Warwick Business School

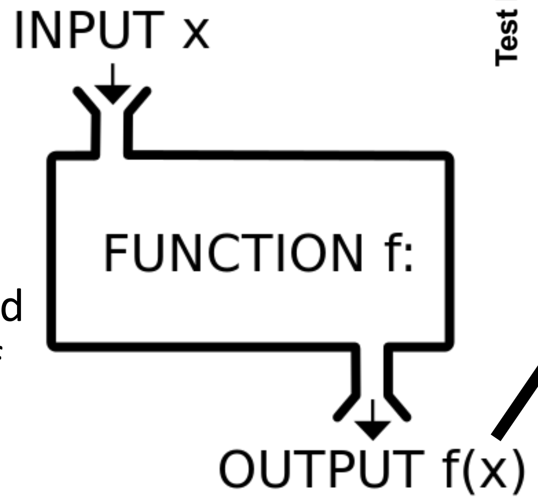
Data Science & Generative AI

Dr Michael Mortenson
Associate Professor (Reader)
michael.mortenson@wbs.ac.uk

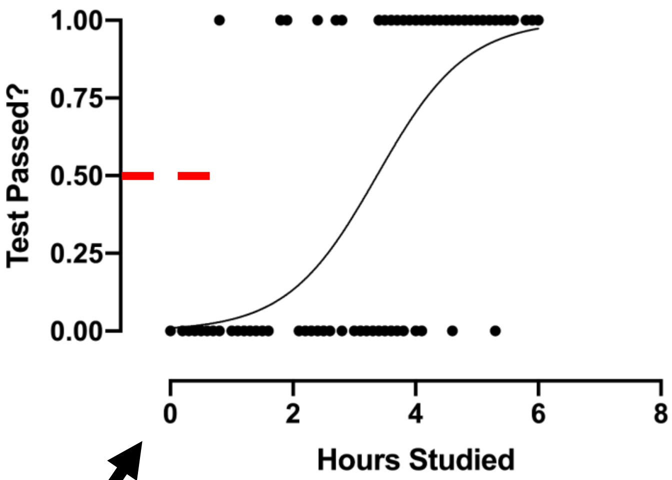
Session 5: Ensemble Models

1.1 The Story So Far …

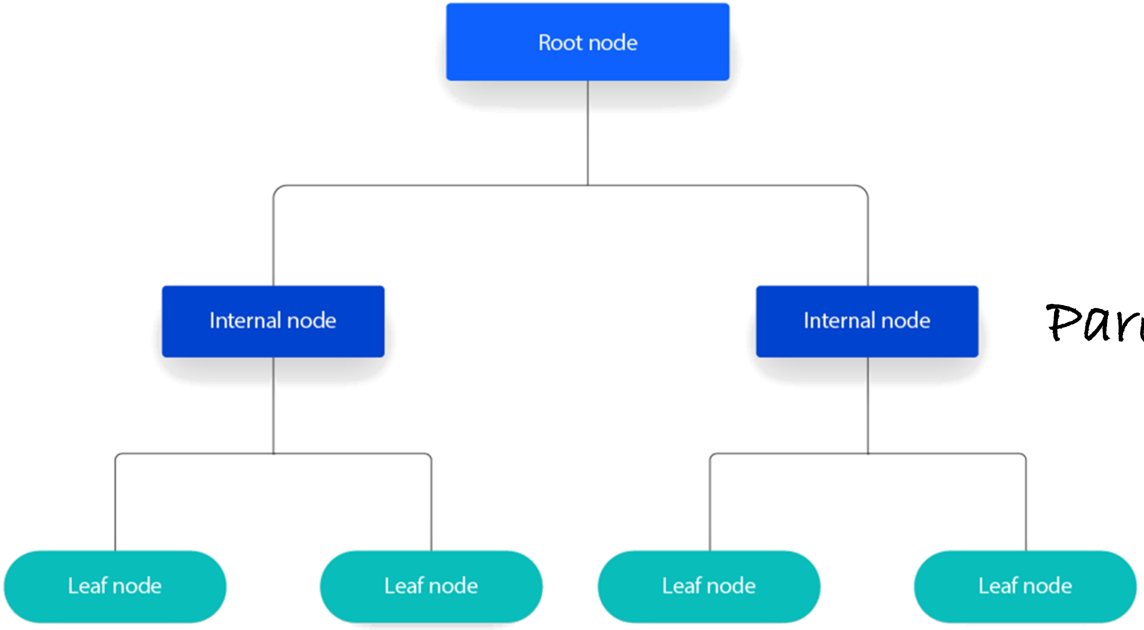
$$Y = \alpha + \beta_1 x_1 + \ \ldots + \ \beta_n x_n$$



Sigmoid / logistic function:
Takes the regression output and turns it into the probability of class 1 (0 to 1)



1.1 The Story So Far …



Parent

Child

1.3 However …

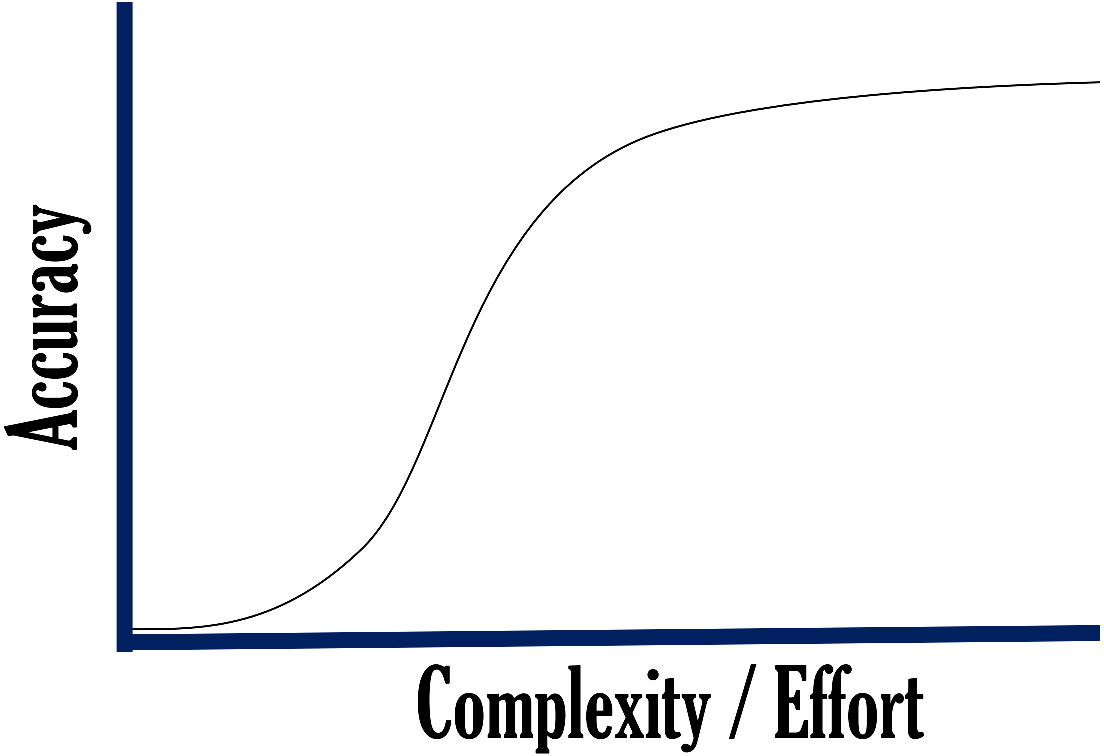
- **Logistic Regression**
  - Limited to *linear* problems – many problems are *non-linear*;
  - Fixed parameter size (number of features + 1). Means that we likely underfit complex problems.
- **Decision Trees**
  - In practice, one of two things happens:
    1. We learn a model that is too simple (tree is too shallow) and we underfit;
    2. We learn a model that is too complex (tree is too deep) and we overfit.

1.4 AI Building Blocks

- **Software has eaten the world.**
- **AI has eaten software.**
- **Machine learning has eaten AI.**
- **Neural networks have eaten machine learning.**
- *We'll learn about neural networks next week.*
- *Learning about neural networks is much easier if we first understand logistic regression and the stuff we'll talk about today.*

1.5 MDTGA

1.6 Building Models is Hard



1.7 If Your Problem …

- Is relatively non-complex;
- Will be based on relatively small data;
- Will be based on structured data;
- Requires some form of explainability (explainable AI – XAI);

  and/or
- Needs to be completed (or prototyped) quickly …
- Logistic regression or the methods we will discuss today are likely to perform best (and are still widely used in business).

Session Aims

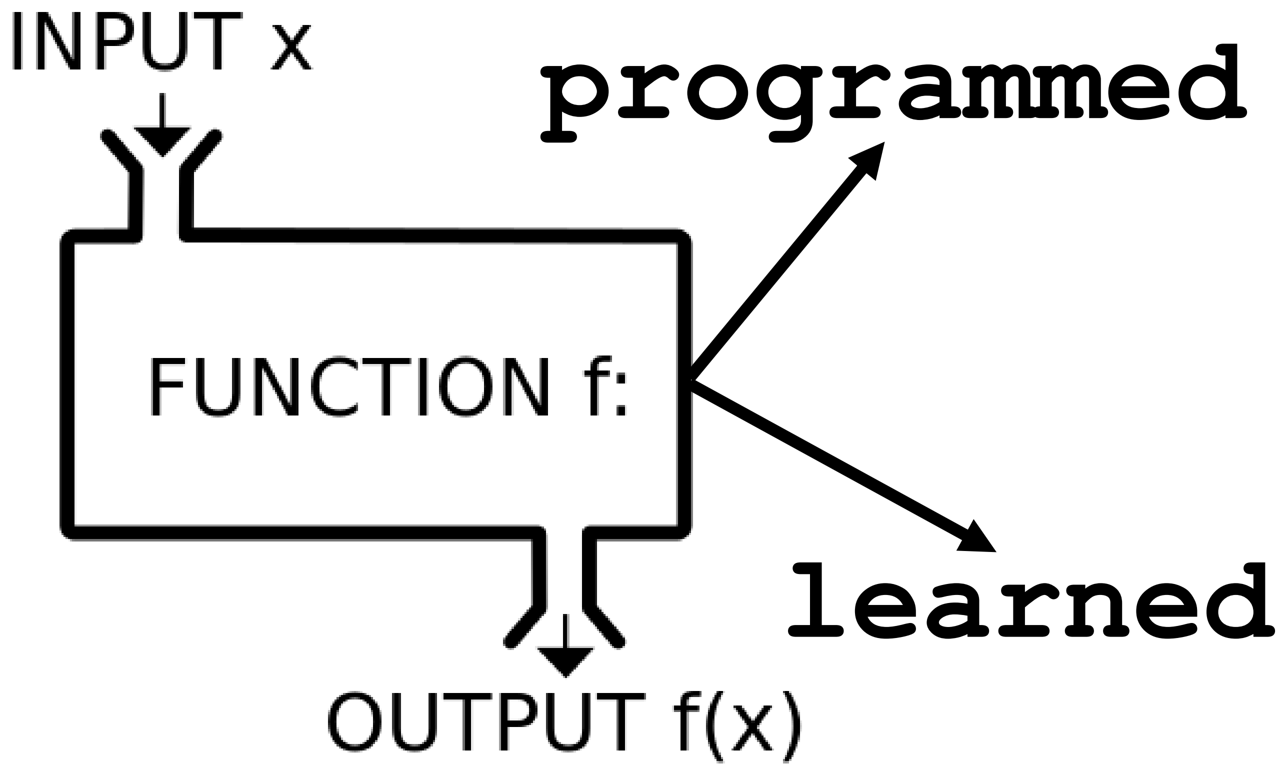
Introduction

Random Forests

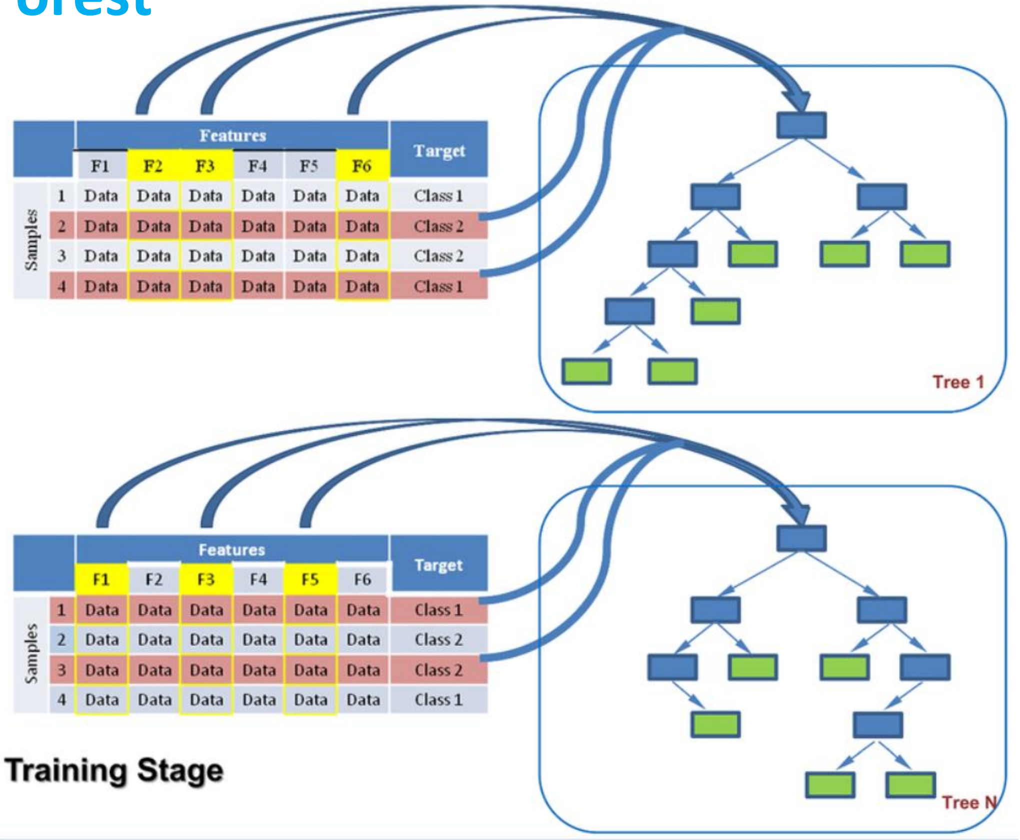
Gradient Boosted Decision Trees (GBDT)

Asynchronous Tasks

2.1 ML Programs: Code + data

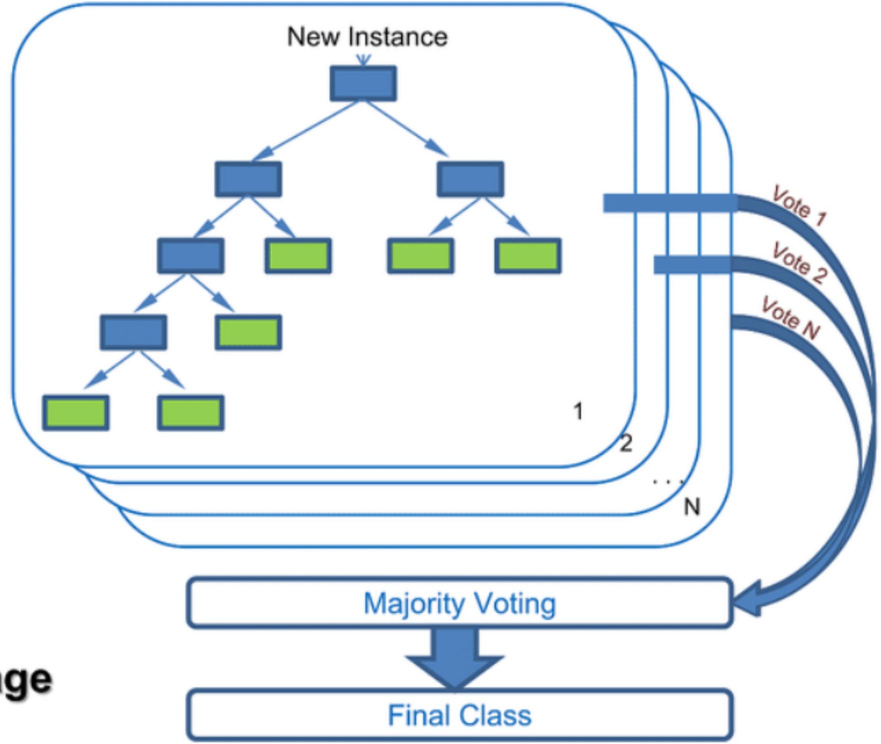


2.2 Random Forest



Virnodkar S *et al*. (2020). Remote sensing and machine learning for crop water stress determination in various crops: a critical review. *Precision Agriculture* **21**.

2.2 Random Forest



Prediction
[redacted] Stage

Virnodkar S *et al*. (2020). Remote sensing and machine learning for crop water stress determination in various crops: a critical review. *Precision Agriculture* **21**.

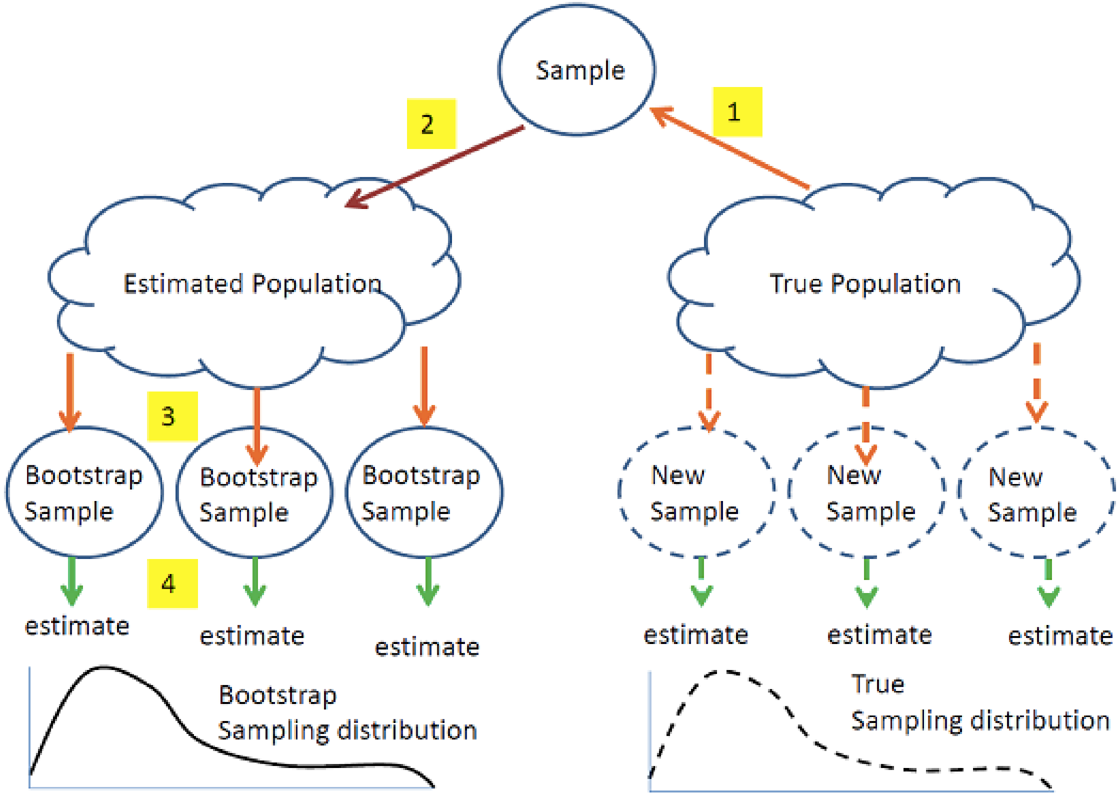
2.2 Random Forest

- Random forests are an ensemble of decision trees all trained on different data;
- We first split our dataset (N) into M smaller chunks of data (usually by features). We then train M models, each trained on part of the data.
- However, in many cases we do not want to reduce the size of our training data. We want all our trees to learn from a “full” dataset …

2.3 Bagging (Bootstrap Aggregating)

	Real	Hypothesised / Estimated
Unobtainable	**Production (future) Data**	**The True Population**
Obtainable	**The Training Sample**	**A Synthetic Sample**

2.3 Bagging (Bootstrap Aggregating)



https://online.stat.psu.edu/stat555/node/119/

2.4 Random Forest (with Bagging)

- Use bootstrap sampling (sampling with replacement) to create M datasets.
- We then train M models, each trained on a separate dataset.
- Each of these M datasets are sampled datasets.
- These are synthetic (fake) samples – but based on the distribution of the data. E.g. it will look like the real data we have but with random variations.
- **DON'T BE SCARED OF RANDOMNESS!!**

2.5 Random Forest Hyperparameters

- `n_estimators`: Number of trees to build (M). Corresponds to the number of boostrapped data samples we will create.
- The usual decision tree hyperparameters.

Session Aims

Introduction

Random Forests

Gradient Boosted Decision Trees (GBDT)

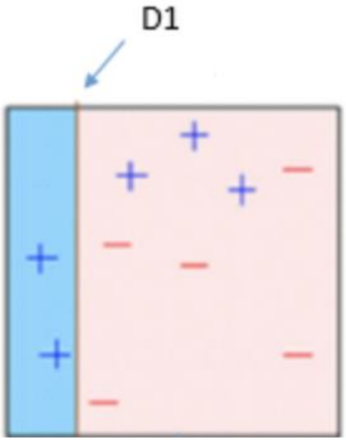
Asynchronous Tasks

3.1 What is the Hardest Part of Writing an Essay?

Image Credit: Blink. Blink. Blink. • Joe Peacock's Website

3.2 The Blindmen and the Elephant

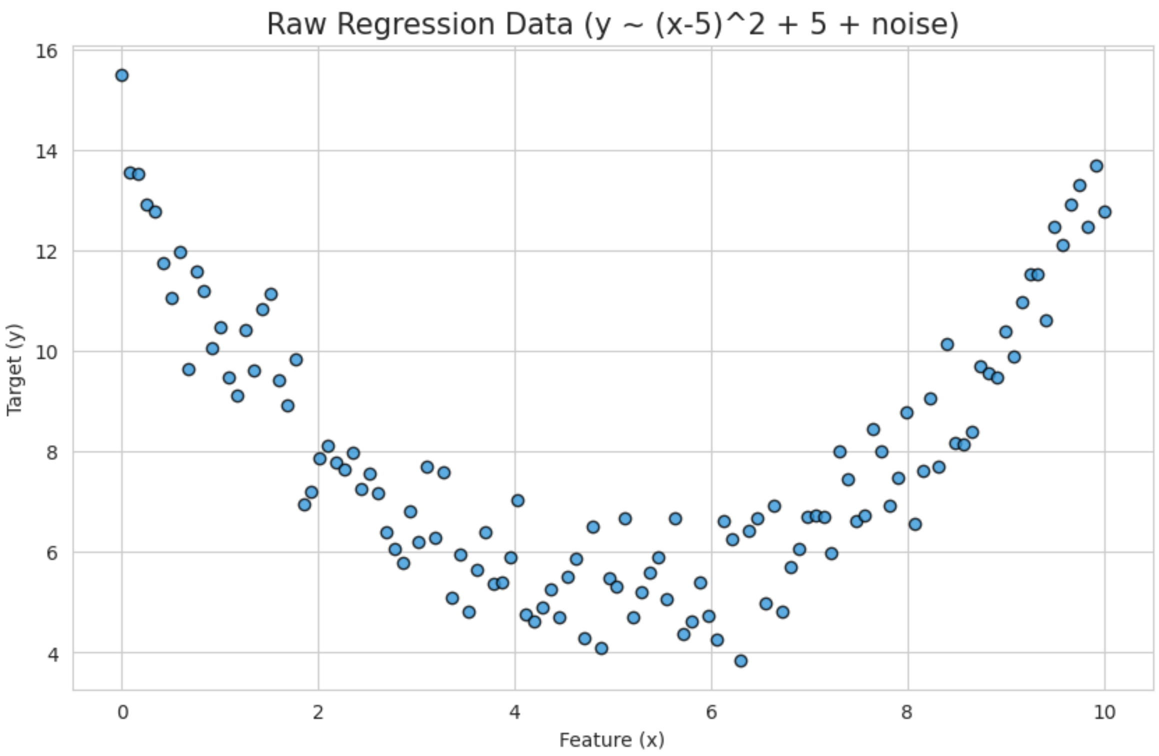
3.2 The Blindmen and the Elephant



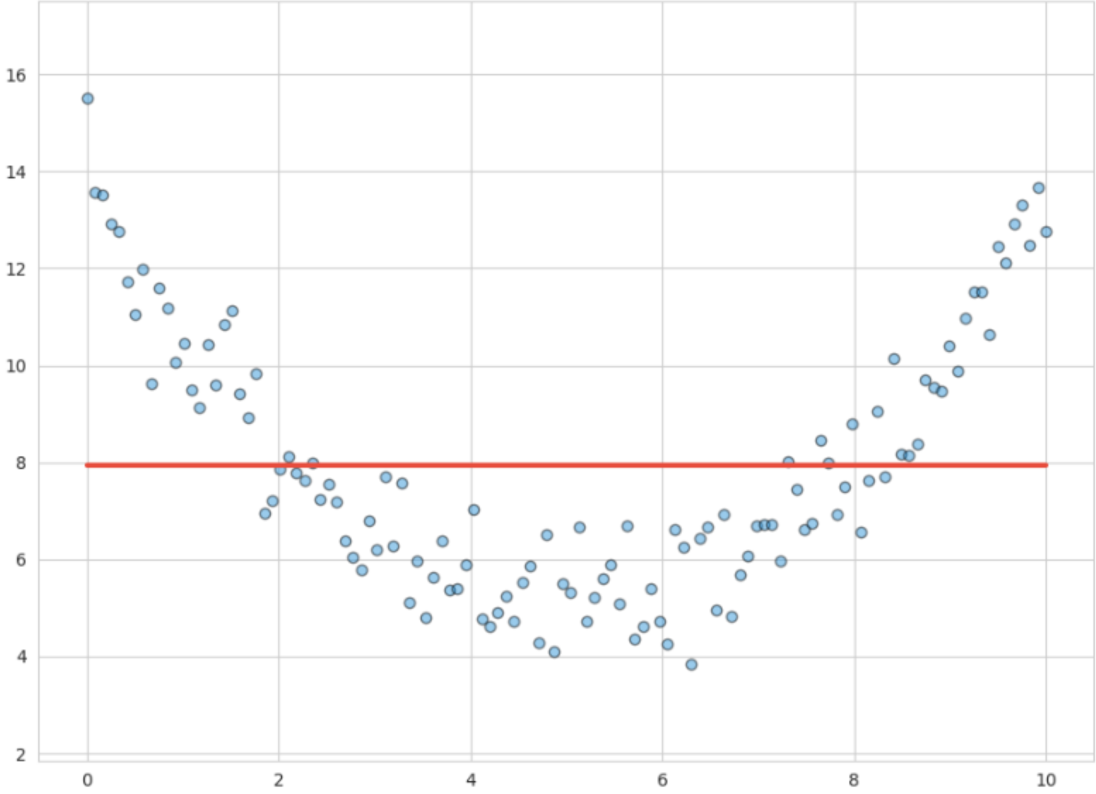
3.3 Boosting

1. Begin with a very weak algorithm configuration (such as a `max_depth=1` decision tree) and assign data and initial weight (all data assigned an equal weight of $1/n$);
2. Run the decision tree algorithm;
3. Increase the weights for data which was incorrectly classified, and decrease the weights for data correctly classified;
4. Run the algorithm with bias towards the higher weighted data. Add the resultant lines/rules to the existing model;
5. Repeat (3) to (4) until we reach the max trees / `n_estimators` hyperparameter. After multiple iterations, the algorithm will find a far more sophisticated model!

3.4 Gradient Boosted Decision Trees (Regression)

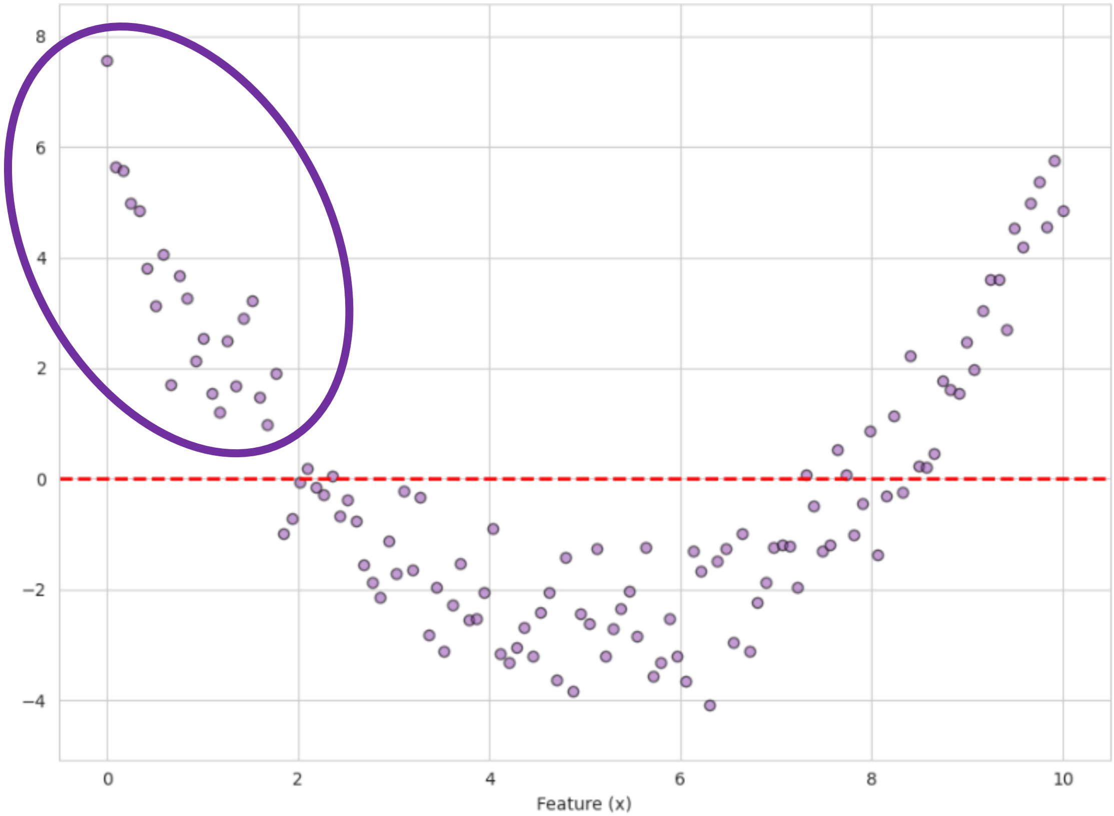


3.4 Gradient Boosted Decision Trees (Regression)



3.4 Gradient Boosted Decision Trees (Regression)

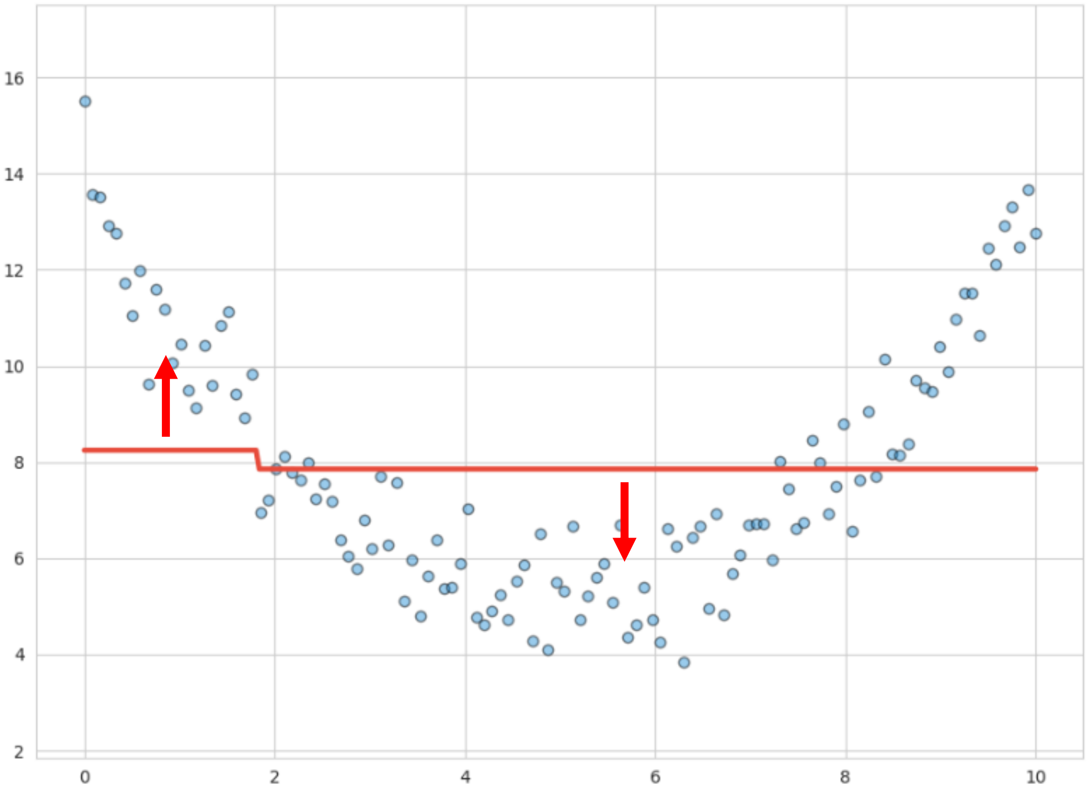
- Residuals are what is left after the model.
- I.e. $residual = y - \hat{y}$.
- This tells us how much error is associated with each data point.
- The higher the error, the greater the weight we will assign to the point in the next iteration.



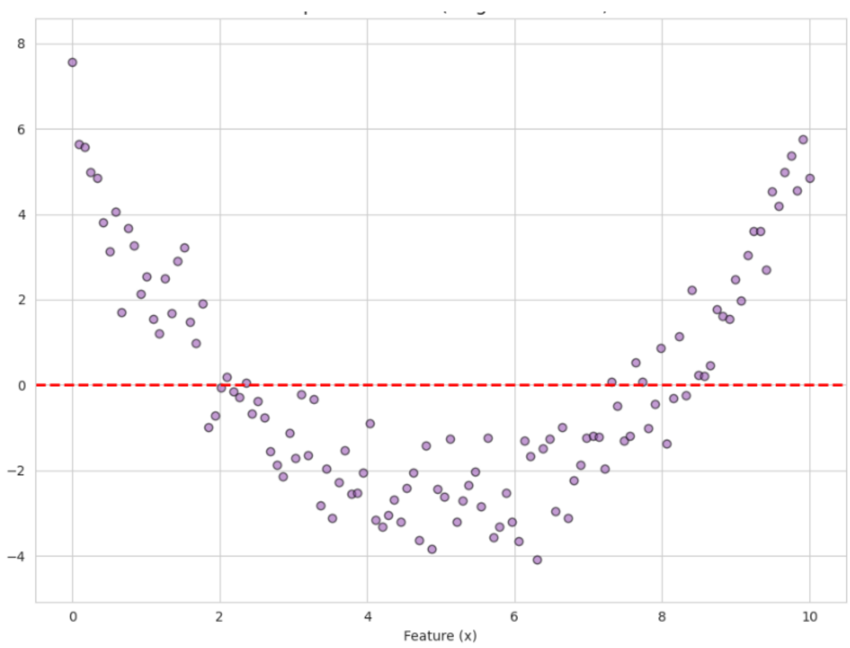
3.4 Gradient Boosted Decision Trees (Regression)

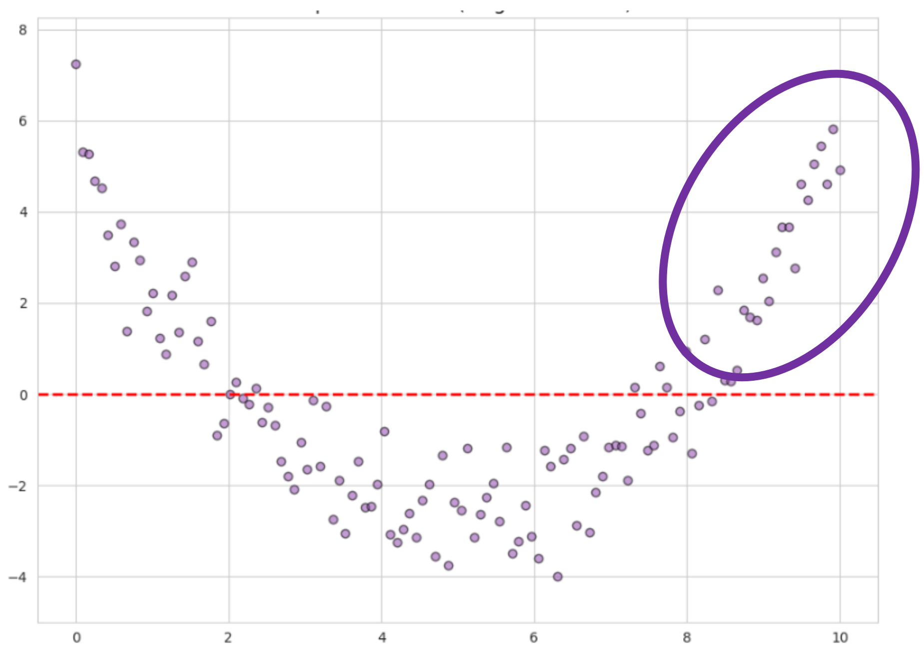
- Next we learn a new tree to modify the existing rule (based on the weights of each data point);
- In this case we will learn the tree:

  if $x \leq 1.81$:
  $\quad \hat{y} \mathrel{+}= 0.32$
  else:
  $\quad \hat{y} \mathrel{-}= 0.07$



3.4 Gradient Boosted Decision Trees (Regression)





3.4 Gradient Boosted Decision Trees (Regression)

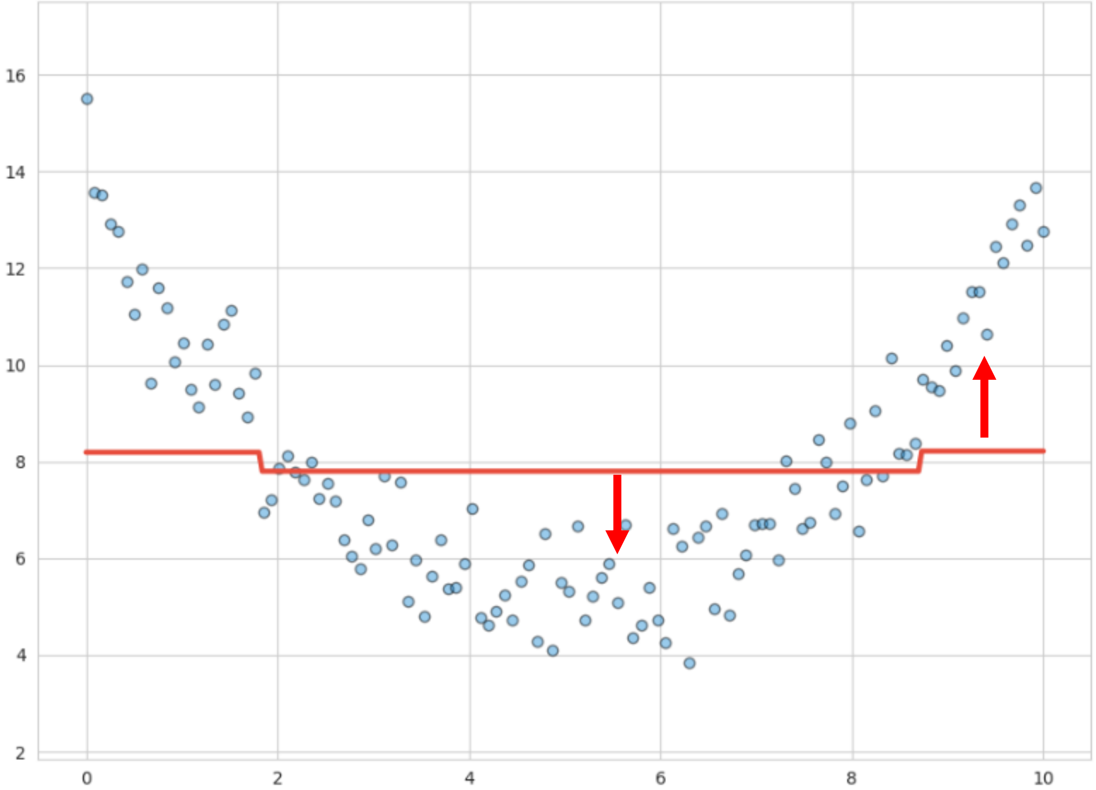
- We will again learn a new tree using the new weights;
- In this case we will learn the tree:

  `if` $x \leq 8.70$`:`

  $\hat{y} -= 0.06$

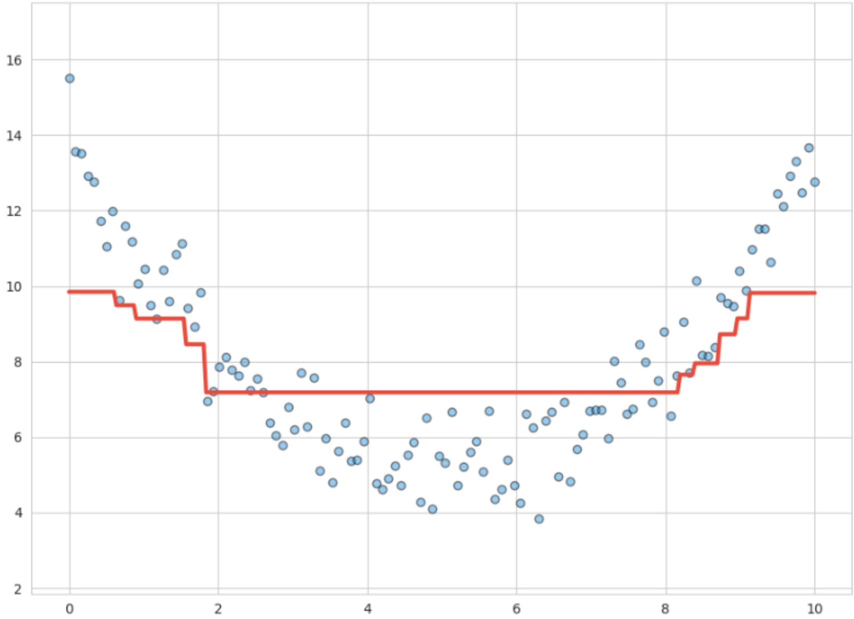
  `else:`

  $\hat{y} += 0.36$

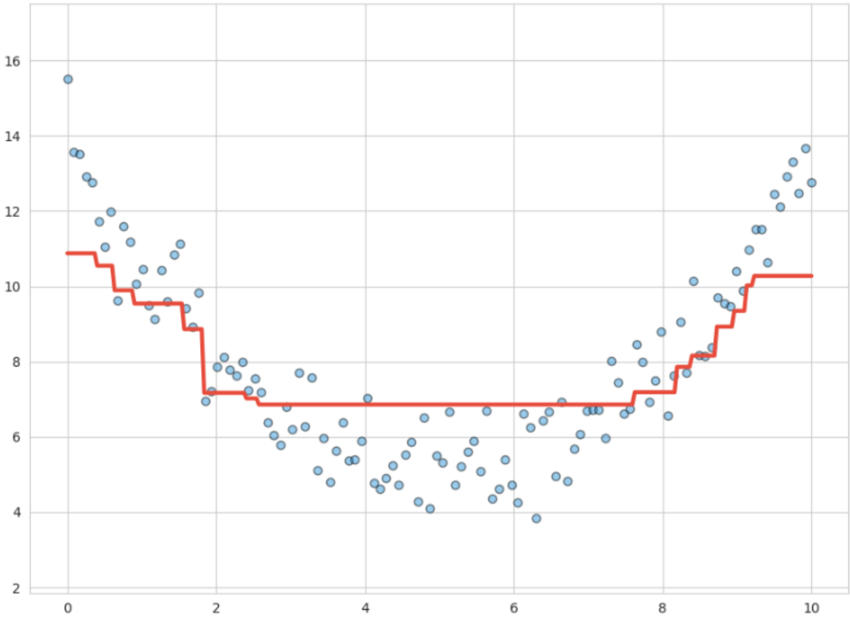


3.4 Gradient Boosted Decision Trees (Regression)

```
--- Visualising Model at Iteration 16 ---
```

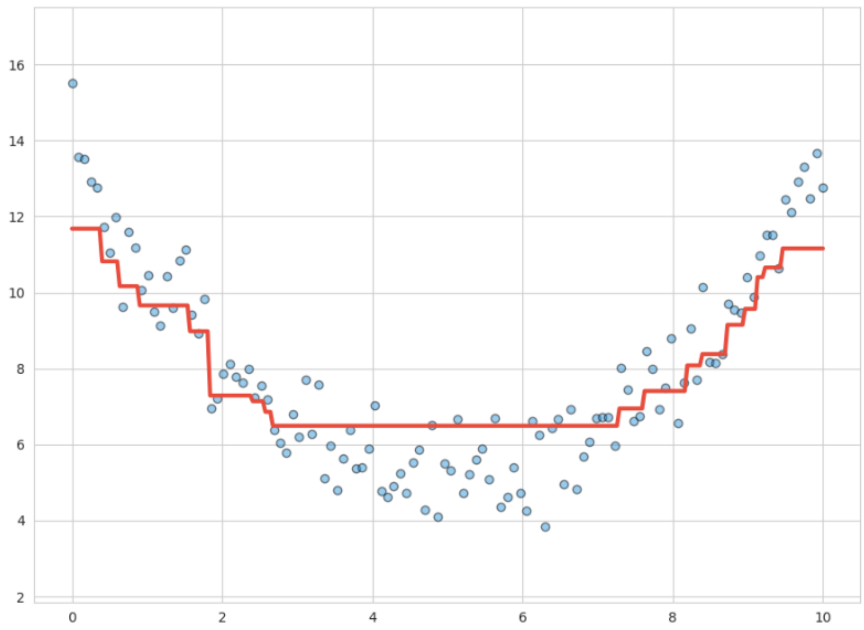



```
--- Visualising Model at Iteration 26 ---
```

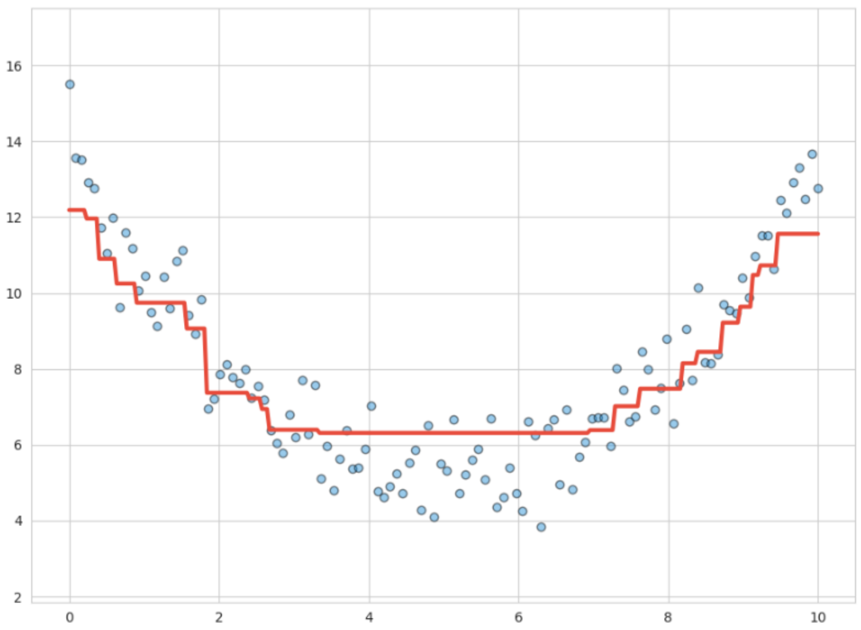


3.4 Gradient Boosted Decision Trees (Regression)

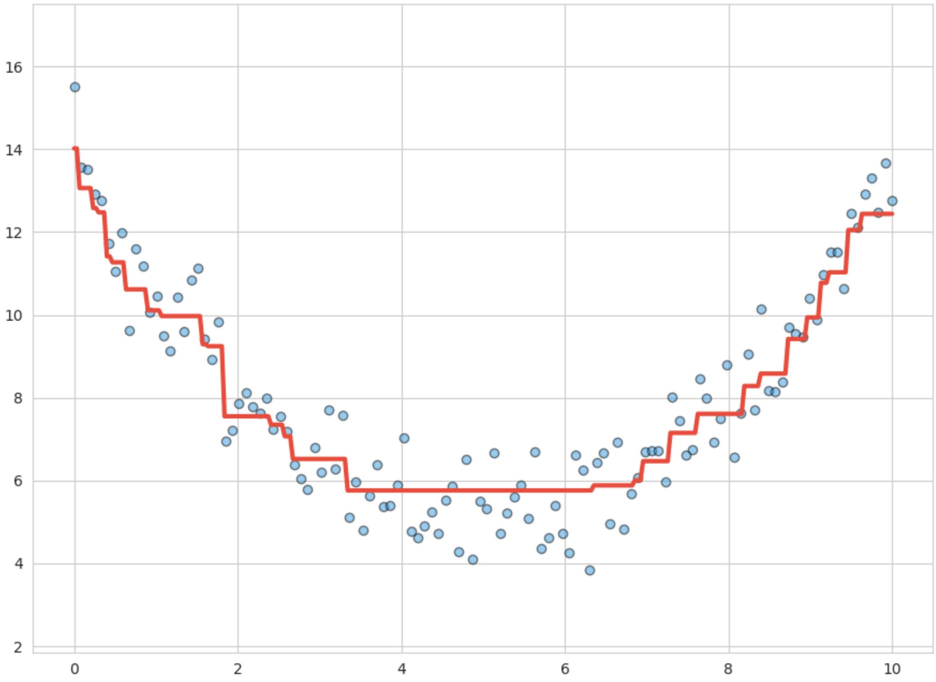
```
--- Visualising Model at Iteration 41 ---
```

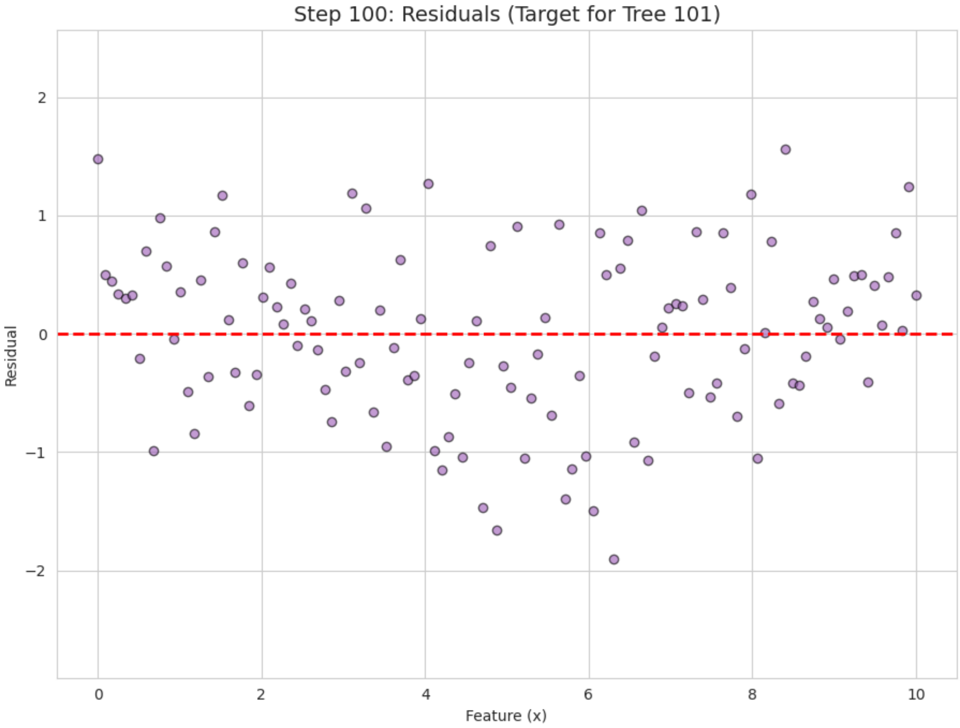



```
--- Visualising Model at Iteration 51 ---
```



3.4 Gradient Boosted Decision Trees (Regression)





3.5 GBDT: Hyperparameters

- `n_estimators`: Number of trees to build (M). Corresponds to the number of sequential trees to build.
- Loss function: How to measure error/performance – e.g. (MSE or *log loss*). **NOTE**: this is not the same as normal classification trees.
- Learning rate (η): How strongly we react to loss (e.g. how we adjust the $\hat{y}$ outputs). A very low value will take a very long time to find a good line and potentially underfit. A high value will jump around possible solutions and potentially overfit.
- The usual decision tree hyperparameters.

3.8 MDTGA

Session Aims

Introduction

Random Forests

Gradient Boosted Decision Trees (GBDT)

<u>**Asynchronous Tasks**</u>