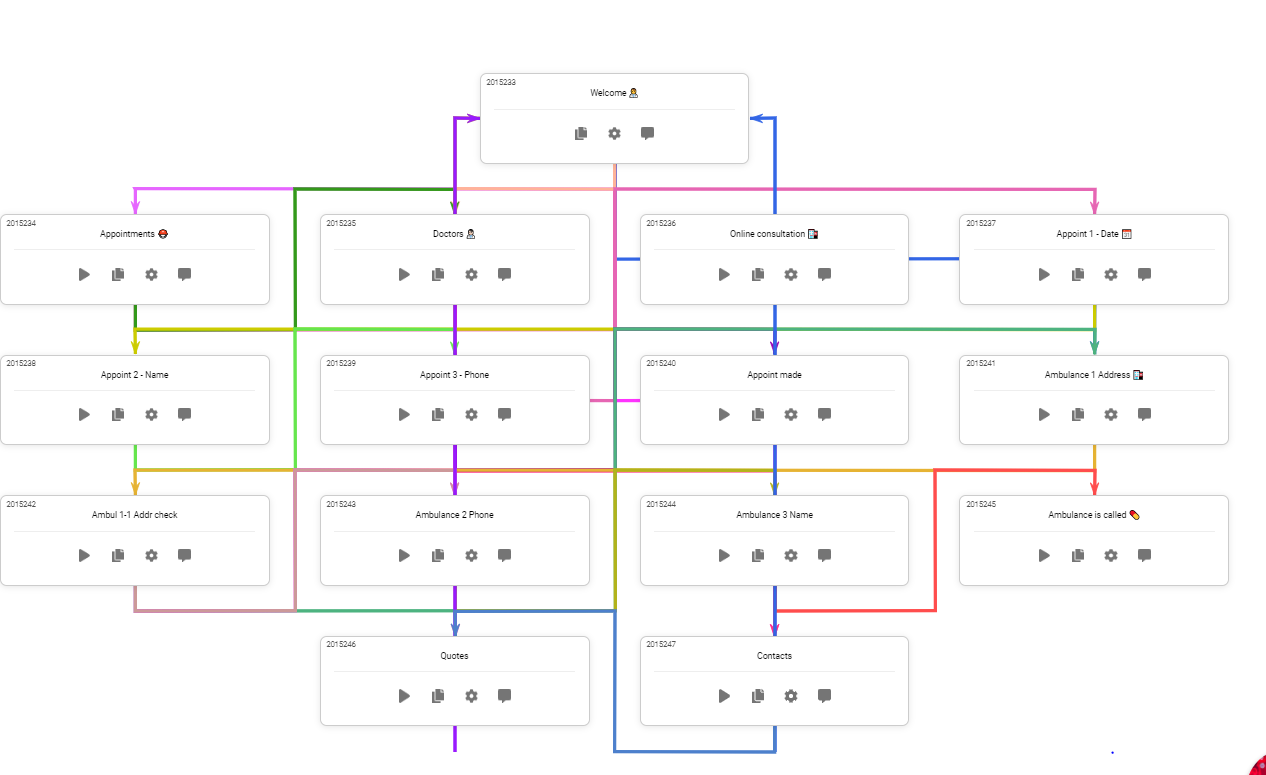
A tree can be “learned” by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions. The construction of decision tree classifier does not require any domain knowledge or parameter setting, and therefore is appropriate for exploratory knowledge discovery. Decision trees can handle high dimensional data. In general decision tree classifier has good accuracy. Decision tree induction is a typical inductive approach to learn knowledge on classification.

Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. An instance is classified by starting at the root node of the tree,testing the attribute specified by this node,then moving down the tree branch corresponding to the value of the attribute as shown in the above figure.This process is then repeated for the subtree rooted at the new node.



There are 5 types of plugins available:

* **eBay** - for a quick search of goods on eBay.
* **Giphy** - a service for searching gifs or stickers.
* **Weather** - a plugin that allows you to get a weather forecast for any city you want.
* **Calendar** - access to your Google calendar. Authorization required.
* **Jira** - using Jira via bot. Authorization required. Your domain must start with “[https://”](https://xn--ivg/).
* **Trello** - with this plugin, your users will be able to create, view, edit or remove the Boards that they own or have access to, Lists, Activities, Cards and Notifications.
* Let NN = number of training examples, kk = number of features, and dd = depth of the decision tree.
* A decision tree would calculate a quality function based on each split of the data, and it does this for each feature in every node that is not a leaf node. This happens as long as there are levels (depth) to continue to. In the best case of a balanced tree the depth would be in O(logN)O(log⁡N), but the decision tree does locally optimal splits without caring much about balancing. This means that the worst case of depth being in O(N)O(N) is possible - basically when each split simply splits data in 1 and n-1 examples, where n is the number of examples of the current node.
* So to conclude, the time complexity for decision trees is inO(Nkd)O(Nkd). This means that it’s actually somewhere in between being in O(NklogN)O(Nklog⁡N) and O(N2k)O(N2k).
* The uncertainty here is due to the non-deterministic way in which decision trees are built, always splitting data based on locally optimal thresholds with close to no consideration for overall balance. Keep in mind building a globally optimal decision tree is an NP-hard problem.
* he depth of a tree is O(logn), where n is the number of rows of data and the tree is assumed to be relatively balanced. (Try proving that to yourself. If you have 8 data points you have at max 8 leafs, and assuming a balanced binary tree that is a height of 3 or log\_2(8)).
* For each of the splits in the tree, you will need to test every feature for all values to determine the value split that minimizes the loss function.
* Assuming number of features is m, the run time is O(mnlogn).
* Assuming that each node of the decision tree requires O(1) to calculate, the complexity of the tree is bounded by its depth, e.g, longest route from the root to a leaf node.