**Diabetes Dataset - Naive Bayes** simple, effective and commonly-used, machine learning classifier. It is a probabilistic. Naive Bayes classifiers have been especially popular for text classification.

**Applications:** **Real time Prediction, Multi class Prediction:** we can predict the probability of multiple classes of target variable.

**Text classification/ Spam Filtering/ Sentiment Analysis:** have higher success rate (in social media analysis, to identify positive and negative customer sentiments)

**Backtracking Algorithm:** Backtracking is finding the solution of a problem whereby the solution depends on the previous steps taken.

In backtracking, we first take a step and then we see if this step taken is correct or not i.e., whether it will give a correct answer or not. And if it doesn’t, then we just come back and change our first step. This is accomplished by recursion. Thus, in backtracking, we first start with a partial sub-solution of the problem (which may or may not lead us to the solution) and then check if we can proceed further with this sub-solution or not. If not, then we just come back and change it.

Thus, the general steps of backtracking are:

• start with a sub-solution

• check if this sub-solution will lead to the solution or not

• If not, then come back and change the sub-solution and continue again

**Hill Climbing** is a technique to solve certain optimization problems. In this technique, we start with a suboptimal solution and the solution is improved repeatedly until some condition is maximized.

The idea of starting with a sub-optimal solution is compared to starting from the base of the hill, improving the solution is compared to walking up the hill, and finally maximizing some condition is compared to reaching the top of the hill.

Hence, the hill climbing technique can be considered as the following phases −

● Constructing a sub-optimal solution obeying the constraints of the problem

● Improving the solution step-by-step

● Improving the solution until no more improvement is possible

**Binary Search:** also known as half-interval search, logarithmic search, is a search algorithm that finds the position of a target value within a sorted array. Binary search compares the target value to the middle element of the array. If they are not equal, the half in which the target cannot lie is eliminated and the search continues on the remaining half, again taking the middle element to compare to the target value, and repeating this until the target value is found. If the search ends with the remaining half being empty, the target is not in the array.

**Worst case time complexity – O(log n)**

**Space – O(1)**

**Graph traversal** means visiting every vertex and edge exactly once in a well-defined order. While using certain graph algorithms, you must ensure that each vertex of the graph is visited exactly once. The order in which the vertices are visited are important and may depend upon the algorithm.

During a traversal, it is important that you track which vertices have been visited. The most common way of tracking vertices is to mark them.

**BFS – Breadth First Traversal,** most commonly used approach. BFS is a traversing algorithm where you should start traversing from a selected node (source or starting node) and traverse the graph layer-wise thus exploring the neighbour nodes. You must then move towards the next-level neighbour nodes.

As the name BFS suggests, you are required to traverse the graph breadthwise as follows:

1. First move horizontally and visit all the nodes of the current layer

2. Move to the next layer

The distance between the nodes in layer 1 is comparatively lesser than the distance between the nodes in layer 2. Therefore, in BFS, you must traverse all the nodes in layer 1 before you move to the nodes in layer 2.

***Trip History Analysis - Recursive partitioning,* or “*classification and regression trees****,*” is a recursive partitioning method which is a fundamental tool in data mining. It is a prediction method often used with dichotomous outcomes that avoids the assumptions of linearity. This technique creates prediction rules by repeatedly dividing the sample into subgroups, with each subdivision being formed by separating the sample on the value of one of the predictor variables. The end result is a set of branching questions that forms a treelike structure in which each final branch provides a yes/no prediction of the outcome. rpart is for modelling decision trees.

Step 1: grow the trees; Step 2: Examine the result