, but we have to use (log) odds.

Odds

$$Odds = \frac{\pi}{1 - \pi} = exp(\beta_0 + \beta_1 X_1 + ... + \beta_p X_p)$$

Expected no. of $Y = 1$ per $Y = 0$. *Example*: Odds of 1.5 means 1.5 loan defaults per non-defaults.

Log Odds

$$ln\left(\frac{\pi}{1 - \pi}\right) = \beta_0 + \beta_1 X_1 + ... + \beta_p X_p$$

Logarithm of odds. Also called logits. All following interpretation templates work for log odds when replacing $exp(\beta_j)$ with β_j, "odds" with "log odds", and removing "multiplicatively".

Example: loan default

covariate	β	$exp(\beta)$	95% CI of β	p-value
Intercept	-11	0.0	[-14.4;-8.4]	0.00
student	-1.8	0.16	[-3.4; -0.3]	0.02
balance	0.62	1.86	[0.48; 0.79]	0.00
income	-0.25	0.78	[-5.0; 4.6]	0.92

- student "yes" or "no"; categorical; reference: "no"
- balance: outstanding credit card balance in $100s
- income: yearly income in $100 000
- CI stands for confidence interval of β

Intercept

$exp(\beta_0)$ is the odds of $Y = 1$ given all continuous covariates are 0 and all categorical covariates are set to the reference category. *Example*: The odds of loan default is $exp(-11) \approx 0.000017$ given student=no, balance=0 and income=0.

Continuous Coefficient

Template: A one-unit increase in X_j changes the odds for $Y = 1$ multiplicatively by $exp(\beta_j)$, holding all other covariates constant. *Example*: A $100 increase in credit balance increases the odds for loan default 1.86-fold, holding student status and income constant.

- Positive $\beta_j \rightarrow$ increasing X_j increases odds.
- Negative $\beta_j \rightarrow$ increasing X_j decreases odds.

Categorical Coefficient

Template: Changing X_j from [*reference*] to [*other category*] changes the odds for $Y = 1$ multiplicatively by $exp(\beta_j)$, holding all other covariates constant. *Example*: Changing student status from "no" to "yes" decreases the odds of loan default by a factor of 0.16, holding all other covariates constant.

p-value

The p-value for a coefficient β_j is the probability to observe this value or a more extreme one under the Null Hypothesis of $\beta_j = 0$ (meaning that the covariate doesn't affect the odds of the outcome).
If $p < \alpha$, a pre-defined threshold (often $\alpha = 0.05$), the coefficient is significantly different from 0. *Example*: The effect of student status is significantly different from 0 ($p = 0.02$).

Interpretation of Confidence Interval

- The confidence interval measures the uncertainty of a coefficient.
- If you were to repeat the analysis 100 times with new samples and compute a 95% confidence interval each time, you would expect 95/100 of the intervals to contain the true coefficient.
- If the confidence interval doesn't cover 0, it's equivalent to rejecting the Null Hypothesis to the chosen α-level.

(Log) Odds Ratio

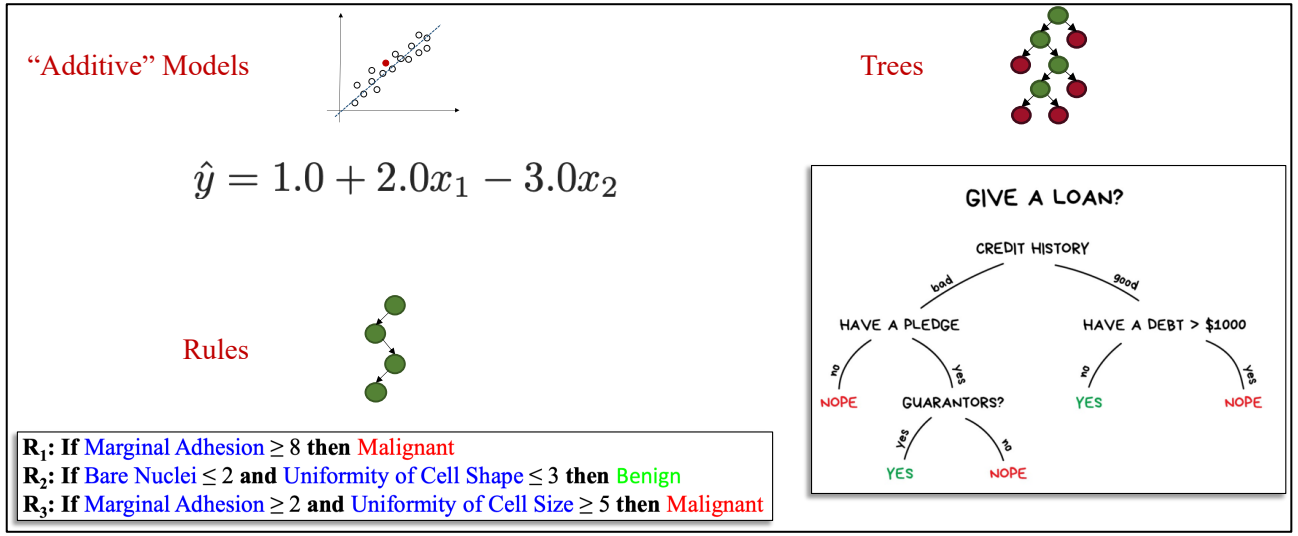
$$OR = \frac{Odds_1}{Odds_2} \quad \text{and} \quad logOR = \frac{ln(Odds_1)}{ln(Odds_2)}$$

(Log) odds ratios compare (log) odds of two data points.
Coefficient interpretations are based on (log) odds ratios, as they compare data points with $X_j + 1$ and with X_j: $OR = Odds(X_j + 1)/Odds(X_j) = exp(\beta_j)$. For categorical covariates the ratio reflects the change from reference to another category. I recommend just using the provided templates and not getting a headache from the (log) odds ratios.

(link)

Big Picture

- Linear Models
- Generalized Additive Models
- Explainable Boosting Machines
- Decision Trees
- Rule-based Approaches



Generalized Additive Models (GAMs)

Model	Form	Intelligibility	Accuracy
Linear Model	$y = \beta_0 + \beta_1 x_1 + ... + \beta_n x_n$	+++	+
Generalized Linear Model	$g(y) = \beta_0 + \beta_1 x_1 + ... + \beta_n x_n$	+++	+
Additive Model	$y = f_1(x_1) + ... + f_n(x_n)$	++	++
Generalized Additive Model	$g(y) = f_1(x_1) + ... + f_n(x_n)$	++	++
Full Complexity Model	$y = f(x_1, ..., x_n)$	+	+++

Table 1: From Linear to Additive Models.

(link)

$$g(\mathbb{E}[Y]) = f_0 + \sum_{j=1}^{p} f_j(X_j)$$

contribution of each feature

link function (regression or classification)

shape functions (splines, trees, …)

Backfitting: The shape functions are trained in-place* sequentially by fitting each one to the residual up to that iteration.

When (gradient) boosting is used to learn the shape functions, at each iteration of boosting the algorithm cycles through the features.

* previously trained version is discarded

Generalized Additive Models (GAMs)

$$\log\left(\frac{\mu(X)}{1-\mu(X)}\right) = \beta_0 + X^\intercal \beta$$

$$g(\mathbb{E}[Y]) = f_0 + \sum_{j=1}^{p} f_j(X_j)$$

link function
(regression or classification)

regression
(*identity*)

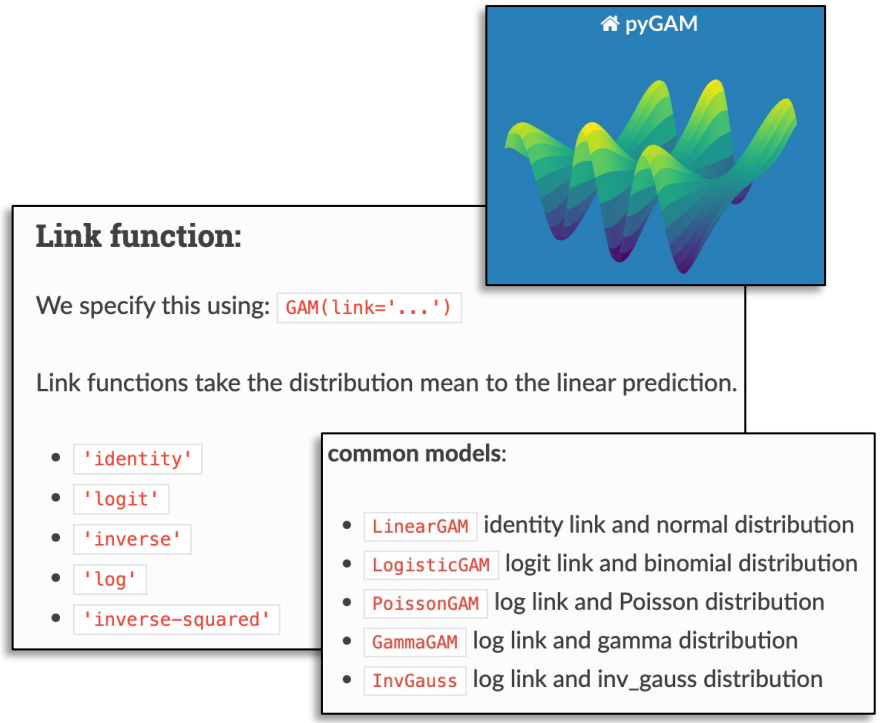
$$g(\mathbb{E}[Y]) = Y$$

binary classification
(*logit*)

$$g(\mathbb{E}[Y]) = \log\left(\frac{\mathbb{E}[Y]}{1-\mathbb{E}[Y]}\right)$$

count data
(*log*)

$$g(\mathbb{E}[Y]) = \log(\mathbb{E}[Y])$$



Interpretation

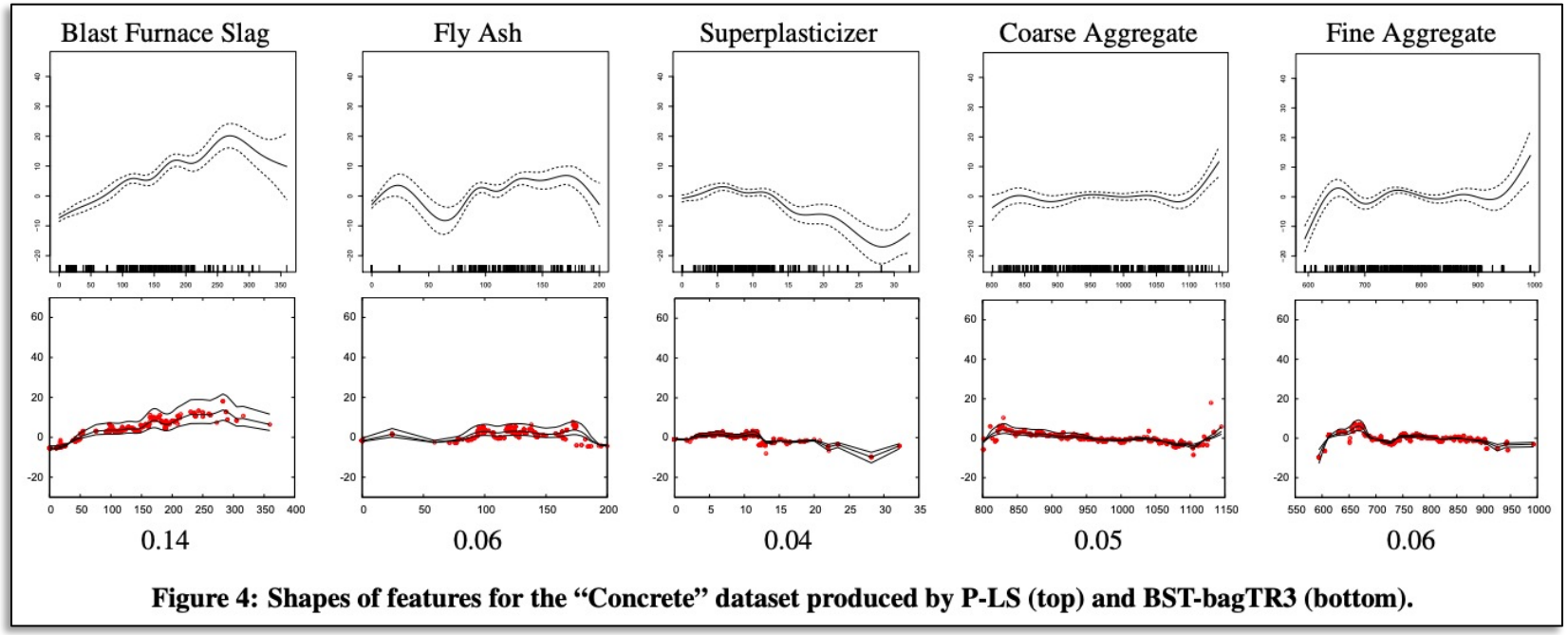
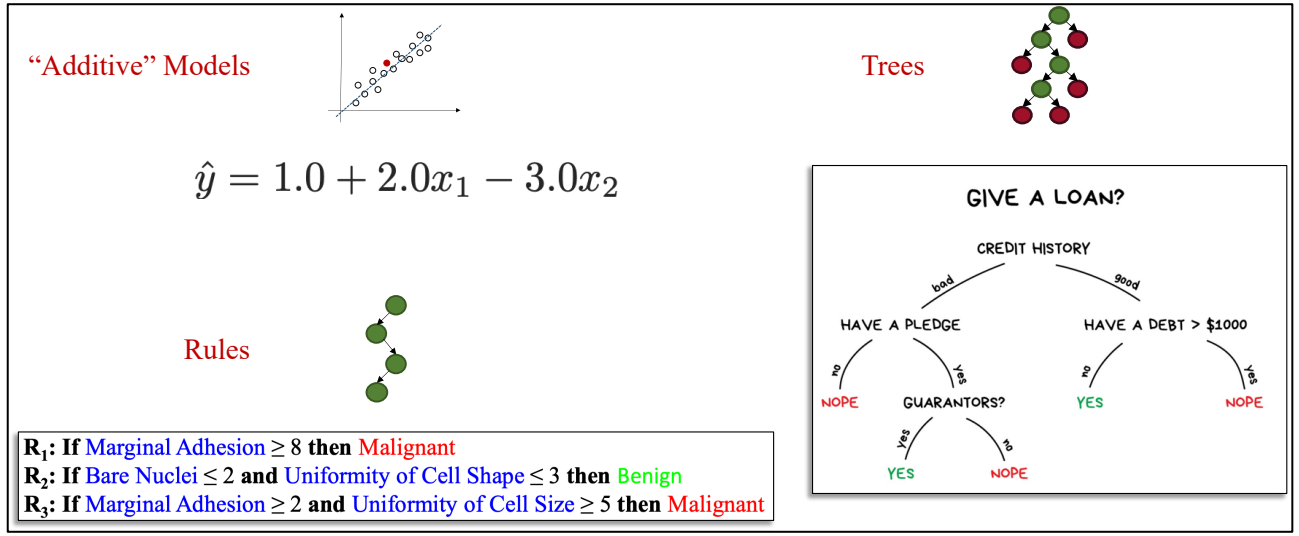


Figure 4: Shapes of features for the "Concrete" dataset produced by P-LS (top) and BST-bagTR3 (bottom).

Lou, Y., Caruana, R. & Gehrke, J. "Intelligible models for classification and regression," in proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining KDD '12, 150–158, 2012.

Big Picture

- Linear Models
- Generalized Additive Models
- Explainable Boosting Machines
- Decision Trees
- Rule-based Approaches



Explainable Boosting Machines (EBMs)

> Our first contribution in this paper is to build models that are more powerful than GAMs, but are still intelligible. We observe that two-dimensional interactions can still be rendered as heatmaps of $f_{ij}(x_i, x_j)$ on the two-dimensional x_i, x_j-plane, and thus a model that includes only one- and two-dimensional components is still intelligible.

(link)

$$g(\mathbb{E}[Y]) = f_0 + \sum_{j=1}^{p} f_j(X_j) + \sum_{j=1}^{p} \sum_{k \neq j}^{p} f_{jk}(X_j, X_k)$$

one- and two-dimensional shape functions

cross-interaction between two features

When the number of feature is high, considering all interactions is very costly. Thus, only strong interactions are used after ranking the interactions.

Testing strong interactions: ANOVA, partial dependence, statistical tests, greedy forward selection strategy.

Interpretation

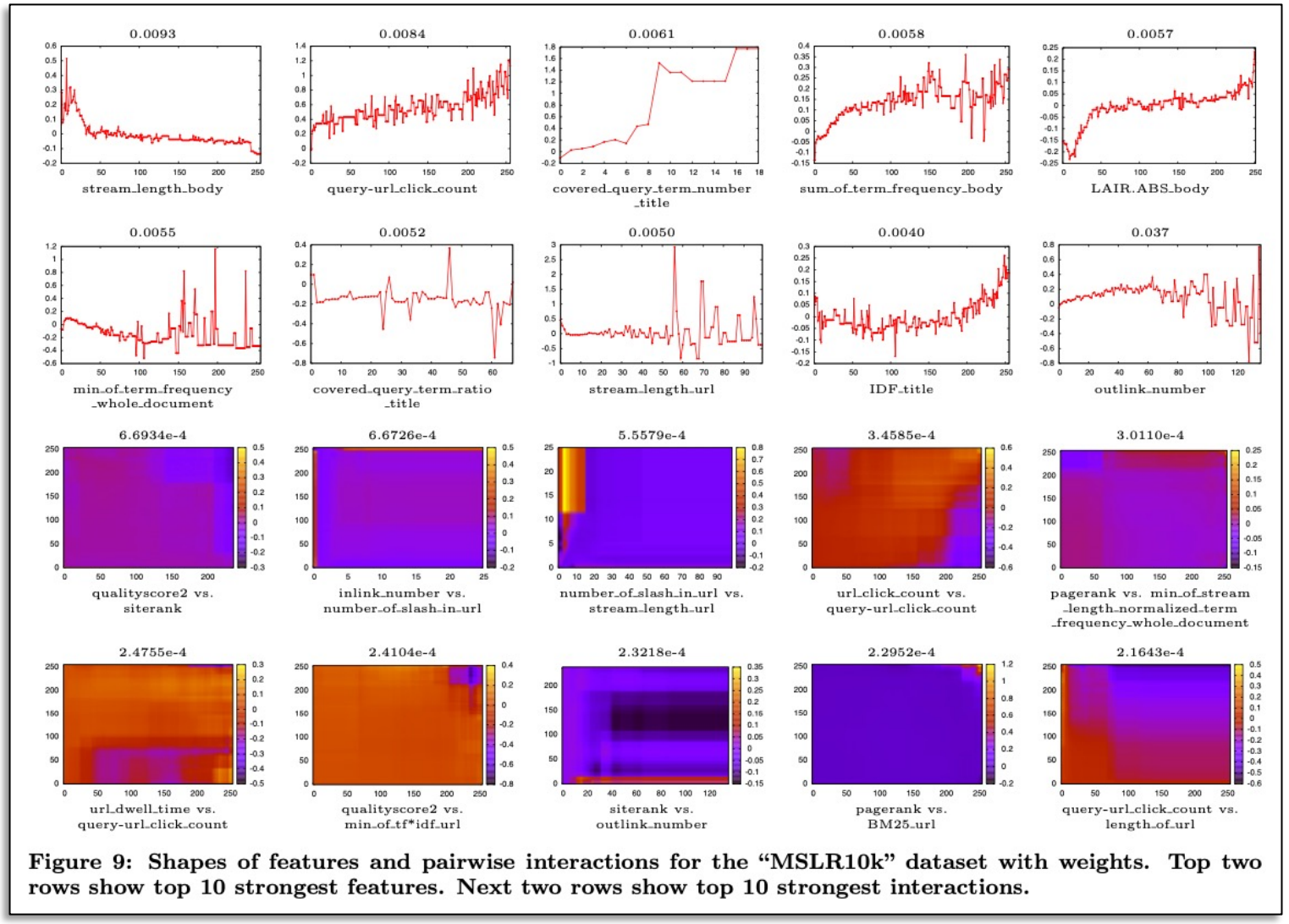
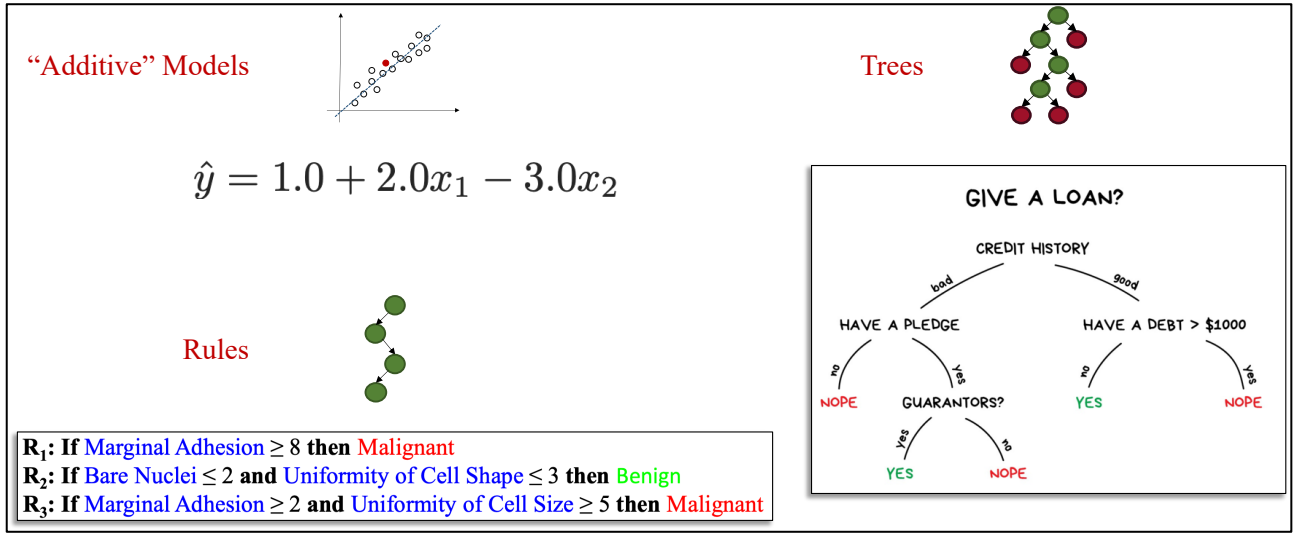


Figure 9: Shapes of features and pairwise interactions for the "MSLR10k" dataset with weights. Top two rows show top 10 strongest features. Next two rows show top 10 strongest interactions.

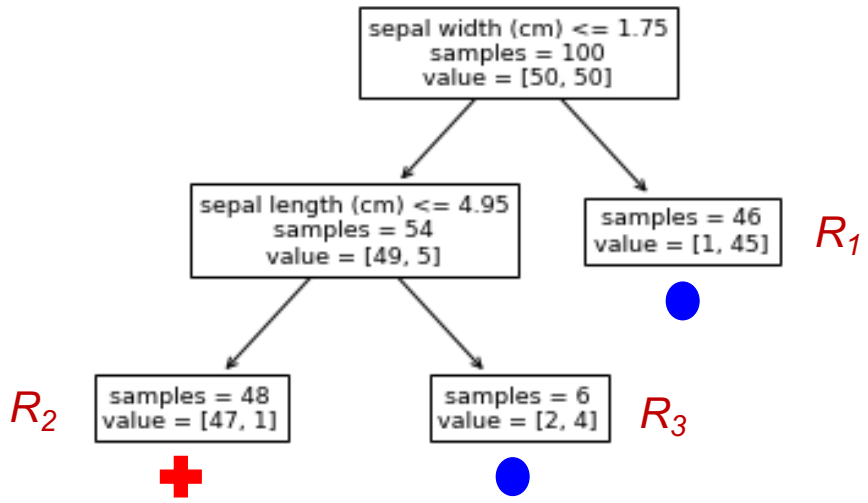
Lou, Y., Caruana, R., Gehrke, J. & Hooker, G. "Accurate intelligible models with pairwise interactions," in proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining KDD '13, 623–631, 2012.

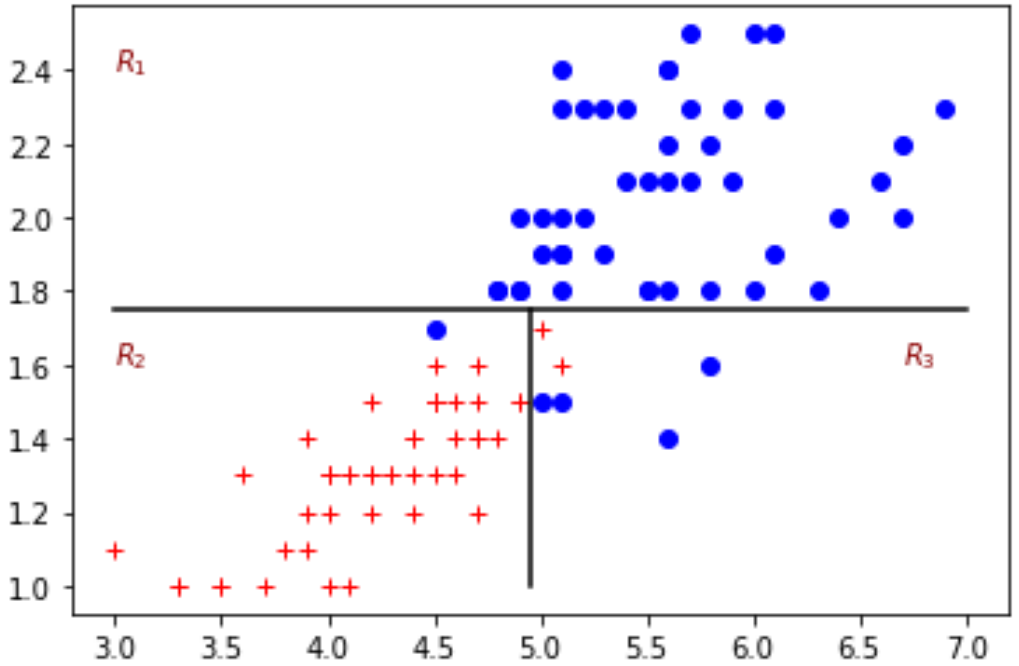
Big Picture

- Linear Models
- Generalized Additive Models
- Explainable Boosting Machines
- Decision Trees
- Rule-based Approaches



Example Tree





Regression Trees

$$X_1, X_2, \ldots, X_p \xrightarrow{\text{distinct and nonoverlapping regions}} R_1, R_2, \ldots, R_J$$

Prediction: Mean of the response values for the training observations in R_j

$$R_1, R_2, \ldots, R_J \quad ?$$

Goal: Finding the regions such that RSS is minimized

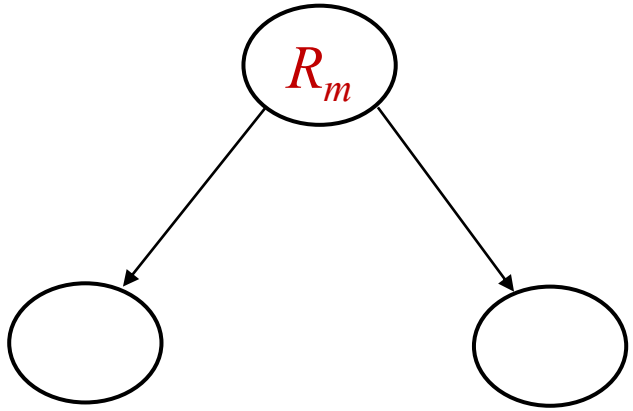
$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

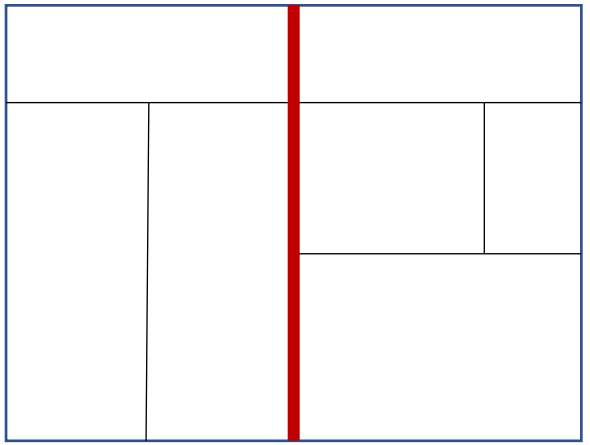
$\hat{y}_{R_j}$: mean response within R_j

Regression Trees

Recursive Binary Splitting

$$E(R_m) = \sum_{i:x_i \in R_m} (y_i - \hat{y}_{R_m})^2$$





n_{R_1} : number of samples in region $R_1(j, s)$

n_{R_2} : number of samples in region $R_2(j, s)$

$n_R = n_{R_1} + n_{R_2}$

$$R_1(j,s) = \{X | X_j < s\} \qquad R_2(j,s) = \{X | X_j \geq s\}$$

Find j and s that minimizes the weighted error

$$\frac{n_{R_1}}{n_R} E(R_1(j,s)) + \frac{n_{R_2}}{n_R} E(R_2(j,s))$$

sklearn.tree.DecisionTreeRegressor

Parameters: criterion : {"squared_error", "friedman_mse", "absolute_error", "poisson"}, default="squared_error"

Regression Trees

Tree Pruning

Goal: Avoiding overfitting with a fully grown or large tree
(Selecting a subtree that leads to a lowest test error rate)

$$\sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

$T \subset T_0$: subtree

$|T|$: number of terminal nodes in T

R_m : region corresponding to the mth terminal node

α : fixed parameter

$\alpha \uparrow$ $\quad$ $|T| \downarrow$

Use k-fold cross validation to choose α

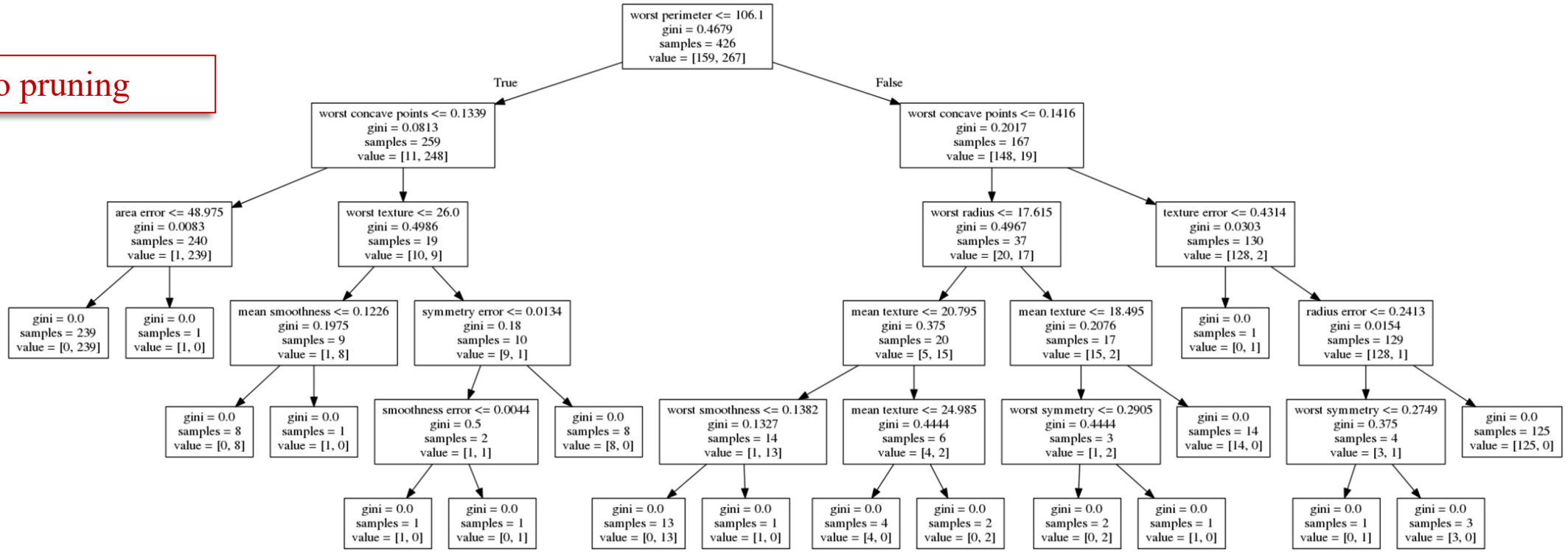
Regression Trees

Tree Pruning

sklearn.tree.DecisionTreeRegressor

class `sklearn.tree.DecisionTreeRegressor`(*, *criterion='squared_error', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, ccp_alpha=0.0*) [source]

no pruning



(link)

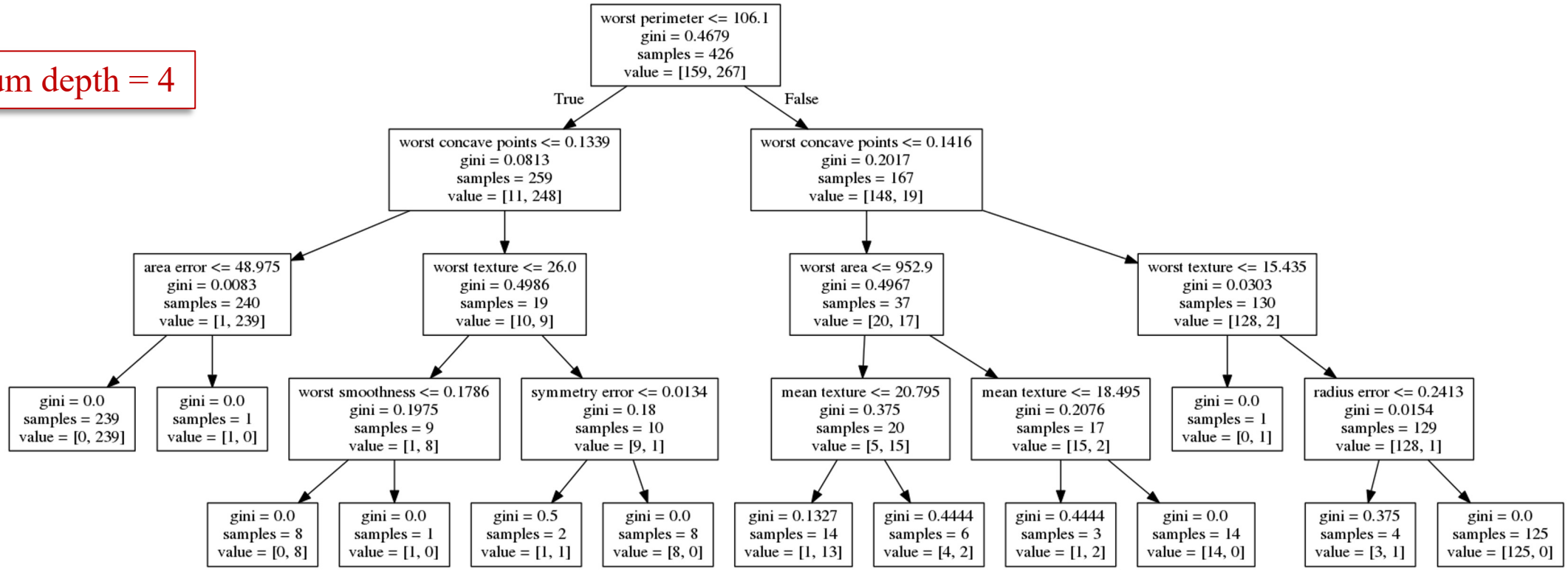
Regression Trees

Tree Pruning

sklearn.tree.DecisionTreeRegressor

class `sklearn.tree.DecisionTreeRegressor`(*, *criterion='squared_error'*, *splitter='best'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features=None*, *random_state=None*, *max_leaf_nodes=None*, *min_impurity_decrease=0.0*, *ccp_alpha=0.0*) [source]

maximum depth = 4



(link)

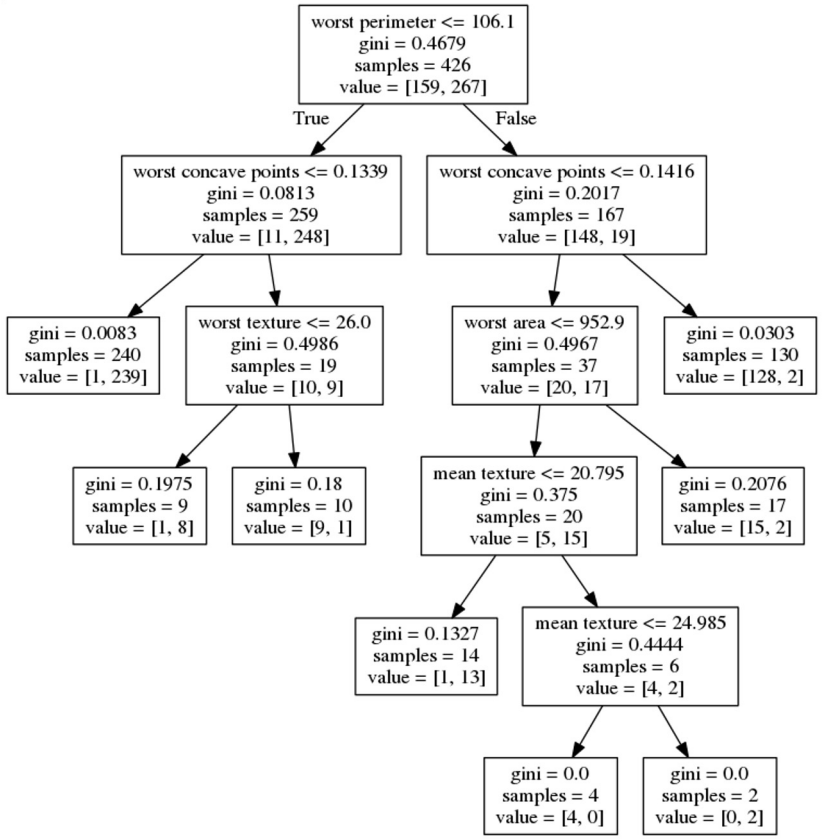
Regression Trees

Tree Pruning

sklearn.tree.DecisionTreeRegressor

class `sklearn.tree.DecisionTreeRegressor`(*, *criterion='squared_error'*, *splitter='best'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features=None*, *random_state=None*, *max_leaf_nodes=None*, *min_impurity_decrease=0.0*, *ccp_alpha=0.0*) [source]

maximum leaf nodes = 8



(link)

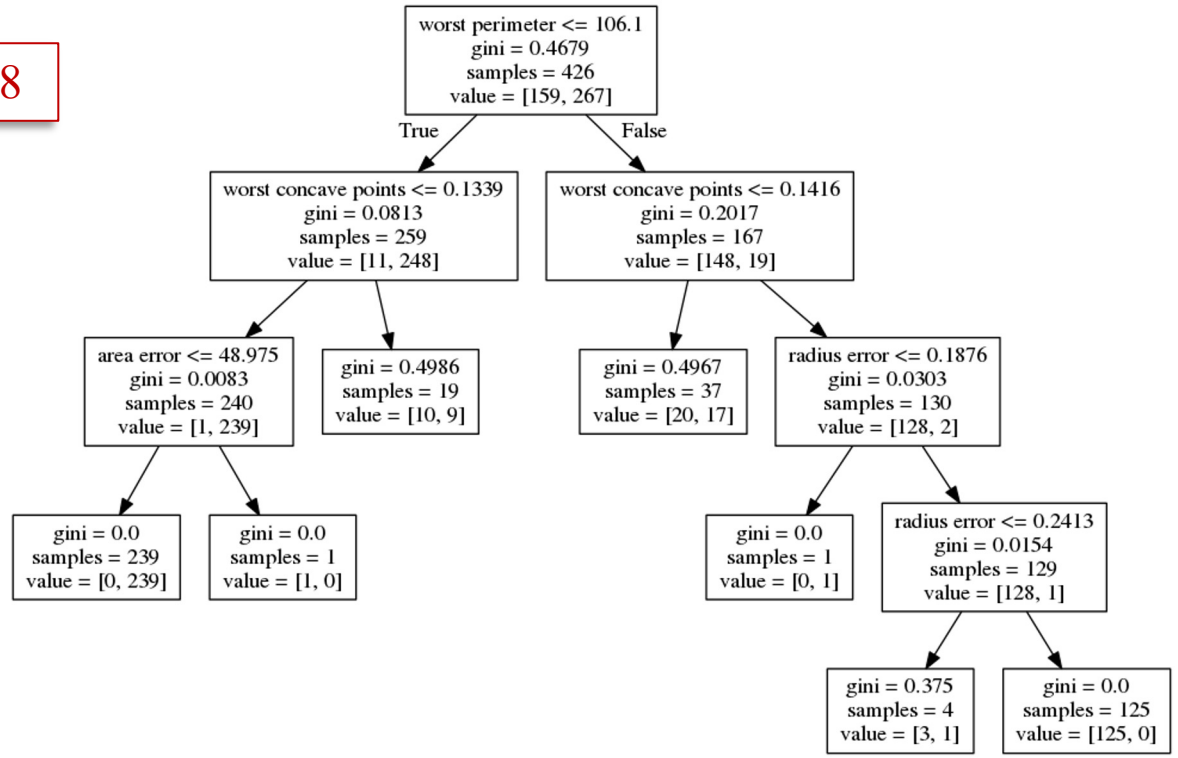
Regression Trees

Tree Pruning

sklearn.tree.DecisionTreeRegressor

class `sklearn.tree.DecisionTreeRegressor`(*, *criterion='squared_error'*, *splitter='best'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features=None*, *random_state=None*, *max_leaf_nodes=None*, *min_impurity_decrease=0.0*, *ccp_alpha=0.0*) [source]

minimum samples for split = 8



(link)

Classification Trees

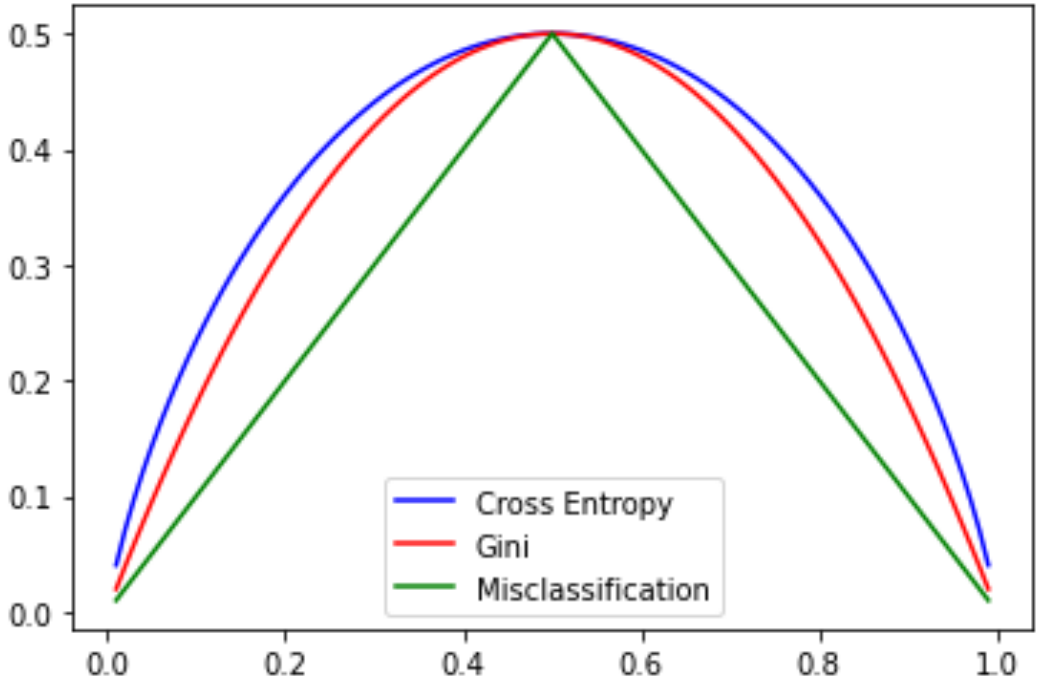
Similar to a regression tree but uses different error measures based on *purity* of a region

$\hat{p}_{mk}$: proportion of class-*k* training observations in the *m*th region

Misclassification error: $1 - \max_k \{\hat{p}_{mk}\}$

Cross entropy: $-\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$

Gini index: $\sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$



Classification Trees

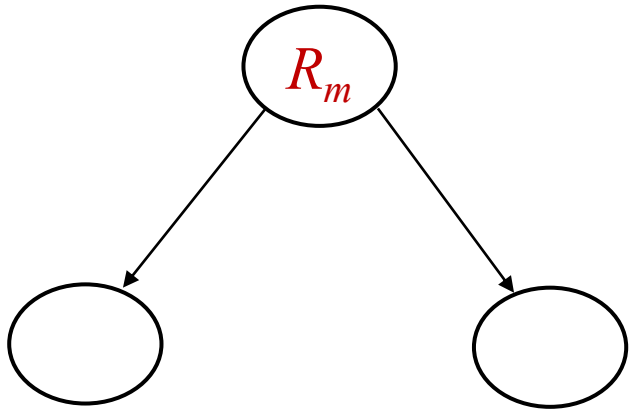
Recursive Binary Splitting

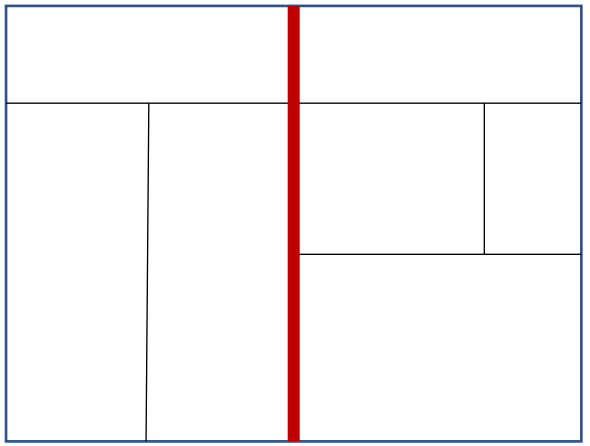
$$E(R_m) = 1 - \max_k \{\hat{p}_{mk}\}$$

n_{R_1} : number of samples in region $R_1(j, s)$

n_{R_2} : number of samples in region $R_2(j, s)$

$n_R = n_{R_1} + n_{R_2}$





$$R_1(j, s) = \{X | X_j < s\} \qquad R_2(j, s) = \{X | X_j \geq s\}$$

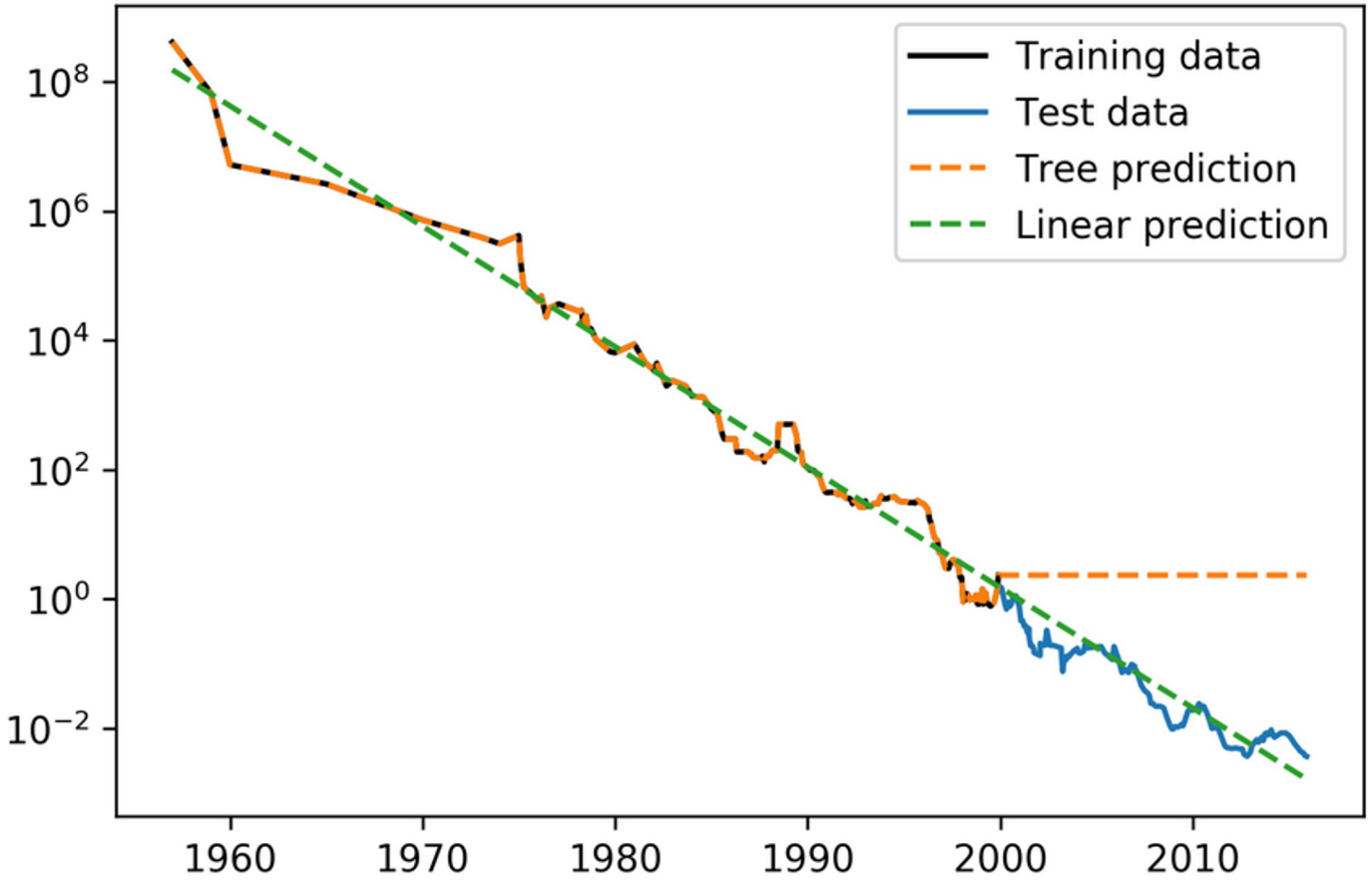
Find j and s that minimizes the weighted error

$$\frac{n_{R_1}}{n_R} E(R_1(j, s)) + \frac{n_{R_2}}{n_R} E(R_2(j, s))$$

sklearn.tree.DecisionTreeClassifier

Parameters: criterion : {"gini", "entropy", "log_loss"}, default="gini"

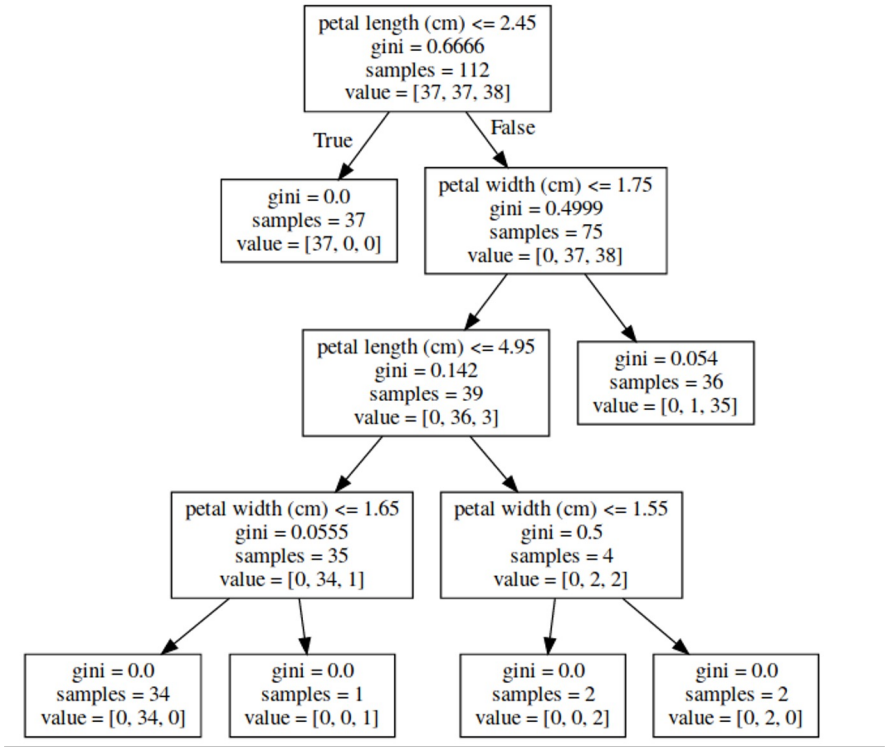
Lack of Extrapolation



(link)

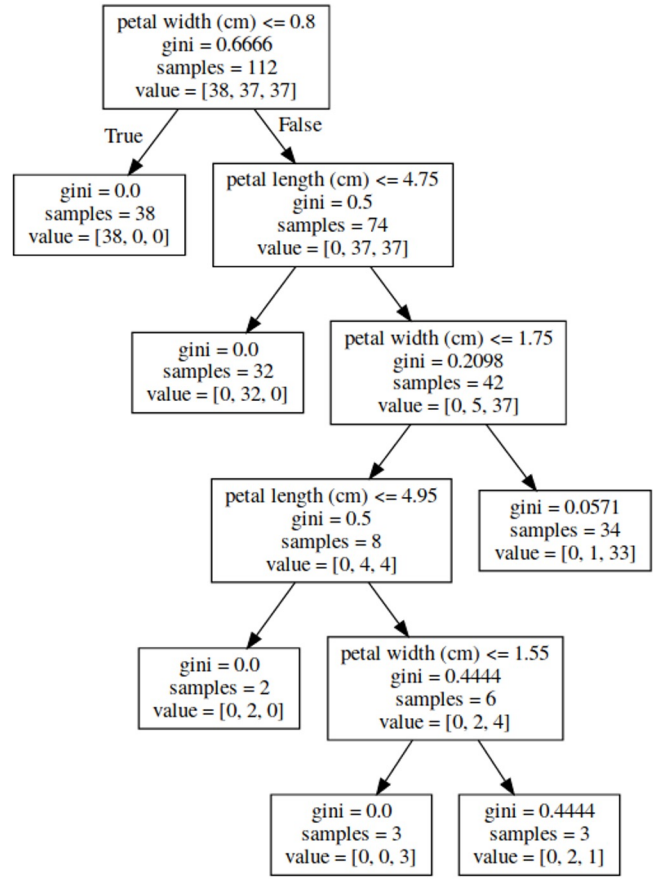
Instability

```
iris = load_iris()
X_train, X_test, y_train, y_test = train_test_split(
    iris.data, iris.target, stratify=iris.target, random_state=0)
tree = DecisionTreeClassifier(max_leaf_nodes=6).fit(X_train, y_train)
tree_dot = export_graphviz(tree, out_file=None, feature_names=iris.feature_names)
graphviz.Source(tree_dot)
```



(link)

```
X_train, X_test, y_train, y_test = train_test_split(
    iris.data, iris.target, stratify=iris.target, random_state=1)
tree = DecisionTreeClassifier(max_leaf_nodes=6).fit(X_train, y_train)
tree_dot = export_graphviz(tree, out_file=None, feature_names=iris.feature_names)
graphviz.Source(tree_dot)
```



(link)

Computational Complexity

Volume 5, number 1 INFORMATION PROCESSING LETTERS May 1976

CONSTRUCTING OPTIMAL BINARY DECISION TREES IS NP-COMPLETE*

Laurent HYAFIL
IRIA – Laboria, 78150 Rocquencourt, France
and
Ronald L. RIVEST
Dept. of Electrical Engineering and Computer Science, M.I.T., Cambridge, Massachusetts 02139, USA

Received 7 November 1975, revised version received 26 January 1976

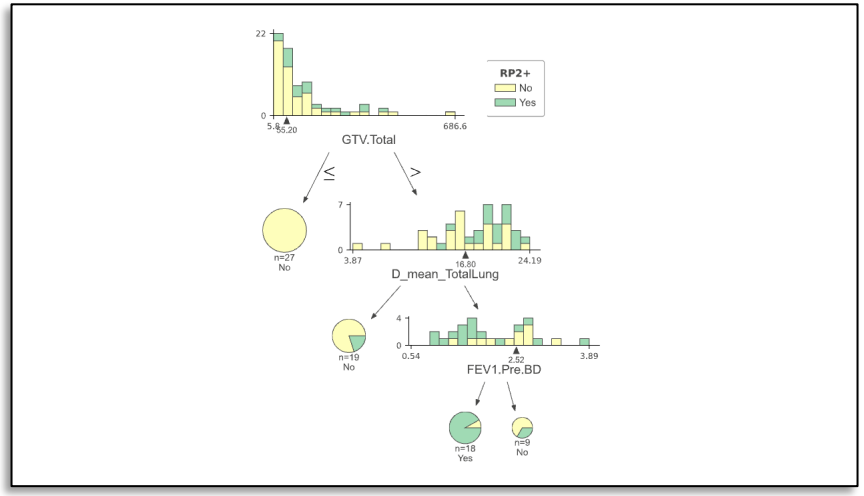
Binary decision trees, computational complexity, NP-complete

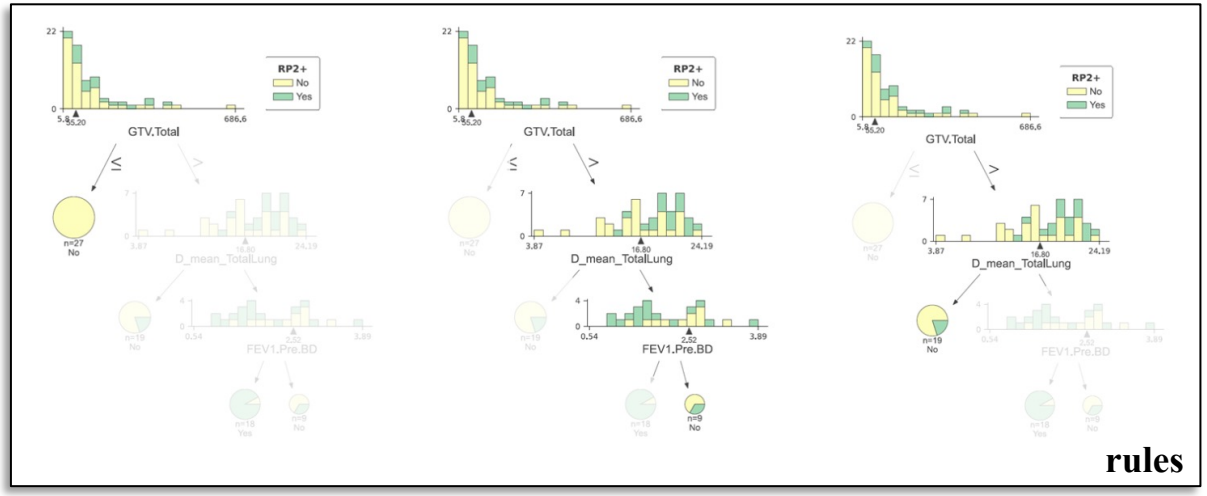
We demonstrate that constructing optimal binary decision trees is an NP-complete problem, where an optimal tree is one which minimizes the expected number of tests required to identify the unknown object.

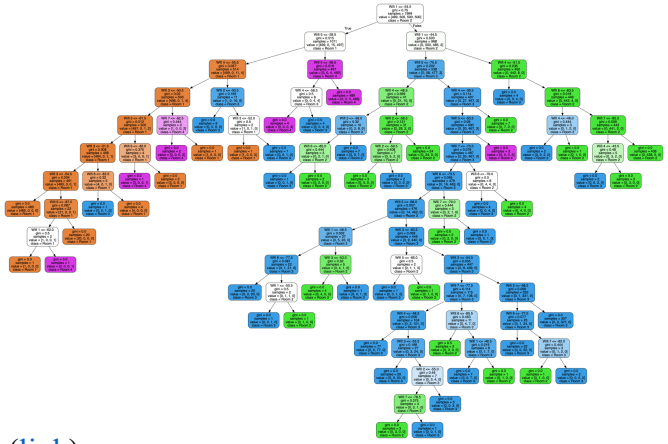
([link])

Optimization for Tree Growing

$$\mathcal{D} = \{(\boldsymbol{x}_i, y_i) : i \in \mathcal{I}\}$$
$$\mathcal{I} = \{1, \ldots, n\}, \text{ sample indices}$$







(link)

$$\min_{T \in \mathcal{T}} \ell_{\mathcal{D}}(T) + \lambda \Omega(T)$$

total loss

model complexity

all possible trees

(Binary) Optimal Classification Trees: (Bin)OCT
Optimal Sparse Decision Trees: OSDT → FSDT
…

Optimization for Tree Growing

$$\min_{T \in \mathcal{T}} \ell_{\mathcal{D}}(T) + \lambda \Omega(T)$$

- $\ell_{\mathcal{D}}(T)$: total loss
- $\lambda \Omega(T)$: model complexity
- $\mathcal{T}$: all possible trees

$$\text{minimize} \quad R_{xy}(T) + \alpha |T|$$

$$\text{subject to} \quad N_x(l) \geq N_{\min}, \quad l \in \text{leaves of } T$$

- $R_{xy}(T)$: classification error
- $|T|$: number of branches in T
- T: tree
- $N_x(l)$: number of points in leaf l
- $N_{\min}$: minimum number of samples in any leaf

Optimal Classification Trees (OCT)[+]

For a tree $d = (d_{un}, \delta_{un}, d_{\text{split}}, \delta_{\text{split}}, K, H_d)$, we define its objective function as a combination of the misclassification error and a sparsity penalty on the number of leaves:

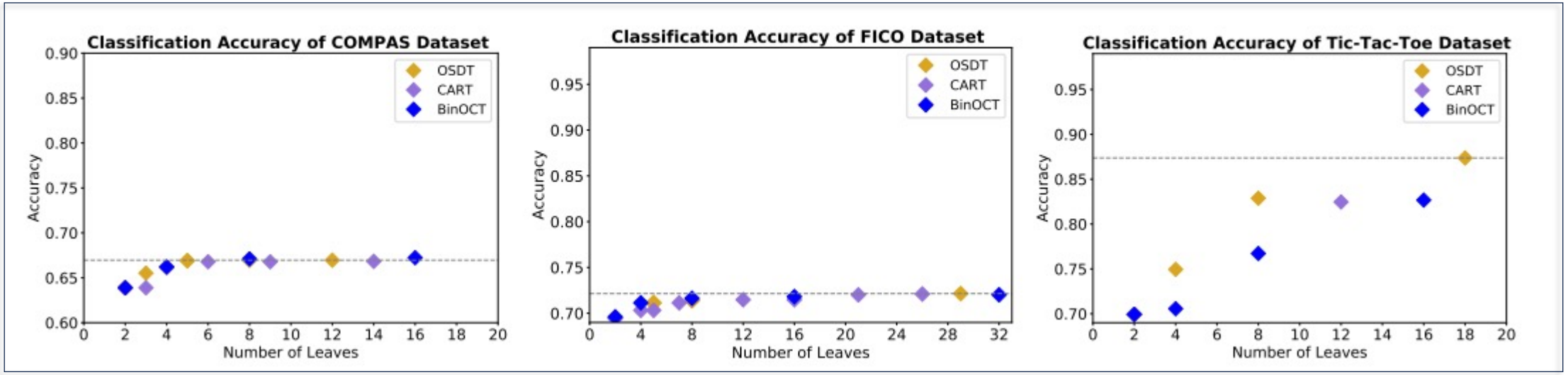
$$R(d, \mathbf{x}, \mathbf{y}) = \ell(d, \mathbf{x}, \mathbf{y}) + \lambda H_d. \tag{4}$$

$R(d, \mathbf{x}, \mathbf{y})$ is a regularized empirical risk. The loss $\ell(d, \mathbf{x}, \mathbf{y})$ is the misclassification error of d, *i.e.*, the fraction of training data with incorrectly predicted labels. H_d is the number of leaves in the tree d. λH_d is a regularization term that penalizes bigger trees.

Optimal Sparse Decision Trees (OSDT)[✦]

[+] Bertsimas, D. & Dunn, J. "Optimal classification trees," Machine Learning, 106(7), 1039-1083, 2017.
[✦] Hu, X., Rudin, C. & Seltzer, M. "Optimal sparse decision trees," in Advances in Neural Information Processing Systems, 7267–7275, 2019.

Optimization for Tree Growing



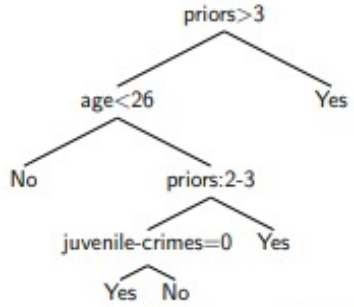
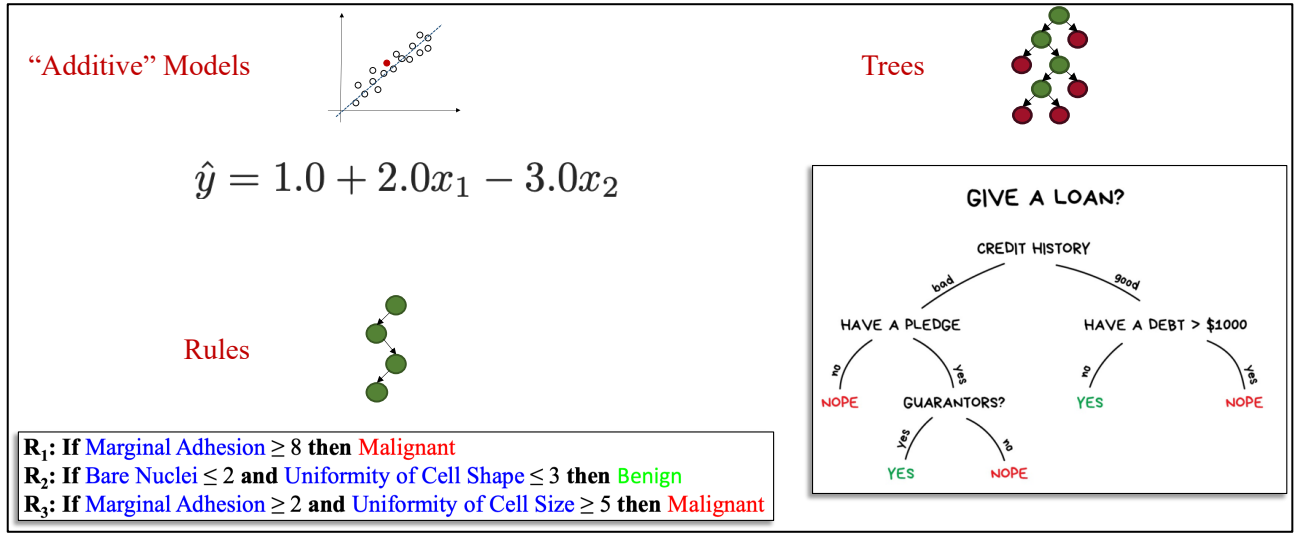


Figure 4: An optimal decision tree generated by OSDT on the COMPAS dataset. ($\lambda = 0.005$, accuracy: 66.90%)

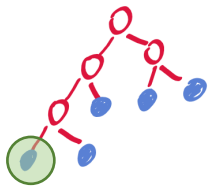
Hu, X., Rudin, C. & Seltzer, M. "Optimal sparse decision trees," in Advances in Neural Information Processing Systems, 7267–7275, 2019.

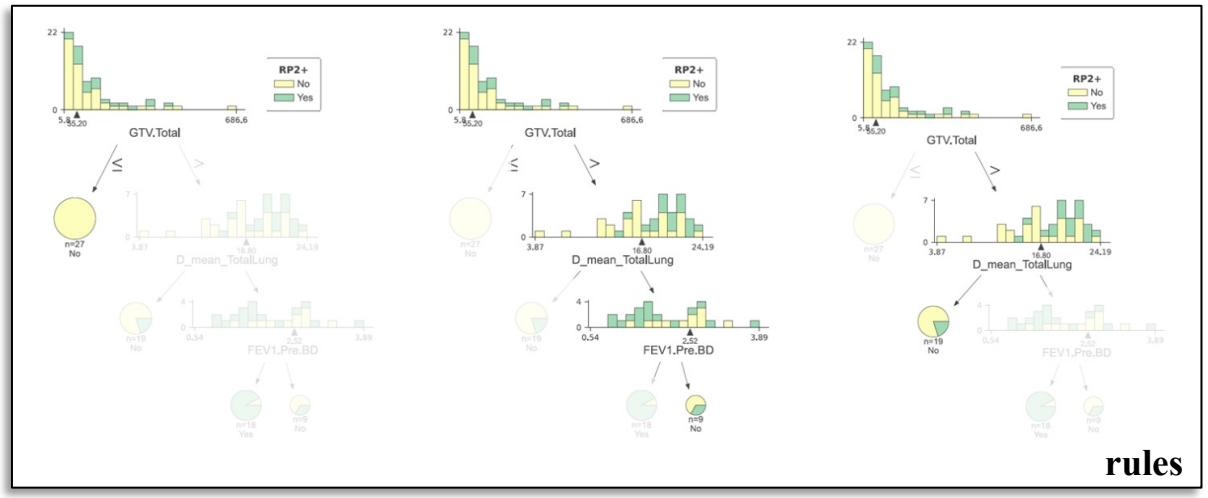
Big Picture

- Linear Models
- Generalized Additive Models
- Explainable Boosting Machines
- Decision Trees
- Rule-based Approaches

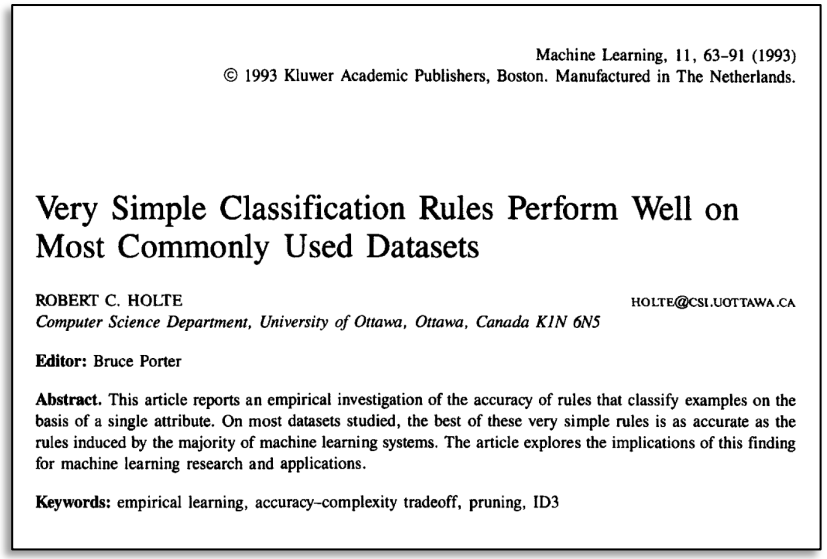


Decision Rules





R_1: If Marginal Adhesion ≥ 8 **then** Malignant
R_2: If Bare Nuclei ≤ 2 **and** Uniformity of Cell Shape ≤ 3 **then** Benign
R_3: If Marginal Adhesion ≥ 2 **and** Uniformity of Cell Size ≥ 5 **then** Malignant



Machine Learning, 11, 63–91 (1993)
© 1993 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

Very Simple Classification Rules Perform Well on Most Commonly Used Datasets

ROBERT C. HOLTE HOLTE@CSI.UOTTAWA.CA
Computer Science Department, University of Ottawa, Ottawa, Canada K1N 6N5

Editor: Bruce Porter

Abstract. This article reports an empirical investigation of the accuracy of rules that classify examples on the basis of a single attribute. On most datasets studied, the best of these very simple rules is as accurate as the rules induced by the majority of machine learning systems. The article explores the implications of this finding for machine learning research and applications.

Keywords: empirical learning, accuracy–complexity tradeoff, pruning, ID3

(link)

Optimization

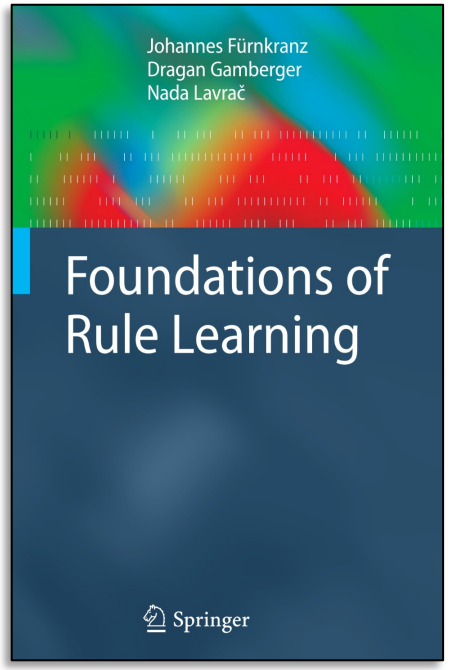
total loss

model complexity

all possible rules

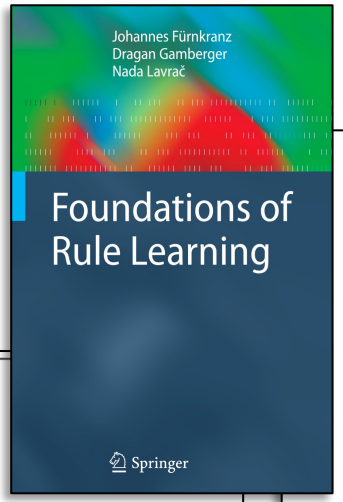
$$\min_{R \in \mathcal{R}} \ \ell_{\mathcal{D}}(R) + \lambda \Omega(R)$$

Boolean Decision Rules (BDR)
Rule Generation (RUG)
…



(link)

Rule-based Models



function LEARNRULEBASE($\mathcal{E}$)

Input:
$\mathcal{E}$ set of training examples

Algorithm:
$\mathcal{R} := \emptyset$
for each class c_i, $i = 1$ **to** C **do**
$\mathcal{P}_i := \{$subset of examples in $\mathcal{E}$ with class label $c_i\}$
$\mathcal{N}_i := \{$subset of examples in $\mathcal{E}$ with other class labels$\}$
$\mathcal{R}_i :=$ LEARNSETOFRULES(c_i,$\mathcal{P}_i$,$\mathcal{N}_i$)
$\mathcal{R} := \mathcal{R} \cup \mathcal{R}_i$
endfor
$\mathcal{R} := \mathcal{R} \cup \{$default rule $(c_{max} \leftarrow true)\}$
where c_{max} is the majority class in $\mathcal{E}$.

Output:
$\mathcal{R}$ the learned rule set

Fig. 2.10 Constructing a set of rules in a multiclass learning setting

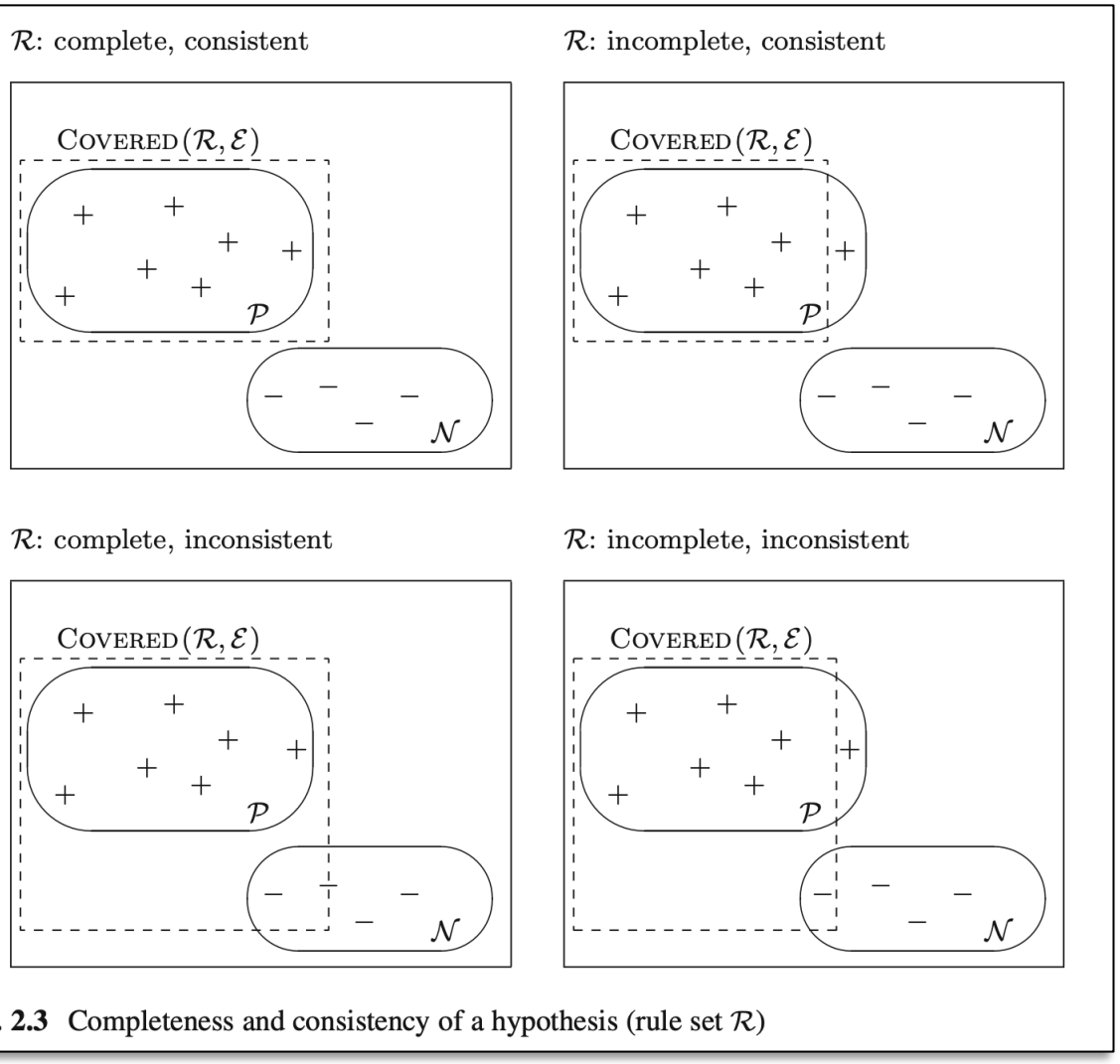


Fig. 2.3 Completeness and consistency of a hypothesis (rule set $\mathcal{R}$)

Rule Generation

$$\mathcal{D} = \{(\boldsymbol{x}_i, y_i) : i \in \mathcal{I}\}$$

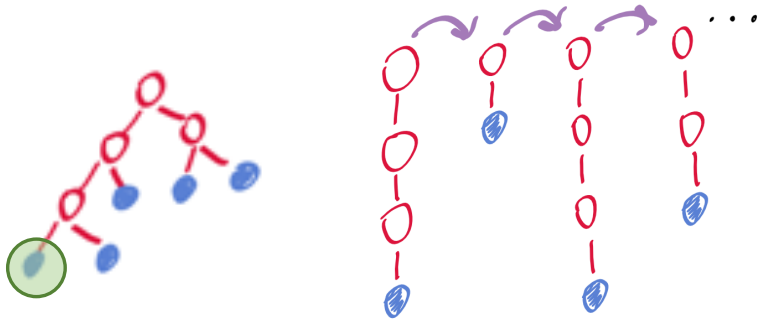
$$\mathcal{I} = \{1, \ldots, n\}, \text{ sample indices}$$

$$\boldsymbol{x}_i \in \mathbb{R}^p, y_i \in \{1, \ldots, K\}$$

multi-class

$$\min_{R \in \mathcal{R}} \ \ell_{\mathcal{D}}(R) + \lambda \Omega(R)$$

- total loss
- model complexity
- all possible rules



$$\min_{w_R \geq 0, R \in \mathcal{R}} \underbrace{\sum_{i \in \mathcal{I}} \ell \left(\sum_{R \in \mathcal{R}} \hat{y}_{iR} w_R, y_i \right)}_{\ell_{\mathcal{D}}(R)} + \underbrace{\lambda \sum_{R \in \mathcal{R}} c_R w_R}_{\lambda \Omega(R)}$$

- prediction of rule for a sample
- weight of the rule
- cost of a rule

(Bin)OCT, FSDT, BDR,…

Integer Programming

- do not scale well
- mostly for binary classification

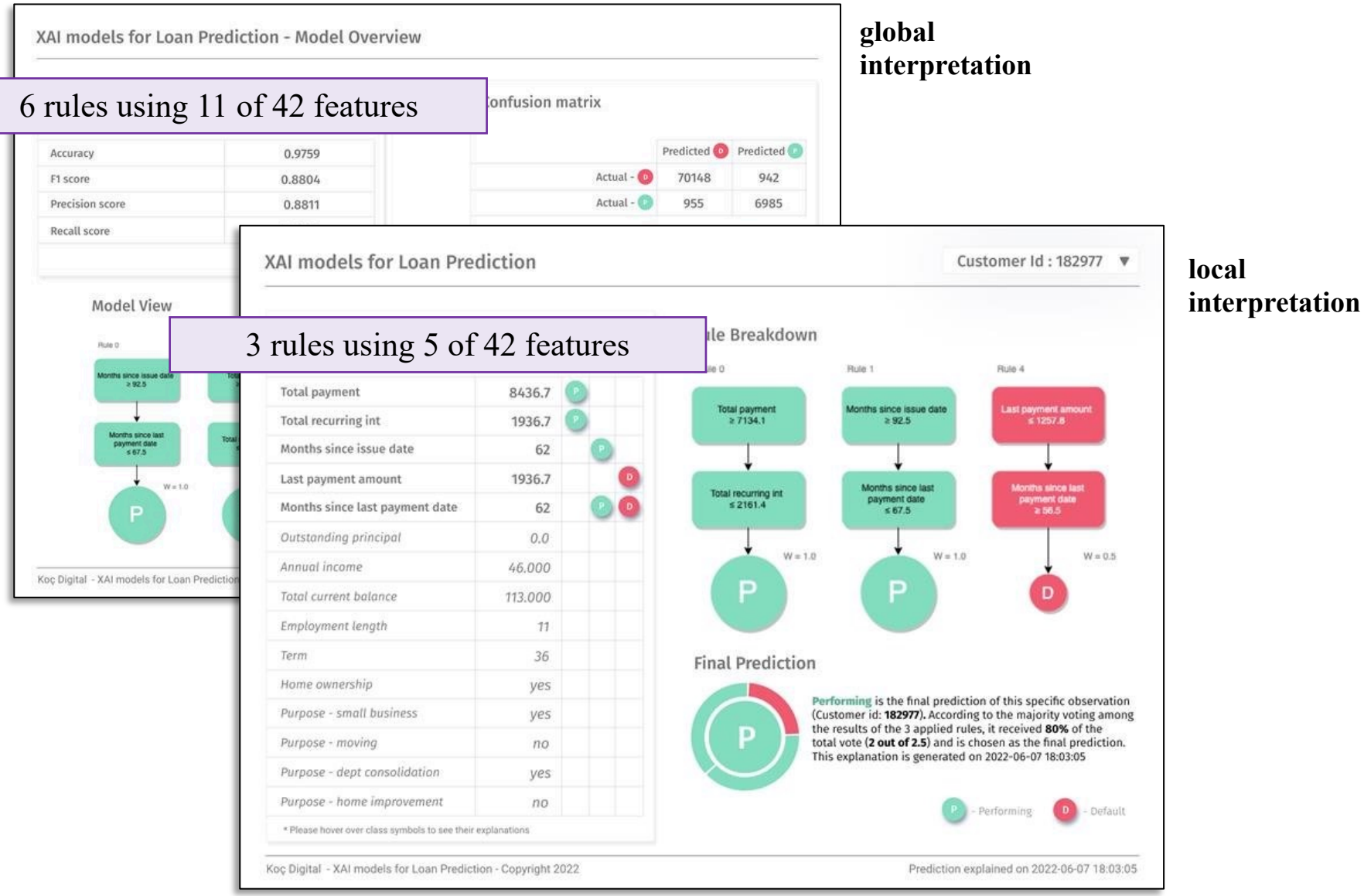
RUG

Linear Programming

Scalable Rule Generation for Learning

Interpretation

~450K samples

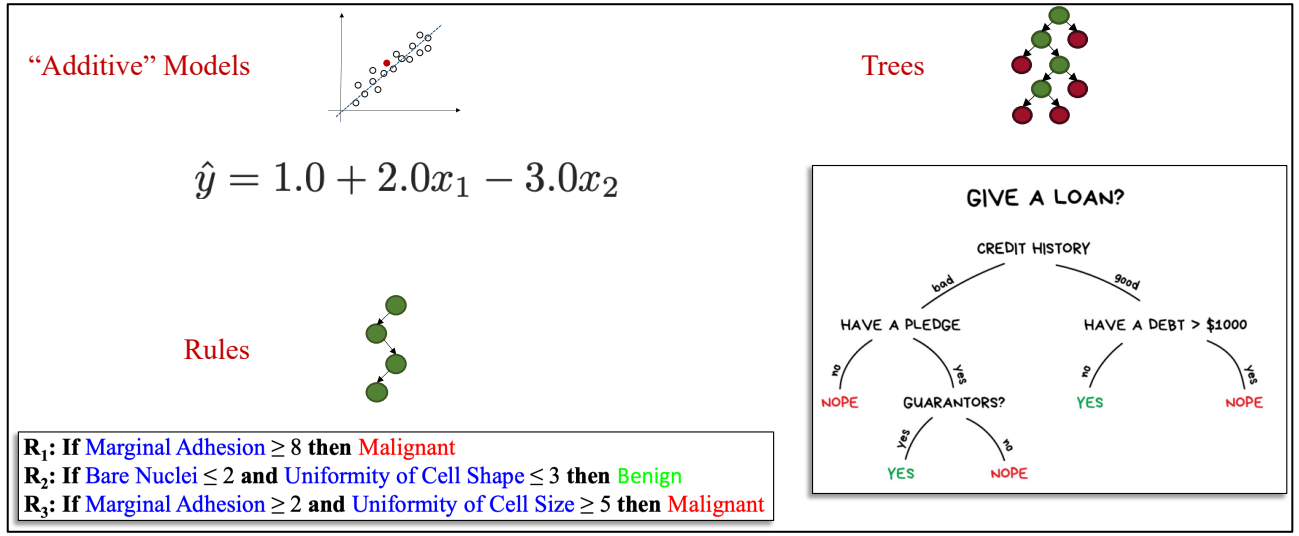


global interpretation

local interpretation

Big Picture

- Linear Models
- Generalized Additive Models
- Explainable Boosting Machines
- Decision Trees
- Rule-based Approaches



Reading Material