Data Science Bootcamp

Hyperiondev

Decision Trees II

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

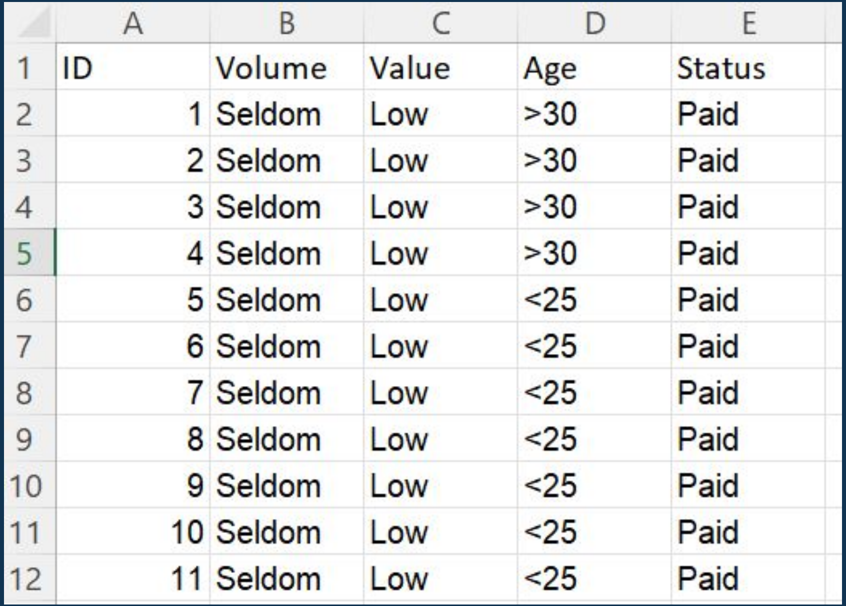
Go to: github.com/HyperionDevBootcamps

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

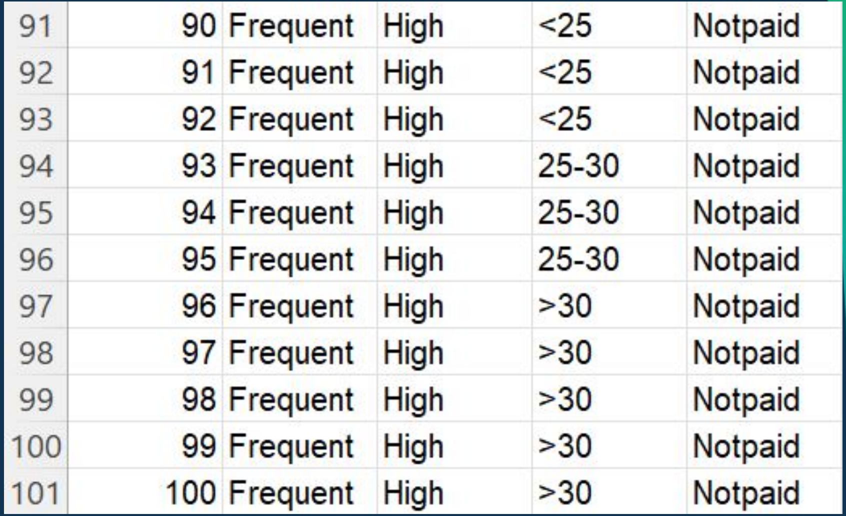
Objectives

- Get a general idea of how to build trees
- Learning about ensemble methods
- Understand how to improve the predictive power of decision trees

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

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)

Hyperiondev

Problem

- ★ The bank would like to identify customers who are good to lend money to and those who are not.
- ★ We create a tree diagram to create classification rules to help in this decision making process

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

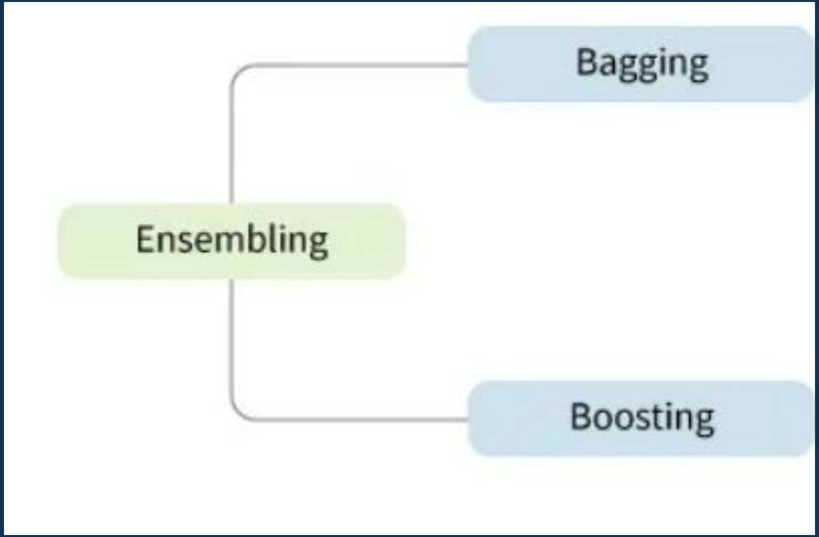
Ensembles

- ★ Decision Trees are easy to understand, apply, interpret and visualise. However, they are not very robust.

- ★ The predictions of decision trees have very high variance. Ideally, we'd like our models to capture general patterns, and not to be so dependent on the data they have been trained on because that leads to a bit of noise or a different sample changing predictions entirely.

Ensembles

★ Multiple learning algorithms are used together to achieve higher predictive performance than if you used an individual algorithm by itself.



Ensembles

★ Ensemble techniques tackle this problem by treating a tree’s predictions as votes towards labels. Rather than trying to create one classifier that makes perfect predictions, ensemble methods aggregate the predictions of multiple classifiers into a single, improved prediction.

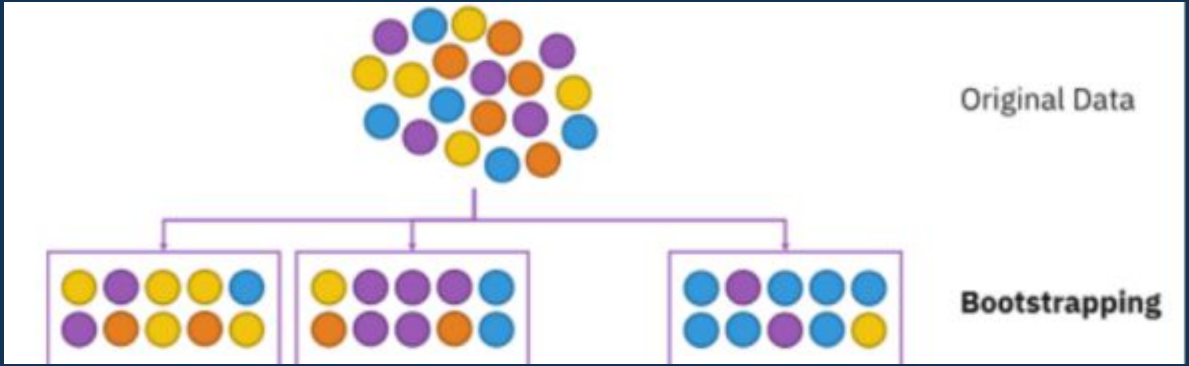
★ Aside from Random Forests, ensemble methods can and do get applied to methods other than decision trees, but trees can benefit in particular due to how flexible they are.

Bootstrapping

- ★ The principle behind ensemble methods is that different classifiers identify different patterns. Some of these are noise and others are useful.
- ★ Training many different classifiers and combining the result will help identify and get rid of some noise and retain only the best patterns.
- ★ The most straightforward way to leverage this fact would be to train multiple models on the same training data and take an average, but this is not very effective. One step up in sophistication is bootstrapping.
- ★ In bootstrapping, samples are drawn from a dataset repeatedly. If the training portion on a dataset is size N, the samples are each a size smaller than N. The samples are drawn with replacement, meaning an instance can occur in more than one sample. The collection of samples is called the bootstrap data set.

Bootstrapping

One half of the full picture:

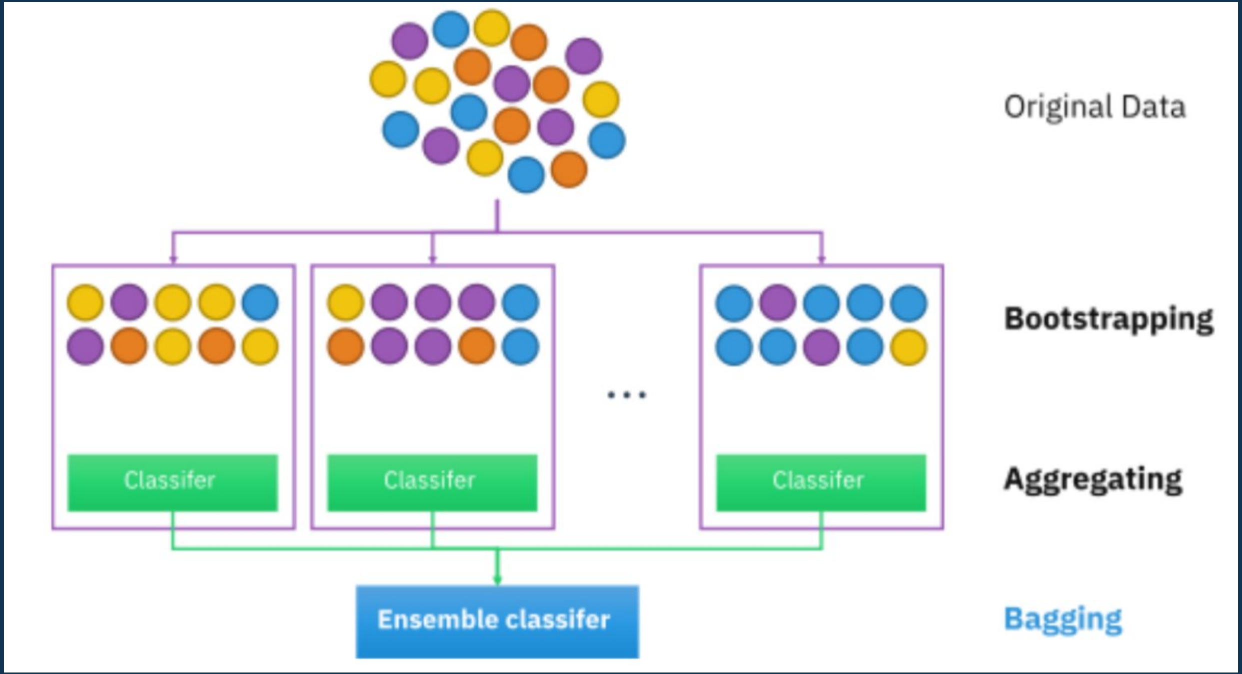


Hyperiondev

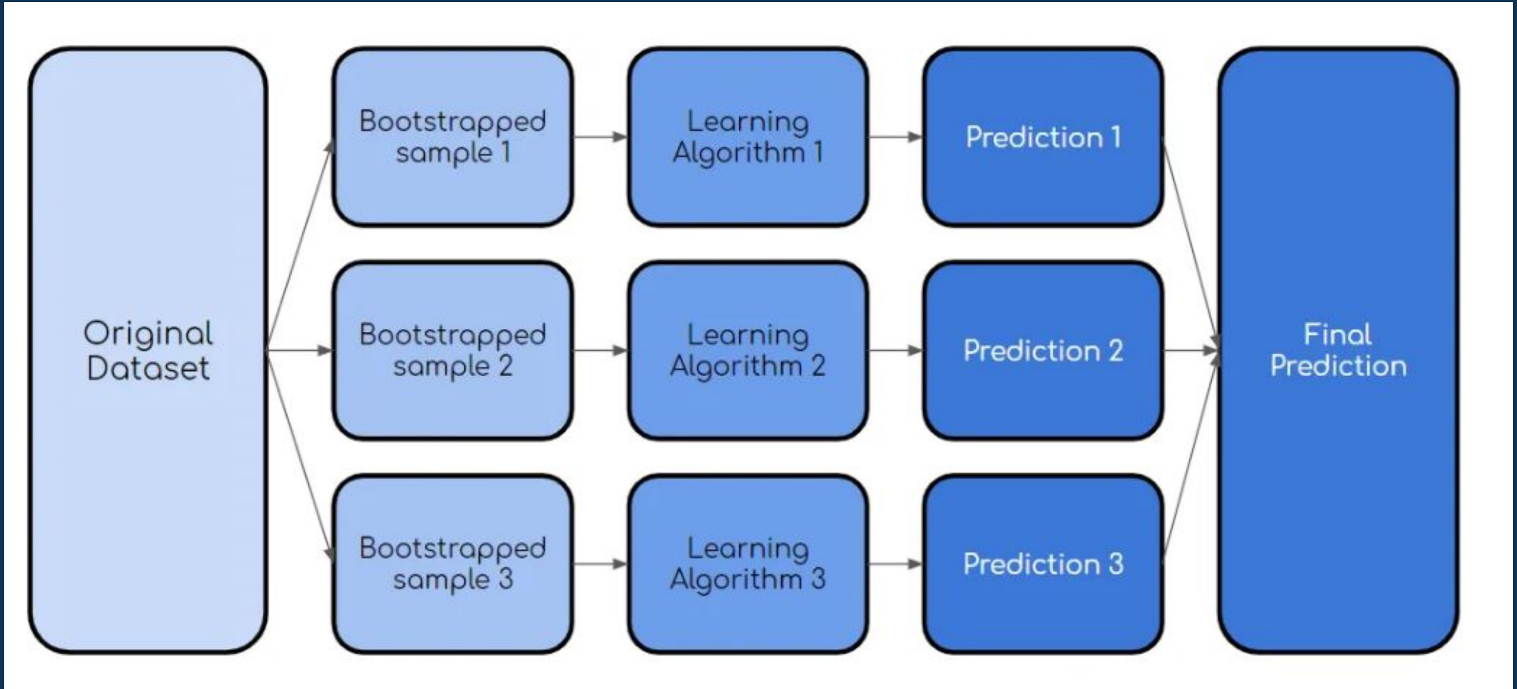
Bagging

- ★ The next part is to aggregate predictions. Bagging (short for Bootstrap aggregation) refers to the step in which the predictions of models fitted to the sample are combined into a final prediction.
- ★ Regression approaches yield predictions that are continuous and can be aggregated by taking the average.
- ★ In a classification scenario with qualitative labels, a common method is to take the majority vote. The majority vote for an instance is the class that is predicted for that instance by the most classifiers.
- ★ The performance of the ensemble is then tested on some held-out data

Bagging


Original Data
Bootstrapping
Aggregating
Bagging
Classifer
Classifer
Classifer
...
Ensemble classifer

Bagging



Bagging Process

Hyperiondev

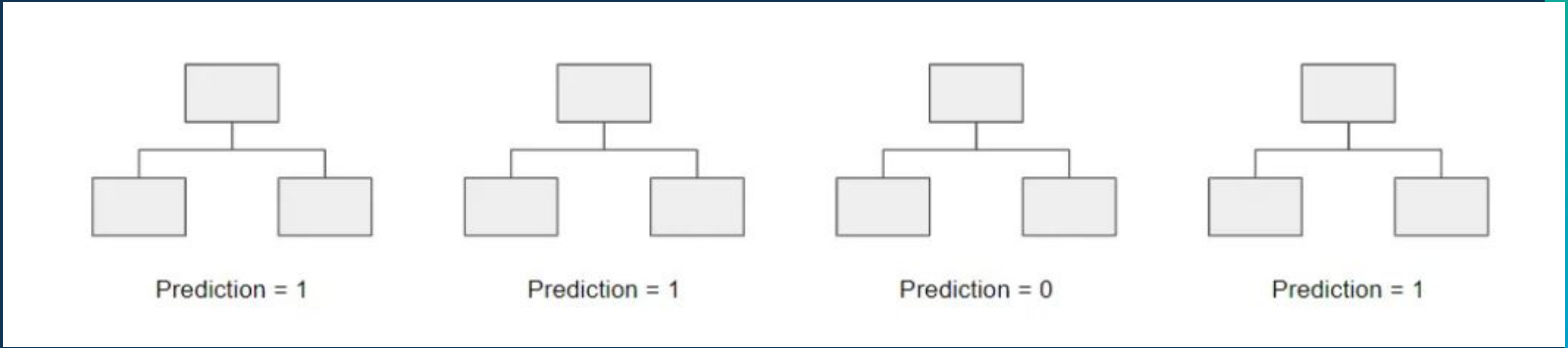
Random Forests

- ★ Random forests provide an improvement over bagged trees by decorrelating the trees. As with regular bagged trees, a number of decision trees are built using bootstrapped training samples. However, in regular bagging, it is possible that one variable that is a particularly strong variable for predicting splitting, which is likely to be used early on by each tree, results in similar trees.

- ★ With random forests, this behaviour is prevented by only allowing one of a random subset of independent variables to be considered as split candidates from the full set of variables. This causes the trees to be more dissimilar as the strong splitting candidate variable is not always present in the random subset, hence we can think of this process as decorrelating the trees.

Random Forests

★ Averaging many uncorrelated quantities leads to a larger reduction in variance than averaging many correlated quantities. This makes the average of the resulting trees less variable and hence more reliable.

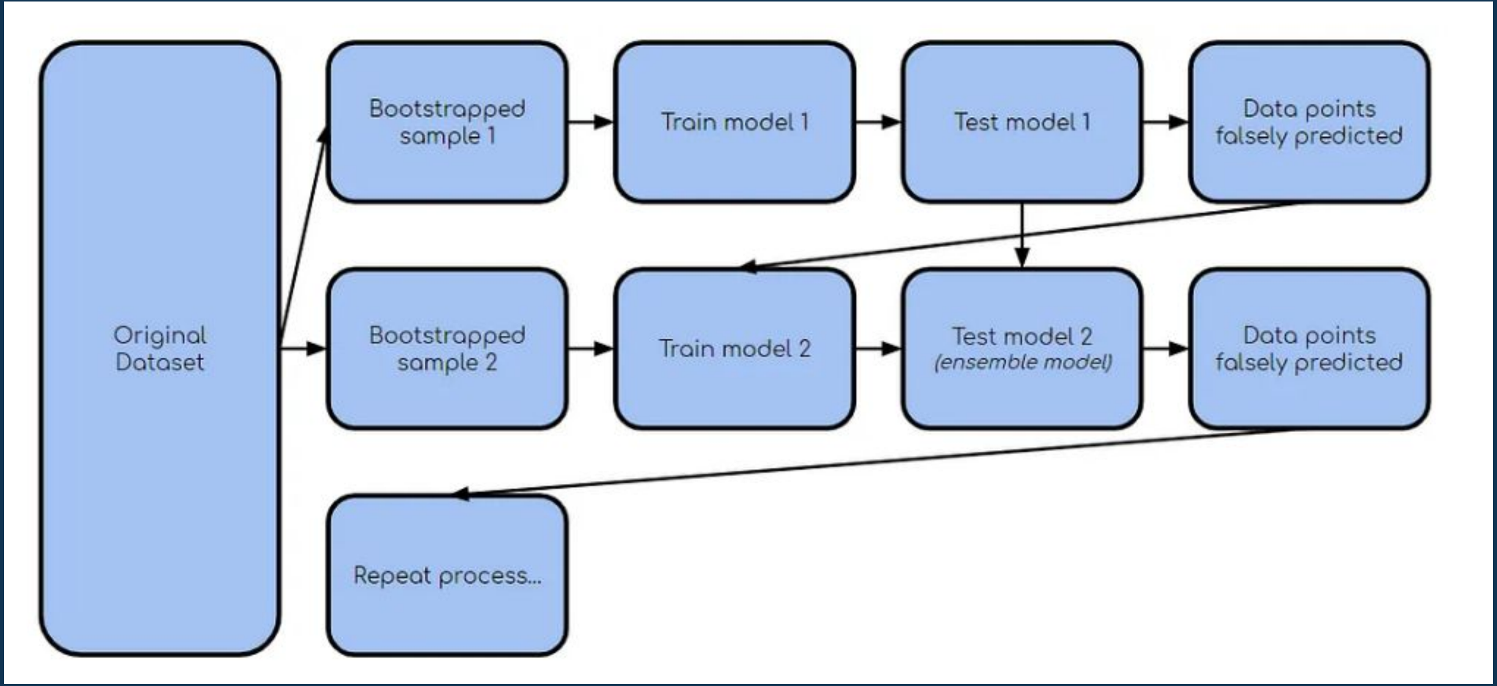


Boosting

- ★ Bagging lets models run in parallel on different samples without communicating with each other. An alternative approach is to run models on different samples but to do so sequentially in such a way that models later in the sequence have some information about how previous models fared at the task.

- ★ Boosting attempts this by incorporating the residuals (another name for the error) of earlier models into the data provided to later models. In principle, this causes each model to correct the mistakes of a previous model.

Steps for Boosting



Boosting

- ★ How this approach works is that each model focuses on the errors of the previous model. The subsequent model may make new mistakes that the first one did not as a result of the weighting, but that is okay. Each model is a specialist on some subset of the data: in itself quite weak, but a ‘boost’ to the performance of the ensemble.

- ★ As a result, boosting algorithms avoids a scenario where particularly difficult instances are misclassified by every single model. Furthermore, if a vanilla or bagged model performed rather poorly, a boosted model may improve performance.

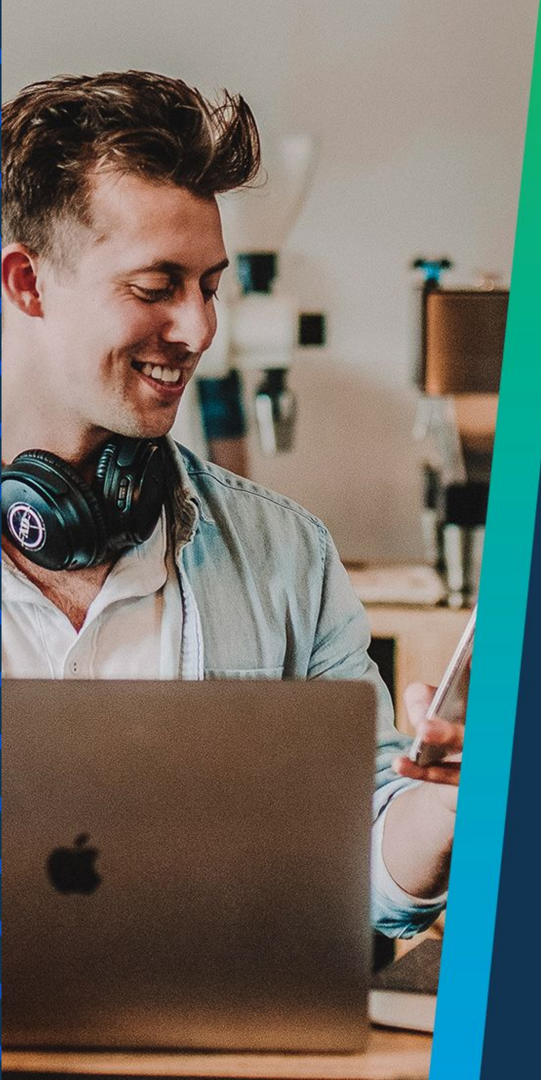
Boosting

- ★ On the downside, the outcome of boosting depends heavily on the order the models were placed in, and there are no methods for determining which order is best besides trial and error. Secondly, unlike bagging, boosting does not avoid overfitting, it only lowers classifier variance.

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