Toggle code

Week 9 - Tree-Based Methods

Dr. David Elliott

1. Introduction

Tree-based methods *stratify* or *segment* the predictor space into a number of simple regions[1].

As the spliting rules to make these decision regions can be summerised in a tree structure, these approaches are called *decision trees*.

A decision tree can be thought of as breaking data down by asking a series of questions in order to categorise samples into the same class.

```
Already Cloned
```

Terminology[5]

Root node: no incoming edge, zero, or more outgoing edges.

Internal node: one incoming edge, two (or more) outgoing edges.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

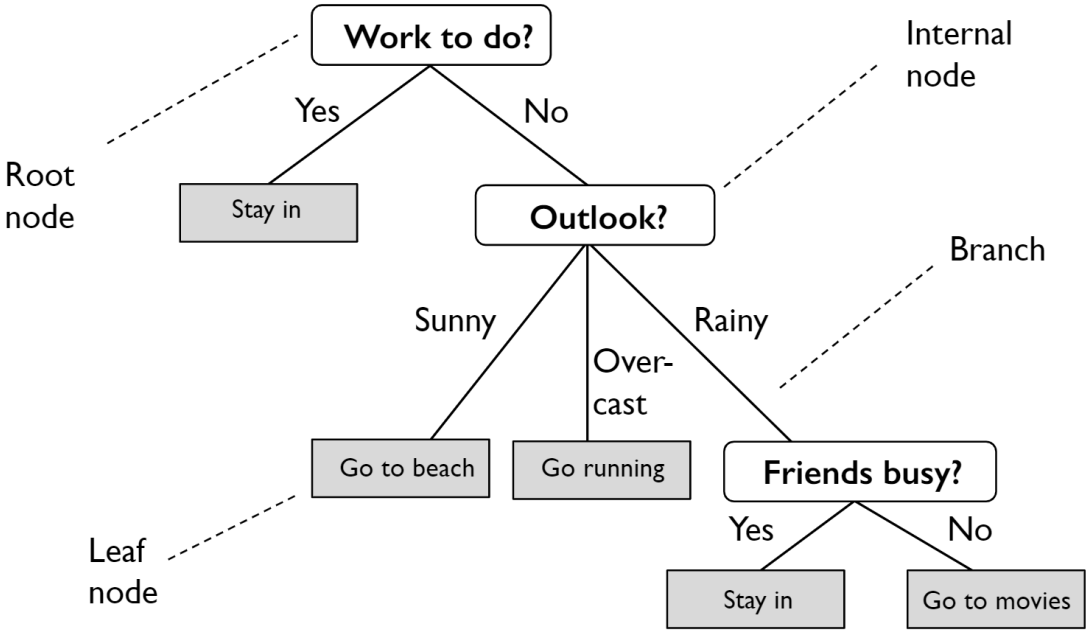
Leaf node: each leaf node is assigned a class label if nodes are pure; otherwise, the class label is determined by majority vote.

Parent and child nodes: If a node is split, we refer to that given node as the parent node, and the resulting nodes are called child nodes.

Notes

- Leaves are typically drawn upside down, so they are at the bottom of the tree

Figure 1: Categorical Decision Tree



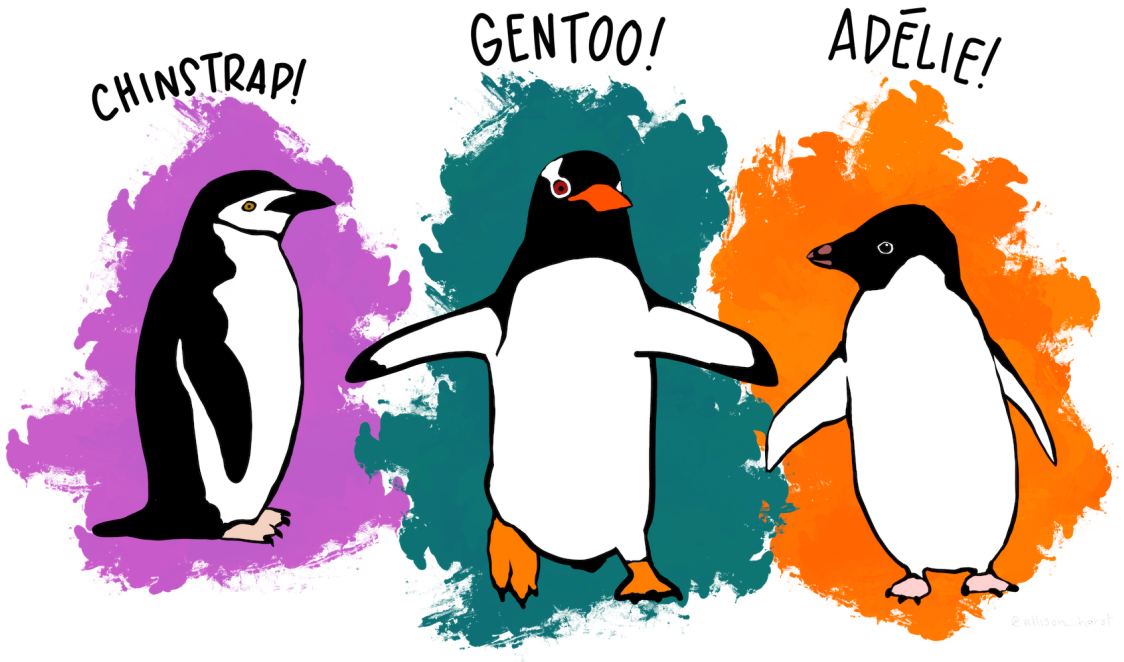
Dataset Example: Penguins

The *"palmer penguins"* dataset[2] contains data for 344 penguins from 3 different species and from 3 islands in the Palmer Archipelago, Antarctica.

Artwork by @allison_horst

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

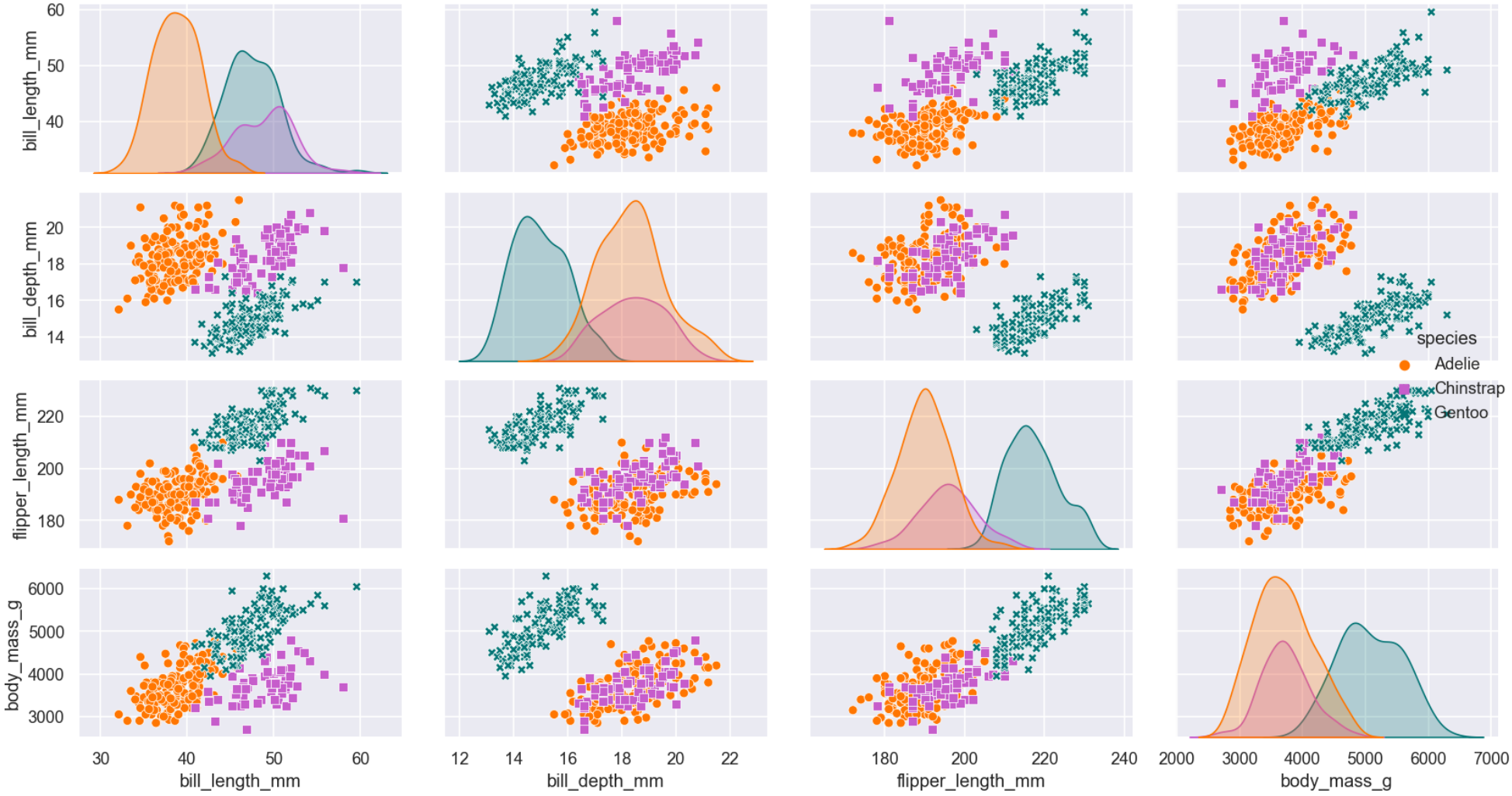
Figure 2: Penguin Buddies



	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	Male
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	Female
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	Female
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	Female

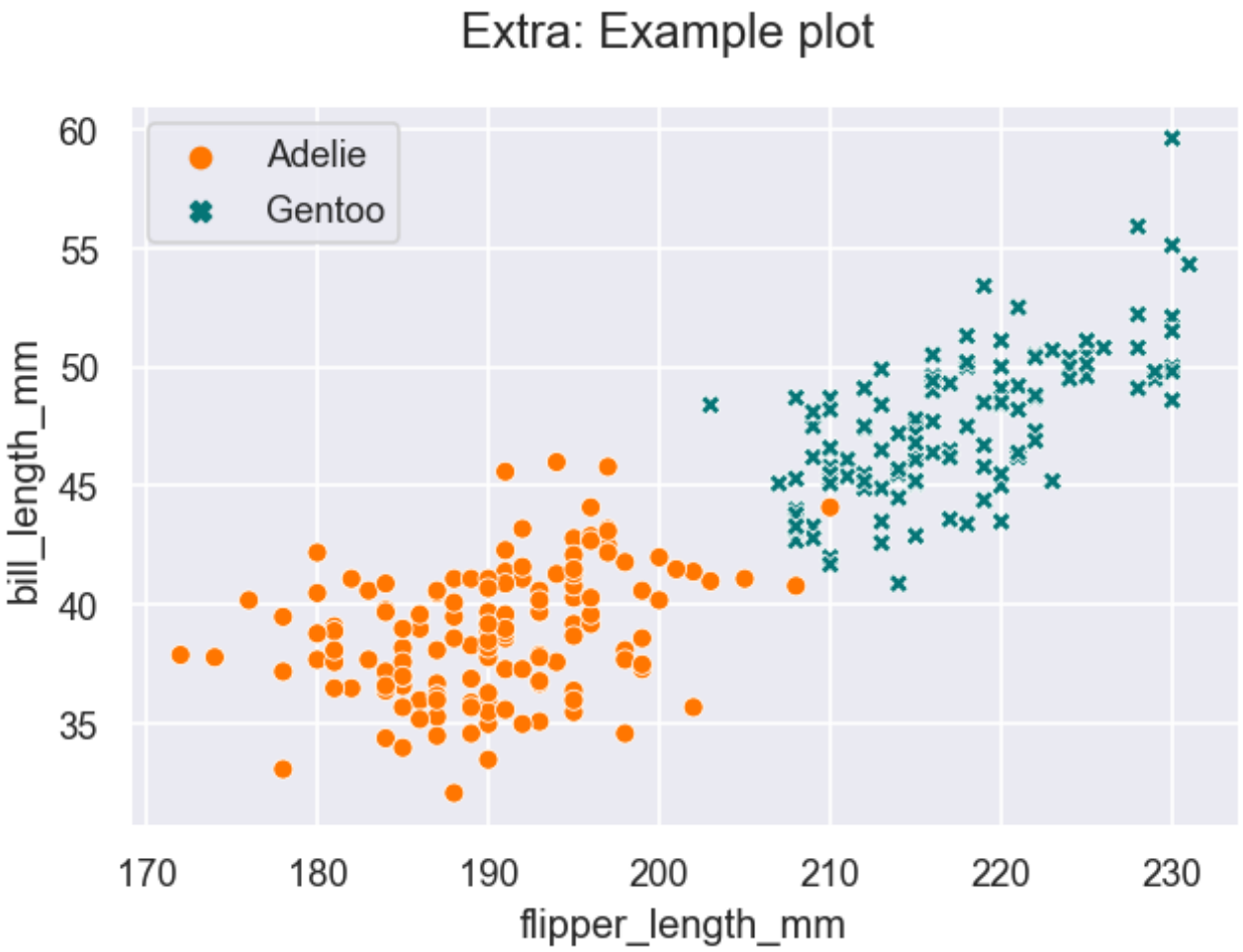
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Figure 3: Penguins Data Pairplot



Create PDF in your applications with the Pdfcrowd HTML to PDF API

PDFCROWD



2. General Decision Tree Algorithm

An algorithm starts at a tree root and then splits the data based on the features that give the best split based on a splitting criterion.

Generally this splitting procedure occours until[3,4]...

- ...all the samples within a given node all belong to the same class,

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

- ...the maximum depth of the tree is reached,
- ...a split cannot be found that improves the model.

NOTES

- The process of growing a decision tree can be expressed as a recursive algorithm as follows[5]:
    1. Pick a feature such that when parent node is split, it results in the largest information gain and stopping if information gain is not positive.
    2. Stop if child nodes are pure or no improvement in class purity can be made.
    3. Go back to step 1 for each of the two child nodes.

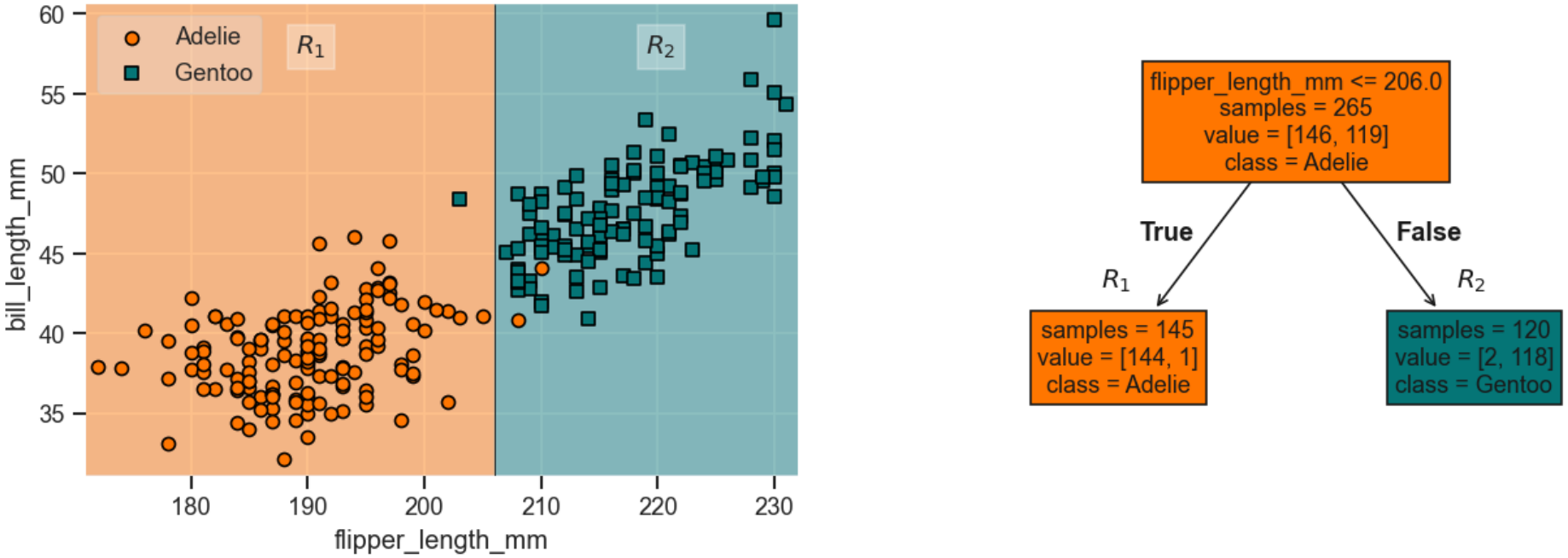
Below is an example of a very shallow decision tree where we have set `max_depth = 1`.

Terminology (Reminder)[1]

- The regions R_1 and R_2 are known as *terminal nodes* or *leaves* of the tree.
- The points where the predictor space is split are known as the *internal nodes*. In this case as we only have one split this is the *"root node"*.
- The segments of the trees that connect the nodes are *branches*.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Figure 4: Shallow Penguins Tree



Extra: `dtreeviz`

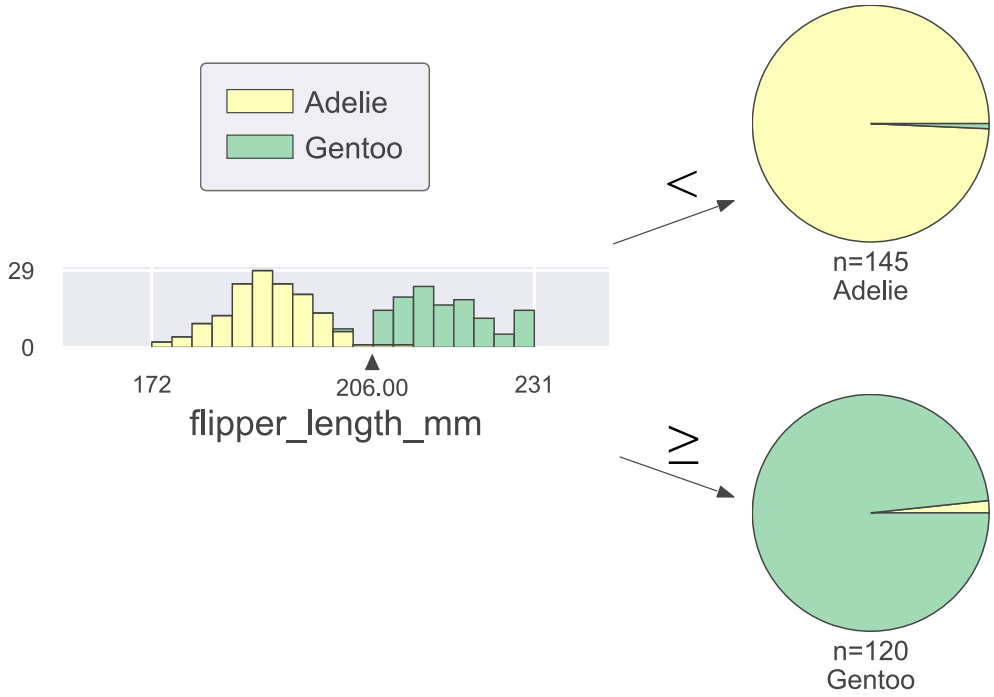
Heres an additional visualisation package with extra features such as bein able to follow the path of a hypothetical test sample.

I don't use `dtreeviz` in the lectures, as it can be a bit of a hassle to setup. However you may also find this a useful way of thinking about the splitting.

Notes

- *"For classifiers, however, the target is a category, rather than a number, so we chose to illustrate feature-target space using histograms as an indicator of feature space distributions."* https://explained.ai/decision-tree-viz/index.html

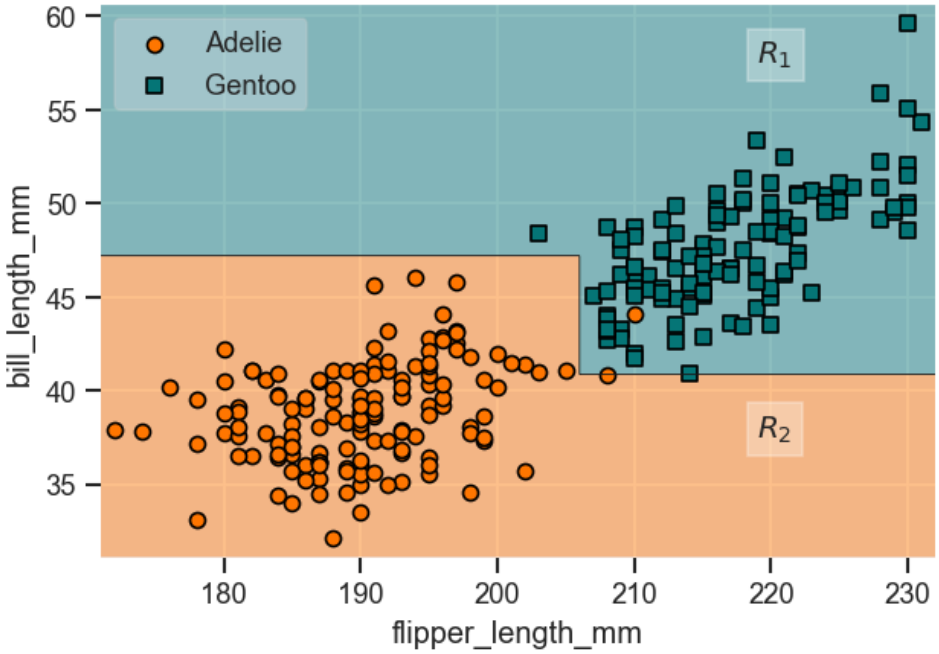
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

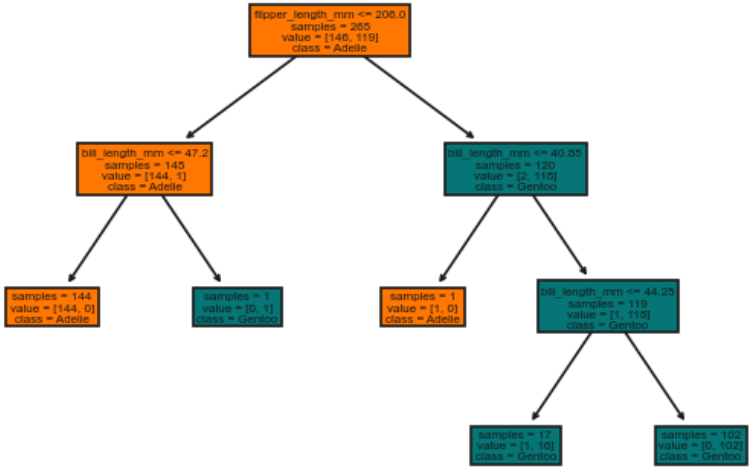


We can make the tree *"deeper"*, and therefore more complex, by setting the `max_depth = 3`.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Figure 5: Penguins Tree (max_depth = 3)





Extrra

You may be wondering, where is the decision boundary part for the node on the 3rd level in figure 5? Why bother doing this split if it doesn't do anything to our decision boundary?

Well, although not contributing to the decision boundry (because either side of the split is still going to be classified as Gentoo) this split does improve our "splitting criterion" (information gain), as the right leaf node is now pure. More on this later in the notes!

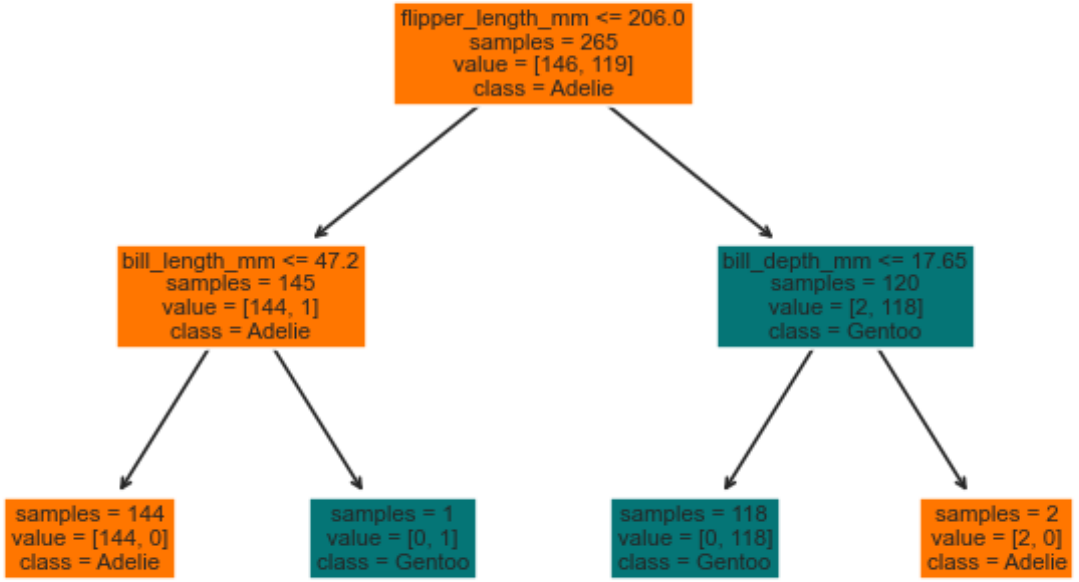
We could also use more than 2 features as seen below.

NOTES

- Although more features are harder to plot decision boundaries (see `dtreeviz`).

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Figure 6: Multiple Features Penguins Tree (max_depth = 3)



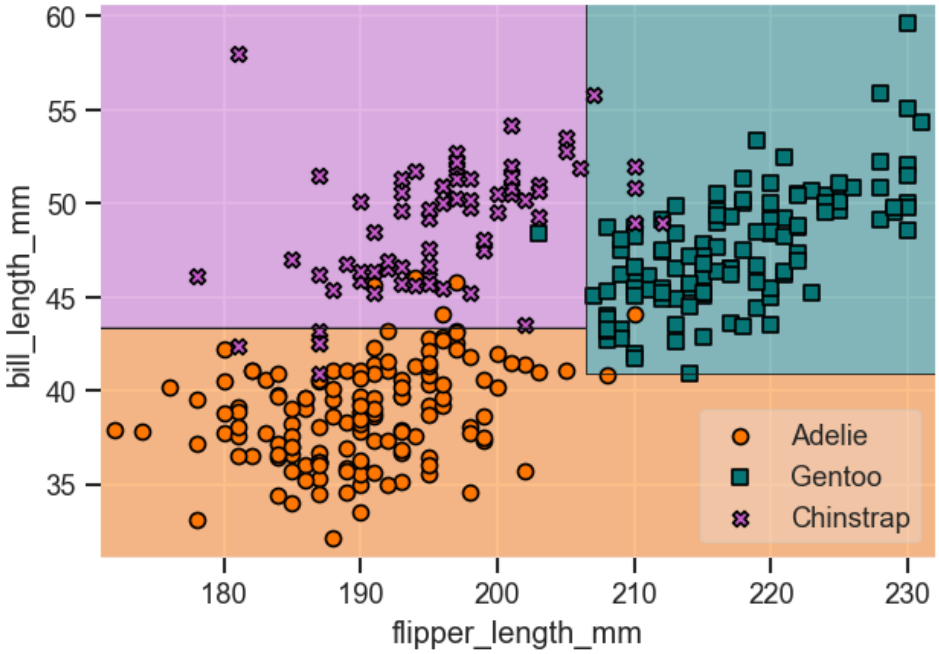
Notes

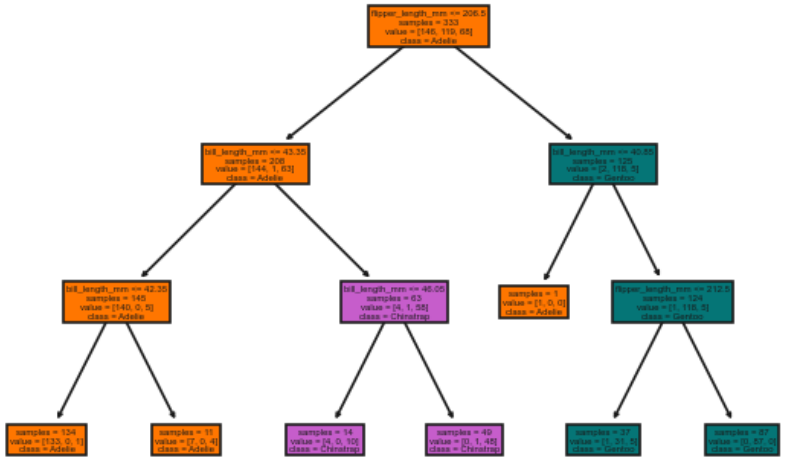
- There is an example of a 3D decision region on pg. 547 in Murphy, K. P. (2012). Machine learning: a probabilistic perspective. MIT press.

We could also also easily extend this to have more than a 2 (binary) class labels.

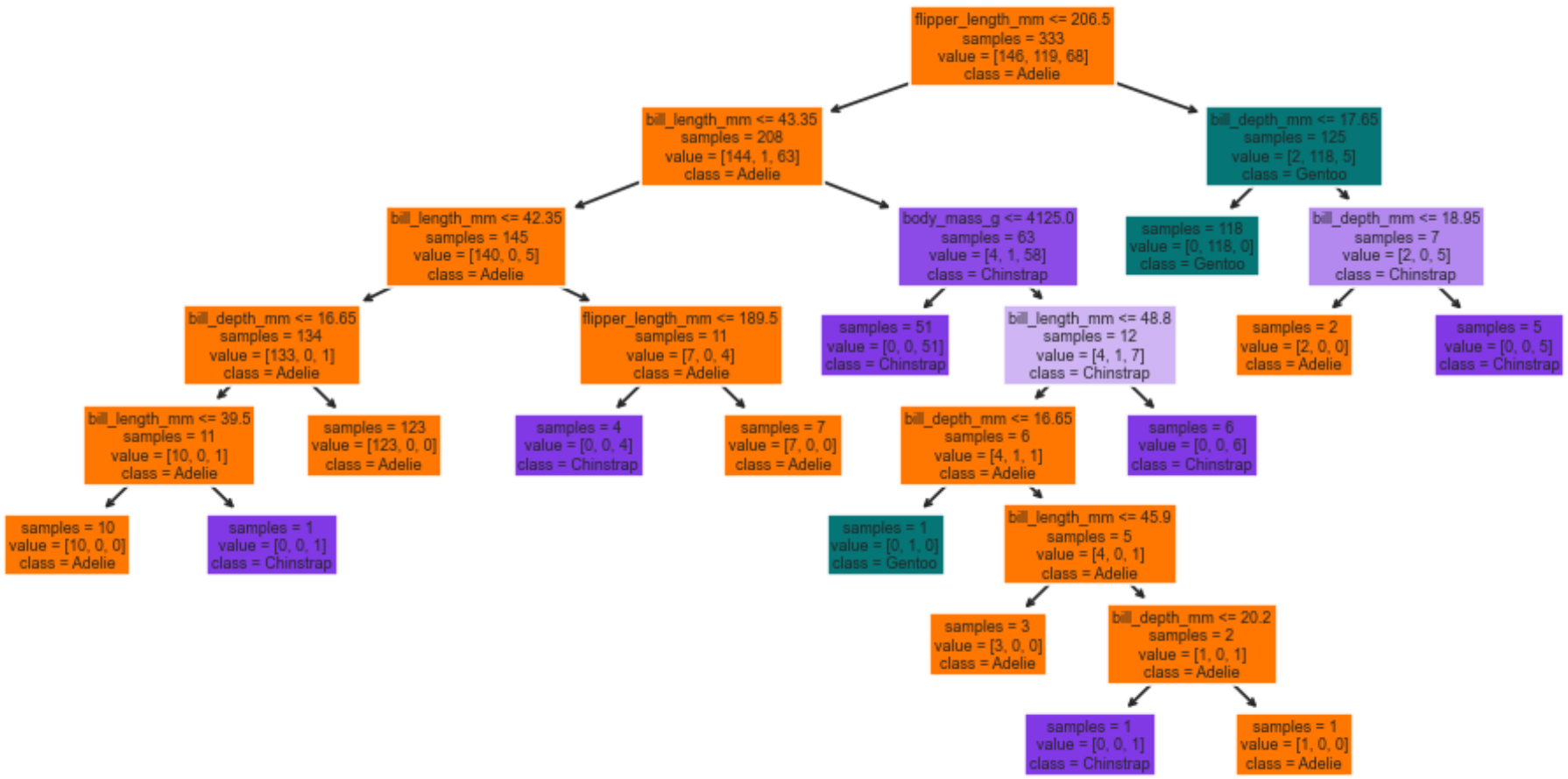
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Figure 7: Multiple Penguin Classes Tree (max_depth = 3)





Create PDF in your applications with the Pdfcrowd HTML to PDF API
PDFCROWD



Estimating Class Probabilities[3]

We can estimate the probability that an instance belongs to a particular class easily.

For our new observation we...

1. traverse the tree to find the leaf node the instance is assigned to,
2. return the ratio of training instances of class c in that node,

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

3. assign our observation to the class with the highest probability.

Example

Using the model in figure 7, we could find the probability of the following penguins species:

	bill_length_mm	flipper_length_mm
65	41.6	192.0

	Adelie	Gentoo	Chinstrap
0	0.0	0.0204	0.9796

```
Predicted Species is Chinstrap
```

In a general sense this approach is pretty simple, however there are a number of design choices and considerations we have to make including[5]:

- How do we decide which features to select for splitting a parent node into child nodes?
- How do we decide where to split features?
- When do we stop growing a tree?
- How do we make predictions if no attributes exist to perfectly separate non-pure nodes further?

3. Specific Decision Tree Algorithms

Most decision tree algorithms address the following implimentation choices differently[5]:

- **Splitting Criterion:** Information gain, statistical tests, objective function, etc.
- **Number of Splits:** Binary or multi-way.
- **Variables:** Discrete vs. Continuous.
- **Pruning**: Pre- vs. Post-pruning.

There are a number of decision tree algorithms, prominant ones include:

- Iterative Dichotomizer 3 (ID3)

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

- C4.5
- **CART**

Notes

- Binary means *nodes* always have two children.

CART

Scikit-Learn uses an optimised version of the Classification And Regression Tree (CART) algorithm.

- **Splitting Criterion:** Information gain
- **Number of Splits:** Binary
- **Independent Variables (Features):** Continuous
- **Dependent variable**: Continuous or Categorical
- **Pruning**: Pre- & Post-pruning

Notes

- *"scikit-learn uses an optimised version of the CART algorithm; however, scikit-learn implementation does not support categorical variables for now."* https://scikit-learn.org/stable/modules/tree.html

Information Gain[4]

An algorithm starts at a tree root and then splits the data based on the feature, f, that gives the largest information gain, IG.

To split using information gain relies on calculating the difference between an impurity measure of a parent node, D_p, and the impurities of its child nodes, D_j; information gain being high when the sum of the impurity of the child nodes is low.

We can maximise the information gain at each split using,

$$IG(D_p, f) = I(D_p) - \sum_{j=1}^{m} \frac{N_j}{N_p} I(D_j),$$

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

where I is out impurity measure, N_p is the total number of samples at the parent node, and N_j is the number of samples in the jth child node.

Some algorithms, such as Scikit-learn's implimentation of CART, reduce the potential search space by implimenting binary trees:

$$IG(D_p, f) = I(D_p) - \frac{N_{\text{left}}}{N_p} I(D_{\text{left}}) - \frac{N_{\text{right}}}{N_p} I(D_{\text{right}}).$$

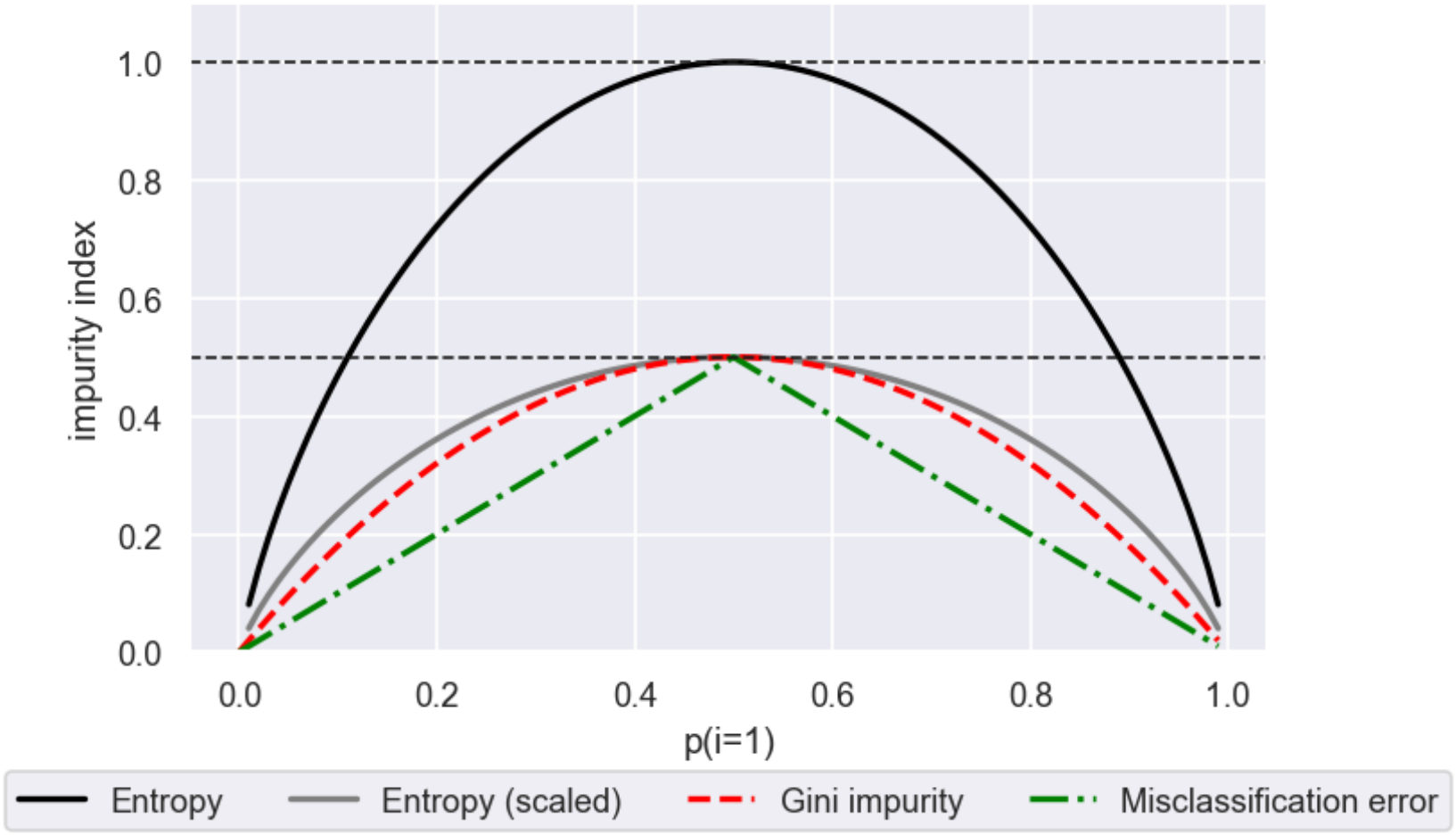
Notes

- The CART algorithm is *greedy* - meaning it searches for the optimum split at each level. It does not check if this is the best split to improve impurity further down the tree[3].
- To find the optimal tree is known as an *NP-Complete* problem, meaning it is intractable even for small training sets[3].

Three impurity measures that are commonly used in binary decision trees are the *classification error* (I_E), *gini impurity* (I_G), and *entropy* (I_H)[4].

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Figure 8: Impurity Measures



Classification Error[4]

This is simply the fraction of the training observations in a region that does belongs to the most common class:

$$I_E = 1 - \max\{p(i|t)\}$$

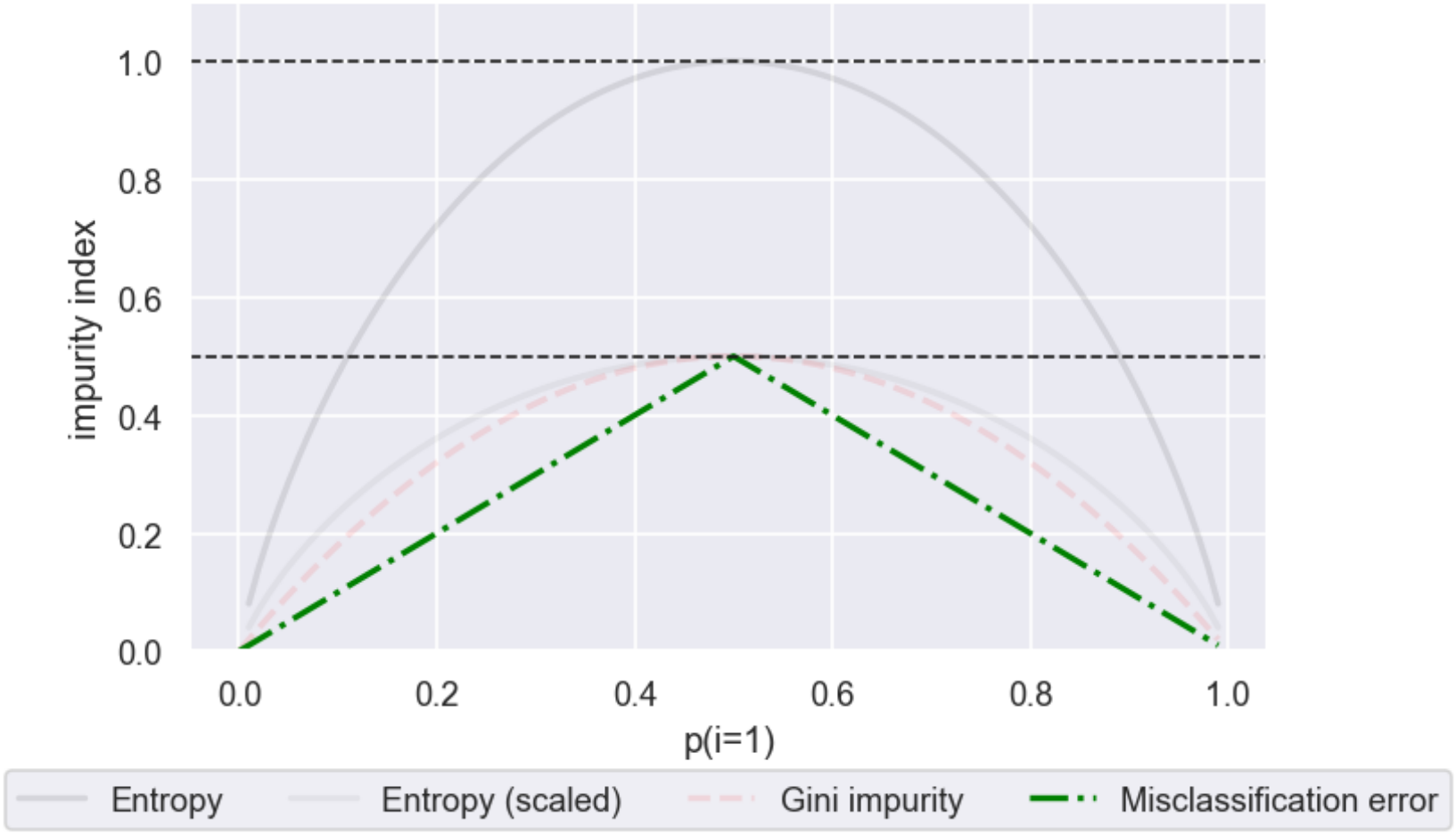
Here $p(i|t)$ is the proportion of the samples that belong to the ith class c, for node t.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Notes

- Another way of writing this is $E = 1 - \max_k(\hat{p}_{mk})$, where $\hat{p}_{mk}$ is the proportion of training observations in the mth region that are from the kth class[1].

Figure 9: Classification Error



Entropy Impurity[4]

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

For all non-empty classes ($p(i|t) \neq 0$), entropy is given by

$$I_H = -\sum_{i=1}^{c} p(i|t) \log_2 p(i|t).$$

The entropy is therefore 0 if all samples at the node belong to the same class and maximal if we have a uniform class distribution.

For example in binary classification ($c = 2$):

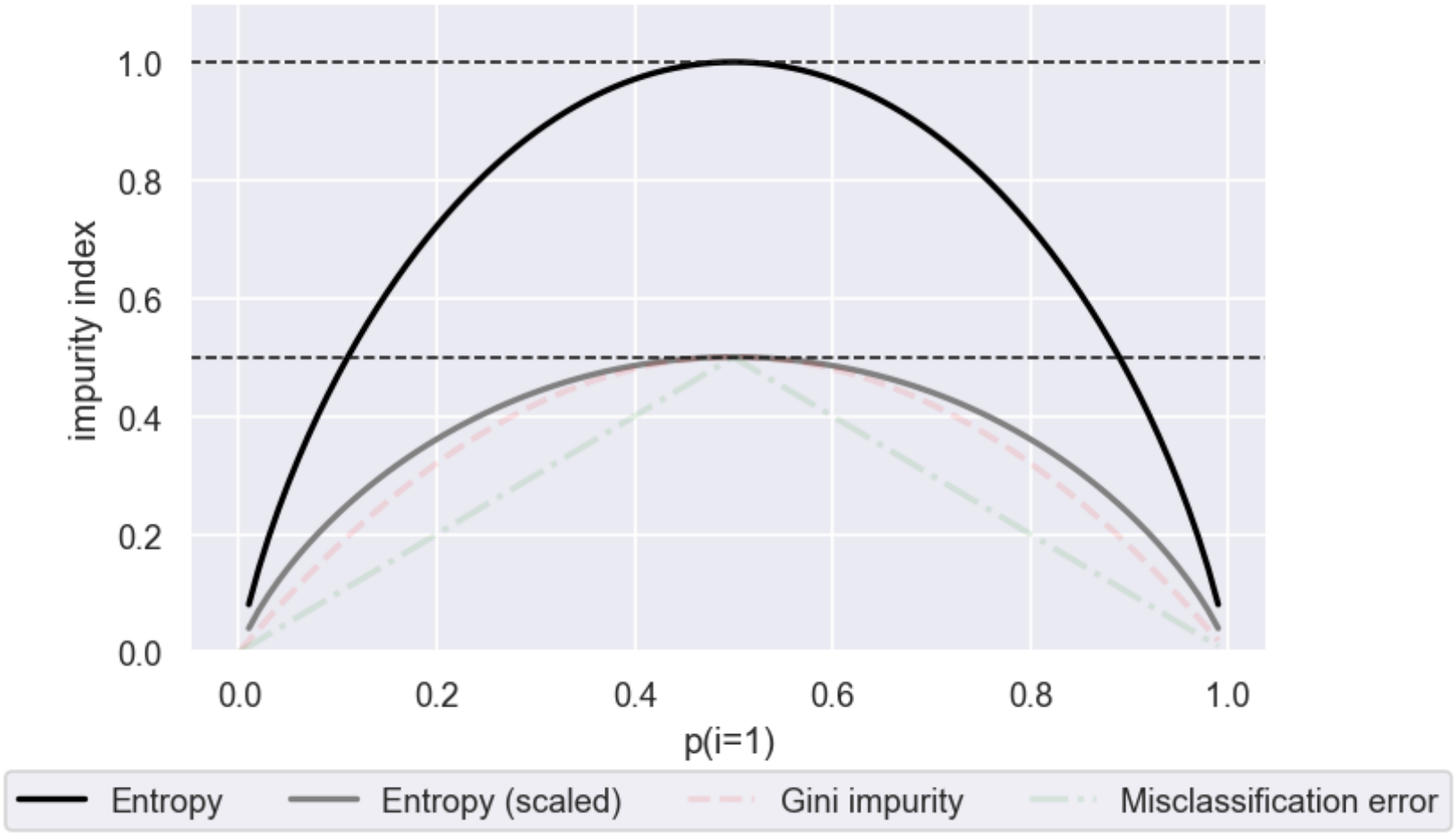
- $I_H = 0$ if $p(i = 1|t) = 1$ or $p(i = 0|t) = 0$
- $I_H = 1$ if $p(i = 1|t) = 0.5$ or $p(i = 0|t) = 0.5$

Notes

- In binary classification, entropy reaches its minimum (0) when all examples in a given node have the same label; on the other hand, the entropy is at its maximum of 1 when exactly one-half of examples a node are labeled with 1, which would be useless for classification.[6]
- Another way of writing this is[1] $D = -\sum_{k=1}^{K} \hat{p}_{mk} log \hat{p}_{mk}$.
  - Entropy will take on a value near 0 if the $\hat{p}_{mk}$'s are all near 0 or 1, therefore will take on a small value if the mth node is pure.

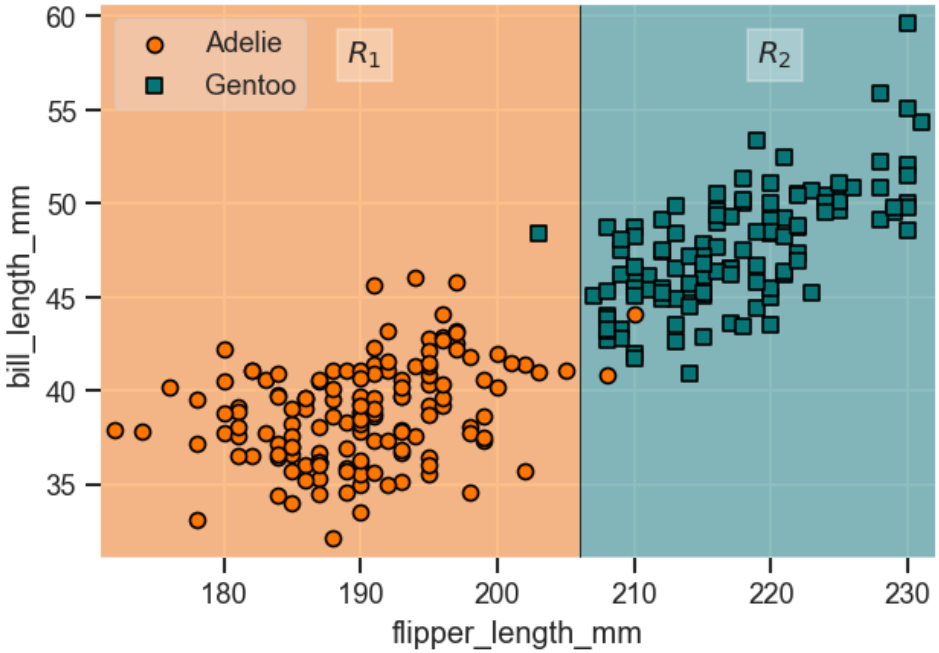
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

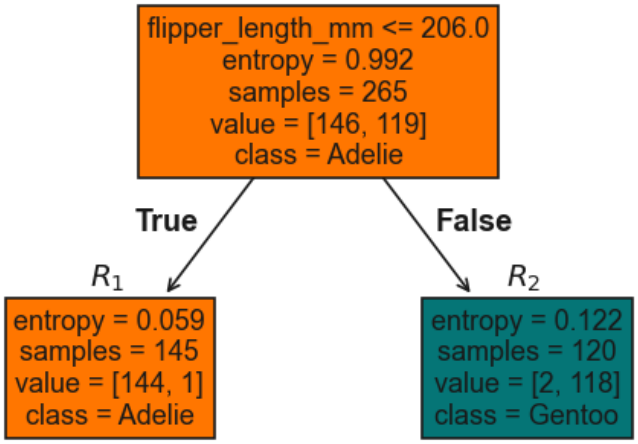
Figure 10: Entropy Impurity



Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Extra: Shallow Tree with Entropy





Gini Impurity[4]

Gini Impurity is an alternative measurement, which minimises the probabilty of misclassification,

$$I_G(t) = \sum_{i=1}^{c} p(i|t)(1 - p(i|t)) \tag{1}$$

$$= 1 - \sum_{i=1}^{c} p(i|t)^2. \tag{2}$$

This measure is also maximal when classes are perfectly mixed (e.g. $c = 2$):

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

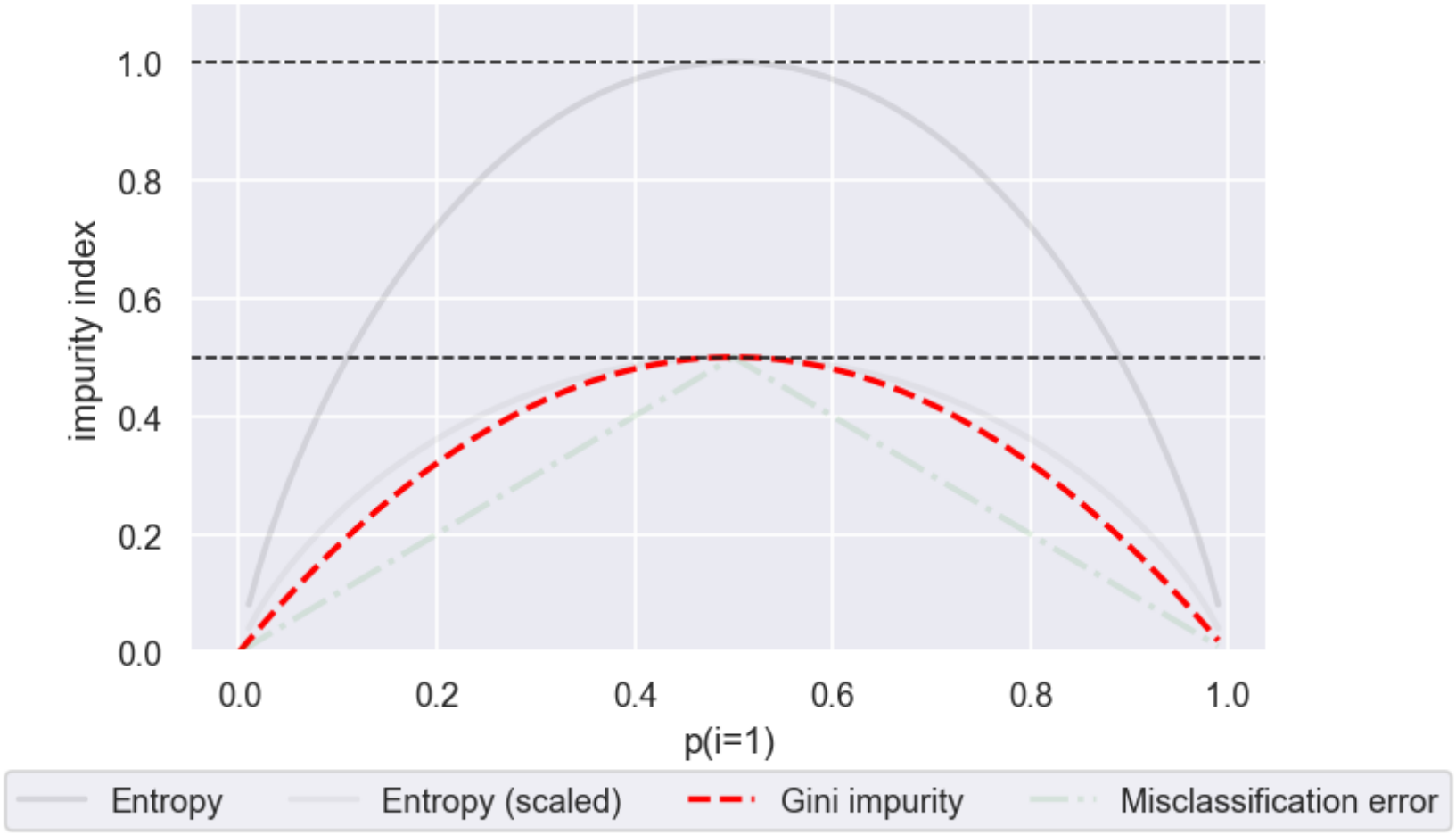
$$I_G(t) = 1 - \sum_{i=1}^{c} 0.5^2 = 0.5. \tag{3}$$

Notes

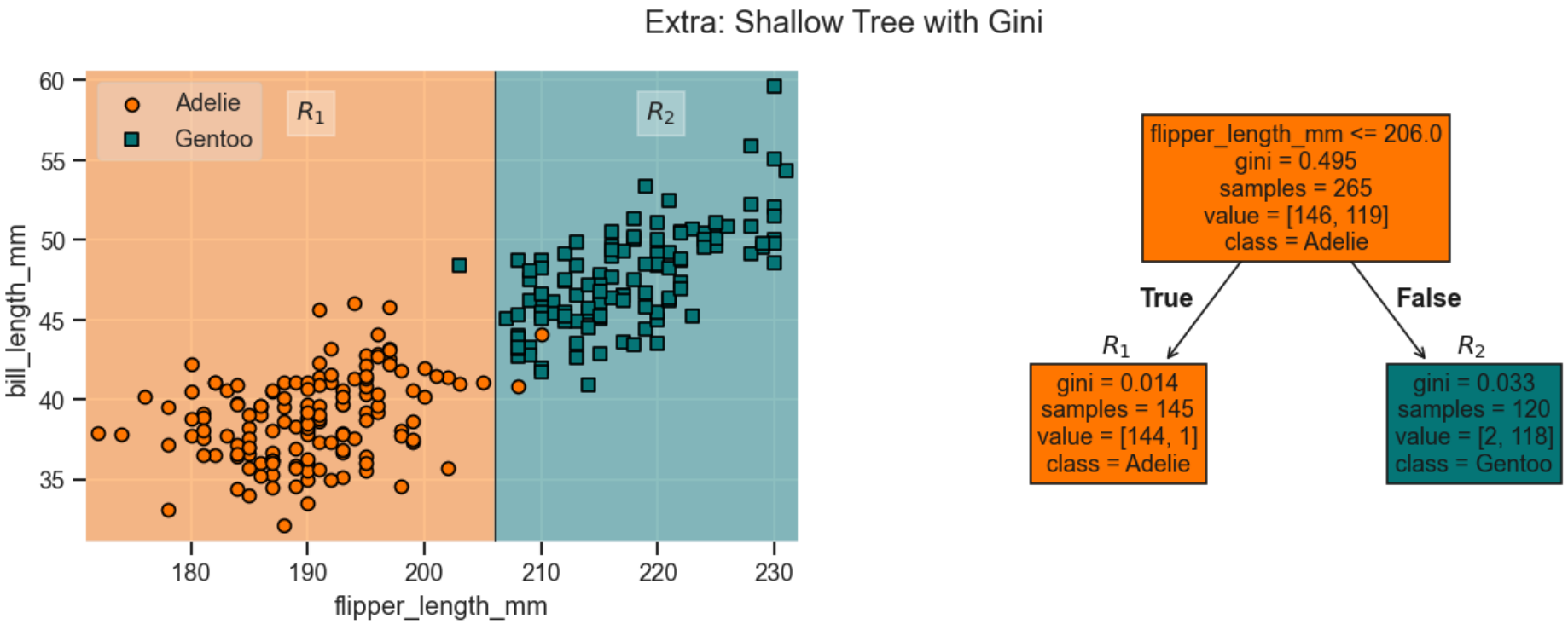
- It is also a measure of node *"purity"* as a small value indicates a node contains predominantly observations from a single class.
- Another way of writing this is[1] for K classes, $G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$.
  - It takes a small value if all of the $\hat{p}_{mk}$'s are close to 0 or 1.
- Whether we use entropy or Gini impurity generally does not really matter, because both have the same concave/bell shape.
  - Gini is computationally more efficient to compute than entropy due to the lack of the log.
  - When they do differ, Gini is more likely to isolate the most frequenct class to its own branch and entropy produce slightly more ballanced trees[3].

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Figure 11: Gini Impurity



Create PDF in your applications with the Pdfcrowd HTML to PDF API
PDFCROWD



Why not Classification Error?

Classification Error is rarely used for information gain in practice.

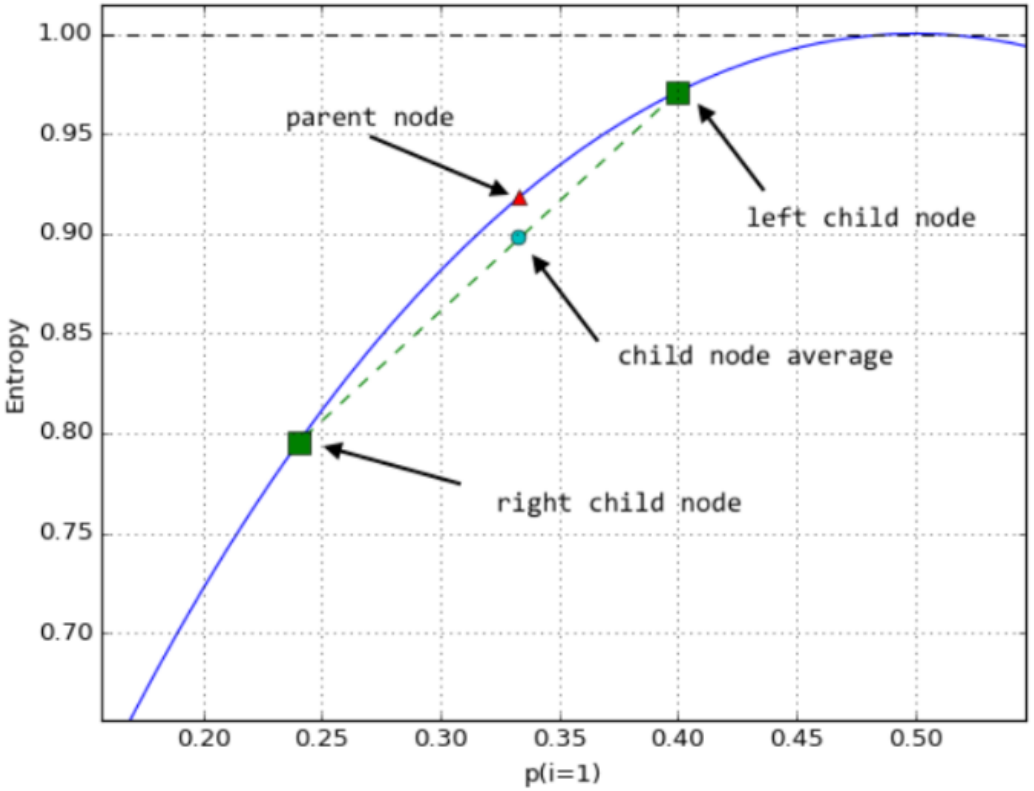
This is because it can mean that tree growth gets stuck and error doesnt improve, this is not the case for a concave function such as entropy or gini.

Notes

- Classification Error its not even an option in `scikit-learn`.
- Another example of this is given in exersise 3.

Figure 12: Child Node Averages

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



Feature Importance[10,11]

Decision trees allow us assess the *importance* of each feature for classifying the data,

$$fi_j = \frac{\sum_{t \in s} ni_t}{\sum_t^m ni_t}$$

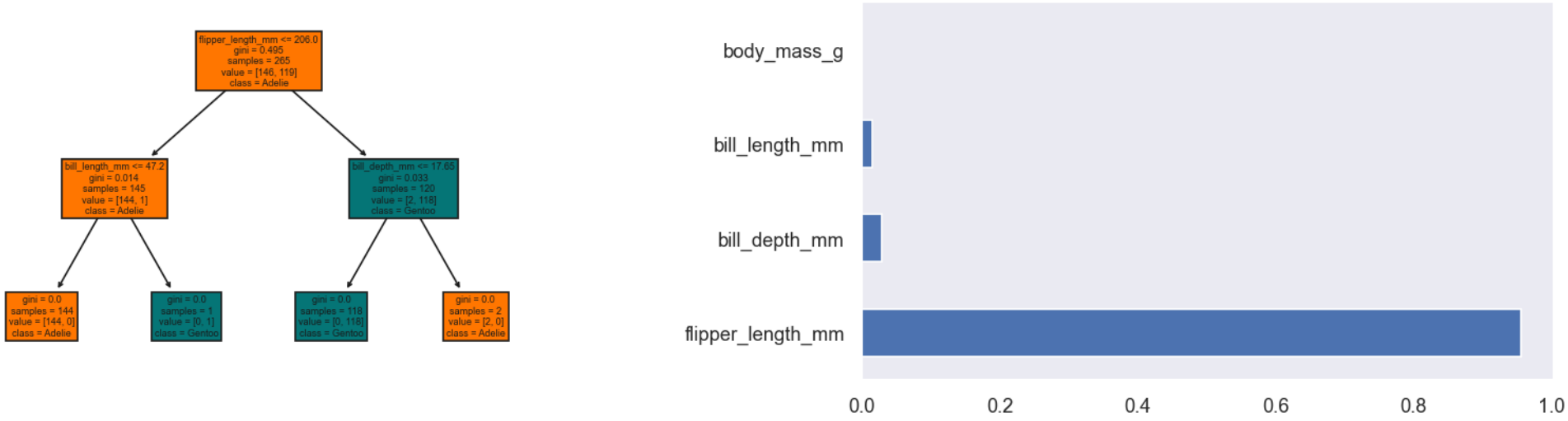
where ni_t is the tth nodes importance, and s are the indices of nodes that split on feature fi_j.

We often assess the normalized total reduction of the criterion (e.g. Gini) brought by that feature,

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

$$norm fi_j = \frac{fi_j}{\sum_j^p fi_j}.$$

Figure 13: Feature Importances for Classifying Adelie and Gentoo Penguins



Pruning

Question: When do we stop growing a tree?

Occam's razor: Favor a simpler hypothesis, because a simpler hypothesis that fits the data equally well is more likely or plausible than a complex one[5].

To minimize overfitting, we can either set limits on the trees before building them (pre-pruning), or reduce the tree by removing branches that do not significantly contribute (post-pruning).

NOTES

- In other words, if decision trees are not pruned, they have a high risk of overfitting to the training data[5].

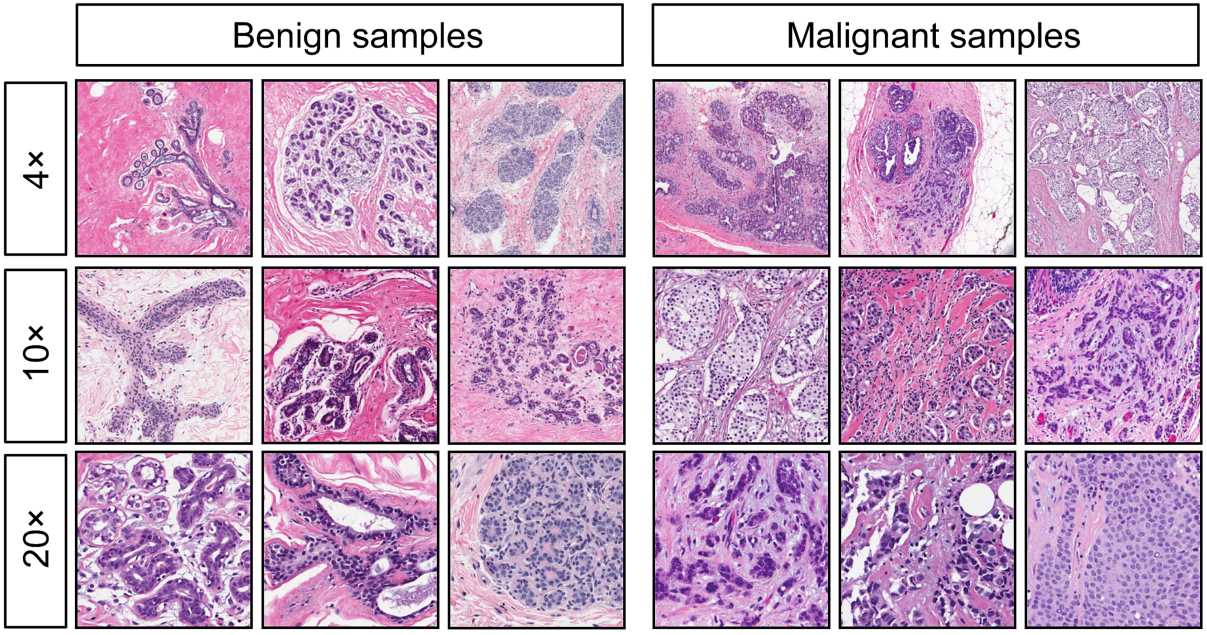
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Dataset Example: Breast Cancer Wisconsin Dataset[12]

Digitized image of a fine needle aspirate of a breast mass. The features describe characteristics of the cell nuclei present in the image.

The dataset was created from digitized images of healthy (benign) and cancerous (malignant) tissues.

Image from Levenson et al. (2015), PLOS ONE, doi:10.1371/journal.pone.0141357.



Notes

- Fine needle aspiration is a type of biopsy procedure.

Extra

You can explore the data below although I recomend limiting the number of features to plot for ease of viewing.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 30 columns):
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   mean radius             569 non-null    float64
 1   mean texture            569 non-null    float64
 2   mean perimeter          569 non-null    float64
 3   mean area               569 non-null    float64
 4   mean smoothness         569 non-null    float64
 5   mean compactness        569 non-null    float64
 6   mean concavity          569 non-null    float64
 7   mean concave points     569 non-null    float64
 8   mean symmetry           569 non-null    float64
 9   mean fractal dimension  569 non-null    float64
 10  radius error            569 non-null    float64
 11  texture error           569 non-null    float64
 12  perimeter error         569 non-null    float64
 13  area error              569 non-null    float64
 14  smoothness error        569 non-null    float64
 15  compactness error       569 non-null    float64
 16  concavity error         569 non-null    float64
 17  concave points error    569 non-null    float64
 18  symmetry error          569 non-null    float64
 19  fractal dimension error 569 non-null    float64
 20  worst radius            569 non-null    float64
 21  worst texture           569 non-null    float64
 22  worst perimeter         569 non-null    float64
 23  worst area              569 non-null    float64
 24  worst smoothness        569 non-null    float64
 25  worst compactness       569 non-null    float64
 26  worst concavity         569 non-null    float64
 27  worst concave points    569 non-null    float64
 28  worst symmetry          569 non-null    float64
 29  worst fractal dimension 569 non-null    float64
dtypes: float64(30)
memory usage: 133.5 KB
None
```

Pre-Pruning

An a priori limit on nodes, or tree depth, is often set to avoid overfitting due to a deep tree[4,5].

Notes

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

- Another one is to stop growing if a split is not statistically significant (e.g., χ^2 test)[5]. However this is not yet availble in `sklearn`, although I'm sure you can find some code for it somewhere on the internet.

We could also set a minimum number of data points for each node[5].

Post-Pruning

In general, post-pruning consists of going back through the tree once it has been created and removing branches that do not significantly contribute to the error reduction and replacing them with leaf nodes[6]

Two common approaches are *reduced-error pruning* and *cost-complexity pruning*

- **Reduced-error pruning**[5]
  - Greedily remove nodes based on validation set performance
  - Generally improves performance but can be problematic for limited data set sizes.
- **Cost-complexity pruning**[5]
  - Recursively finds the node with the “weakest link”.
  - Nodes are characterized by $\alpha \geq 0$, and nodes with the smallest effective α are pruned first[7].
  - The trees are then defined as $I + \alpha|N|$, where I is an impurity measure, such as the total misclassification rate of the terminal nodes, α is a tuning parameter, and $|N|$ is the total number of nodes[5].

Notes

- An early "bad" split may lead to a good split later, therfore we may want to grow a large tree first and prune it back to obtain a *subtree*[1].
- α is the price to pay for having a tree with many nodes, so this tends to minimise to a smaller subtree. This is reminstant of the lasso.

Cost-complexity pruning[7]

Using `Scikit-learn`, we can recursively fit a complex tree with no prior pruning and have a look at the effective alphas and the corresponding total leaf impurities at each step of the pruning process.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

As alpha increases, more of the tree is pruned, thus creating a decision tree that generalizes better.

We can select the alpha that reduces the distance between the train and validation scores.

Notes

- In `Scikit-learn` 0.22 the parameter `ccp_alpha` was introduced (short for Cost Complexity Pruning- Alpha)
- `DecisionTreeClassifier.cost_complexity_pruning_path` returns the effective alphas and the corresponding total leaf impurities at each step of the pruning process.

Then we can train a decision tree using the chosen effective alpha.

Other Algorithms

ID3 - Iterative Dichotomizer 3[8]

- **Splitting Criterion:** Maximises Information Gain/ Minimises Entropy
- **Number of Splits:** Multiway
- **Variables:** Discrete binary and multi-category features
- **Pruning**: None

C4.5[9]

- **Splitting Criterion:** Maximises Information Gain/ Minimises Entropy
- **Number of Splits:** Multiway
- **Variables:** Discrete & Continuous (expensive)
- **Pruning**: Post-Pruning
- can handle missing data
- C5.0 is the latest release version under a proprietary license

Notes

- you can read an introduction to ID3 in pgs.27-30 of Burkov (2019)

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Associated Exercises

Now might be a good time to try exercises 1-6.

References

1. James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. An introduction to statistical learning. Vol. 112. New York: springer, 2013.
2. Gorman KB, Williams TD, Fraser WR (2014). Ecological sexual dimorphism and environmental variability within a community of Antarctic penguins (genus Pygoscelis). PLoS ONE 9(3):e90081. https://doi.org/10.1371/journal.pone.0090081
3. Géron, A. (2017). Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems. " O'Reilly Media, Inc.".
4. Raschka, Sebastian, and Vahid Mirjalili. Python Machine Learning, 2nd Ed. Packt Publishing, 2017.
5. https://github.com/rasbt/stat479-machine-learning-fs19/blob/master/06_trees/06-trees__notes.pdf
6. Burkov, A. (2019). The hundred-page machine learning book (Vol. 1). Canada: Andriy Burkov.
7. https://scikit-learn.org/stable/auto_examples/tree/plot_cost_complexity_pruning.html#:~:text=Cost%20complexity%20pruning%20provides%20another,the%20
8. Quinlan, J. R. (1986). Induction of decision trees. Machine learning, 1 (1), 81-106
9. Quinlan, J. R. (1993). C4. 5: Programming for machine learning. Morgan Kauffmann, 38, 48.
10. https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier
11. https://towardsdatascience.com/the-mathematics-of-decision-trees-random-forest-and-feature-importance-in-scikit-learn-and-spark-f2861df67e3#:~:text=Feature%20importance%20is%20calculated%20as,the%20more%20important%20the%20feature.
12. https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD