# INTRODUCTION

Ensemble modeling is a powerful way to improve the performance of our model. Ensemble methods is a machine learning technique that combines several base models (individual models) together in order to produce one optimal predictive model.

# ERROR IN ENSEMBLE LEARNING

The error emerging from any model can be broken down into three components mathematically.



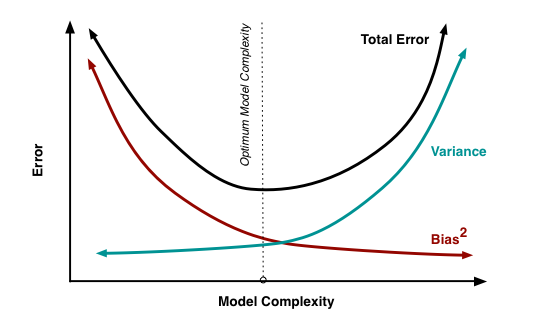
Bias error is useful to quantify how much on an average are the predicted values different from the actual value. A high bias error means we have an under-performing model which keeps on missing important trends. Hence bias is associated with under-fitting.

Variance on the other side quantifies how are the prediction made on same observation different from each other. A high variance model will over-fit on your training population and perform badly on any observation beyond training. Hence variance is associated with overfitting.

Normally, as you increase the complexity of your model, you will see a reduction in error due to lower bias in the model. However, this only happens till a particular point. As you continue to make your model more complex, you end up over-fitting your model and hence your model will start suffering from high variance.

A good model should maintain a balance between these two types of errors. This is known as the trade-off management of bias-variance errors.

Ensemble learning is one way to execute this trade off analysis.



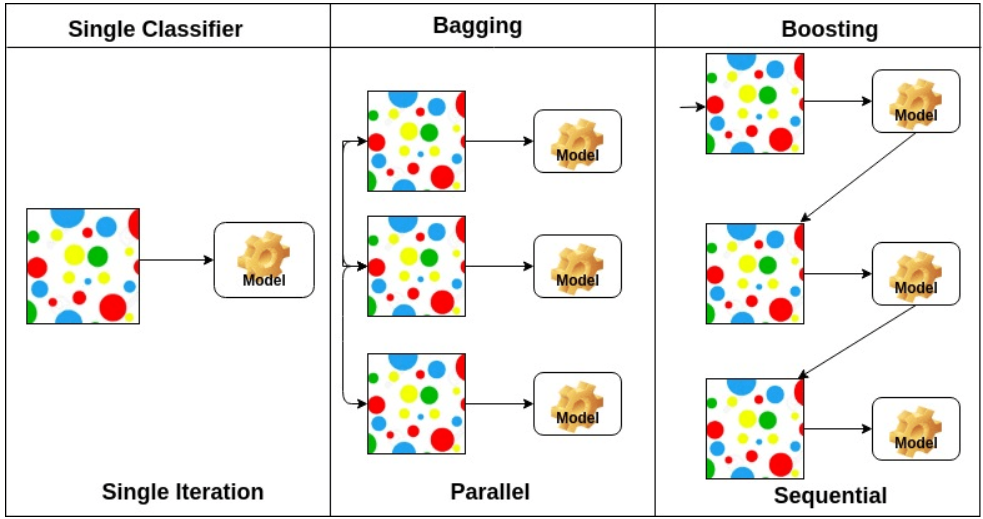
Ensemble methods are meta-algorithms that combine several machine learning techniques into one predictive model in order to decrease variance or overfitting (bagging), bias or under-fitting (boosting), or improve predictions (stacking).

# TYPES OF ENSEMBLE METHODS

Ensemble methods can be divided into two groups:

* Bagging to decrease the model’s variance.
* Boosting to decreasing the model’s bias.
* Stacking to increasing the predictive force of the classifier.

In ensemble learning we use multiple base learners which give different outputs.



Learners can be made different by:

* Using different algorithms.
* Using different training sets.
* Using same algorithm and different hyper parameters in neural networks.

While using decision trees, we can use different criteria for splitting like Entropy or Gini etc. so that different learners split on different features hence producing different outputs.

## BAGGING ALGORITHMS

In this method, we create several estimators (usually fully-grown tress or over fitted estimators) independently and then average their predictions. The combined estimator is usually better than any of the single base estimator because its variance is reduced (usually by 1/k if learners are not correlated, where k is the no of learners).

**B**ootstrap **AGG**regation uses bootstrap sampling to obtain the data subsets for training the base learners. For aggregating the outputs of base learners, bagging uses voting for classification and averaging for regression.

It is also known as averaging or parallel ensemble.

Example: Bagged Decision Trees, Random Forest, Forests of Randomized Trees, Extra Tree.

## BOOSTING ALGORITHMS

Boosting allows you to build a strong learning by combining the outputs of a set of weak learners (decision stumps). How cool is that?

Once created, the models make predictions which may be weighted by their demonstrated accuracy and the results are combined to create a final output prediction.

It is also known as Sequential Ensemble.

Example: AdaBoost, Gradient Tree Boosting.

## BAGGING VS BOOSTING

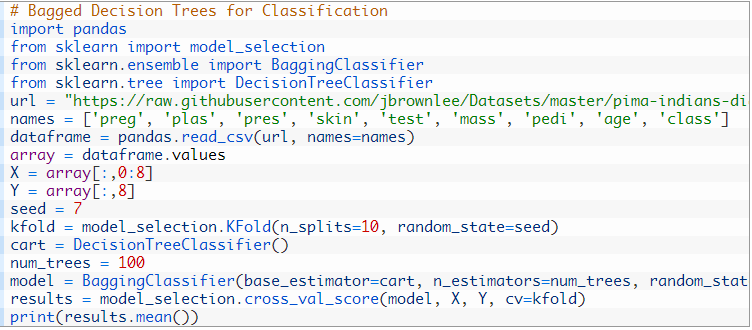
Below are few main differences:

* Bagging methods work best with strong and complex models (e.g., fully developed decision trees). In contrast with boosting methods which usually work best with weak models (e.g., shallow decision trees).
* Use bagging if you would like to decrease model variance error. Use boosting if you would like to decrease model bias error.
* While using bagging ensemble method we usually use unstable algorithms so that each learner has high variance. Example are Decision tree and neural network. In boosting a weak learner is turned into a strong learner.
* Boosting is an iterative process where as bagging is parallel process. In boosting iterative process is repeated K times and weight assigned to each instance is changed based on the output where as in Bagging we construct K learners which learners from training data simultaneously.

## BAGGED DECISION TREES

Bagging performs best with algorithms that have high variance. A popular example are decision trees, often constructed without pruning.

In the example below see an example of using the BaggingClassifier with the Classification and Regression Trees algorithm (DecisionTreeClassifier). A total of 100 trees are created.



## RANDOM FOREST

Random forest is an extension of bagged decision trees.

Samples of the training dataset are taken with replacement, but the trees are constructed in a way that reduces the correlation between individual classifiers. Specifically, rather than greedily choosing the best split point in the construction of the tree, only random subset of features are considered for each split.

You can construct a Random Forest model for classification using the RandomForestClassifier class.

The example below provides an example of Random Forest for classification with 100 trees and split points chosen from a random selection of 3 features.



## ADABOOST

An AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.

An AdaBoost regressor is a meta-estimator that begins by fitting a regressor on the original dataset and then fits additional copies of the regressor on the same dataset but where the weights of instances are adjusted according to the error of the current prediction. As such, subsequent regressors focus more on difficult cases.

The example below demonstrates the construction of 30 decision trees in sequence using the AdaBoost algorithm.

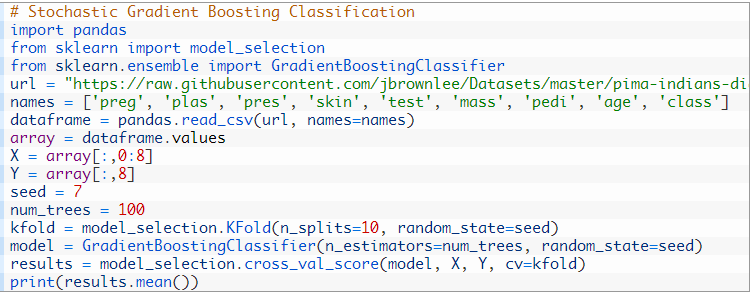


## GRADIENT BOOSTING

Stochastic Gradient Boosting (also called Gradient Boosting Machines) are one of the most sophisticated ensemble techniques. It is also a technique that is proving to be perhaps of the best techniques available for improving performance via ensembles.

You can construct a Gradient Boosting model for classification using the GradientBoostingClassifier class.

The example below demonstrates Stochastic Gradient Boosting for classification with 100 trees.



## VOTING ENSEMBLE

Voting is one of the simplest ways of combining the predictions from multiple machine learning algorithms.

It works by first creating two or more standalone models from your training dataset. A Voting Classifier can then be used to wrap your models and average the predictions of the sub-models when asked to make predictions for new data.

The predictions of the sub-models can be weighted, but specifying the weights for classifiers manually or even heuristically is difficult. More advanced methods can learn how to best weight the predictions from sub models, but this is called stacking (stacked aggregation) and is currently not provided in scikit-learn.

You can create a voting ensemble model for classification using the VotingClassifier class.

The code below provides an example of combining the predictions of logistic regression, classification and regression trees and support vector machines together for a classification problem.

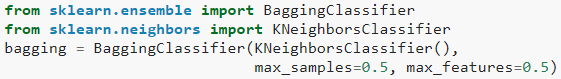


# ENSEMBLE IN SKLEARN

In scikit-learn, bagging methods are offered as a unified BaggingClassifier meta-estimator (resp. BaggingRegressor), taking as input a user-specified base estimator along with parameters specifying the strategy to draw random subsets.

In particular, max\_samples and max\_features control the size of the subsets (in terms of samples and features), while bootstrap and bootstrap\_features control whether samples and features are drawn with or without replacement.

When using a subset of the available samples the generalization accuracy can be estimated with the out-of-bag samples by setting oob\_score=True.



# ENSEMBLE LEARNING APPLICATIONS

Some of the applications of ensemble classifiers include:

## CHANGE DETECTION

Change detection is an image analysis problem, consisting of the identification of places where the land cover has changed over time. Change detection is widely used in fields such as urban growth, forest and vegetation dynamics, land use and disaster monitoring.

## MALWARE DETECTION

Classification of malware codes such as computer viruses, computer worms, trojans, ransomware and spywares with the usage of machine learning techniques. Ensemble learning systems have shown a proper efficacy in this area

## INTRUSION DETECTION

An intrusion detection system monitors computer network or computer systems to identify intruder codes like an anomaly detection process. Ensemble learning successfully aids such monitoring systems to reduce their total error

## FACE RECOGNITION

Face recognition, which recently has become one of the most popular research areas of pattern recognition.

## FRAUD DETECTION

Fraud detection deals with the identification of bank fraud, such as money laundering, credit card fraud and telecommunication fraud, which have vast domains of research and applications of machine learning. Because ensemble learning improves the robustness of the normal behavior modelling, it has been proposed as an efficient technique to detect such fraudulent cases and activities in banking and credit card systems.

## FINANCIAL DECISION-MAKING

The accuracy of prediction of business failure is a very crucial issue in financial decision-making. Therefore, different ensemble classifiers are proposed to predict financial crises and financial distress

# REFERENCE

<https://scikit-learn.org/stable/modules/ensemble.html>

<https://www.wikiwand.com/en/Ensemble_learning>

<http://rasbt.github.io/mlxtend/user_guide/regressor/StackingRegressor/>

<http://rasbt.github.io/mlxtend/user_guide/classifier/StackingClassifier/>