Ensembling Techniques

Suppose you decide you want to go to 'Rome' for your vacation. However, you are not sure if it is a good place to visit during Summer. So you ask a bunch of people

1. A travel guide, whose opinions about travel destinations are 70% times similar to yours.
2. A YouTube trip vlogger, who is 80% times similar to your opinions about a destination.
3. A close friend of yours, who is 60% of times similar to your opinions.

Though individually each one would may have some sort of bias(For eg: Your friend said no cause of his aversion towards forts in Rome)but when taken together, the probability of them being wrong simultaneously is equal to-

- $P = (1 - 0.7) \times (1 - 0.8) \times (1 - 0.6)$
- $P = 0.024$

Which means that there is 97.6% chance that their opinion will be good (given their opinions are independent from each other).

Ensemble works on similar principles.

Ensemble methods is a machine learning technique that combines several base models in order to produce one optimal predictive model.it helps in better predictive performance compared to single model. The Basic idea is to learn a set of classifiers (experts) and to allow them to vote.

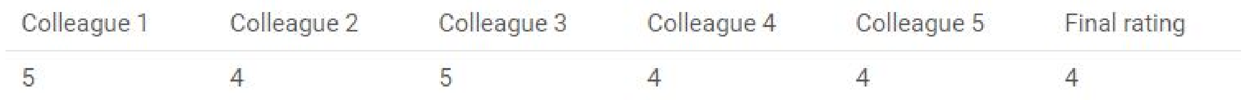
Here are some Simple Ensemble Techniques, a few simple but powerful techniques, namely:

1. Max Voting
2. Averaging
3. Weighted Averaging

Max Voting

The max voting method is generally used for classification problems. In this technique, multiple models are used to make predictions for each data point. The predictions by each model are considered as a ‘vote’. The predictions which we get from the majority of the models are used as the final prediction.

For example, when you asked 5 of your friends to rate the movie (out of 5); assuming that three of them rated it as 4 while two of them gave it a 5, we can say that the majority gave a rating of 4, so the final rating will be taken as 4. You can consider this as taking the mode of all the predictions.
The result of max voting would be something like this:



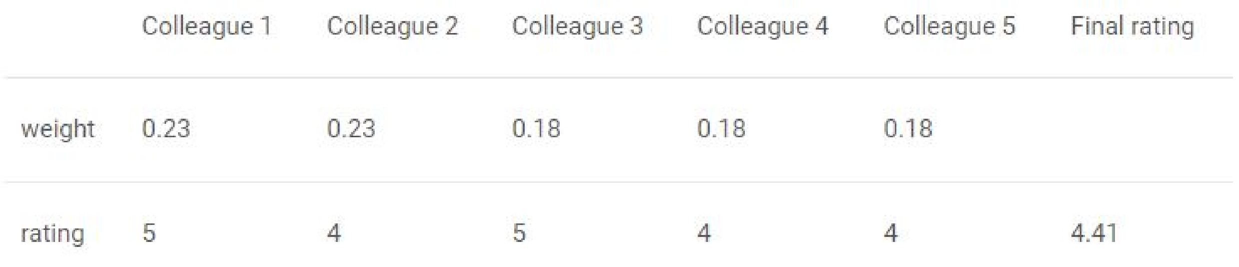
Colleague 1	Colleague 2	Colleague 3	Colleague 4	Colleague 5	Final rating
5	4	5	4	4	4

Averaging

Similar to the max voting technique, multiple predictions are made for each data point in averaging. In this method, we take an average of predictions from all the models and use it to make the final prediction. Averaging can be used for making predictions in regression problems or while calculating probabilities for classification problems.
For example, in the below case, the averaging method would take the average of all the values.
i.e. (5+4+5+4+4)/5 = 4.4

Weighted Average

This is an extension of the averaging method. All models are assigned different weights defining the importance of each model for prediction. For instance, if two of your friends are critics, while others have no prior experience in this field, then the answers by these two friends are given more importance as compared to the other people.
The result is calculated as [(5*0.23) + (4*0.23) + (5*0.18) + (4*0.18) + (4*0.18)] = 4.41.

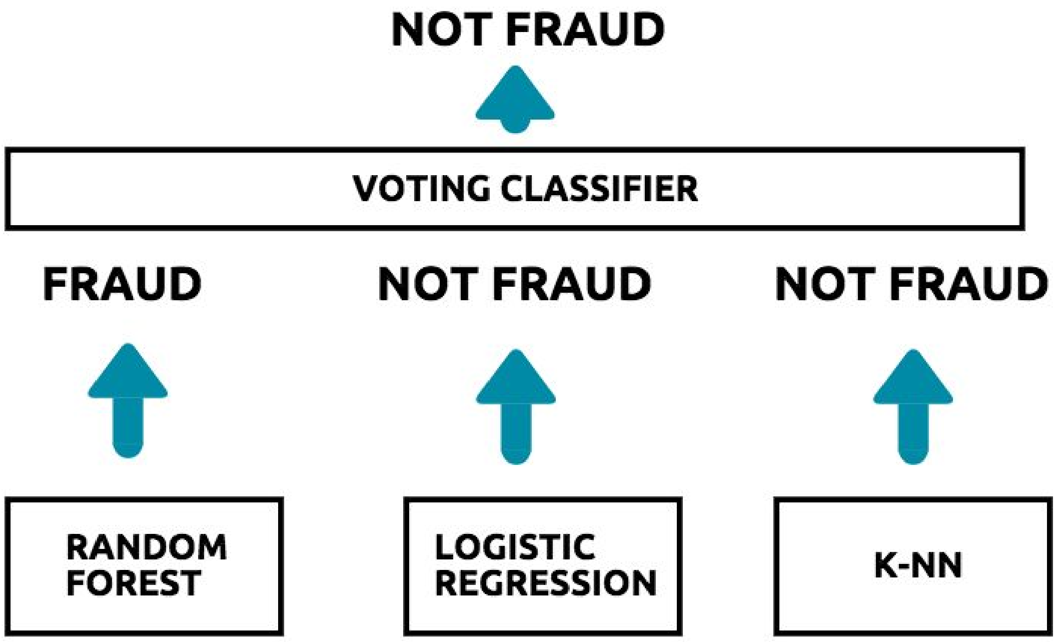


	Colleague 1	Colleague 2	Colleague 3	Colleague 4	Colleague 5	Final rating
weight	0.23	0.23	0.18	0.18	0.18	
rating	5	4	5	4	4	4.41

Advance techniques:

Naive Aggregation:

In Naive aggregation, we build different models and the prediction which we get after averaging (regression) or by voting (classification) are considered as the final prediction.



The most intuitive way to combine models is averaging out their individual prediction.

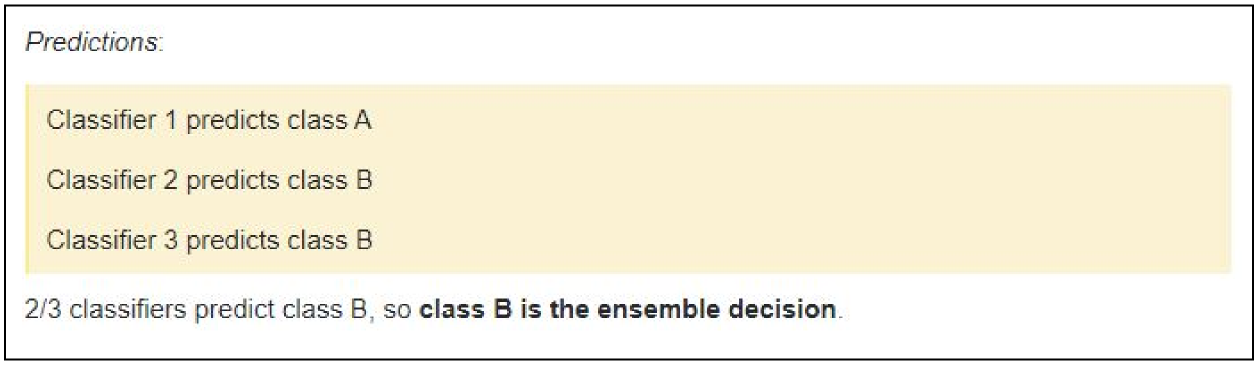
As we talk about vote, voting can be done based on true facts as well as false facts which in turn, turnout to be a strong or weak vote. Here too, we can classify votes based on weak learners and strong learners, which produce the final classifier.

Weak Learner: Given a labeled dataset, a Weak Learner produces a classifier which is at least a little more accurate than random classification.

Strong Learner: We call a machine learning model a Strong Learner which, given a labeled dataset, can produce results arbitrarily well-correlated with the true classification.

There are two ways in which this voting can be done by using **hard voting** or **soft voting.**

In hard voting, the voting classifier takes the majority of its base learners' predictions



Predictions:

Classifier 1 predicts class A

Classifier 2 predicts class B

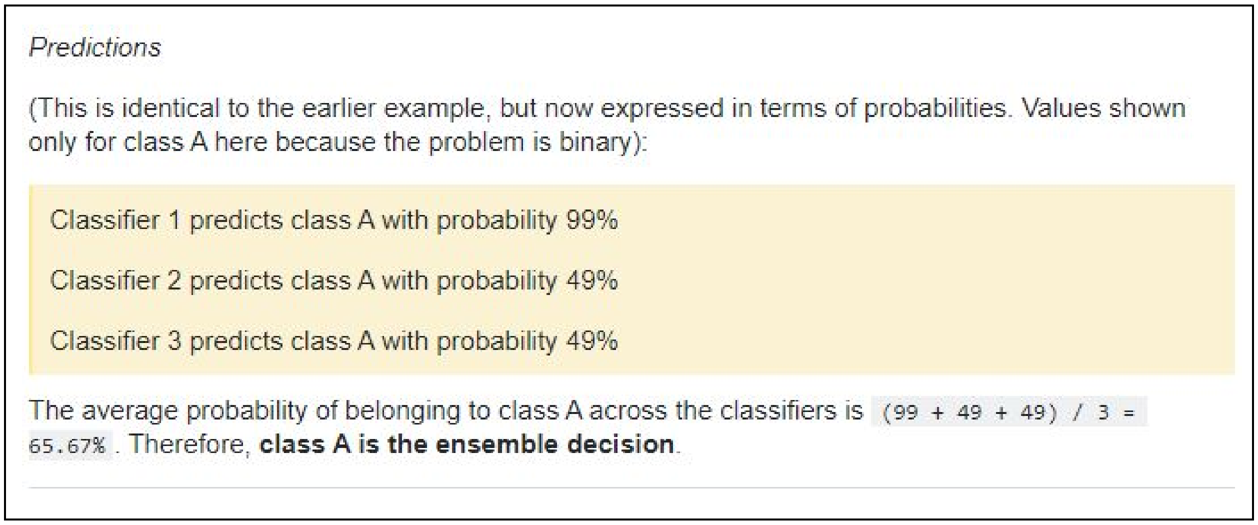
Classifier 3 predicts class B

2/3 classifiers predict class B, so **class B is the ensemble decision**.

In **soft voting**, the voting classifier takes into account the probability values by its base learners

Hard Voting :- consider each and every classifiers from base learner's predictions

Soft Voting :- calculate probability of classifiers from base learner



Predictions

(This is identical to the earlier example, but now expressed in terms of probabilities. Values shown only for class A here because the problem is binary):

Classifier 1 predicts class A with probability 99%

Classifier 2 predicts class A with probability 49%

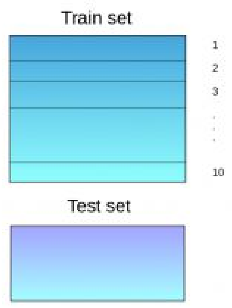
Classifier 3 predicts class A with probability 49%

The average probability of belonging to class A across the classifiers is `(99 + 49 + 49) / 3 = 65.67%`. Therefore, **class A is the ensemble decision**.

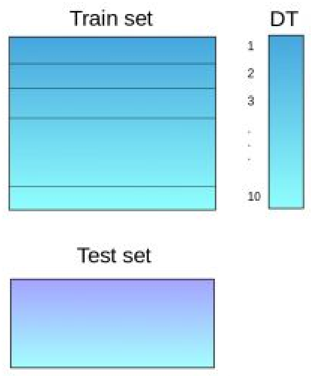
Stacking

Stacking is an ensemble learning technique that uses predictions from multiple models (for example decision trees, knn or svm) to build a new model. This model is used for making predictions on the test set. Below is a step-wise explanation for a simple stacked ensemble:

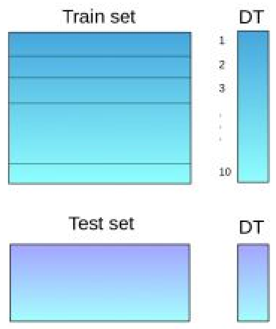
1. The train set is split into 10 parts.



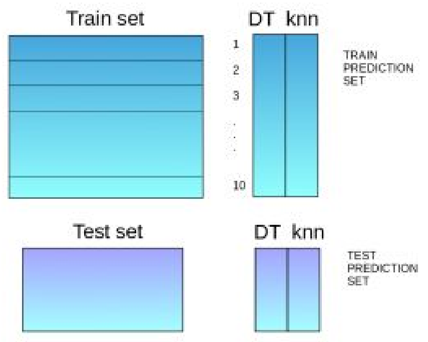
2. A base model (suppose a decision tree) is fitted on 9 parts and predictions are made for the 10th part. This is done for each part of the train set.



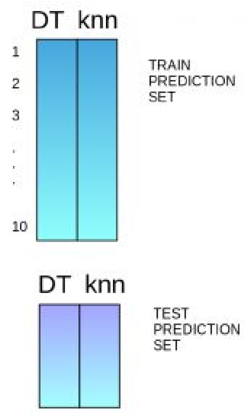
3. The base model (in this case, decision tree) is then fitted on the whole train dataset.
4. Using this model, predictions are made on the test set.



5. Steps 2 to 4 are repeated for another base model (say knn) resulting in another set of predictions for the train set and test set.



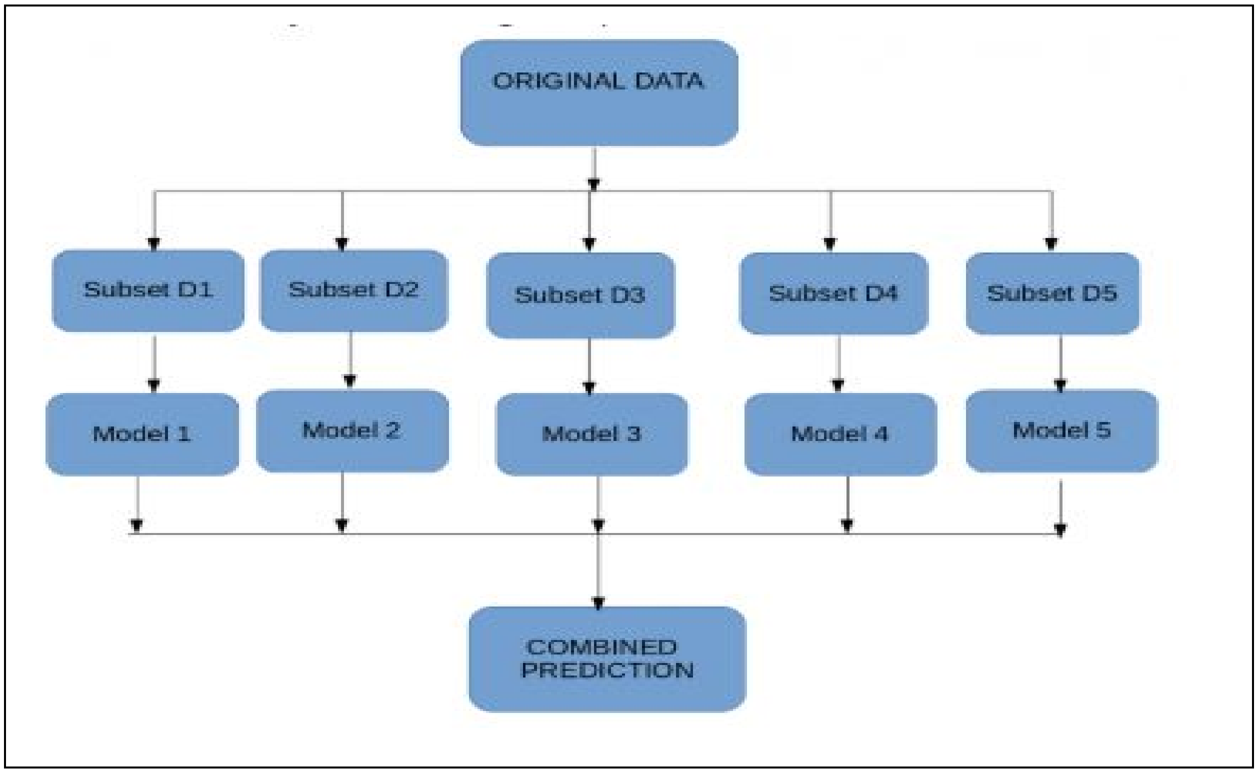
6. The predictions from the train set are used as features to build a new model.



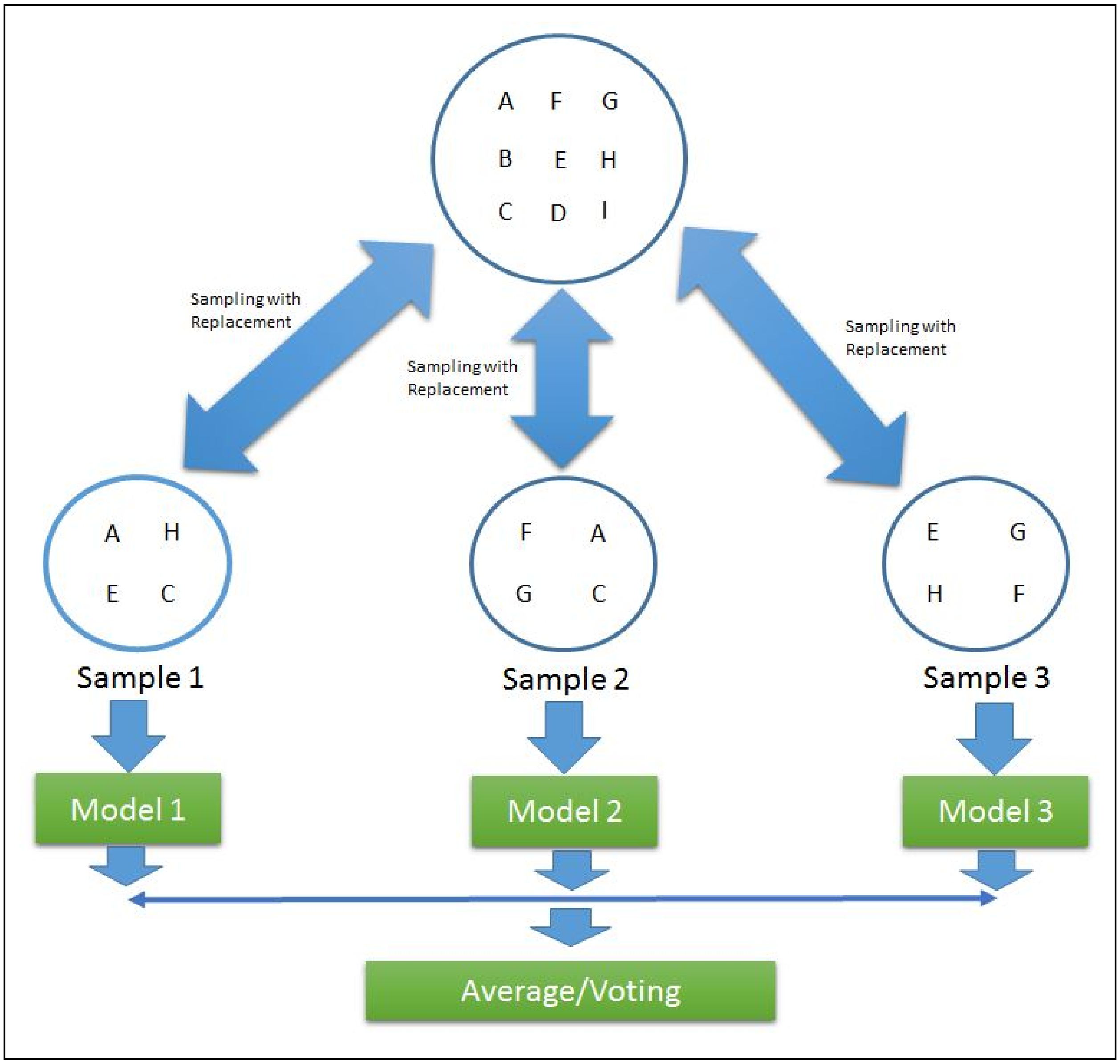
7. This model is used to make final predictions on the test prediction set.

Bootstrap Aggregation(Bagging)

Bagging stands for bootstrap aggregation.
Building multiple models (typically of the same type) from different subsamples of the training dataset.



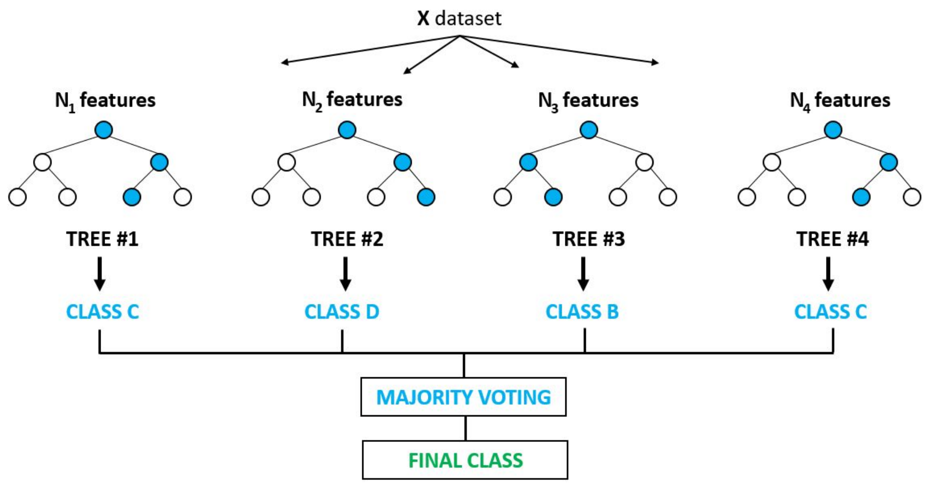
Bootstrapping is a sampling technique in which we create subsets of observations from the original dataset, with replacement.



Stacking

Stacking is a special case of Bagging, where each model is trained with a different subset without replacement.

The Random Forests Algorithm



Let's understand the algorithm in layman's terms. Suppose you want to go on a trip and you would like to travel to a place which you will enjoy.

So what do you do to find a place that you will like? You can search online, read reviews on travel blogs and portals, or you can also ask your friends.

Let's suppose you have decided to ask your friends, and talked with them about their past travel experience to various places. You will get some recommendations from every friend. Now you have to make a list of those recommended places. Then, you ask them to vote (or select one best place for the trip) from the list of recommended places you made. The place with the highest number of votes will be your final choice for the trip.

In the above decision process, there are two parts. First, asking your friends about their individual travel experience and getting one recommendation out of multiple places they have visited. This part is like using the decision tree algorithm. Here, each friend makes a selection of the places he or she has visited so far.

The second part, after collecting all the recommendations, is the voting procedure for selecting the best place in the list of recommendations. This whole process of getting recommendations from friends and voting on them to find the best place is known as the random forests algorithm.

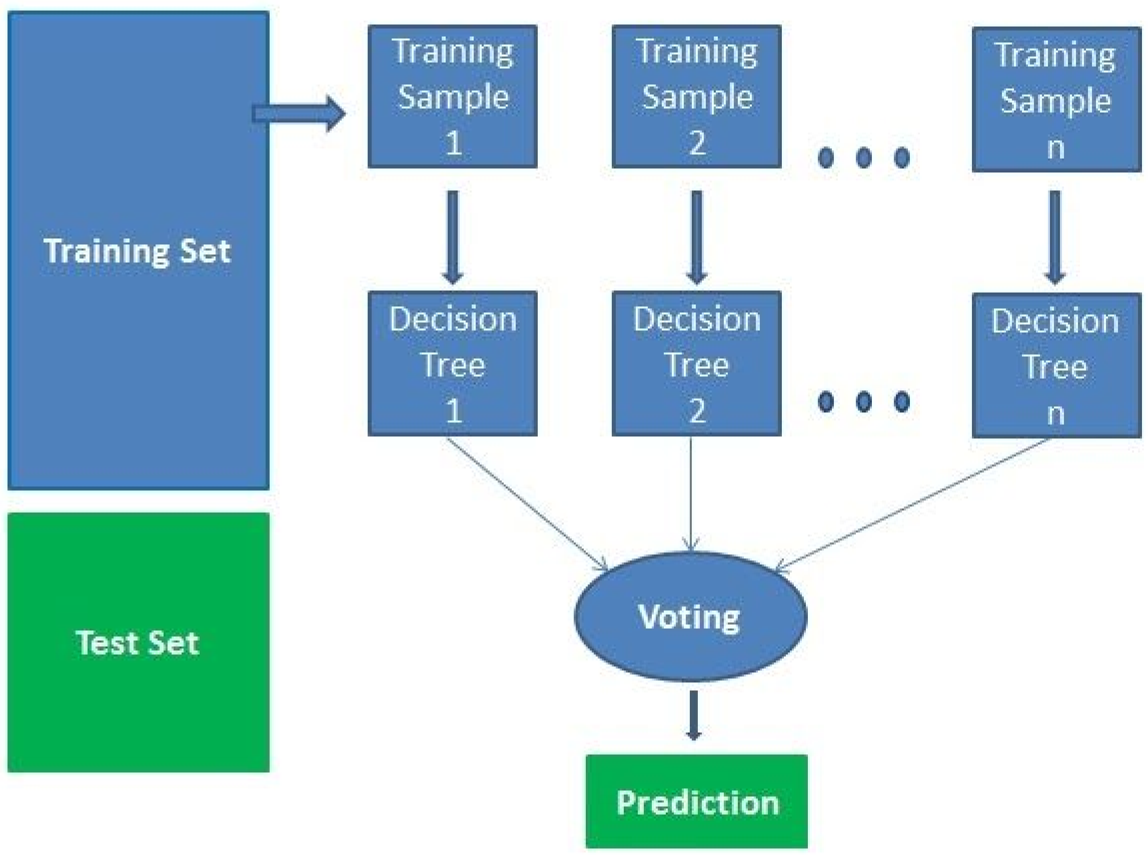
It technically is an ensemble method (based on the divide-and-conquer approach) of decision trees generated on a randomly split dataset. This collection of decision tree classifiers is also known as the forest. The individual decision trees are generated using an attribute selection indicator such as information gain, gain ratio, and Gini index for each attribute. Each tree depends on an independent random sample. In a classification problem, each tree votes and the most popular class is chosen as the final result. In the case of regression, the average of all the tree outputs is considered as the final result. It is simpler and more powerful compared to the other non-linear classification algorithms.

How does the algorithm work?

It works in four steps:

1. Select random samples from a given dataset.
2. Construct a decision tree for each sample and get a prediction result from each decision tree.
3. Perform a vote for each predicted result.

4. Select the prediction result with the most votes as the final prediction.



Advantages:

- Random forests are considered as a highly accurate and robust method because of the number of decision trees participating in the process.
- It does not suffer from the overfitting problem. The main reason is that it takes the average of all the predictions, which cancels out the biases.
- The algorithm can be used in both classification and regression problems.

- Random forests can also handle missing values. There are two ways to handle these: using median values to replace continuous variables, and computing the proximity-weighted average of missing values.
- You can get the relative feature importance, which helps in selecting the most contributing features for the classifier.

Disadvantages:

- Random forests are slow in generating predictions because they have multiple decision trees. Whenever it makes a prediction, all the trees in the forest have to make a prediction for the same given input and then perform voting on it. This whole process is time-consuming.
- The model is difficult to interpret compared to a decision tree, where you can easily make a decision by following the path in the tree.

Finding important features

Random forests also offer a good feature selection indicator. Scikit-learn provides an extra variable with the model, which shows the relative importance or contribution of each feature in the prediction. It automatically computes the relevance score of each feature in the training phase. Then it scales the relevance down so that the sum of all scores is 1.

This score will help you choose the most important features and drop the least important ones for model building.

Random forest uses gini importance or mean decrease in impurity (MDI) to calculate the importance of each feature. Gini importance is also known as the total decrease in node impurity. This is how much the model fit or accuracy decreases when you drop a variable. The larger the decrease, the more significant the variable is. Here, the mean decrease is a significant parameter for variable selection. The Gini index can describe the overall explanatory power of the variables.

Random Forests vs Decision Trees

- Random forests are a set of multiple decision trees.
- Deep decision trees may suffer from overfitting, but random forests prevent overfitting by creating trees on random subsets.
- Decision trees are computationally faster.
- Random forests are difficult to interpret, while a decision tree is easily interpretable and can be converted to rules.