ML Fundamentals

Instituto Balseiro

Instituto Balseiro
06/09/2022

Inscripciones

- Cerraron 02/09
  - hablar con Nora Bayo en Oficina de Doctorados o escribir a doctorados@ib.edu.ar
- Entregamos la P1, que **debe entregarse antes del 13 de septiembre a las 11:59, sin excepción**.
  - pregunten si tienen dudas sobre el enunciado!

Lecture 6
Trees

Andy Wynn IG: @Gallery94art

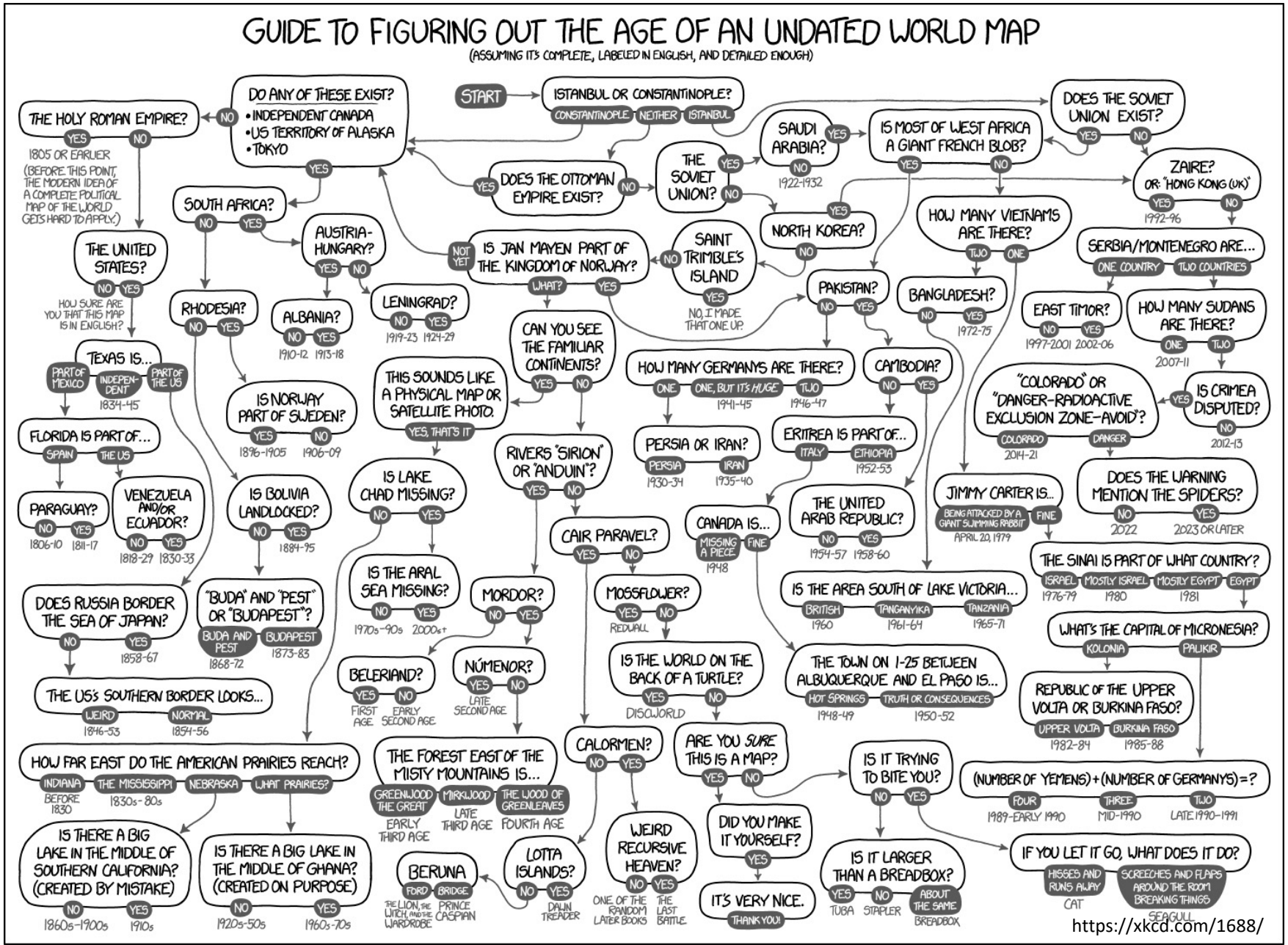
Luis G. Moyano — Fundamentos de ML
Instituto Balseiro

ML Fundamentals –Trees

- Decision trees
  - Regression
  - Classification: Gini coeff vs. Cross Entropy
  - Prunning
- Ensembles
- Random Forests
- Boosting and GBMs
- XGBoost
  - parameters & tips

if, then

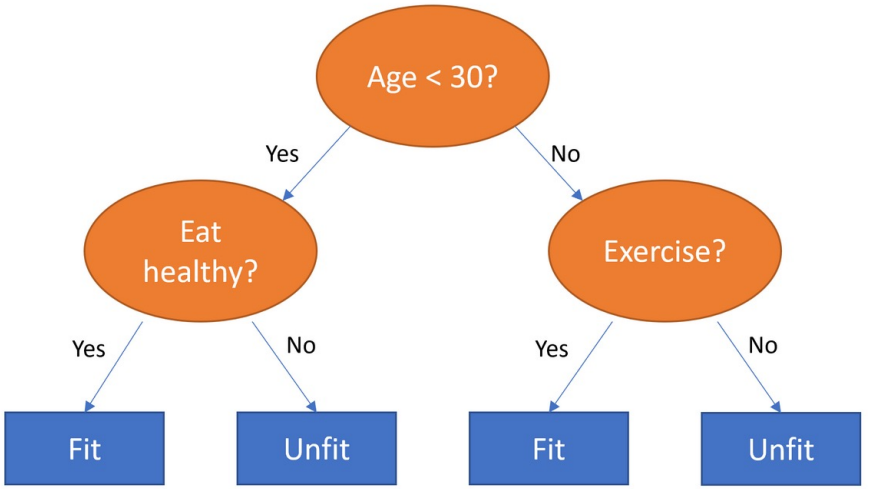
A decision tree

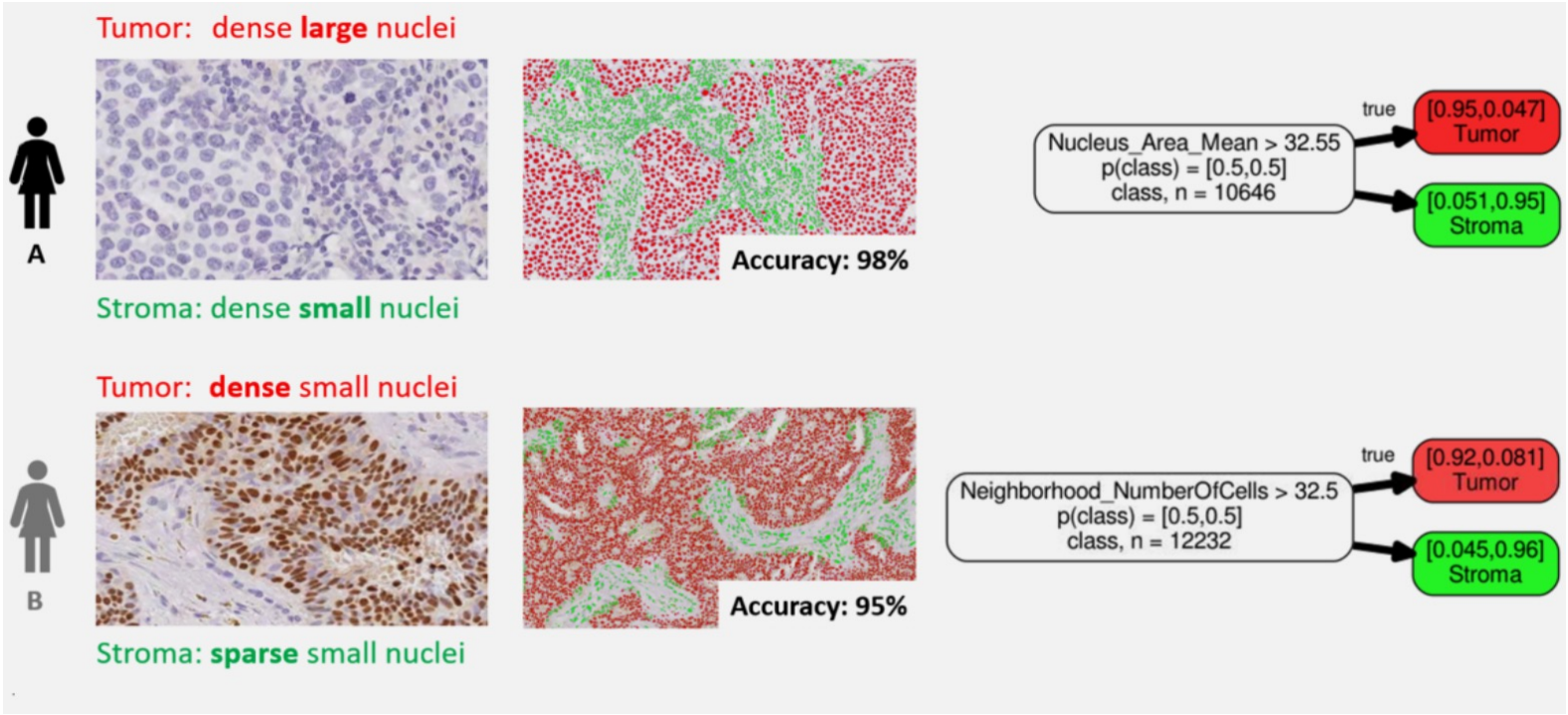


https://xkcd.com/1688/

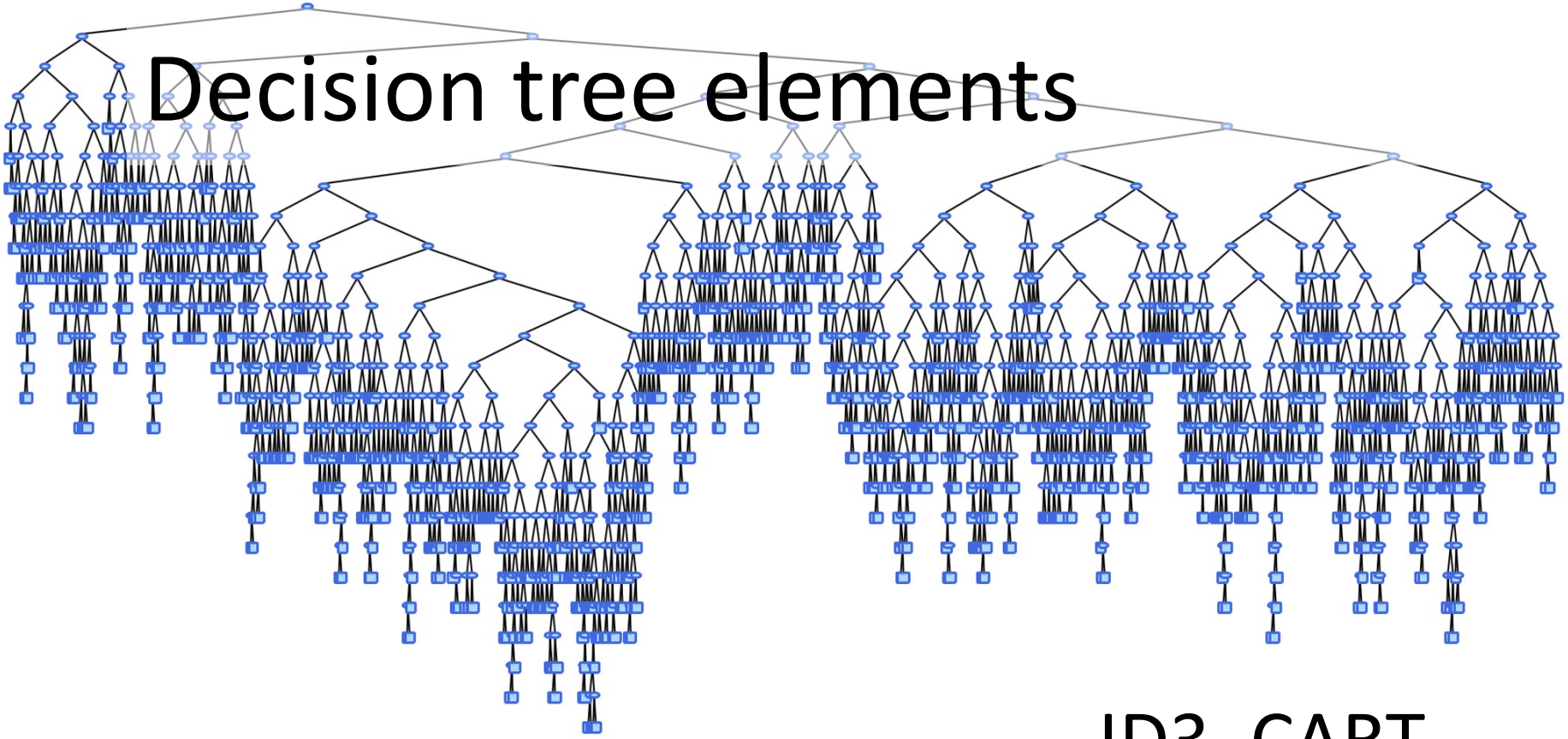
Why trees?

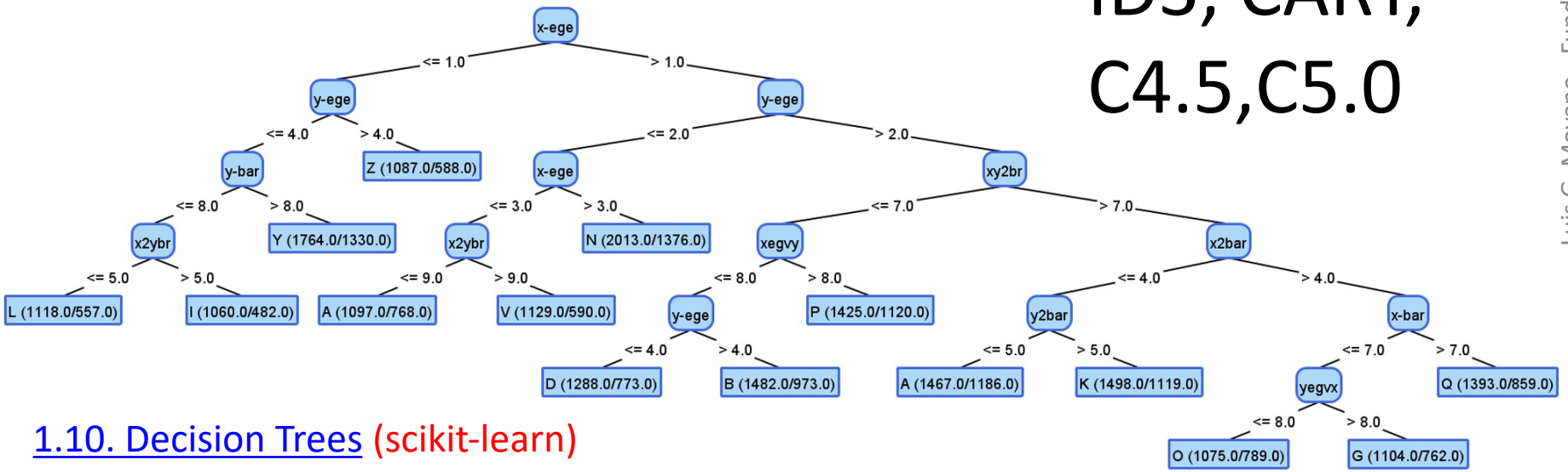
- What is a tree?
- Why bother with trees?
- What can we do with them?





Decision tree elements



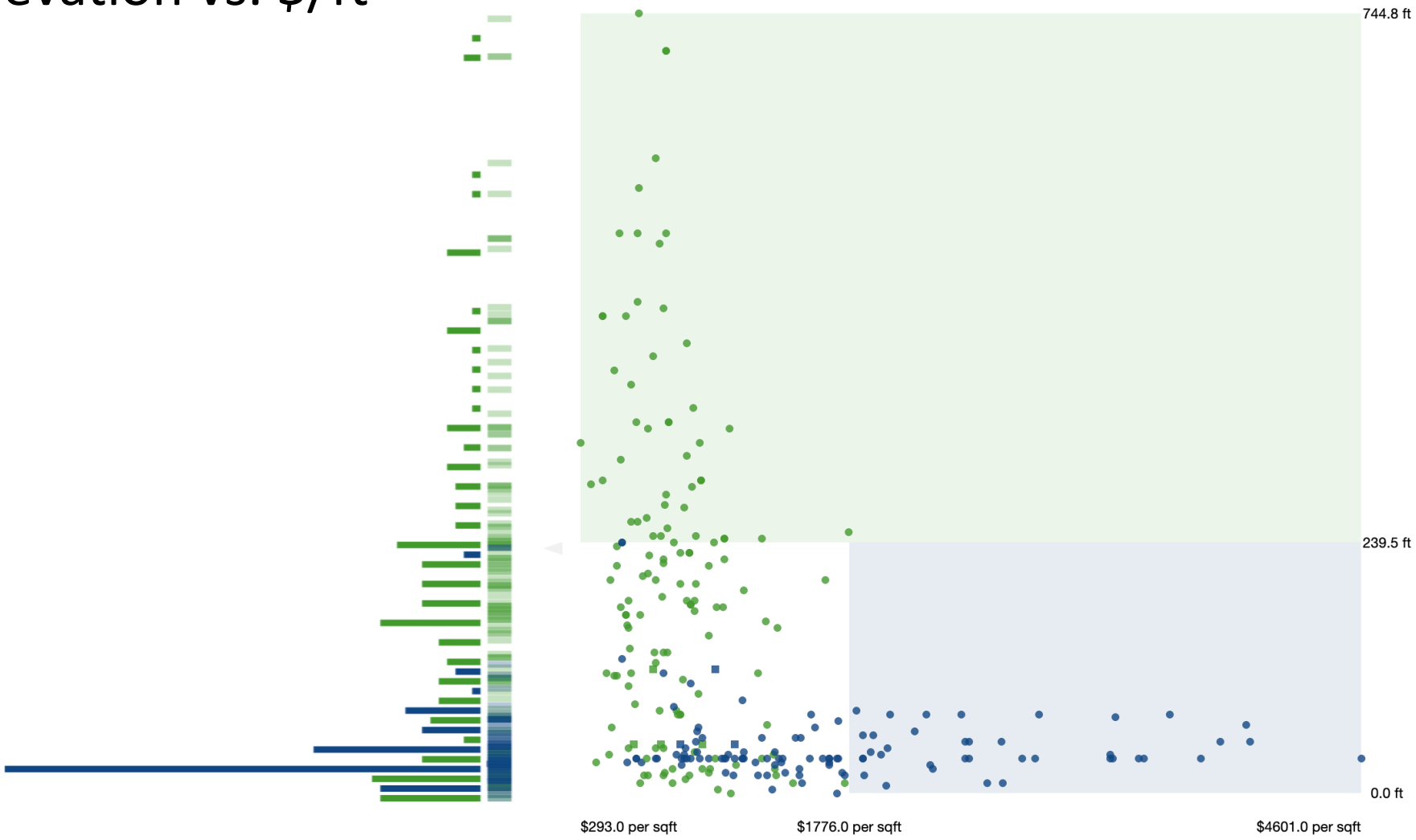


ID3, CART,
C4.5,C5.0

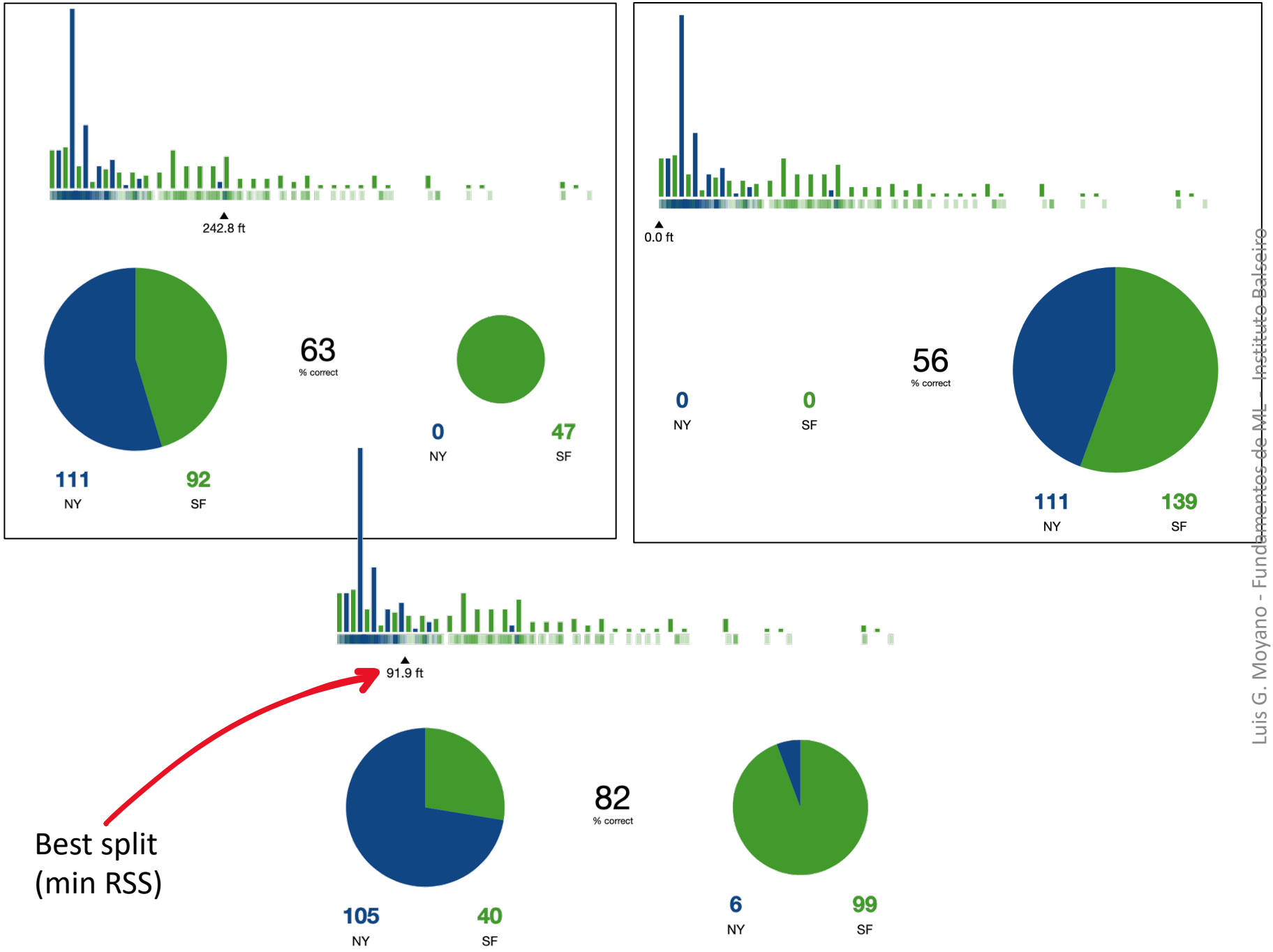
[1.10. Decision Trees](#) (scikit-learn)

NYC-SF Housing data

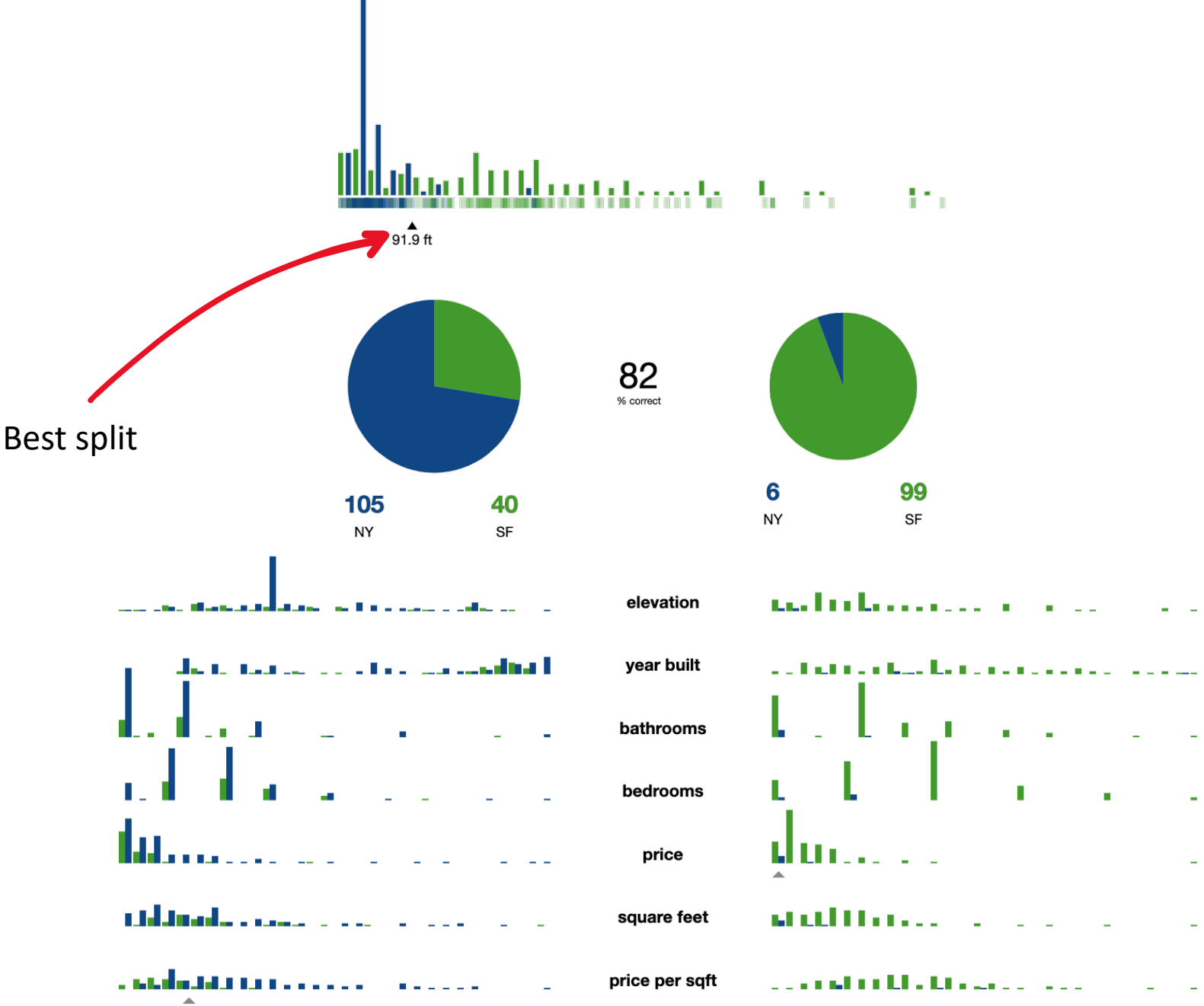
Elevation vs. $/ft^2



http://www.r2d3.us/visual-intro-to-machine-learning-part-1/



http://www.r2d3.us/visual-intro-to-machine-learning-part-1/

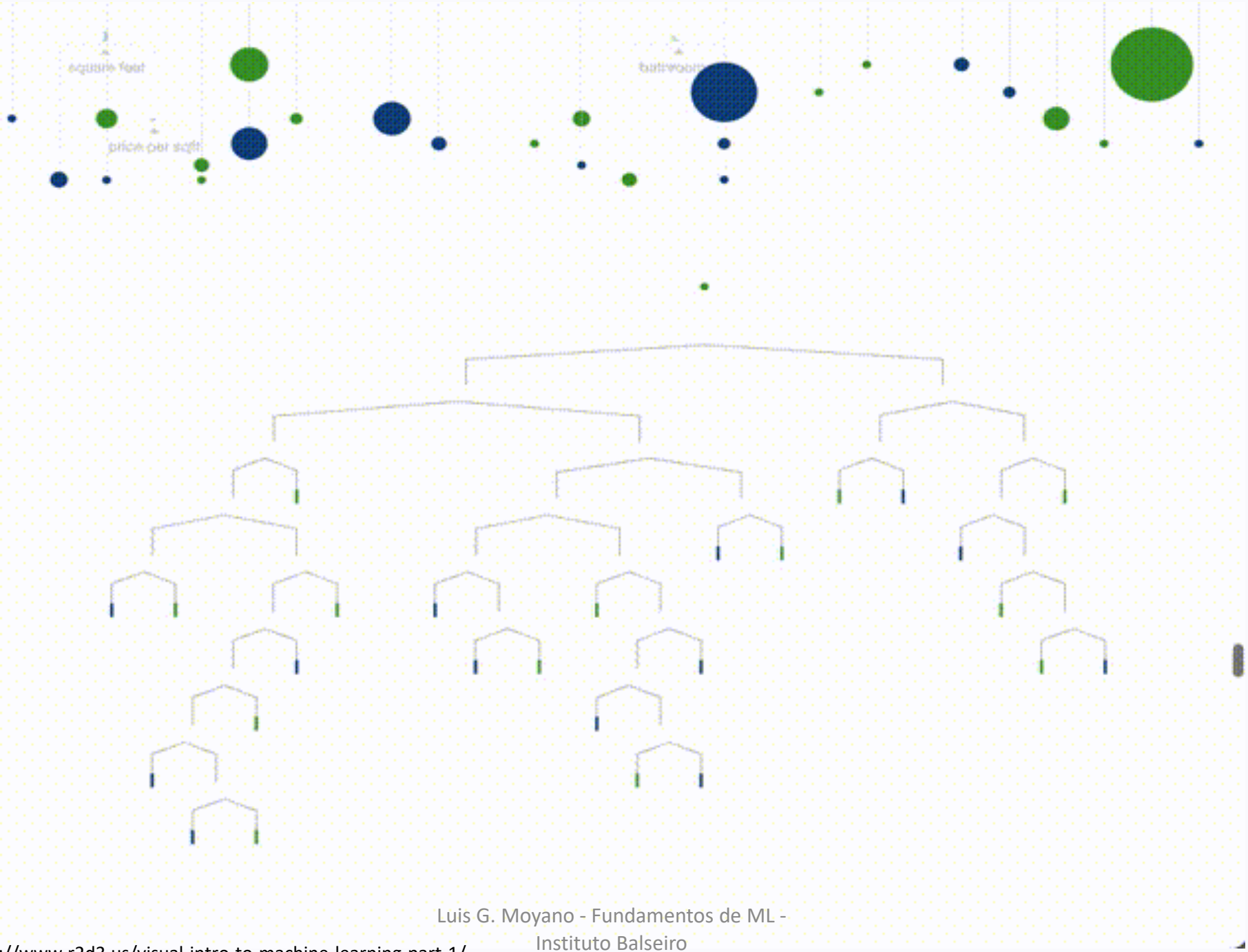


http://www.r2d3.us/visual-intro-to-machine-learning-part-1/

square foot is, at $1061 per sqft, is the best variable for the next if-then statement. For higher elevation homes, it is price, at $514,500

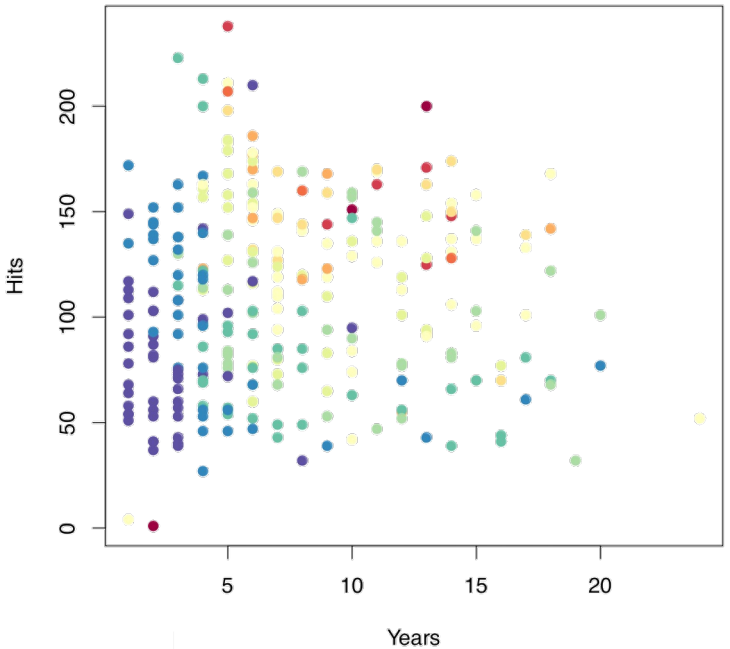
.

http://www.r2d3.us/visual-intro-to-machine-learning-part-1/



http://www.r2d3.us/visual-intro-to-machine-learning-part-1/

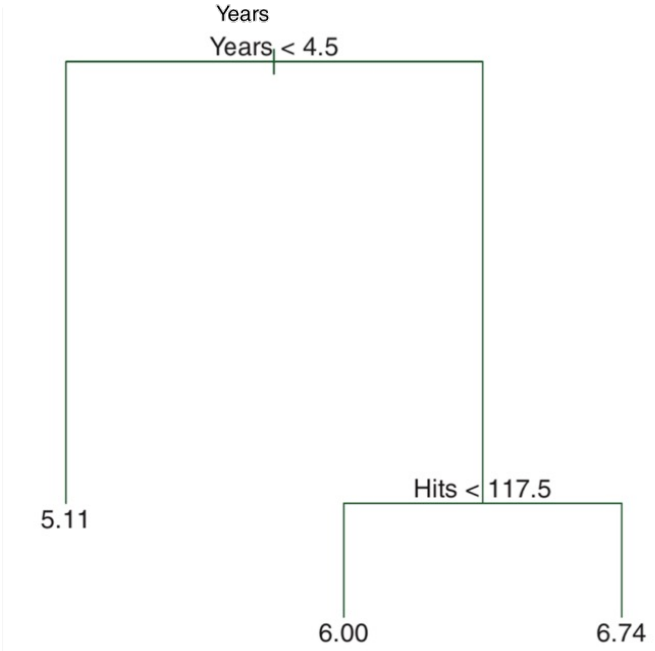
Building a regression tree

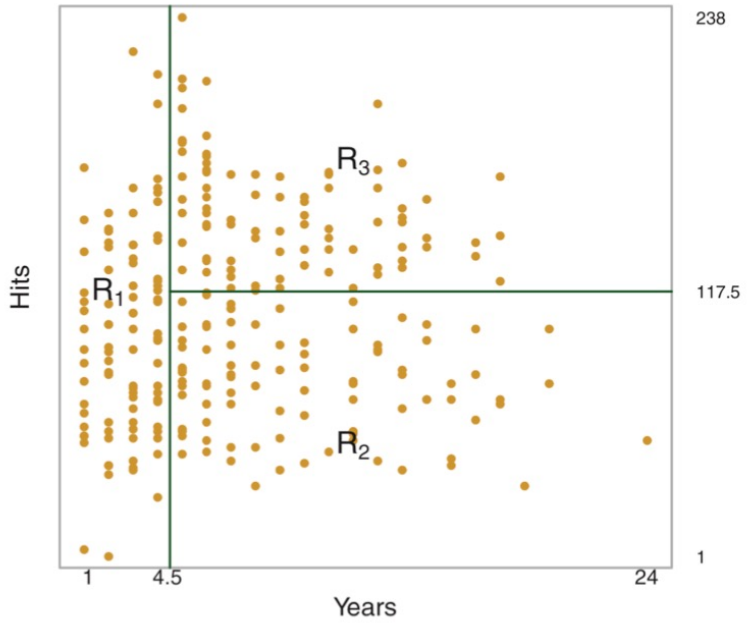


Baseball salary data
Salary is color-coded
(in Kelvins).

tree.DecisionTreeRegressor
tree.DecisionTreeClassifier

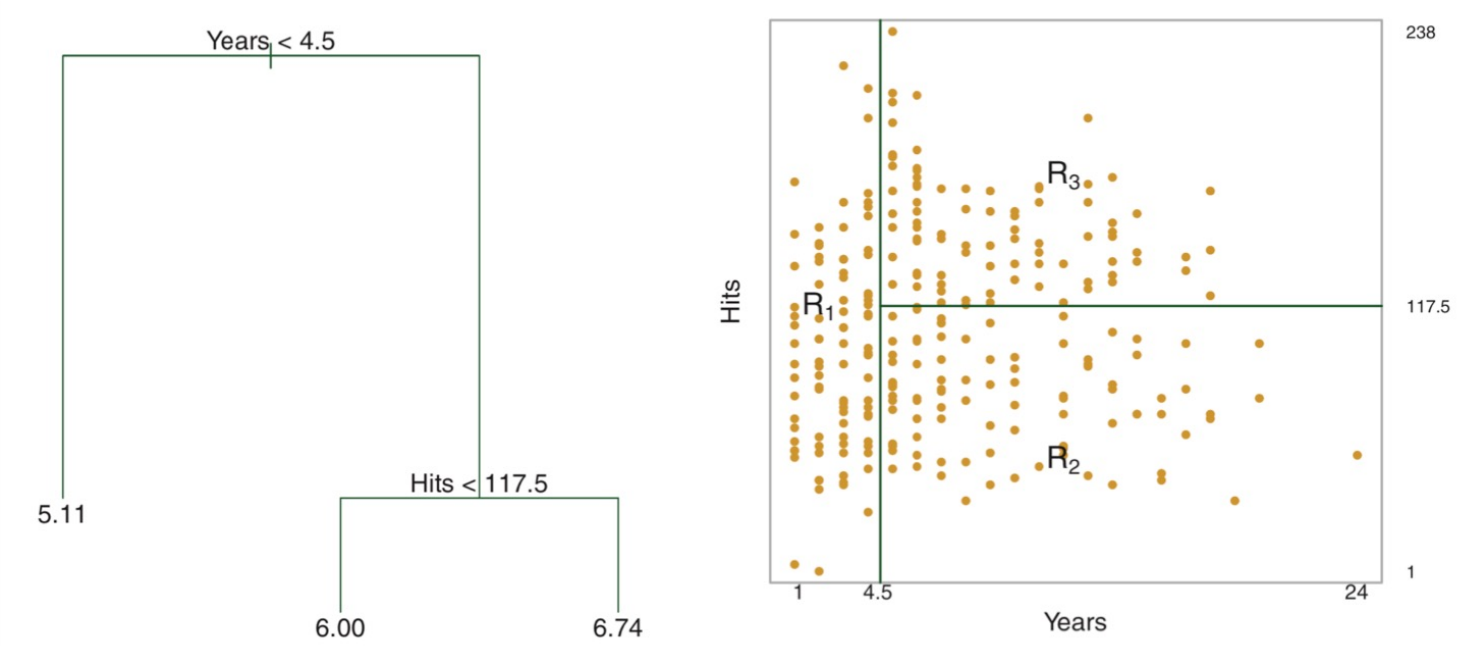
Overall, the tree stratifies or segments the players into three regions of predictor space: $R_1 = \{X \mid \texttt{Years} < 4.5\}$, $R_2 = \{X \mid \texttt{Years} >= 4.5, \texttt{Hits} < 117.5\}$, and $R_3 = \{X \mid \texttt{Years} >= 4.5, \texttt{Hits} >= 117.5\}$.





Recursive binary splitting

1. Top-down: splits predictor space ***recursively***



1. Greedy: best split at each step (horizon effect)

$$\sum_{i:\, x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:\, x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2,$$

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

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

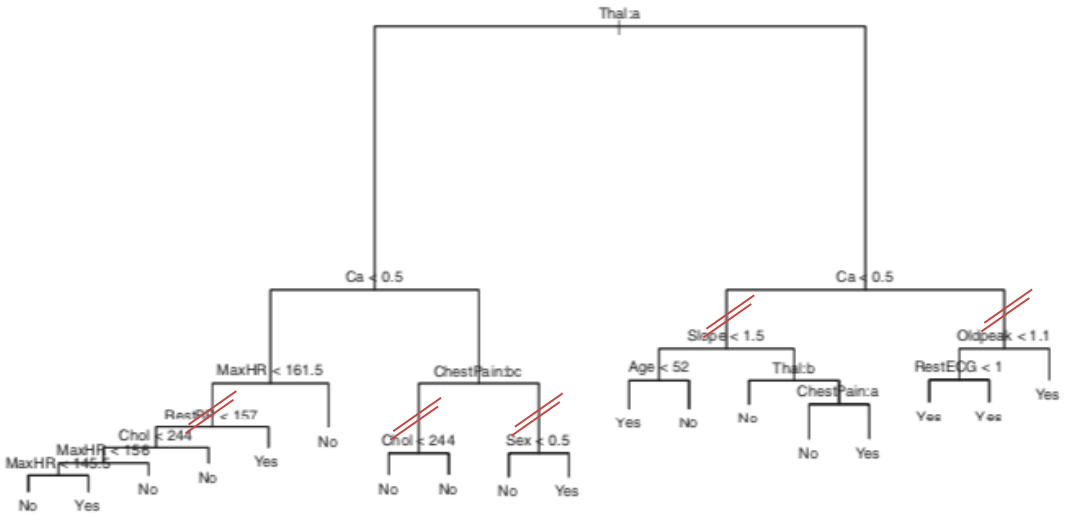
2. Stop by some criterion

TABLE 10.1. *Some characteristics of different learning methods. Key:* ▲ *= good,* ◆ *=fair, and* ▼ *=poor.*

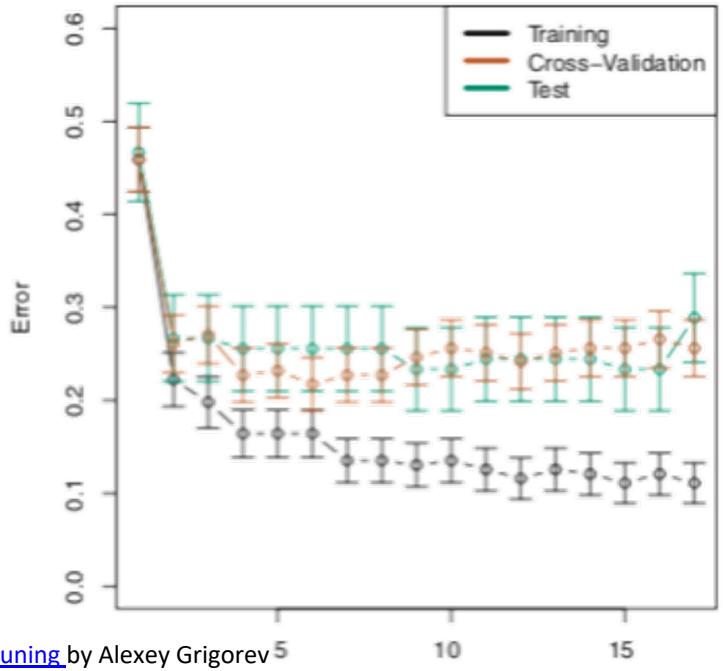
Characteristic	Neural Nets	SVM	Trees	MARS	k-NN, Kernels
Natural handling of data of "mixed" type	▼	▼	▲	▲	▼
Handling of missing values	▼	▼	▲	▲	▲
Robustness to outliers in input space	▼	▼	▲	▼	▲
Insensitive to monotone transformations of inputs	▼	▼	▲	▼	▼
Computational scalability (large N)	▼	▼	▲	▲	▼
Ability to deal with irrelevant inputs	▼	▼	▲	▲	▼
Ability to extract linear combinations of features	▲	▲	▼	▼	◆
Interpretability	▼	▼	◆	▲	▼
Predictive power	▲	▲	▼	◆	▲

2013 - Book - Hastie - The Elements of Statistical Learning= Data Mining, Inference, and Prediction

Weakest link pruning



Prune



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

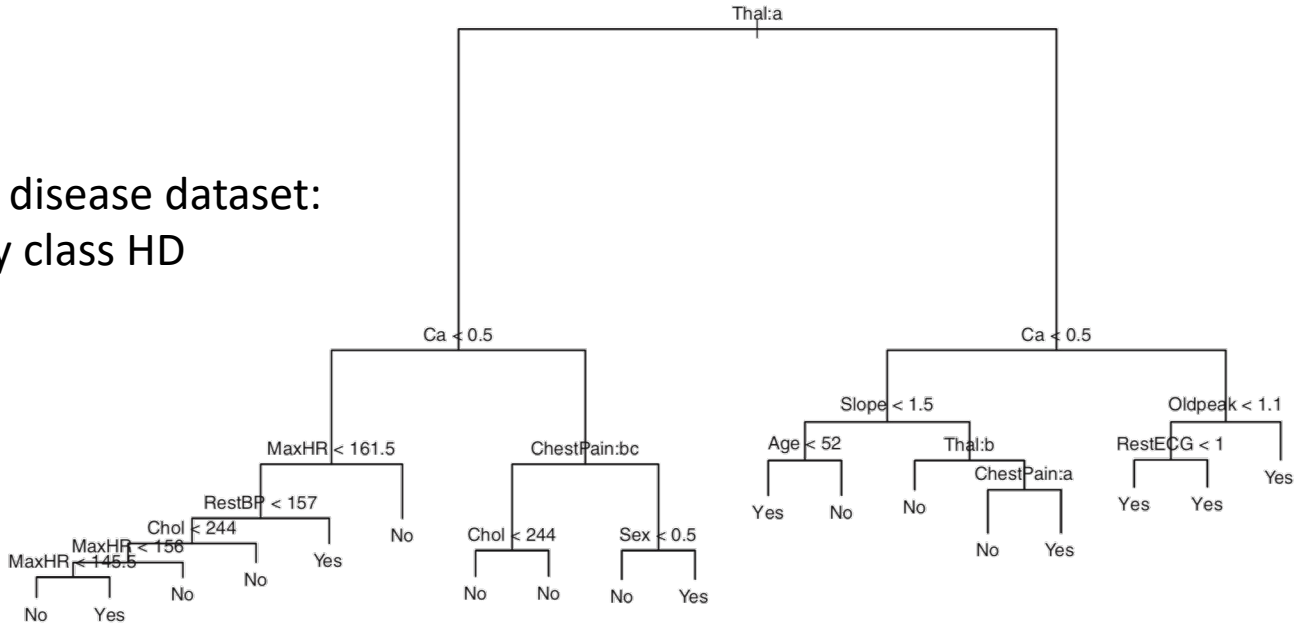
L_2 regularization plus an additional term limiting number of leaves

[cost-complexity pruning](#) by Alexey Grigorev

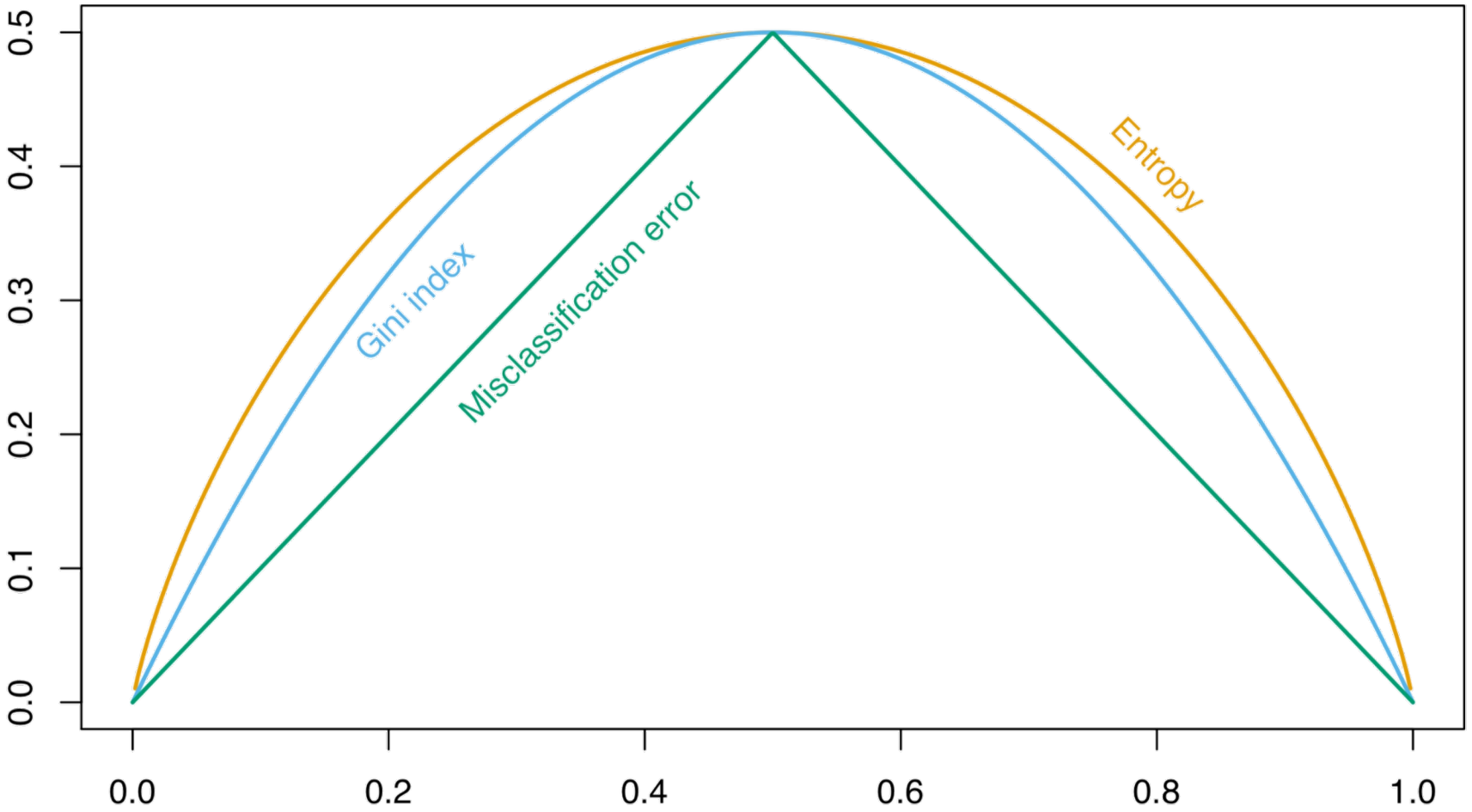
Building a classification tree

- Similar to regression, but for **qualitative** response.
- Observation belongs to **most commonly occurring class** of training observations in the region to which it belongs (i.e., *Mode* instead of *Mean*).
- Quantitative but also qualitative values are allowed
- RSS cannot be used as a criterion for making the binary splits.

Heart disease dataset:
Binary class HD



Measures of node impurity



Misclassification error: $E = 1 - \max_k(\hat{p}_{mk})$.

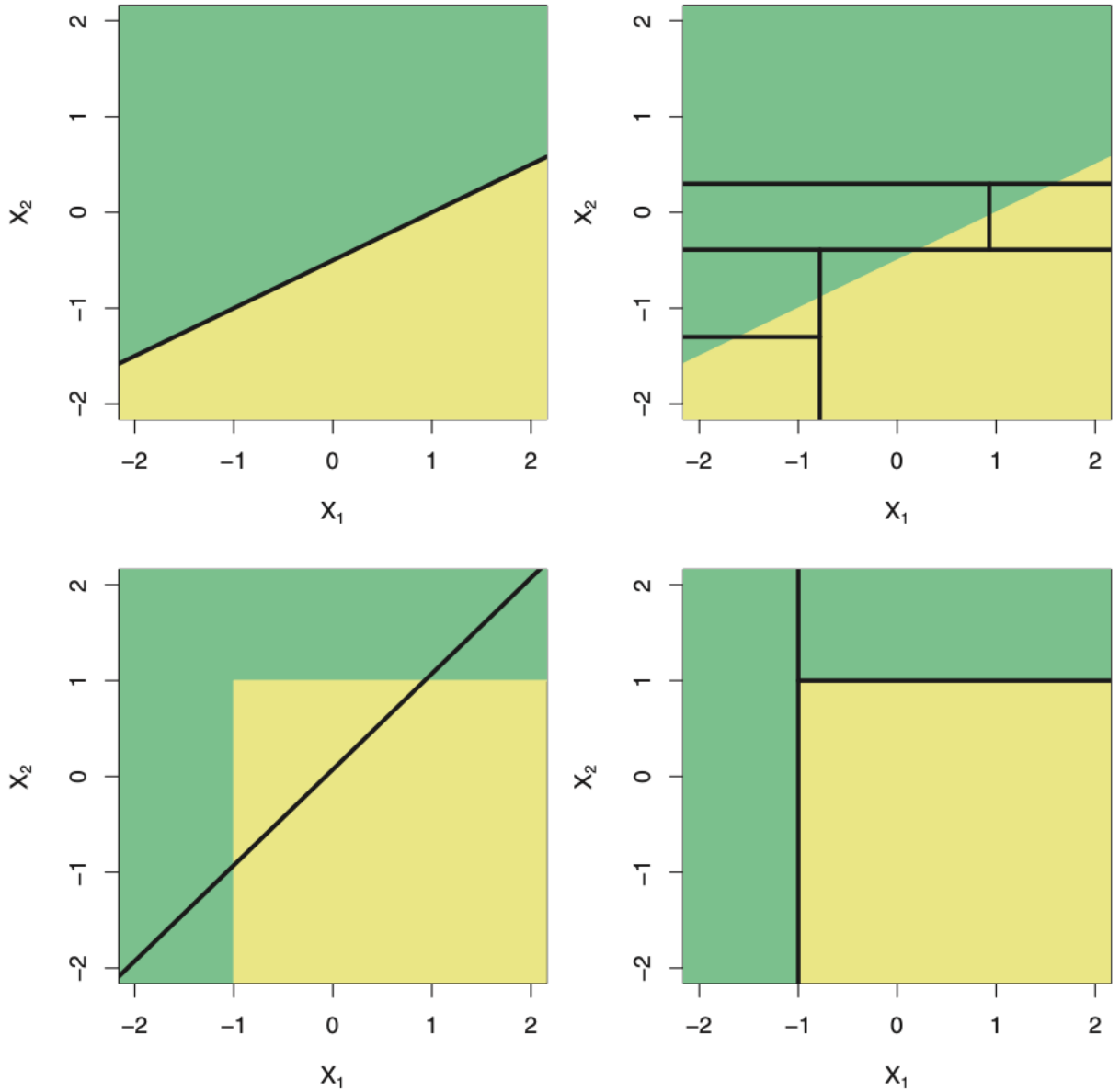
Gini index: $\sum_{k \neq k'} \hat{p}_{mk}\hat{p}_{mk'} = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$.

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

With k-class frequency in R_m:

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k)$$

Decision trees vs linear models: it depends



Trees - Pros & Cons

▲ Trees are easy to interpret.

▲ They closely mirror human decision-making (compared to other approaches).

▲ Easy to visualize graphically (especially if they are small).

▲ Trees can easily handle qualitative predictors and missing data.

▼ They are often bad predictors, high tendency to overfit (high variance)

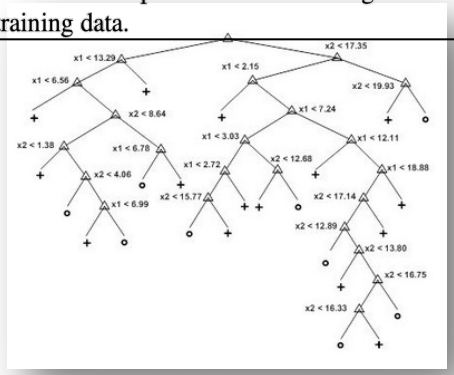
Decision Trees do not Generalize to New Variations

Yoshua Bengio, Olivier Delalleau and Clarence Simard
Dept. IRO, Université de Montréal
C.P. 6128, Montreal, Qc, H3C 3J7, Canada
{*bengioy,delallea,simardcl*}*@iro.umontreal.ca*
http://www.iro.umontreal.ca/∼lisa
Technical Report 1304

June 8th, 2007

Abstract

The family of decision tree learning algorithms is among the most widespread and studied. Motivated by the desire to develop learning algorithms that can generalize when learning highly variable functions such as those presumably needed to achieve artificial intelligence, we study some theoretical limitations of decision trees. We demonstrate that they can be seriously hurt by the curse of dimensionality, but most importantly, that they cannot generalize to variations not seen in the training set. This is because a decision tree creates a partition of the input space and needs at least one example in each of the regions associated with a leaf in order to make a sensible prediction in that region. A better understanding of the fundamental reasons for this limitation suggests that one should use *forests* instead of trees, which provide a form of distributed representation and can generalize to variations not encountered in the training data.



Ensemble learning

Concordia res parvae crescunt – Unity makes strength

Can a set of weak learners create a single strong learner? (Michael Kearns, '88)

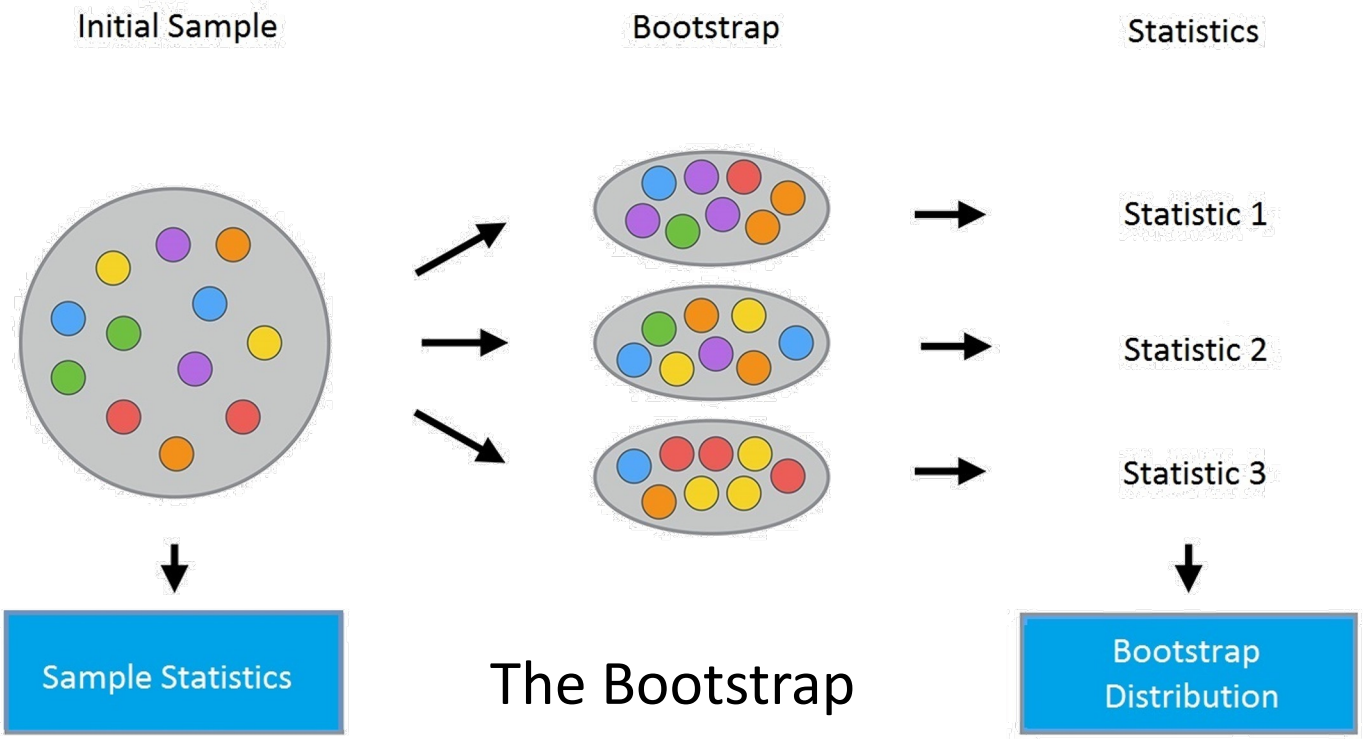
sklearn.ensemble module

Resampling methods

The Bootstrap

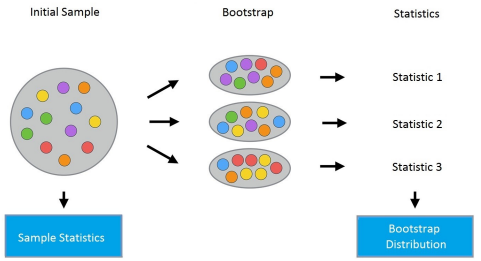
Leo Breiman
https://www.stat.berkeley.edu/users/breiman/

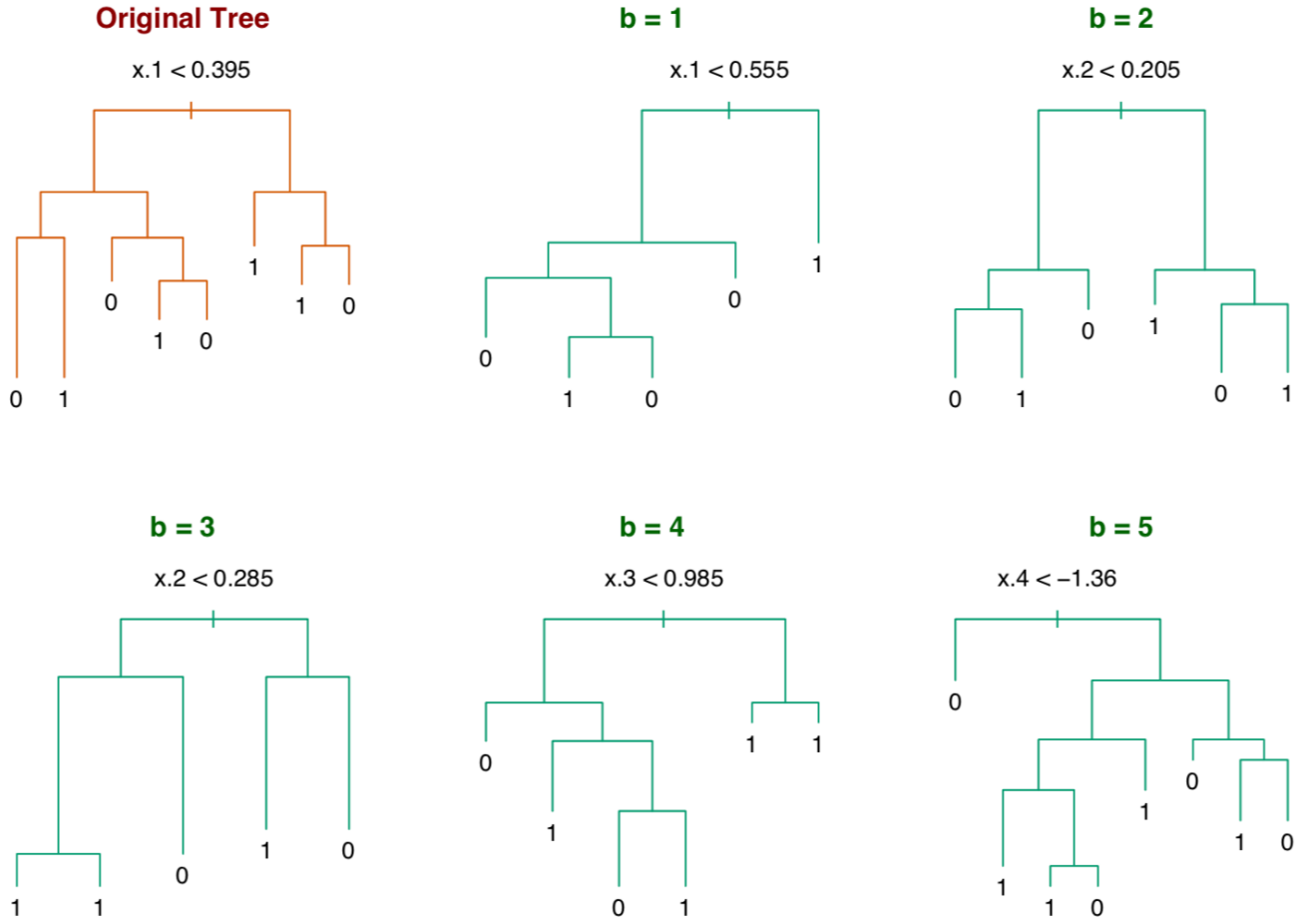
Adele Cutler
https://www.usu.edu/math/adele/



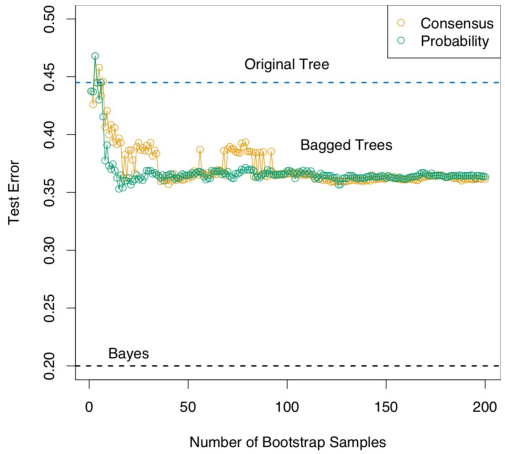
Bagging

Average B (unpruned?) trees out of a bootstrapped sample, with B large enough (e.g., 100 or more), *Bagged trees "don't overfit"*



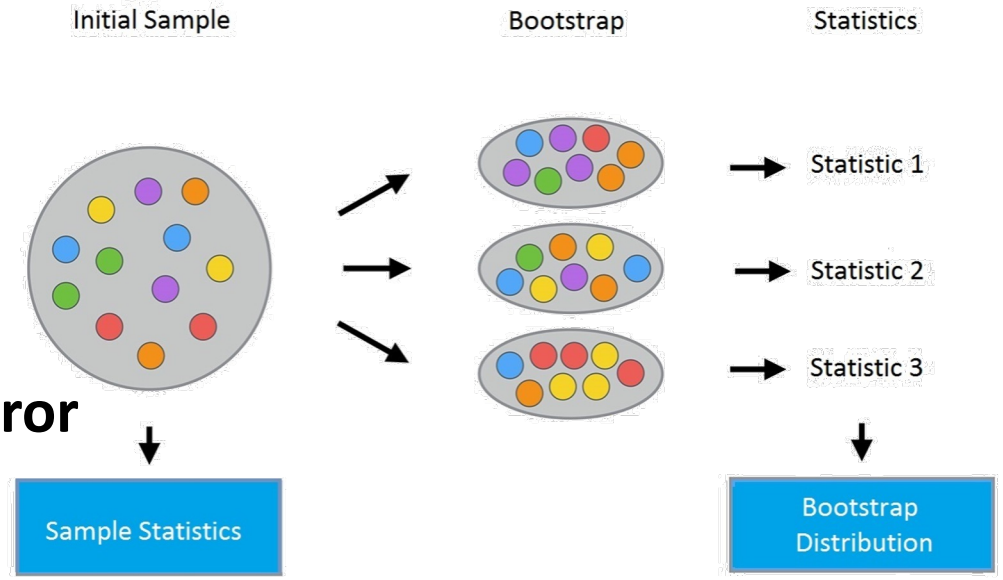


$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x).$$



ensemble.BaggingClassifier
ensemble.BaggingRegressor

Out-of-Bag (OOB) Error Estimation



- Method **to estimate the test error** of a bagged model.

- In bagging, trees are repeatedly fit to bootstrapped subsets of the observations. On average, each bagged tree makes use of around 2/3 of the observations (actually, 1-1/e). The remaining ~1/3 are referred to as the **out-of-bag (OOB) observations.**

- We can predict the response for the i^{th} observation using each of the trees in which that observation was OOB. This will yield around ~B/3 predictions for the ith observation, which we can average and then estimate the error.

OOB Errors for Random Forests (scikit-learn)


Random Forests

Do we Need Hundreds of Classifiers to Solve Real World Classification Problems?

(2014)

Manuel Fernández-Delgado MANUEL.FERNANDEZ.DELGADO@USC.ES
Eva Cernadas EVA.CERNADAS@USC.ES
Senén Barro SENEN.BARRO@USC.ES
CITIUS: Centro de Investigación en Tecnoloxías da Información da USC
University of Santiago de Compostela
Campus Vida, 15872, Santiago de Compostela, Spain

Dinani Amorim DINANIAMORIM@GMAIL.COM
Departamento de Tecnologia e Ciências Sociais- DTCS
Universidade do Estado da Bahia
Av. Edgard Chastinet S/N - São Geraldo - Juazeiro-BA, CEP: 48.305-680, Brasil

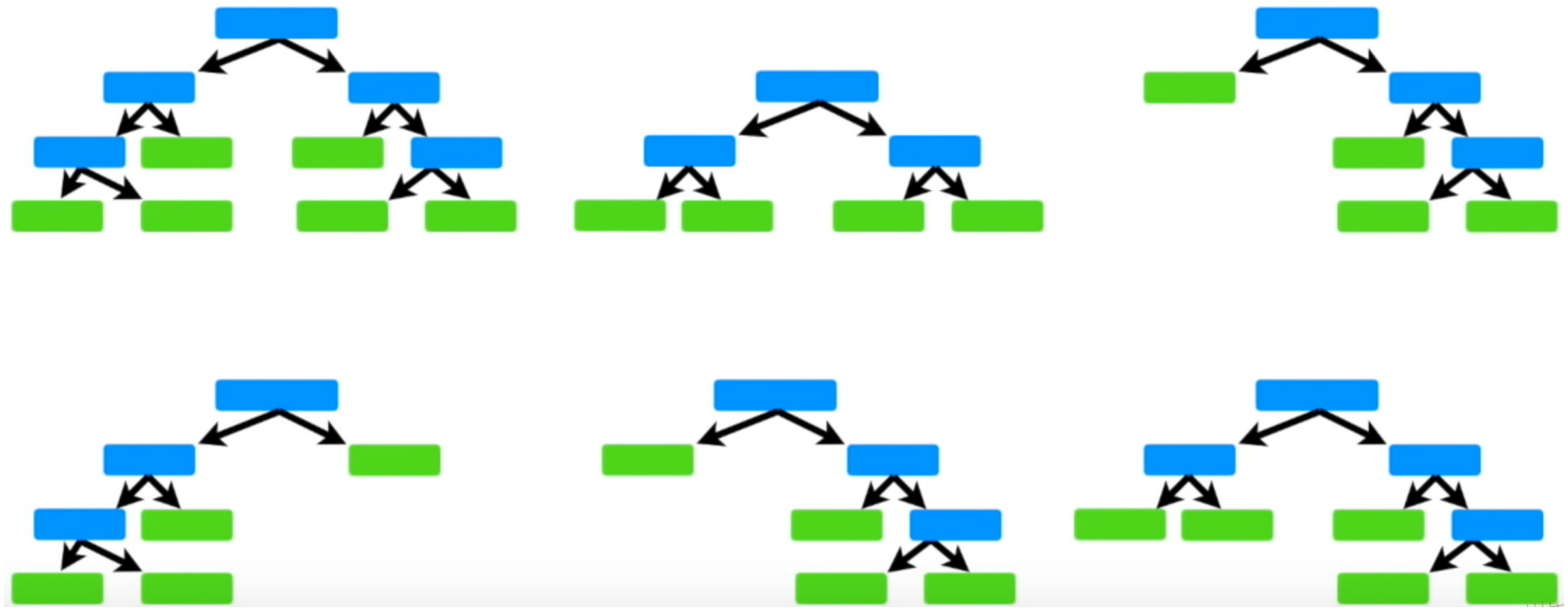
Editor: Russ Greiner

Abstract

We evaluate **179 classifiers** arising from **17 families** (discriminant analysis, Bayesian, neural networks, support vector machines, decision trees, rule-based classifiers, boosting, bagging, stacking, random forests and other ensembles, generalized linear models, nearest-neighbors, partial least squares and principal component regression, logistic and multinomial regression, multiple adaptive regression splines and other methods), implemented in Weka, R (with and without the caret package), C and Matlab, including all the relevant classifiers available today. We use **121 data sets**, which represent **the whole UCI** data base (excluding the large-scale problems) and other own real problems, in order to achieve significant conclusions about the classifier behavior, not dependent on the data set collection. **The classifiers most likely to be the bests are the random forest** (RF) versions, the best of which (implemented in R and accessed via caret) achieves 94.1% of

Random Forests **randomly select a subgroup of attributes at each split**

- **Random forests** provide an improvement over bagged trees by way of a small tweak that **decorrelates** the trees, reducing the variance when we average the trees.
- The idea is, for each bagged tree, to **randomly select a subgroup of predictors.**

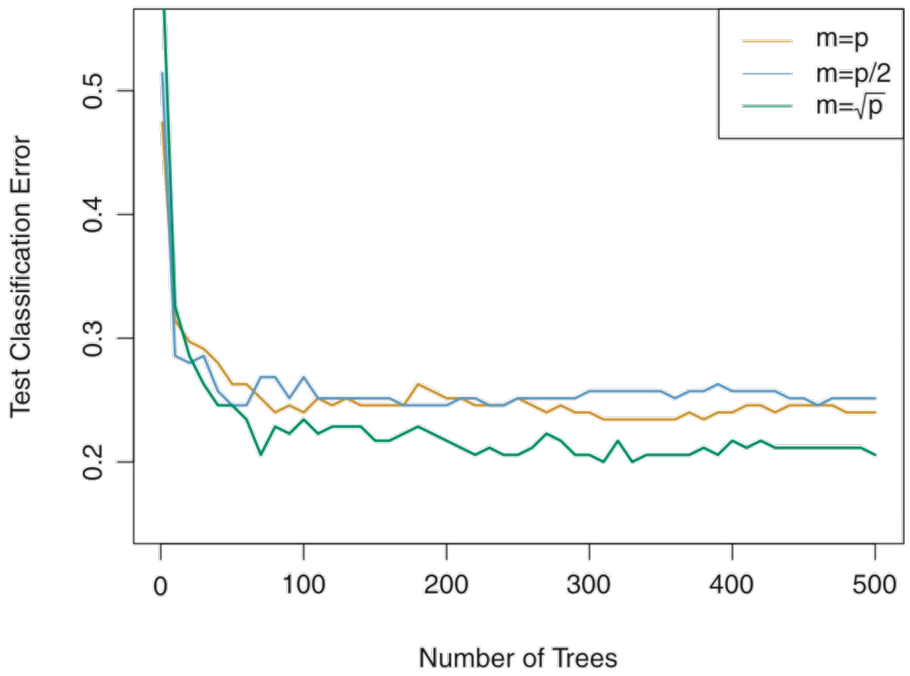


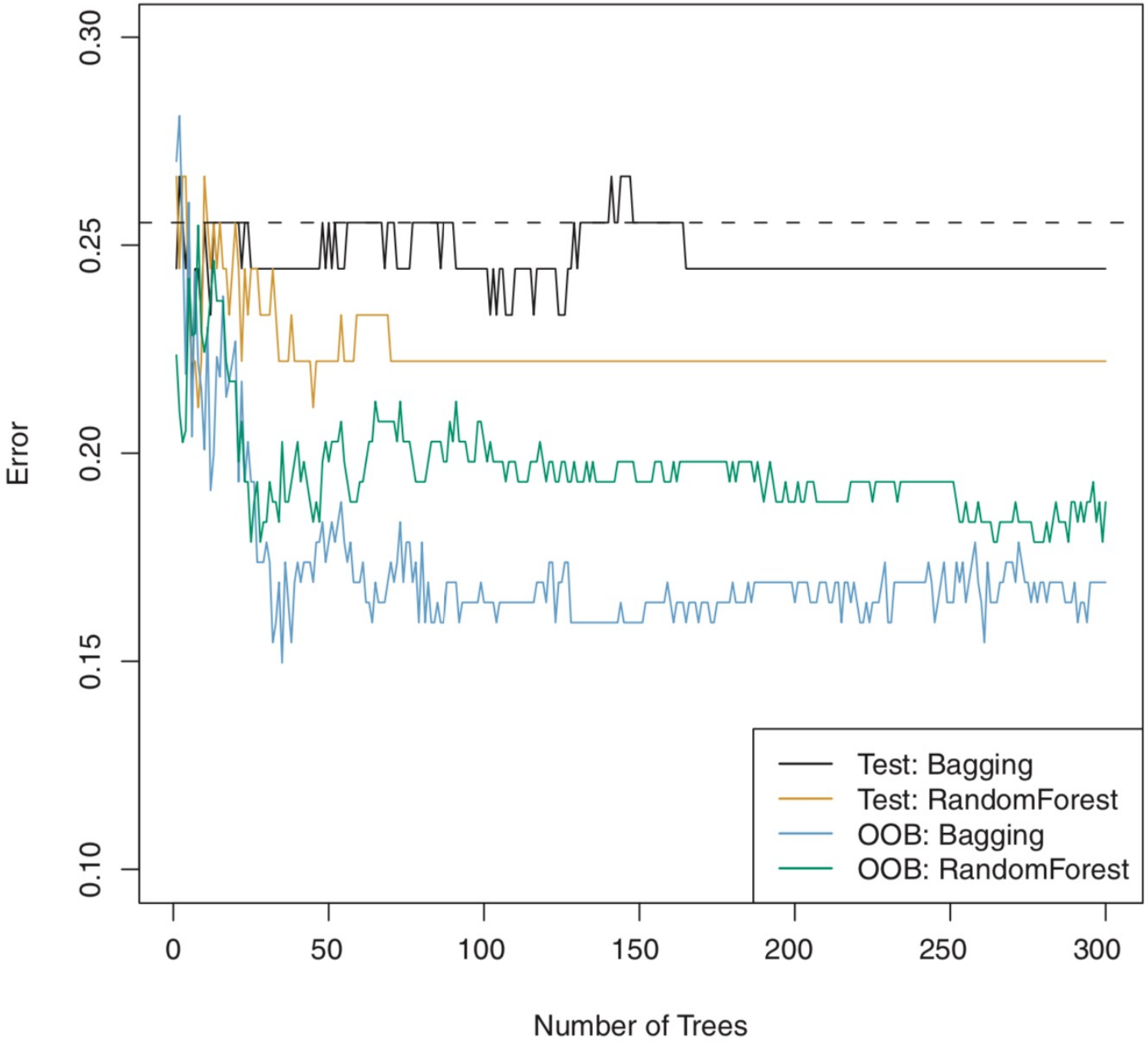
ensemble.RandomForestClassifier
ensemble.RandomForestRegressor

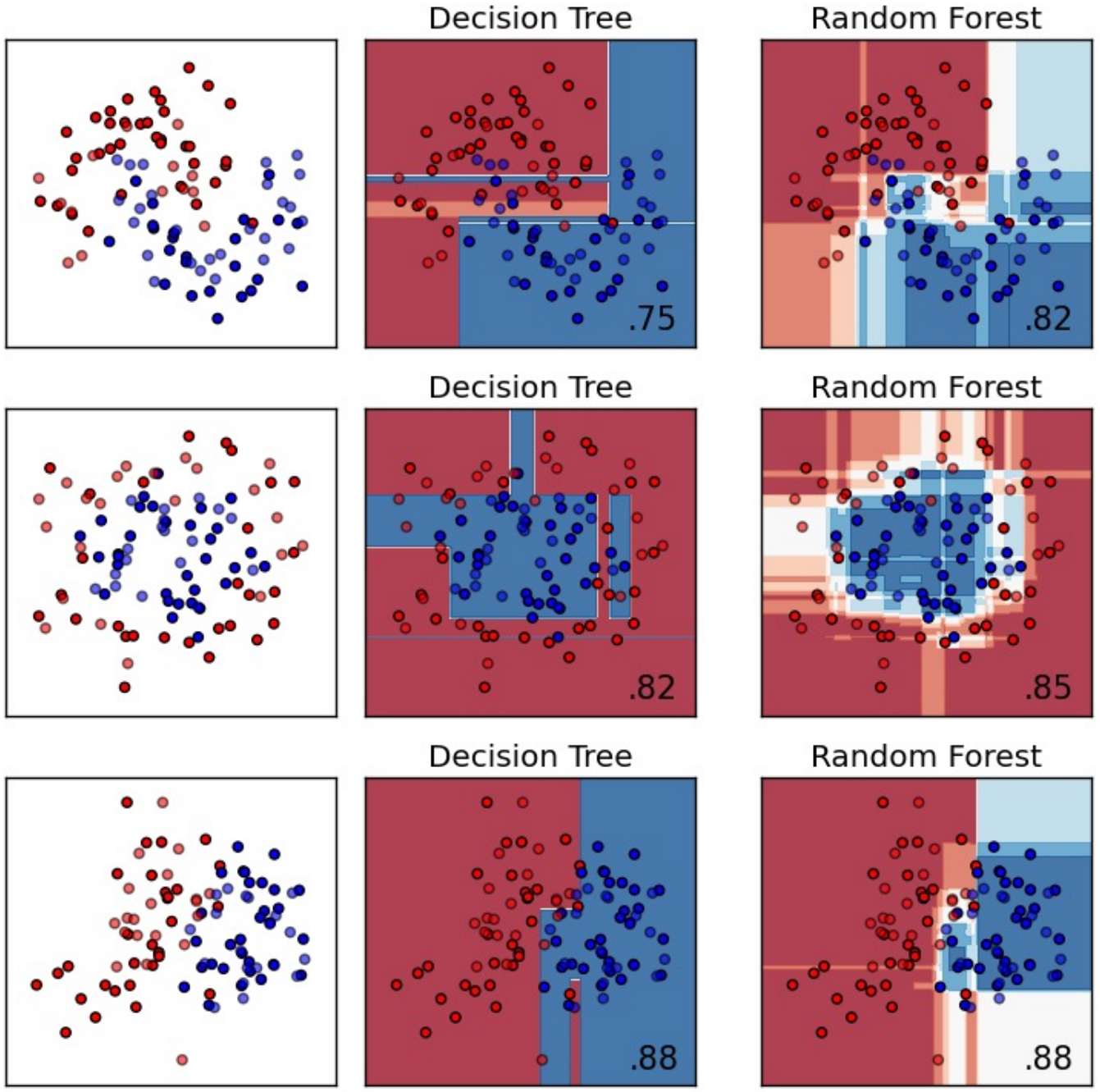
Random Forests

- As in bagging, we build a number of decision trees on bootstrapped training samples.
- But when building these decision trees, each time a split in a tree is considered, a **random selection of m predictors** is chosen as split candidates from the full set of p predictors. The split is allowed to use only one of those m predictors.
- A fresh selection of m predictors is taken at each split, and a typically choosen m is m ≈ sqrt(p). (Bagging is recovered if m = p).

max_features parameter in scikt-learn







https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html

RF Feature importance

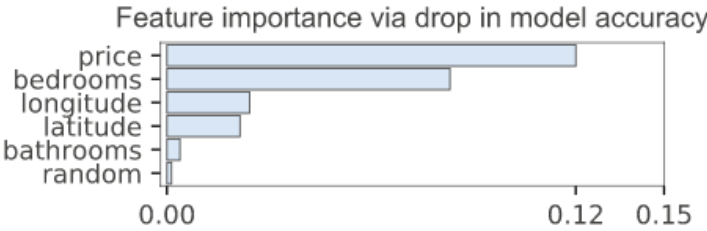


Figure 2(b). Importances derived by permuting each column and computing change in out-of-bag accuracy using `scikit-learn` Random Forest **classifier**.

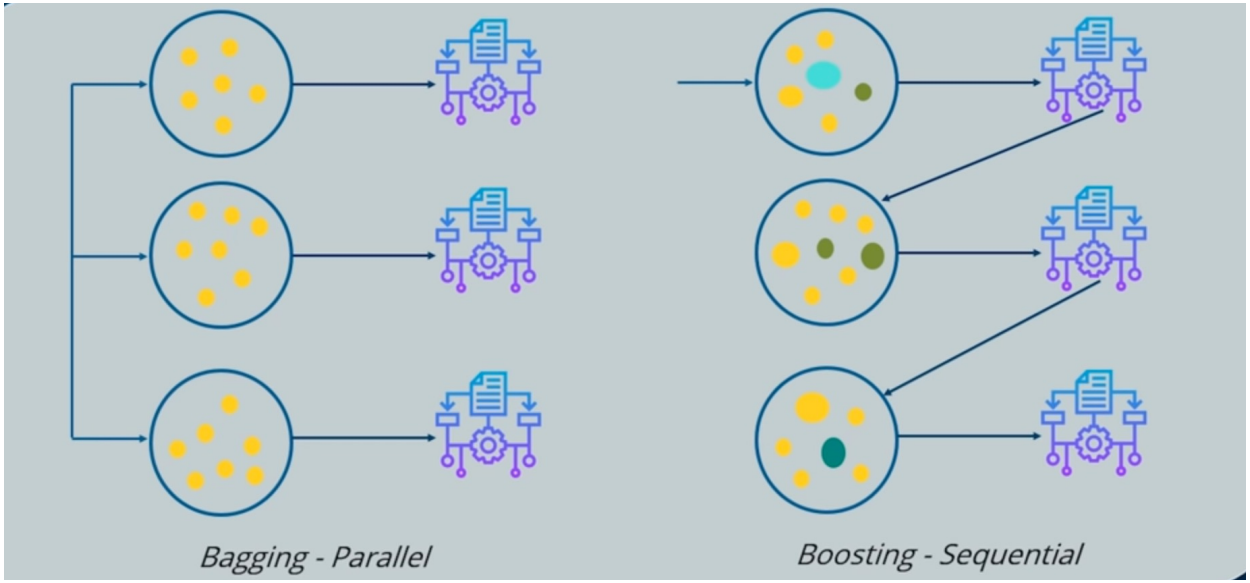
- RF provide feature importance scores
  - *mean decrease in impurity, or Gini importance (how much impurity decreases when feature is used in split).*
- Scikit-learn's Random Forest feature importance (as well as R's default) strategies **are biased**.
- Better: feature importance **by permutation**
  - measures drop of accuracy/R^2 after permutating a given variable values (breaking any discriminative power on the target).

https://scikit-learn.org/stable/modules/permutation_importance.html

https://explained.ai/rf-importance/index.html

Boosting methods

- Boosting is a **general approach** for improving the predictions resulting from statistical learning methods.
- Succesful technique for ensembles of weak learners.
- As in *Bagging*, Boosting also grows several trees, but **does not** involve bootstrap.
- Rather, Boosting works growing the trees **sequentially.**
- Most frequently used:
  - **AdaBoost.M1**, Freund and Schapire (1997)
  - **Gradient Boosting Machines** (GBMs), Friedman (1999).



ensemble.GradientBoostingClassifier and
ensemble.AdaBoostClassifier,
also check "Multi-Class AdaBoost," J. Zhu et al. (2006).

2022 – Zhang et al. - Building Robust Ensembles via Margin Boosting

Gradient boosting involves 3 elements

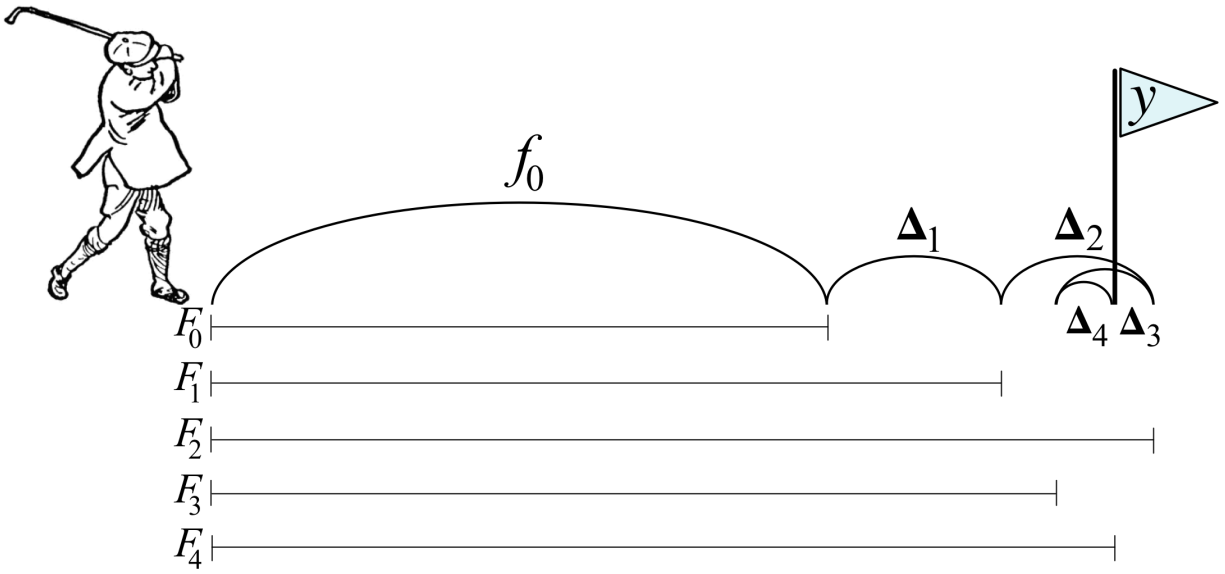
1. A loss function to be optimized.

   Squared error, absolute error, logarithmic loss... Differentiable.

2. A weak learner to make predictions (*on the residuals*).

   Most usually, stumps (AdaBoost) or short trees (Gradient Boosting). Sometimes other models such as DNNs (Zhang, Bengio et al 2022, applying boosting in DL).

3. An additive model to add weak learners to minimize the loss function.



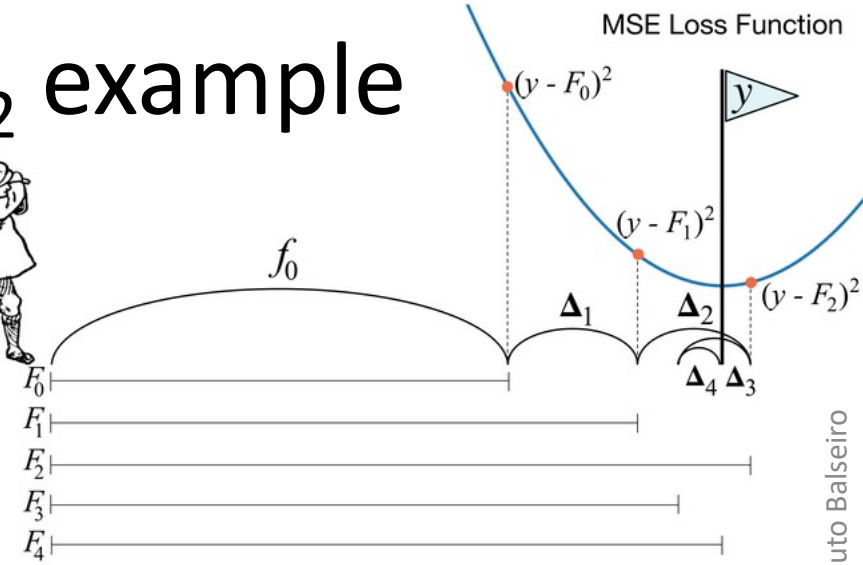
https://explained.ai/gradient-boosting/L2-loss.html

Gradient boosting – L_2 example

$$\begin{aligned}\hat{y} &= f_0(\mathbf{x}) + \Delta_1(\mathbf{x}) + \Delta_2(\mathbf{x}) + \ldots + \Delta_M(\mathbf{x}) \\ &= f_0(\mathbf{x}) + \sum_{m=1}^{M} \Delta_m(\mathbf{x}) \\ &= F_M(\mathbf{x})\end{aligned}$$

$$\begin{aligned}F_0(\mathbf{x}) &= f_0(\mathbf{x}) \\ F_m(\mathbf{x}) &= F_{m-1}(\mathbf{x}) + \Delta_m(\mathbf{x})\end{aligned}$$

$$F_m(X) = F_{m-1}(X) + \eta\Delta_m(X)$$

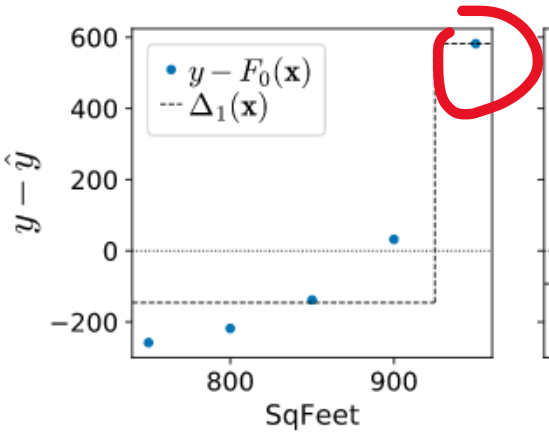


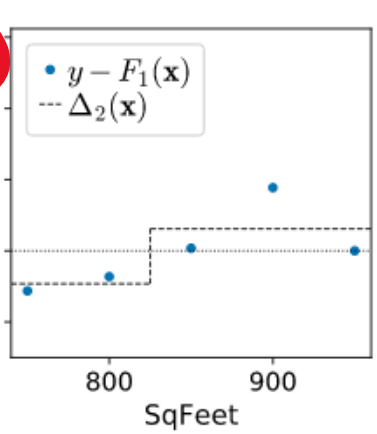
L_2 Loss

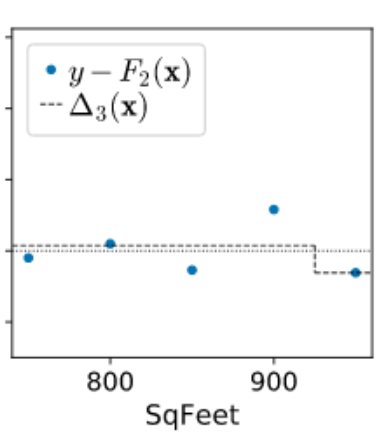
residual vector

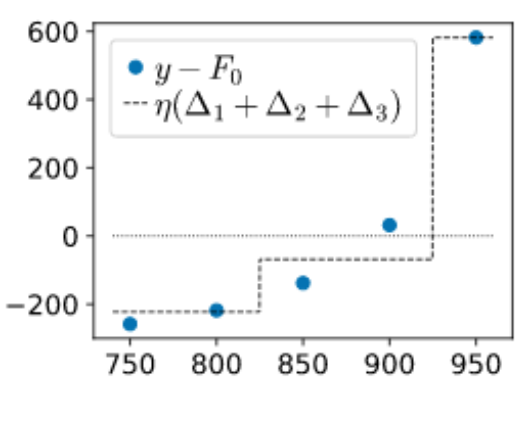
$$L(\mathbf{y}, F_M(X)) = \frac{1}{N}\sum_{i=1}^{N}(y_i - F_M(\mathbf{x}_i))^2$$

sqfeet	rent	F_0	$\mathbf{y} - F_0$	Δ_1	F_1	$\mathbf{y}$-F_1	Δ_2	F_2	$\mathbf{y} - F_2$	Δ_3	F_3
750	1160	1418	-258	-145.5	1272.5	-112.5	-92.5	1180	-20	15.4	1195.4
800	1200	1418	-218	-145.5	1272.5	-72.5	-92.5	1180	20	15.4	1195.4
850	1280	1418	-138	-145.5	1272.5	7.5	61.7	1334.2	-54.2	15.4	1349.6
900	1450	1418	32	-145.5	1272.5	177.5	61.7	1334.2	115.8	15.4	1349.6
950	2000	1418	582	582	2000	0	61.7	2061.7	-61.7	-61.7	2000







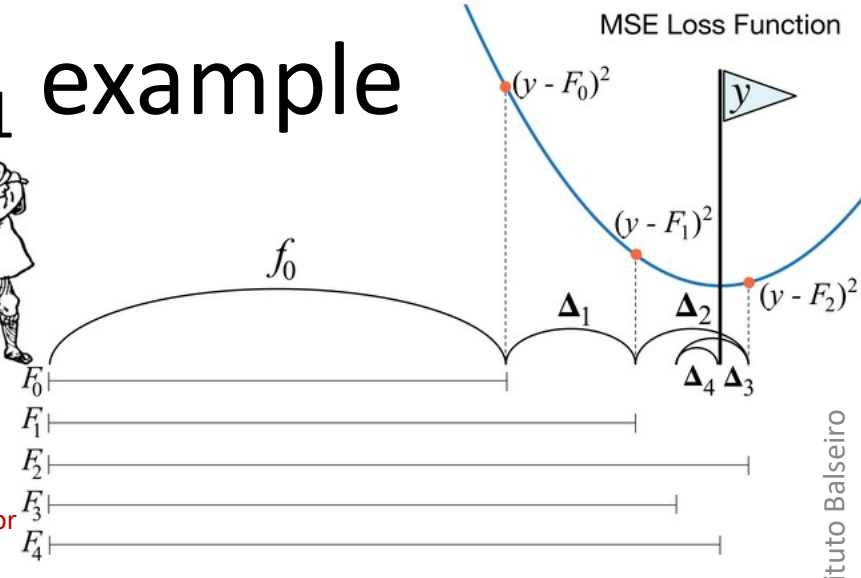


Gradient boosting – L_1 example

$$\begin{aligned}\hat{y} &= f_0(\mathbf{x}) + \Delta_1(\mathbf{x}) + \Delta_2(\mathbf{x}) + \ldots + \Delta_M(\mathbf{x}) \\ &= f_0(\mathbf{x}) + \sum_{m=1}^{M} \Delta_m(\mathbf{x}) \\ &= F_M(\mathbf{x})\end{aligned}$$

$$\begin{aligned}F_0(\mathbf{x}) &= f_0(\mathbf{x}) \\ F_m(\mathbf{x}) &= F_{m-1}(\mathbf{x}) + \Delta_m(\mathbf{x})\end{aligned}$$

$$F_m(X) = F_{m-1}(X) + \eta\Delta_m(X)$$

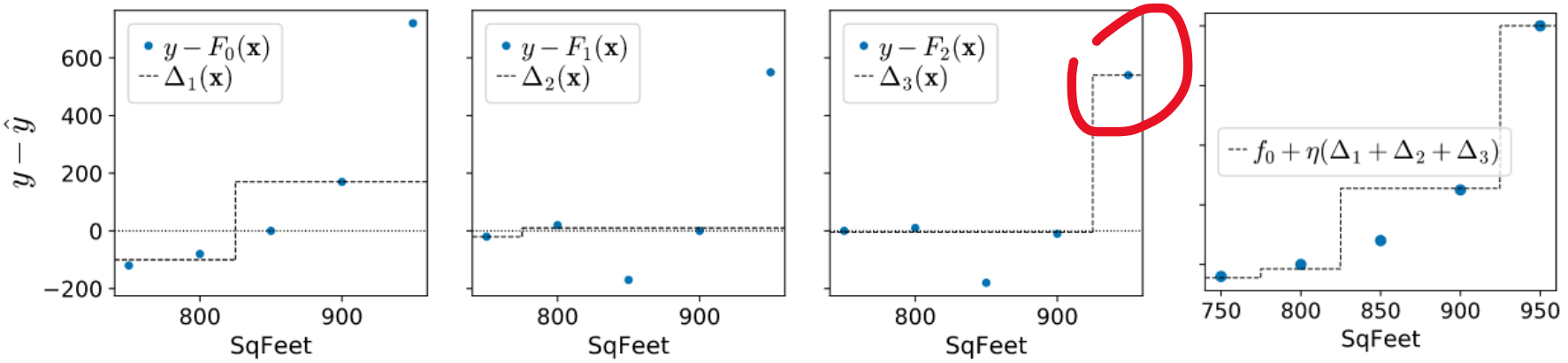


L_1 Loss

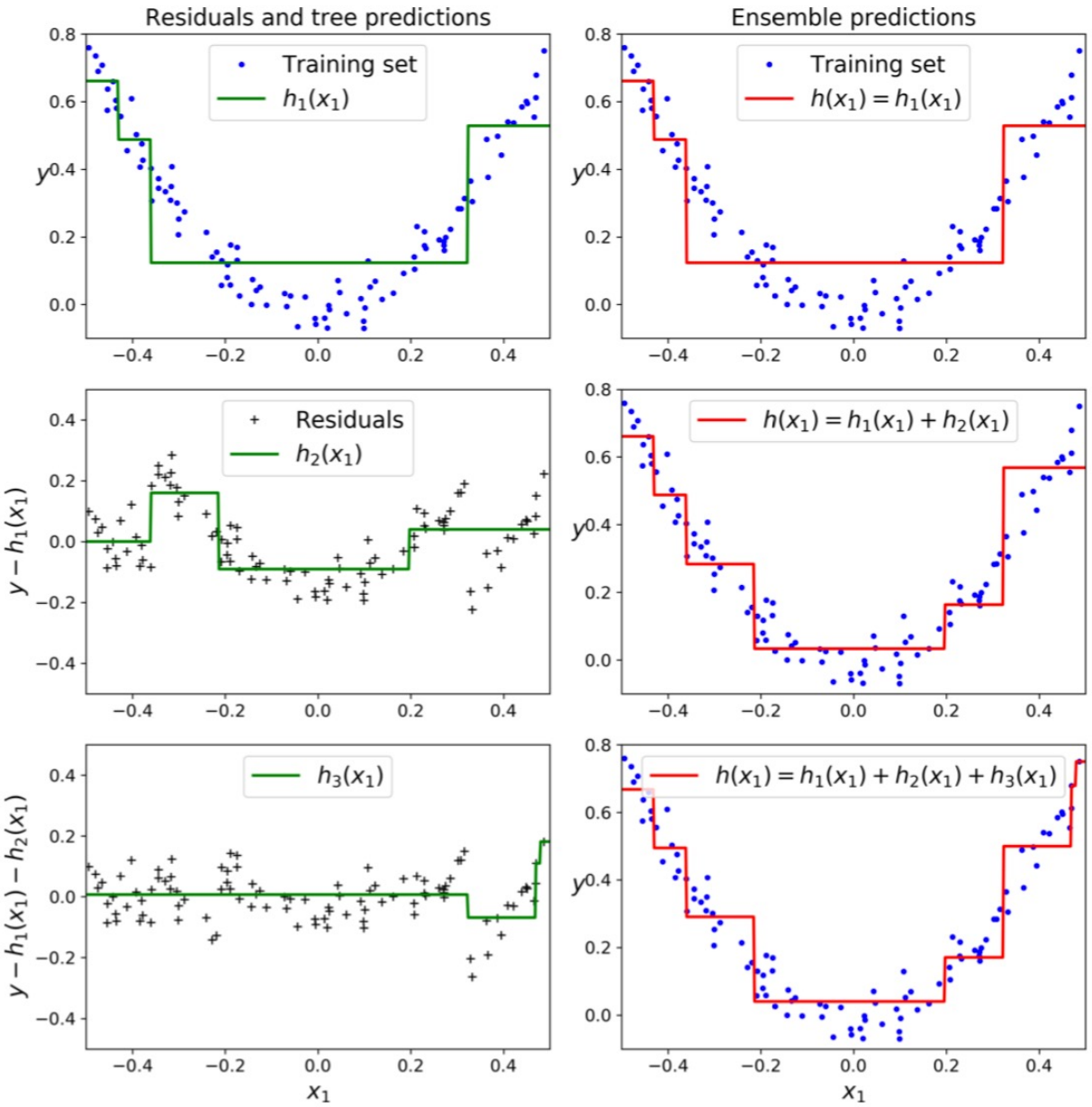
sign residual vector

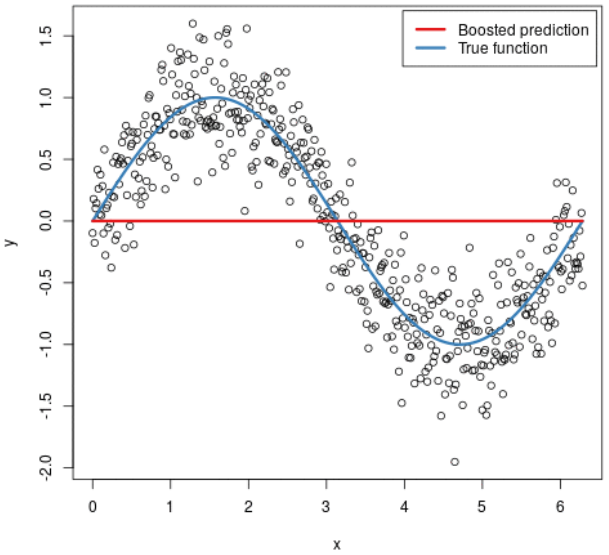
$$L(\mathbf{y}, F_M(X)) \quad \frac{1}{N}\sum_i^N |y_i - F_m(\mathbf{x}_i)|$$

sqfeet	rent	F_0	y-F_0	$sign$(y-F_0)	Δ_1	F_1	y-F_1	$sign$ y-F_1	Δ_2	F_2	y-F_2	$sign$ y-F_2	Δ_3	F_3
750	1160	1280	-120	-1	-100	1180	-20	-1	-20	1160	0	0	-5	1155
800	1200	1280	-80	-1	-100	1180	20	1	10	1190	10	1	-5	1185
850	1280	1280	0	0	170	1450	-170	-1	10	1460	-180	-1	-5	1455
900	1450	1280	170	1	170	1450	0	0	10	1460	-10	-1	-5	1455
950	2000	1280	720	1	170	1450	550	1	10	1460	540	1	540	2000



Boosting methods





Where's the gradient in GBMs?

- GBMs do gradient descent in **function space**

Gradient descent

$$\mathbf{x}_t = \mathbf{x}_{t-1} - \eta \nabla f(\mathbf{x}_{t-1})$$

Gradient boosting

$$\hat{\mathbf{y}}_m = \hat{\mathbf{y}}_{m-1} + \eta(-\nabla L(\mathbf{y}, \hat{\mathbf{y}}_{m-1}))$$

$$\hat{\mathbf{y}}_m = F_m(X)$$

MSE: $L(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$

$$\nabla_{\hat{\mathbf{y}}} L(\mathbf{y}, \hat{\mathbf{y}}) = -2(\mathbf{y} - \hat{\mathbf{y}})$$

MAE: $L(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{i=1}^{N} |y_i - \hat{y}_i|$

$$\nabla_{\hat{\mathbf{y}}} L(\mathbf{y}, \hat{\mathbf{y}}) = -sign(\mathbf{y} - \hat{\mathbf{y}})$$

- GBMs do not *compute* the direction vector as gradient descent training does, it *approximates* it via weak learners

Gradient Boosting, revisited

- Fit a decision tree **to the residuals (or their sign)** from the model, and repeat.

- Add new tree into the fitted function and update residuals.

- By fitting small trees to the residuals, we slowly **improve** the model in areas **where it does not perform wel**l.
  - Trees can be **small**, with just a few terminal nodes.

- The shrinkage parameter η slows the process down, allowing further different shaped trees to attack the residuals.

- If learners are trained in the residual vectors, MSE is optimized. If on sign vectors, MAE is optimized.

Gradient Boosting formalized

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.
2. For $b = 1, 2, \ldots, B$, repeat:
   - 2.1 Fit a tree $\hat{f}^b$ with d splits ($d + 1$ terminal nodes) to the training data (X, r).
   - 2.2 Update $\hat{f}$ by adding in a shrunken version of the new tree:

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x).$$

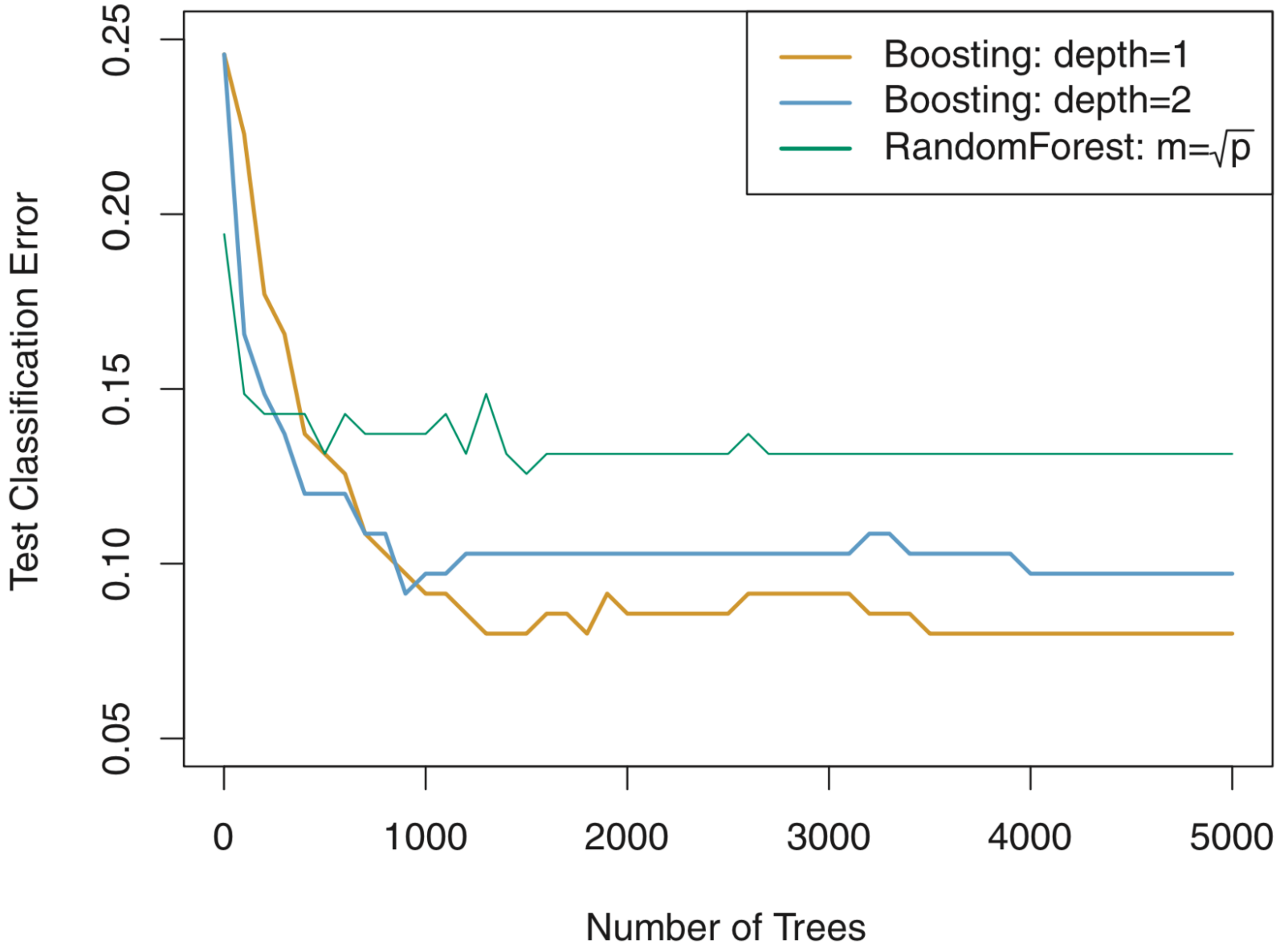
   - 2.3 Update the residuals,

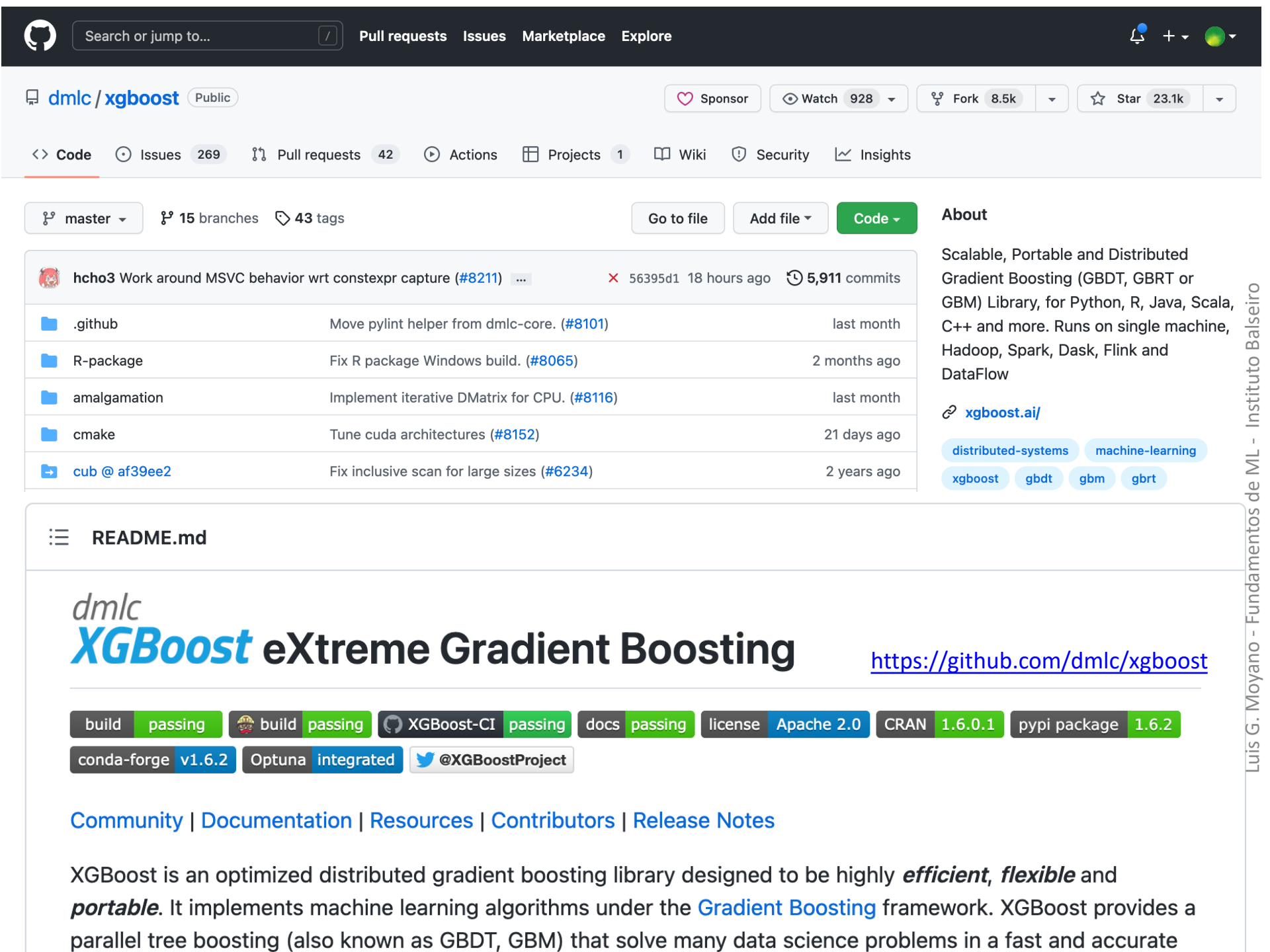
$$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i).$$

3. Output the boosted model,

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x).$$

Boosting methods





```
# First XGBoost model for Pima Indians dataset
from numpy import loadtxt
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
# load data
dataset = loadtxt('pima-indians-diabetes.csv', delimiter=",")
# split data into X and y
X = dataset[:,0:8]
Y = dataset[:,8]
# split data into train and test sets
seed = 7
test_size = 0.33
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=test_size,
    random_state=seed)
# fit model on training data
model = XGBClassifier()
model.fit(X_train, y_train)
# make predictions for test data
predictions = model.predict(X_test)
# evaluate predictions
accuracy = accuracy_score(y_test, predictions)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

XGBoost is faster

Trees consider the potential loss for all possible splits to create a new split.

XGBoost tackles this inefficiency by looking at the distribution of features across all data points in a leaf and using this information to reduce the search space of possible feature splits.

https://www.gormanalysis.com/blog/gradient-boosting-explained/

λ = reg_alpha, α = reg_lambda

XGBoost

$$L = \sum_{i=0}^{n} loss(y_{res}, h(x)) + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 + \alpha \sum_{j=1}^{T} |w_j|$$

- **Variance reduction**
  - Shrinkage parameter, which affects the learning rate
  - Stochastic Gradient Boosting (column and row subsampling)
  - Regularized Gradient Boosting, with both L1 and L2 regularization.
- **Sparsity is dealt with in a fast and consistent way**
  - presence of missing values in the data
  - frequent zero entries in the statistics
  - artifacts of feature engineering such as one-hot encoding.

  All these cases should be encoded as '0' and contribute to the minimization of the Loss function.

  Additionally, binary classes should be encoded as '0' and '1'.

XGBoost - Parameters and tips

- **Parameters** https://xgboost.readthedocs.io/en/latest/parameter.html
  - n_estimators = 100 (number of trees) (target 500-1000)
  - learning rate = 0.1 (shrinkage). (2 to 10)/trees
  - max_depth = 3
  - min_samples_split = 2
  - min_samples_leaf = 1
  - subsample = 1.0. [0.5, 0.75, 1.0]
  - colsample_bytree [0.4, 0.6, 0.8, 1.0]
  - gamma = [0.1, 1, 5, 20]
- **Tips**
  - Run the default configuration (and presumably review learning curves).
  - If the system is overlearning, slow the learning down using *learning_rate* (and viceversa).
  - To control complexity: tweak *max_depth*, *min_child_weight* and *gamma*
  - To add noise for robustness: tweak *subsample* and *colsample_bytree*.
  - Some people avoid touching the regularization penalties (*reg_alpha* and *reg_lambda*).
  - Set *tree_method* to *hist* or *gpu_hist* for faster computation.

  - More detailed notes on tuning here. Also check this example

Jason Brownlee - XGBoost with Python. 1.10

ML Fundamentals –Trees

- Decision trees
  - Regression
  - Classification: Gini coeff vs. Cross Entropy
  - Prunning
- Ensembles
- Random Forests
- Boosting and GBMs
- XGBoost
  - parameters & tips

Now, Practice.

Next, unsupervised I.