|  |  |  |  |
| --- | --- | --- | --- |
| **Topic: Decision trees** | **Theory** | **Mathematics** | **Numerical** |
|  |  |  |

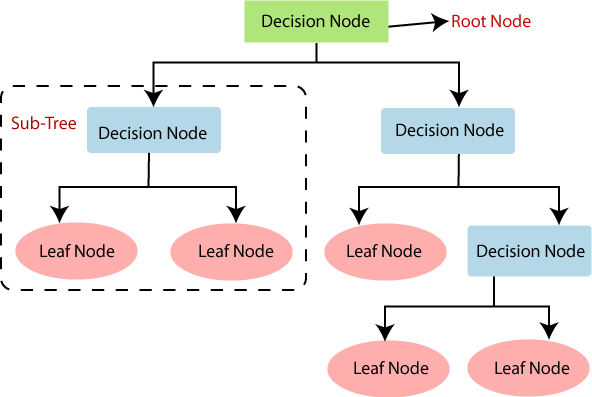
**Theory questions 1. Why use decision trees?** There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning

model. Below are the two reasons for using the Decision tree:

* Decision Tree is a **Supervised learning technique** that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems.
* It is a tree-structured classifier, where **internal nodes represent the features of a dataset, branches represent the decision rules** and **each leaf node represents the outcome.**
* In a Decision tree, there are two nodes, which are the **Decision Node** and **Leaf Node.** Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
* The decisions or the test are performed on the basis of features of the given dataset.

#### It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.

* It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
* A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into sub-trees. Below diagram explains the general structure of a decision tree.



* It is a tree-structured classifier, where **internal nodes represent the features of a dataset, branches represent the decision rules** and **each leaf node represents the outcome.**
* Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.
* The logic behind the decision tree can be easily understood because it shows a tree-like structure.

**2. Explain decision tree terminology.**

The decision tree comprises of root node, leaf node, branch nodes, parent/child node etc.

following is the explanation of this terminology.

* **Root Node:** Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
* **Leaf Node:** Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
* **Splitting:** Splitting is the process of dividing the decision node/root node into sub- nodes according to the given conditions.
* **Branch/Sub Tree:** A tree formed by splitting the tree.
* **Pruning:** Pruning is the process of removing the unwanted branches from the tree.
* **Parent/Child node:** The root node of the tree is called the parent node, and other nodes are called the child nodes.

**3. How does the Decision Tree algorithm Work for classification?**

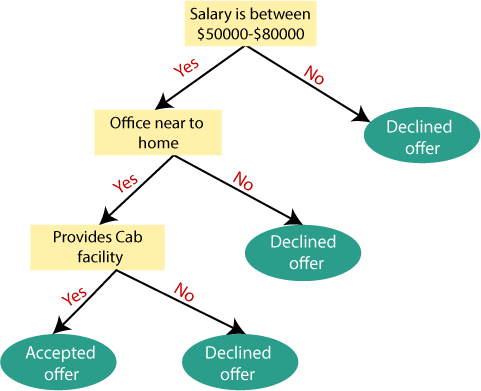
In a decision tree, for predicting the class of the given dataset, the algorithm starts from the

root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node. For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree.

* **Step-1:** Begin the tree with the root node, says S, which contains the complete dataset.
* **Step-2:** Find the best attribute in the dataset using **Attribute Selection Measure (ASM)**

#### i.e. information gain and Gini index.

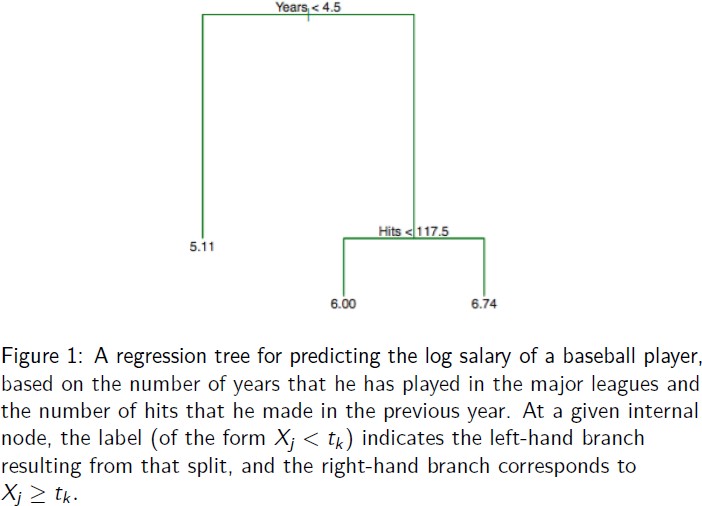
* **Step-3:** Divide the S into subsets that contains possible values for the best attributes.
* **Step-4:** Generate the decision tree node, which contains the best attribute.
* **Step-5:** Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

**Example:** Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or Not. So, to solve this problem, the decision tree starts with the root node (Salary attribute by ASM).

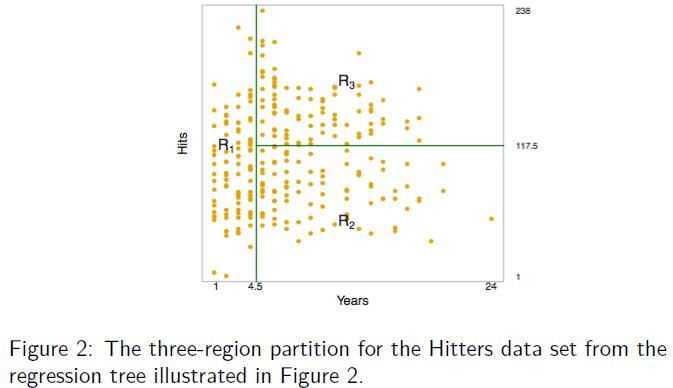
The root node splits further into the next decision node (distance from the office) and one leaf node based on the corresponding labels. The next decision node further gets split into one decision node (Cab facility) and one leaf node. Finally, the decision node splits into two leaf nodes (Accepted offers and Declined offer). See the above figure.

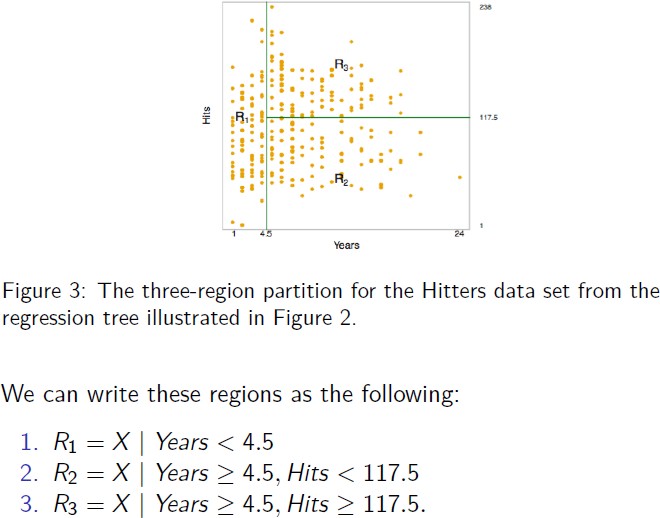
**4. How does the Decision Tree algorithm Work for regression?** The general idea is that we will segment the predictor space into a number of simple regions. In order to make a prediction for a given observation, we typically use the mean of the training data in the region to which it belongs. Since the set of splitting rules used to segment the predictor space can be summarized by a tree such approaches are called decision tree methods. These methods are simple and useful for interpretation. We want to

predict a response or class *Y* from inputs *X*1*,X*2*, . . .Xp*. We do this by growing a binary tree. At each internal node in the tree, we apply a test to one of the inputs, say *Xi* . Depending on the outcome of the test, we go to either the left or the right sub-branch of the tree. Eventually we come to a leaf node, where we make a prediction. This prediction aggregates or averages all the training data points which reach that leaf. In order to motivate regression trees, we begin with a simple example. Our motivation is to predict a baseball player’s Salary based on Years (the number of years that he has played in the major leagues) and Hits (the number of hits that he made in the previous year). We first remove observations that are

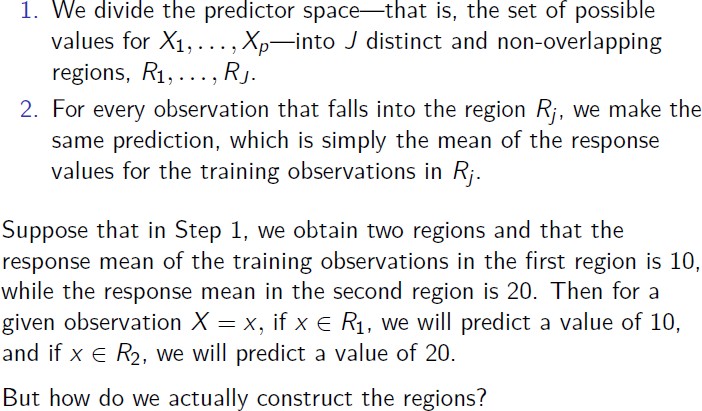
missing Salary values and log-transform Salary so that its distribution has more of a typical bell-shape. Recall that Salary is measured in thousands of dollars.

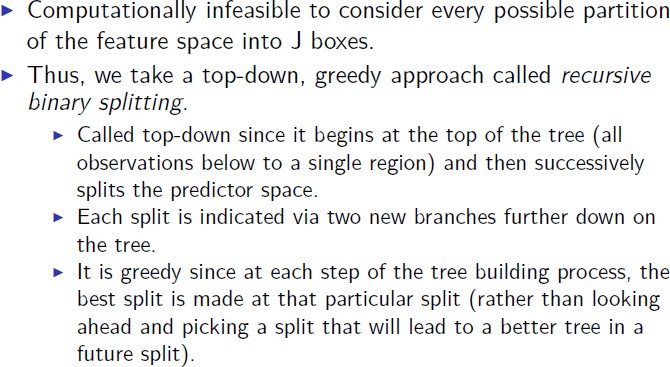
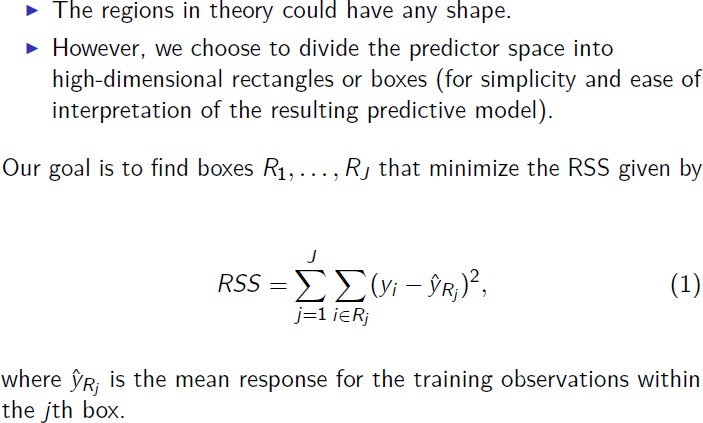
The tree represents a series of splits starting at the top of the tree. The top split assigns observations having Years < 4.5 to the left branch. The predicted salary for these players is given by the mean response value for the players in the data set with Years < 4.5.For such players, the mean log salary is 5.107, and so we make a prediction of e5.107 thousands of dollars, i.e. 165, 174. How would you interpret the rest (right branch) of the tree?





* In keeping with the tree analogy, the regions *R*1*, R*2*,* and *R*3 are known as **terminal nodes** or **leaves** of the tree.
* As is the case for Figure 2, decision trees are typically drawn upside down, in the sense that the **leaves are at the bottom of the tree**.
* The points along the tree where the predictor space is split are referred to as **internal nodes**.
* In Figure 2, the two internal nodes are indicated by the text *Years <* 4*.*5 and *Hits <* 117*.*5*.*
* We refer to the segments of the trees that connect the nodes as **branches**.
* Years is the most important factor in determining Salary, and players with less experience earn lower salaries than more experienced players.
* Given that a player is less experienced, the number of hits that he made in the previous year seems to play little role in his salary.
* But among players who have been in the major leagues for five or more years, the number of hits made in the previous year does affect salary, and players who made more hits last year tend to have higher salaries.
* The regression tree shown in Figure 2 is likely an over-simplification of the true relationship between Hits, Years, and Salary, but it‘s a very nice easy interpretation over more complicated approaches.





**Mathematics based questions 5. Explain entropy reduction, information gain and Gini index in decision tree.** While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a

technique which is called as **Attribute selection measure or ASM.** By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:

#### Information Gain:

* Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.
* It calculates how much information a feature provides us about a class.
* According to the value of information gain, we split the node and build the decision tree. A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

#### Information Gain= Entropy(S) – [(Weighted Average) \* Entropy (each feature)] Entropy:

Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in

data. Entropy can be calculated as:

#### Entropy(s) = – P(yes)log2 P(yes) – P(no) log2 P(no)

Where, S= Total number of samples, P(yes)= probability of yes, P(no)= probability of no

#### Gini Index:

* Gini index is a measure of impurity or purity used while creating a decision tree in the CART (Classification and Regression Tree) algorithm.
* An attribute with the low Gini index should be preferred as compared to the high Gini index. Gini index can be calculated using the formula: **Gini Index= 1 – ∑jPj2**
* It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.

#### What are advantages and limitations of the decision trees?

**Advantages of the Decision Tree**

* + It is simple to understand as it follows the same process which a human follow while making any decision in real-life.
  + It can be very useful for solving decision-related problems.
  + It helps to think about all the possible outcomes for a problem.
  + There is less requirement of data cleaning compared to other algorithms.

#### Disadvantages of the Decision Tree

* + The decision tree contains lots of layers, which makes it complex.
  + It may have an overfitting issue, which can be resolved using the **Random Forest algorithm.**
  + For more class labels, the computational complexity of the decision tree may increase.

**7. Many times while training decision tree tends to overfit. What is the reason behind it and how to avoid it?**

Decision tree tends to overfit since at each node, it will make the decision among a subset of

all the features (columns), so when it reaches a final decision, it is a complicated and long decision chain. Only if a data point satisfies all the rules along this chain, the final decision can be made. This kind of specific rule on training dataset make it very specific for the training set, on the other hand, cannot generalize well for new data points that it has never seen. Especially when your dataset has many features (high dimension), it tends to overfit more. In J48 decision tree, over fitting happens when algorithm gets information with exceptional attributes. This causes many fragmentations in the process distribution. Statistically unimportant nodes with least examples are known as fragmentations. Usually J48 algorithm builds trees and grows its branches ‗just deep enough to perfectly classify the training examples‘. This approach performs better with noise free data. But most of the time this strategy overfits the training examples with noisy data. At present there are two strategies which are widely used to bypass this overfitting in decision tree learning. Those are: 1) If tree grows taller, stop it from growing before it reaches the maximum point of accurate classification of the training data. 2) Let the tree to over-fit the training data then post-prune tree. By default, the decision tree model is allowed to grow to its full depth. Pruning refers to a technique to remove the parts of the decision tree to prevent growing to its full depth. By tuning the hyperparameters of the decision tree model one can prune the trees and prevent them from overfitting. There are two types of pruning Pre-pruning and Post-pruning. Now let's discuss the in-depth understanding and hands-on implementation of each of these pruning techniques.

#### Pre-Pruning:

The pre-pruning technique refers to the early stopping of the growth of the decision tree. The pre-pruning technique involves tuning the hyperparameters of the decision tree model prior to the training pipeline. The hyperparameters of the decision tree including **max\_depth**, **min\_samples\_leaf**, **min\_samples\_split** can be tuned to early stop the growth of the tree and prevent the model from overfitting.

#### Post-Pruning:

The Post-pruning technique allows the decision tree model to grow to its full depth, then removes the tree branches to prevent the model from overfitting. ***Cost complexity pruning (ccp)*** is one type of post-pruning technique. In case of cost complexity pruning, the **ccp\_alpha** can be tuned to get the best fit model.

#### Problems/Numerical 8. Problems on calculating entropy and information gain

**Problem 1:** If we decided to arbitrarily label all 4 gumballs as red, how often would one of the gumballs is incorrectly labelled?

#### https://miro.medium.com/max/700/1*erKcPX4hLgOadCyoyNVlTQ.png4 red and 0 blue:

The impurity measurement is 0 because we would never incorrectly label any of the 4 red gumballs here. If we arbitrarily chose to label all the balls ‗blue‘, then our index would still be 0, because we would always incorrectly label the gumballs.

The gini score is always the same no matter what arbitrary class you take the probabilities of because they always add to 0 in the formula above.

A gini score of 0 is the most pure score possible.

#### https://miro.medium.com/max/700/1*rpi08Qustg9vSKYcL-UUWQ.pngred and 2 blue:

The impurity measurement is 0.5 because we would incorrectly label gumballs wrong about half the time. Because this index is used in binary target variables (0,1), a gini index of 0.5 is the least pure score possible. Half is one type and half is the other. **Dividing gini scores by**

#### 0.5 can help intuitively understand what the score represents. 0.5/0.5 = 1, meaning the grouping is as impure as possible (in a group with just 2 outcomes).

1. https://miro.medium.com/max/700/1*2TokVtgv-dbBYy3cg8EigA.png**red and 1 blue:**

The impurity measurement here is 0.375. If we divide this by 0.5 for more intuitive understanding we will get 0.75, which is the probability of incorrectly/correctly labeling.

#### Problem 2:

How does entropy work with the same gumball scenarios stated in problem 1?

#### https://miro.medium.com/max/700/1*UVWAeM4YuboxpgTnbXr8MA.pngred and 0 blue:

Unsurprisingly, the impurity measurement is 0 for entropy as well. This is the max purity score using information entropy.

#### https://miro.medium.com/max/700/1*slKimbAlcP3F5_fOl0cb4A.pngred and 2 blue:

The impurity measurement is 1 here, as it‘s the maximum impurity obtainable.

#### https://miro.medium.com/max/700/1*VuvJX7gw44VDWF-MyuBo6A.pngred and 1 blue:

The purity/impurity measurement is 0.811 here, a bit worse than the gini score.

Calculate entropy for following example. For the set X = {a,a,a,b,b,b,b,b}

Total instances: 8 Instances of b: 5 Instances of a: 3



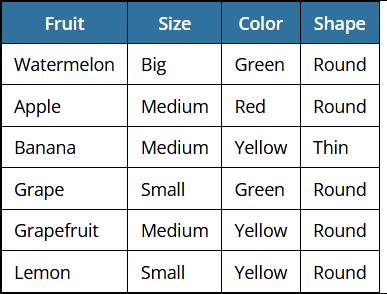
= - [0.375 \* (-1.415) + 0.625 \* (-0.678)]

= - (-0.53-0.424)

#### = 0.954

**Problem 3:**

#### Problem 4:

In the below mini-dataset, the label we‘re trying to predict is the type of fruit. This is based off the size, color, and shape variables.

#### Calculate the information gained if we select the *color* variable.

3 out of the 6 records are yellow, 2 are green, and 1 is red. Proportionally, the probability of a yellow fruit is 3 / 6 = 0.5; 2 / 6 = 0.333 for green, and 1 / 6 = 0.1666 for red. Using the formula from above, we can calculate it like this:

Information gain = - ([3/6 \* log2(3/6)] + [2/6 \* log2(2/6)] + [1/6 \* log2(1/6)]) **= 1.459148 Calculate the information gained if we select the *size* variable.**

Information gain = - ([3/6 \* log2(3/6)] + [2/6 \* log2(2/6)] + [1/6 \* log2(1/6)]) **= 1.459148**

In this case, 3 / 6 of the fruits are medium-sized, 2 / 6 are small, 1 / 6 is big.

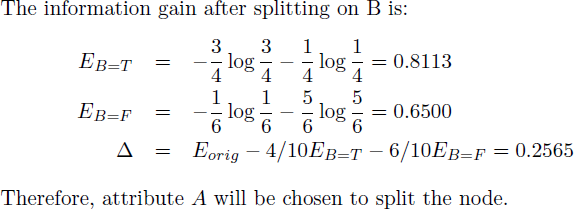
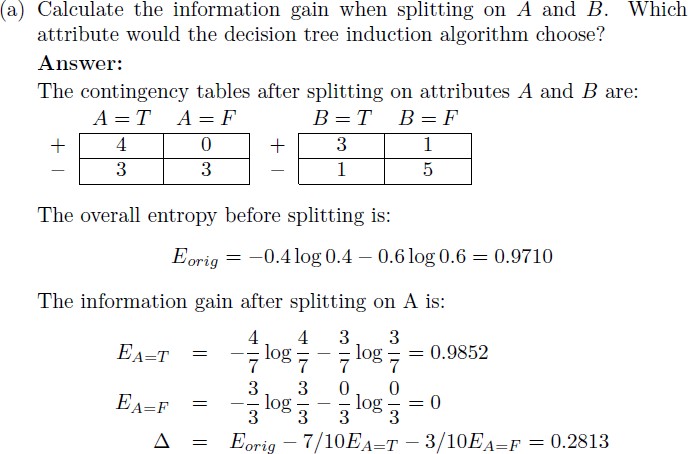
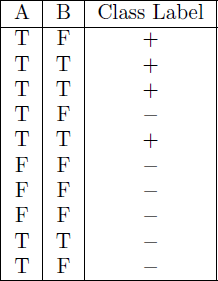
#### Calculate the information gained if we select the *shape* variable.

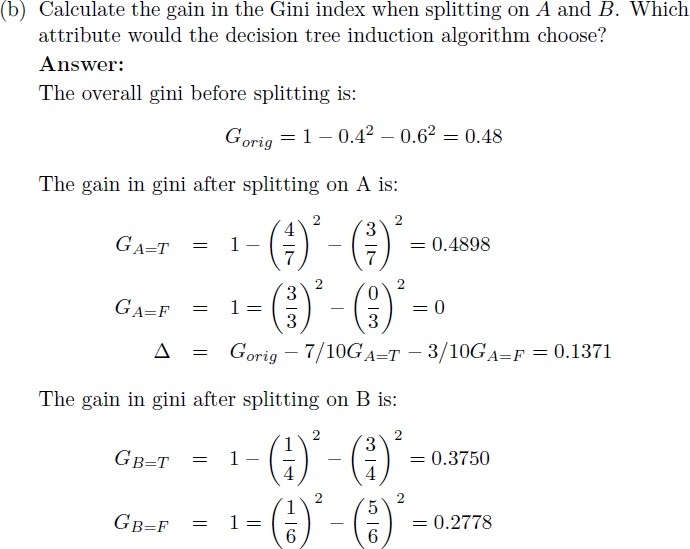
Here, 5 / 6 of the fruits are round and 1 / 6 is thin.

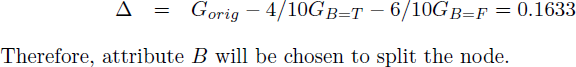
Information gain = - ([5/6 \* log2(5/6)] + [1/6 \* log2(1/6)]) **= 0.650022**

#### Problem 5:

Consider the following data set for a binary class problem.

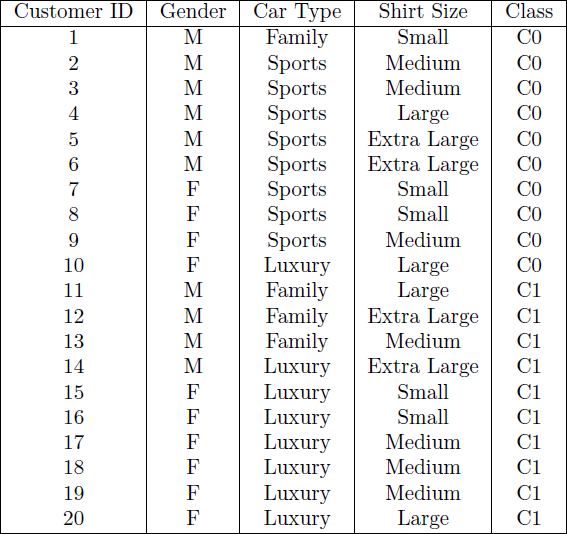


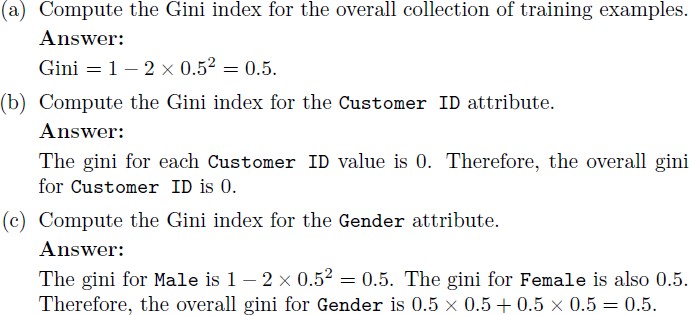


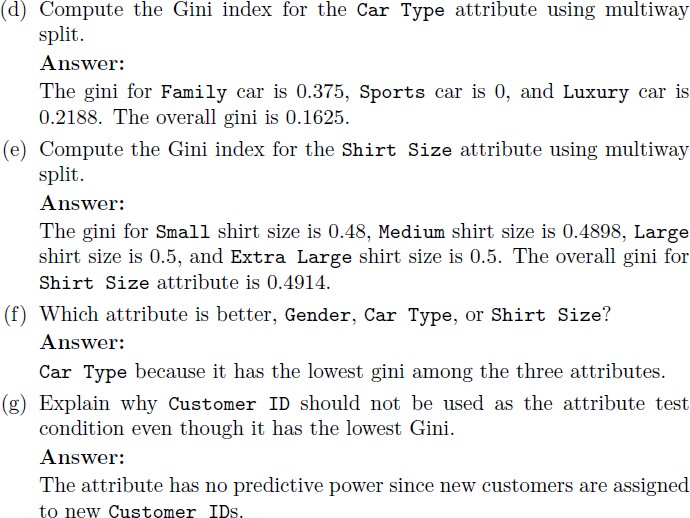


#### Problem 6:

Consider the training examples shown in Table below for a binary classification problem.

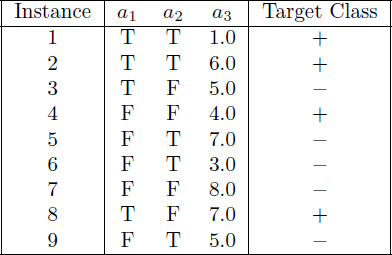


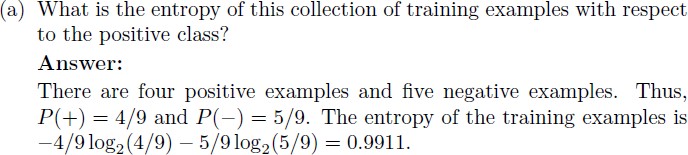


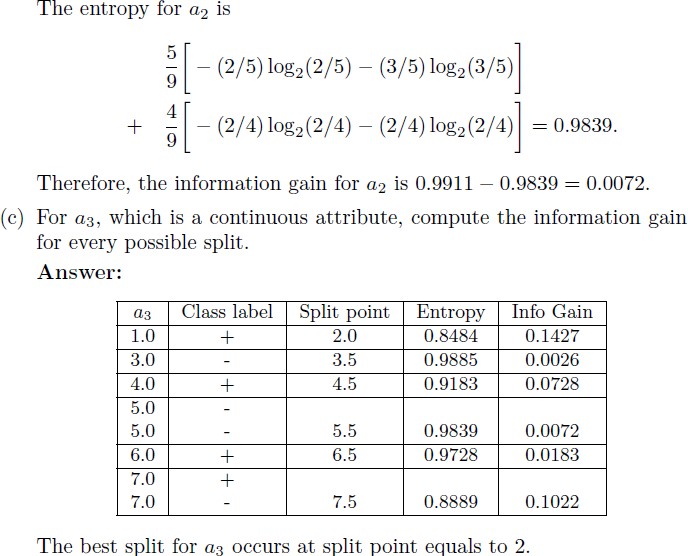
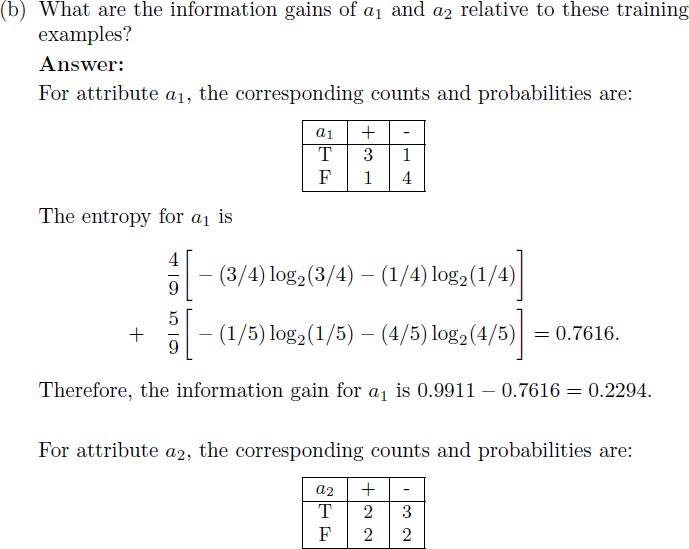


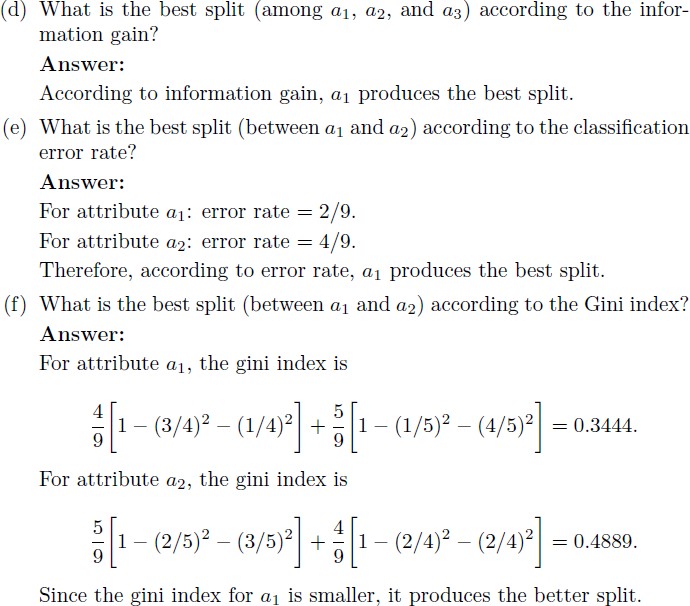
#### Problem 7:

Consider the training examples shown in Table below for a binary classification problem.









|  |  |  |  |
| --- | --- | --- | --- |
| **Topic: Random forest tree** | **Theory** | **Mathematics** | **Numerical** |
|  |  |  |

**Theory questions 9. Why use random forest trees?** 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.**

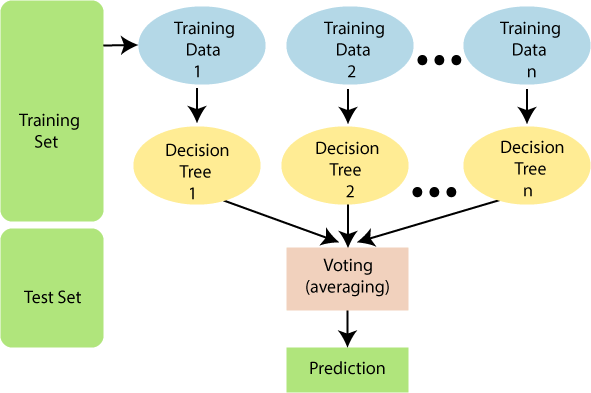
#### Assumptions for Random Forest

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.

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



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.
* It can be used for both classifications as well as regression tasks.
* Overfitting problem that is censorious and can make results poor but in case of the random forest the classifier will not overfit if there are enough trees.
* It can be used for categorical values as well.



1. **How does the random forest tree work for classification?** 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.

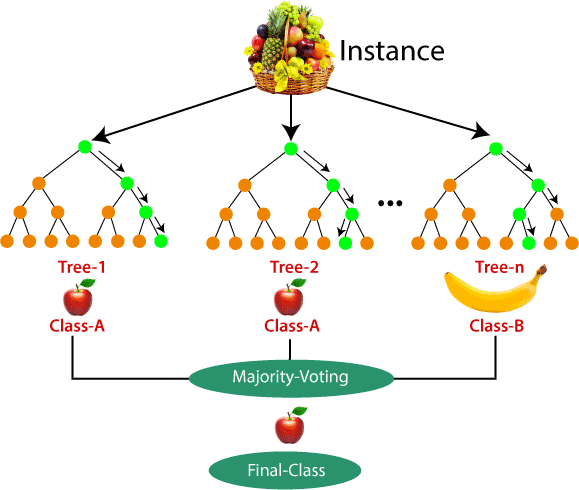
**Step-2:** Build the decision trees associated with the selected data points (Subsets).

**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.

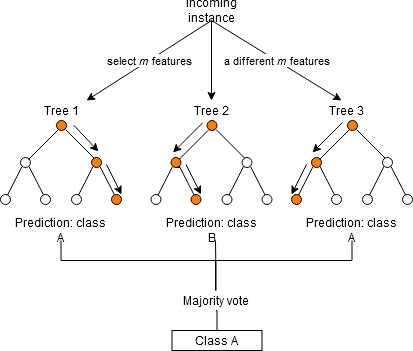
The working of the algorithm can be better understood by the below example:

**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:

#### Explain random forest tree terminology.

* **Bagging:** Given the training set of *N* examples, we repeatedly sample subsets of the training data of size *n* where *n* is less than *N*. Sampling is done at random but with replacement. This subsampling of a training set is called *bootstrap aggregating*, or *bagging*, for short.
* **Random subspace method:** If each training example has *M* features, we take a subset of them of size *m* < *M* to train each estimator. So no estimator sees the full training set, each estimator sees only *m* features of *n* training examples.
* **Training estimators:** We create *Ntree* decision trees, or estimators, and train each one on a different set of *m* features and *n* training examples. The trees are not pruned, as they would be in the case of training a simple decision tree classifier.
* **Perform inference by aggregating predictions of estimators:** To make a prediction for a new incoming example, we pass the relevant features of this example to each of the *Ntree* estimators. We will obtain *Ntree* predictions, which we need to combine to produce the overall prediction of the random forest. In the case of classification, we will use

majority voting to decide on the predicted class, and in the case of regression, we will take the mean value of the predictions of all the estimators.



***Random forest inference for a simple classification example with Ntree* = 3**

This use of many estimators is the reason why the random forest algorithm is called an *ensemble method*. Each individual estimator is a weak learner, but when many weak estimators are combined together they can produce a much stronger learner. Ensemble methods take a 'strength in numbers' approach, where the output of many small models is combined to produce a much more accurate and powerful prediction.

#### What are advantages and limitations of the random forest tree?

**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.
* It is fast and can deal with missing values data as well.
* Using random forest you can compute the relative feature importance.
* It can give good accuracy even if the higher volume of data is missing.

#### Limitations of Random Forest

* Although random forest can be used for both classification and regression tasks, it is not more suitable for Regression tasks.
* Random forest is a complex algorithm that is not easy to interpret.
* Complexity is large.
* Predictions given by random forest takes many times if we compare it to other algorithms
* Higher computational resources are required to use a random forest algorithm.

**13. What is the difference between simple decision tree and random forest tree?**

|  |  |  |
| --- | --- | --- |
| **SN** | **Random Forest** | **Decision Tree** |
| 1. | While building a random forest the  number of rows is selected randomly. | Whereas, it built several decision trees  and find out the output. |
| 2. | It combines two or more decision trees  together. | Whereas the decision is a collection of  variables or data set or attributes. |
| 3. | It gives accurate results. | Whereas it gives less accurate results. |
| 4. | By using multiple trees it reduces the  chances of overfitting. | On the other hand, decision trees, it has  the possibility of overfitting, which is an error that occurs due to variance or due to bias. |
| 5. | Random forest is more complicated to  interpret. | Whereas, the decision tree is simple so it  is easy to read and understand. |
| 6. | In a random forest, we need to  generate, process, and analyze trees so that this process is slow, it may take one hour or even days. | The decision tree is not accurate but it  processes fast which means it is fast to implement. |
| 7. | It has more computation because it has  n number of decision trees, so more decision trees more computation. | Whereas it has less computation. |
| 8. | It has complex visualization, but it plays  an important role to show hidden patterns behind the data. | On the other hand, it is simple to  visualize because we just need to fit the decision tree model. |
| 9. | The classification and regression  problems can be solved by using random forest. | Whereas a decision tree is used to solve  the classification and regression problems. |
| 10. | It uses the random subspace method  and bagging during tree construction, which has built-in feature importance. | Whereas a decision is made based on the  selected sample‘s feature, this is usually a feature that is used to make a decision, decision tree learning is a process to find the optimal value for each internal tree node. |

Decision trees are simple but suffer from some serious problems- overfitting, error due to

variance or error due to bias. Random Forest is the collection of decision trees with a single and aggregated result. Using multiple trees in the random forest reduces the chances of

overfitting. And they are complex to understand. A decision tree is easy to read and understand whereas random forest is more complicated to interpret. A single decision tree is not accurate in predicting the results but is fast to implement. More trees will give a more robust model and prevents overfitting. In the forest, we need to generate process and analyze each and every tree. Therefore this process is a slow process and can sometimes take hours or even days.

**Mathematics based questions 14. Explain Bagging and Boosting in training random forest tree.** The Ensemble learning helps improve machine learning results by combining several models. This approach allows the production of better predictive performance compared to a single

model. Basic idea is to learn a set of classifiers (experts) and to allow them to vote. **Bagging** and **Boosting** are two types of **Ensemble Learning**. These two decrease the variance of a single estimate as they combine several estimates from different models. So the result may be a model with higher stability. Let‘s understand these two terms in a glimpse.

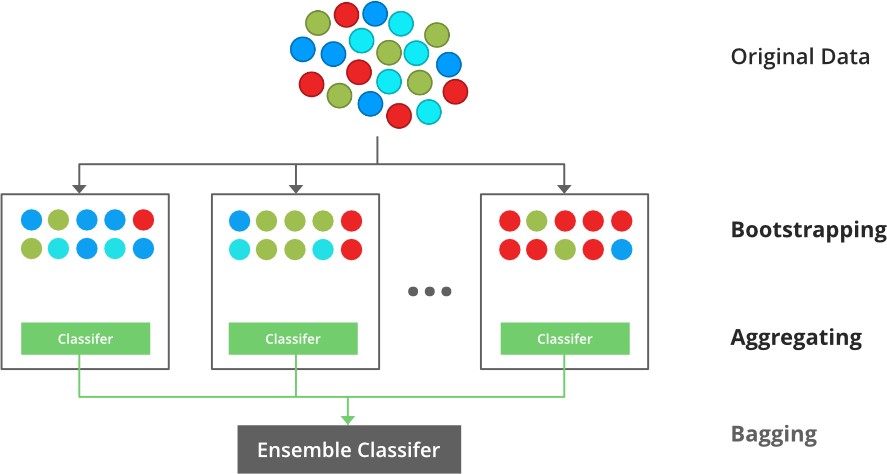
1. **Bagging**: It is a homogeneous weak learners‘ model that learns from each other independently in parallel and combines them for determining the model average.
2. **Boosting**: It is also a homogeneous weak learners‘ model but works differently from Bagging. In this model, learners learn sequentially and adaptively to improve model predictions of a learning algorithm.

**Bagging: B**ootstrap **Agg**regating, also known as bagging, is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression. It decreases the variance and helps to avoid [overfitting.](https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/) It is usually applied to decision tree methods. Bagging is a special case of the model averaging approach.

#### Description of the Technique

Suppose a set D of d tuples, at each iteration i, a training set Di of d tuples is sampled with replacement from D (i.e., bootstrap). Then a classifier model Mi is learned for each training set D < i. Each classifier Mi returns its class prediction. The bagged classifier M\* counts the votes and assigns the class with the most votes to X (unknown sample).

#### Implementation Steps of Bagging

* **Step 1:** Multiple subsets are created from the original data set with equal tuples, selecting observations with replacement.
* **Step 2:** A base model is created on each of these subsets.
* **Step 3:** Each model is learned in parallel from each training set and independent of each other.
* **Step 4:** The final predictions are determined by combining the predictions from all the models.

An illustration for the concept of bootstrap aggregating (Bagging)

#### Example of Bagging

The Random Forest model uses Bagging, where decision tree models with higher variance are present. It makes random feature selection to grow trees. Several random trees make a Random Forest.

#### Boosting

Boosting is an ensemble modelling technique that attempts to build a strong classifier from the number of weak classifiers. It is done by building a model by using weak models in series. Firstly, a model is built from the training data. Then the second model is built which tries to correct the errors present in the first model. This procedure is continued and models are added until either the complete training data set is predicted correctly or the maximum number of models is added.

#### Boosting Algorithms

There are several boosting algorithms. The original ones, proposed by **Robert Schapire** and **Yoav Freund** were not adaptive and could not take full advantage of the weak learners. Schapire and Freund then developed [AdaBoost,](https://www.geeksforgeeks.org/implementing-the-adaboost-algorithm-from-scratch/) an adaptive boosting algorithm that won the prestigious Gödel Prize. AdaBoost was the first really successful boosting algorithm developed for the purpose of binary classification. AdaBoost is short for Adaptive Boosting and is a very popular boosting technique that combines multiple ―weak classifiers‖ into a single ―strong classifier‖.

**Algorithm:**

1. *Initialise the dataset and assign equal weight to each of the data point.*
2. *Provide this as input to the model and identify the wrongly classified data points.*

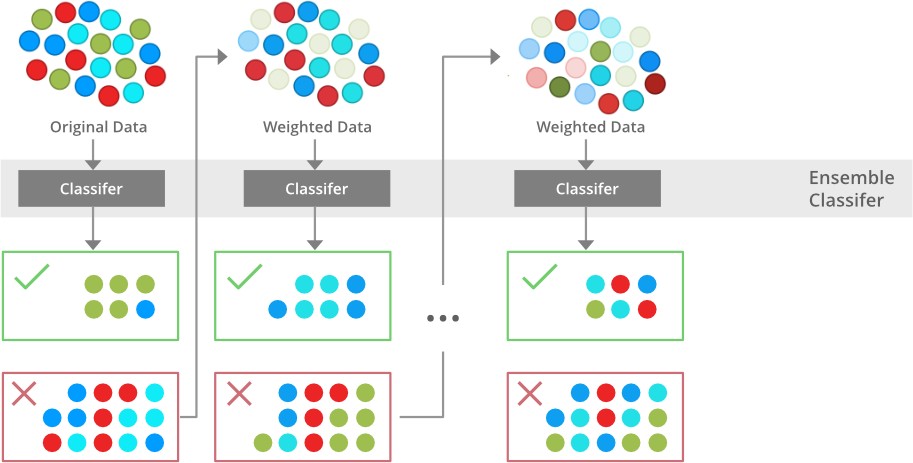
### Increase the weight of the wrongly classified data points.

1. *if (got required results) Goto step 5*

### Else

*Goto step 2*

### End



An illustration presenting the intuition behind the boosting algorithm, consisting of the parallel learners and weighted dataset

#### Similarities between Bagging and Boosting

Bagging and Boosting, both being the commonly used methods, have a universal similarity of being classified as ensemble methods. Here we will explain the similarities between them.

* Both are ensemble methods to get N learners from 1 learner.
* Both generate several training data sets by random sampling.
* Both make the final decision by averaging the N learners (or taking the majority of them

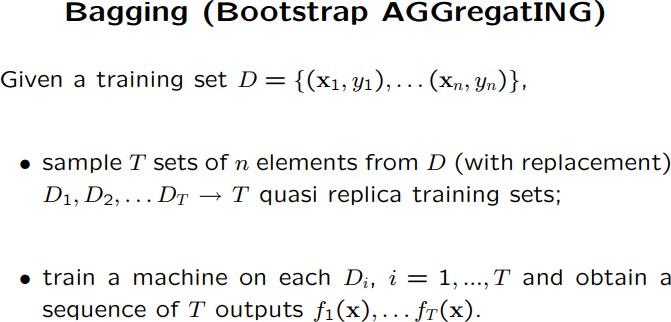
i.e Majority Voting).

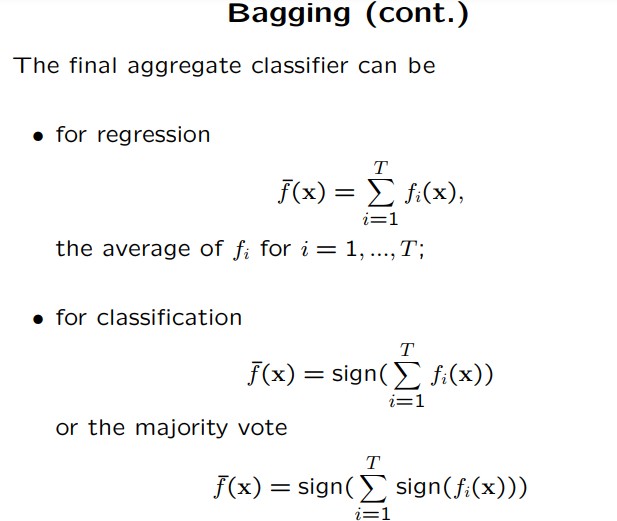
* Both are good at reducing variance and provide higher stability.

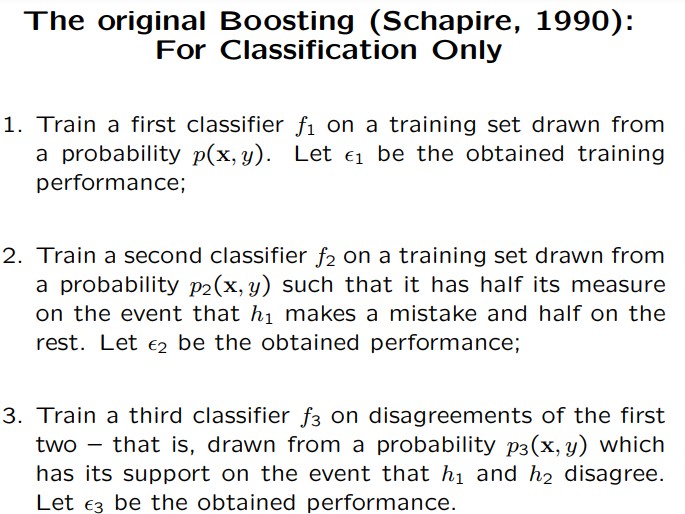
#### Differences between Bagging and Boosting

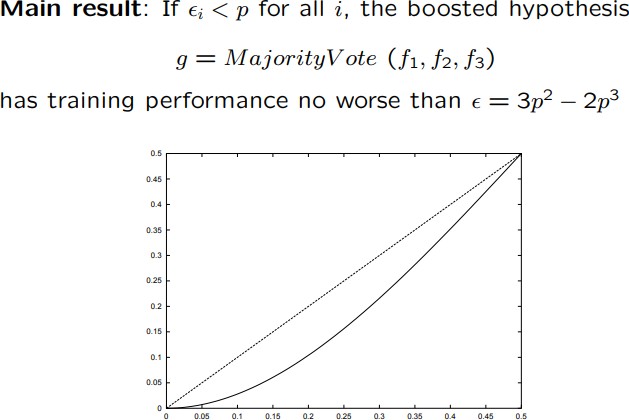
|  |  |  |
| --- | --- | --- |
| **SN** | **Bagging** | **Boosting** |
| 1. | The simplest way of combining predictions  that belongs to the same type. | A way of combining predictions that  belong to the different types. |
| 2. | Aim to decrease variance, not bias. | Aim to decrease bias, not variance. |
| 3. | Each model receives equal weight. | Models are weighted according to  their performance. |
| 4. | Each model is built independently. | New models are influenced  by the performance of previously built models. |

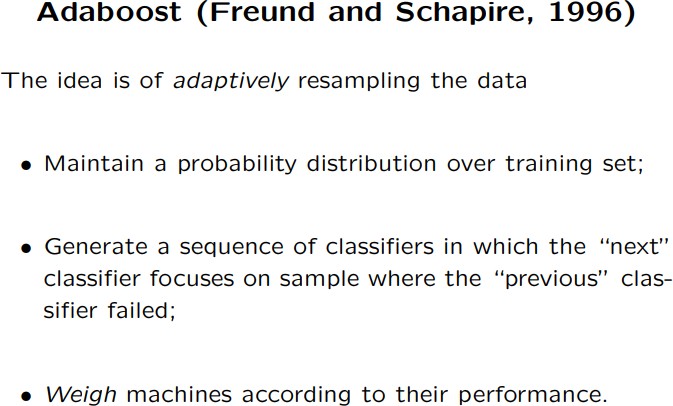
|  |  |  |
| --- | --- | --- |
| 5. | Different training data subsets are randomly  drawn with replacement from the entire training dataset. | Every new subset contains the  elements that were misclassified by previous models. |
| 6. | Bagging tries to solve the over-fitting  problem. | Boosting tries to reduce bias. |
| 7. | If the classifier is unstable (high variance),  then apply bagging. | If the classifier is stable and simple  (high bias) the apply boosting. |
| 8. | Example: The Random forest model uses  Bagging. | Example: The AdaBoost uses Boosting  techniques |

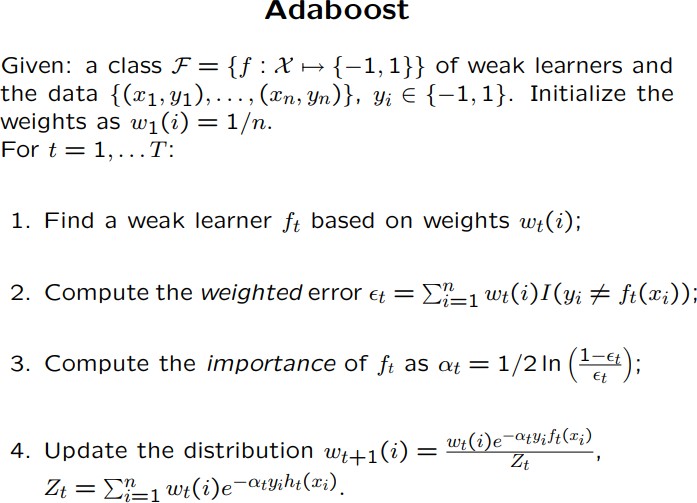


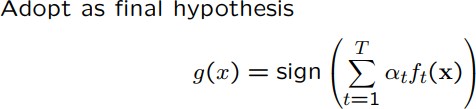












**15. Which is the best, Bagging or Boosting?**

* There‘s not an outright winner; it depends on the data, the simulation and the circumstances.

Bagging and Boosting decrease the variance of your single estimate as they combine several estimates from different models. So the result may be a model with **higher stability**.

* If the problem is that the single model gets a very low performance, Bagging will rarely get a **better bias**. However, Boosting could generate a combined model with lower errors as it optimises the advantages and reduces pitfalls of the single model.
* By contrast, if the difficulty of the single model is **over-fitting**, then Bagging is the best option. Boosting for its part doesn‘t help to avoid over-fitting; in fact, this technique is faced with this problem itself. Thus, Bagging is effective more often than Boosting.

**16. What are the main advantages of using a random forest versus a single decision tree?**

In an ideal world, we'd like to reduce both bias-related and variance-related errors. This issue

is well-addressed by random forests. A random forest is nothing more than a series of decision trees with their findings combined into a single final result. They are so powerful because of their capability to reduce overfitting without massively increasing error due to

bias. Random forests, on the other hand, are a powerful modelling tool that is far more resilient than a single decision tree. They combine numerous decision trees to reduce overfitting and bias-related inaccuracy, and hence produce usable results.