**Chapter 7: Ensemble Learning and Random Forests**

# Ensemble: The group of predictors is called an ensemble, and the Ensemble Learning algorithm is called an Ensemble Method.

@ <https://medium.com/ml-research-lab/ensemble-learning-the-heart-of-machine-learning-b4f59a5f9777> ---> Contains Important Sublinks

@ <https://towardsdatascience.com/practical-guide-to-ensemble-learning-d34c74e022a0>

@ <https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models/>

# An ensemble of decision trees is known as Random Forest.

# Hard Voting classifier: In this, we aggregate the predictions of many classifiers, and the final prediction result is the one with the most votes.

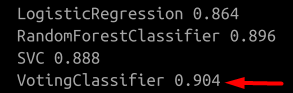
**#** Even if classifiers are **weak learners ( which does slightly better than random guessing), the ensemble can still be a strong learner (achieving high accuracy), provided there are many weak learners,** and they are **sufficiently diverse.**

# Law of Large Number: As the number of trails or observations increases, the actual or observed probability approaches the theoretical or expected probability.

E.g., Suppose an ensemble containing 1000 classifiers that are individually correct only 50% of the time ( barely better than random guessing). If you predict the **majority voted** class, you can hope for up to **75% accuracy**!

# **But** this is true only when the **classifiers are pertly independent, making no correlated errors**.

# One way to have diverse classification is to train them using very different algorithms. This increases the chance of making various errors, which improves the ensemble’s accuracy.



# Soft Voting Classifier: If all the classifiers can predict the class probability(predict\_proba method), Scikit can predict the class with the highest probability, averaged over all the individual classifiers.

#, E.g., SVC, by default, doesn’t predict probabilities. We have to set the probability hyperparameter to TRUE, making SVC class use cross-validation to estimate class probabilities, slowing down the training. Using soft voting, 91.2% accuracy is achieved using the same conditions as the above example.

# It often gives higher performance than hard voting because it gives more weight to highly confident votes.

**Bagging and Pasting in Scikit-Learn**

**@** [**https://towardsdatascience.com/wisdom-of-the-crowd-voting-classifier-bagging-pasting-random-forest-and-extra-trees-289ef991e723**](https://towardsdatascience.com/wisdom-of-the-crowd-voting-classifier-bagging-pasting-random-forest-and-extra-trees-289ef991e723)

Bagging ( Bagging Classifier or Bagging Regression ): This trains an ensemble of 500 decision trees classifier(or regressor) trained on 100 training instances randomly sampled from the training set with replacement.

**Bagging Classifier performs soft voting instead of hard voting**.

# Bootstrapping introduces a bit more diversity in the subsets that each predictor is trained on, so bagging ends up with a slightly higher bias than pasting, but this also means that predictors are less correlated, so the ensemble’s variance is reduced.

# For pasting, just set the bootstrap parameter to False

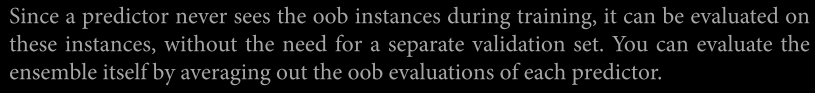
# n\_jobs parameter tells Scikit-Learn the no. of CPU core to use for training and predictions (-1 for using all core available)

# Cross Validation can be used to check which one is working better, Bagging or Pasting, but things become very slow.

**Out of Bag Evaluation**

# Limitations in Bagging: With Bagging, some instances are sampled several times, and some cases are never sampled at all :

Only 63% of the training instances are sampled, and the remaining 37% are not even touched for predictions; these 37% instances are known as Out of Bag (oob) instances.



In Scikit-Learn, you can set oob\_score=True when creating a BaggingClassifier to request an automatic oob evaluation after training.

# oob\_decision\_function\_ variable: This evaluates the class probabilities for each training instance.

**Random Patches and Random Subspaces**

# BaggingClassifier supports **features sampling**: Each predictor will be trained on a random subset of the input features, and two hyperparameters, max\_features, and bootstrap control it.

# Sampling both training instances and features is called the **Random Patches method**.

# Keeping all training instances (i.e., bootstrap=False and max\_sam ples=1.0) but sampling features (i.e., bootstrap\_features=True and/or max\_fea tures smaller than 1.0) is called the **Random Subspaces method.**

# **Sampling features result in even more predictor diversity, trading a bit more bais for lower variance.**

**Random Forests**

**@** [**https://towardsdatascience.com/random-forest-explained-7eae084f3ebe**](https://towardsdatascience.com/random-forest-explained-7eae084f3ebe)

Random Forests is an ensemble of Decision Trees, generally trained via the bagging method (or sometimes pasting), typically with max\_samples set to the size of the training set.

# The Random Forest algorithm introduces extra randomness when growing trees; instead of searching for the best feature when splitting a node, it searches for the best feature among a random subset of features.

# This results in a greater tree diversity, which (once again) trades a higher bias for a lower variance, generally yielding an overall better model.

**Extremely Randomized Trees (Extra-Trees) Ensemble**

**@** [**https://towardsdatascience.com/an-intuitive-explanation-of-random-forest-and-extra-trees-classifiers-8507ac21d54b**](https://towardsdatascience.com/an-intuitive-explanation-of-random-forest-and-extra-trees-classifiers-8507ac21d54b)

In this ensemble, trees are even more random by using a random threshold for each feature rather than searching for the best possible thresholds.

# This trades more bias for a lower variance and is much faster to train.

# We always need to use cross-validation and tuning the hyperparameter using the grid search to check whether RandomForestClassifier will perform better or ExtraForestClassifier. ‘

**Feature Importance (**feature\_importances\_ variable**)**

It measures the relative importance of features by checking how much that feature reduces the impurity on average (across all trees in the forest).

**Boosting:** Boosting (hypothesis boosting) refers to any ensemble that can combine several weak learners into a strong learner.

@ https://towardsdatascience.com/boosting-and-adaboost-clearly-explained-856e21152d3e

**AdaBoost:** In the Adaboost method, the predictor corrects its predecessor by paying more attention to the training instances that the predecessor under fitted.

@ <https://medium.com/analytics-vidhya/ensemble-method-adaboost-712547c7e2c7>

@ <https://www.analyticsvidhya.com/blog/2021/09/adaboost-algorithm-a-complete-guide-for-beginners/>

@ <https://medium.com/machine-learning-101/https-medium-com-savanpatel-chapter-6-adaboost-classifier-b945f330af06>

1) First base classifier (such as Decision Trees) is trained and makes predictions on the training data.

2) The relative weight of misclassified training instances is then increased.

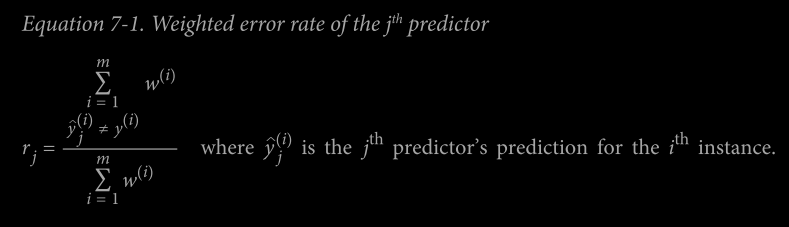
3) Second Classifier is trained using the updated weights, and again, it makes predictions on the training set.

4) Again, the weights are updated, and the process repeats several times.

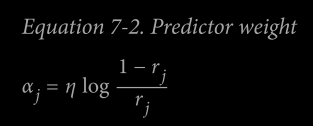
# Drawback of this sequential learning is that it cannot be completely or partially parallelized since each predictor can only be trained after the previous predictor has been trained and evaluated. As a result, it does not scale as well as bagging or pasting.

**Adaboost Algorithm:**

1) Each instance weight w(i) is initially set to 1/m and then weighted error rate calculated.



**2)** Then the predictor’s weight is computed,



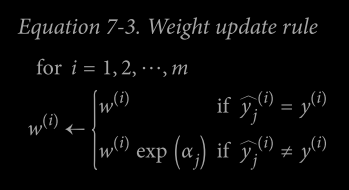
Where η is the learning rate hyperparameter (default value=1)

# The more accurate is the predictor higher its value will be.

# If it is a random guess, then its weight is approx Zero

# If it is less accurate than random guessing, its weight will be negative.

4) Now, the next instance weights are updated (i.e., the misclassified instance’s weight is increased)

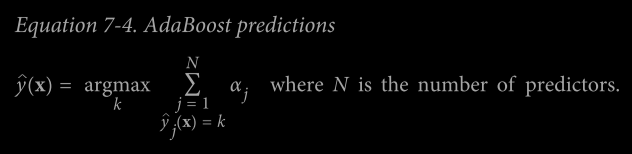


5) New predictor is trained using updated weights, and the whole process is repeated.

6) The algorithm stops when the desired predictors are reached or a perfect predictor is found.

7) Adaboost computes the predictions of all the predictors and weighs them using the predictor weights

The predictor class is the one that receives the majority of weighted votes



# SAMME (Stagewise Additive Modeling using a multiclass exponential loss function), When there are just two classes, SAMME is equivalent to AdaBoost.

# If predictors can compute the probability, then Scikit uses SAMME.R (a variant of SAMME and R stand for real)

Which relies on probability rather than predictions and generally performs better.

# The Decision Stump is a Decision Trees with max\_depth=1 (only two leaf nodes).

# **If the Adaboost ensemble is overfitting the training set, you can reduce the number of estimators or more strongly regularize the base estimator**.

**Gradient Boosting**

**@** [**https://www.analyticsvidhya.com/blog/2021/09/gradient-boosting-algorithm-a-complete-guide-for-beginners/#h2\_5**](https://www.analyticsvidhya.com/blog/2021/09/gradient-boosting-algorithm-a-complete-guide-for-beginners/#h2_5)

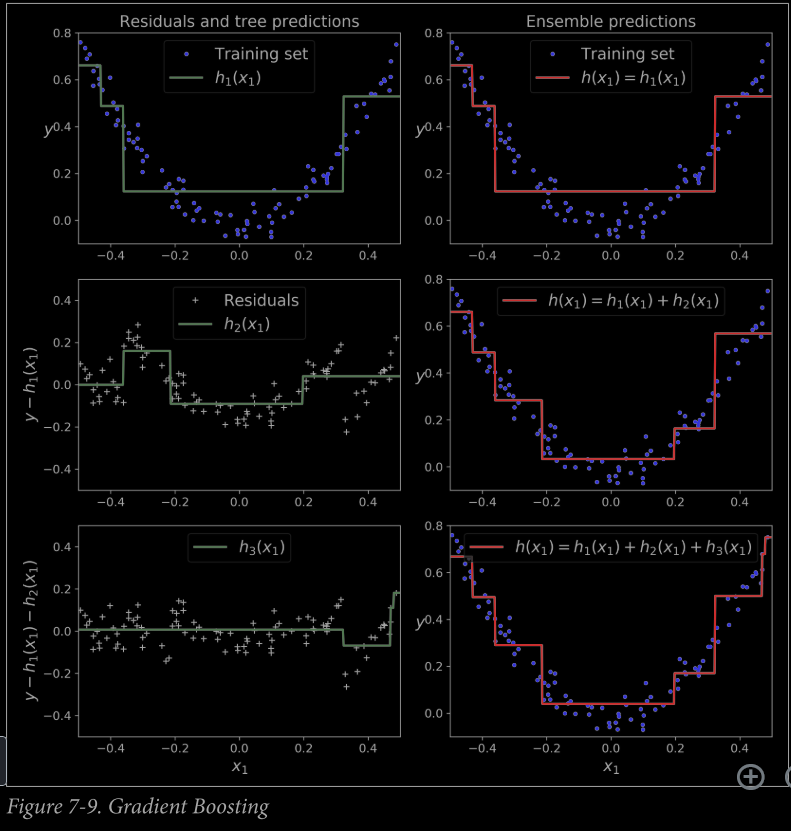
**@** [**https://towardsdatascience.com/all-you-need-to-know-about-gradient-boosting-algorithm-part-1-regression-2520a34a502**](https://towardsdatascience.com/all-you-need-to-know-about-gradient-boosting-algorithm-part-1-regression-2520a34a502)

**@** [**https://towardsdatascience.com/all-you-need-to-know-about-gradient-boosting-algorithm-part-2-classification-d3ed8f56541e**](https://towardsdatascience.com/all-you-need-to-know-about-gradient-boosting-algorithm-part-2-classification-d3ed8f56541e)

Gradient Boosting algorithm also works sequentially, adding predictors to an ensemble, and it tweaks the residual errors of the previous predictor.

# Learning hyperparameter scales the contribution of each tree.

# If it is low, more trees are required to fit the ensemble and vice-versa.



# Finding optimal no. of trees: Early Stopping

“A simple way to implement this is to use the staged\_predict() method: it returns an iterator over the predictions made by the ensemble at each stage of training (with one tree, two trees, etc.”

“It is also possible to implement early stopping by actually stopping training early (instead of training a large number of trees first and then looking back to find the optimal number). You can do so by setting warm\_start=True,”

# GBRT ensemble measures the validation error at each stage of training to find the optimal number of trees and finally trains another GBRT ensemble using the optimal number of trees.

# GradientBoostRegressor class supports a subsample hyperparameter, which specifies the fraction of training instances to be used for training each tree.

**This trades a higher bais for a lower variance and speeds up training considerably. This technique is called Stochastic Gradient Boosting.**

# Other Cost functions can be used in Gradient Boosting, and this is controlled by the loss hyperparameter.

# XGBoost (Extreme Gradient Boosting) is an optimized implementation of Gradient Boosting, which is extremely fast, scalable, and portable.

@ <https://towardsdatascience.com/machine-learning-part-18-boosting-algorithms-gradient-boosting-in-python-ef5ae6965be4>

@ <https://medium.com/@pushkarmandot/how-exactly-xgboost-works-a320d9b8aeef>

@ <https://medium.com/analytics-vidhya/introduction-to-xgboost-algorithm-d2e7fad76b04>

@ <https://medium.com/@gabrieltseng/gradient-boosting-and-xgboost-c306c1bcfaf5>

**Stacking (Stacked Generalization)**

Steps:

1) The training set is split into two subsets.

2) The first step is to train the predictors in the first layer.

3) The first layer predictors are now used to make predictions on the second (held-out) set. (Second Layer)

4) For each instance in the hold-out set, there are three predicted values. The new training set is created using these predictions and is to be used as input features, keeping the target values same.

5) The blender is trained on the new training set.

# Stacking is implemented using brew (available at https://github.com/ viisar/brew)

