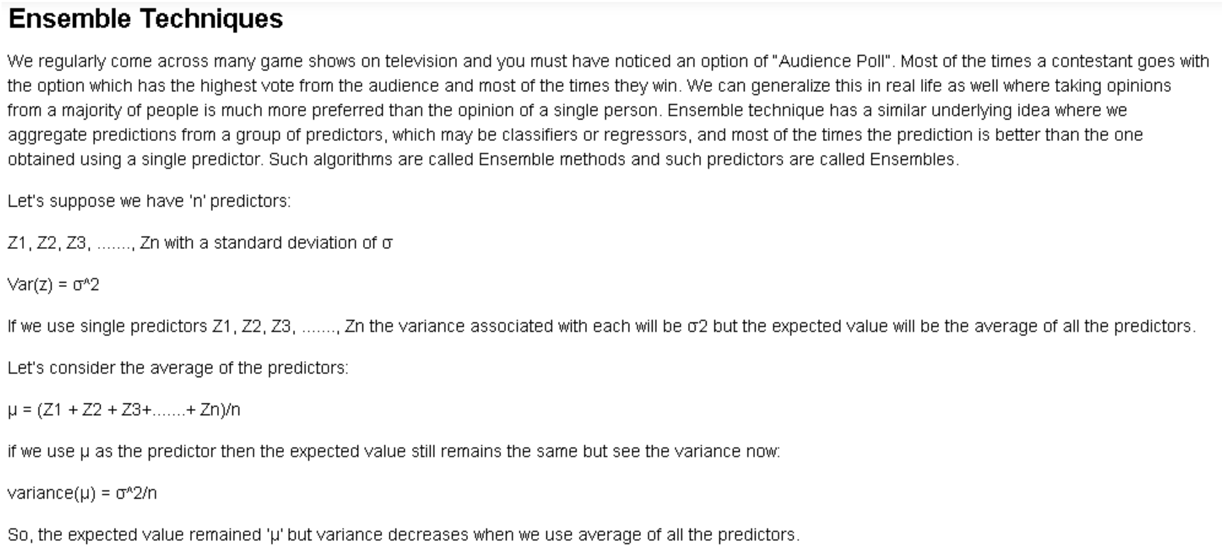
Ensemble and Random Forest

1) What do you understand by Ensemble technique?

Ensemble methods are meta-algorithms that combine several machine learning techniques into one predictive model in order to decrease variance (bagging), bias (boosting), or improve predictions (stacking). Ensemble methods can be divided into two groups: sequential ensemble methods where the base learners are generated sequentially (e.g. AdaBoost). The basic motivation of sequential methods is to exploit the dependence between the base learners. The overall performance can be boosted by weighing previously mislabeled examples with higher weight. parallel ensemble methods where the base learners are generated in parallel (e.g. Random Forest). The basic motivation of parallel methods is to exploit independence between the base learners since the error can be reduced dramatically by averaging. Most ensemble methods use a single base learning algorithm to produce homogeneous base learners, i.e. learners of the same type, leading to homogeneous ensembles. There are also some methods that use heterogeneous learners, i.e. learners of different types, leading to heterogeneous ensembles. In order for ensemble methods to be more accurate than any of its individual members, the base learners have to be as accurate as possible and as diverse as possible.

2) Explain the idea behind ensemble techniques.



Ensemble Techniques

We regularly come across many game shows on television and you must have noticed an option of "Audience Poll". Most of the times a contestant goes with the option which has the highest vote from the audience and most of the times they win. We can generalize this in real life as well where taking opinions from a majority of people is much more preferred than the opinion of a single person. Ensemble technique has a similar underlying idea where we aggregate predictions from a group of predictors, which may be classifiers or regressors, and most of the times the prediction is better than the one obtained using a single predictor. Such algorithms are called Ensemble methods and such predictors are called Ensembles.

Let's suppose we have 'n' predictors:

Z1, Z2, Z3, ..., Zn with a standard deviation of σ

Var(z) = σ^2

If we use single predictors Z1, Z2, Z3, ..., Zn the variance associated with each will be σ2 but the expected value will be the average of all the predictors.

Let's consider the average of the predictors:

µ = (Z1 + Z2 + Z3+.......+ Zn)/n

if we use µ as the predictor then the expected value still remains the same but see the variance now:

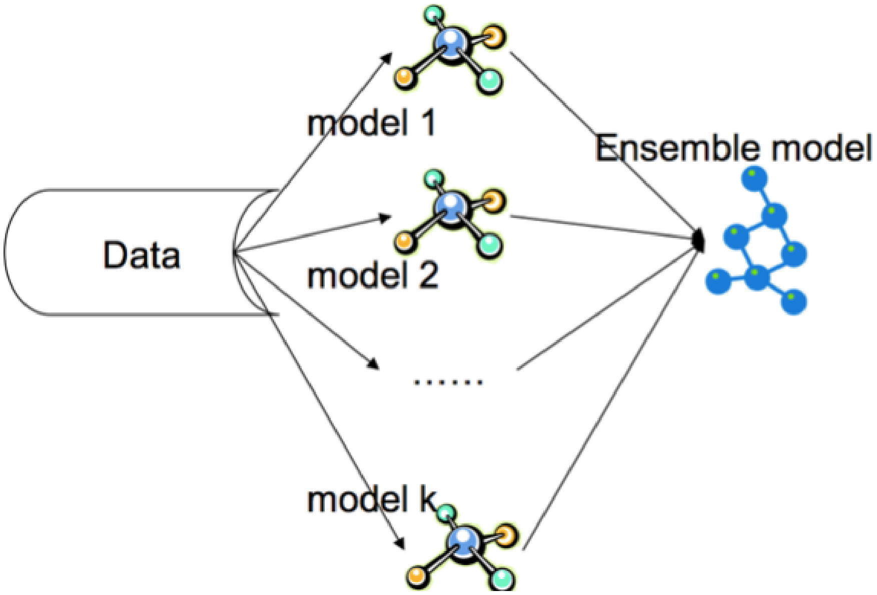
variance(µ) = σ^2/n

So, the expected value remained 'µ' but variance decreases when we use average of all the predictors.

This is why taking mean is preferred over using single predictors.

Ensemble methods take multiple small models and combine their predictions to obtain a more powerful predictive power.

There are few very popular Ensemble techniques which we will talk about in detail such as Bagging, Boosting, stacking etc.



3) What is Bootstrapping? How is sampling done in bootstrapping?

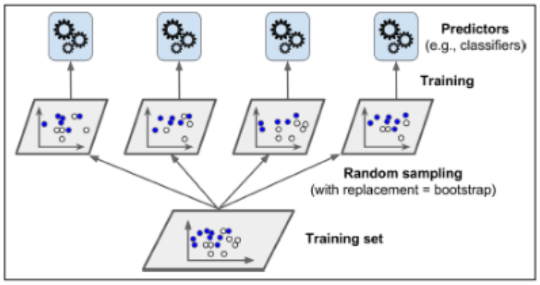
Bagging (Bootstrap Aggregation)

In real life scenarios we don't have multiple different training sets on which we can train our model separately and at the end combine their result. Here, bootstrapping comes into picture. Bootstrapping is a technique of sampling different sets of data from a given training set by using replacement. After bootstrapping the training dataset, we train model on all the different sets and aggregate the result. This technique is known as Bootstrap Aggregation or Bagging.

Let's see definition of bagging:

Bagging is the type of ensemble technique in which a single training algorithm is used on different subsets of the training data where the subset sampling is done with replacement (bootstrap). Once the algorithm is trained on all the subsets, then bagging makes the prediction by aggregating all the predictions made by the algorithm on different subsets. In case of regression, bagging prediction is simply the mean of all the predictions and in the case of classifier, bagging prediction is the most frequent prediction (majority vote) among all the predictions.

Bagging is also known as parallel model since we run all models parallelly and combine there results at the end.



- **Advantages of a Bagging Model**

1) Bagging significantly decreases the variance without increasing bias.

2) Bagging methods work so well because of diversity in the training data since the sampling is done by bootstraping.

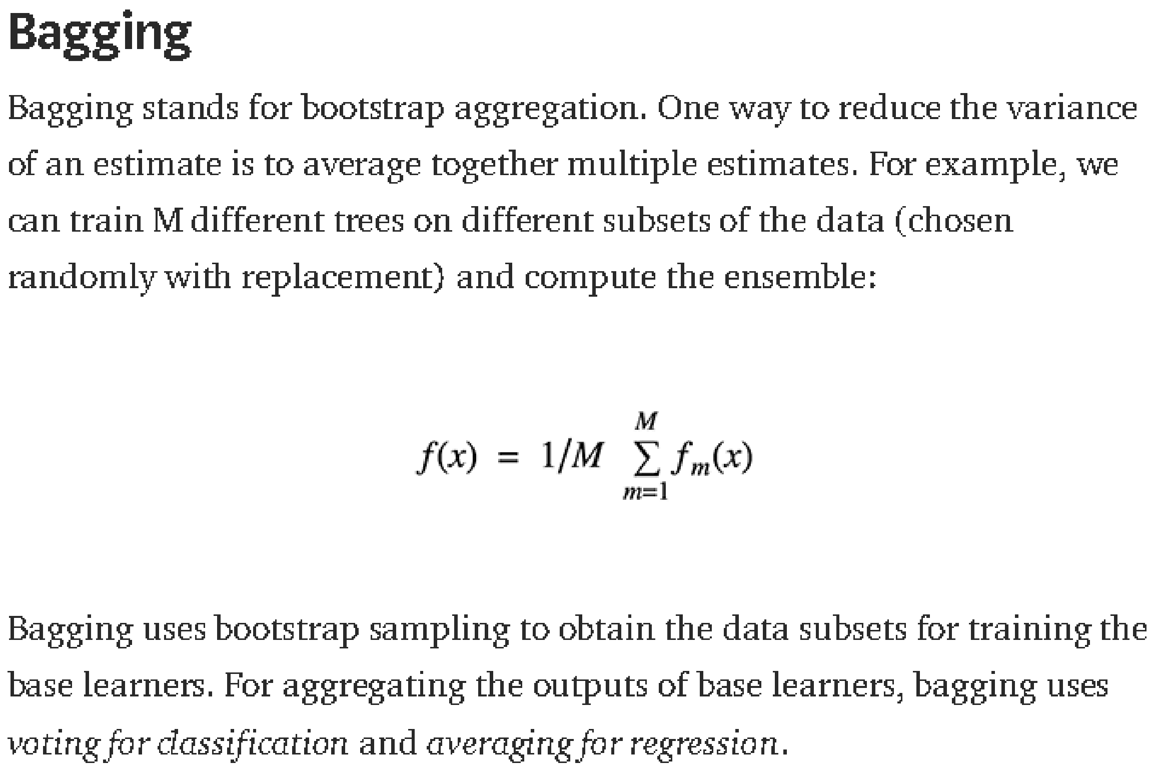
3) Also, if the training set is very huge, it can save computional time by training model on relatively smaller data set and still can increase the accuracy of the model.

4) Works well with small datasets as well.

- **Disadvantage of a Bagging Model**

The main disadvantage of Bagging is that it improves the accuracy of the model on the expense of interpretability i.e. if a single tree was being used as the base model, then it would have a more attarctive and easily interpretable diagram, but with use of bagging this interpretability gets lost.

4) What is bagging?



Bagging

Bagging stands for bootstrap aggregation. One way to reduce the variance of an estimate is to average together multiple estimates. For example, we can train M different trees on different subsets of the data (chosen randomly with replacement) and compute the ensemble:

$$f(x) = 1/M \sum_{m=1}^{M} f_m(x)$$

Bagging 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*.

5) How prediction is made in Bagging?

Voting and Averaging Based Ensemble Methods

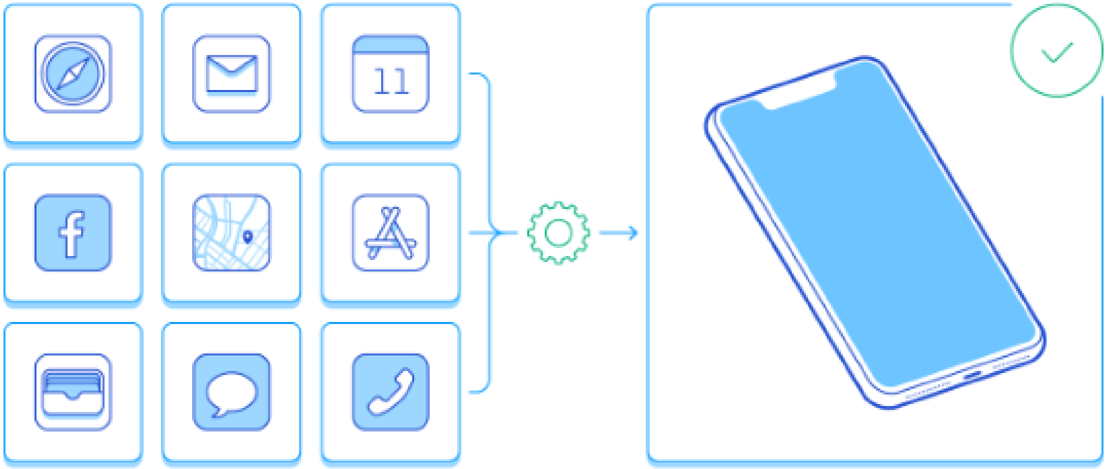
Voting and averaging are two of the easiest ensemble methods. They are both easy to understand and implement. Voting is used for classification and averaging is used for regression.

In both methods, the first step is to create multiple classification/regression models using some training dataset. Each base model can be created using different splits of the same training dataset and same algorithm, or using the same dataset with different algorithms, or any other method. The following Python-esque pseudocode shows the use of same training dataset with different algorithms.

Majority Voting Every model makes a prediction (votes) for each test instance and the final output prediction is the one that receives more than half of the votes. If none of the predictions get more than half of the votes, we may say that the ensemble method could not make a stable prediction for this instance. Although this is a widely used technique, you may try the most voted prediction (even if that is less than half of the votes) as the final prediction. In some articles, you may see this method being called “plurality voting

Weighted Voting Unlike majority voting, where each model has the same rights, we can increase the importance of one or more models. In weighted voting you count the prediction of the better models multiple times. Finding a reasonable set of weights is up to you.

Simple Averaging In simple averaging method, for every instance of test dataset, the average predictions are calculated. This method often reduces overfit and creates a smoother regression model. The following pseudocode code shows this simple averaging method:



6) How Ensemble technique solves the high variance issue with Decision trees?

The learning algorithms which output only a single hypothesis tends to suffer from basically three issues. These issues are the statistical problem, the computational problem and the representation problem which can be partly overcome by applying ensemble methods.

The learning algorithm which suffers from the statistical problem is said to have high variance. The algorithm which exhibits the computational problem is sometimes described as having computational variance and the learning algorithm which suffers from the representational problem is said to have a high bias. These three fundamental issues can be said as the three important ways in which existing learning algorithms fail. The ensemble methods promise of reducing both the bias and the variance of these three shortcomings of the standard learning algorithm.

7) What is pasting? How is it different from bagging?

Pasting is an ensemble technique similar to bagging with the only difference being that there is no replacement done while sampling the training dataset. This causes less diversity in the sampled datasets and data ends up being correlated. That's why bagging is more preffered than pasting in real scenarios.

Pasting and Bagging are very similar, the main difference being that Bagging samples with replacement (which is called "bootstrapping") while Pasting samples without replacement

Bagging is to use the same training for every predictor, but to train them on different random subsets of the training set. When sampling is performed with replacement, this method is called bagging (short for bootstrap aggregating). When sampling is performed without replacement, it is called pasting. In other words, both approaches are similar. In both cases you are sampling the training data to build multiple instances of a classifier. In both cases a training item could be sampled and used to train multiple instances in the collection of classifiers that is produced. In bagging, it is possible for a training sample to be sampled multiple times in the training for the same predictor. This type of bootstrap aggregation is a type of data enhancement, and it is used in other contexts as well in ML to artificially increase the size of the training set.

Computationally bagging and pasting are very attractive because in theory and in practice all of the classifiers can be trained in parallel. Thus if you have a large number of CPU cores, or even a distributed memory computing cluster, you can independently train the individual classifiers all in parallel

8) What is Out Of Bag evaluation?

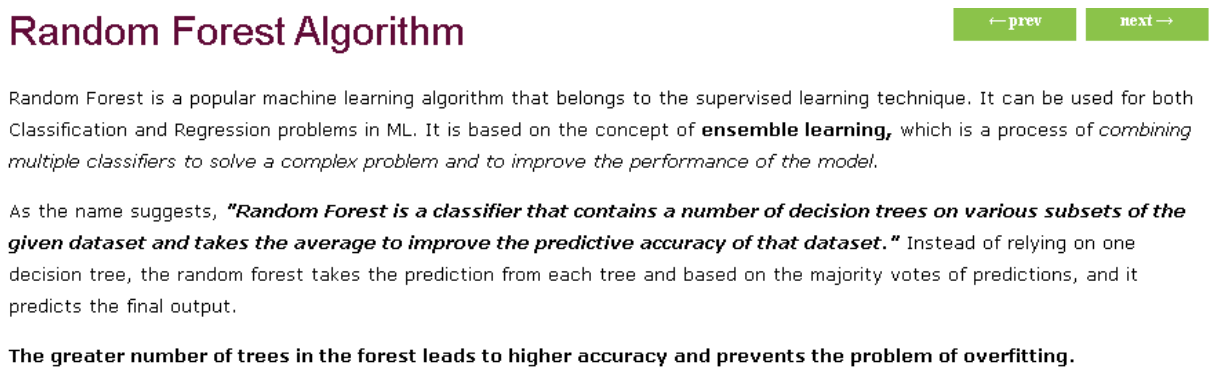
In bagging, when different samples are collected, no sample contains all the data but a fraction of the original dataset. There might be some data which are never sampled at all. The remaining data which are not sampled are called out of bag instances. Since the model never trains over these data, they can be used for evaluating the accuracy of the model by using these data for predicition. We do not need validation set or cross validation and can use out of bag instances for that purpose.

Out of bag (OOB) score is a way of validating the Random forest model. Below is a simple intuition of how is it calculated followed by a description of how it is different from validation score and where it is advantageous.

only a subset of DTs is used for determining the OOB score. This leads to reducing the overall aggregation effect in bagging. Thus in general, validation on a full ensemble of DTs is better than a subset of DT for estimating the score. However, occasionally the dataset is not big enough and hence set aside a part of it for validation is unaffordable. Consequently, in cases where we do not have a large dataset and want to consume it all as the training dataset, the OOB score provides a good trade-off. Nonetheless, it should be noted that validation score and OOB score are unalike, computed in a different manner and should not be thus compared.

Type *Markdown* and LaTeX: α^2

9) How does a Random Forest model works?



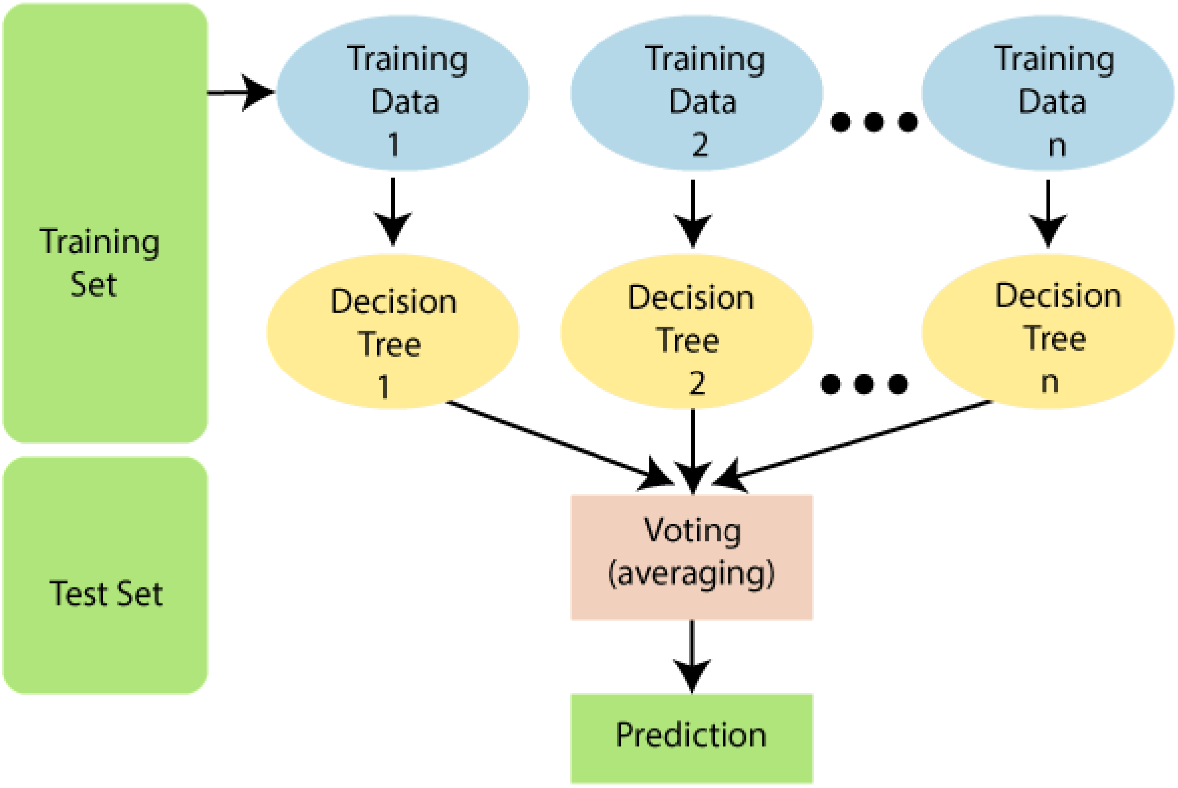
Random Forest Algorithm

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning,** which is a process of *combining multiple classifiers to solve a complex problem and to improve the performance of the model.*

As the name suggests, ***"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset."*** Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

The below diagram explains the working of the Random Forest algorithm:



Assumptions for Random Forest

Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random forest classifier:

- There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
- The predictions from each tree must have very low correlations.

Why use Random Forest?

Below are some points that explain why we should use the Random Forest algorithm:

- It takes less training time as compared to other algorithms.
- It predicts output with high accuracy, even for the large dataset it runs efficiently.
- It can also maintain accuracy when a large proportion of data is missing.

How does Random Forest algorithm work?

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps and diagram:

Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

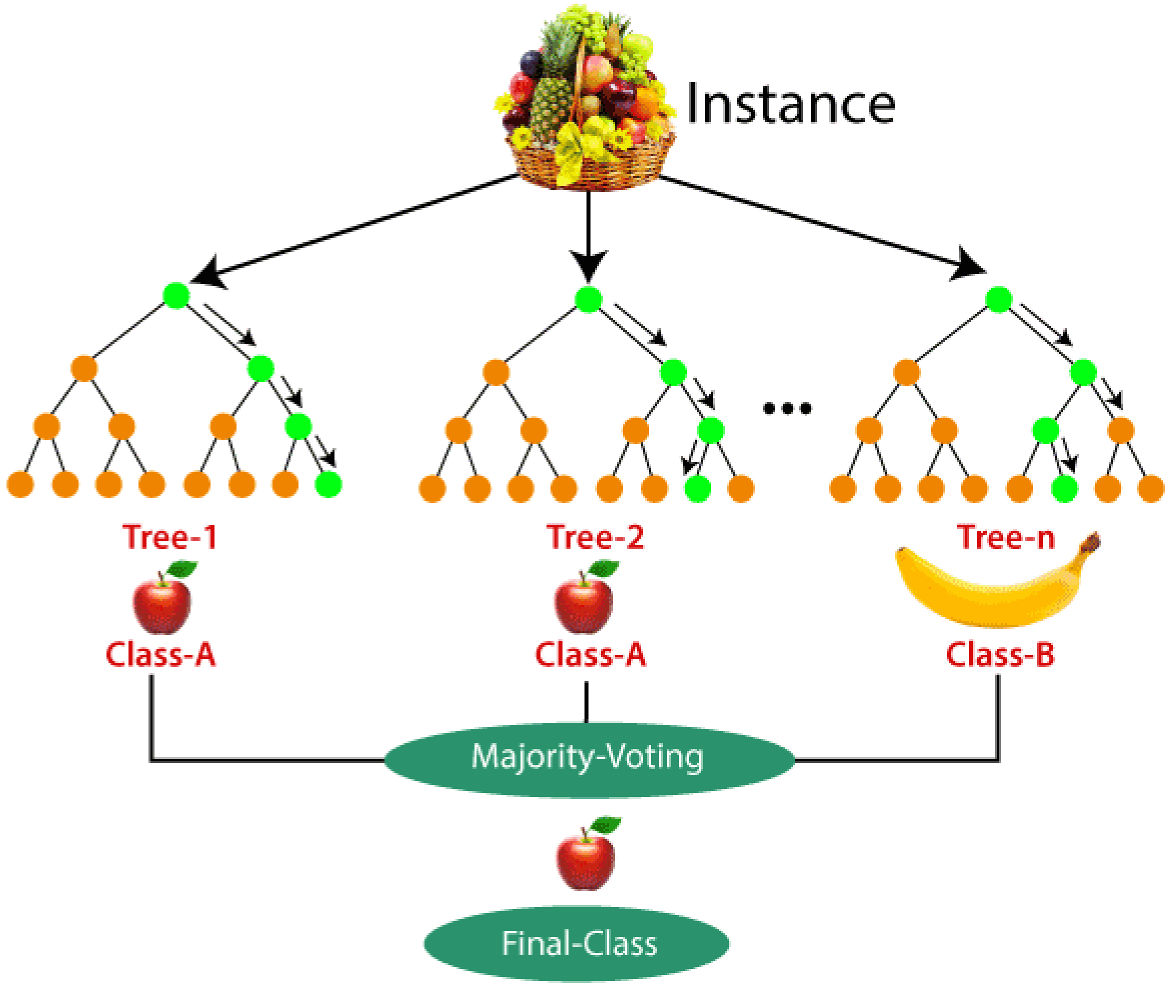
Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 & 2.

Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

The working of the algorithm can be better understood by the below example:

Example: Suppose there is a dataset that contains multiple fruit images. So, this dataset is given to the Random forest classifier. The dataset is divided into subsets and given to each decision tree. During the training phase, each decision tree produces a prediction result, and when a new data point occurs, then based on the majority of results, the Random Forest classifier predicts the final decision. Consider the below image:



Applications of Random Forest

There are mainly four sectors where Random forest mostly used:

1. **Banking:** Banking sector mostly uses this algorithm for the identification of loan risk.
2. **Medicine:** With the help of this algorithm, disease trends and risks of the disease can be identified.
3. **Land Use:** We can identify the areas of similar land use by this algorithm.
4. **Marketing:** Marketing trends can be identified using this algorithm.

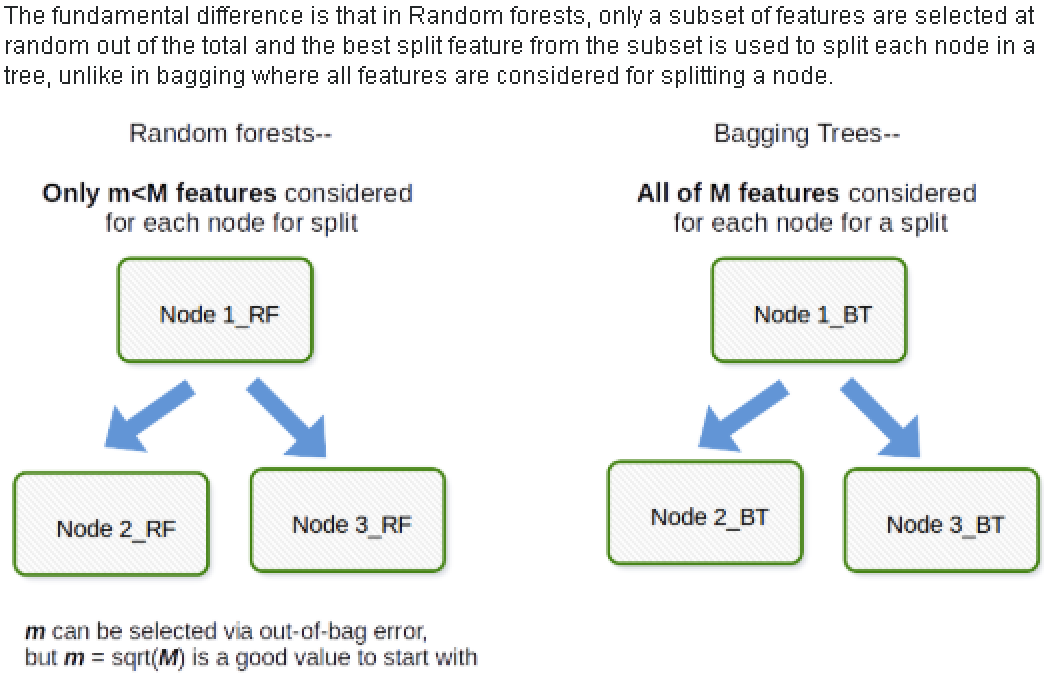
Advantages of Random Forest

- Random Forest is capable of performing both Classification and Regression tasks.
- It is capable of handling large datasets with high dimensionality.
- It enhances the accuracy of the model and prevents the overfitting issue.

Disadvantages of Random Forest

- Although random forest can be used for both classification and regression tasks, it is not more suitable for Regression tasks.

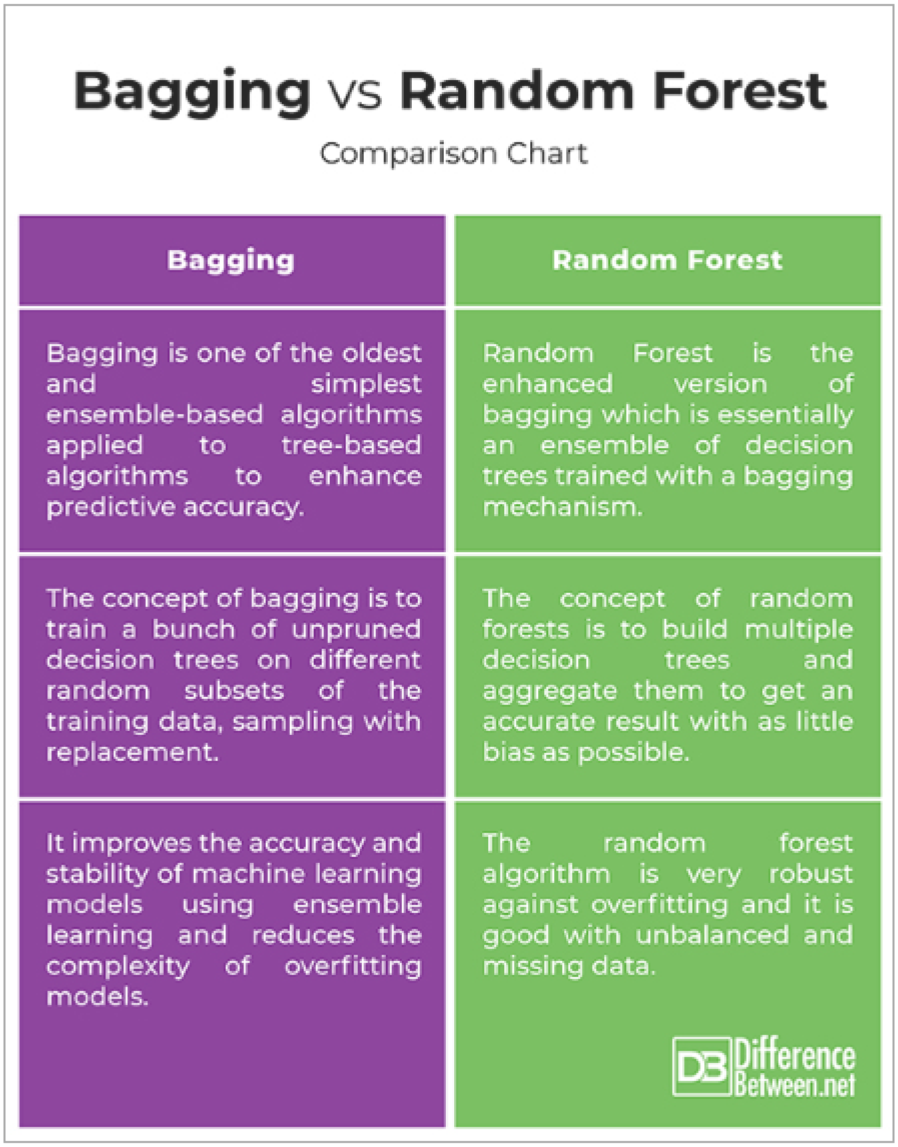
10) What is the difference between Bagging and Random forest? Why do we use Random forest more commonly than Bagging?



The fundamental difference is that in Random forests, only a subset of features are selected at random out of the total and the best split feature from the subset is used to split each node in a tree, unlike in bagging where all features are considered for splitting a node.

Random forests--
Only m<M features considered for each node for split
Node 1_RF
Node 2_RF
Node 3_RF
Bagging Trees--
All of M features considered for each node for a split
Node 1_BT
Node 2_BT
Node 3_BT

m can be selected via out-of-bag error, but ***m*** = sqrt(***M***) is a good value to start with



Bagging vs **Random Forest**

Comparison Chart

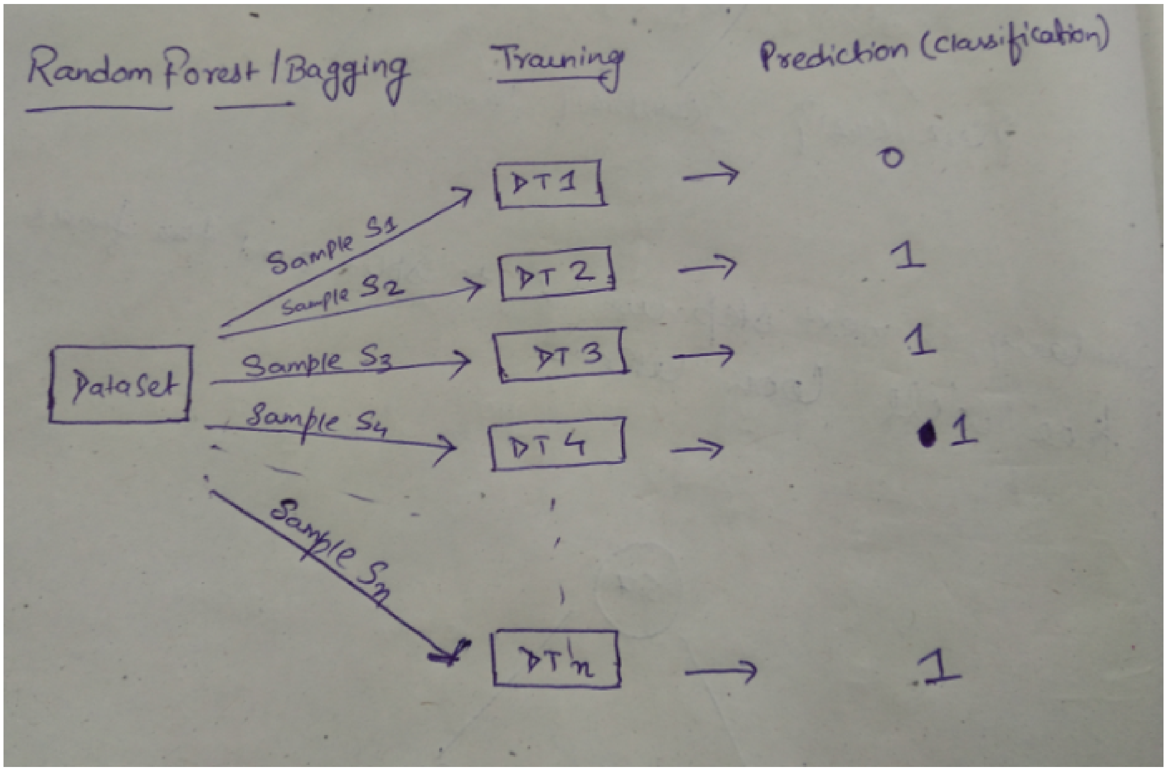
Bagging	Random Forest
Bagging is one of the oldest and simplest ensemble-based algorithms applied to tree-based algorithms to enhance predictive accuracy.	Random Forest is the enhanced version of bagging which is essentially an ensemble of decision trees trained with a bagging mechanism.
The concept of bagging is to train a bunch of unpruned decision trees on different random subsets of the training data, sampling with replacement.	The concept of random forests is to build multiple decision trees and aggregate them to get an accurate result with as little bias as possible.
It improves the accuracy and stability of machine learning models using ensemble learning and reduces the complexity of overfitting models.	The random forest algorithm is very robust against overfitting and it is good with unbalanced and missing data.

DB Difference Between.net

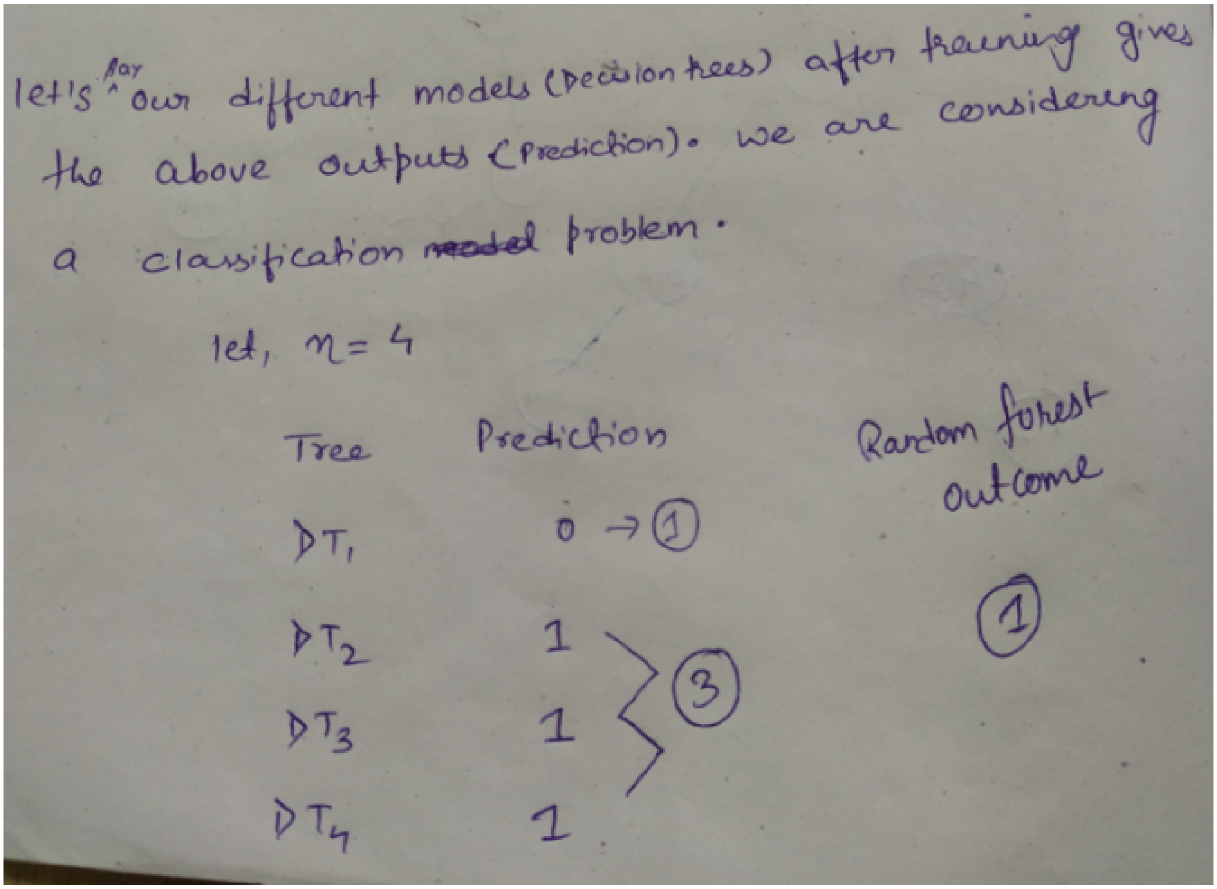
11) What is feature sampling?

```
Now, the difference with in random forest is how the trees are formed. In bootstraping we allow all the sample data to
be used for splitting the nodes but not   with random forests.  When building a decision tree, each time a split is to
happen, a random sample of ‘m’ independent variable are chosen from the total ‘p’ independent variable. Only those ‘m’
independent variable are allowed to be used for the split.


Why is that?

Suppose in those ‘p’ independent variables, 1 independent variable is very strong. Now each sample this independent var
iable will remain the strongest. So, whenever trees will be built for these sampled data, this independent variable wil
l be chosen by all the trees for splitting and thus will result in similar kind of tree formation for each bootstrap mo
del. **This introduces correaltion in the dataset and averaging correalted dataset results do not lead low variance.**
That’s why in random forest the choice for selecting node for split is limited and it introduces randomness in the form
ation of the trees as well.

Most of the independent variable are not allowed to be considered for split.
Generally, value of ‘m’ is taken as m ≈√p , where ‘p’ is the number of independent variable in the sample.

When m=p , the random forest model becomes bagging model.

**This method is also referred as “Feature Sampling”**
```

12) How prediction is made in Random Forest?



e dataset different samples are created by bootstrapping and these samples are used to train different decision trees. Once the



let's say our different models (Decision trees) after training gives the above outputs (Prediction). we are considering a classification ~~model~~ problem.

let, $n = 4$

Tree	Prediction		Random forest outcome
DT_1	0 →	1	1
DT_2	1	3	
DT_3	1		
DT_4	1		

13) When should we not use Random forest model?

Decision Tree is a stand alone model, while a Random Forest is an ensemble of Decision Trees. Decision Tree is a weak learner. It is prone to over fitting (high variance) and is highly dependent on the training sample. Since a RF consists of a huge number of decision trees, it adds regularization and hence becomes a strong learner.

Decision Trees Decision Tree algorithm is an algorithm that can be used to solve regression as well as classification problems. The main objective of the creation of a decision tree is to build a training model. This training model is used to predict the value or class of the recipient variables. The level of understanding of the decision trees algorithm is much easier than the other classification algorithms.

In the random forest classifier, every decision tree forecasts a response for an occurrence and the endmost response is decided through voting. On contrary, in classification, the response received by majority voting of Decision Tree is the final response and in regression, the final response is the average of all the responses.

Why use Random Forest Algorithm To answer this question, we will suggest some of its advantages which will clear your mind why use Random Forest Algorithm in machine learning.

Random forest algorithm can be used for both classifications and regression task. It provides higher accuracy. Random forest classifier will handle the missing values and maintain the accuracy of a large proportion of data. If there are more trees, it won’t allow overfitting trees in the model. It has the power to handle a large data set with higher dimensionality How does it work In the random forest, we grow multiple trees in a model. To classify a new object based on new attributes each tree gives a classification and we say that tree votes for that class. The forest chooses the classifications having the most votes of all the other trees in the forest and takes the average difference from the output of different trees. In general, Random Forest built multiple trees and combines them together to get a more accurate result.

While creating random trees it split into different nodes or subsets. Then it searches for the best outcome from the random subsets. This results in the better model of the algorithm. Thus, in a random forest, only the random subset is taken into consideration.

To give you a clear idea about the working of a random tree, let us see an example.

Suppose we formed a thousand random trees to form the random forest to detect a ‘hand’. Each random forest will predict the different outcomes or the class for the same test features. A small subset of the forest will look at the random set of features, for example, hand or fingers. Suppose some hundred random decision trees predict some unique targets such as thumb, fingers or human. Then the votes of the finger are calculated out of a hundred random decisions and also the votes of thumb and human. If votes of the finger are higher, then the final random forest will return the finger as a predicted target. This type of voting is called majority voting. The same applies to the rest of the fingers of the hand, if the algorithm predicts the rest of the fingers to be fingers of a hand, then the high-level decision tree can vote that an image is a ‘hand’. This is why the random forest is also known as Ensemble machine learning algorithm.

In machine learning, this algorithm helps in several ways and most of the applications are underway. Below we have discussed the use of this algorithm in machine learning in a few sectors.

When to use Random Forest Analysis There are several applications where the random forest can be applied. We will discuss some of the sectors where random forest can be applied. We will also look closer when the random forest analysis comes into the role.

Banking Sector: The banking sector consists of most users. There are many loyal customers and also fraud customers. To determine whether the customer is a loyal or fraud, Random forest analysis comes in. With the help of a random forest algorithm in machine learning, we can easily determine whether the customer is fraud or loyal. A system uses a set of a random algorithm which identifies the fraud transactions by a series of the pattern.

Medicines: Medicines needs a complex combination of specific chemicals. Thus, to identify the great combination in the medicines, Random forest can be used. With the help of machine learning algorithm, it has become easier to detect and predict the drug sensitivity of a medicine. Also, it helps to identify the patient’s disease by analyzing the patient’s medical record.

Stock Market: Machine learning also plays role in the stock market analysis. When you want to know the behavior of the stock market, with the help of Random forest algorithm, the behavior of the stock market can be analyzed. Also, it can show the expected loss or profit which can be produced while purchasing a particular stock.

E-Commerce: When you will find it difficult to recommend or suggest what type of products your customer should see. This is where you can use a random forest algorithm. Using a machine learning system, you can suggest the products which will be more likely for a customer. Using a certain pattern and following the product’s interest of a customer, you can suggest similar products to your customers.

There are a couple of obvious cases where random forests will struggle:

Sparsity - When the data are very sparse, it's very plausible that for some node, the bootstrapped sample and the random subset of features will collaborate to produce an invariant feature space. There's no productive split to be had, so it's unlikely that the children of this node will be at all helpful. XGBoost can do better in this context.

Data are not axis-aligned - Suppose that there is a diagonal decision boundary in the space of two features, x1 and x2. Even if this is the only relevant dimension to your data, it will take an ordinary random forest model many splits to describe that diagonal boundary. This is because each split is oriented perpendicular to the axis of either x1 or x2. (This should be intuitive because an ordinary random forest model is making splits of the form x1>4.) Rotation forest, which performs a PCA projection on the subset of features selected for each split, can be used to overcome this: the projections into an orthogonal basis will, in principle, reduce the influence of the axis-aligned property because the splits will no longer be axis-aligned in the original basis.

This image provides another example of how axis-aligned splits influence random forest decisions. The decision boundary is a circle at the origin, but note that this particular random forest model draws a box to approximate the circle. There are a number of things one could do to improve this boundary; the simplest include gathering more data and building more trees.

14) What is Stacking?

Bagging

Bagging or Bootstrap Aggregation is a powerful, effective and simple ensemble method. The method uses multiple versions of a training set by using the bootstrap, i.e. sampling with replacement and t it can be used with any type of model for classification or regression. Bagging is only effective when using unstable (i.e. a small change in the training set can cause a significant change in the model) non-linear models.

Boosting

Boosting is a meta-algorithm which can be viewed as a model averaging method. It is the most widely used ensemble method and one of the most powerful learning ideas. This method was originally designed for classification but it can also be profitably extended to regression. The original boosting algorithm combined three weak learners to generate a strong learner.

Stacking

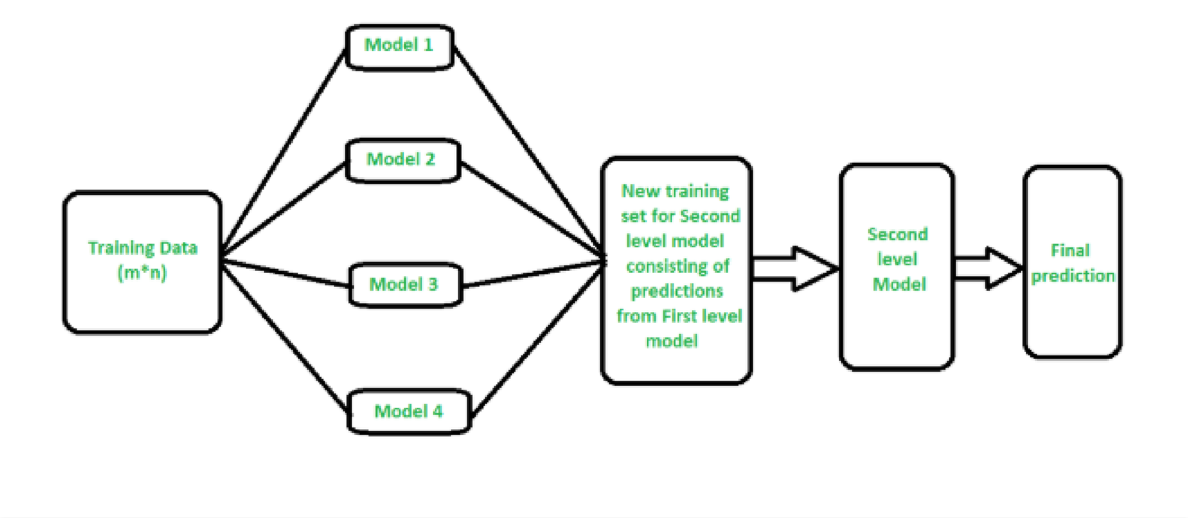
Stacking is concerned with combining multiple classifiers generated by using different learning algorithms on a single dataset which consists of pairs of feature vectors and their classifications. This technique consists of basically two phases, in the first phase, a set of base-level classifiers is generated and in the second phase, a meta-level classifier is learned which combines the outputs of the base-level classifiers.

15) Explain the working behind Stacking.

Stacking in Machine Learning

Stacking is a way to ensemble multiple classifications or regression model. There are many ways to ensemble models, the widely known models are ***Bagging*** or ***Boosting***. Bagging allows multiple similar models with high variance are averaged to decrease variance. Boosting builds multiple incremental models to decrease the bias, while keeping variance small.

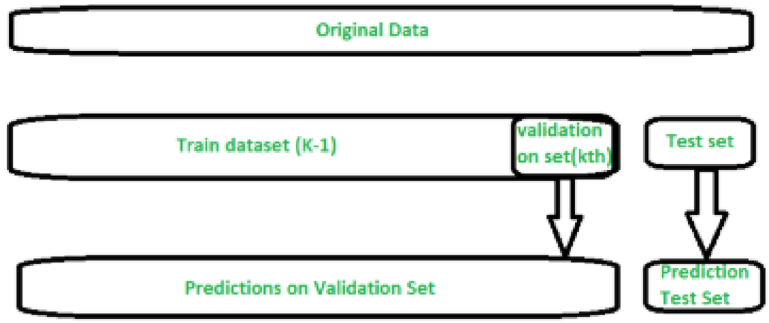
Stacking (sometimes called *Stacked Generalization*) is a different paradigm. The point of stacking is to explore a space of different models for the same problem. The idea is that you can attack a learning problem with different types of models which are capable to learn some part of the problem, but not the whole space of the problem. So, you can build multiple different learners and you use them to build an intermediate prediction, one prediction for each learned model. Then you add a new model which learns from the intermediate predictions the same target.
This final model is said to be stacked on the top of the others, hence the name. Thus, you might improve your overall performance, and often you end up with a model which is better than any individual intermediate model. Notice however, that it does not give you any guarantee, as is often the case with any machine learning technique.


Model 1
Model 2
Training Data (m*n)
Model 3
Model 4
New training set for Second level model consisting of predictions from First level model
Second level Model
Final prediction

How stacking works?

How stacking works?

1. We split the training data into K-folds just like K-fold cross-validation.
2. A base model is fitted on the K-1 parts and predictions are made for Kth part.
3. We do for each part of the training data.
4. The base model is then fitted on the whole train data set to calculate its performance on the test set.
5. We repeat the last 3 steps for other base models.
6. Predictions from the train set are used as features for the second level model.
7. Second level model is used to make a prediction on the test set.



Blending –

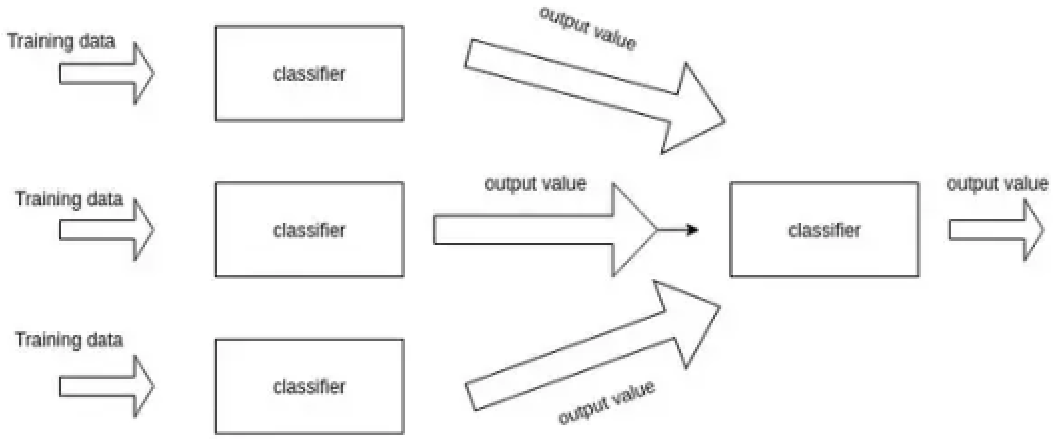
Blending is a similar approach to stacking.

- The train set is split into training and validation sets.
- We train the base models on the training set.
- We make predictions only on the validation set and the test set.
- The validation predictions are used as features to build a new model.
- This model is used to make final predictions on the test set using the prediction values as features.

Stacking is a way of combining multiple models, that introduces the concept of a meta learner. It is less widely used than bagging and boosting. Unlike bagging and boosting, stacking may be (and normally is) used to combine models of different types. The procedure is as follows:

1. Split the training set into two disjoint sets.
2. Train several base learners on the first part.
3. Test the base learners on the second part.
4. Using the predictions from 3) as the inputs, and the correct responses as the outputs, train a higher level learner.

In stacking, the combining mechanism is that the output of the classifiers (Level 0 classifiers) will be used as training data for another classifier (Level 1 classifier) to approximate the same target function. Basically, you let the Level 1 classifier to figure out the combining mechanism.



```
In [ ]:
```