Machine Learning INF2008

Lecture 06: Tree Based Models: Bagging: Random Forests

Donny Soh

Singapore Institute of Technology

What are Ensemble techniques?

What are the types of errors?

Specifically, all errors comprise of

- Reducible Errors
    - Bias Error
    - Variance Error
- Irreducible Errors

Irreducible errors are errors that cannot be reduced no matter what algorithm you apply. It is usually caused by unknown variables that are influencing the target y variable.

In addition, reducible errors are subdivided into two types of errors and they are known as bias and variance errors.

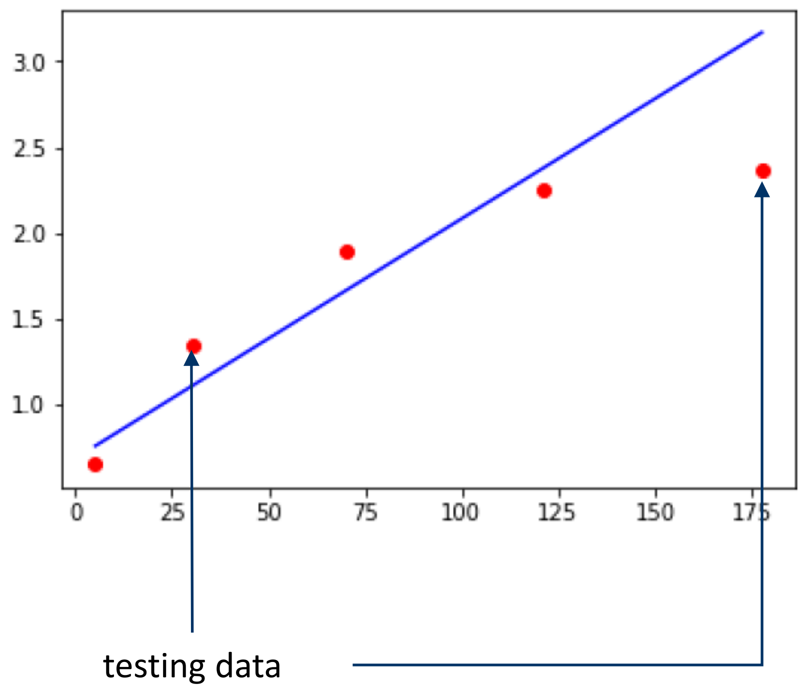
Bias errors are caused by shortcoming of algorithms, being unable to fit properly to the data.

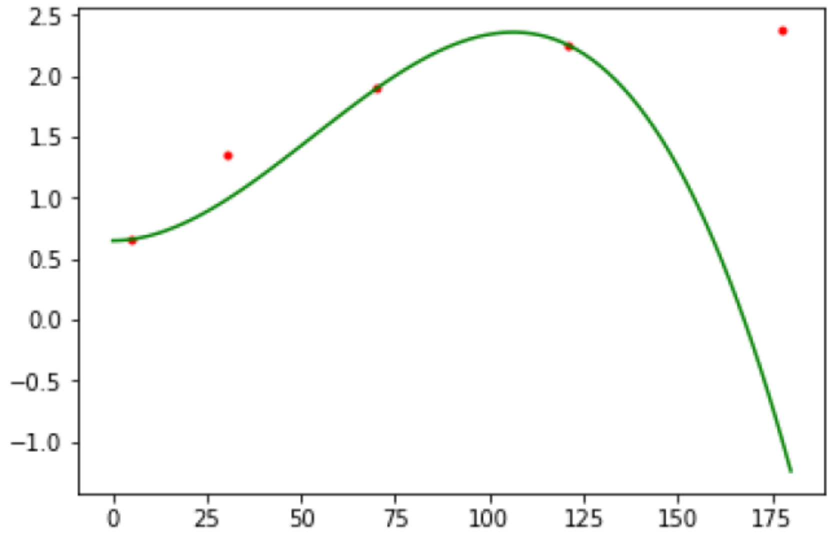
Variance errors are errors when emerge when different data is being used.

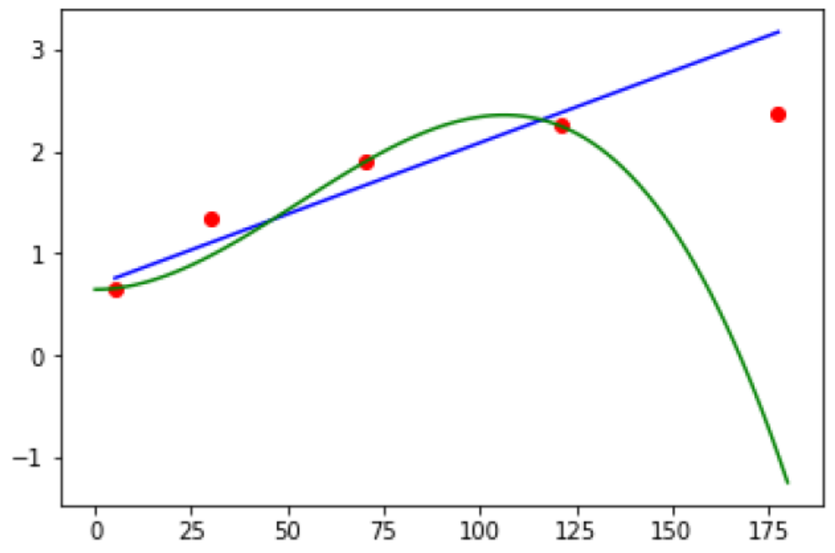
Hence in short, bias errors are an example of underfitting while variance errors are an example of overfitting.

We will now use an example to explain the difference between bias and variance errors.

Bias versus Variance Errors

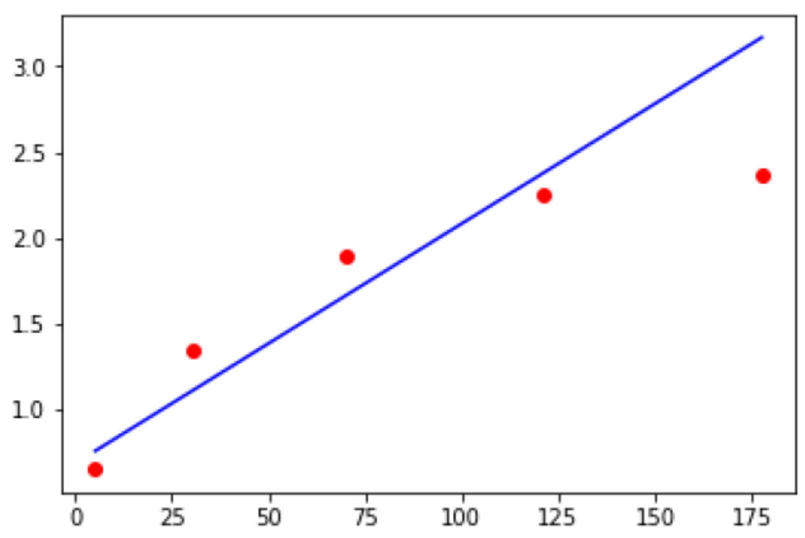


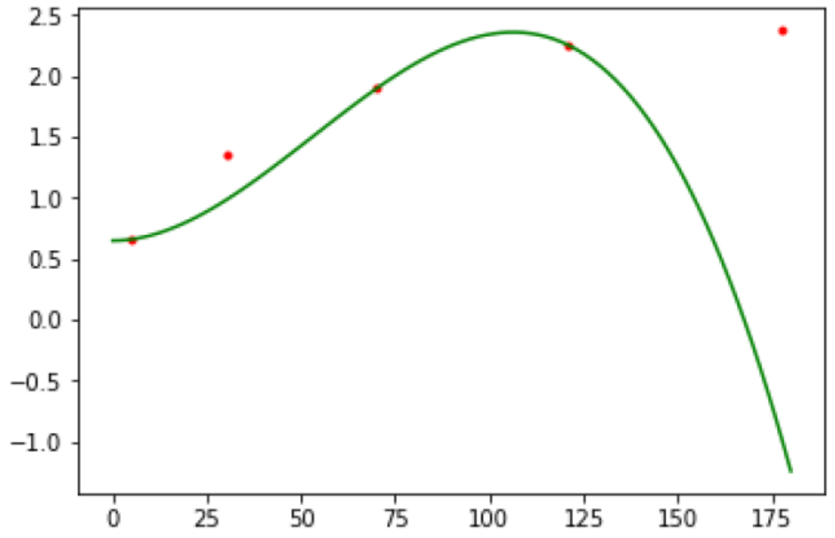


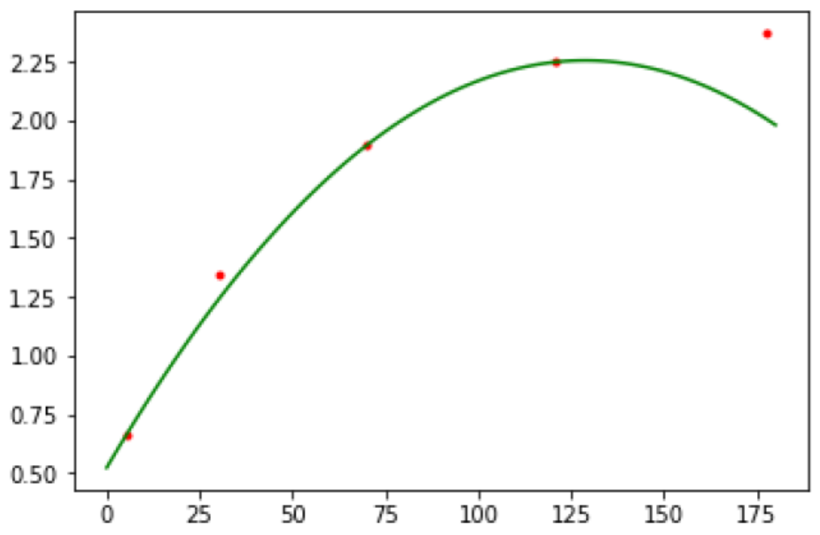


	Train Data	Test Data
Linear	0.16	0.59
Polynomial (n=3)	0	3.08

Bias versus Variance Errors







	Train Data	Test Data
Linear	0.16	0.59
Polynomial (n=3)	0	3.08
Polynomial (n=2)	0	0.27

What is Bagging?

It is a technique which targets to reduce variance errors without compromising much on the bias errors.

The most common form of bagging occurs when it is applied to decision trees. This form of bagging is commonly referred to as Random Forests.

In short, random forests generates multiple decision trees through a process known as Bootstrapping.

An Aggregated average prediction results is taken from these individual trees and hence the name Bootstrap Aggregation.

The Bootstrap Algorithm

The term bootstrap comes from the technique of generating new training datasets D_i from the original dataset D_0 by random sampling with replacement.

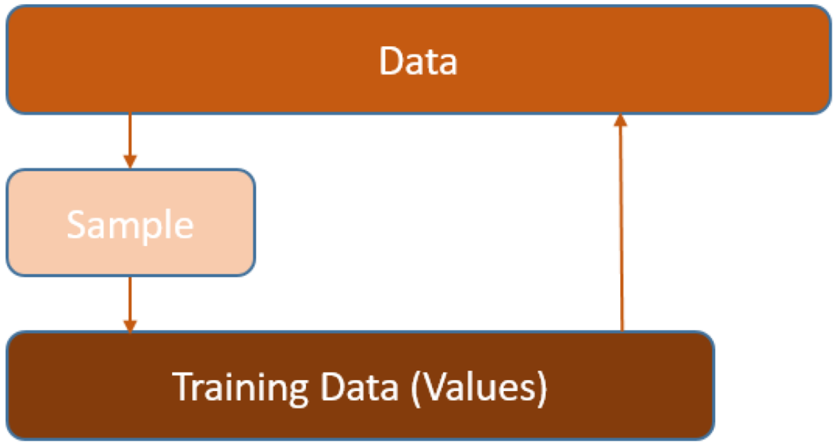
So for instance if in your original dataset D_0 you have 10,000 samples, you will generate sub-datasets D_i of size (eg) 7,500.

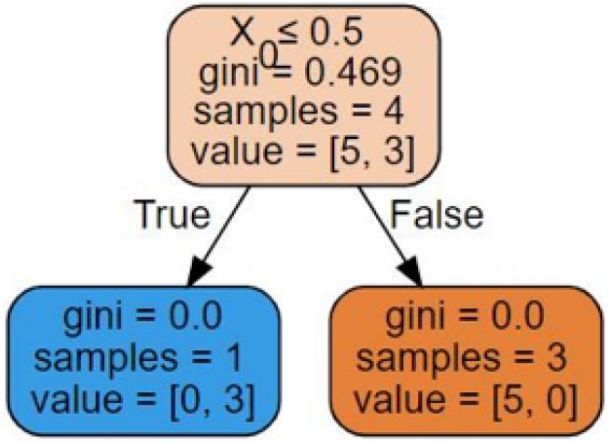
To create a dataset D_1 of size 7,500 you will draw out a single sample from original dataset D_0, place a copy of this single sample into your new dataset D_1 and then replace it back into the original dataset D_0.

This process is repeated until you obtain all 7,500 in your dataset D_1.

This process of creating the datasets is repeated for all D_1 to D_N sub-datasets.

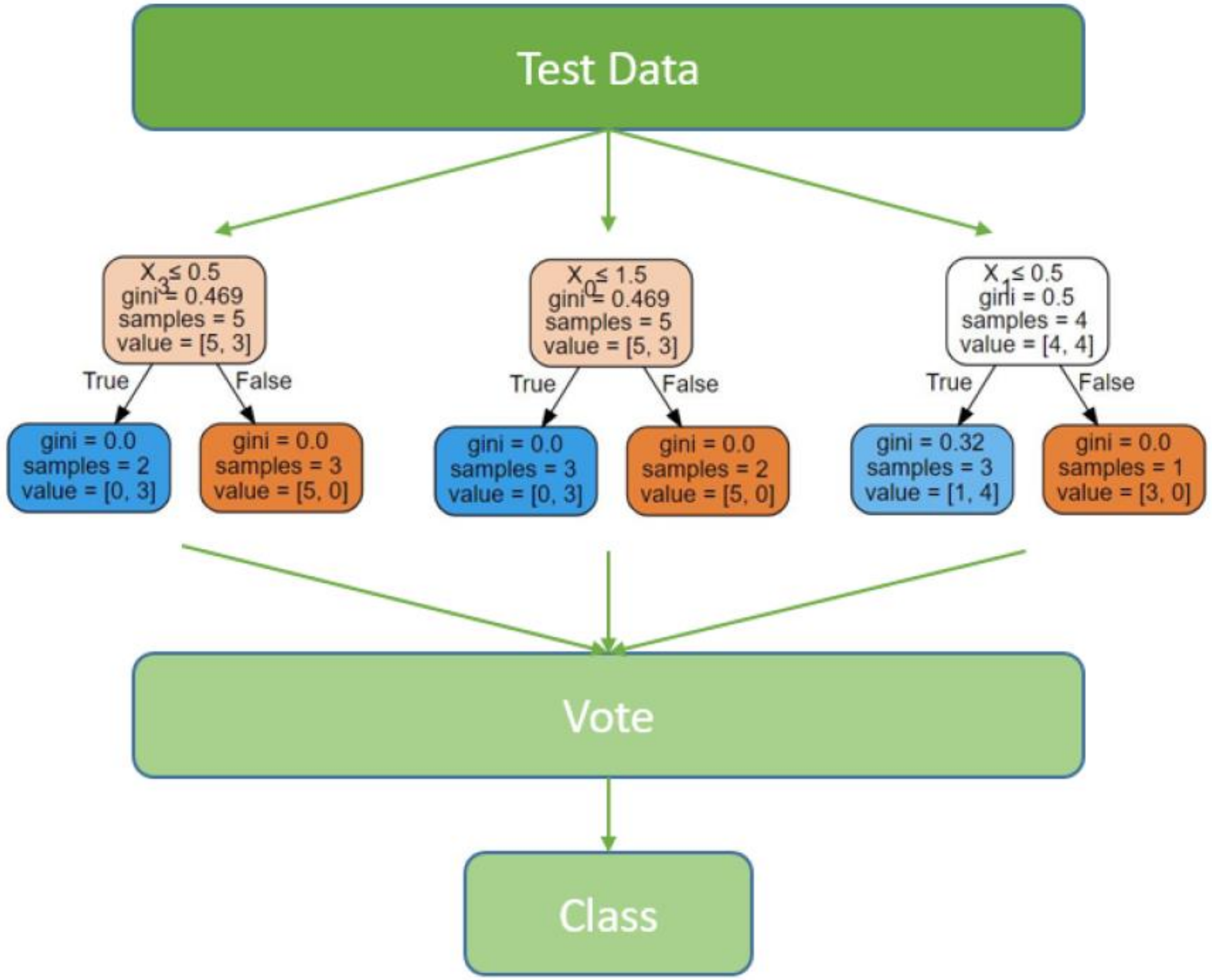
This process can be parallelized for each of the trees in the random forest.





Testing Phase

During the aggregation phase, the test data is sent over to all the individual trees and the results are aggregated over these trees.



Hyperparameters of the Random Forest Estimator

The function prototype / interface is the following:

```
RandomForestClassifier
        (n_estimators=’warn’,
        criterion=’gini’,
        max_depth=None,
        min_samples_split=2,
        min_samples_leaf=1,
        min_weight_fraction_leaf=0.0,
        max_features=’auto’,
        max_leaf_nodes=None,
        min_impurity_decrease=0.0,
        min_impurity_split=None,
        bootstrap=True,oob_score=False,
        n_jobs=None,
        random_state=None,
        verbose=0,
        warm_start=False,
        class_weight=None)
```

Feature	Meaning	Default Value
n_estimators	The number of decision trees in the random forest.	100
criterion	How good the split at the decision node is. We covered entropy. The other method is gini.	gini
max_depth	The maximum depth of the tree. A value of 1 would mean the forest is made of stumps.	-
min_samples_split	The min samples to split at an internal decision node.	2
min_samples_leaf	The min number of samples at a leaf node.	1
max_features	The number of features to consider at any split node.	sqrt(num_features)
bootstrap	Always put true, else the entire dataset will be used.	True

What is Boosting?

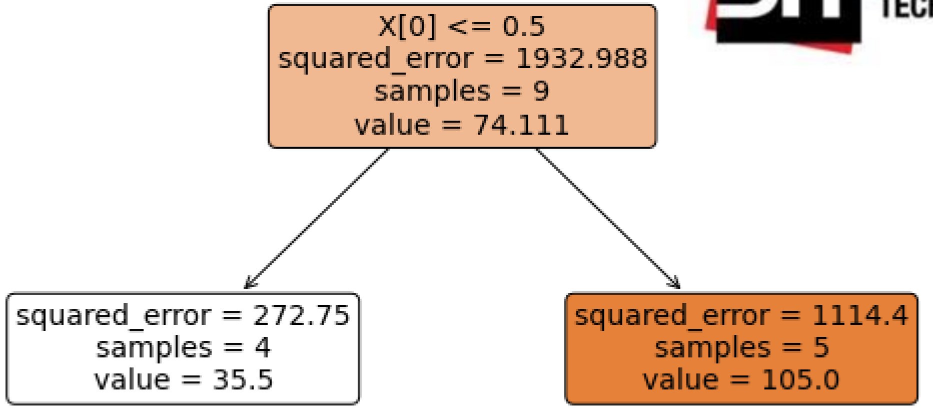
Imagine that we are trying to predict the mobile usage of a person

- The features we have access to are LikesMobileGames, LikesJohnLennon and LikesNasiLemak.
- In general, you would consider someone who LovesMobileGames to be positively / negatively correlated to the mobile data usage
- Conversely you would consider someone that loves John Lennon to be positively / negatively correlated to the data usage.
- The last feature LikesNasiLemak probably does not affect the mobile usage but in most cases you won't know. Hence decide to use all of the data. After loading the data, you see the following:

	LikesMobileGames	LikesJohnLennon	LikesNasiLemak	MobileUsage
0	0	1	1	24
1	0	1	0	26
2	0	1	0	28
3	1	1	1	46
4	0	1	1	64
5	1	0	0	90
6	1	1	1	125
7	1	0	0	130
8	1	0	1	134

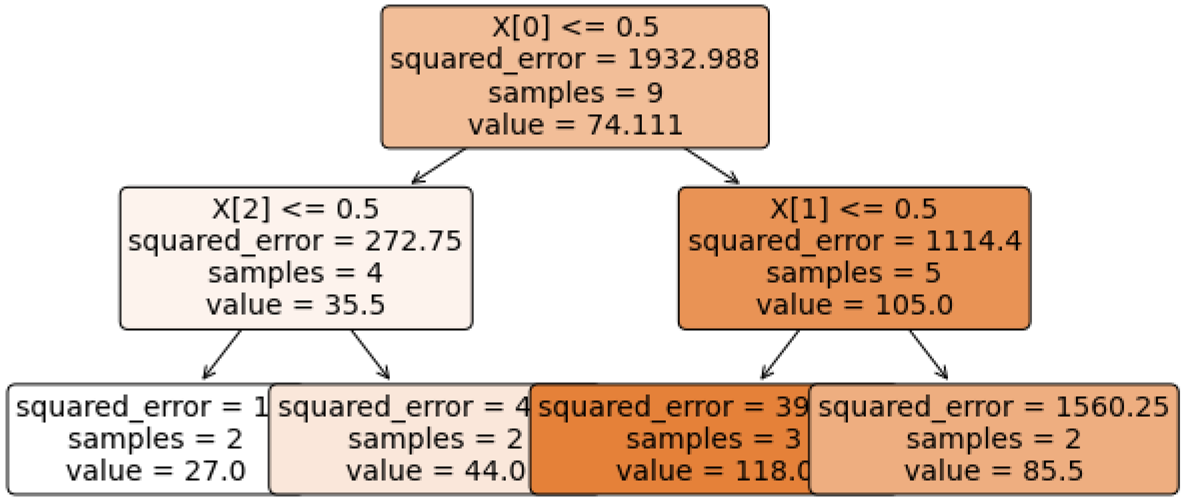
Boosting Motivation

Assuming if we create a single model with the depth of the tree set to one.



This tree is decent However it does not use the feature LikesJohnLennon, which we think will also be very important to the analysis.

If we allow the depth of the tree to be two, we end up with the following:



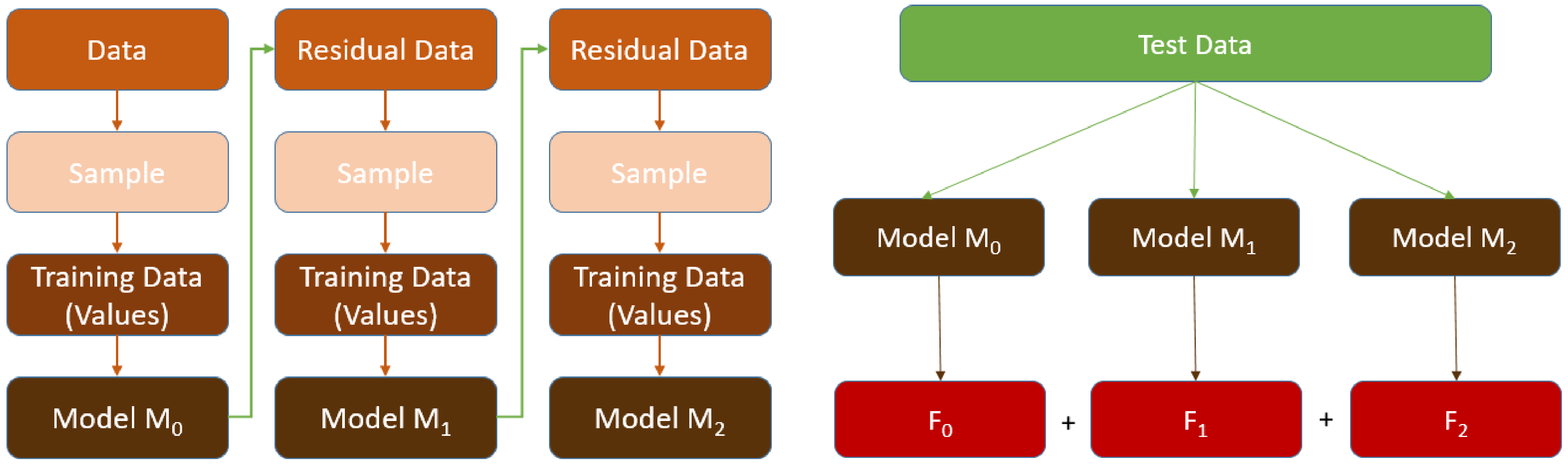
So although now we are using the feature LikesJohnLennon, we have included the probably irrelevant feature LikesNasiLemak. This example shows the issue encountered with using just one decision tree. However this issue can be resolved using ensemble learning techniques such as Boosting.

Boosting Intuition

Intuitively, the easiest way to think about tree boosting is the following:

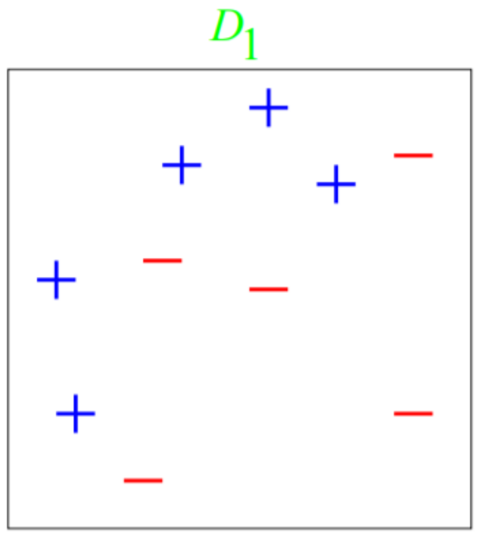
- A weak regression tree is created from the input data.
- From the weak regression tree, the residuals are calculated. The residuals are the difference between the prediction and the actual data.
- These residuals are then used to train further weak regression trees.
- The natural intuition is finding out the data that is incorrectly modelled by the first weak regression tree.
- Then subsequently creating another regression tree to cover the shortfalls of the first regression tree.

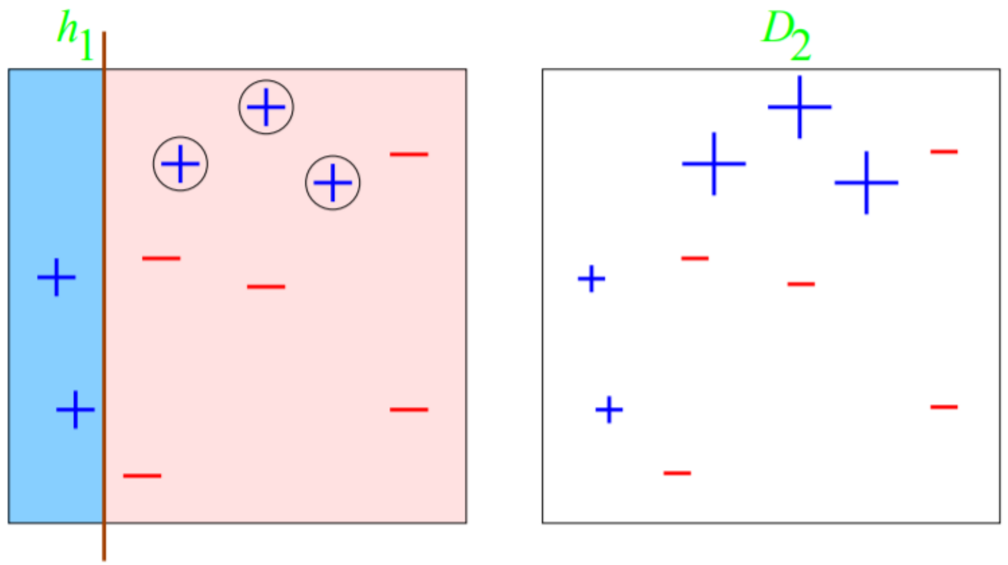
During the boosting phase, the rows and columns are also randomly sampled at each iteration. At the testing phase, the results from each model are summed up together to form the final prediction:



Boosting Toy Example (I)

Suppose we have the following data:





Next we train our first model.

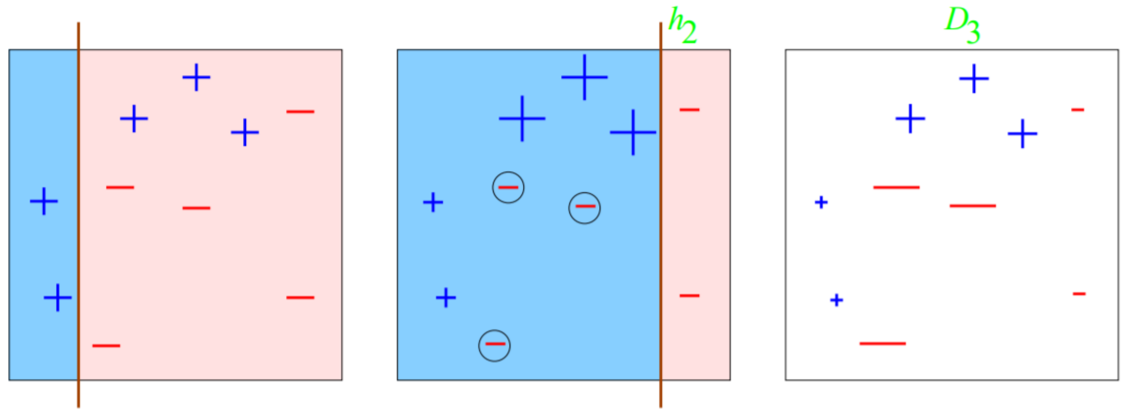
Notice that there are some data points that are predicted correctly and some that are predicted incorrectly (points that are circled).

For the points that are predicted incorrected, we would increase the weightage of these points (notice the size increase).

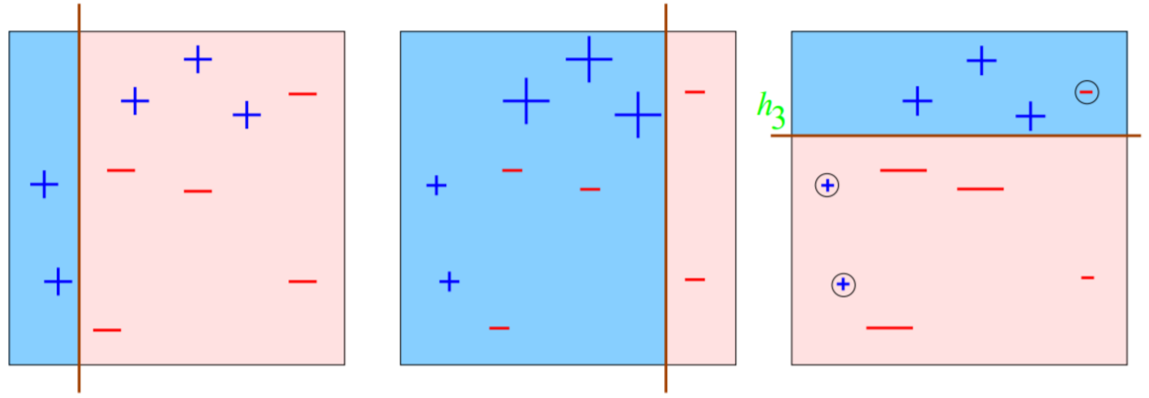
A new model is next trained on these points.

Credits: www.cs.princeton.edu/~schapire

Boosting Toy Example (II)

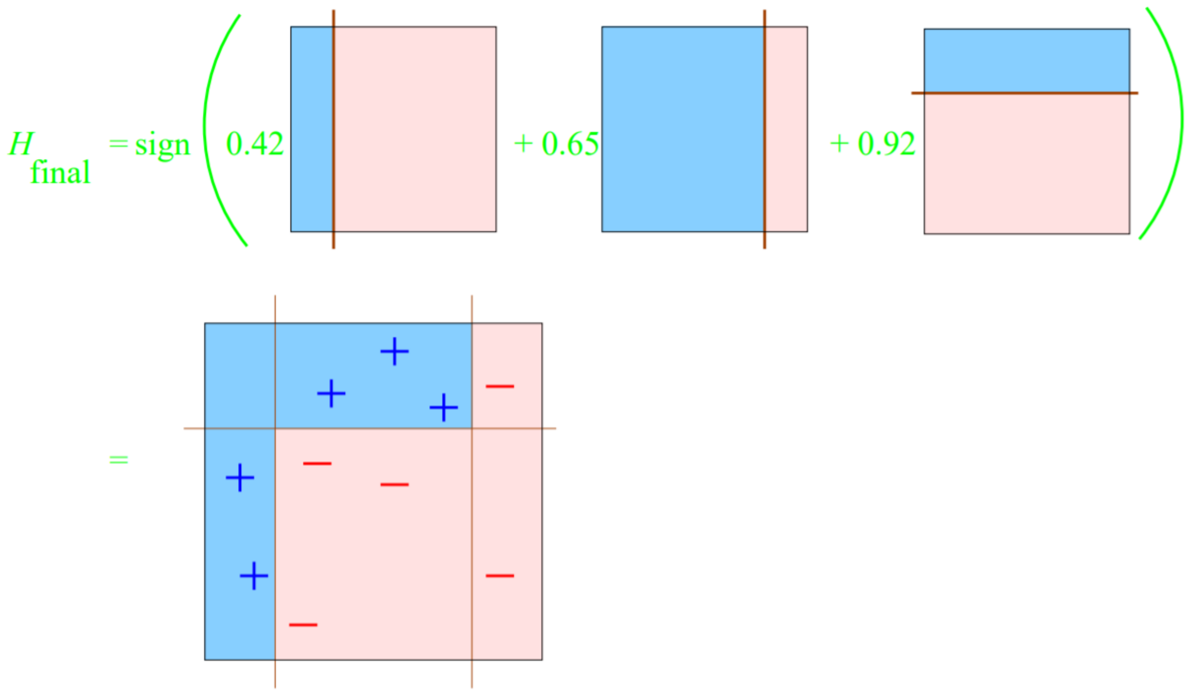


The second model is trained. Similarly the data points that are incorrectly predicted will have their weightage increased.



This is continued until the final third model.

Boosting Toy Example (III)

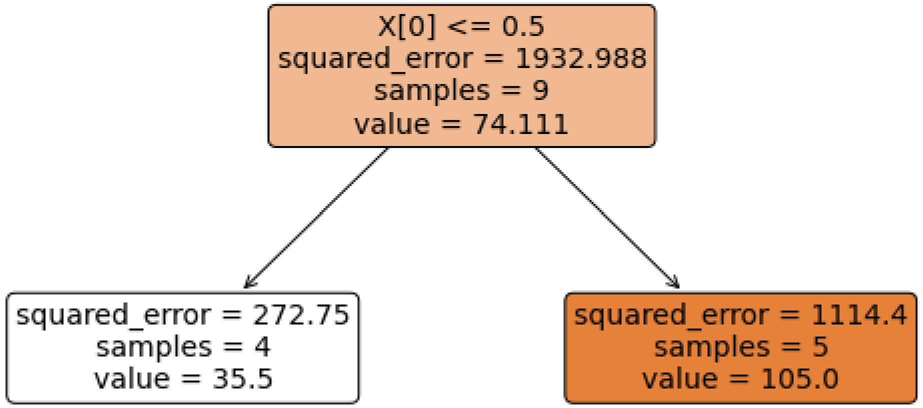

$H_{\text{final}} = \text{sign}\left(0.42 \ + 0.65 \ + 0.92 \right)$
=
+
−

In this case, we stop training after three models. We are able to combine all these three models to form an ensemble classifier that looks like the following:

Nasi Lemak Example (I)

	LikesMobileGames	LikesJohnLennon	LikesNasiLemak	MobileUsage
0	0	1	1	24
1	0	1	0	26
2	0	1	0	28
3	1	1	1	46
4	0	1	1	64
5	1	0	0	90
6	1	1	1	125
7	1	0	0	130
8	1	0	1	134

Truth	F_0	R_0	h_0	F_1	R_1
24	74.1	-50.1	-38.6	35.5	-11.5
26	74.1	-48.1	-38.6	35.5	-9.5
28	74.1	-46.1	-38.6	35.5	-7.5
46	74.1	-28.1	30.9	105	-59
64	74.1	-10.1	-38.6	35.5	28.5
90	74.1	15.9	30.9	105	-15
125	74.1	50.9	30.9	105	20
130	74.1	55.9	30.9	105	25
134	74.1	59.9	30.9	105	29

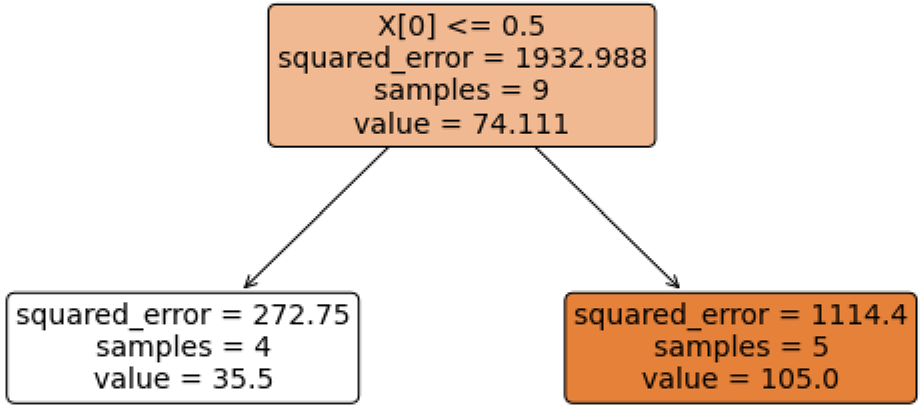


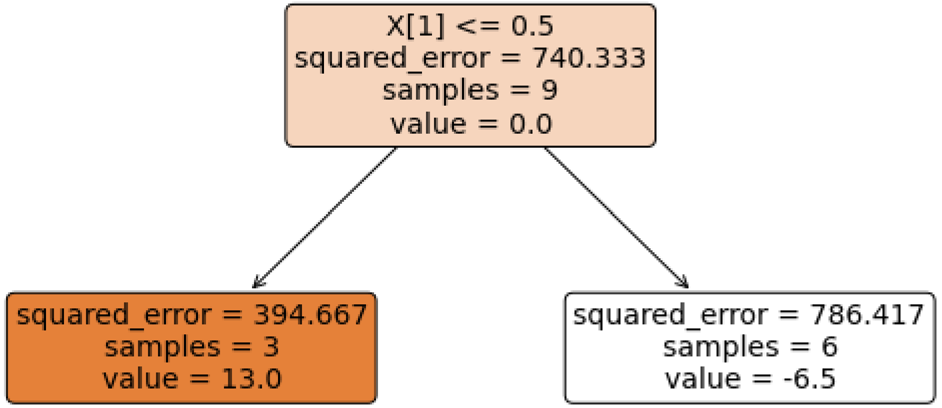
Nasi Lemak Example (II)

	LikesMobileGames	LikesJohnLennon	LikesNasiLemak	MobileUsage
0	0	1	1	24
1	0	1	0	26
2	0	1	0	28
3	1	1	1	46
4	0	1	1	64
5	1	0	0	90
6	1	1	1	125
7	1	0	0	130
8	1	0	1	134

Truth	F_0	R_0	h_0	F_1	R_1
24	74.1	-50.1	-38.6	35.5	-11.5
26	74.1	-48.1	-38.6	35.5	-9.5
28	74.1	-46.1	-38.6	35.5	-7.5
46	74.1	-28.1	30.9	105	-59
64	74.1	-10.1	-38.6	35.5	28.5
90	74.1	15.9	30.9	105	-15
125	74.1	50.9	30.9	105	20
130	74.1	55.9	30.9	105	25
134	74.1	59.9	30.9	105	29

Truth	F_0	R_0	h_0	F_1	R_1	h_1	F_2
24	74.1	-50.1	-38.6	35.5	-11.5	-6.5	29
26	74.1	-48.1	-38.6	35.5	-9.5	-6.5	29
28	74.1	-46.1	-38.6	35.5	-7.5	-6.5	29
46	74.1	-28.1	30.9	105	-59	-6.5	98.5
64	74.1	-10.1	-38.6	35.5	28.5	-6.5	29
90	74.1	15.9	30.9	105	-15	13	118
125	74.1	50.9	30.9	105	20	-6.5	98.5
130	74.1	55.9	30.9	105	25	13	118
134	74.1	59.9	30.9	105	29	13	118


X[0] <= 0.5
squared_error = 1932.988
samples = 9
value = 74.111
squared_error = 272.75
samples = 4
value = 35.5
squared_error = 1114.4
samples = 5
value = 105.0


X[1] <= 0.5
squared_error = 740.333
samples = 9
value = 0.0
squared_error = 394.667
samples = 3
value = 13.0
squared_error = 786.417
samples = 6
value = -6.5