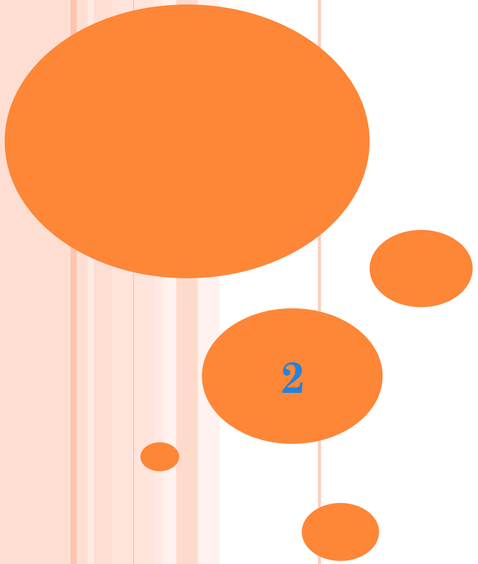
Machine Learning with Python

Ensemble Method (Bagging and Boosting)

Arghya Ray

Ensemble Methods: Bagging and Boosting



Rationale

- The ensemble classifier is likely to have a lower error rate (boosting)
- The variance of the ensemble classifier will be lower than had we used certain unstable classification models (such as decision trees and neural nets) that have high variability (bagging and boosting)
- Suppose we have an ensemble of binary classifiers each with error rate 0.20. If the individual classifiers agree the error rate will be the same as for the individual classifiers, the ensemble classifier will make an error only when the majority of individual classifiers make an error

Rationale (cont.)

Let ε represent the individual classifier error rate. The probability that k of the 5 individual classifiers will make the same wrong prediction is:

$$\binom{5}{k}\epsilon^k(1-\epsilon)^{5-k} = \binom{5}{k}0.2^k(1-0.2)^{5-k}$$

And the probability that all 5 of the classifiers will make an error is:

$$\binom{5}{5}0.2^5(0.8)^0 = 0.00032$$

And the error rate for the ensemble is:

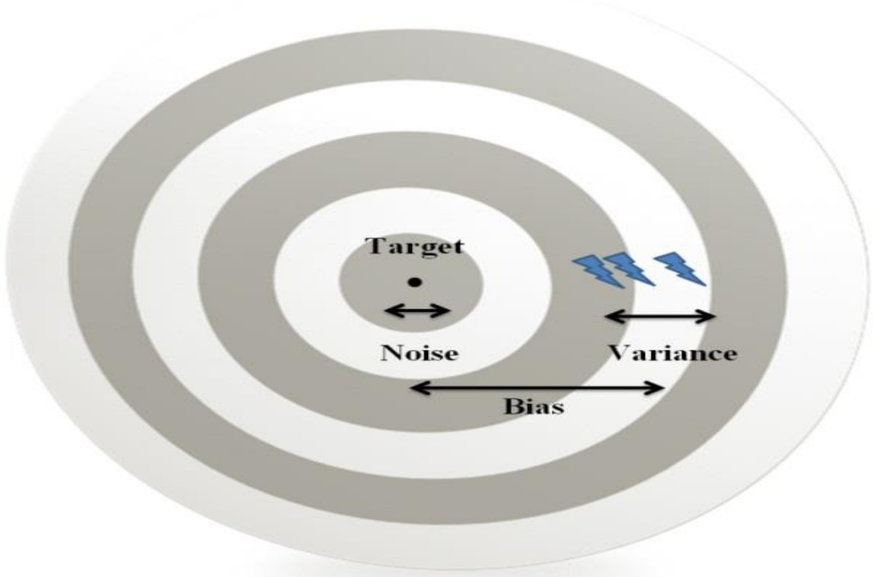
$$Error\ rate_{\ Ensemble\ classifier} = \sum_{i=3}^{5}\binom{5}{i}\epsilon^i(1-\epsilon)^{5-i} = \sum_{i=3}^{5}\binom{5}{i}0.2^i(0.8)^{5-i}$$
$$= 0.0512 + 0.0064 + 0.0003 = 0.05792$$

Bias, Variance, and Noise

- We would like our models to have a low prediction error, which can be decomposed as:

$$(y - \hat{y}) = Bias + Variance + Noise$$

- Where *Bias* refers to the average distance between the predictions ($\hat{y}$, represented by the lightning darts in Figure 25.1) and the target (y, the bull's eye),
- And *Variance* measures the variability in the predictions $\hat{y}$ themselves, and
- And *Noise* represents the lower bound on the prediction error that the predictor can possibly achieve.



Bias, Variance, and Noise (cont.)

- To reduce prediction error we need to reduce the bias, variance or noise. Unfortunately noise is an intrinsic characteristic of the prediction problem and can not be reduced.
- Bagging can reduce the variance of the classifier models
- Boosting can reduce both bias and variance and thus offers a way to short-circuit the *bias-variance tradeoff*, where efforts to reduce bias necessarily increase the variance and vice-versa.

When to Apply Bagging

"Some classification and regression methods are unstable in the sense that small perturbations in their training sets or in construction may result in large changes in the constructed predictor." (Breiman 1998)

Classification Algorithm	Stable or Unstable
Classification and Regression Trees	Unstable
C4.5	Unstable
Neural Networks	Unstable
k-Nearest Neighbor	Stable
Discriminant Analysis	Stable
Naïve Bayes	Stable

When to Apply Bagging (cont.)

- Bagging works best with unstable models where there is room for improvement in reducing variability as it is a method for reducing variance.

- Applying bagging to stable models can degrade their performance

- Bagging works with bootstrap samples of the original data, each of which contains only about 63% of the data

Bagging

Bagging coined by Leo Breiman to refer to the following *Boostrap Aggregating* Algorithm:

1. Samples (with replacement) are repeatedly taken from the training data set, so that each record has an equal probability of being selected, and each sample is the same size as the original training data set. These are the bootstrap samples.

2. A classification or estimation model is trained on each bootstrap sample drawn in Step 1, and a prediction is recorded for each sample.

3. The bagging ensemble prediction is then defined to be the class with the most votes in Step 2 (for classification models) or the average of the predictions made in Step 2 (for estimation models).

Bagging Example

- Consider a small data set in which x is the variable value and y is the classification.

x	0.2	0.4	0.6	0.8	1
y	1	0	0	0	1

- We have a one-level decision tree classifier that chooses a value of k to minimize leaf node entropy

- If bagging is not used the best the classifier can do is a 20% error rate with a split at $x \leq 0.3$ or $x \leq 0.9$

Bagging Example (cont.)

Step 1: Bootstrap samples are taken, and Step 2: one-level decision tree classifiers (base classifiers) are trained on each sample.

Bootstrap Sample							Base Classifier
1	x	0.2	0.2	0.4	0.6	1	$x \leq 0.3 \Rightarrow y = 1$
	y	1	1	0	0	1	$otherwise\ y = 0$
2	x	0.2	0.4	0.4	0.6	0.8	$x \leq 0.3 \Rightarrow y = 1$
	y	1	0	0	0	0	$otherwise\ y = 0$
3	x	0.4	0.4	0.6	0.8	1	$x \leq 0.9 \Rightarrow y = 0$
	y	0	1	0	0	1	$otherwise\ y = 1$
4	x	0.2	0.6	0.8	1	1	$x \leq 0.9 \Rightarrow y = 0$
	y	1	0	0	1	1	$otherwise\ y = 1$
5	x	0.2	0.2	1	1	1	$x \leq 0.1 \Rightarrow y = 0$
	y	1	1	1	1	1	$otherwise\ y = 1$

Bagging Example (cont.)

Step 3. For each record tally votes and select the majority class. Since we have 0/1 classifier the majority is the proportion of 1's. If the proportion is less than 0.5 then the bagging prediction is 0, otherwise 1.

Bootstrap Sample	$x = 0.2$	$x = 0.4$	$x = 0.6$	$x = 0.8$	$x = 1$
1	1	0	0	0	0
2	1	0	0	0	0
3	0	0	0	0	1
4	0	0	0	0	1
5	1	1	1	1	1
Proportion	0.6	0.2	0.2	0.2	0.6
Bagging Prediction	1	0	0	0	1

Boosting

Boosting is an *adaptive* algorithm developed by Freund and Schapire in 1990's and reduces both error due to variance and error due to bias:

1. All observations have equal weight in the original training data set D_1. An initial "base" classifier h_1 is determined.
2. The observations that were incorrectly classified by the previous base classifier have their weights increased, while the observations that were correctly classified have their weights decreased. This gives us data distribution $D_m, m = 2, \ldots, M$. A new base classifier $h_m, m = 2, \ldots, M$ is determined, based on the new weights. This step is repeated until the desired number of iterations M is achieved.
3. The final boosted classifier is the weighted sum of the M base classifiers.

Boosting Step 1:

Training data D_1 consists of 10 dichotomous values as shown below. An initial base classifier h_1 is determined to separate the two leftmost values. Shaded area represents values classified as "+". Boxed values are those incorrectly classified.

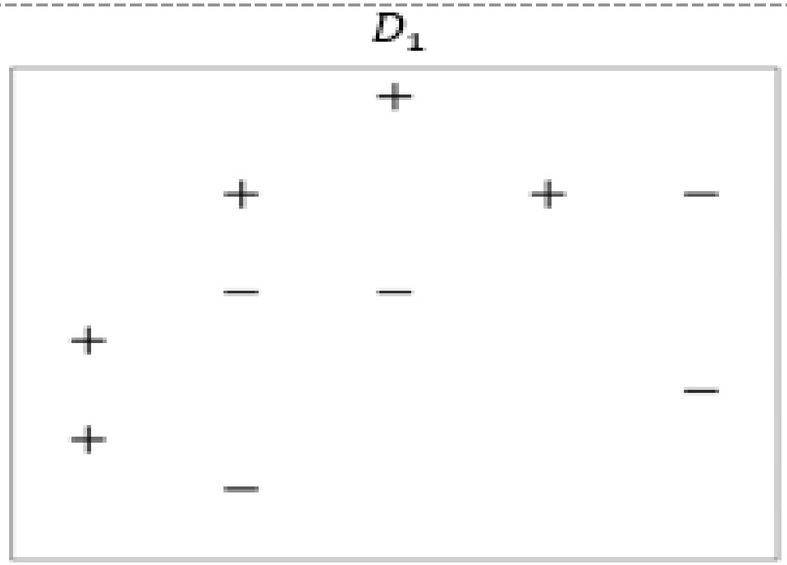


Figure 25.2 Original data

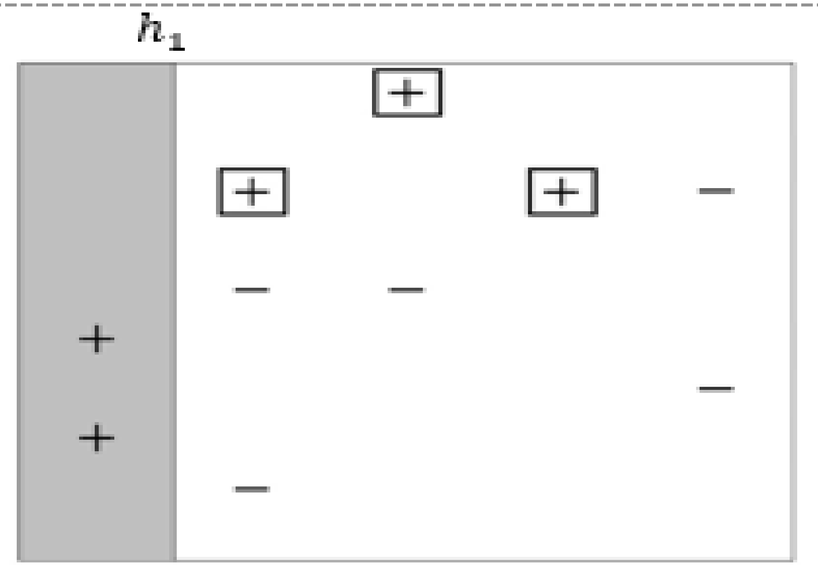


Figure 25.3 Initial base classifier

Boosting Step 2 (first pass):

3 values incorrectly classified by h_1 have weights (represented by relative size in diagrams) increased while other 7 have weights decreased. Based on the new weights a new classifier h_2 is determined

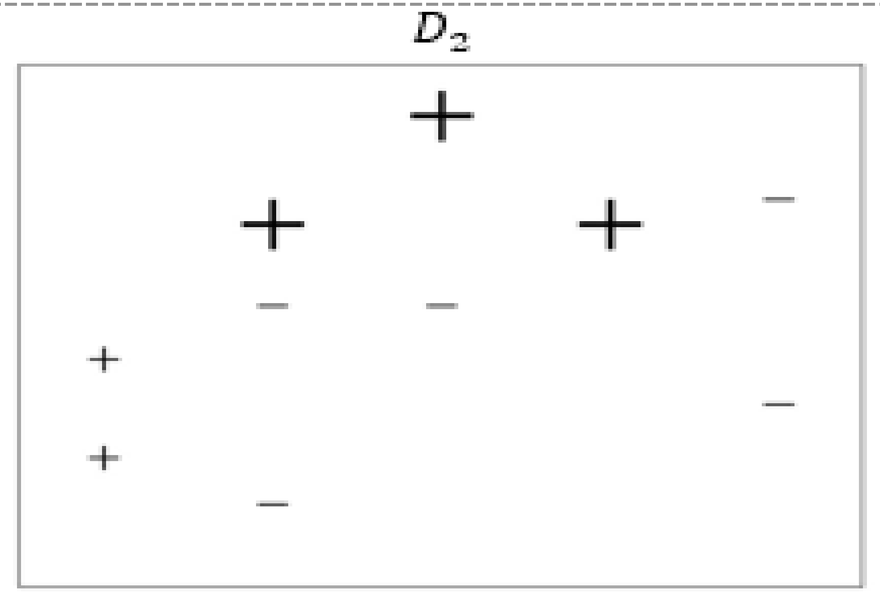


Figure 25.4 First reweighting of the data.

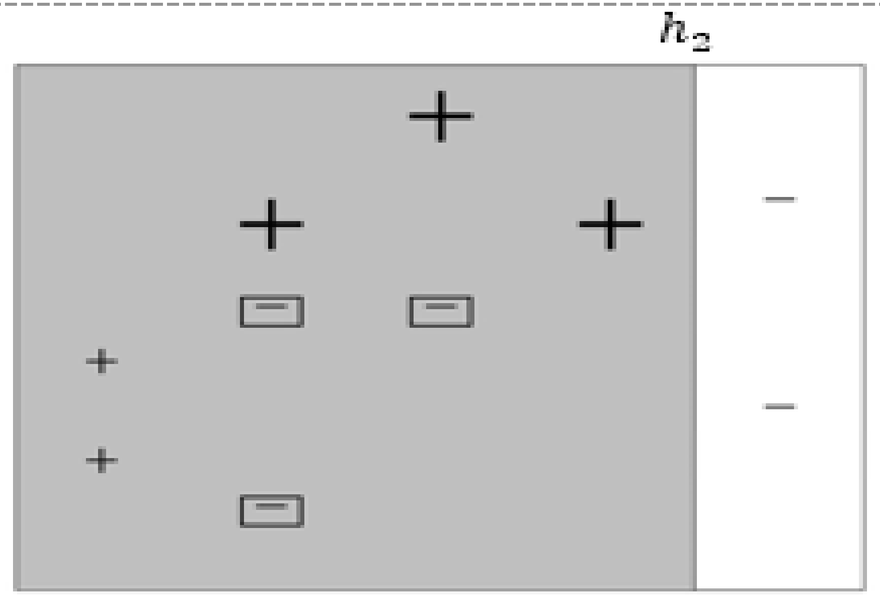


Figure 25.5 Second base classifier.

Boosting Step 2 (second pass):

3 values incorrectly classified by h_2 have weights increased while other 7 have weights decreased. Based on the new weights a new classifier h_3 is determined

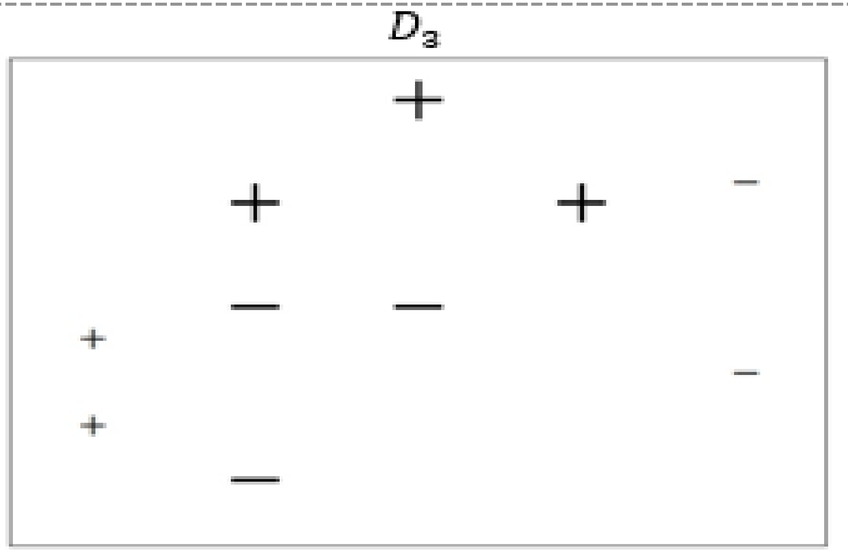


Figure 25.6 Second reweighting of the data.

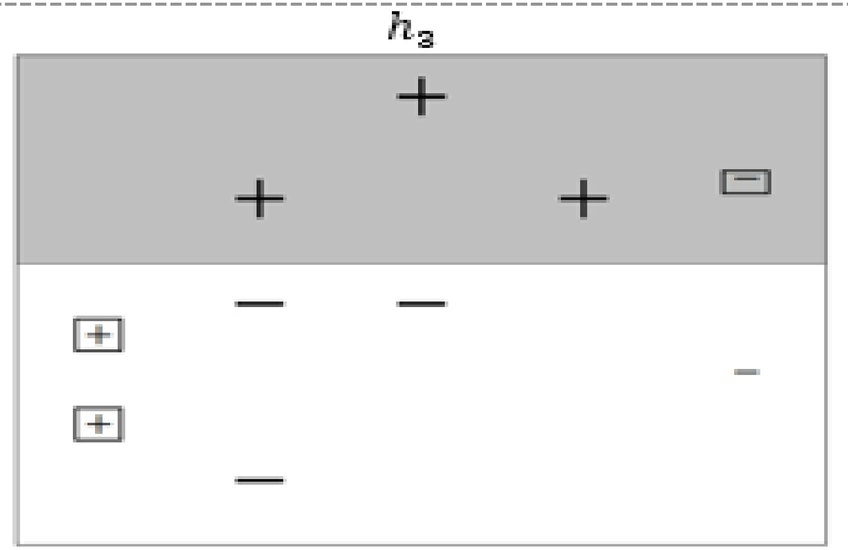
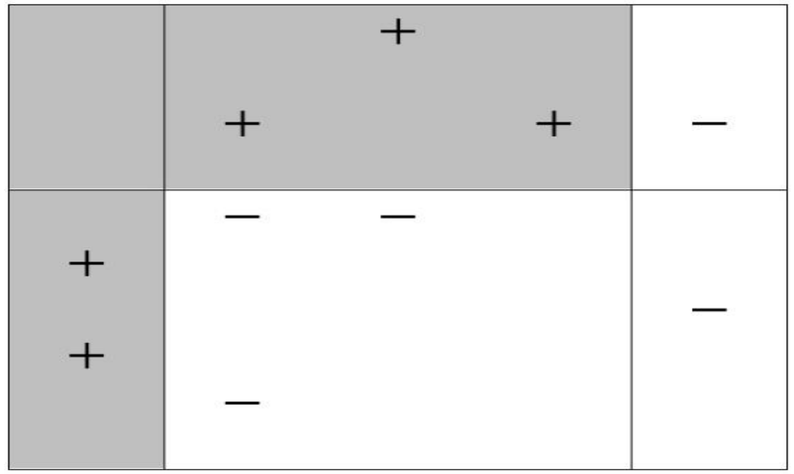


Figure 25.7 Third base classifier.

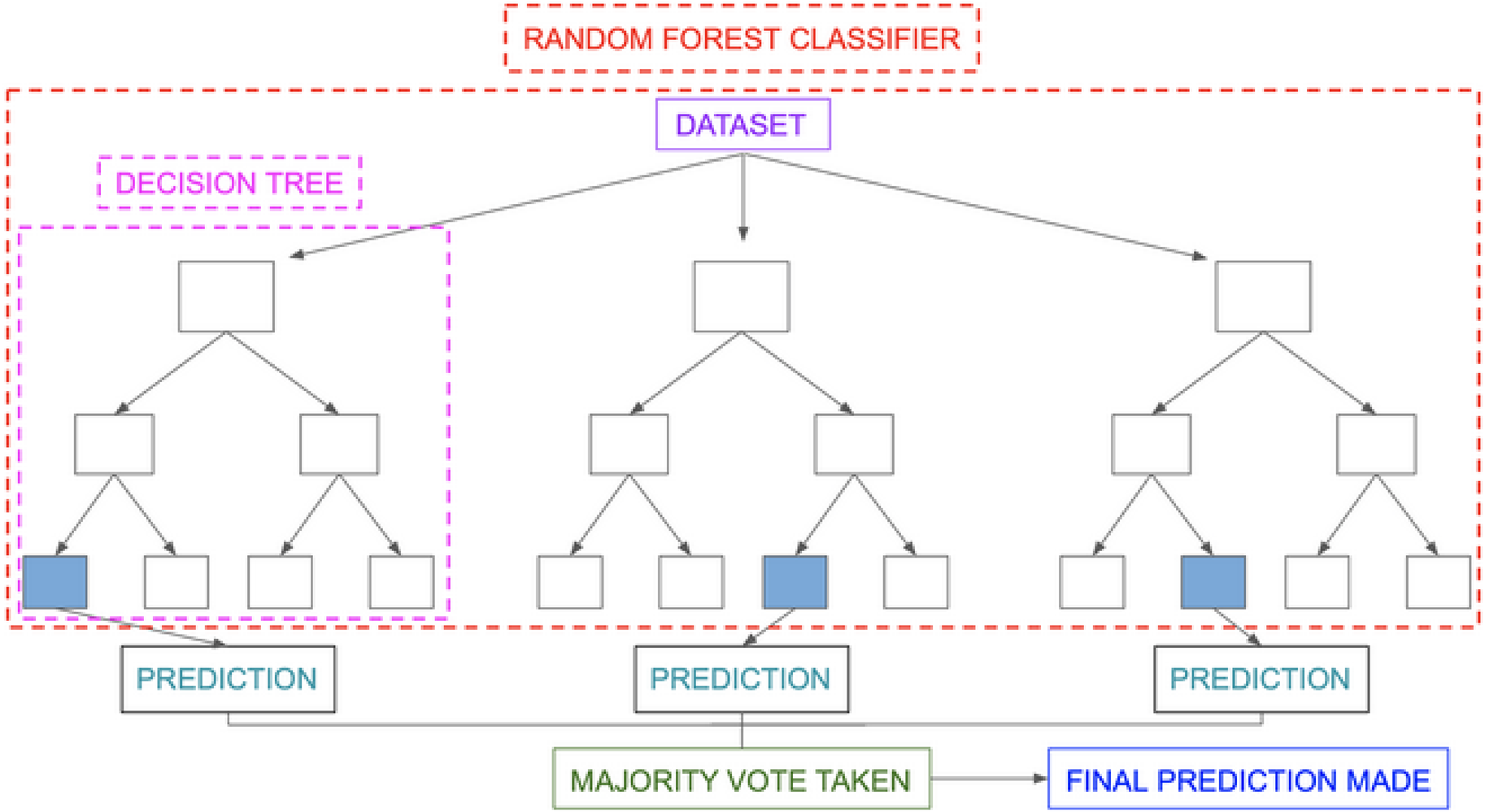
Boosting Step 3:

Final boosted classifier is the weighted sum of the $M = 3$ base classifiers: $\alpha_1 h_1 + \alpha_2 h_2 + \alpha_3 h_3$.



- Weights assigned each classifier proportional to accuracy of classifier
- Boosting performs best when base classifiers area unstable
- Boosting can increase the variance when the base classifier is stable

Random Forest



Good Article: https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/

https://towardsdatascience.com/random-forest-3a55c3aca46d

Classification in random forests:

Classification in random forests employs an ensemble methodology to attain the outcome. The training data is fed to train various decision trees. This dataset consists of observations and features that will be selected randomly during the splitting of nodes.

A rain forest system relies on various decision trees. Every decision tree consists of decision nodes, leaf nodes, and a root node. The leaf node of each tree is the final output produced by that specific decision tree. The selection of the final output follows the majority-voting system. In this case, the output chosen by the majority of the decision trees becomes the final output of the rain forest system.

Regression in random forests

Regression is the other task performed by a random forest algorithm. A random forest regression follows the concept of simple regression. Values of dependent (features) and independent variables are passed in the random forest model.

In a random forest regression, each tree produces a specific prediction. The mean prediction of the individual trees is the output of the regression. This is contrary to random forest classification, whose output is determined by the mode of the decision trees' class.

The content of the slides are prepared from different textbooks.

References:

- Data Mining and Predictive Analytics, By Daniel T. Larose. Copyright 2015 John Wiley & Sons, Inc.
- Predictive Analytics for Dummies, By Anasse Bari, Mohamed Chaouchi, & Tommy Jung, Copyright 2016, John Wiley & Sons, Inc.
- Introduction to Data Mining with Case Studies, By G.K. Gupta. Copyright 2014 by PHI Learning Private Limited.


Thank you..