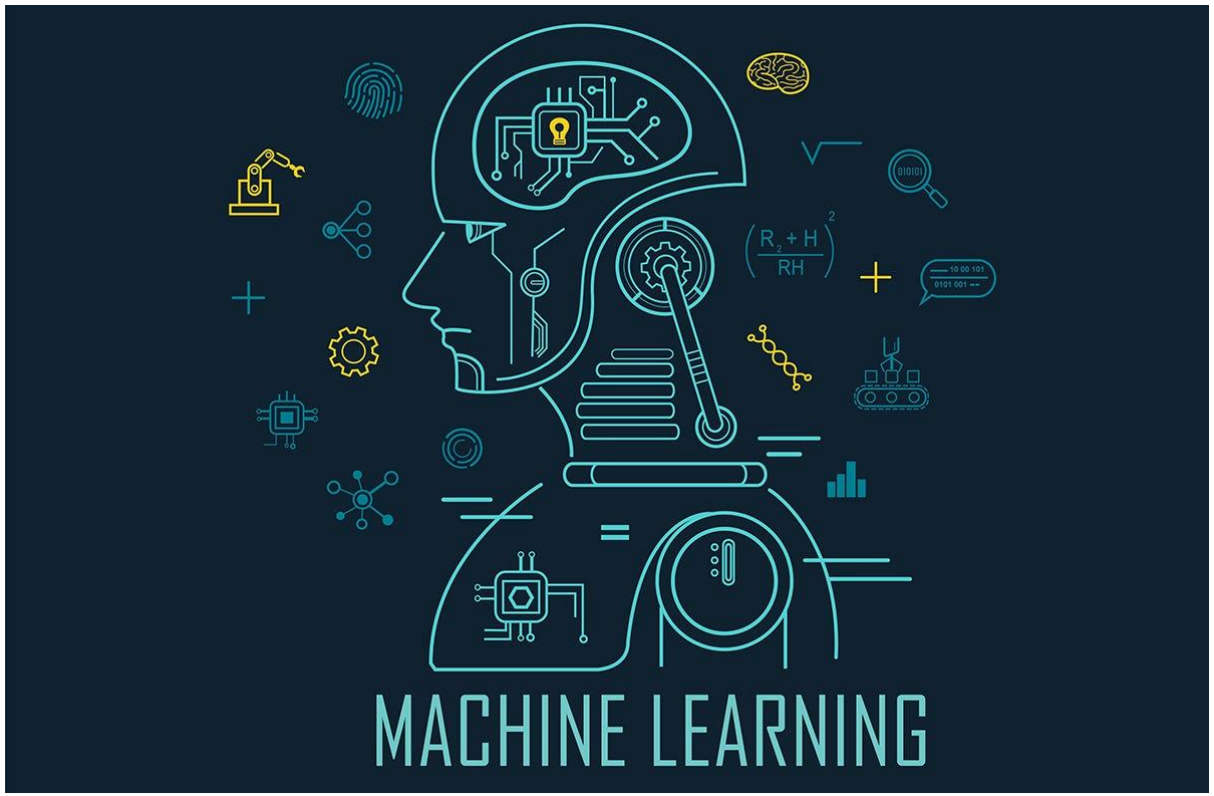
NOVA IMS Information Management School

Machine Learning in Finance

Prof. Ian J. Scott

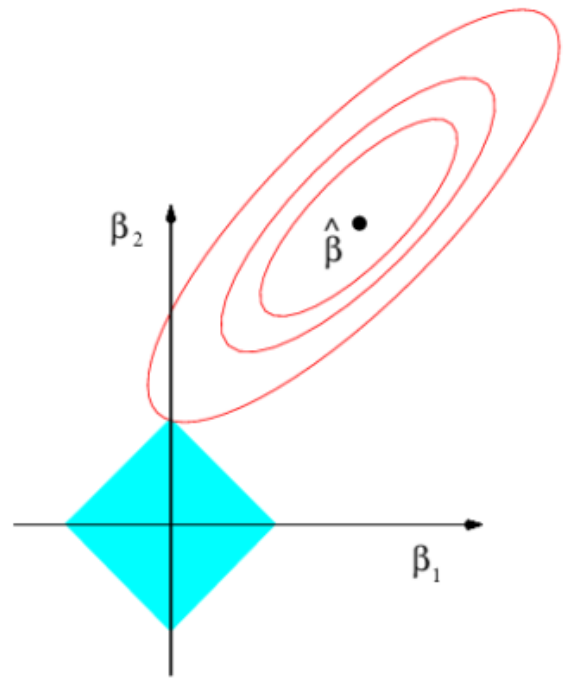
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

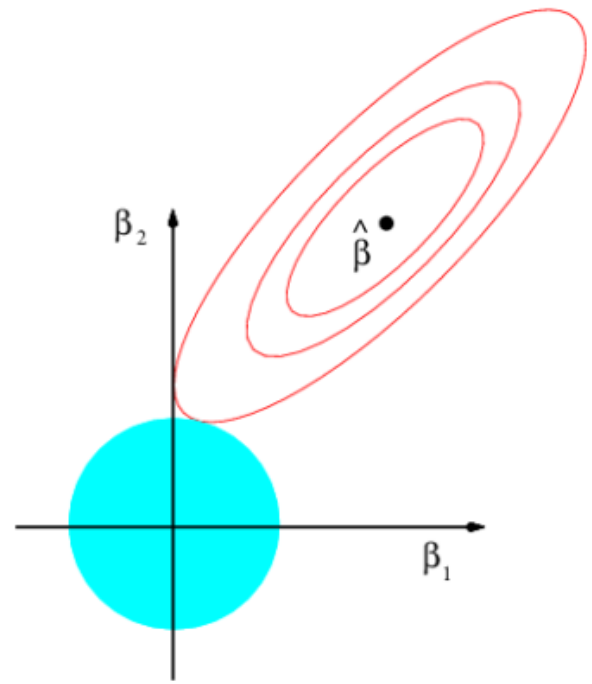


Trees & Forests

$$\underset{\beta}{\text{minimize}} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \right\} \quad \text{subject to} \quad \sum_{j=1}^{p} |\beta_j| \leq s$$

$$\underset{\beta}{\text{minimize}} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \right\} \quad \text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 \leq s,$$





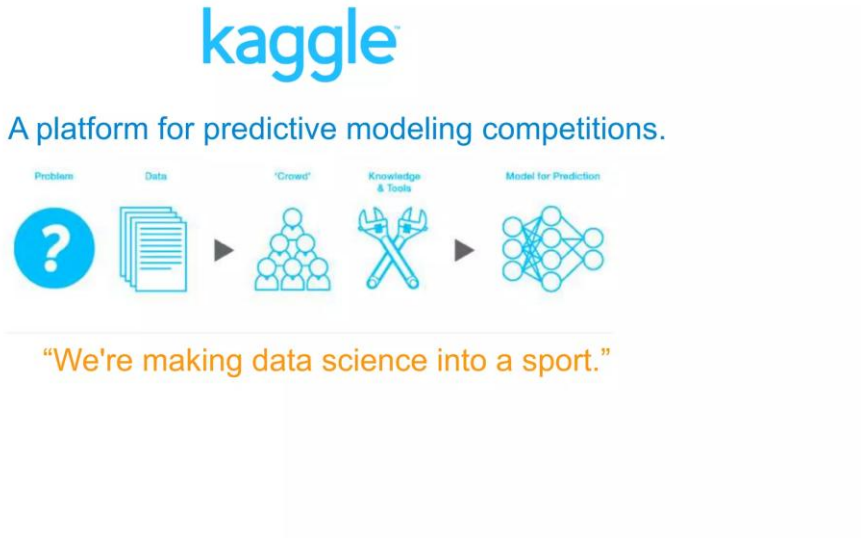
Decision Trees

- **widely used** algorithms for supervised machine learning.
- **nonlinear.**
- **ease of interpretation.**
- large **range of applications** (regression, classification problems including multiple classes, good with categorical and continuous data).

Decision trees are **simple and useful for interpretation**.

However, they typically are **not competitive** with the best supervised learning approaches in terms of prediction accuracy.

Extensions of this idea like bagging, random forests, and boosting are **extremely competitive.**



"we find the strong performance of gradient boosted decision trees"

https://arxiv.org/ftp/arxiv/papers/2009/2009.07701.pdf

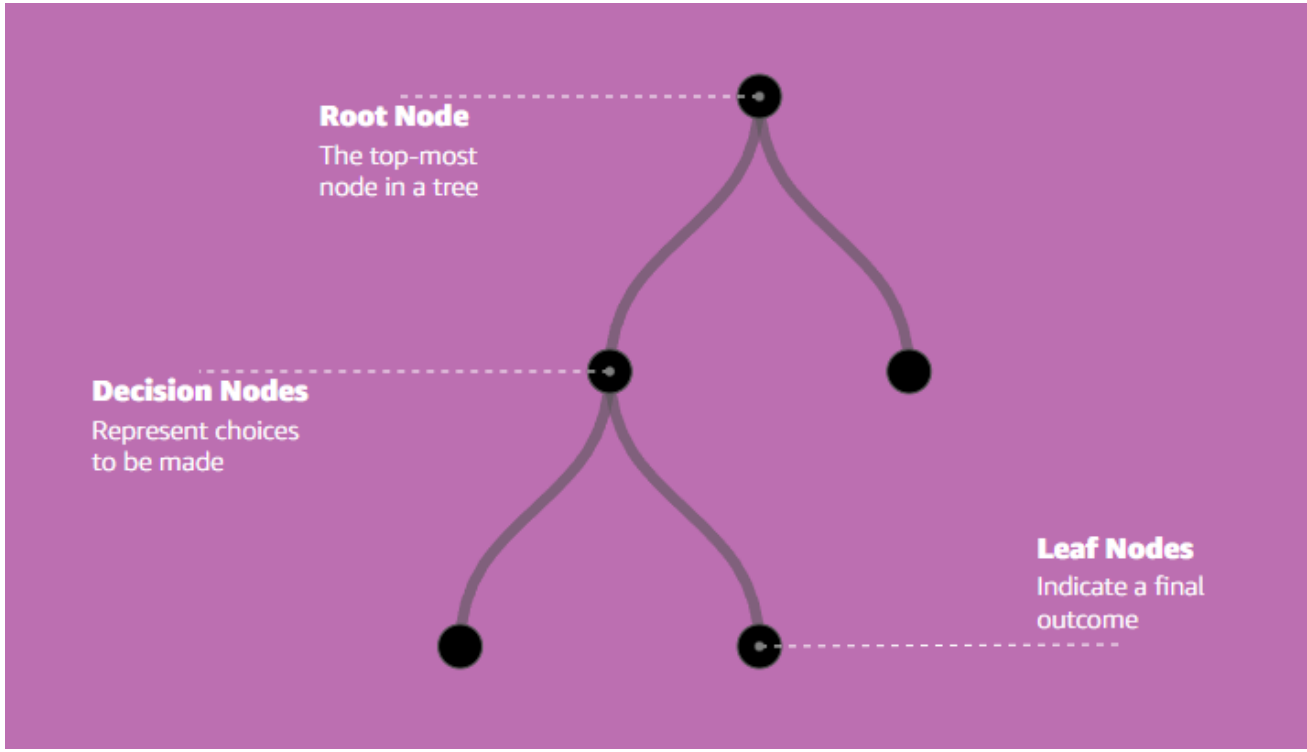
Tree based methods **strongly benefit from a good understanding of when and how to use them** – which requires a good understanding of how they work.

Warning – For forecasting our tree based methods can look amazing on the test dataset and then perform terribly in practice, why?

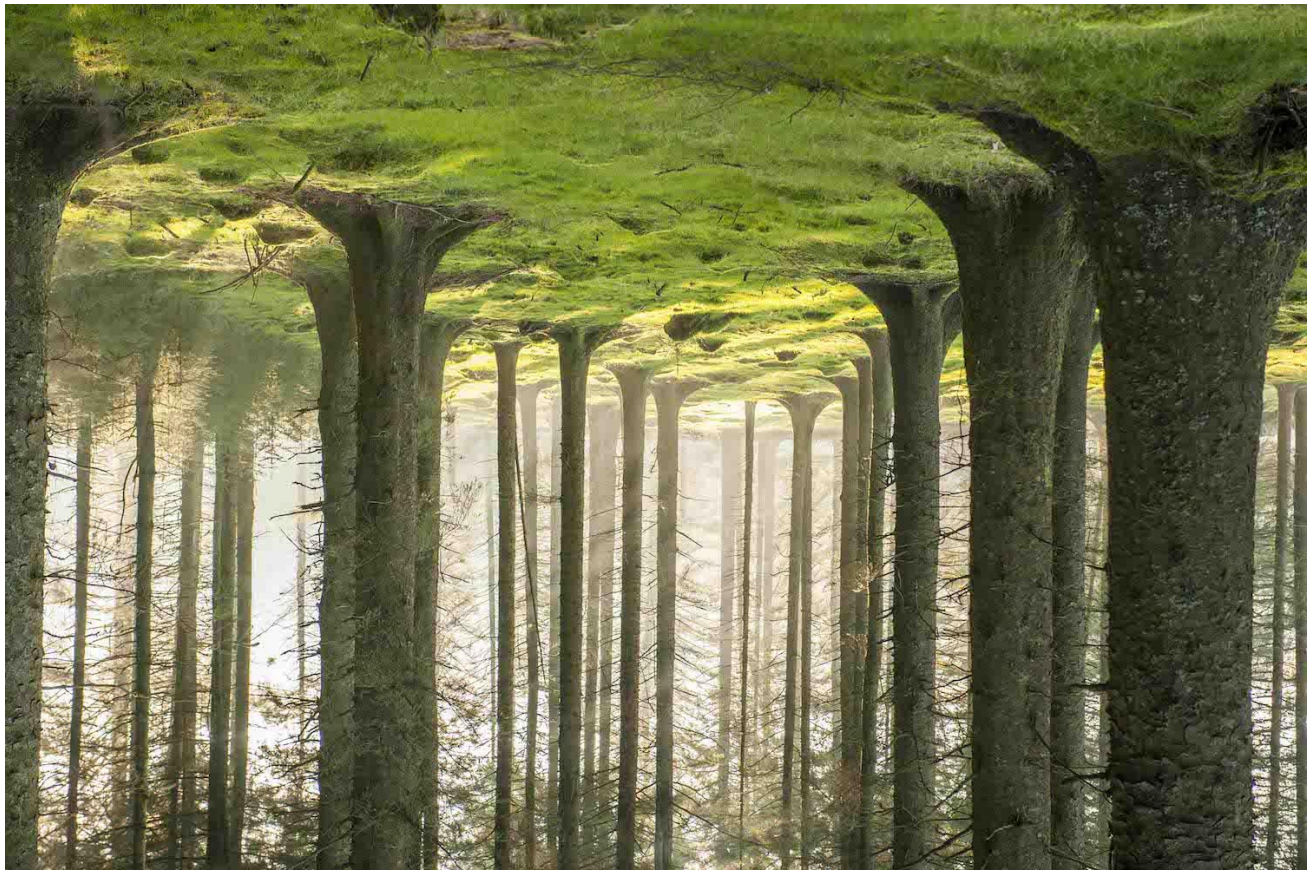
We can apply to both classification and regression.

A decision tree builds an (upside-down) tree structure.

We will look at decision tree to decide if someone is low or high risk of heart attack.



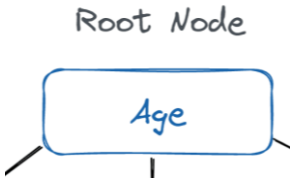
Understanding the risks to prevent a heart attack.

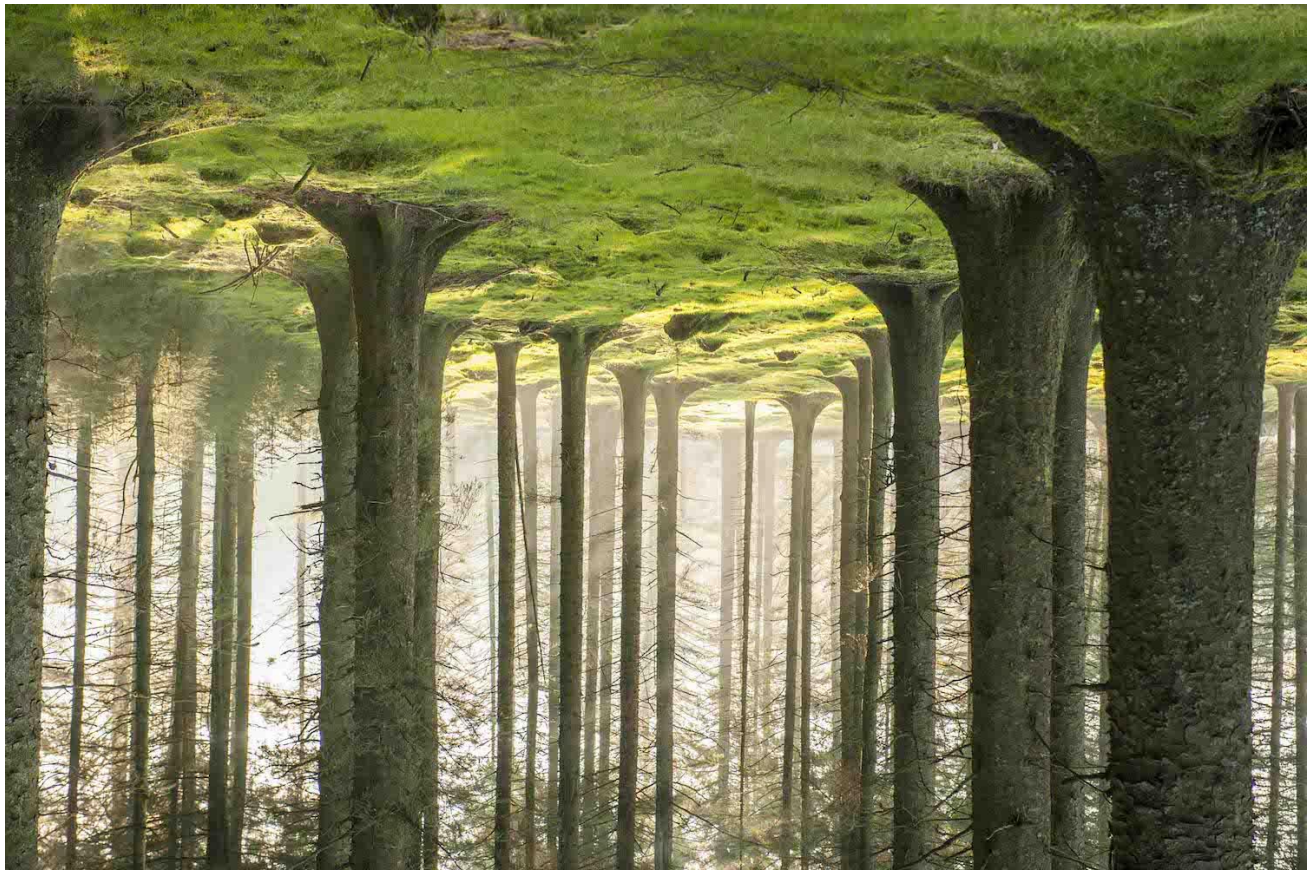


We draw our trees upside down

https://www.datacamp.com/tutorial/decision-tree-classification-python

Understanding the risks to prevent a heart attack.

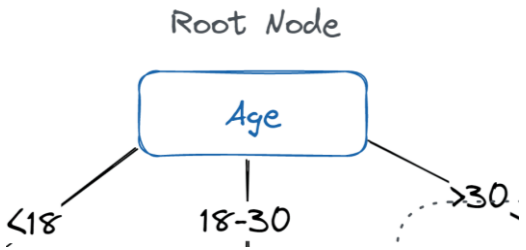


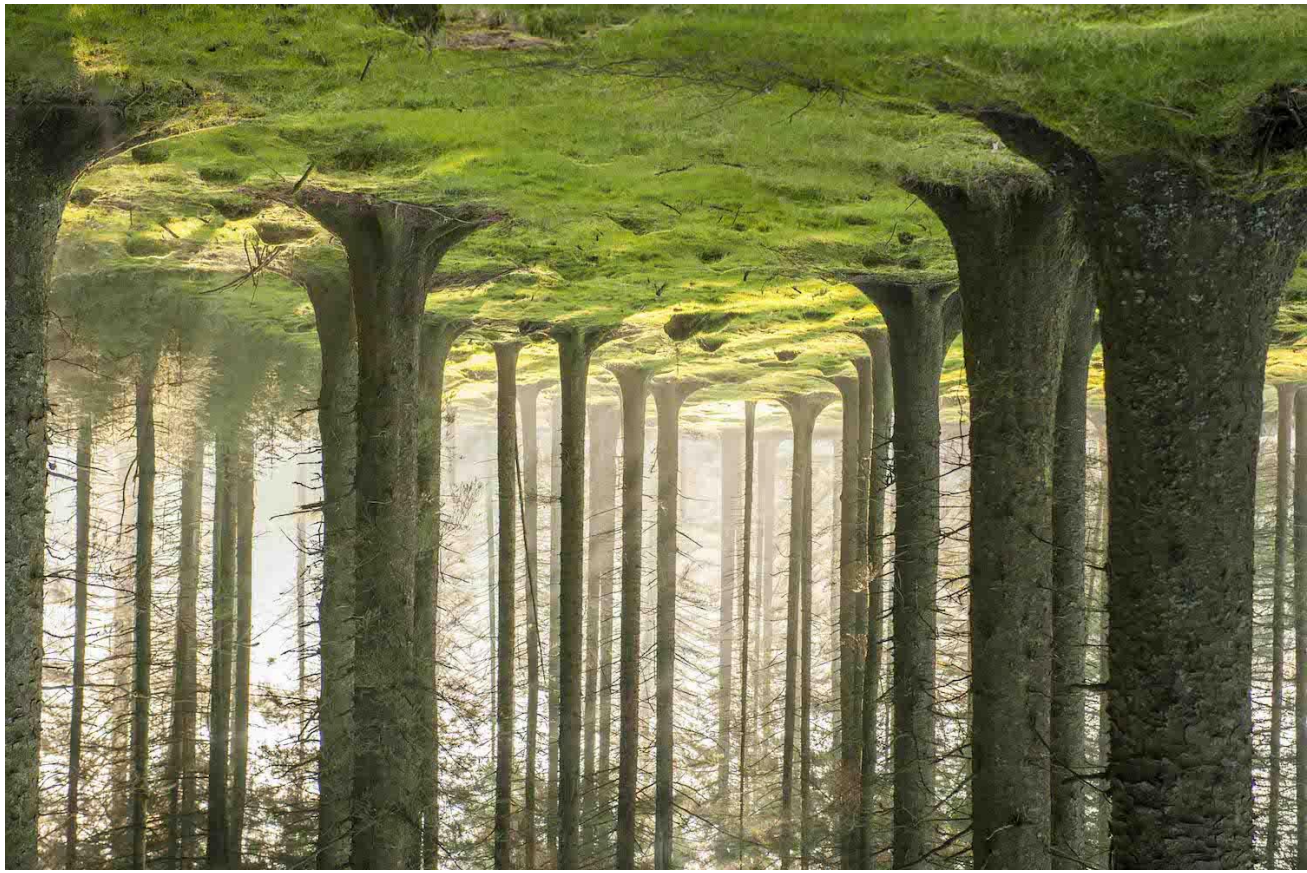


We draw our trees upside down

https://www.datacamp.com/tutorial/decision-tree-classification-python

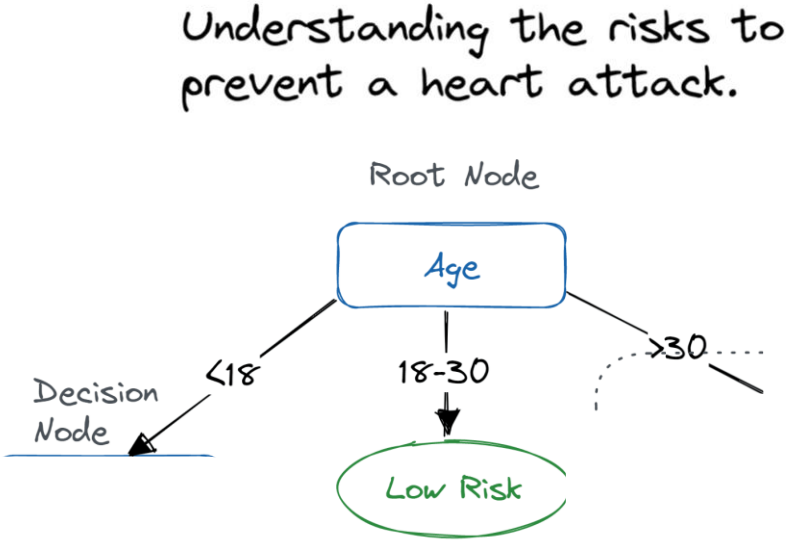
Understanding the risks to prevent a heart attack.

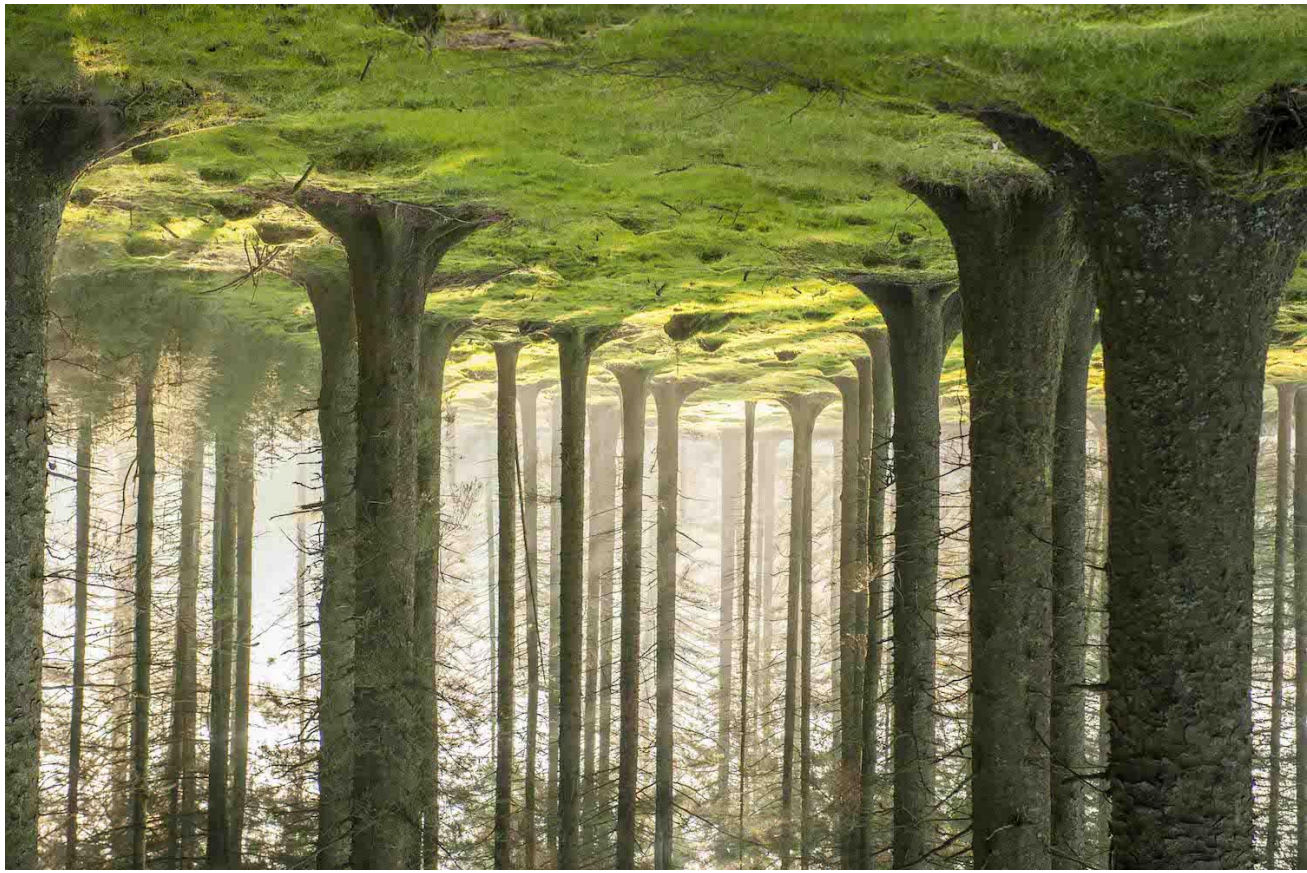




We draw our trees upside down

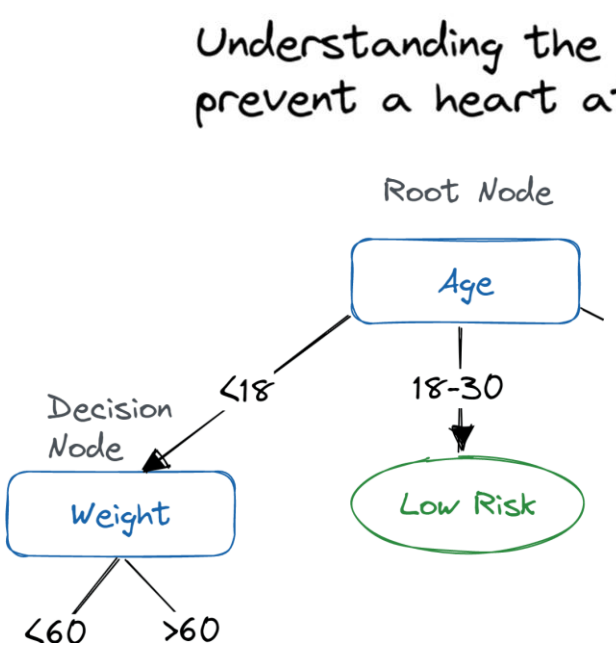
https://www.datacamp.com/tutorial/decision-tree-classification-python

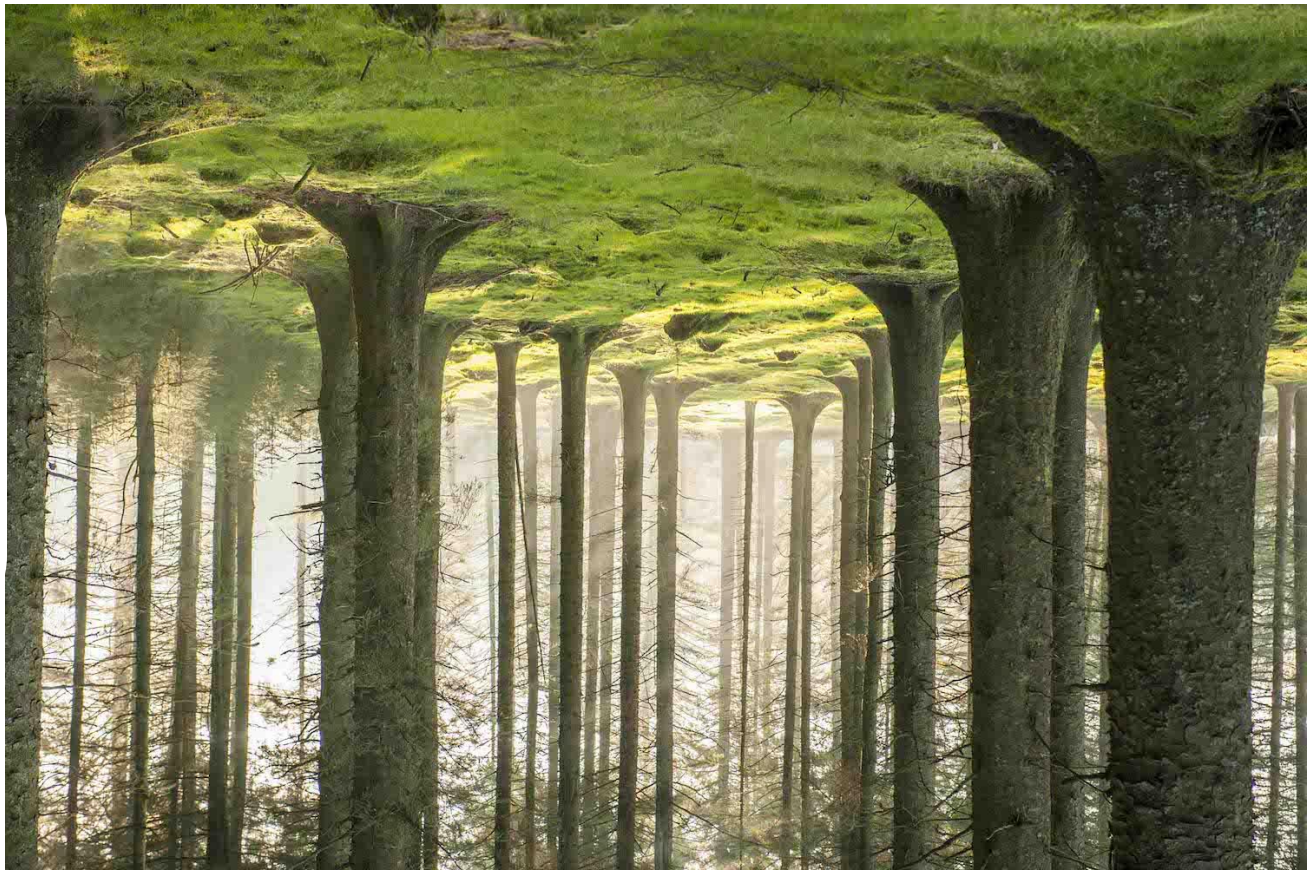




We draw our trees upside down

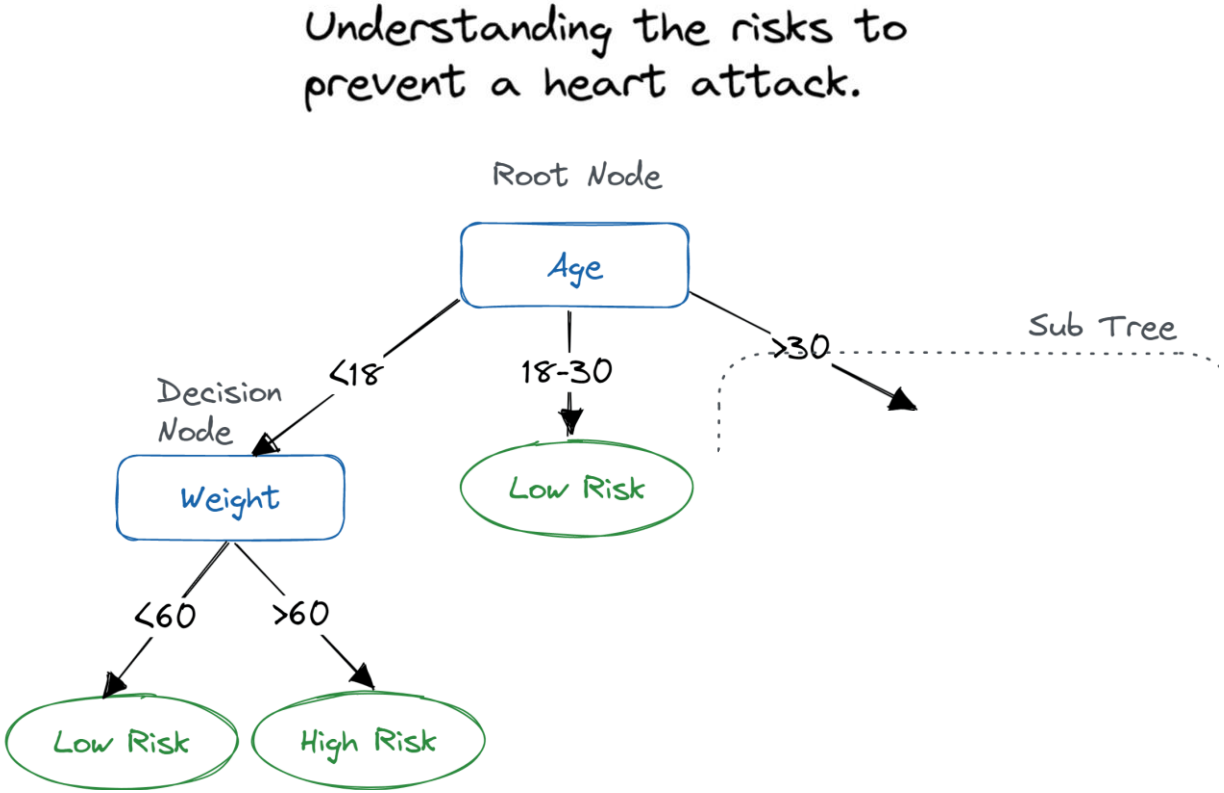
https://www.datacamp.com/tutorial/decision-tree-classification-python

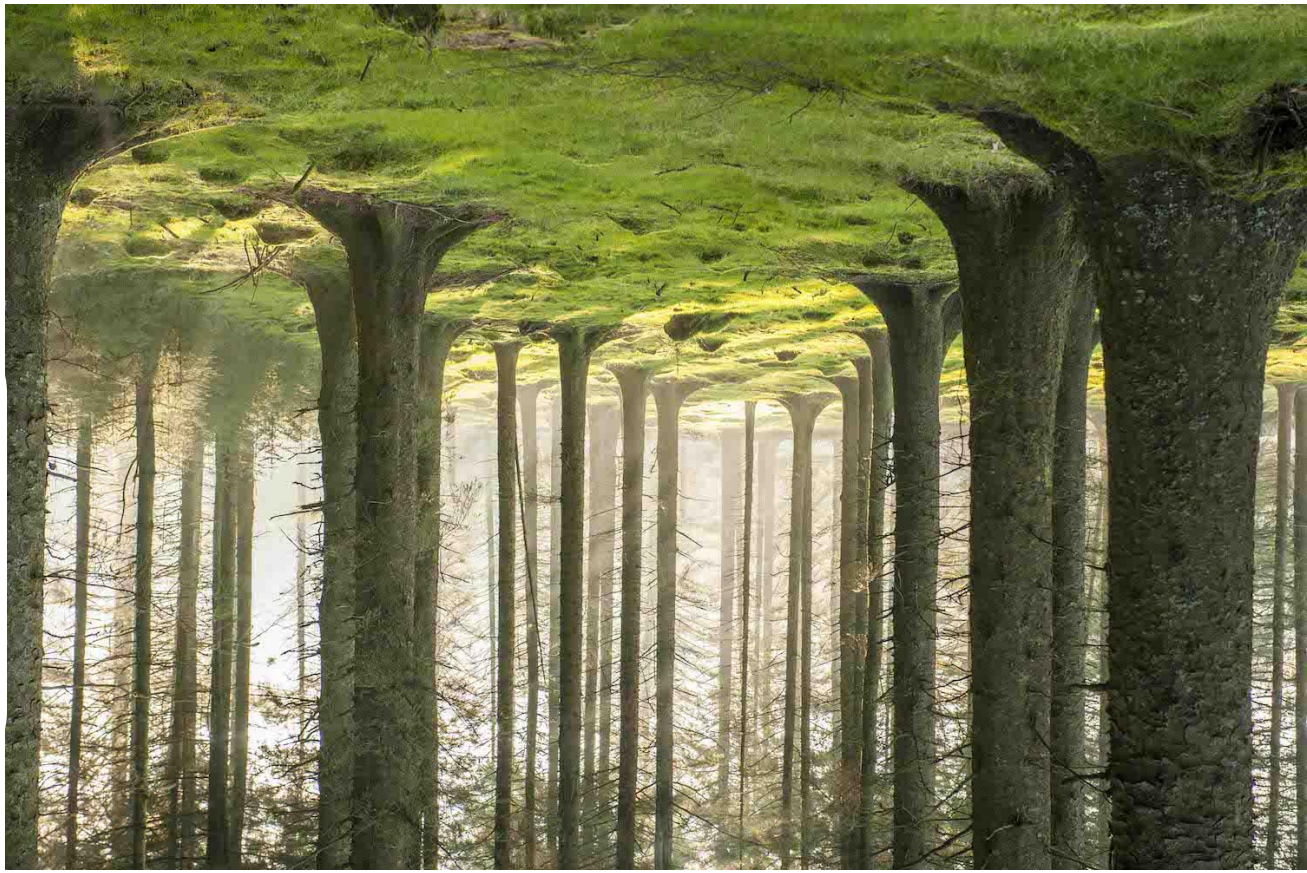




We draw our trees upside down

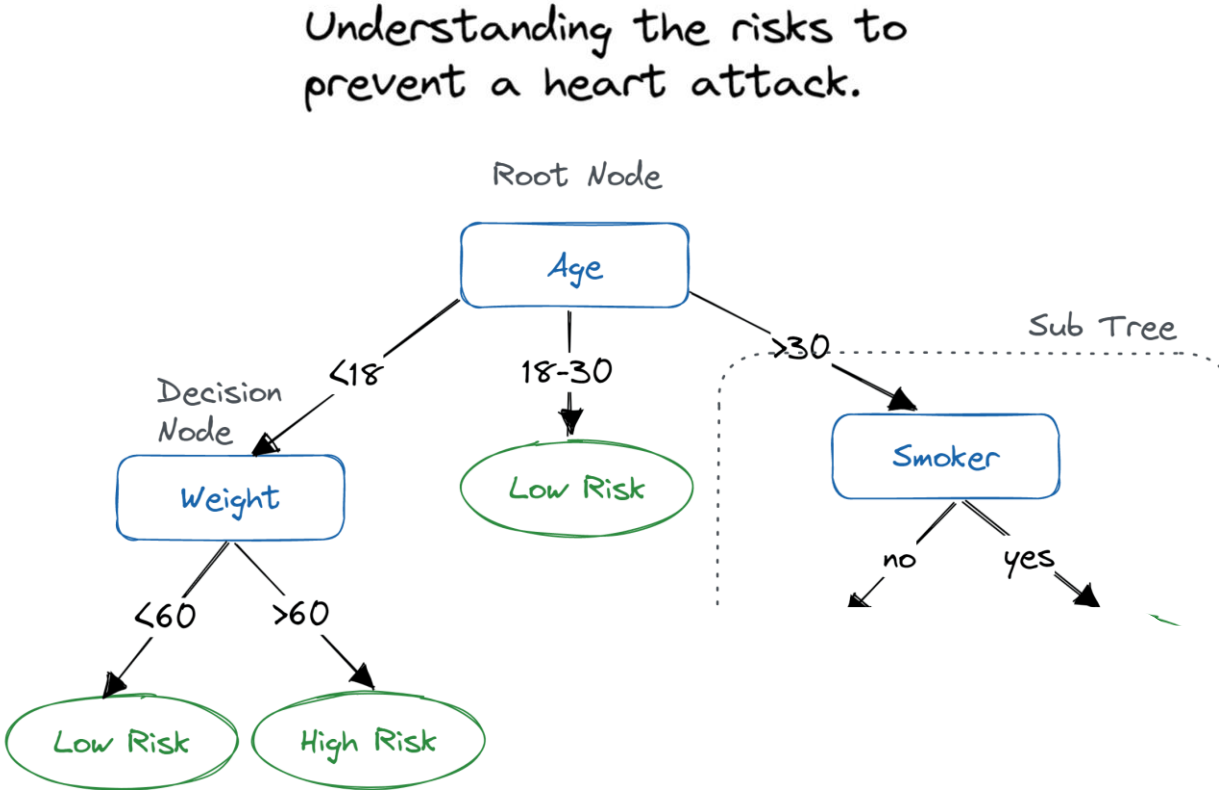
https://www.datacamp.com/tutorial/decision-tree-classification-python

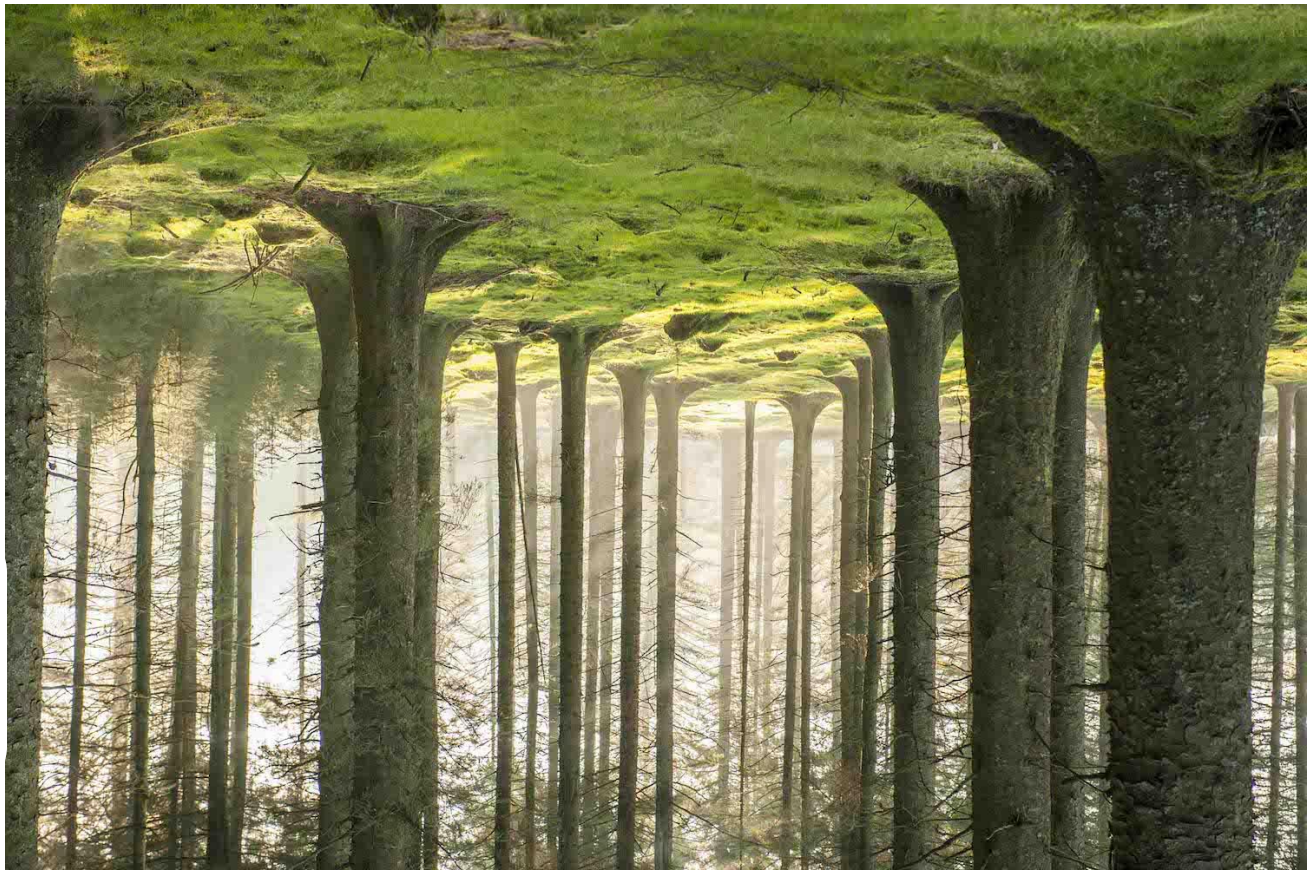




We draw our trees upside down

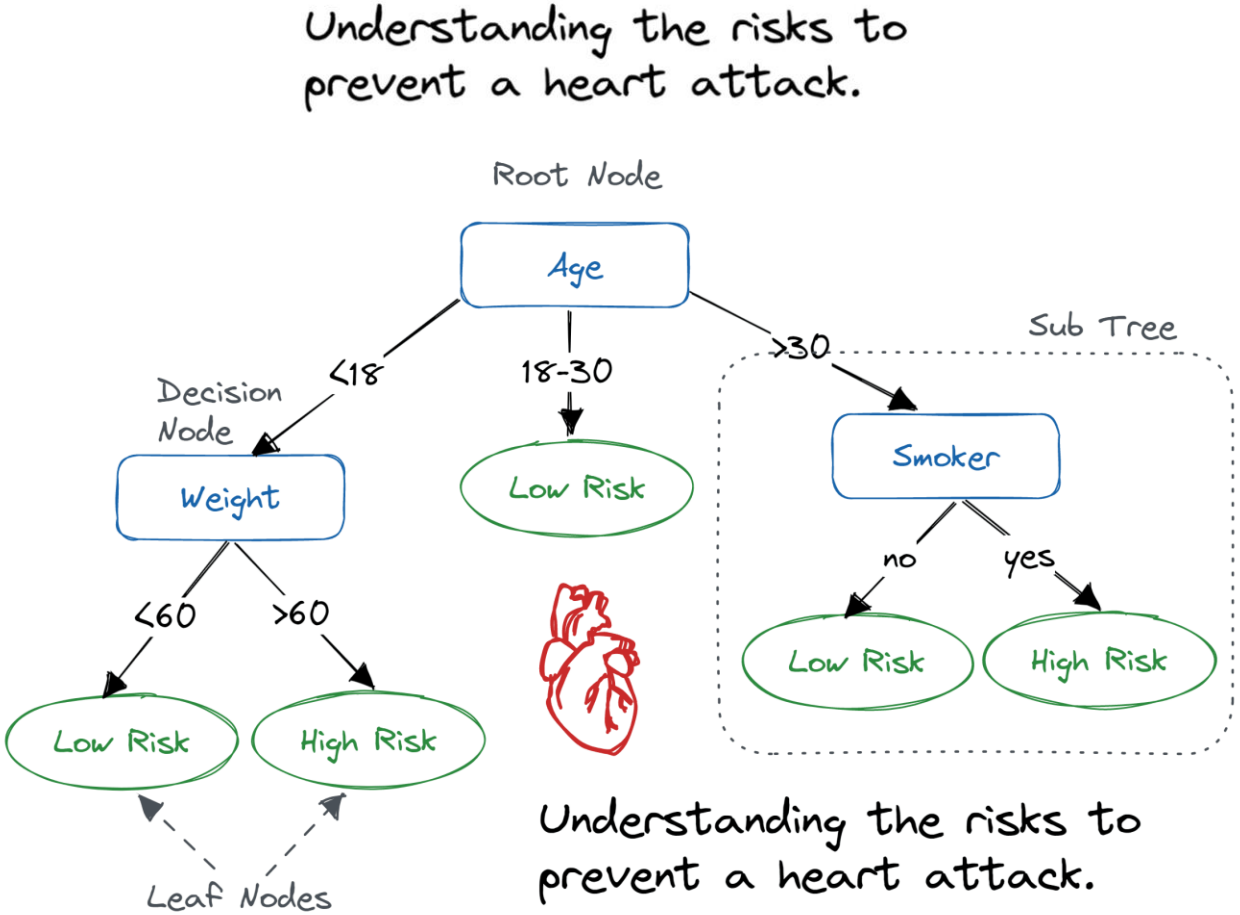
https://www.datacamp.com/tutorial/decision-tree-classification-python

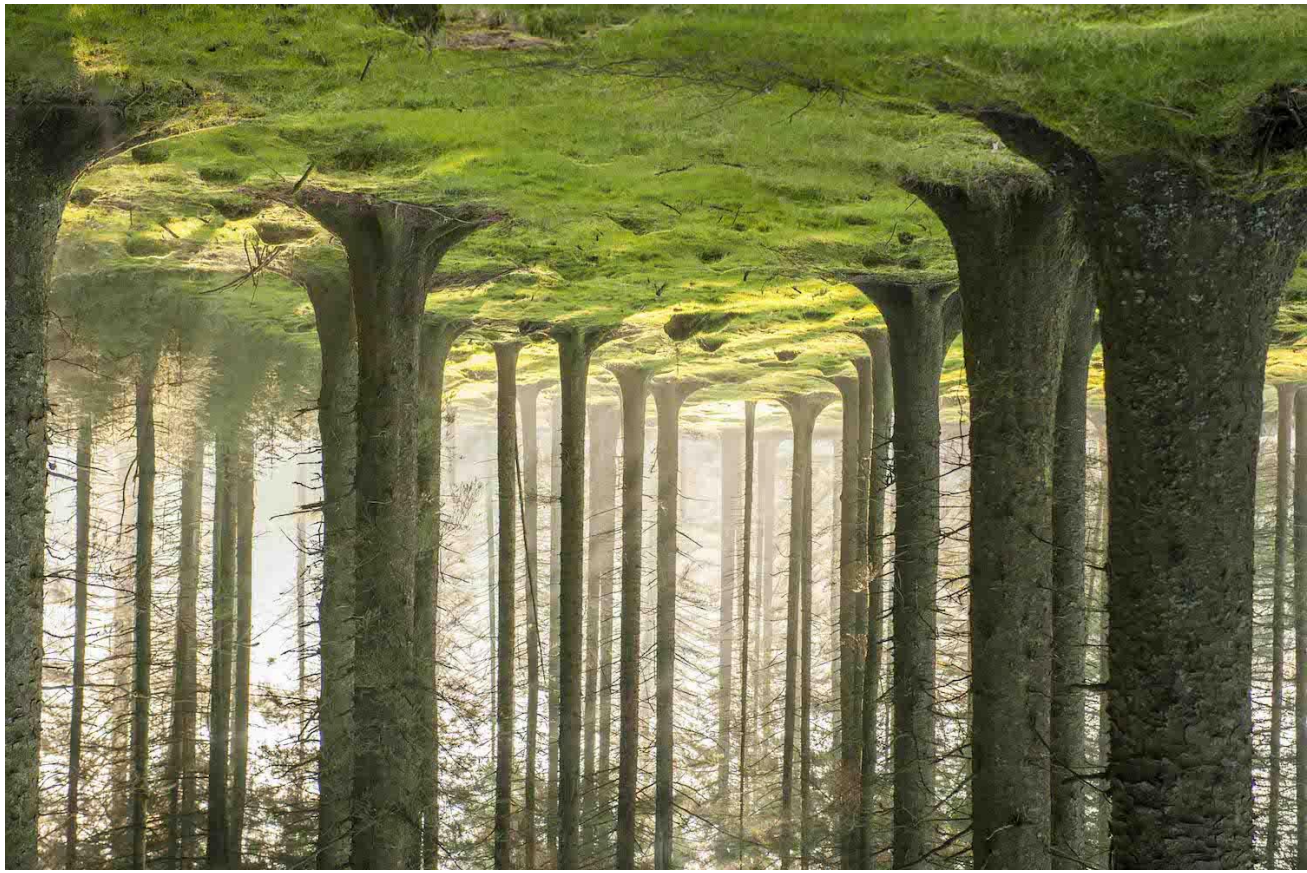




We draw our trees upside down

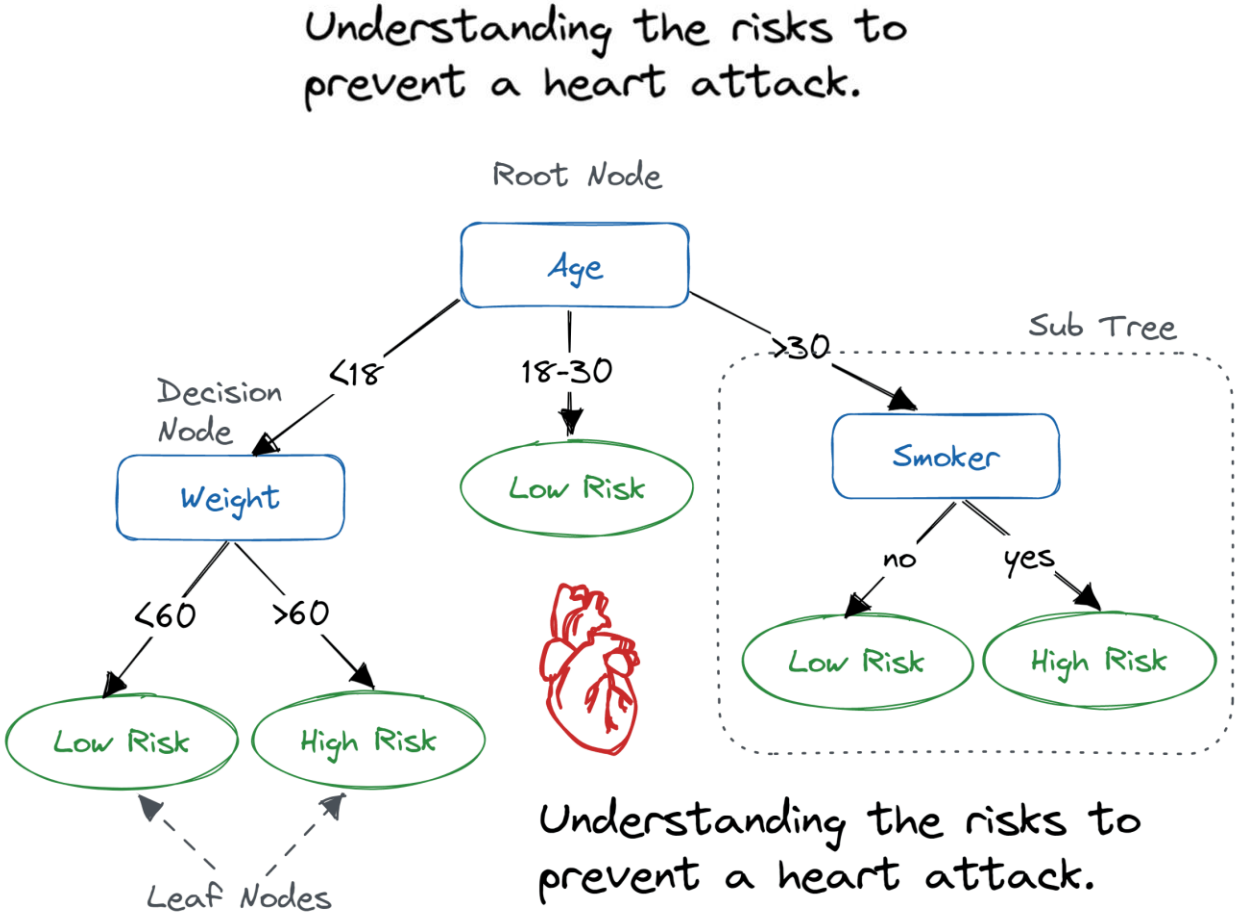
https://www.datacamp.com/tutorial/decision-tree-classification-python





We draw our trees upside down

https://www.datacamp.com/tutorial/decision-tree-classification-python



- Combining variables (interactions)
- Non-linear
- Easy to understand

We draw our trees upside down

https://www.datacamp.com/tutorial/decision-tree-classification-python

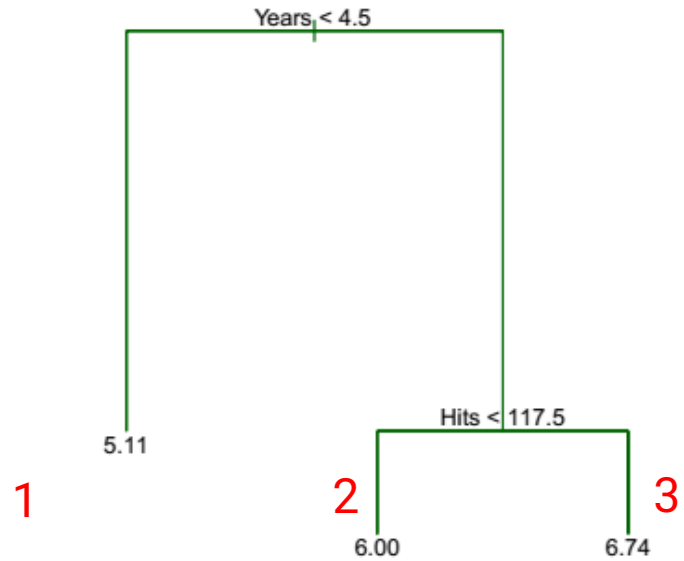


FIGURE 8.1. *For the* `Hitters` *data, a regression tree for predicting the log salary of a baseball player, based on the number of years that he has played in the major leagues and the number of hits that he made in the previous year. At a given internal node, the label (of the form* $X_j < t_k$*) indicates the left-hand branch emanating from that split, and the right-hand branch corresponds to* $X_j \geq t_k$*. For instance, the split at the top of the tree results in two large branches. The left-hand branch corresponds to* `Years<4.5`*, and the right-hand branch corresponds to* `Years>=4.5`*. The tree has two internal nodes and three terminal nodes, or leaves. The number in each leaf is the mean of the response for the observations that fall there.*

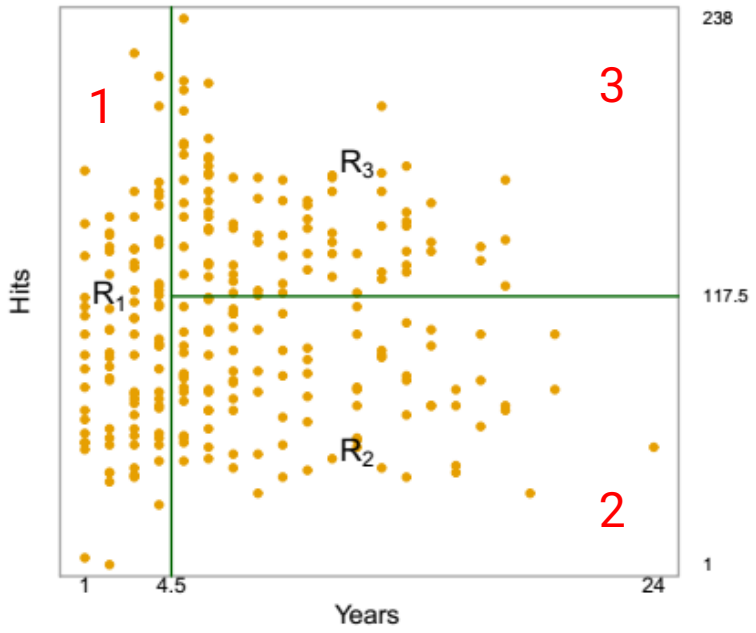


FIGURE 8.2. *The three-region partition for the* `Hitters` *data set from the regression tree illustrated in Figure 8.1.*

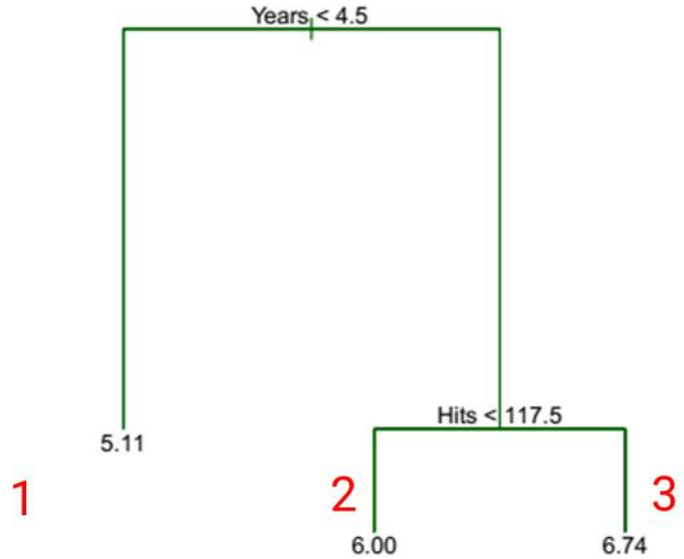


FIGURE 8.1. *For the* `Hitters` *data, a regression tree for predicting the log salary of a baseball player, based on the number of years that he has played in the major leagues and the number of hits that he made in the previous year. At a given internal node, the label (of the form $X_j < t_k$) indicates the left-hand branch emanating from that split, and the right-hand branch corresponds to $X_j \geq t_k$. For instance, the split at the top of the tree results in two large branches. The left-hand branch corresponds to* `Years<4.5`*, and the right-hand branch corresponds to* `Years>=4.5`*. The tree has two internal nodes and three terminal nodes, or leaves. The number in each leaf is the mean of the response for the observations that fall there.*

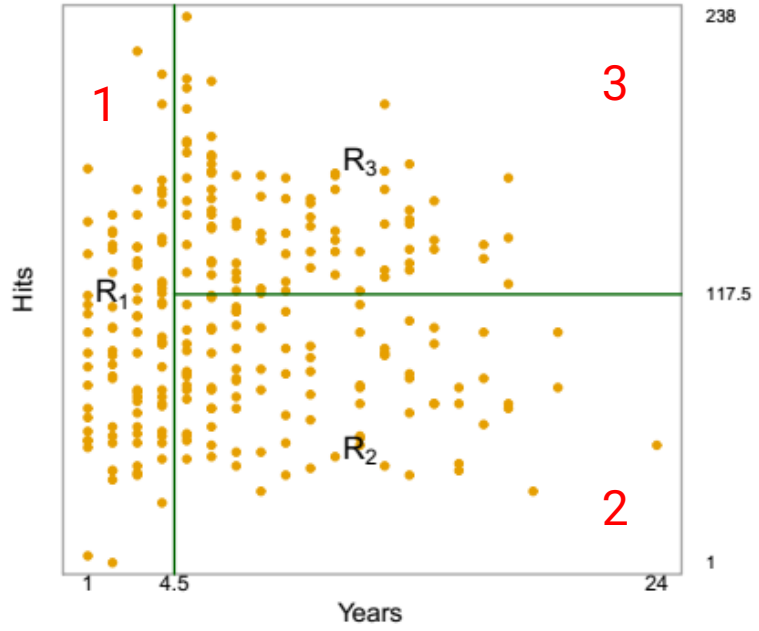


FIGURE 8.2. *The three-region partition for the* `Hitters` *data set from the regression tree illustrated in Figure 8.1.*

- We divide the predictor
- For every observation that falls into the region Rj, we make the same prediction
- The goal is to find boxes that minimize the error

Our regression model only provides 3 different output values. This is continuous but oversimplified.

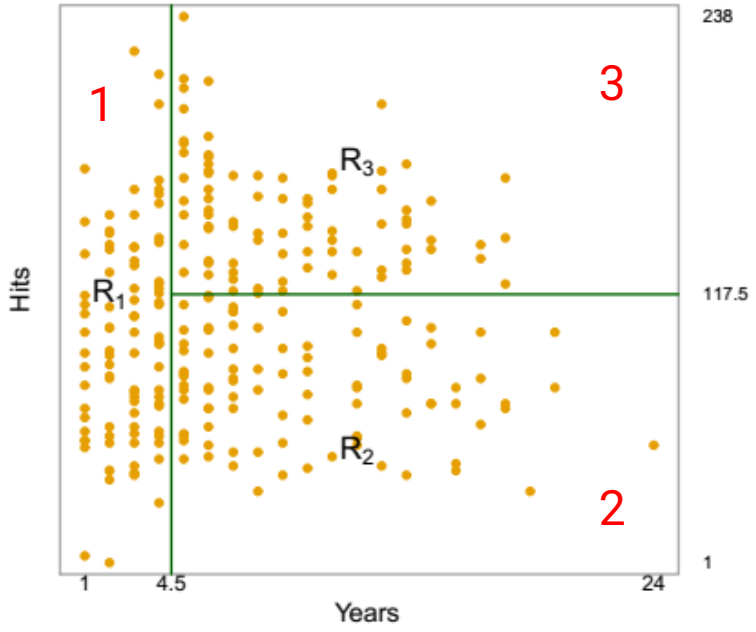


FIGURE 8.2. *The three-region partition for the* `Hitters` *data set from the regression tree illustrated in Figure 8.1.*

- It is computationally infeasible to consider every possible partition of the feature space into J boxes.
- For this reason, we take a top-down, **greedy approach** that is known as recursive binary splitting.
- Greedy algorithms aren't certain to be optimal.

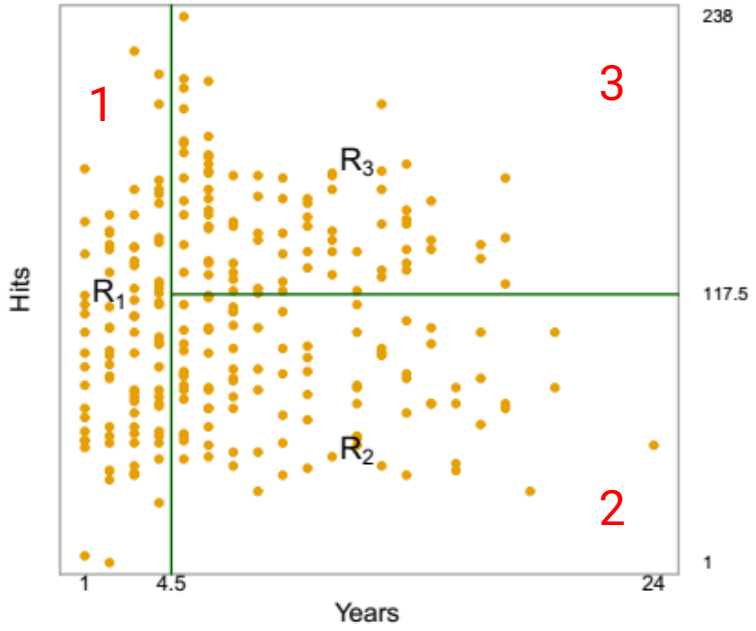
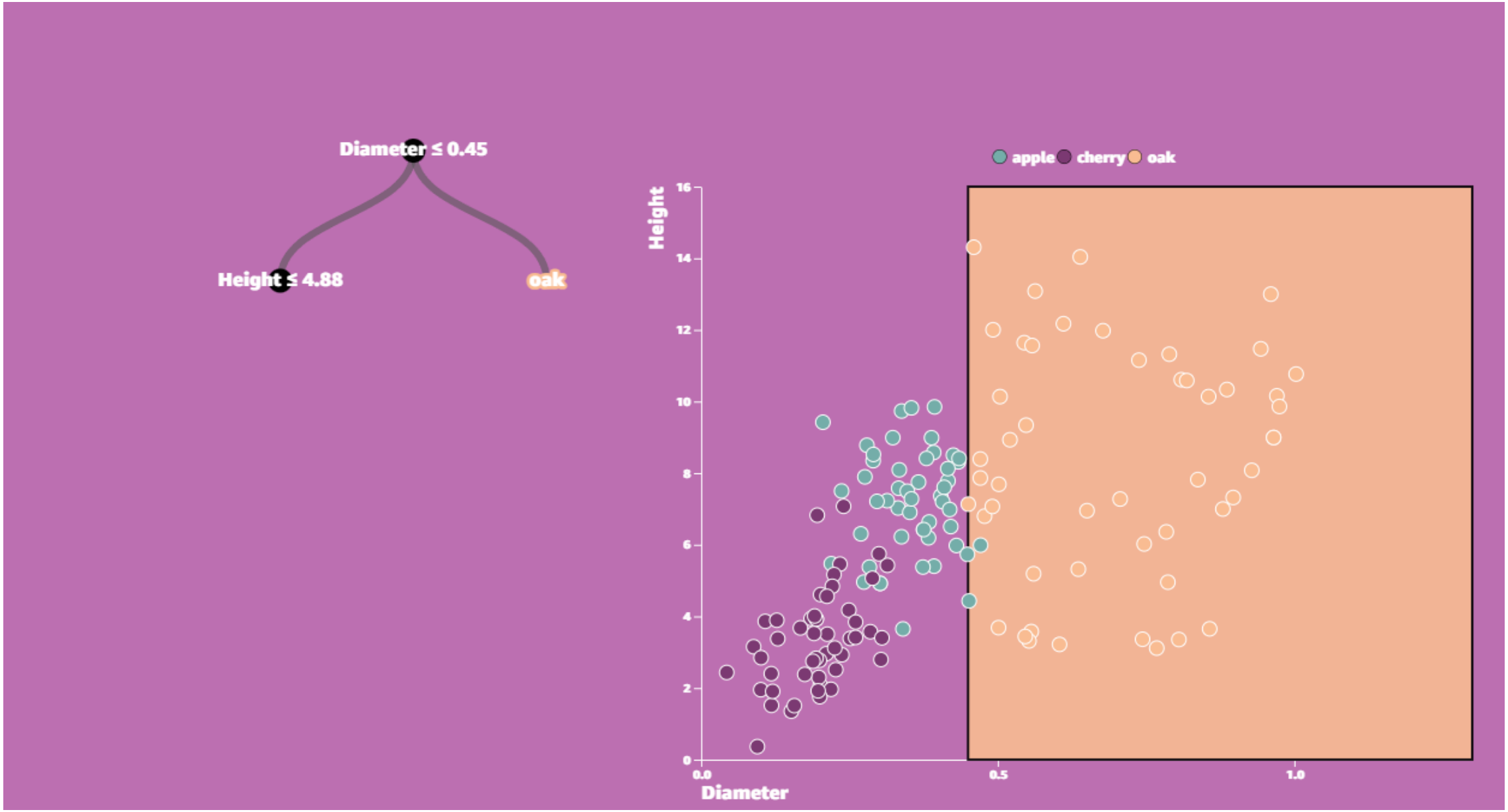
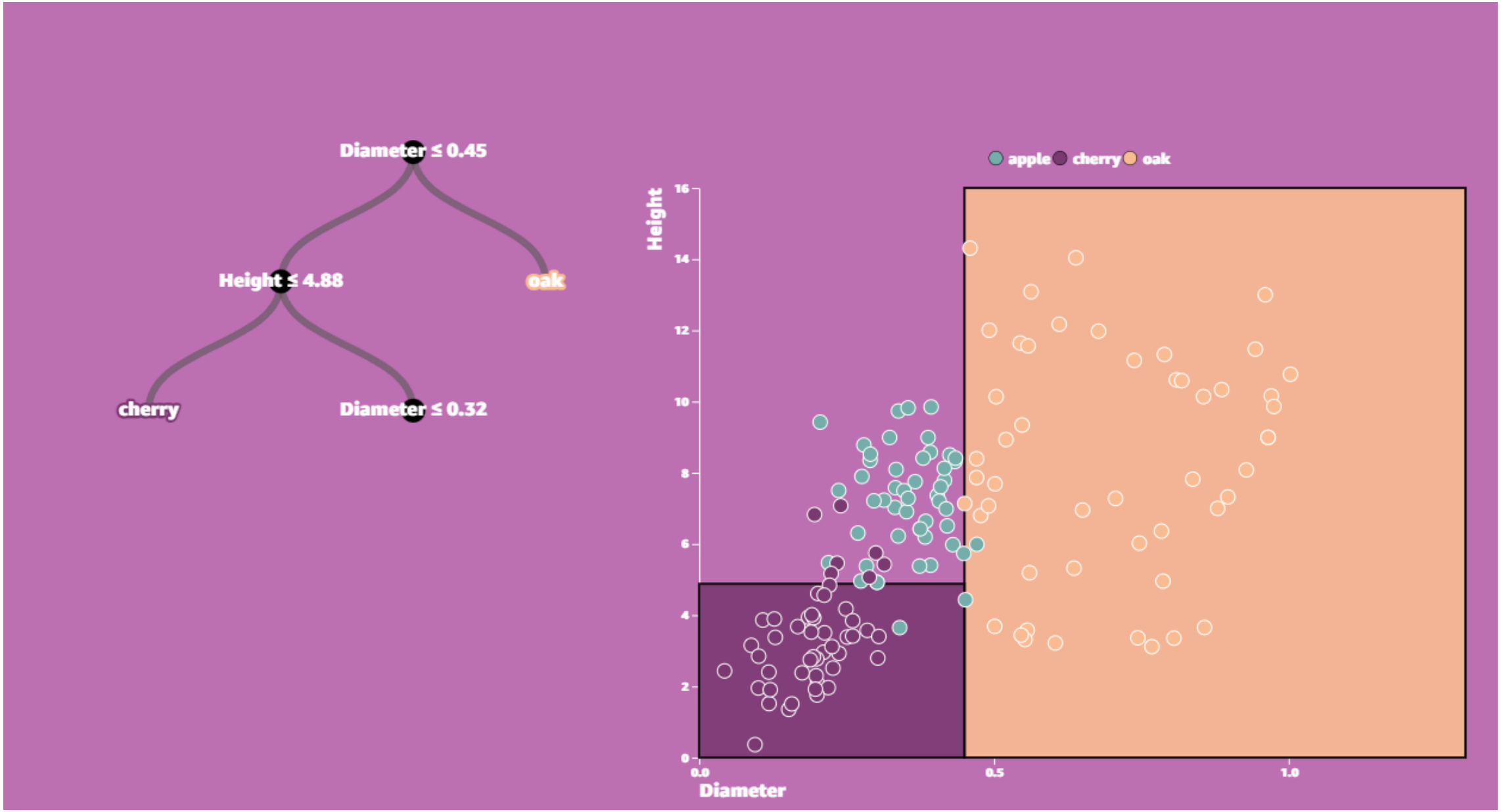


FIGURE 8.2. *The three-region partition for the* `Hitters` *data set from the regression tree illustrated in Figure 8.1.*

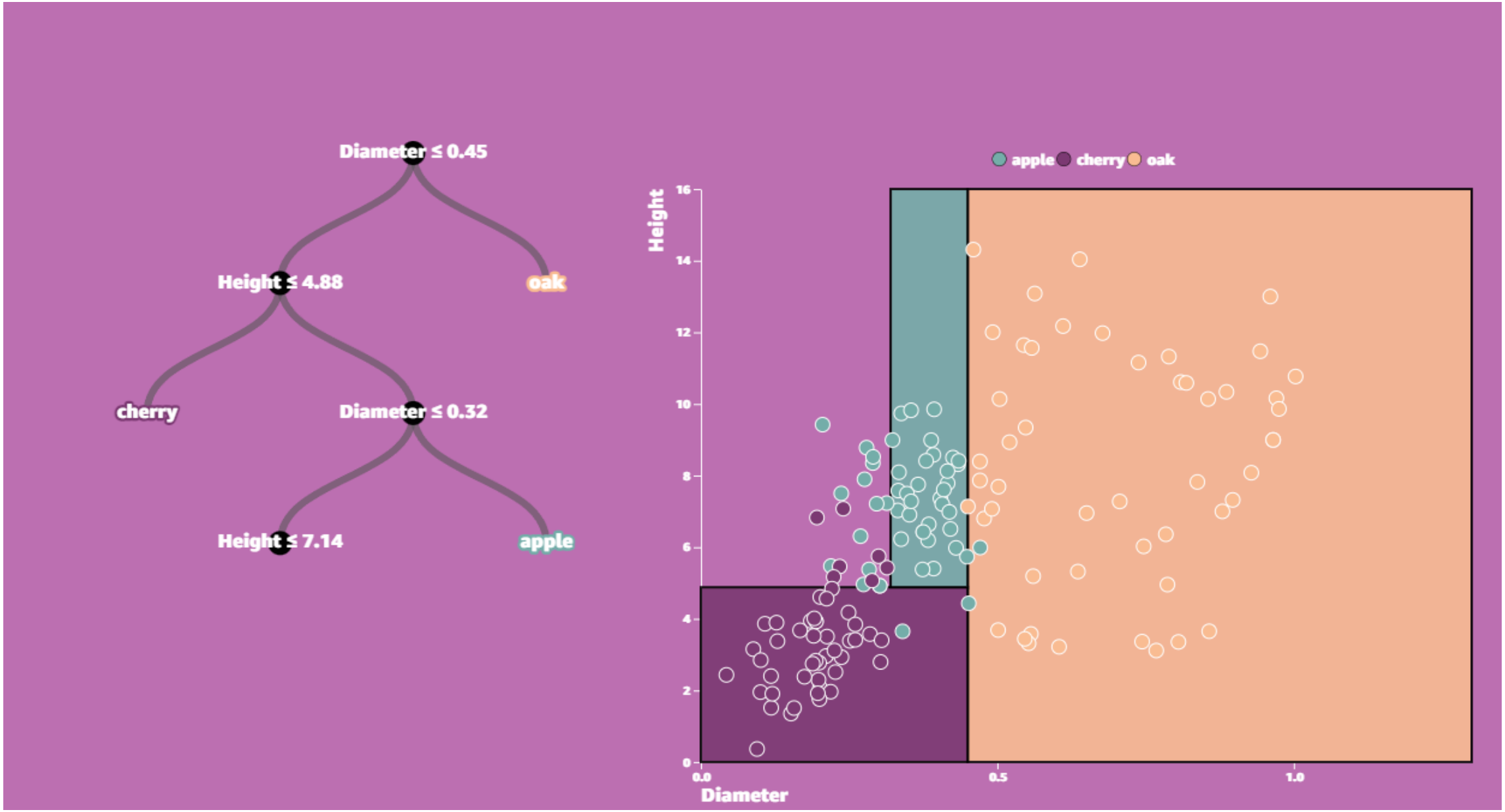
- We select the predictor and the cutpoint s such that splitting the predictor space into regions leads to the greatest possible reduction in RSS.
- Then we repeat - we split **one** of the previously identified regions. We now have three regions.
- Next, again, we look to split one of these three regions further, so as to minimize the RSS.
- The more steps the more regions we have to check, but potentially less variables.



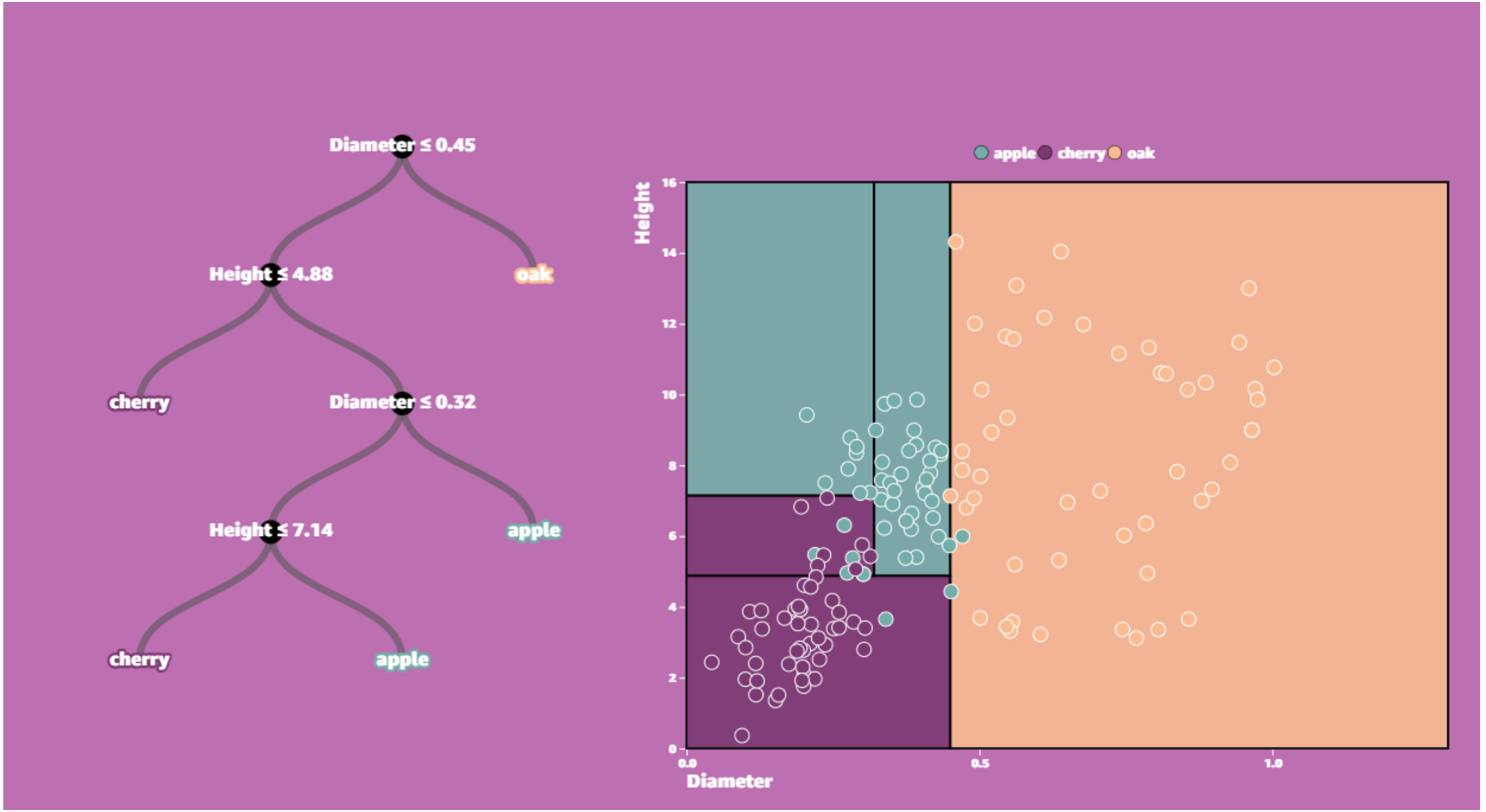
https://mlu-explain.github.io/decision-tree/



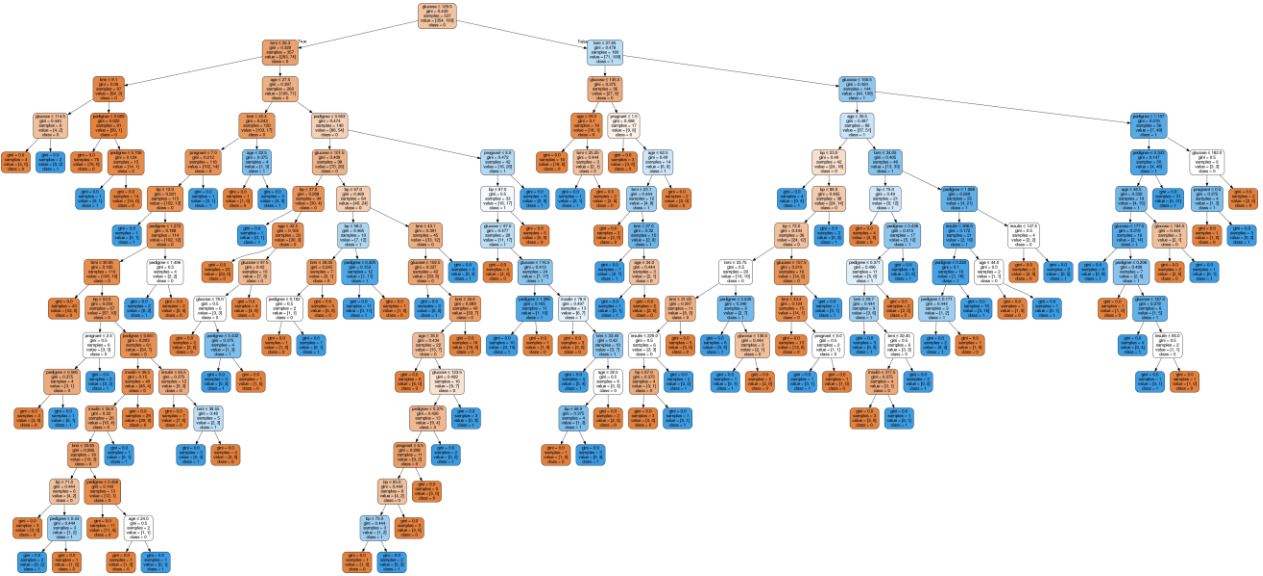
https://mlu-explain.github.io/decision-tree/



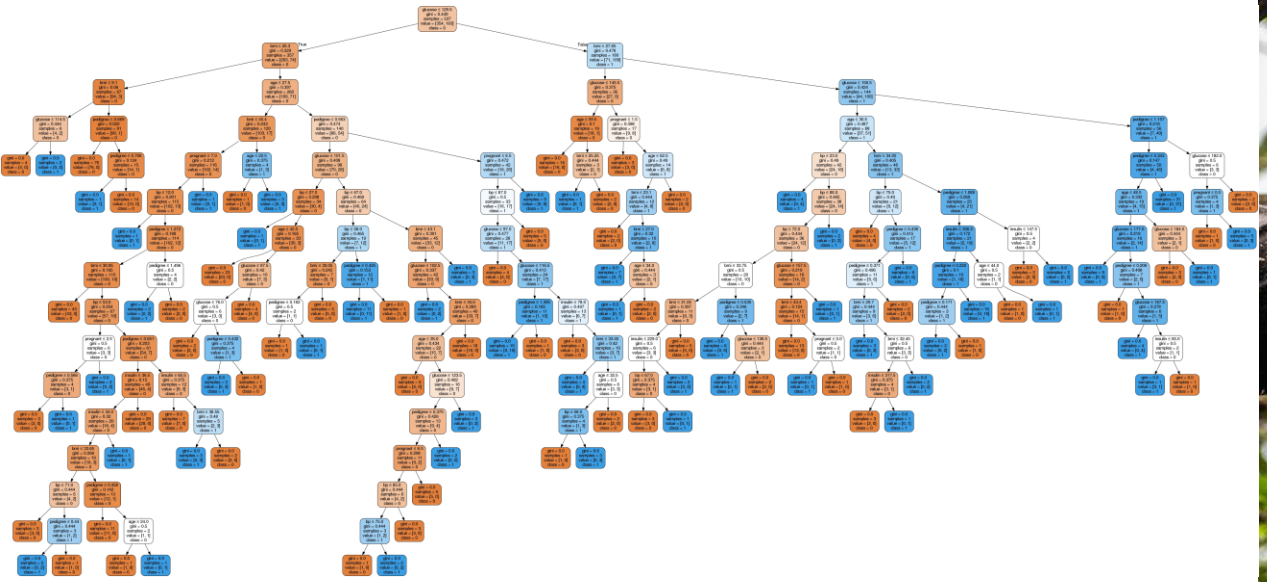
https://mlu-explain.github.io/decision-tree/



https://mlu-explain.github.io/decision-tree/



- Increasing splits will always increase our accuracy on the training data – but we need to worry about **overfitting**.



- This will typically lead to **overfitting**
- In general longer trees are more prone to overfitting

- Decision trees rely on the **combination of decisions – non-linearity**
- so the first split can seem not useful and all the improvement in RSS shows up in the second

- Therefore, the best strategy is to grow a very large tree, and then **prune** it back in order to obtain a **subtree.**
- How do we pick the subtree?
  - Could do CV, but too intensive as many subtrees possible.
  - Instead, we remove the weakest results “pruning”
  - We then combine with CV to assess the best results.

Cost complexity pruning is a specific method of pruning. It introduces a concept called the "cost complexity parameter," often denoted as alpha.

- The idea is to find a good balance between the tree's complexity (depth) and its accuracy.
- A small tree (less complex) that still makes good predictions is generally preferable.
- If alpha is too low, the tree might be too complex (overfit). If it's too high, the tree might become too simple and miss important patterns (underfit).

Cost complexity pruning

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

|T| indicates the number of terminal nodes of the tree T
Rm is the rectangle (i.e. the subset of predictor space) corresponding to the mth terminal node, and
y_{Rm} is the predicted response associated with Rm—that is, the mean of the training observations in Rm.
a controls a trade-off between the subtree's complexity and its fit to the training data.

- a = 0, then the subtree T will simply equal the full tree
- as a increases, there is a price to pay for having a tree with many terminal nodes, and so the quantity will tend to be minimized in a predictable fashion – similar to lasso.

Algorithm 8.1 *Building a Regression Tree*

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.

2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α.

3. Use K-fold cross-validation to choose α. That is, divide the training observations into K folds. For each $k = 1, \ldots, K$:

   (a) Repeat Steps 1 and 2 on all but the kth fold of the training data.

   (b) Evaluate the mean squared prediction error on the data in the left-out kth fold, as a function of α.

   Average the results for each value of α, and pick α to minimize the average error.

4. Return the subtree from Step 2 that corresponds to the chosen value of α.

Algorithm 8.1 *Building a Regression Tree*

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.

2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α.

3. Use K-fold cross-validation to choose α. That is, divide the training observations into K folds. For each $k = 1, \ldots, K$:

   (a) Repeat Steps 1 and 2 on all but the kth fold of the training data.

   (b) Evaluate the mean squared prediction error on the data in the left-out kth fold, as a function of α.

   Average the results for each value of α, and pick α to minimize the average error.

4. Return the subtree from Step 2 that corresponds to the chosen value of α.

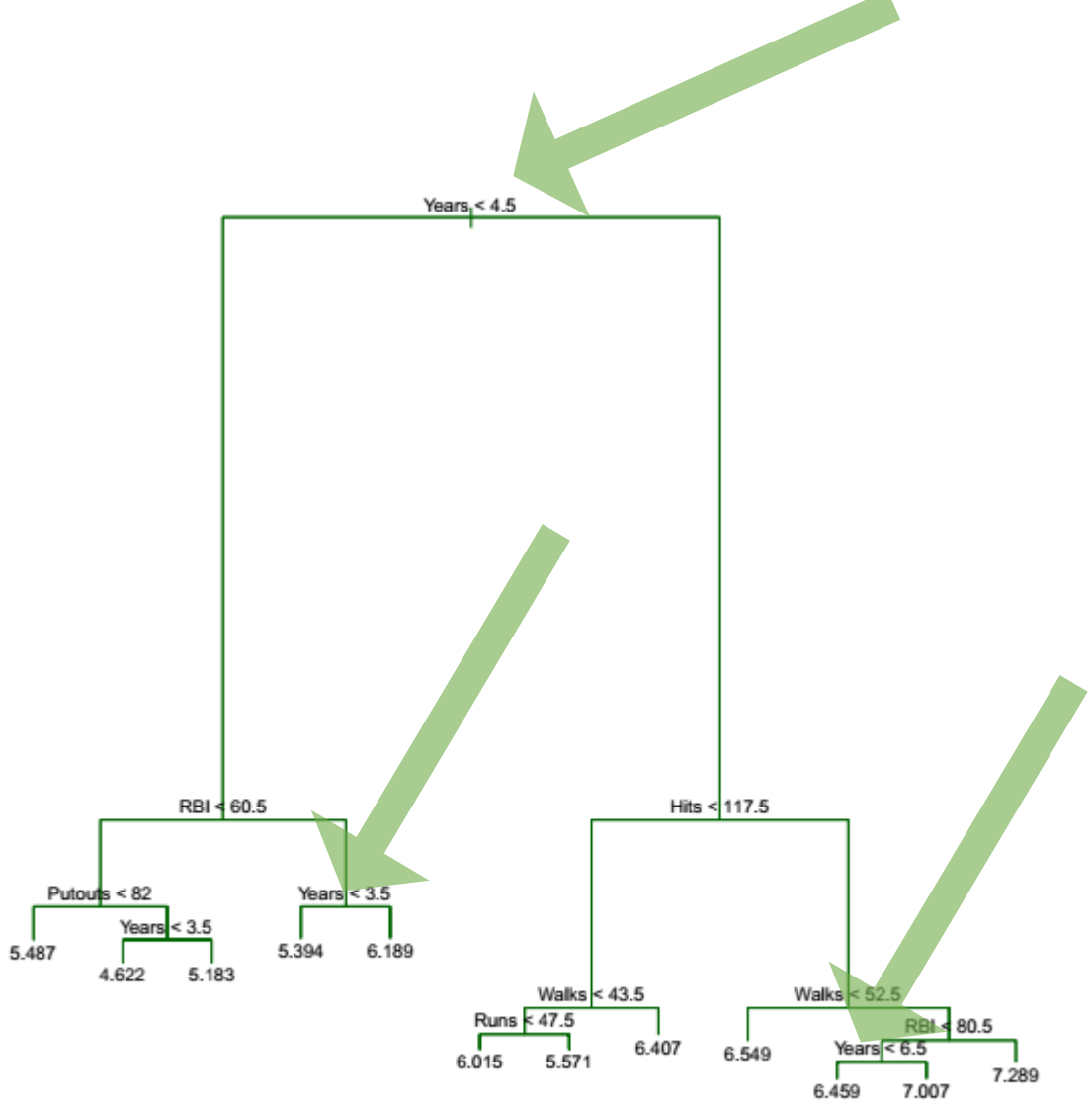


FIGURE 8.4. *Regression tree analysis for the* `Hitters` *data. The unpruned tree that results from top-down greedy splitting on the training data is shown.*

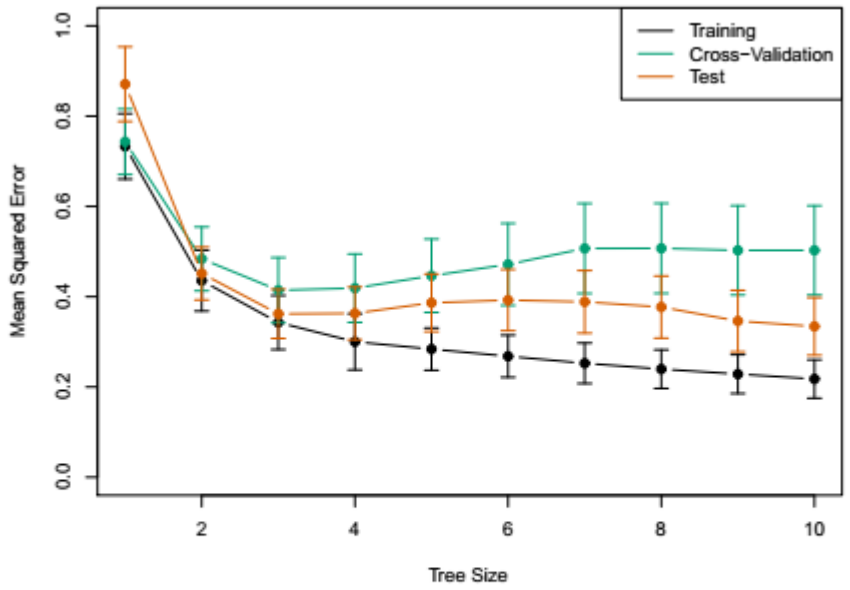


FIGURE 8.5. *Regression tree analysis for the* `Hitters` *data. The training, cross-validation, and test MSE are shown as a function of the number of terminal nodes in the pruned tree. Standard error bands are displayed. The minimum cross-validation error occurs at a tree size of three.*

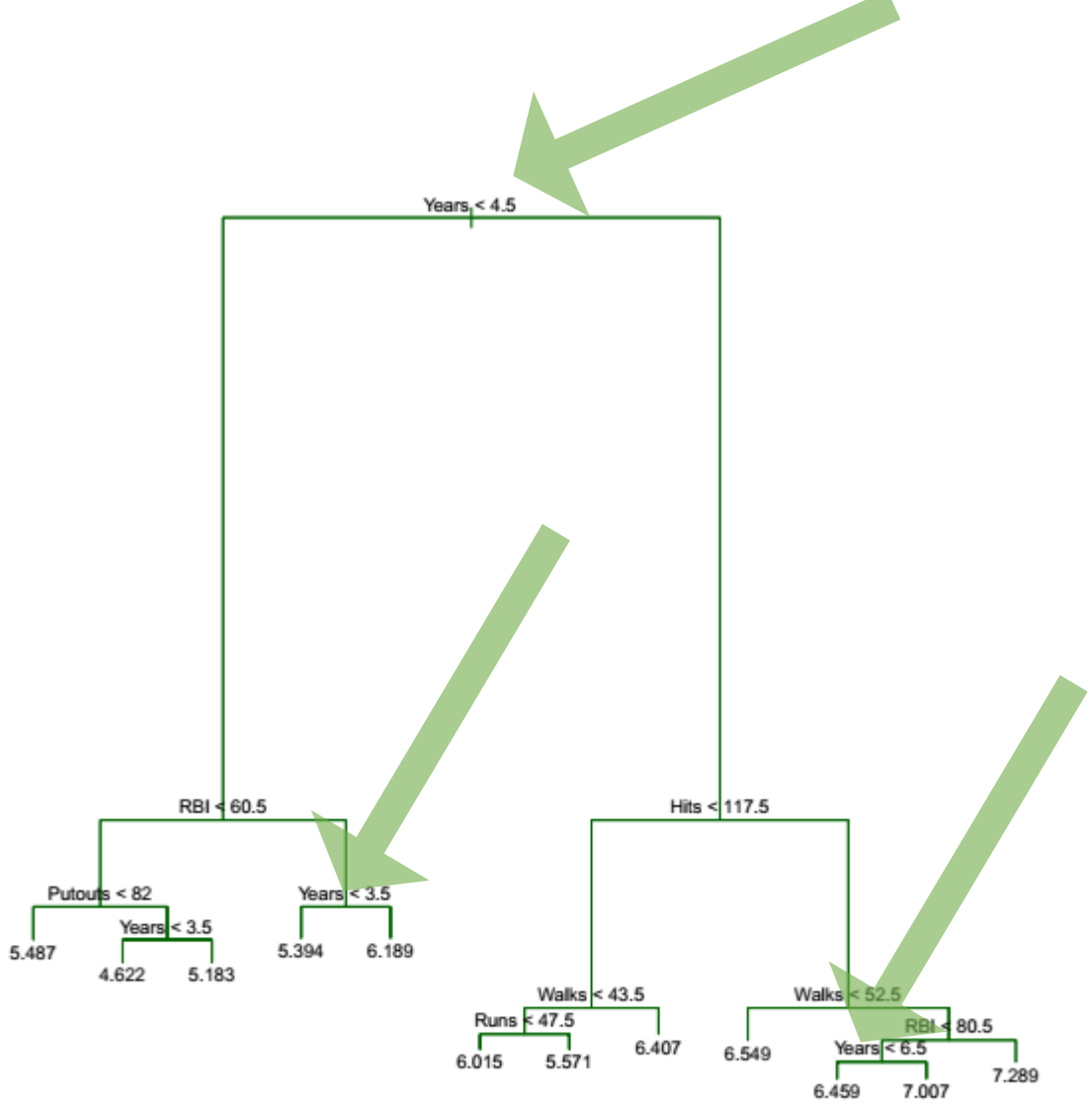


FIGURE 8.4. *Regression tree analysis for the* `Hitters` *data. The unpruned tree that results from top-down greedy splitting on the training data is shown.*

Classification we have different metrics to optimise:

- Gini index
- Entropy

The **classification error rate is not sensitive enough**, we can't grow trees well with this approach.

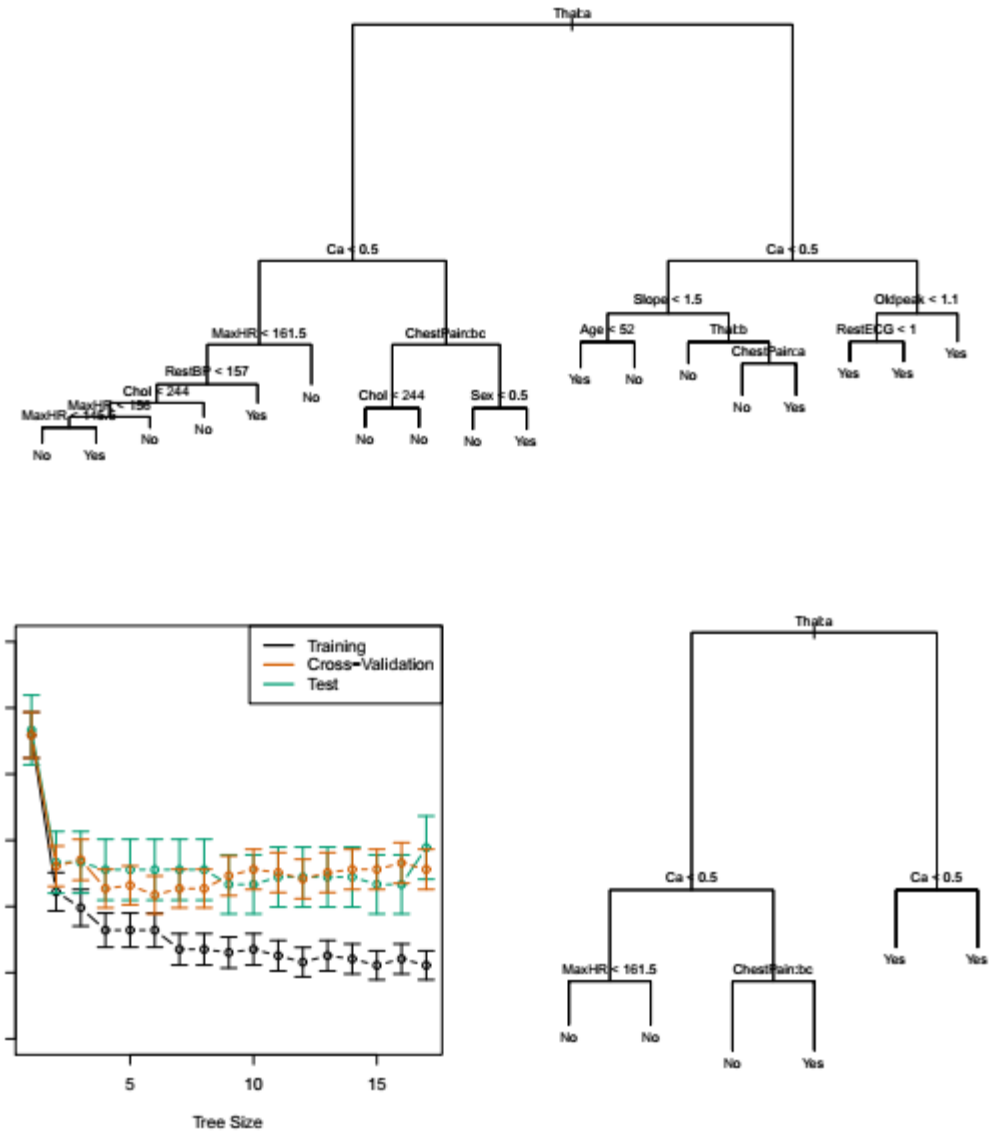


FIGURE 8.6. `Heart` *data.* Top: *The unpruned tree.* Bottom Left: *Cross-validation error, training, and test error, for different sizes of the pruned tree.* Bottom Right: *The pruned tree corresponding to the minimal cross-validation error.*

Gini index

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}),$$

$\hat{p}_{mk}$ represents the proportion of training observations in the *m*th region that are from the *k*th class.

- node purity - Gini index takes on a small value if all of the $\hat{p}_{mk}$ are close to zero or one
- Trying to get the algorithm to create leaves where all examples that end there are one outcome.

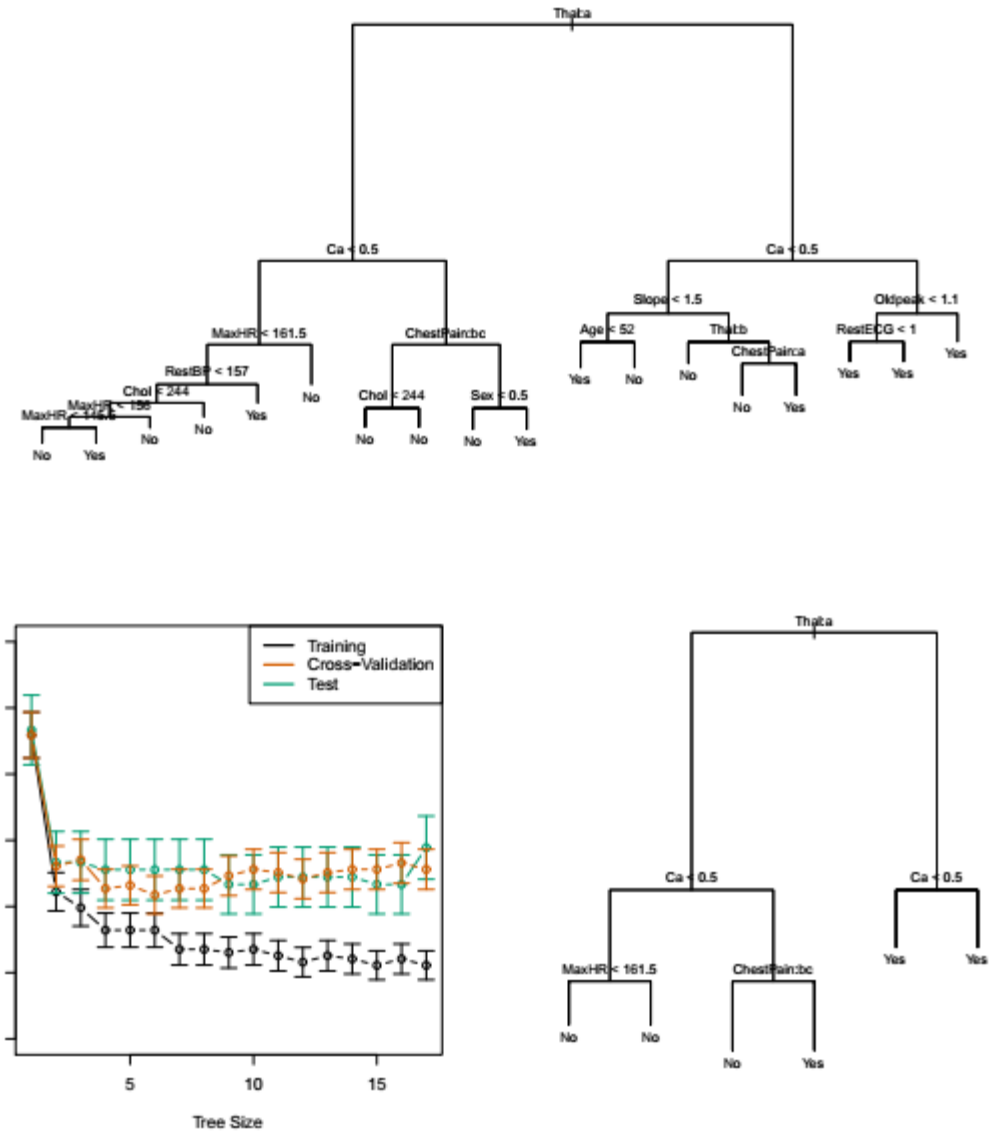


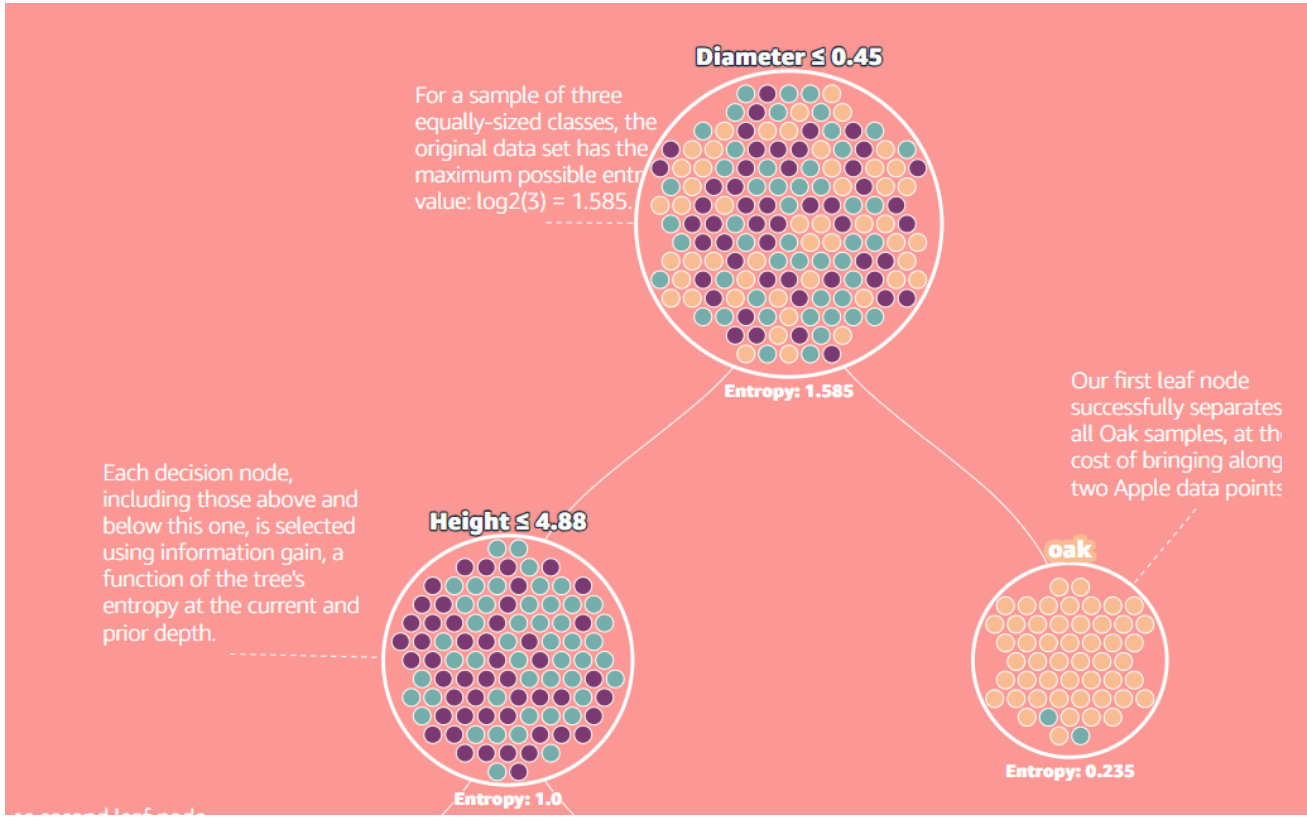
FIGURE 8.6. `Heart` *data.* Top: *The unpruned tree.* Bottom Left: *Cross-validation error, training, and test error, for different sizes of the pruned tree.* Bottom Right: *The pruned tree corresponding to the minimal cross-validation error.*

Entropy

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}.$$

$\hat{p}_{mk}$ represents the proportion of training observations in the *m*th region that are from the *k*th class.

- node purity - entropy takes on a small value if all of the $\hat{p}_{mk}$ are close to zero or one
- Trying to get the algorithm to create leaves where all examples that end there are one outcome.



https://mlu-explain.github.io/decision-tree/

Gini tends to favor splits that isolate the most frequent class in a node.

Entropy focuses on reducing the overall uncertainty of the node.

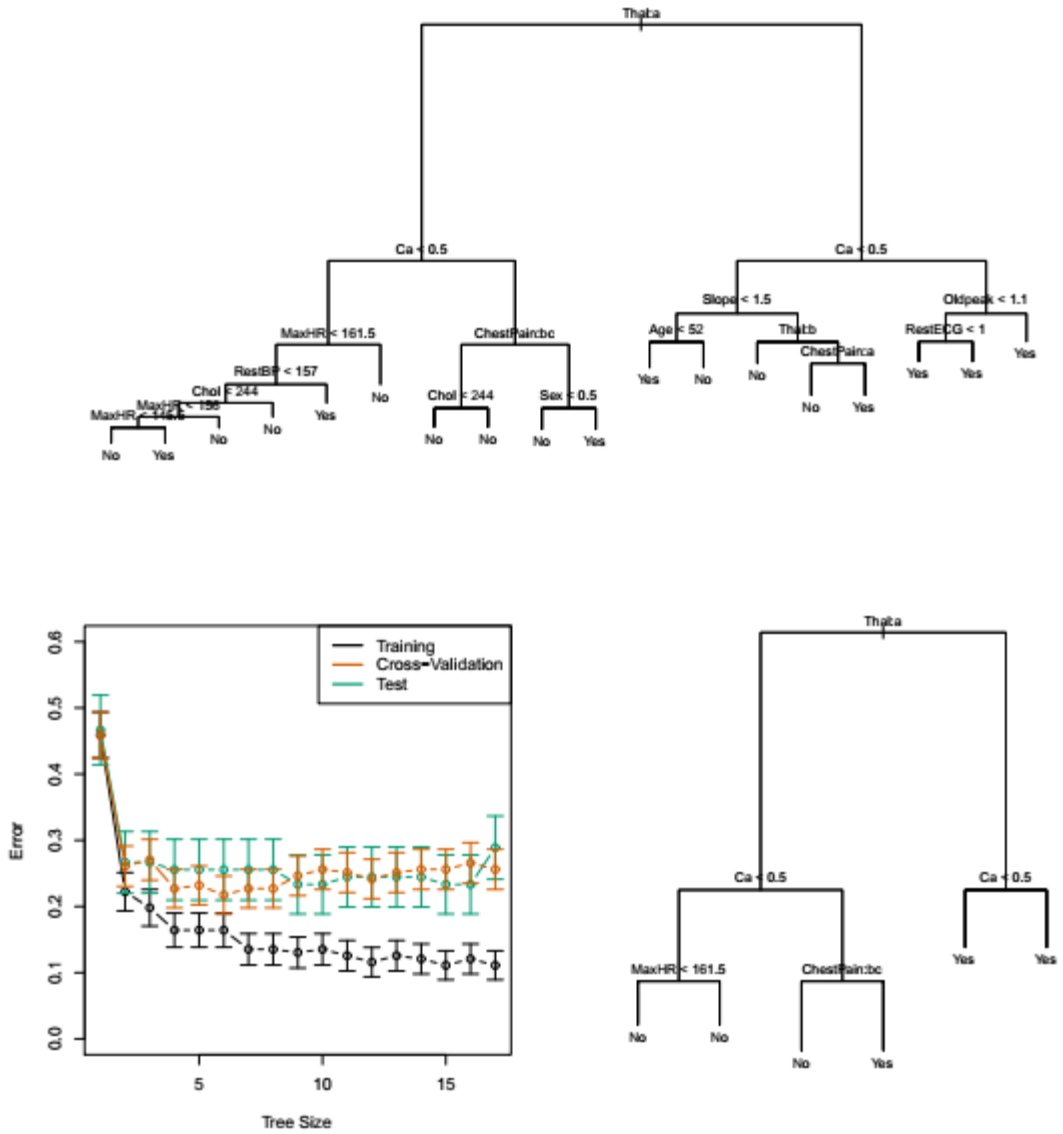


FIGURE 8.6. `Heart` *data.* Top: *The unpruned tree.* Bottom Left: *Cross-validation error, training, and test error, for different sizes of the pruned tree.* Bottom Right: *The pruned tree corresponding to the minimal cross-validation error.*

Trees vs linear regression – which should we choose?

- Trees can fit some non-linear relationships relatively
- Trees can be displayed graphically, and are easily interpreted even by a non-expert (especially if they are small).

In practice trees usually perform badly on their own. But they form the basis for some very effective algorithms.

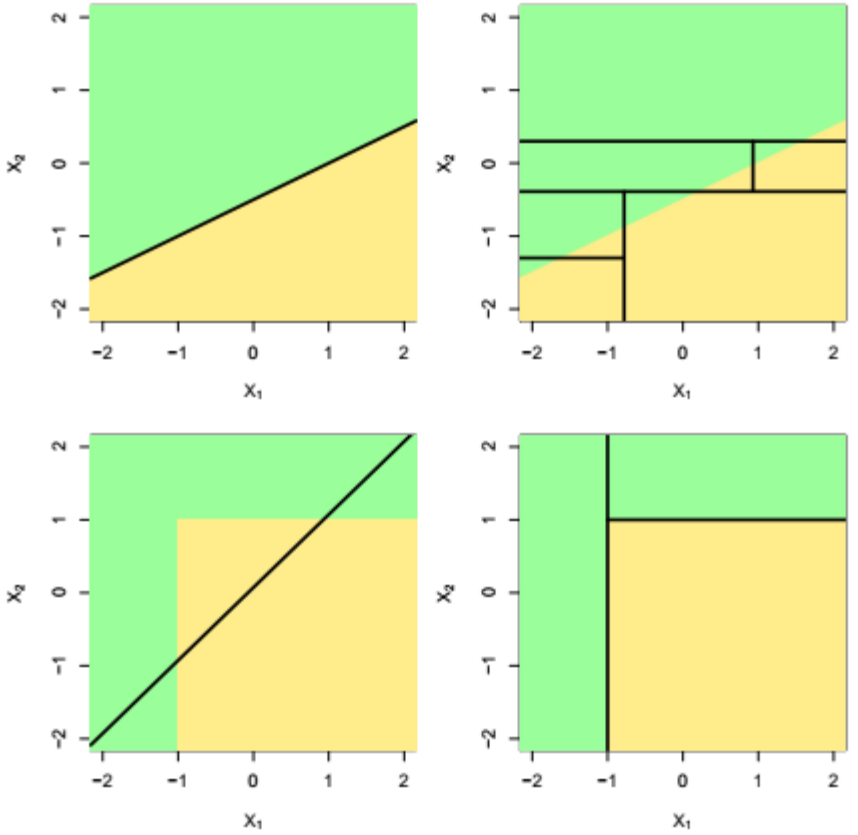


FIGURE 8.7. Top Row: *A two-dimensional classification example in which the true decision boundary is linear, and is indicated by the shaded regions. A classical approach that assumes a linear boundary (left) will outperform a decision tree that performs splits parallel to the axes (right).* Bottom Row: *Here the true decision boundary is non-linear. Here a linear model is unable to capture the true decision boundary (left), whereas a decision tree is successful (right).*

Too big problems:

- Extrapolation
- Variance

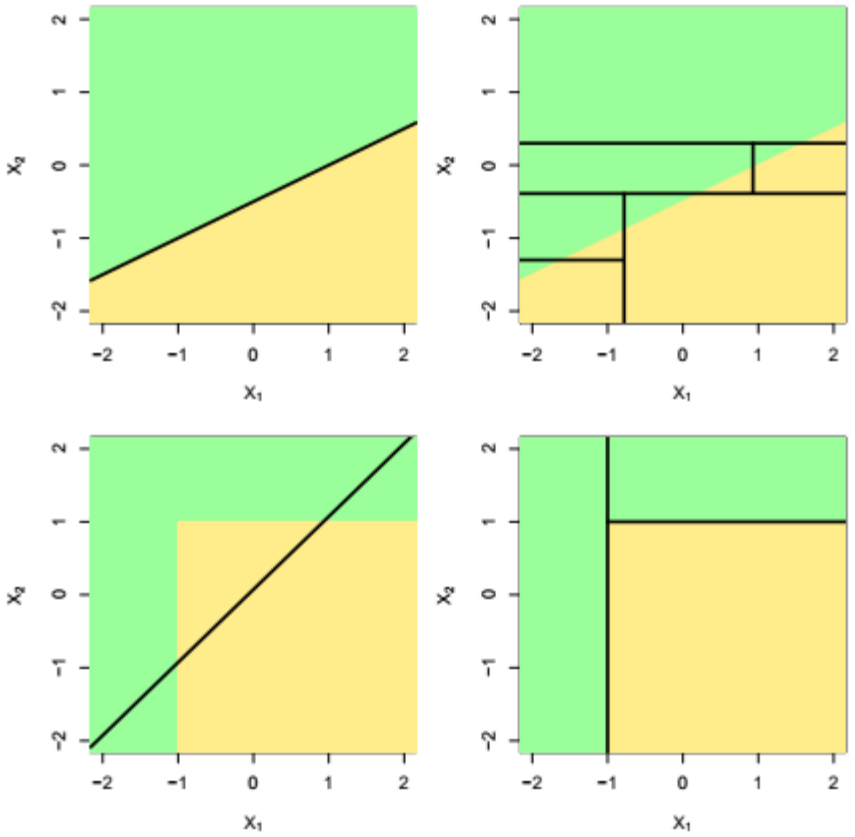


FIGURE 8.7. Top Row: *A two-dimensional classification example in which the true decision boundary is linear, and is indicated by the shaded regions. A classical approach that assumes a linear boundary (left) will outperform a decision tree that performs splits parallel to the axes (right).* Bottom Row: *Here the true decision boundary is non-linear. Here a linear model is unable to capture the true decision boundary (left), whereas a decision tree is successful (right).*

Too big problems:

- Extrapolation (the great tree based weakness)
- Variance (this we can improve)

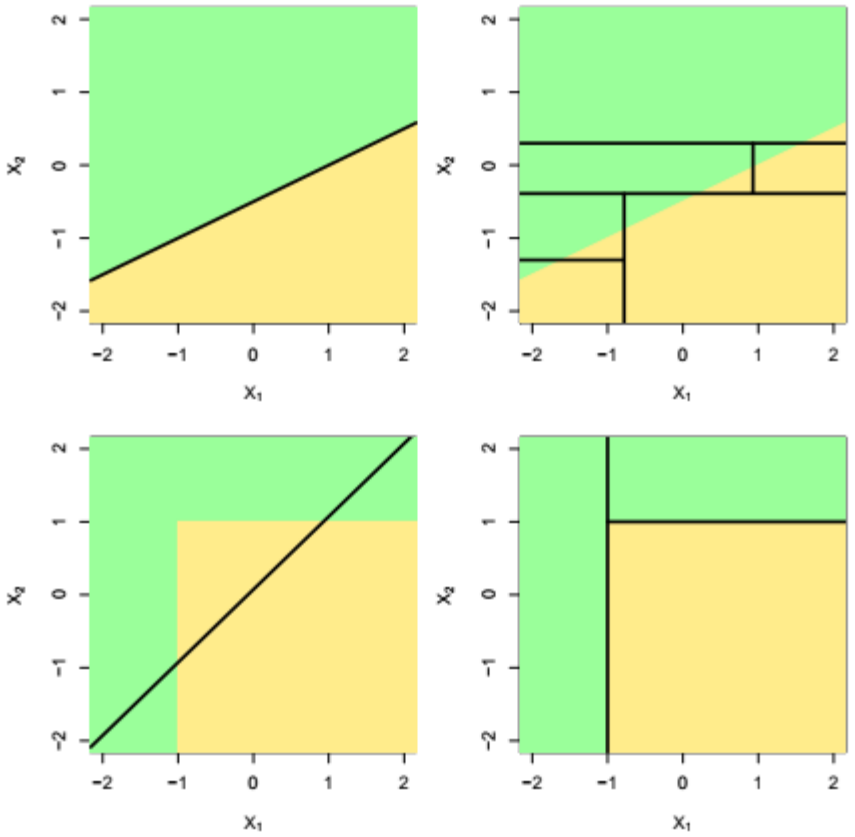


FIGURE 8.7. Top Row: *A two-dimensional classification example in which the true decision boundary is linear, and is indicated by the shaded regions. A classical approach that assumes a linear boundary (left) will outperform a decision tree that performs splits parallel to the axes (right).* Bottom Row: *Here the true decision boundary is non-linear. Here a linear model is unable to capture the true decision boundary (left), whereas a decision tree is successful (right).*

Problem - Extrapolation

The problem - Trees have high variance.

If we split the training data into two parts at random, and fit a decision tree to both halves, the results that we get could be quite different.

In contrast, a procedure with low variance will yield similar results if applied repeatedly to distinct data sets; linear regression tends to have low variance, if the ratio of n to p is moderately large.

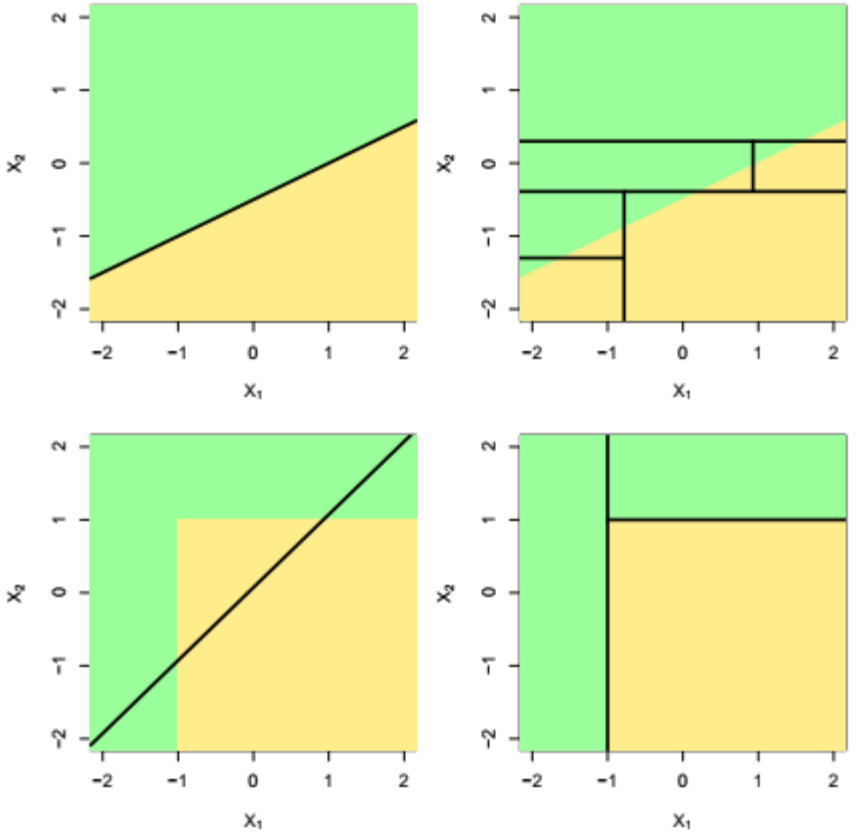


FIGURE 8.7. Top Row: *A two-dimensional classification example in which the true decision boundary is linear, and is indicated by the shaded regions. A classical approach that assumes a linear boundary (left) will outperform a decision tree that performs splits parallel to the axes (right).* Bottom Row: *Here the true decision boundary is non-linear. Here a linear model is unable to capture the true decision boundary (left), whereas a decision tree is successful (right).*

- **Bootstrap aggregation, or bagging**, is a general-purpose procedure for reducing the variance of a statistical learning method which works well here.

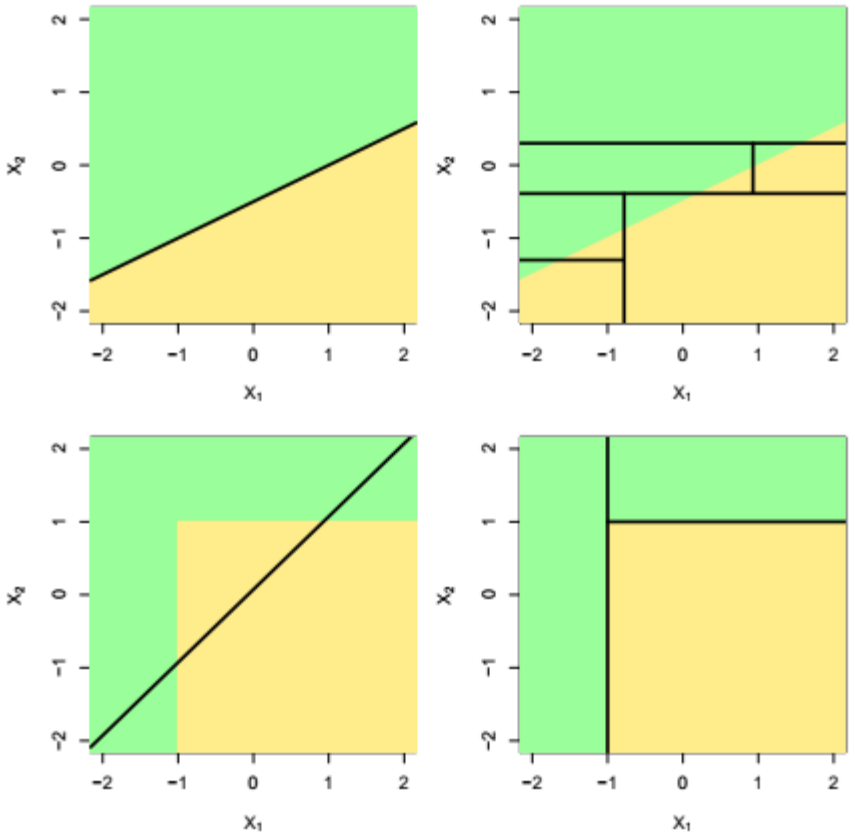
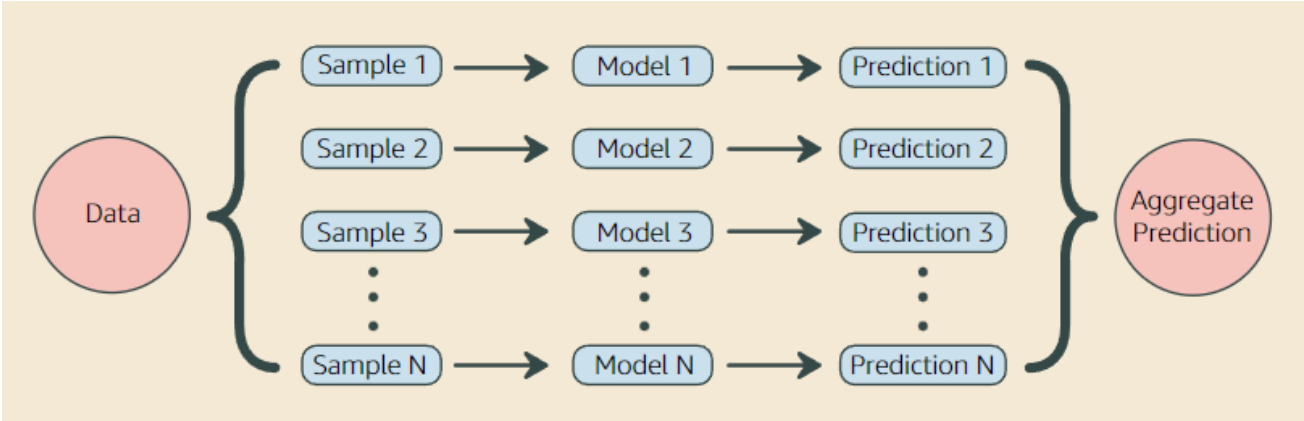


FIGURE 8.7. Top Row: *A two-dimensional classification example in which the true decision boundary is linear, and is indicated by the shaded regions. A classical approach that assumes a linear boundary (left) will outperform a decision tree that performs splits parallel to the axes (right).* Bottom Row: *Here the true decision boundary is non-linear. Here a linear model is unable to capture the true decision boundary (left), whereas a decision tree is successful (right).*

- Averaging a set of observations reduces variance.
- We apply the this approach here, we randomly resample from our data (resample means with replacement), fit a model to each sample and take the average prediction. – Bootstrap aggregating or **Bagging**.
- This is a form of **ensemble learning**.

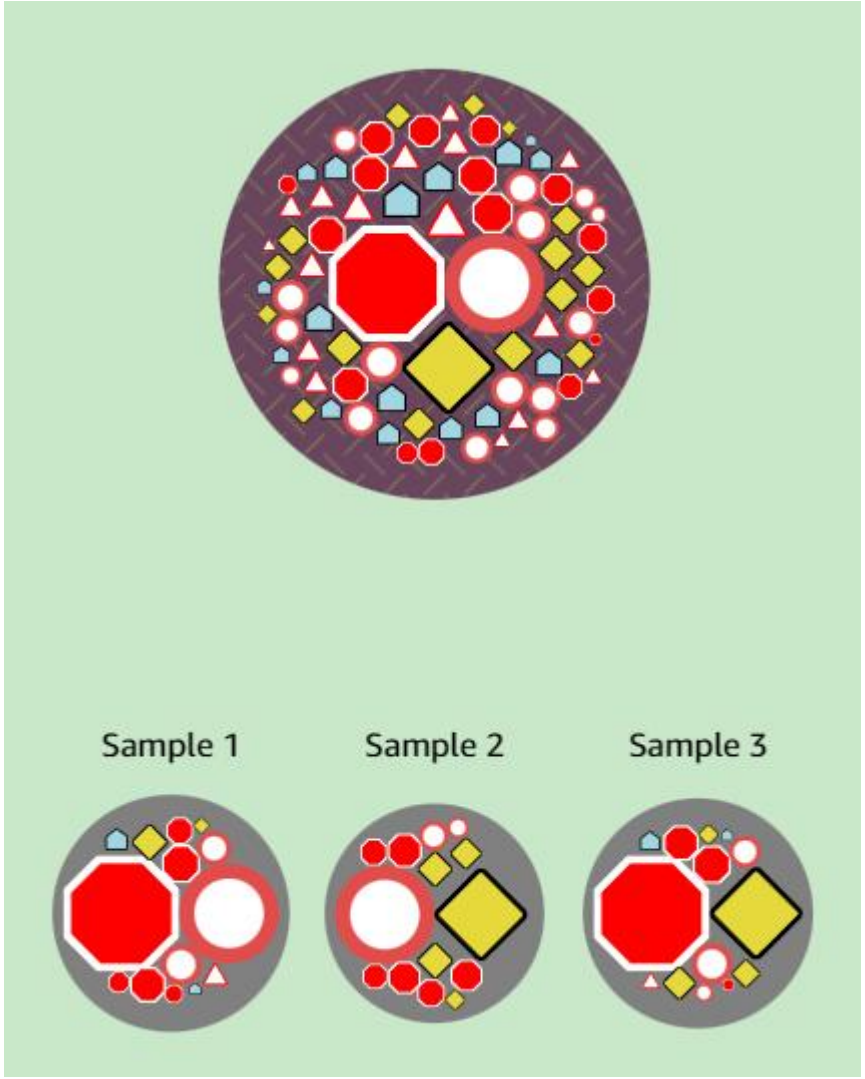


https://mlu-explain.github.io/random-forest/

- Condorcet's Jury Theorem says that if each person is more than 50% correct, then adding more people to vote increases the probability that the majority is correct.
- Wisdom of crowds

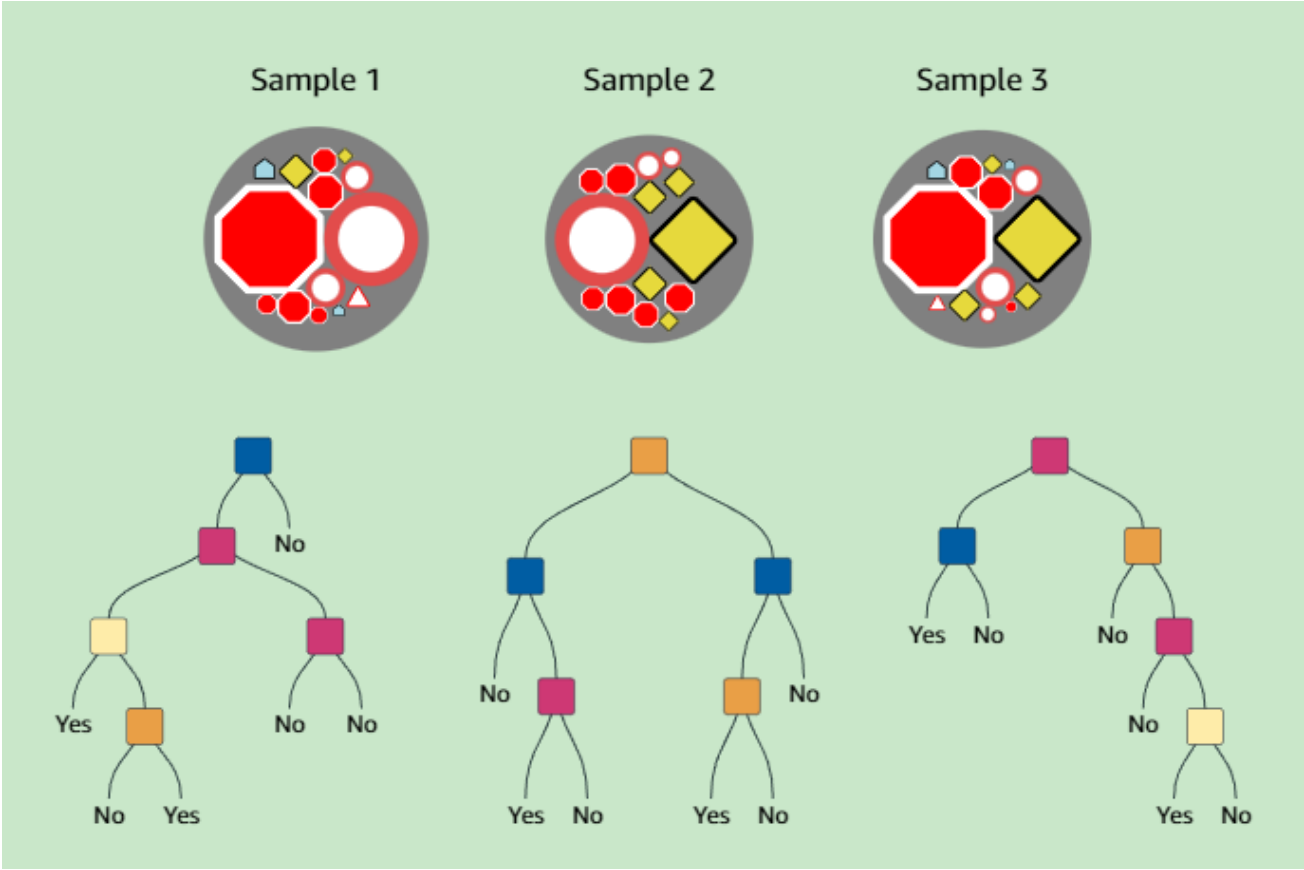
https://mlu-explain.github.io/random-forest/

- If we grow big trees without pruning we will get high variance but low bias, reducing this variance with bagging can move us closer to a lower variance, low bias estimator.
- Note we are going to lose our easy explainability as our results do not come from a single tree.



https://mlu-explain.github.io/random-forest/

- If we grow big trees without pruning we will get high variance but low bias, reducing this variance with bagging can move us closer to a lower variance, low bias estimator.
- Note we are going to lose our easy explainability as our results do not come from a single tree.



https://mlu-explain.github.io/random-forest/

- Bagging gives a natural alternative to cross validation, when we draw each sample we are leaving observations out, we can use these to test our model – Out of bag (OOB) Error

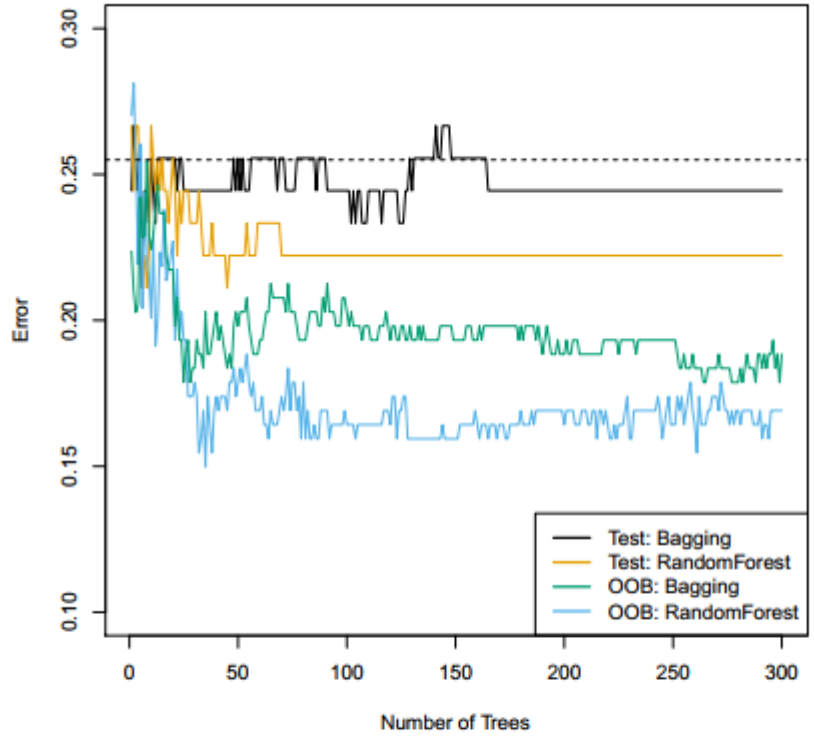


FIGURE 8.8. *Bagging and random forest results for the* `Heart` *data. The test error (black and orange) is shown as a function of* B*, the number of bootstrapped training sets used. Random forests were applied with* $m = \sqrt{p}$*. The dashed line indicates the test error resulting from a single classification tree. The green and blue traces show the OOB error, which in this case is — by chance — considerably lower.*

- Remember we lost our explainability as our results do not come from a single tree. Instead, we can look at variable importance.
- In the case of bagging regression trees, we can record the total amount that the RSS is decreased due to splits over a given predictor.
- Similarly, in the context of bagging classification trees, we can add up the total amount that the Gini index is decreased by splits over a given predictor, averaged over all B trees.

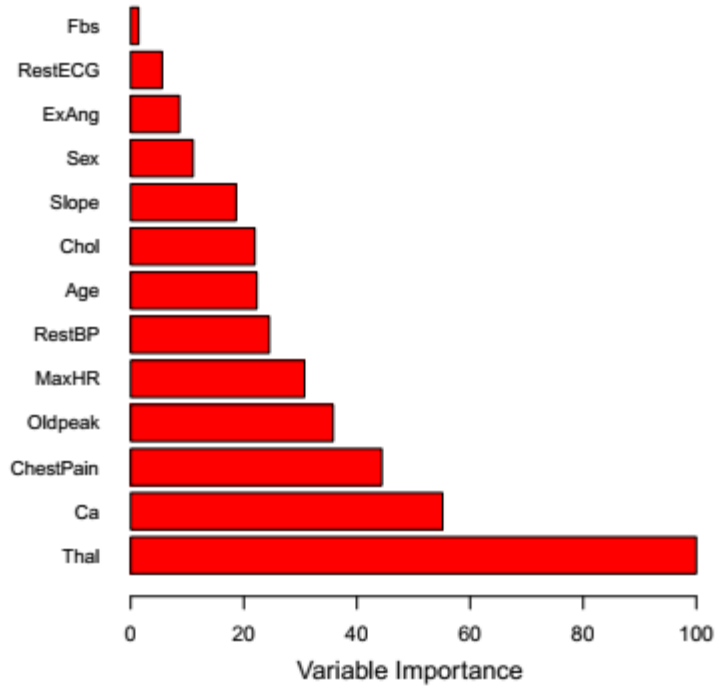


FIGURE 8.9. *A variable importance plot for the* `Heart` *data. Variable importance is computed using the mean decrease in Gini index, and expressed relative to the maximum.*

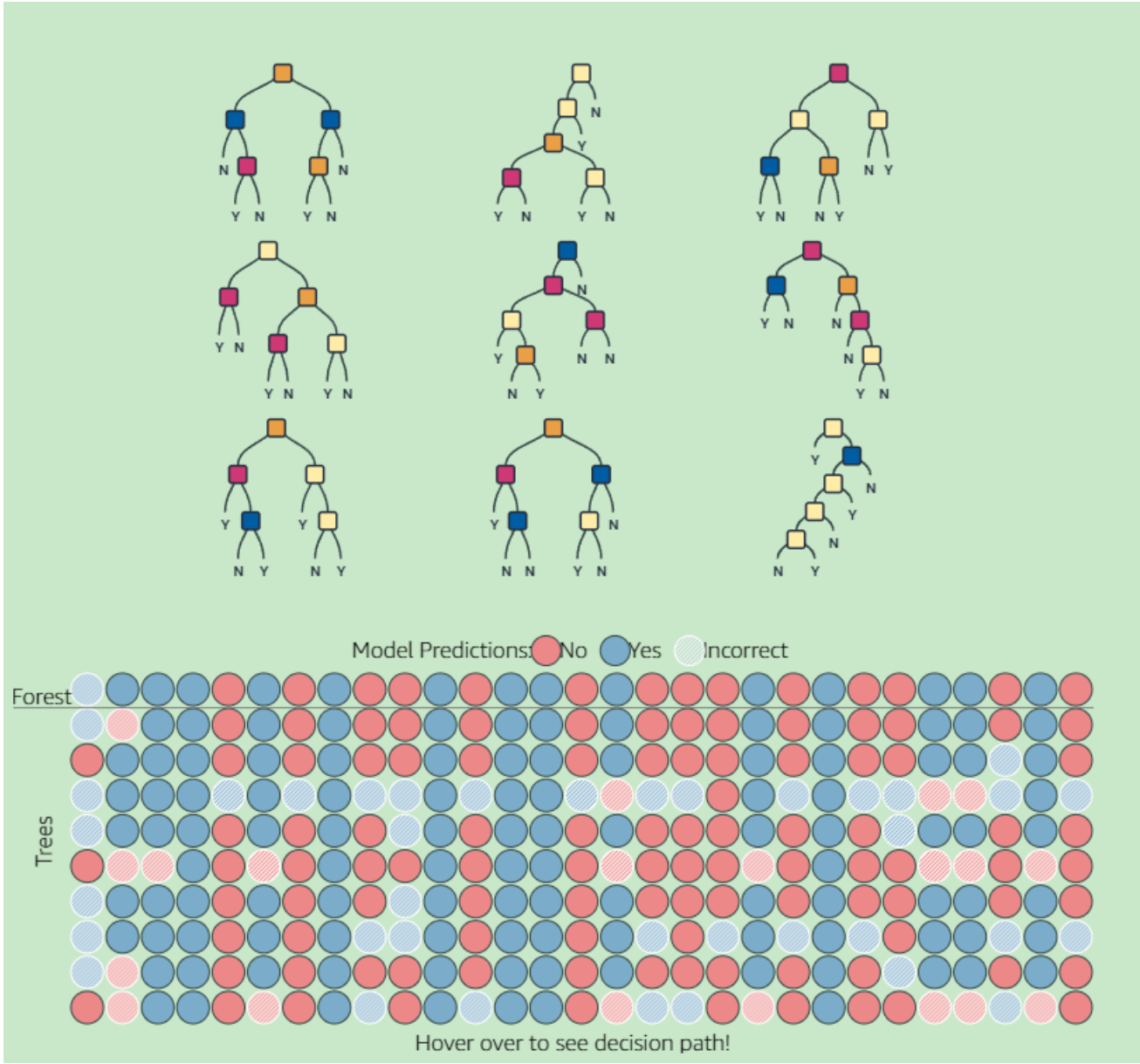
Random forests provide an improvement to bagging specific to trees.

We build a set of decision trees on bootstrapped training samples (a forest).

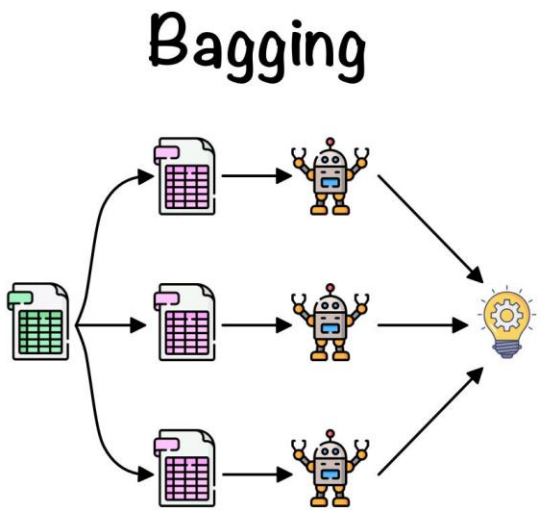
- But when building these decision trees, each time a split in a tree is considered, **a random sample of m predictors is chosen as split candidates** from the full set of p predictors.
- The split is allowed to use only one of those m predictors. A fresh sample of m predictors is taken at each split, and typically we choose $m \approx \sqrt{p}$—that is, the number of predictors considered at each split is approximately equal to the square root of the total number of predictors.

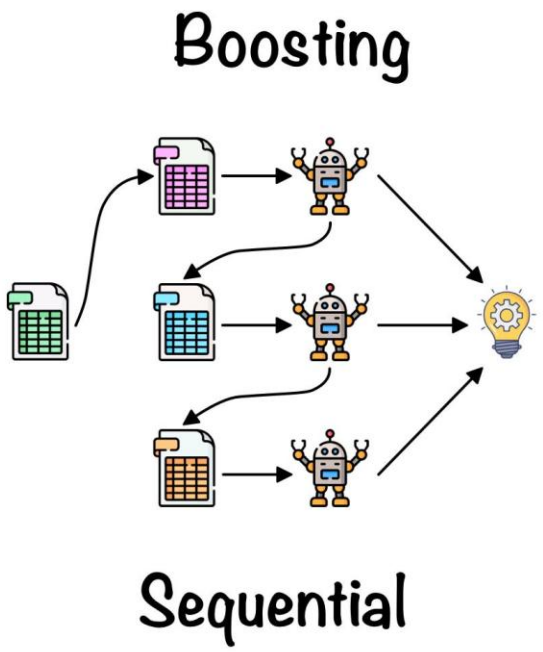
- Random forests give us more diversity in the generated trees.

- If each tree produces the same prediction, then the accuracy cannot improve.
- each circle represents a prediction from a model.
- Using a small value of m in building a random forest will typically be helpful when we have a large number of correlated predictors.



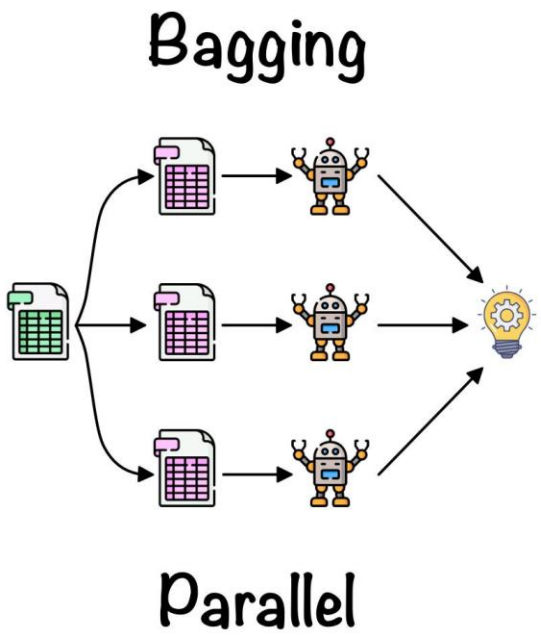
Boosting

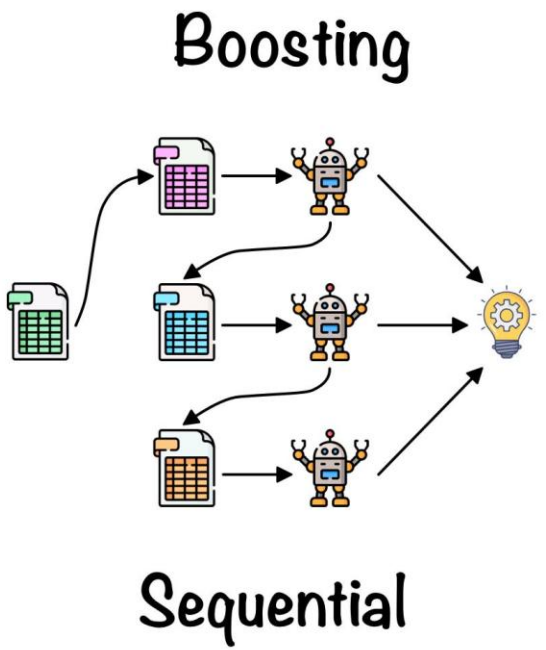




https://towardsdatascience.com/ensemble-learning-bagging-boosting-3098079e5422

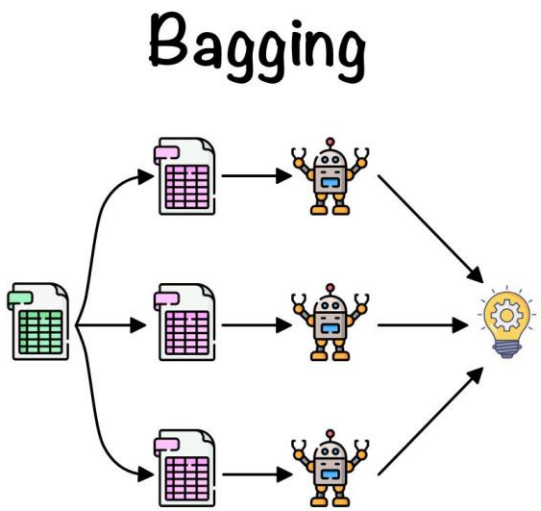
- Boosting
- Gradient Boosting Trees work by iteratively adding weak learners (usually decision trees) to the final model in such a way as to minimize the loss function.
- Trees are grown sequentially: each tree is grown using information from previously grown trees. Each tree is fit on a modified version of the original data set (not a resample).

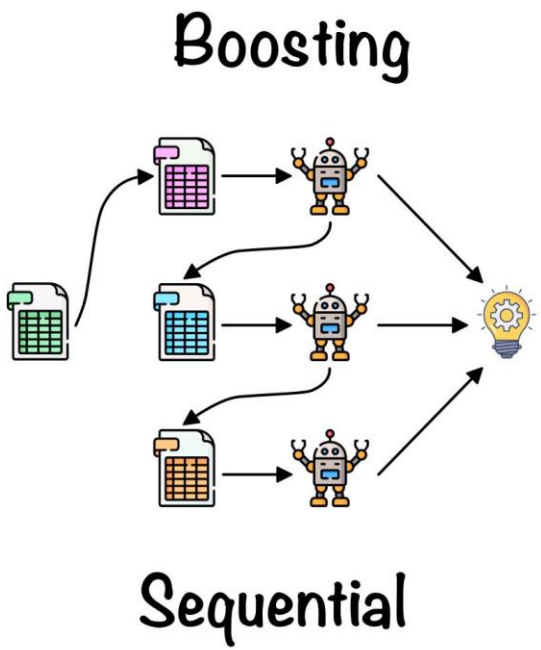




https://towardsdatascience.com/ensemble-learning-bagging-boosting-3098079e5422

Analogy: Imagine you're trying to teach a group of students a complicated subject.

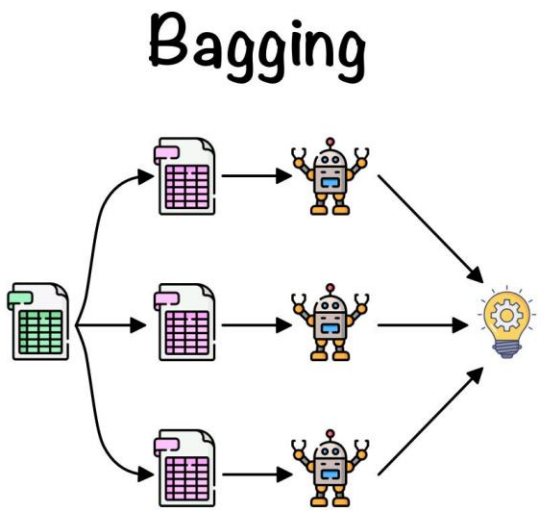


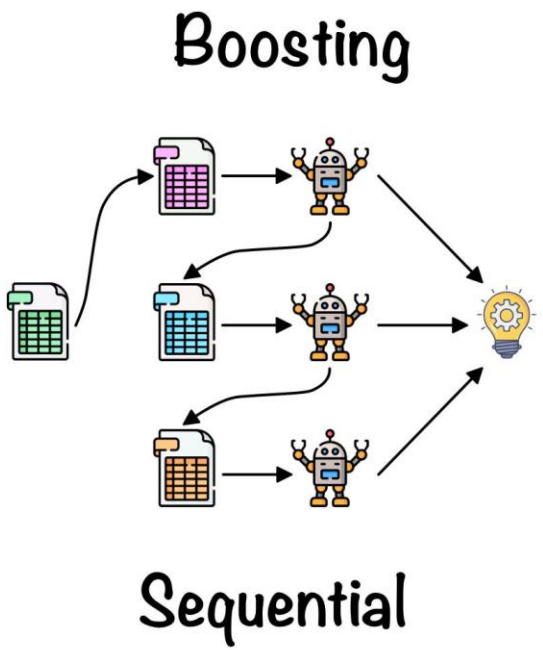


https://towardsdatascience.com/ensemble-learning-bagging-boosting-3098079e5422

Instead of explaining everything all at once, you decide to break it down into easier chunks.

- You first teach the basics.
- You check where the students are making mistakes,
- and then focus on those areas in the next lesson.

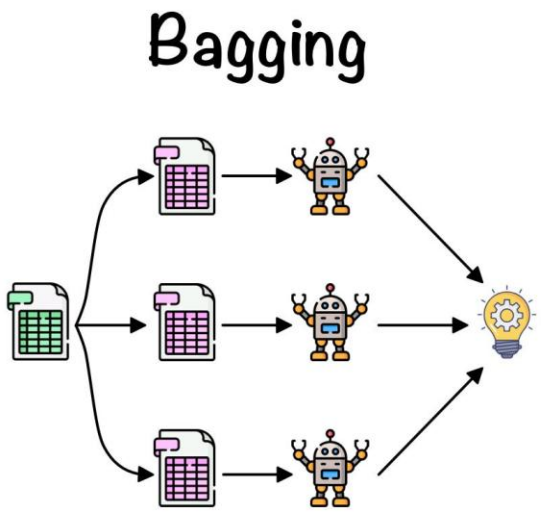


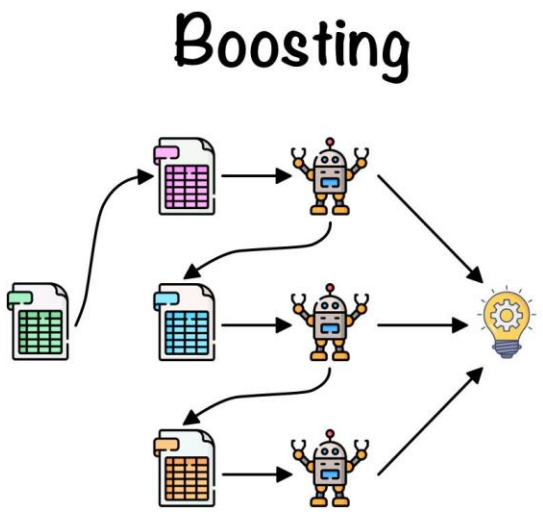


https://towardsdatascience.com/ensemble-learning-bagging-boosting-3098079e5422

GBT works in a similar way.

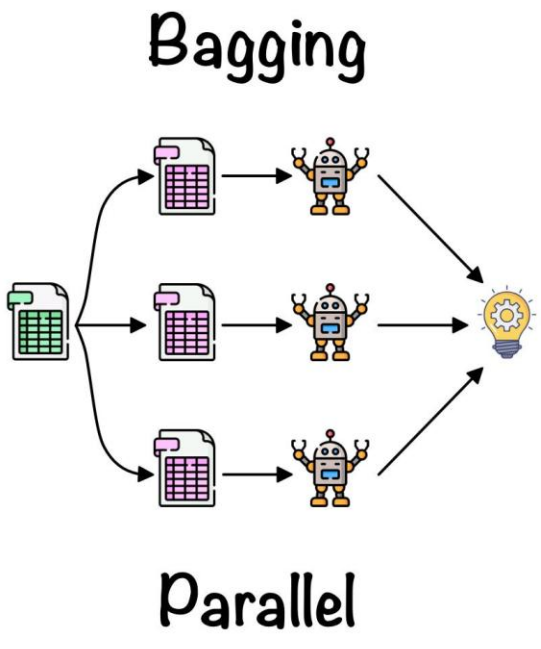
- It builds one tree at a time, where each tree tries to correct the mistakes made by the previous one.
- This is done sequentially until you get a model that's as accurate as possible.

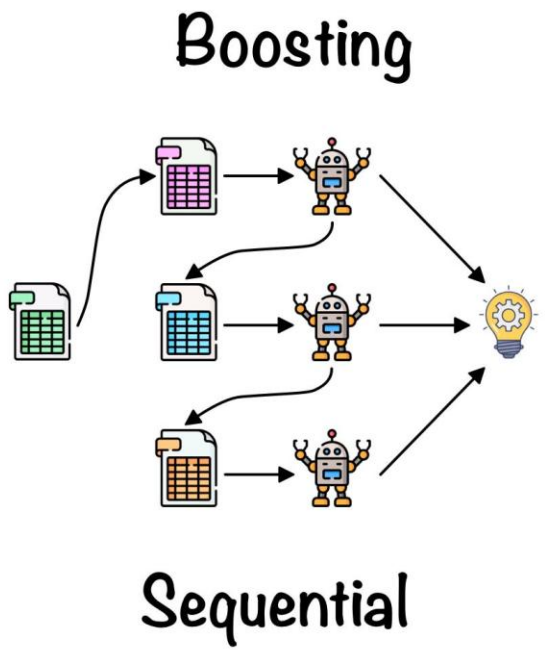




https://towardsdatascience.com/ensemble-learning-bagging-boosting-3098079e5422

- Starting Point: You begin with a basic model, often just a single value that is the best guess for all data points. Think of this as a "starting average."
- Finding Mistakes (Pseudo-Residuals): After each tree is built, GBT looks at where the model went wrong—the difference between the actual values and what the model predicted. These "mistakes" are called "pseudo-residuals."
- Correcting Mistakes: The next tree is specifically built to correct these mistakes. It's like giving extra attention to the questions that most students got wrong on a test.
- Combining the Lessons: Each new tree adds a bit more to what the previous trees have "learned," gradually improving the model.
- Slow and Steady (Shrinkage): GBT doesn't fully trust each tree; instead, it combines them in a way that is more cautious, so that the model doesn't learn too fast and make big mistakes (overfitting).





https://towardsdatascience.com/ensemble-learning-bagging-boosting-3098079e5422

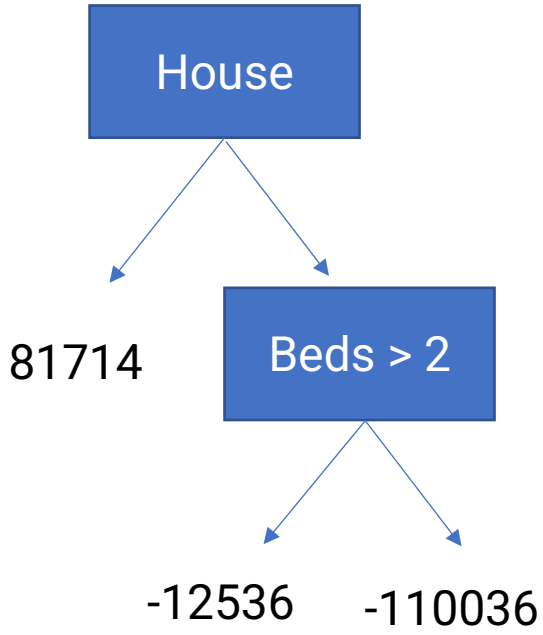
Age	Type	No. Beds	House Price
5	House	3	495875
10	House	2	407500
7	Apartment	4	367000
25	Apartment	1	207750
30	House	2	376500
22	Apartment	2	262000
19	Appartment	3	297750

The average value is 344911

Age	Type	No. Beds	House Price	Predict
5	House	3	495875	344911
10	House	2	407500	344911
7	Apartment	4	367000	344911
25	Apartment	1	207750	344911
30	House	2	376500	344911
22	Apartment	2	262000	344911
19	Appartment	3	297750	344911
				RMSE

Age	Type	No. Beds	House Price	Predict	Residual
5	House	3	495875	344911	150964
10	House	2	407500	344911	62589
7	Apartment	4	367000	344911	22089
25	Apartment	1	207750	344911	-137161
30	House	2	376500	344911	31589
22	Apartment	2	262000	344911	-82911
19	Appartment	3	297750	344911	-47161
				RMSE	2.36865E+05

Age	Type	No. Beds	House Price	Predict	Residual
5	House	3	495875	344911	150964
10	House	2	407500	344911	62589
7	Apartment	4	367000	344911	22089
25	Apartment	1	207750	344911	-137161
30	House	2	376500	344911	31589
22	Apartment	2	262000	344911	-82911
19	Appartment	3	297750	344911	-47161
				RMSE	2.36865E+05



Don’t add the full residual, apply a learning rate

Age	Type	No. Beds	House Price	Predict	Residual	Estimated residual	Prediction	Residual
5	House	3	495875	344911	150964	81714	426625	69250
10	House	2	407500	344911	62589	81714	426625	-19125
7	Apartment	4	367000	344911	22089	-12536	332375	34625
25	Apartment	1	207750	344911	-137161	-110036	234875	-27125
30	House	2	376500	344911	31589	81714	426625	-50125
22	Apartment	2	262000	344911	-82911	-110036	234875	27125
19	Appartment	3	297750	344911	-47161	-12536	332375	-34625
				RMSE	2.36865E+05		RMSE	1.07439E+05

Brand new house, 10 beds – what is the price prediction?

GBT

- Great performance overall.
- Computationally expensive and can be slow on large datasets.
- Prone to overfitting
- We have different flavours of the basic approach.

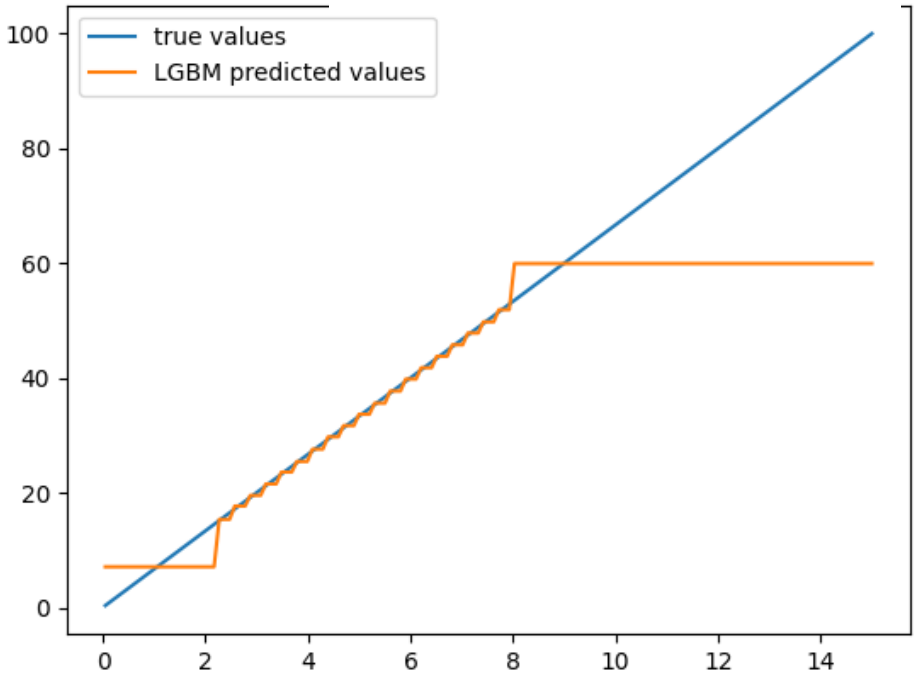
XGBoost (Extreme Gradient Boosting)

- An optimized version of GBT, also employs decision trees as weak learners. Utilizes both parallel and distributed computing and has the capability to handle missing values
- Implements the concept of regularization to prevent overfitting, unlike traditional GBTs.

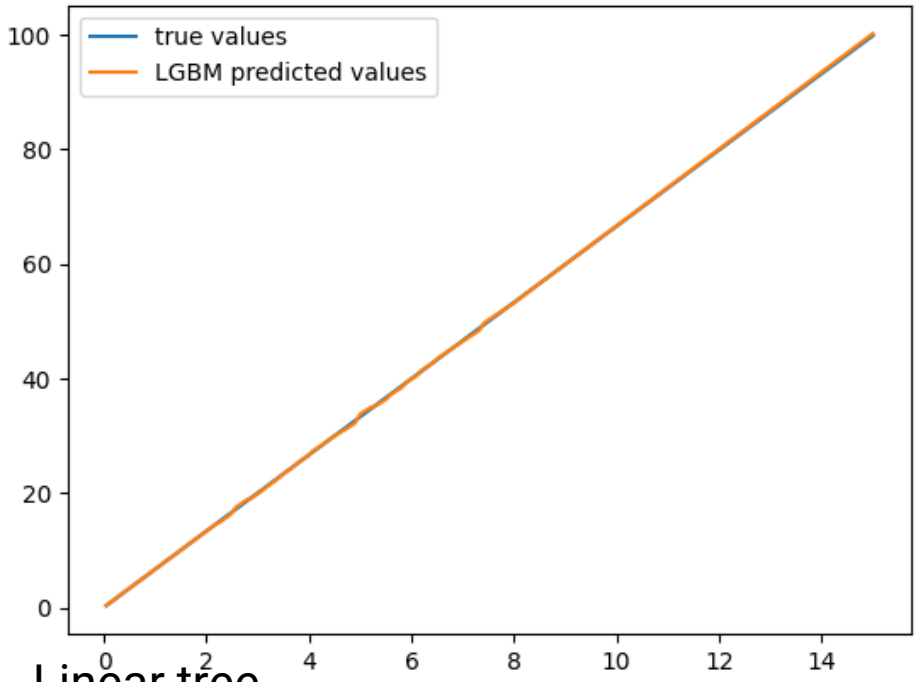
LightGBM (Light Gradient Boosting Machine)

- Uses histogram-based learning, which buckets continuous feature values into discrete bins, speeding up training and reducing memory usage.
- Implements "leaf-wise" tree growth strategy, as opposed to "level-wise," making it more accurate but also prone to overfitting for small datasets.
- LightGBM is designed to be highly efficient both in terms of memory and computational speed.

$$w_j^* = -\frac{G_j}{H_j + \lambda}$$



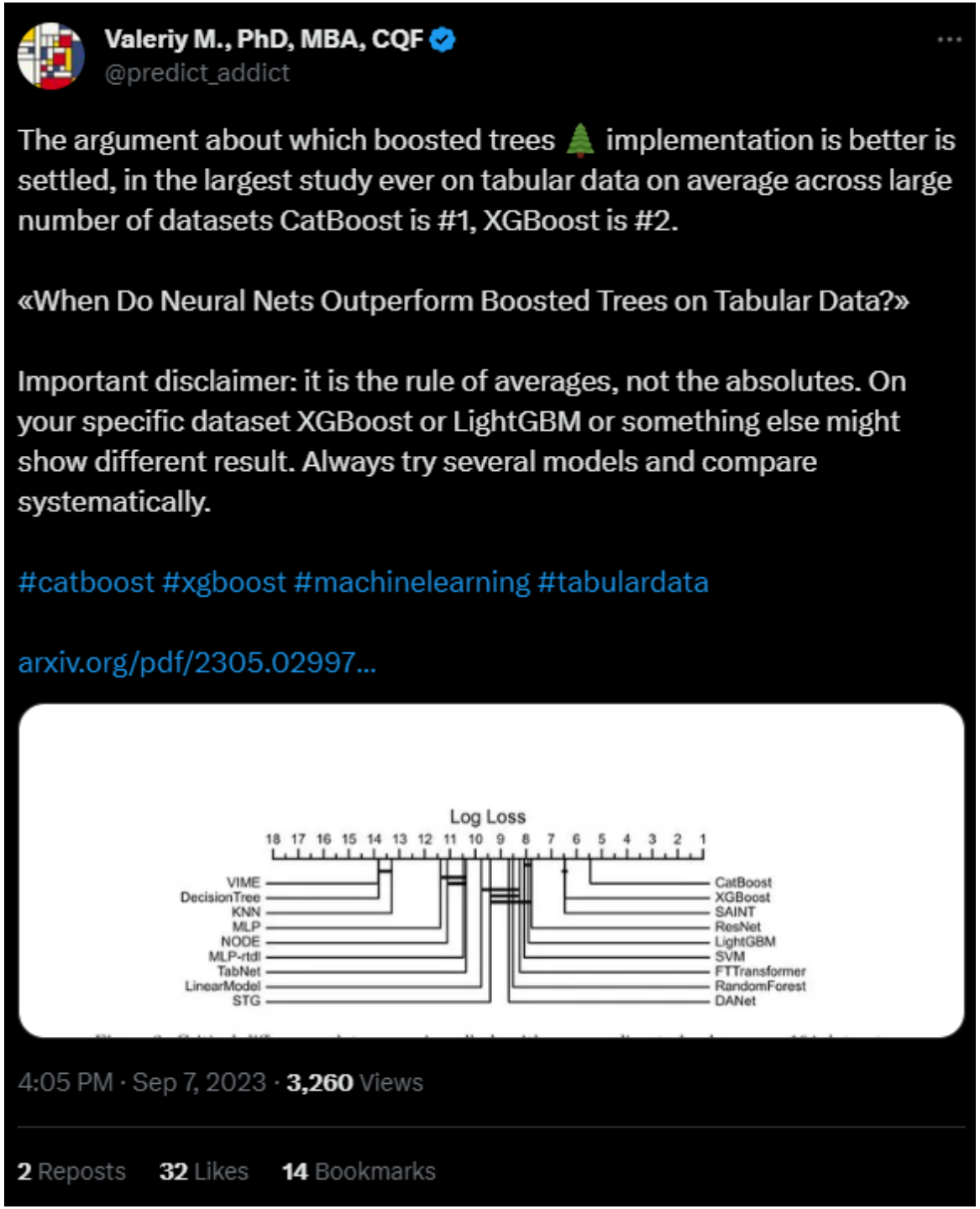
$$w^*(\mathbf{x}) = b + \mathbf{a} \cdot \mathbf{x}_{selected}$$



Linear tree

- CatBoost (Categorical Boosting)
- Explicitly designed to handle categorical variables by employing various forms of target encoding.
- Comparable to XGBoost in terms of speed but provides robustness to overfitting, especially when the dataset includes many categorical features.
- May be less interpretable due to the complex encoding of categorical variables.

- Which to use?
- Catboost current champion
- Problem dependent!!!!



CARL MCBRIDE ELLIS · 1261ST IN THIS COMPETITION · POSTED A MONTH AGO

CatBoost is all you need?

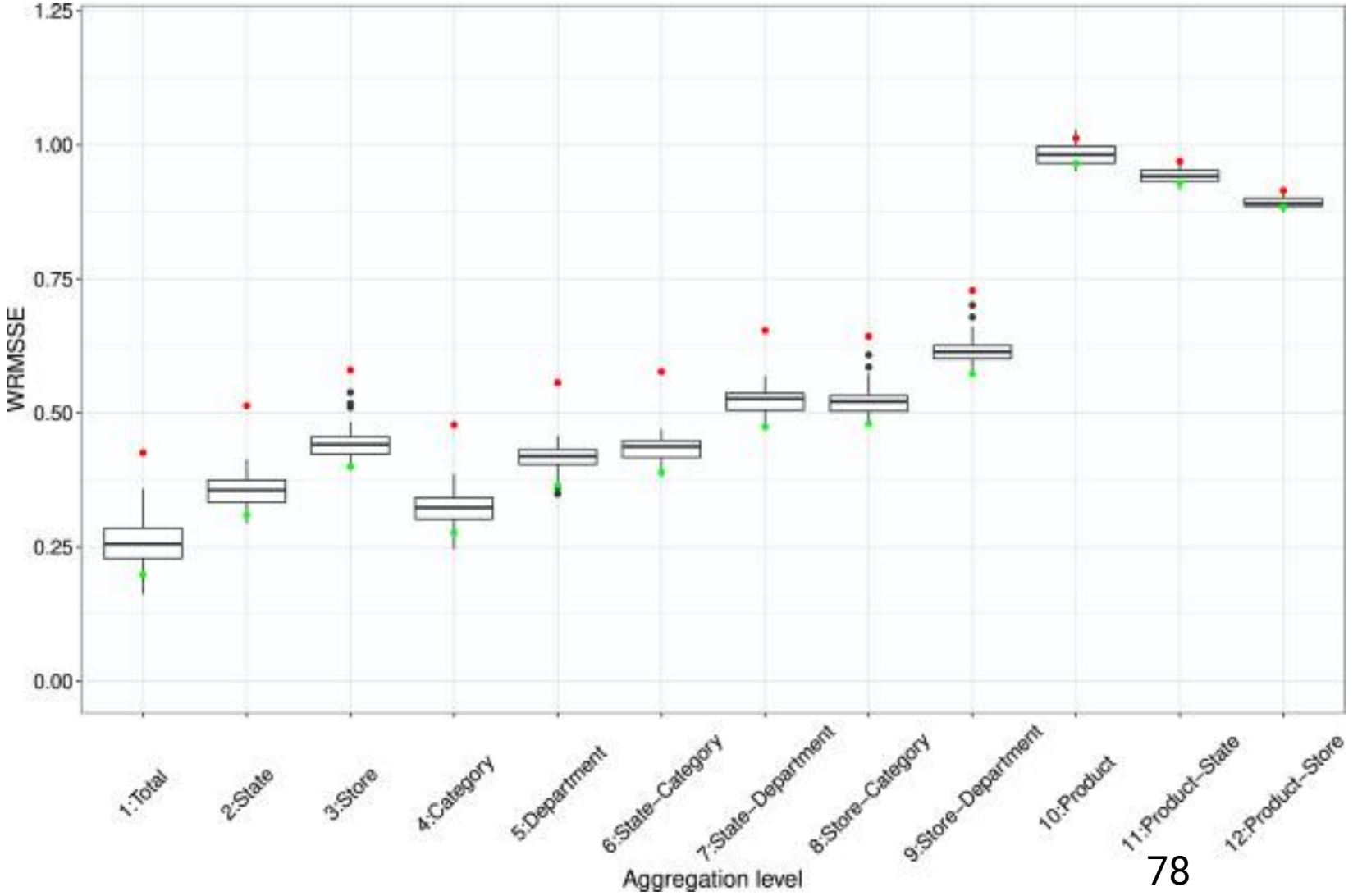
Given that there is now no cap on daily submissions I thought I would take a quick look at the performance of some of the popular estimators with no hyper-parameter tuning or feature selection whatsoever. Here are the results:

Classifier	Public LB score	Private LB score
CatBoost	0.20	0.41
TabPFN	0.20	0.48
XGBoost	0.22	0.54
LightGBM	0.28	0.72

(Source)

In view of this it seems a vanilla CatBoost submission (*i.e.* just using the default parameters) would have been firmly in the medals section.

Light GBM – Multiple levels



Light GBM - Ensemble

First place (YJ_STU; YeonJun In): The winner of the competition was a senior undergraduate student at Kyung Hee University, South Korea, **who used an equal weighted combination (arithmetic mean) of various LightGBM models**, which were trained to produce forecasts for the product-store series using data pooled per store (10 models), store-category (30 models), and store-department (70 models). Two variations were considered for each type of model, where the first applied a recursive approach and the second a non-recursive forecasting approach (Bontempi et al., 2013). In total, 220 models were built and each series was forecast using the average of six models, where each exploited a different learning approach and training set.

- NNs are biased to overly smooth solutions
- Uninformative features affect more MLP-like NNs
- Data are non invariant by rotation, so should be learning procedures

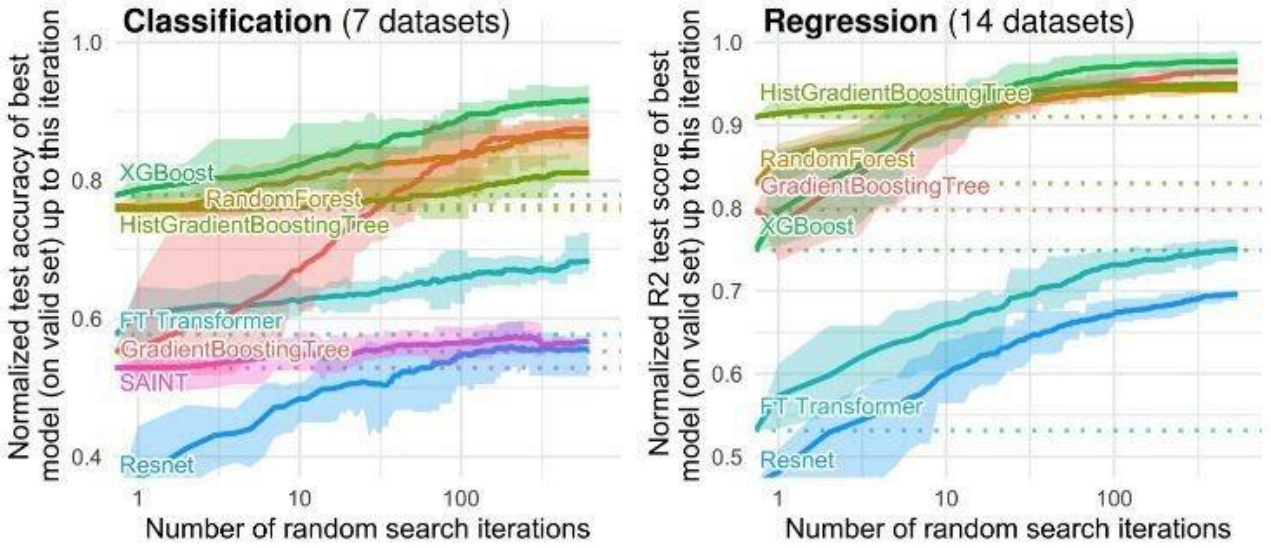


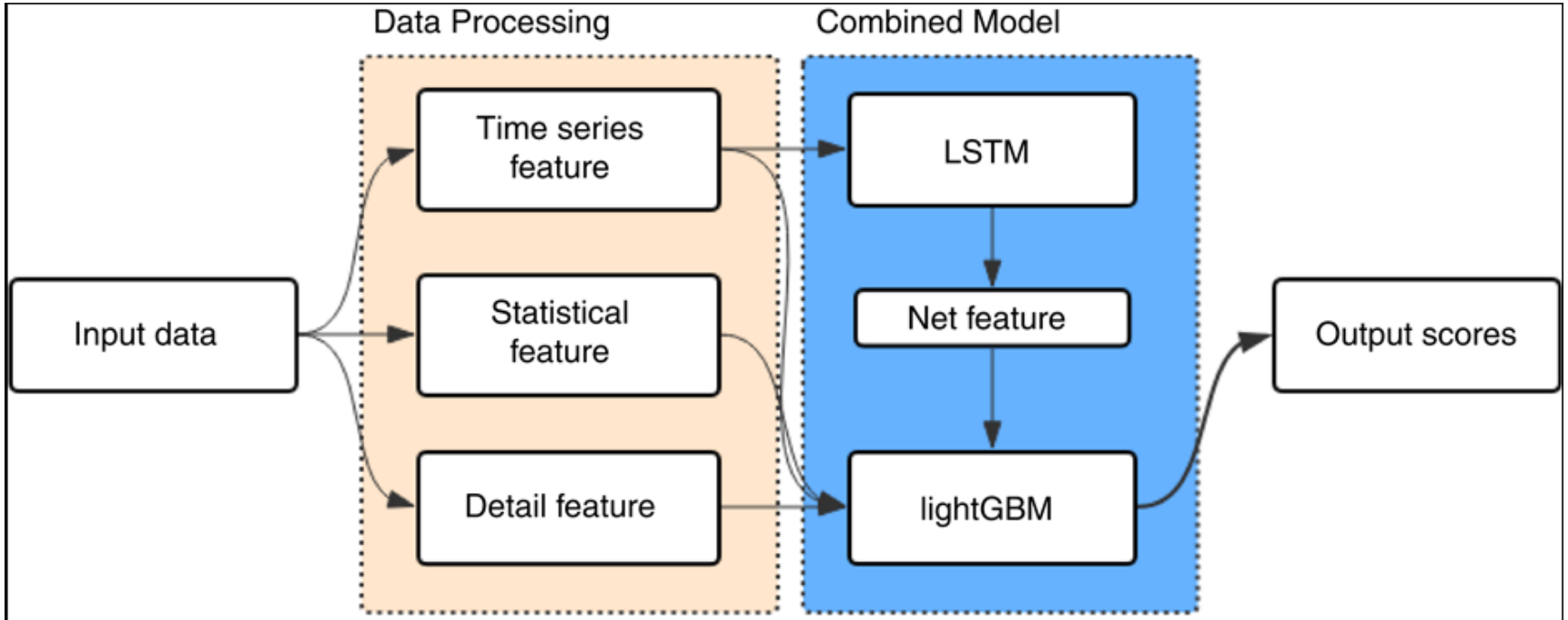
Figure 2: **Benchmark on medium-sized datasets, with both numerical and categorical features.** Dotted lines correspond to the score of the default hyperparameters, which is also the first random search iteration. Each value corresponds to the test score of the best model (on the validation set) after a specific number of random search iterations, averaged on 15 shuffles of the random search order. The ribbon corresponds to the minimum and maximum scores on these 15 shuffles.

https://arxiv.org/abs/2207.08815

Ensembles

- We can add trees to our existing approaches, ML or other!
- Try to fix the errors from our existing models. Remember when we sometime saw something looks like a “pattern” in our residuals? This is something trees could learn.
- Broadly this falls under the category of ensembles!

Ensembles



Notes: The input data are processed into three features: time series feature, statistical feature and Detail feature. The combined model consists of an LSTM model and a lightGBM model. lightGBM model takes features and network parameters of LSTM model as input and outputs the prediction results

Weng, T., Liu, W., & Xiao, J. (2019). Supply chain sales forecasting based on lightGBM and LSTM combination model. *Industrial Management & Data Systems*.

- We don't need stationarity but transformations still useful
- Chop up time series into different samples (moving window of different sizes)
- Add features, features need to contain temporal components (not just amplitude etc)
- Issues with knowledge leakage
- Filtering can be beneficial as overfitting the main problem
- Iteratively predict next step or different models for different future steps, multiple output decision trees (wrapper) – high performance
- In practice useful to check for multicolinearity

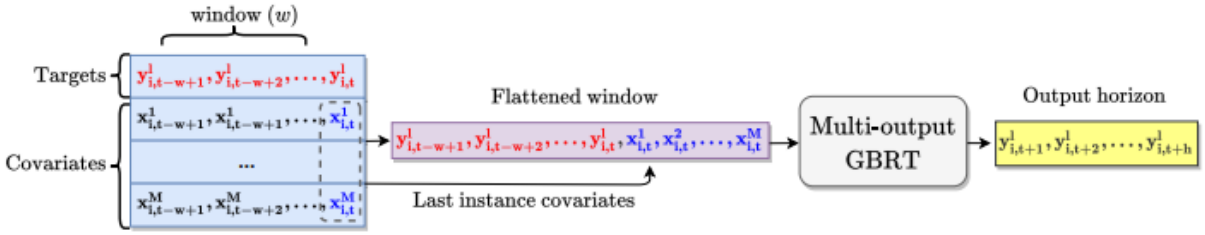


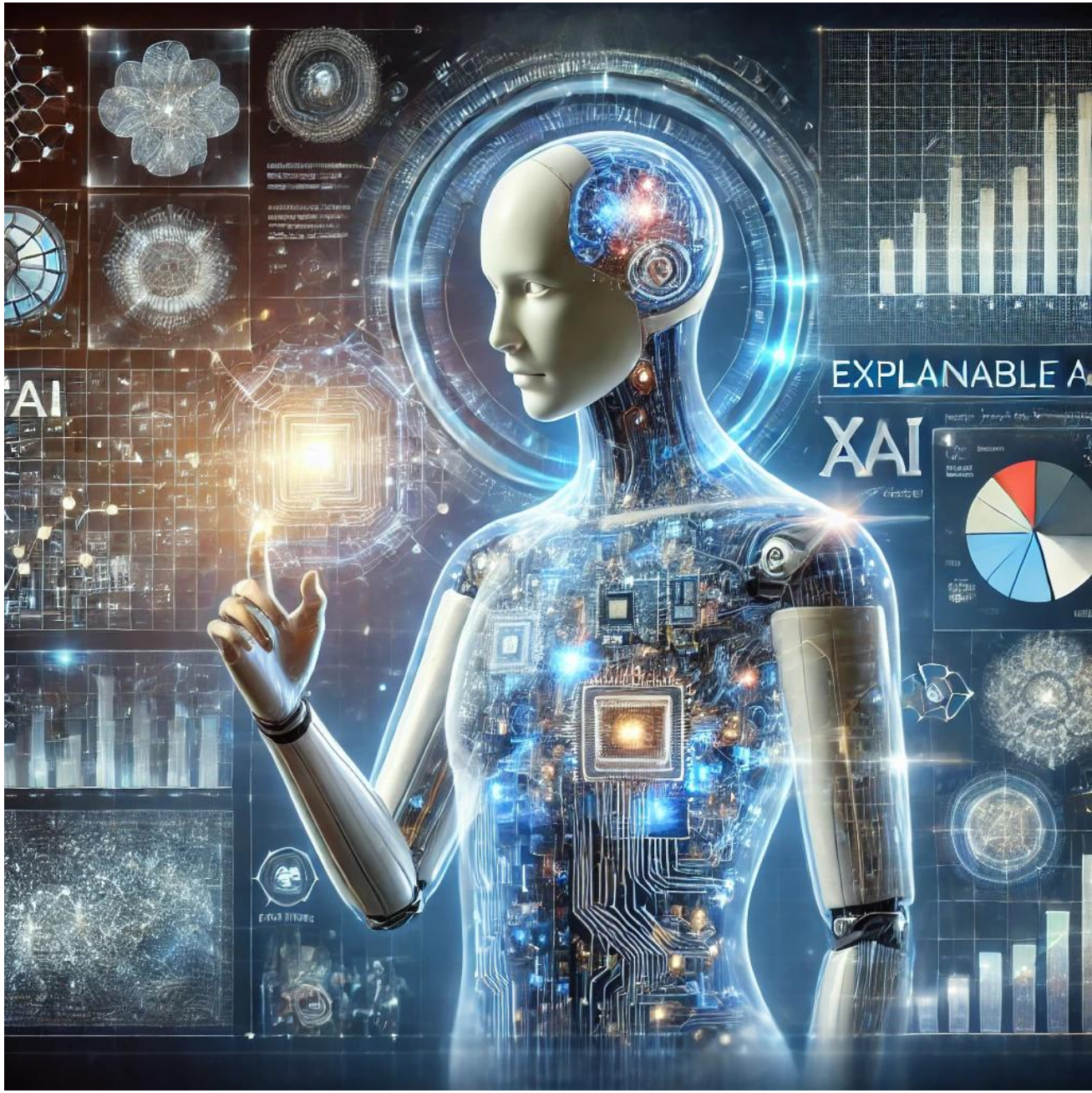
Fig. 1. Reconfiguration of a single window of input instances. $Y^l_{i,t-w+1}, \ldots, Y^l_{i,t}$ are the target instances in the window, whereas $X^1_{i,t-w+1}, \ldots, X^M_{i,t}$ are the (external) features in the window for a particular time series $i \in n$.

hctsa feature name	Description
Distribution	
DN_HistogramMode_5	Mode of z-scored distribution (5-bin histogram)
DN_HistogramMode_10	Mode of z-scored distribution (10-bin histogram)
Simple temporal statistics	
SB_BinaryStats_mean_longstretch1	Longest period of consecutive values above the mean
DN_OutlierInclude_p_001_mdrmd	Time intervals between successive extreme events above the mean
DN_OutlierInclude_n_001_mdrmd	Time intervals between successive extreme events below the mean
Linear autocorrelation	
CO_f1ecac	First $1/e$ crossing of autocorrelation function
CO_FirstMin_ac	First minimum of autocorrelation function
SP_Summaries_welch_rect_area_5_1	Total power in lowest fifth of frequencies in the Fourier power spectrum
SP_Summaries_welch_rect_centroid	Centroid of the Fourier power spectrum
FC_LocalSimple_mean3_stderr	Mean error from a rolling 3-sample mean forecasting
Nonlinear autocorrelation	
CO_trev_1_num	Time-reversibility statistic, $\langle(x_{t+1} - x_t)^3\rangle_t$
CO_HistogramAMI_even_2_5	Automutual information, $m = 2, \tau = 5$
IN_AutoMutualInfoStats_40_gaussian_fmmi	First minimum of the automutual information function
Successive differences	
MD_hrv_classic_pnn40	Proportion of successive differences exceeding 0.04σ [20]
SB_BinaryStats_diff_longstretch0	Longest period of successive incremental decreases
SB_MotifThree_quantile_hh	Shannon entropy of two successive letters in equiprobable 3-letter symbolization
FC_LocalSimple_mean1_tauresrat	Change in correlation length after iterative differencing
CO_Embed2_Dist_tau_d_expfit_meandiff	Exponential fit to successive distances in 2-d embedding space
Fluctuation Analysis	
SC_FluctAnal_2_dfa_50_1_2_logi_prop_r1	Proportion of slower timescale fluctuations that scale with DFA (50% sampling)
SC_FluctAnal_2_rsrangefit_50_1_logi_prop_r1	Proportion of slower timescale fluctuations that scale with linearly rescaled range fits
Others	
SB_TransitionMatrix_3ac_sumdiagcov	Trace of covariance of transition matrix between symbols in 3-letter alphabet
PD_PeriodicityWang_th0_01	Periodicity measure of [31]

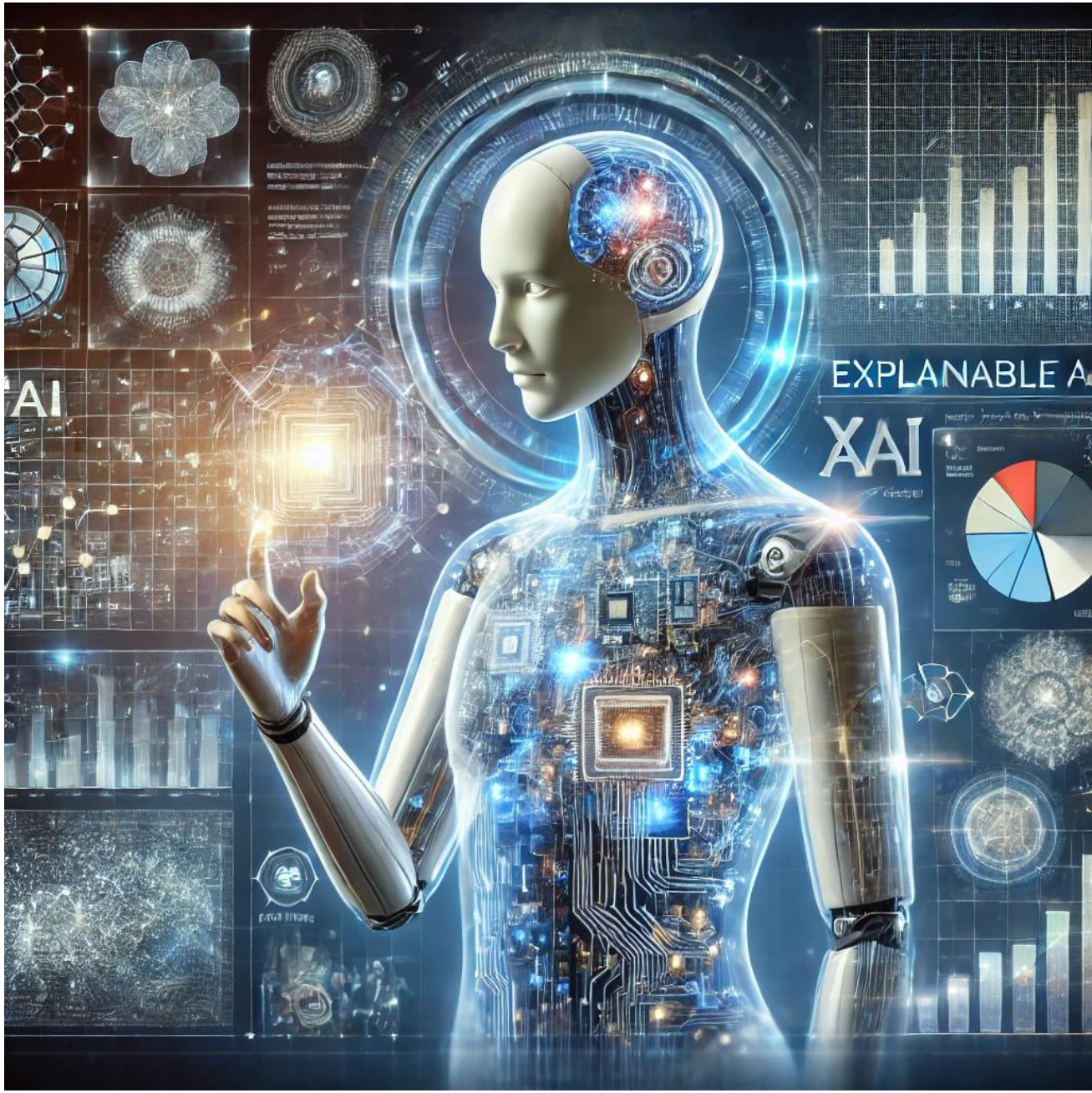
Table 1 The *catch22* feature set spans a diverse range of time-series characteristics representative of the diversity of interdisciplinary methods for time-series analysis. Features in *catch22* capture time-series properties of the distribution of values in the time series, linear and nonlinear temporal autocorrelation properties, scaling of fluctuations, and others.

Explaining Highly Non-Linear models.

- Combining lots of simple learner greatly increases overall performance.
- Makes things harder to interpret.
- Even fundamentally, if an effect is the product of two events, which one caused the effect.

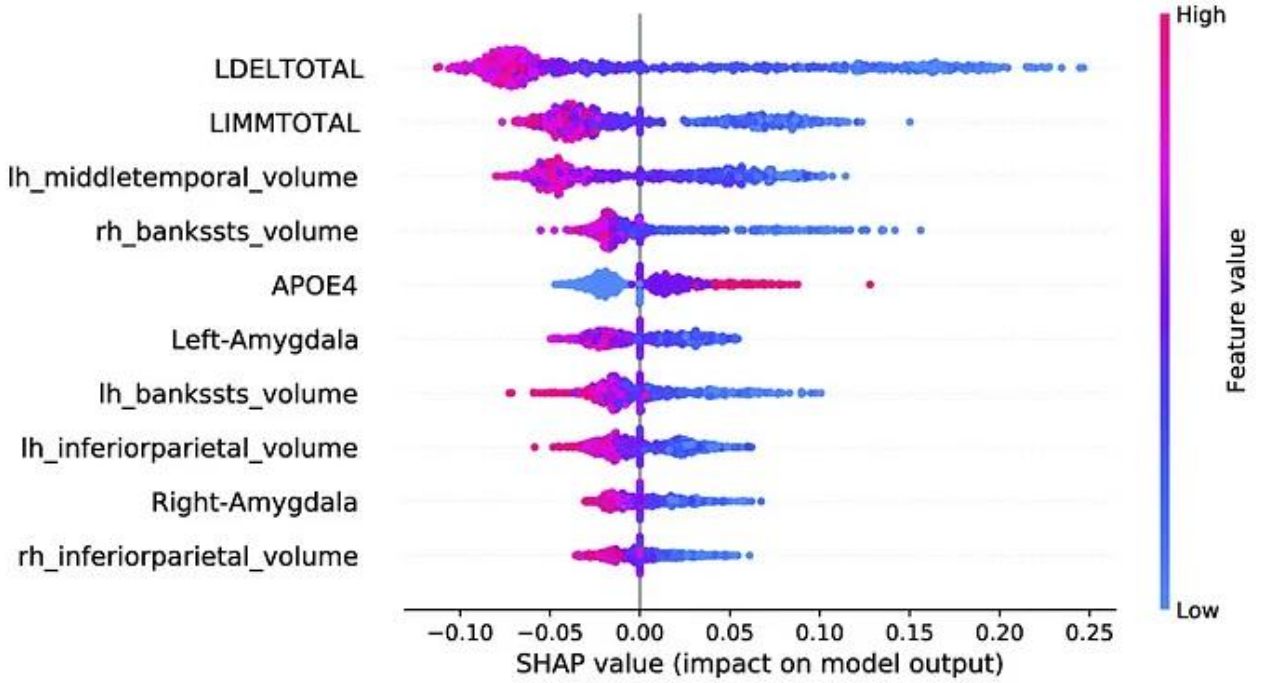


- Imagine you're baking a cake with two key ingredients: flour and sugar.
- The deliciousness of the cake comes from their interaction, but it's hard to say which ingredient “caused” the taste.
- Similarly, in GBTs, when a prediction is influenced by interactions between features, it's unclear which feature is more responsible for the result.



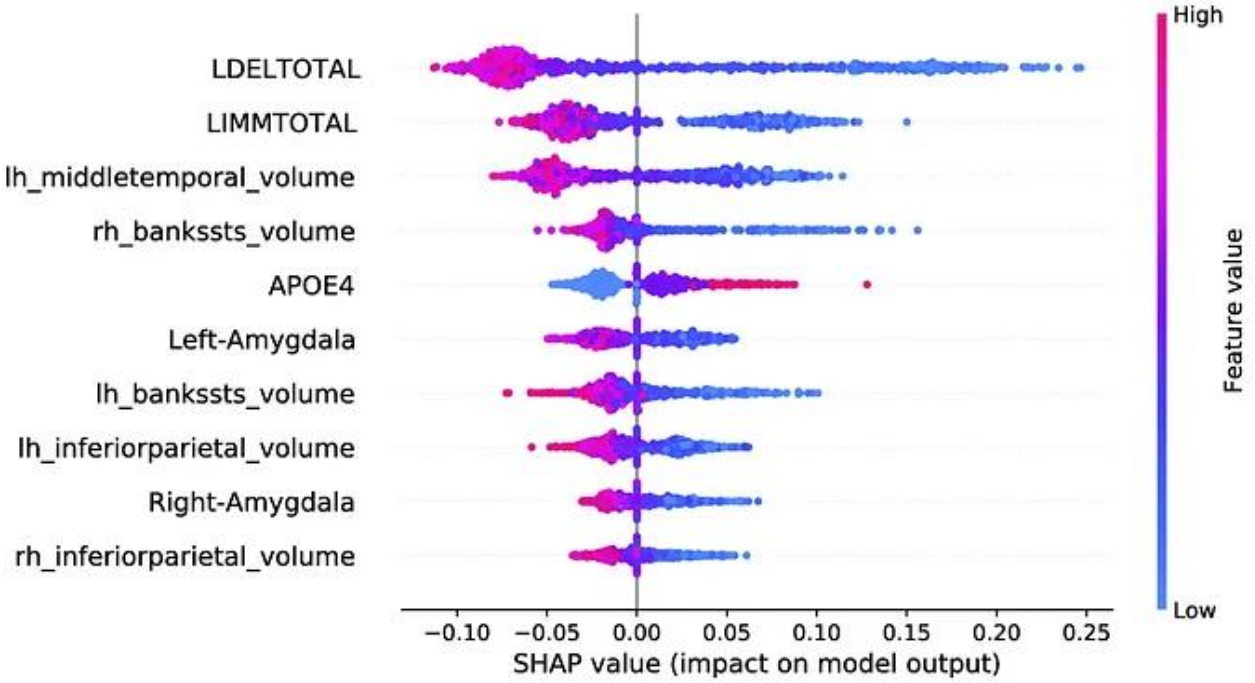
SHAP - SHapley Additive exPlanations

- Shapley values are an extremely popular tool in both economics and explainable AI.
- Shapley values are an approach from game theory to thinking about this divisibility problem.



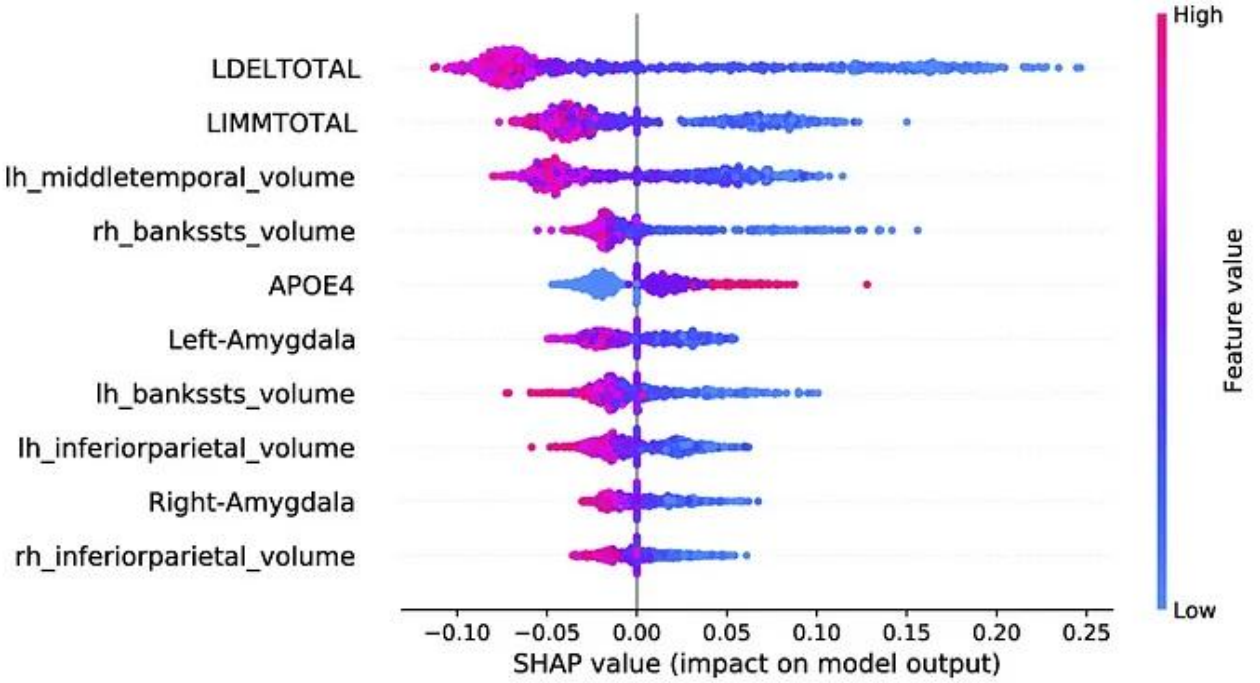
https://www.lesswrong.com/posts/6dixnRRYSLTqCdJzG/understanding-shapley-values-with-venn-diagrams

- On a sunny summer day, you and your two best friends decide to run a lemonade stand. Everyone contributes something special: Emma shares her family's secret recipe, Liam finds premium-quality sugar, and you draw colorful posters.
- The stand is a big hit! The group ends up making $280. But how best to split the profits? Each person contributed in a different way, and the success was clearly due to teamwork…



https://www.lesswrong.com/posts/6dixnRRYSLTqCdJzG/understanding-shapley-values-with-venn-diagrams

- Luckily, Emma has a time machine. She goes back in time — redoing the day with different combinations of team members and recording the profits. This is how each simulation went:



https://www.lesswrong.com/posts/6dixnRRYSLTqCdJzG/understanding-shapley-values-with-venn-diagrams

- Luckily, Emma has a time machine. She goes back in time — redoing the day with different combinations of team members and recording the profits. This is how each simulation went:

Who was involved?	Total profits
Only Emma	$20
Only you	$30
Only Liam	$10
Emma and you	$90
You and Liam	$100
Liam and Emma	$30
Emma, Liam, and you	$280

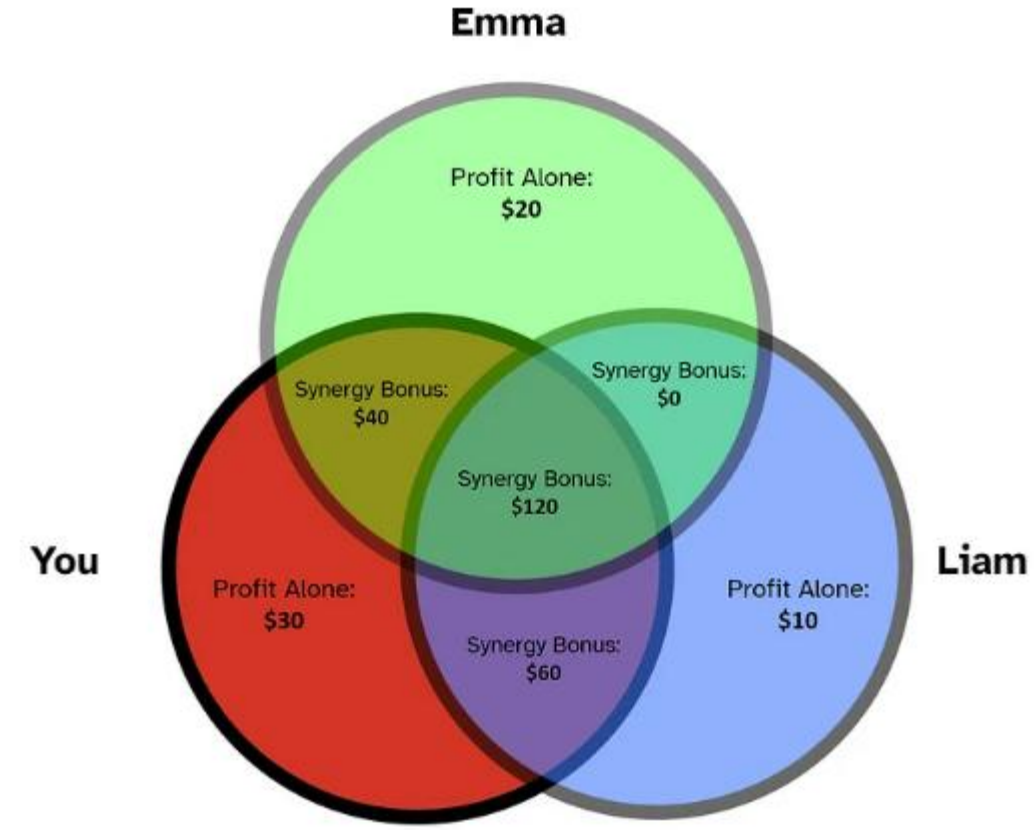
https://www.lesswrong.com/posts/6dixnRRYSLTqCdJzG/understanding-shapley-values-with-venn-diagrams

- Individually, Emma makes 20 dollars running the lemonade stand, and you make 30 dollars. But working together, the team makes 90 dollars.
- The sum of individual profits is 20 + 30 = 50 dollars, which is clearly less than 90 dollars. That extra 90 - 50 = 40 dollars can be attributed to team dynamics.
- In game theory, this bonus is called the "synergy" of you and Emma. Let's visualize our scenario as a Venn diagram.

Who was involved?	Total profits
Only Emma	$20
Only you	$30
Only Liam	$10
Emma and you	$90
You and Liam	$100
Liam and Emma	$30
Emma, Liam, and you	$280

https://www.lesswrong.com/posts/6dixnRRYSLTqCdJzG/understanding-shapley-values-with-venn-diagrams

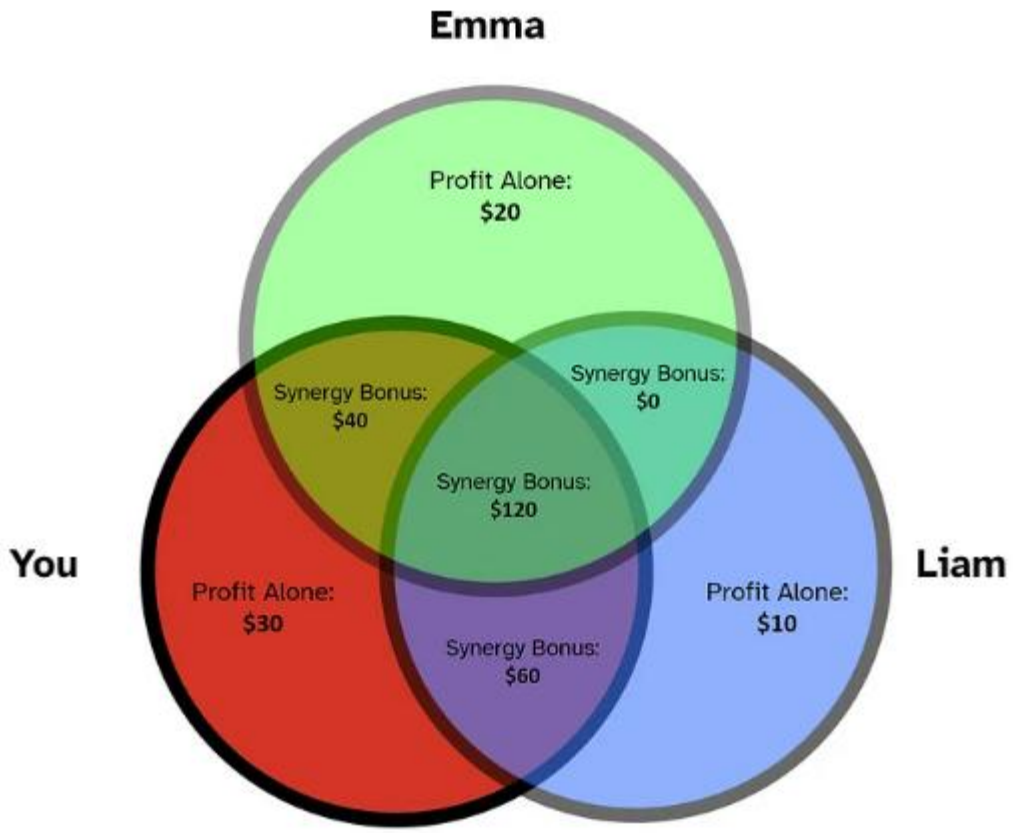
- Individually, Emma makes 20 dollars running the lemonade stand, and you make 30 dollars. But working together, the team makes 90 dollars.
- The sum of individual profits is 20 + 30 = 50 dollars, which is clearly less than 90 dollars. That extra 90 - 50 = 40 dollars can be attributed to team dynamics.
- In game theory, this bonus is called the "synergy" of you and Emma. Let's visualize our scenario as a Venn diagram.



Note: Synergies can also be negative (e.g., if Liam and Emma fought it could hurt profits).

https://www.lesswrong.com/posts/6dixnRRYSLTqCdJzG/understanding-shapley-values-with-venn-diagrams

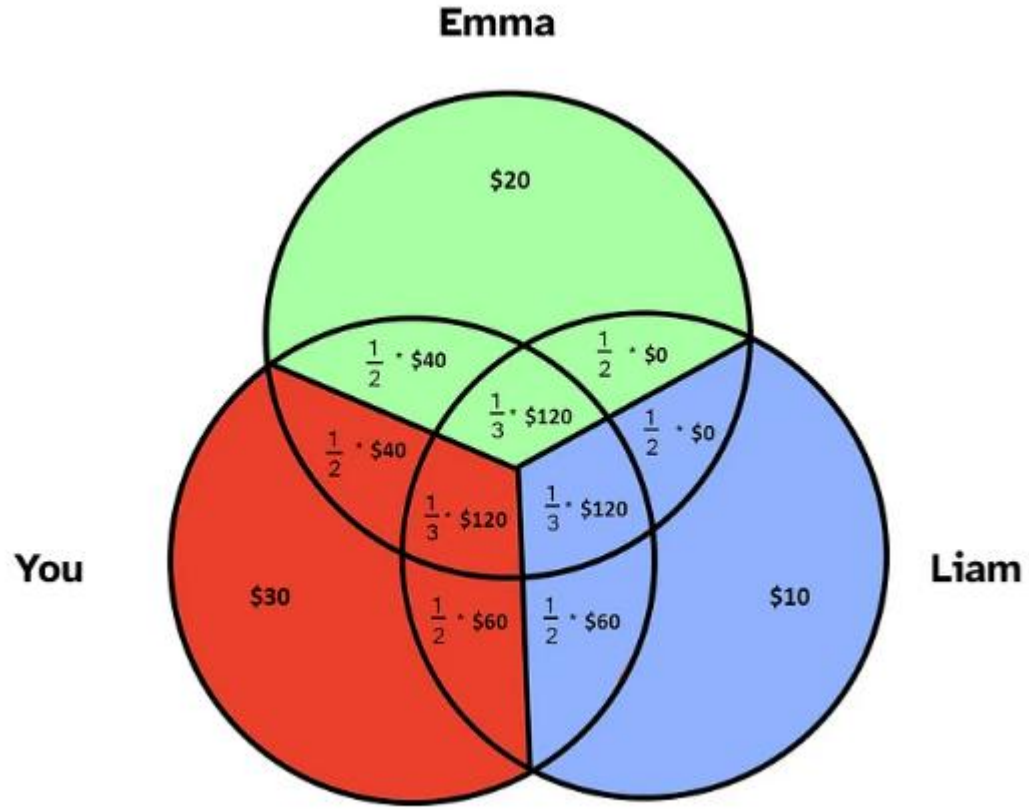
Team members	Sum of synergies	Total profits
Emma	$20	$20
You	$30	$30
Liam	$10	$10
Emma, You	$20 + $30 + $40	$90
You, Liam	$30 + $10 + $60	$100
Liam, Emma	$10 + $20 + $0	$30
Emma, You, Liam	$20 + $30 + $10 + $40 + $60 + $0 + $120	$280



Note: Synergies can also be negative (e.g., if Liam and Emma fought it could hurt profits).

https://www.lesswrong.com/posts/6dixnRRYSLTqCdJzG/understanding-shapley-values-with-venn-diagrams

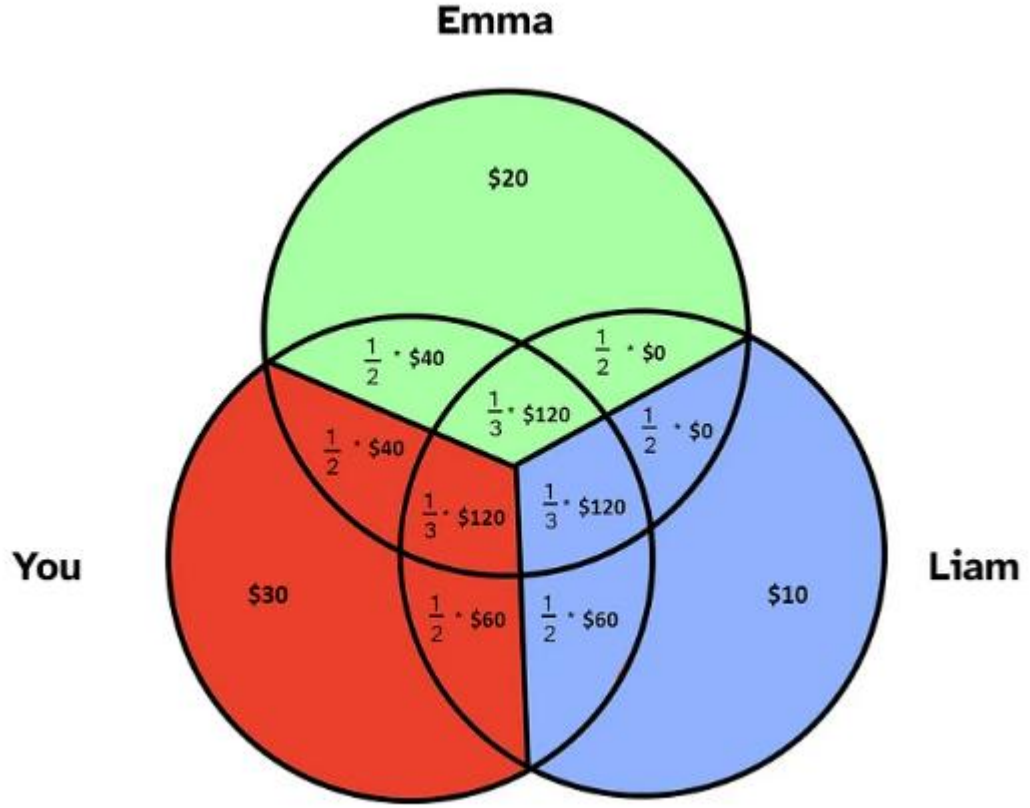
- In this story, the final payouts are the Shapley values of each team member.
- This intuition is all you need to understand Shapley values.



https://www.lesswrong.com/posts/6dixnRRYSLTqCdJzG/understanding-shapley-values-with-venn-diagrams

Shapley values guarantee:

- **Efficiency —** The sum of Shapley values adds up to the total payoff for the full group (in our case, $280).
- **Symmetry —** If two players interact identically with the rest of the group, their Shapley values are equal.
- **Linearity —** If the group runs a lemonade stand on two different days (with different team dynamics on each day), a player's Shapley value is the sum of their payouts from each day.
- **Null player —** If a player contributes nothing on their own and never affects group dynamics, their Shapley value is 0.



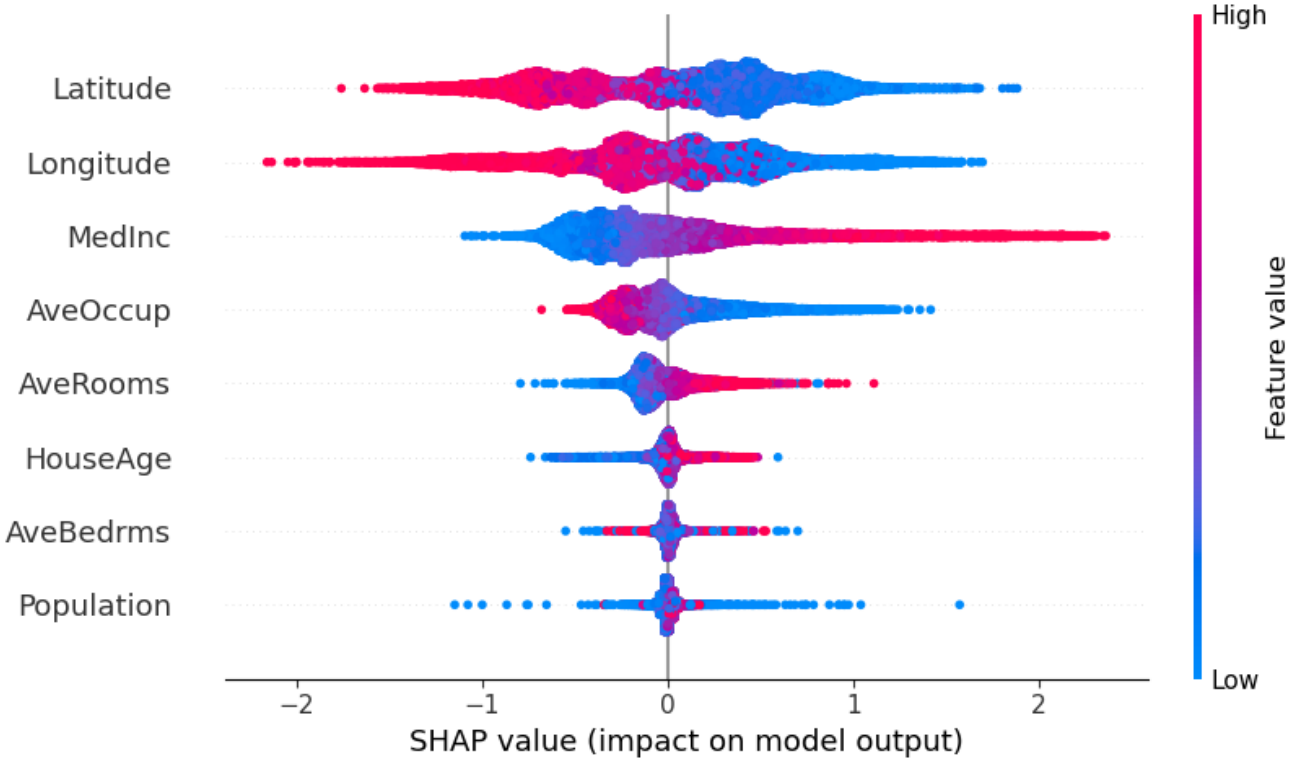
https://www.lesswrong.com/posts/6dixnRRYSLTqCdJzG/understanding-shapley-values-with-venn-diagrams

Players in the Game: Features

In ML, the features of a dataset are treated as players, working together to influence the model's prediction.

Payoff: Model Prediction

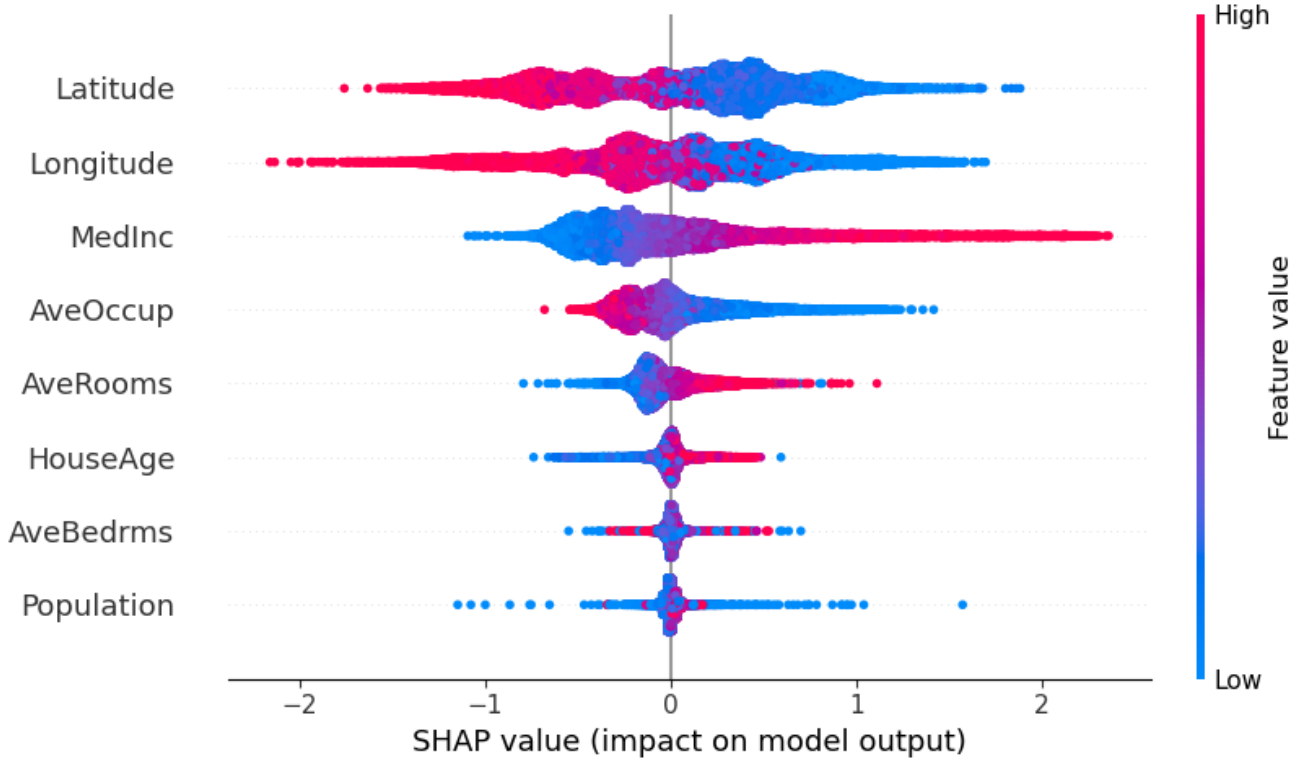
The "payoff" in ML is the predicted outcome (e.g., a classification probability or regression value). Shapley values aim to distribute this payoff fairly among the features based on their contributions.



https://shap.readthedocs.io/en/latest/

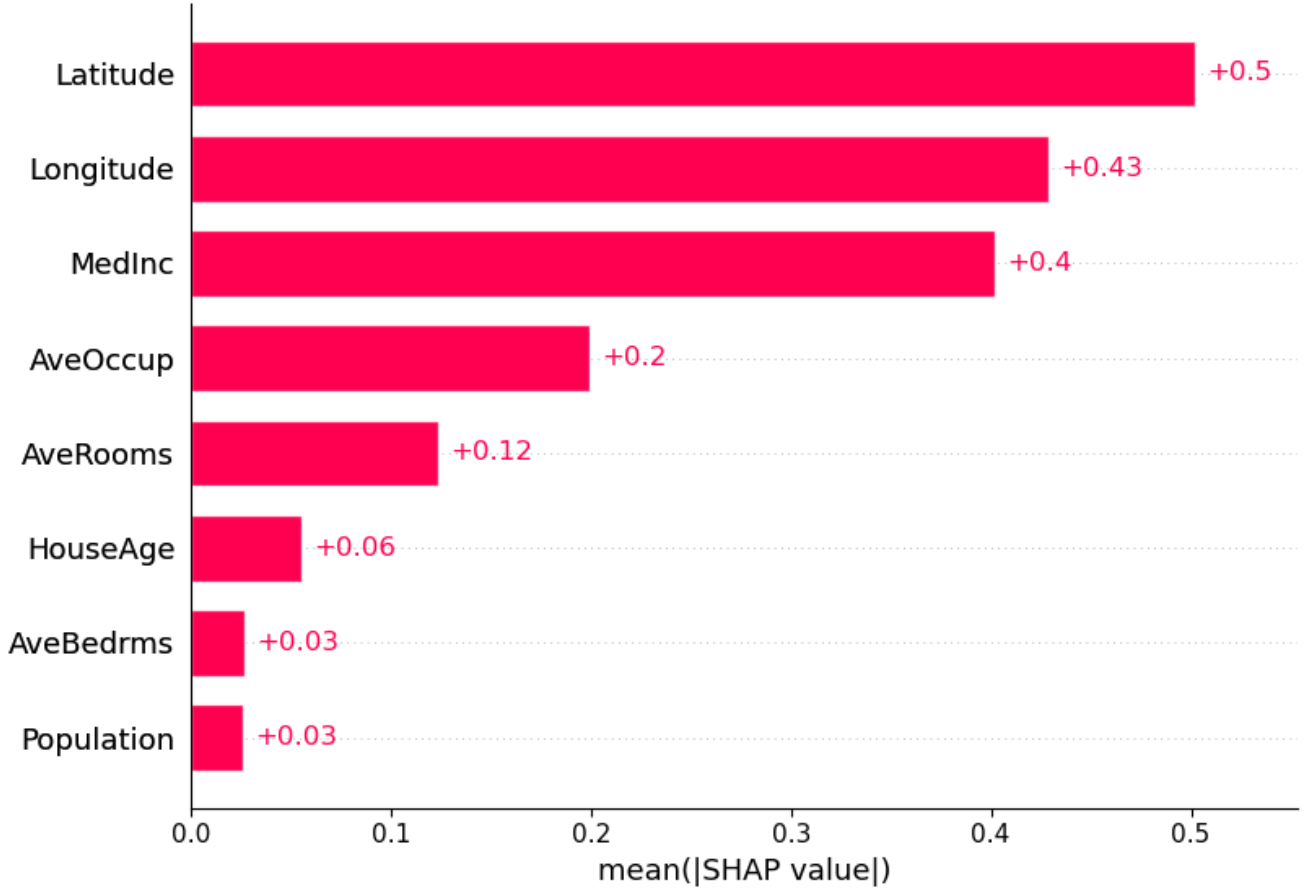
Beeswarm plot-

Each data point in your dataset has a Shapley value for every feature, representing that feature's contribution to the model's prediction for that specific data point.



https://shap.readthedocs.io/en/latest/

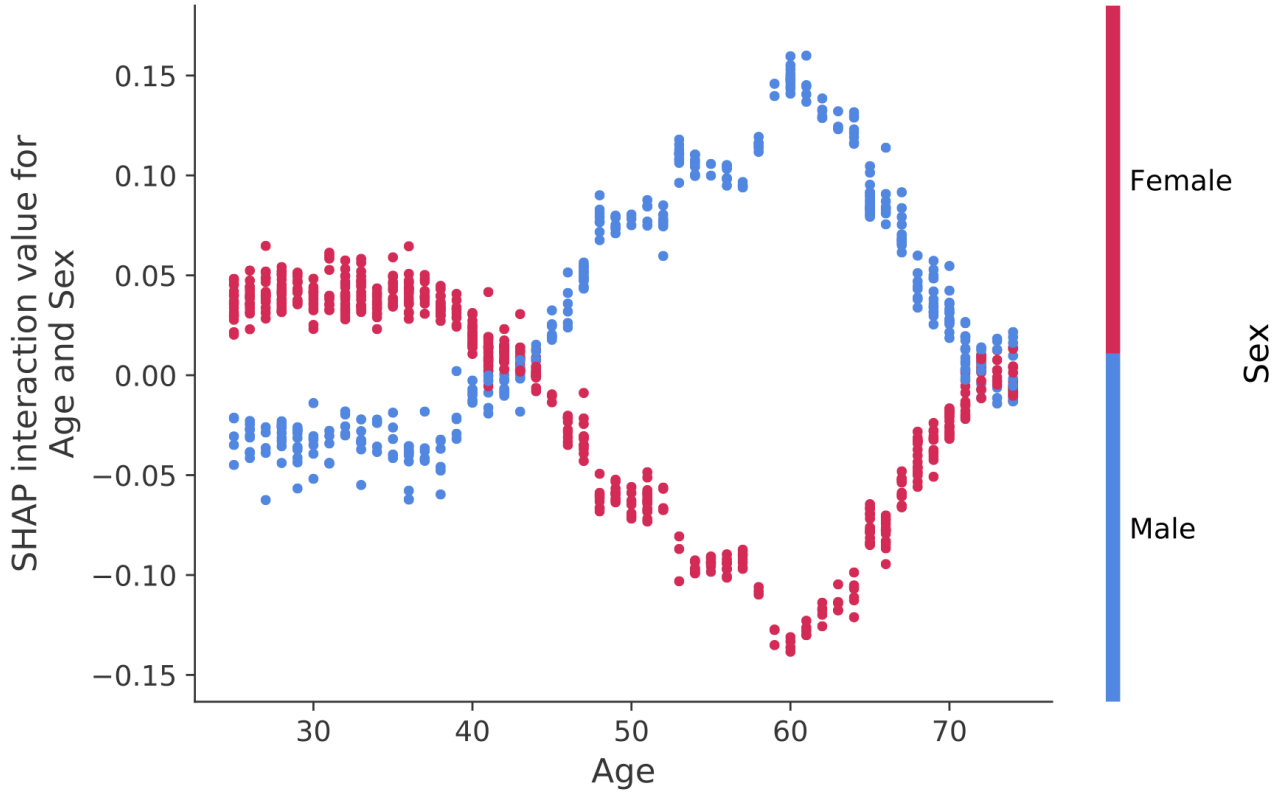
We can also just take the mean absolute SHAP value



https://shap.readthedocs.io/en/latest/

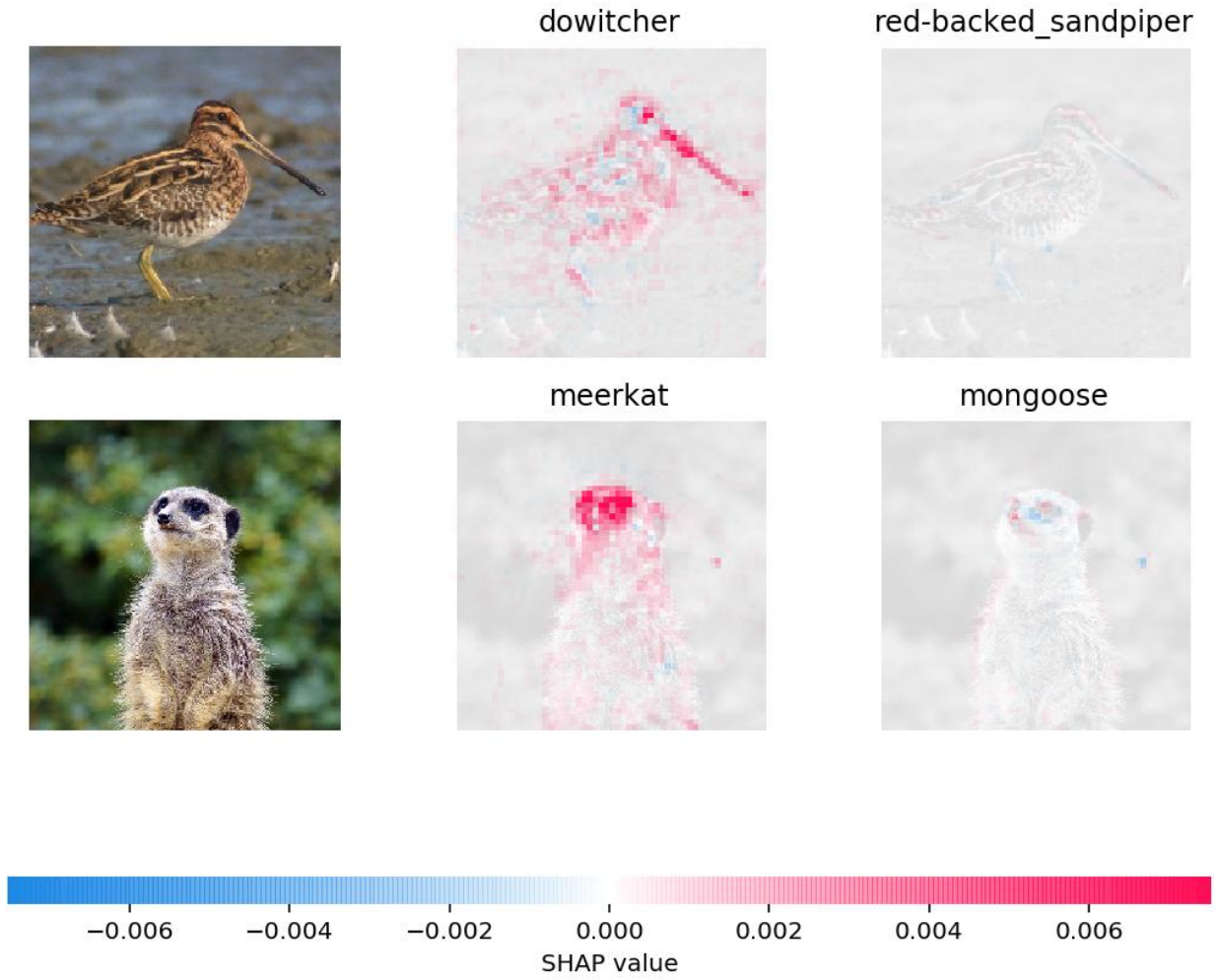
SHAP interaction values are a generalization of SHAP values to higher order interactions. Fast exact computation of pairwise interactions are implemented for tree models.

- This returns a matrix for every prediction, where the main effects are on the diagonal and the interaction effects are off-diagonal.
- These values often reveal interesting hidden relationships, such as how the increased risk of death peaks for men at age 60.



https://shap.readthedocs.io/en/latest/

Used in lots of ML fields



https://shap.readthedocs.io/en/latest/

Lab