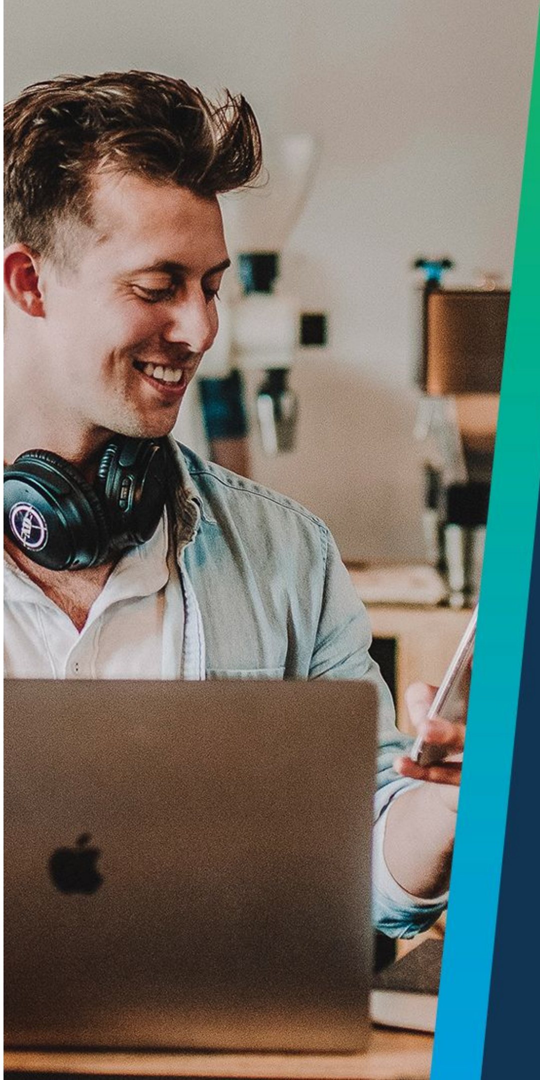


Data Science Bootcamp

Hyperiondev

Decision Trees I

Welcome

Your Lecturer for this session

Sanana Mwanawina

Lecture – Housekeeping

- ❏ The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly.
- ❏ No question is daft or silly - **ask them!**
- ❏ There are Q/A sessions midway and at the end of the session, should you wish to ask any follow-up questions.
- ❏ You can also submit questions here: hyperiondev.com/sbc4-ds-questions
- ❏ For all non-academic questions, please submit a query: hyperiondev.com/support
- ❏ Report a safeguarding incident: hyperiondev.com/safeguardreporting
- ❏ We would love your feedback on lectures: https://hyperionde.wufoo.com/forms/zsgv4m40ui4i0g/

Lecture - Code Repo

Go to: github.com/HyperionDevBootcamps

Then click on the "**C4_DS_lecture_examples**" repository, do view or download the code.

Objectives

- Learn about tree-based methods for regression and classification
- Tree-based methods solve problems using a flowchart

Intro to Decision Trees

- Decision tree work by formulating simple rules that partition data into even smaller regions. Each partition can be thought of as a fork in the road, where a decision will be made.

- The decision is made based on rules which are derived from learning experiences. Decision trees are among the most interpretable machine learning techniques based on how they resemble how humans make decisions.

- For example, to make dinner, one would first think if there are enough ingredients in the fridge to make a meal. If there aren't enough ingredients, we'll need to consider other options. Based on the time of day, you may decide to go to the grocery store or just decide on take-out instead.

Classification Trees

★ Decision trees created for datasets with a categorical dependent variable are called classification trees. As an example, let's look at the Credit Risk dataset.

★ The tree consists of nodes, and each node has a rule that determines whether an instance moves on to the left or right child node. At the end of each possible path is a leaf. In a classification tree, a leaf contains the predicted label for an instance with those input features.

Variables in Credit Risk dataset

Volume: volume of bank transactions per month (Frequent, Seldom)

Value: income per transaction (High, Low)

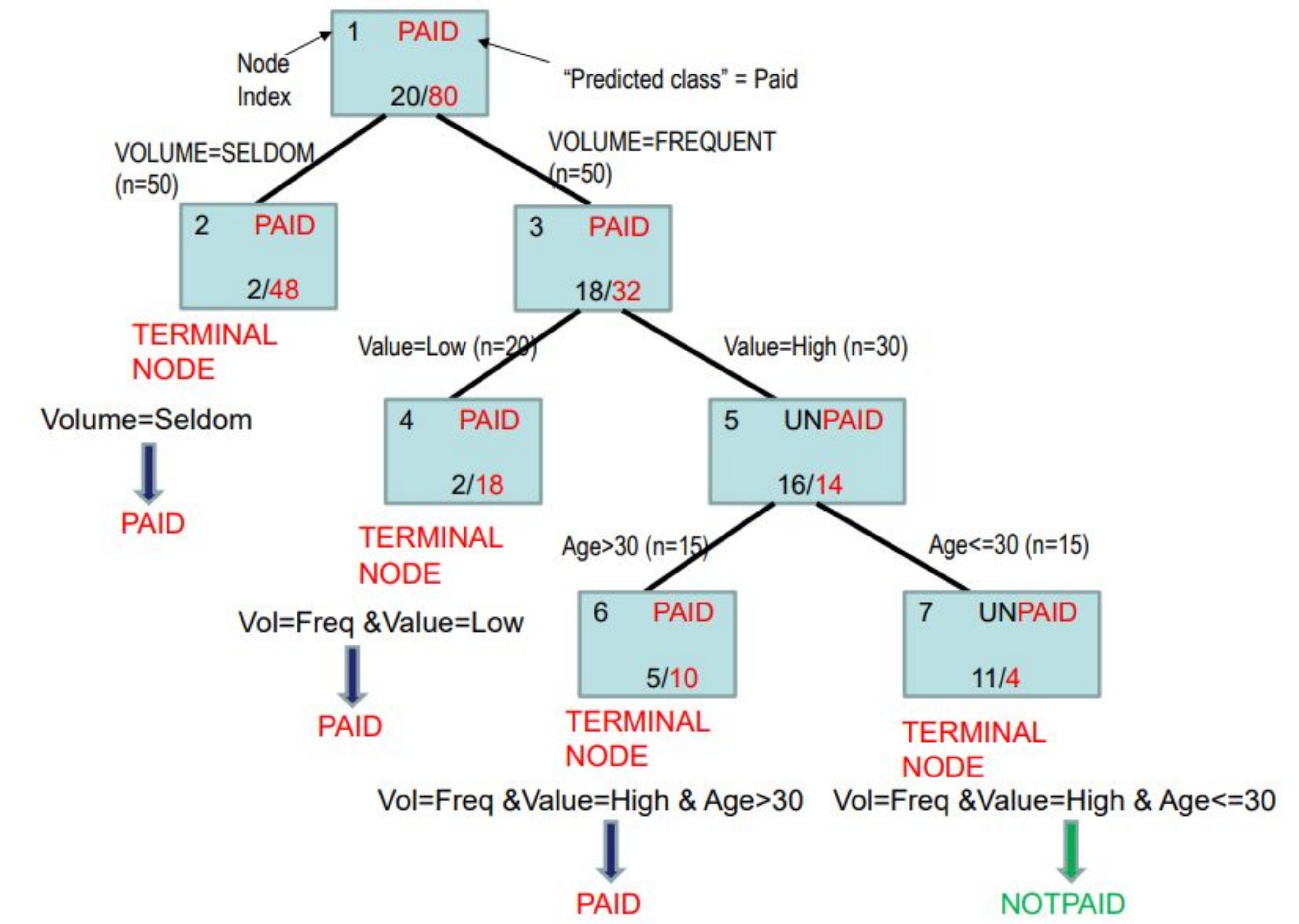
Age: age of customer (Younger than 25, 25 to 30, Older than 30)

Status: whether the loan was repaid (Not Paid/Paid)

Credit Risk dataset

	A	B	C	D	E
1	ID	Volume	Value	Age	Status
2	1	Seldom	Low	>30	Paid
3	2	Seldom	Low	>30	Paid
4	3	Seldom	Low	>30	Paid
5	4	Seldom	Low	>30	Paid
6	5	Seldom	Low	<25	Paid
7	6	Seldom	Low	<25	Paid
8	7	Seldom	Low	<25	Paid
9	8	Seldom	Low	<25	Paid
10	9	Seldom	Low	<25	Paid
11	10	Seldom	Low	<25	Paid
12	11	Seldom	Low	<25	Paid

91	90	Frequent	High	<25	Notpaid
92	91	Frequent	High	<25	Notpaid
93	92	Frequent	High	<25	Notpaid
94	93	Frequent	High	25-30	Notpaid
95	94	Frequent	High	25-30	Notpaid
96	95	Frequent	High	25-30	Notpaid
97	96	Frequent	High	>30	Notpaid
98	97	Frequent	High	>30	Notpaid
99	98	Frequent	High	>30	Notpaid
100	99	Frequent	High	>30	Notpaid
101	100	Frequent	High	>30	Notpaid



Hyperiondev

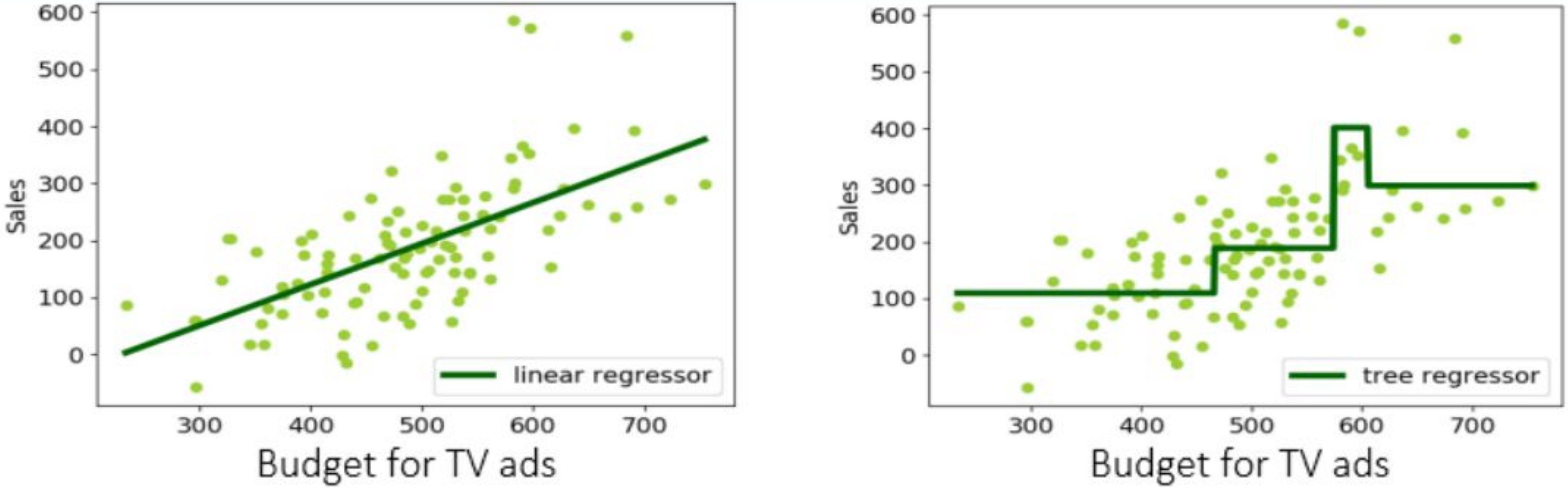
Rules for Classification

- ★ Do not lend money to customers who are 30 or younger and who transact frequently for large values
- ★ Lend money to all other people

Regression Trees

★ Decision trees can also be used for problems with a numerical dependent variable. A regression decision tree divides input features into regions and assigns categories to those regions. The regions and their labels are based on what best fits the data. Again, best fit here means the set of regions that minimises the distance between the predictions of the model and the observed values.

★ So while a linear regression approach to our advertising problem in the previous tasks gave different predictions for each unique value of x, a regression tree gives different predictions for regions of x.



Hyperiondev

Regression Trees

- ★ You can imagine that this approach is more flexible. The regions could capture a steeper or less steep increase in sales if that fits the data better than a linear model. The regions can also be arbitrarily specific or broad. Trees can also be visualised easily to get information about the decisions the model makes, in case we want to change the parameters of the model.

Overfitting & Underfitting

- ★ When discussing trees it needs to be said that due to their flexibility, they are prone to something called "overfitting".
- ★ Overfitting is one of the biggest causes for poor performance of machine learning algorithms, together with its counterpart, underfitting.
- ★ Underfitting refers to a model that can neither model the training data nor generalise to new data. The training error is high because the model was not able to make good predictions based on the input features.

Underfitting

- ★ High training error because the model did not make good predictions based on input features

- ★ May fail to fit the data because it is too simple to capture the patterns

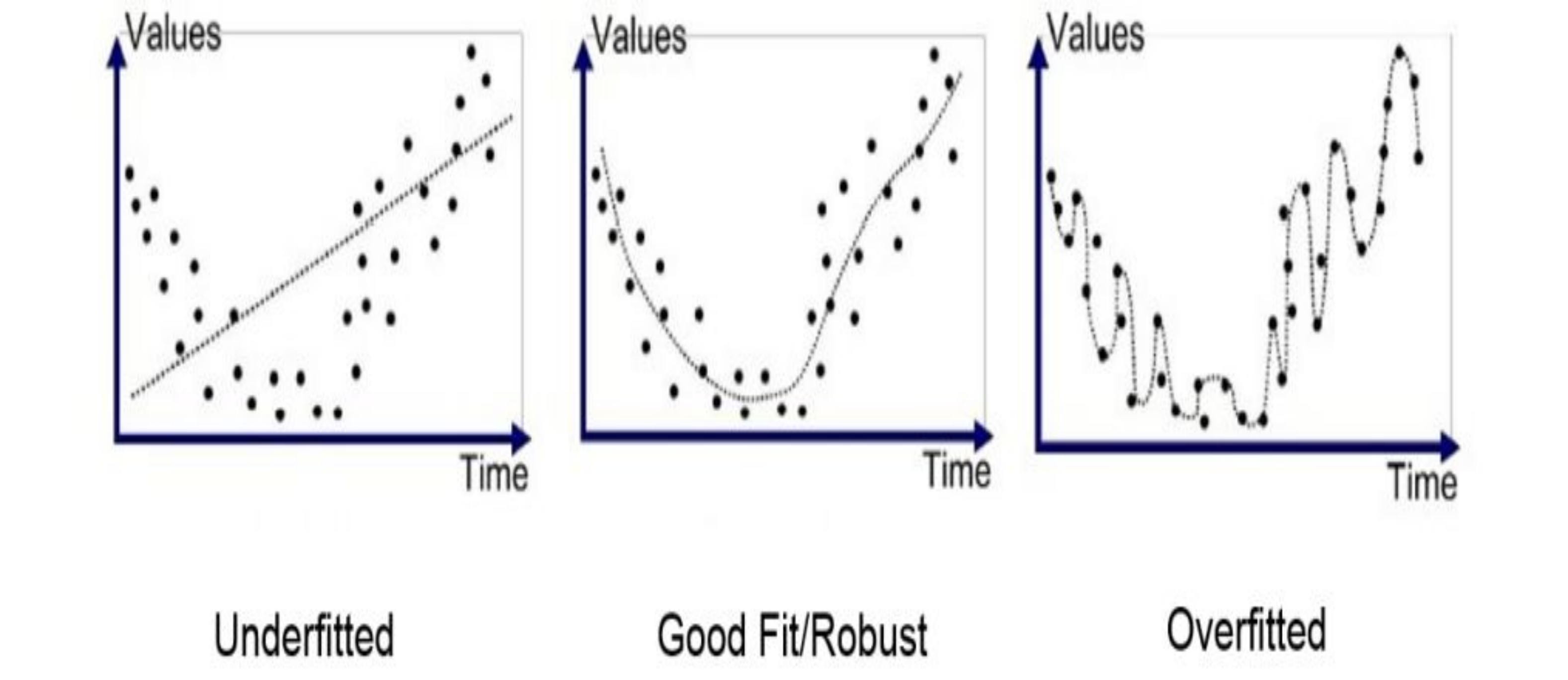
- ★ Underfitting is more commonly caused by a problem with the task set-up (e.g. the features x are not a good predictor of y) or with the training data (e.g. there is too little training data to learn from, or it contains too many mistakes).

- ★ A model that is underfit will have high training and high testing error

Overfitting

★ An underfitting model has high training error and will not perform well on test data. However, that does not mean that a model with low training error is automatically going to do well on test data.

★ A model with very low training error may suffer from overfitting: some of the rules the model learned are only applicable to training data and are not useful for solving the overall problem we want to solve. The model is too tailored and does not generalise well. It is important that the model is able to generalise beyond the training data, as this data is only a sample. If the model is overfitted, it will perform very well on training data, but it won't translate to good performance on test data.


Values
Time
Values
Time
Values
Time
Underfitted
Good Fit/Robust
Overfitted

Diagnosing Over and Underfitting

- What are the causes of overfitting and underfitting? There are many reasons, and it can be typically difficult to determine. Generally speaking, an overly-complex model will overfit, and an overly-simple model will underfit. Think of overfitting as simply “remembering” the training data piece by piece, without learning how to construct a rule.

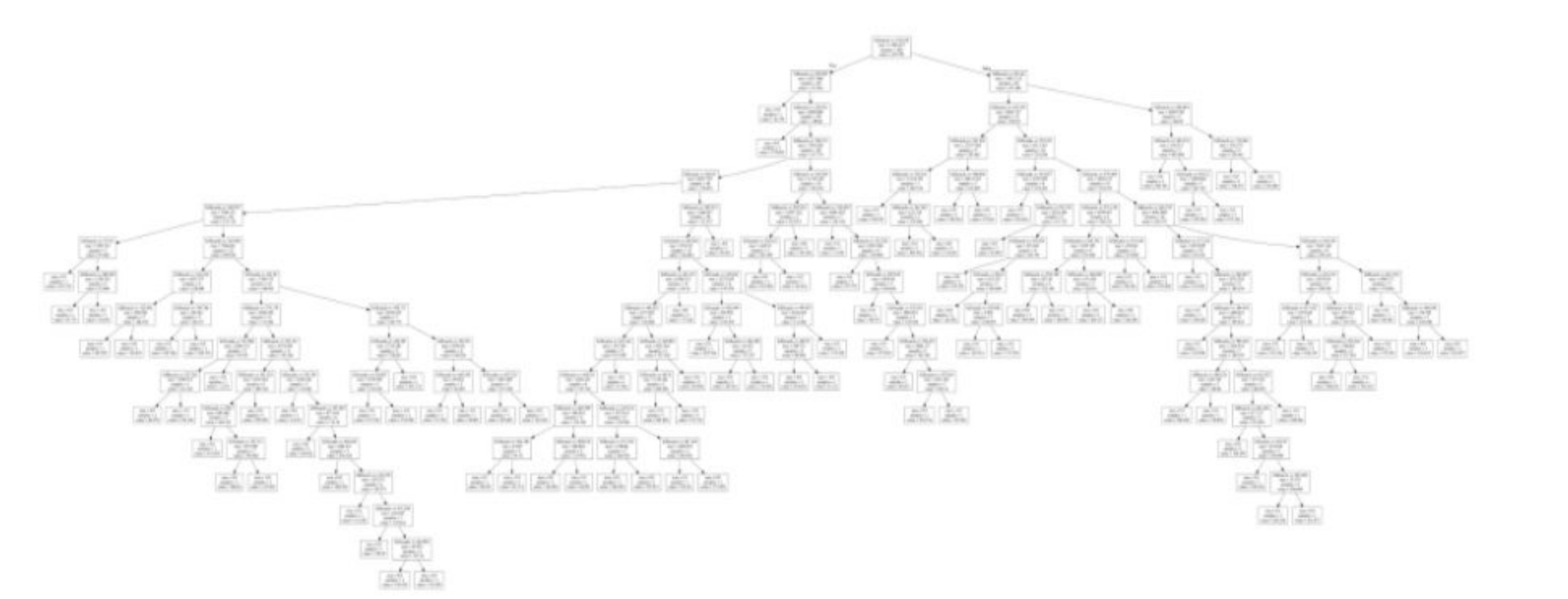
Diagnosing over and underfitting

- The first graph on the left is an incredibly simple model: it's just a straight line. There is no way that a straight line will be able to predict that data: hence you will see a high training error and a high testing error.
- The middle graph is probably the best fit for the data. I'm sure that you can picture more samples of this data being added, and the model being able to give a rough estimate of what that should be. Note that the training error will be lower than an underfit, but higher than an overfit model. However, the testing error will be lower than both - testing error is what is most important. Let this be a lesson: zero error is rarely a good thing, and lower training error doesn't always mean a better model.
- The last graph on the right shows a model that has overfit. The training error here will be incredibly low - practically zero! That being said, if more data were added, it's likely that the model will make a valuable prediction, as it doesn't seem to follow the overall trend of the data.

Pruning

★ As mentioned before, decision trees are very flexible and, therefore, are likely to overfit training data. This problem can be addressed by “pruning” a tree after training in order to remove some of the detail it has picked up and make the model more abstract and more general.

★ A strategy to prevent overfitting is thus to grow a very large tree without any restrictions first, which is then pruned to retain a more general subtree. If those trees were left to develop without restrictions, they would have become this detailed:



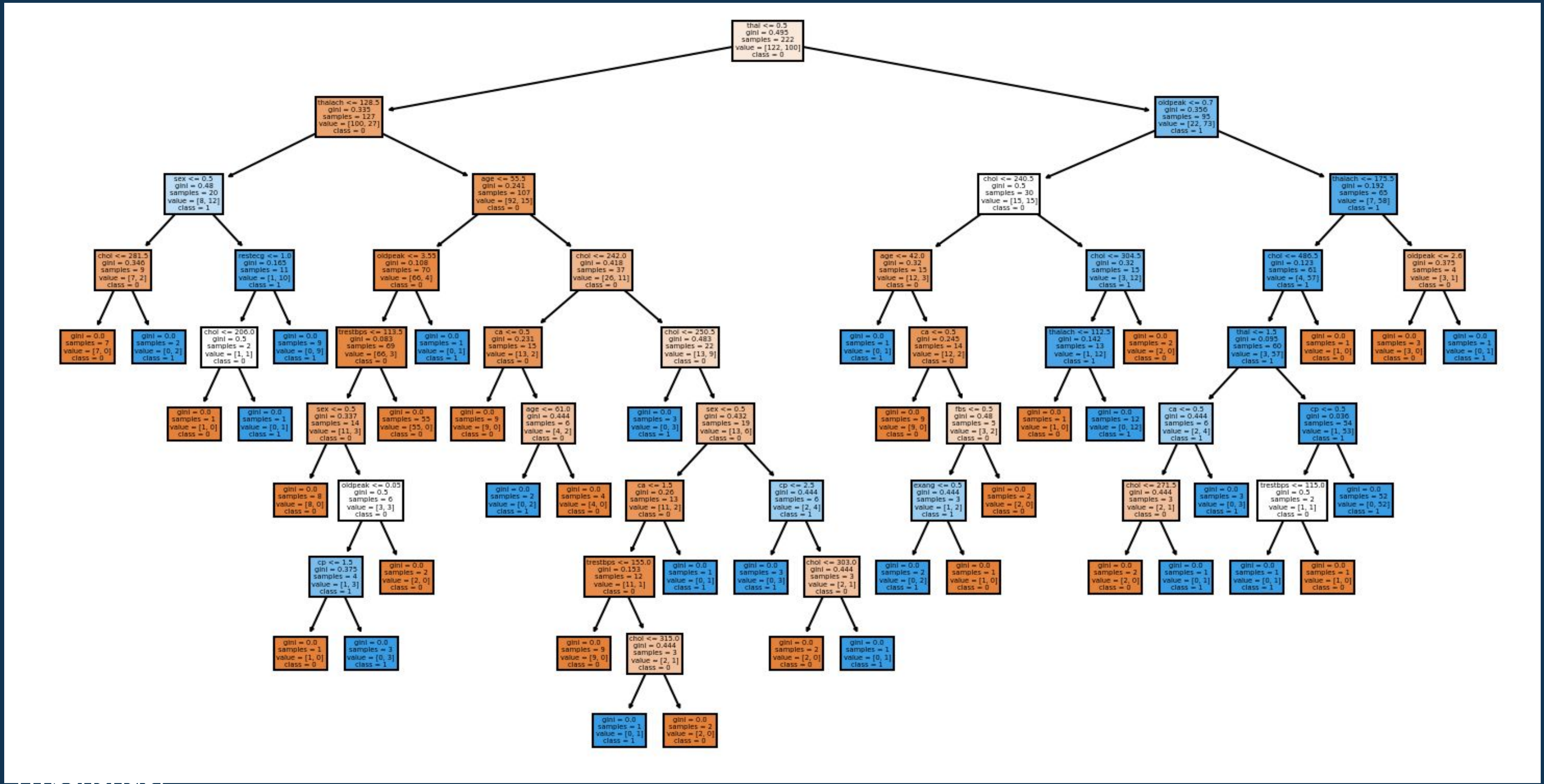
Development Data

★ How do we determine the best way to prune a tree? We can take subtrees of an unpruned tree and compute the test error of predictions of that subtree. We can then select the subtree with the lowest test error rate. However, once we have done that we have fitted our pruning parameter to the test data.

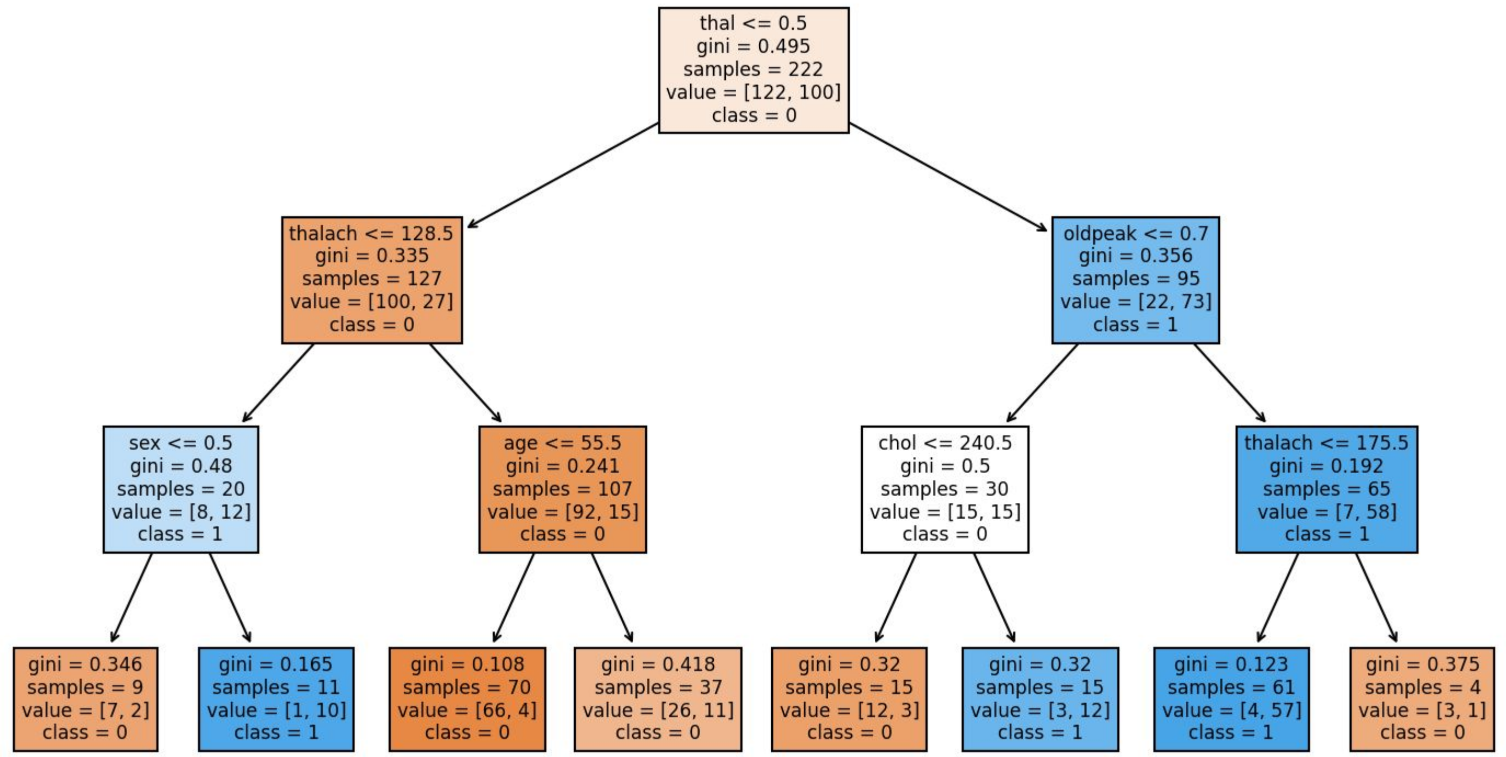
★ This means the test data is no longer completely unseen. Fitting to test data can be avoided by splitting the dataset into three parts instead of two before training: a training set, development set ('dev set'), and test set.

★ The development set is used to see whether the model seems to be generalising well to data that is not in the training set. This makes it possible to spot and try to remedy under- or overfitting. Only at the end do we test the model on totally unseen data. Another term for this third type of held-out data is the validation set.

Pruning



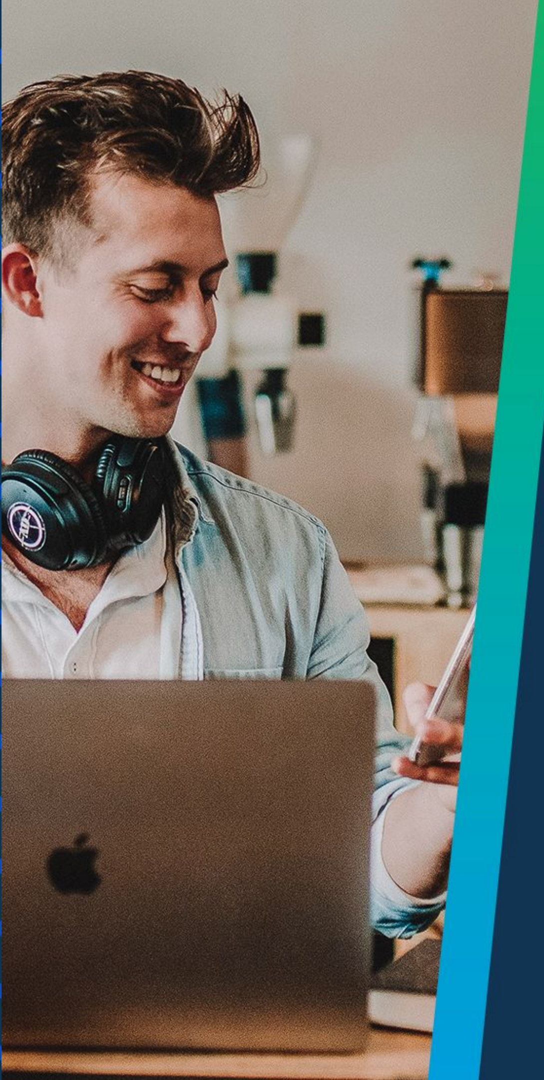
Pruning


thal <= 0.5
gini = 0.495
samples = 222
value = [122, 100]
class = 0
thalach <= 128.5
gini = 0.335
samples = 127
value = [100, 27]
class = 0
oldpeak <= 0.7
gini = 0.356
samples = 95
value = [22, 73]
class = 1
sex <= 0.5
gini = 0.48
samples = 20
value = [8, 12]
class = 1
age <= 55.5
gini = 0.241
samples = 107
value = [92, 15]
class = 0
chol <= 240.5
gini = 0.5
samples = 30
value = [15, 15]
class = 0
thalach <= 175.5
gini = 0.192
samples = 65
value = [7, 58]
class = 1
gini = 0.346
samples = 9
value = [7, 2]
class = 0
gini = 0.165
samples = 11
value = [1, 10]
class = 1
gini = 0.108
samples = 70
value = [66, 4]
class = 0
gini = 0.418
samples = 37
value = [26, 11]
class = 0
gini = 0.32
samples = 15
value = [12, 3]
class = 0
gini = 0.32
samples = 15
value = [3, 12]
class = 1
gini = 0.123
samples = 61
value = [4, 57]
class = 1
gini = 0.375
samples = 4
value = [3, 1]
class = 0

Hyperiondev

Q & A Section

Please use this time to ask any questions relating to the topic explained, should you have any



Hyperiondev

Thank you for joining us