FORM 2
THE PATENTS ACT 1970
(39 of 1970)
&
The Patents [Amendment] Rules, 2006
COMPLETE SPECIFICATION
(See section 10 and rule 13)

1. TITLE OF THE INVENTION

A FAST Algorithm for Constructing Decision Trees

2. APPLICANT

<table>
<thead>
<tr>
<th>NAME</th>
<th>Infosys Technologies Limited</th>
</tr>
</thead>
<tbody>
<tr>
<td>NATIONALITY</td>
<td>IN</td>
</tr>
<tr>
<td>ADDRESS</td>
<td>Plot No. 44, Electronics City, Hosur Road Bangalore, Karnataka - 560 100, India</td>
</tr>
</tbody>
</table>

3. PREAMBLE TO THE DESCRIPTION

COMPLETE

The following specification particularly describes the invention and the manner in which it is to be performed.
TECHNICAL FIELD

The present invention relates to a method used in data mining for constructing a decision tree(s), and more particularly, to the method used in an algorithm for constructing the decision trees quickly, using a prefix tree(s).

BACKGROUND

Data Mining is a process of autonomously extracting an useful information or a knowledge ("actionable assets") or a patterns from a large data stores (a dataset or a databases). The process designed in data mining explores the dataset in search of the consistent patterns and/or a systematic relationship(s) between the dataset variables or attribute. One goal of the data mining is prediction based on the analysis of the dataset. Classification is one of the data mining techniques used to prediction of the dataset. For example, the classification may be used to predict whether the weather on a particular day will be "sunny", "rainy" or "cloudy". The classification is one of the popular techniques used in the data mining environment because of its utility in many areas like fraud detection, business intelligence, and, medical target marketing etc. Few of the popular classification techniques used in the data mining includes a decision trees, a neural networks, a support vector machines, and a Bayesian network based classifiers.

Among the few popular classification techniques disclosed above, the decision tress are attractive area in the data mining environment because of their intuitive representation/ structure, which is very easy to understand/ interpret for humans. Hence the decision trees are not only treated as the classification tool, but also as an important analysis tool for understanding an attribute-attribute relationship among the large dataset. The decision trees may be found in many open-source or commercial data mining toolkits like a Weka, a SAS, a SPSS data Miner and MLC++ etc.
Much of the earlier research on construction of the decision trees was primarily aimed at developing good heuristic techniques, which may generate the high quality decision trees. The heuristic technique approach used in developing the high quality decision trees are highlighted in a C4.5, an ID3, a CART and etc. Such approaches were proposed to use different splitting criteria, while building the decision trees and did not consider the efficiency aspect of the decision trees construction.

Thus there is a need for a novel technique to devise a method, which helps in constructing or building the decision trees efficiently and also is not restricted to a particular splitting criterion.

**SUMMARY OF THE INVENTION**

In one embodiment of the present technique, a method to construct a decision tree from a dataset(s) is disclosed. The method includes building a prefix tree to store the dataset. The prefix tree is build using a predictor attribute(s) value(s) of a set of records of the dataset. Further, the frequencies of the predictor attributes’ values are counted from the prefix tree and stored in a frequency table. The method further comprises determining a best split condition based on the frequencies of the predictor attributes’ values stored in the frequency table. On determining the best split condition, one or more portion of the prefix tree is split to generate at least one new prefix tree(s). The new prefix tree and the retained portion of the prefix tree are used to construct a decision tree. The decision tree created may comprise a decision tree node and at least one of a decision subtree(s) at the decision tree node. The construction of at least one of the decision subtrees occurs in a recursive process by using at least one of the new prefix trees or retained portion of the prefix tree.

In another embodiment of the present technique, the method to build to prefix tree may be initiated after the set of records of the dataset stored in a relational database format is converted into a set of sequences in a sequential database format.

In yet another embodiment of the present technique, the method of splitting a prefix tree(s) into at least one or more new prefix tree(s) is disclosed. The method
includes a process of traversing a path of the prefix tree in a sequential order to locate a prefix tree node comprising a predictor attributes’ value(s) matching a best split condition. Further, adding a new path from a new prefix tree root node of a new prefix tree. The new path added from the new prefix tree root node may be the path selected from the prefix tree while traversing the prefix tree till the best split condition is located. The method further comprises removing a prefix subtree(s) of the prefix tree in entirety from the prefix tree node of the prefix tree where the best split condition is located. Further, appending the prefix subtrees to the end of the path of the new prefix tree.

BRIEF DESCRIPTION OF THE DRAWINGS

The above mentioned features as well other features, aspects, and advantages of the present invention will become better understood when the following detailed description is read with reference to the accompanying drawings in which like characters represent like parts throughout the drawings, wherein:

FIG. 1 is a flow diagram illustrating a method to construct a decision tree from a dataset, according to one possible embodiment of the present technique;

FIG. 2 is a flow diagram of a method illustrating the functionality of a FASTDT algorithm used to construct the decision tree, according to one embodiment of the present technique;

FIG. 3 is a flow diagram illustrates the steps involved in constructing the decision tree, according to one embodiment of the present technique;

FIG. 4 is a flow diagram illustrating a method to split a prefix tree, in one embodiment of the present technique;

FIG. 5 is an exemplary example illustrating a prefix tree node, according to one embodiment of the present technique;
FIG. 6 illustrates building of the prefix tree from the prefix tree node using a path of a first sequences in the exemplary example, according to one embodiment of the present technique;

FIG. 7 illustrates building of the prefix tree from the prefix tree node using a path of a second sequences in the exemplary example, according to one embodiment of the present technique;

FIG. 8 illustrates a portion of the prefix tree built using the path of the first sequences and the second sequences in the exemplary example, according to one embodiment of the present technique;

FIG. 9 illustrates building of the prefix tree using a path of a third sequences in the exemplary example, according to one embodiment of the present technique;

FIG. 10 illustrates the portion of the prefix tree built using the path of the first sequences, the second sequences, and the third sequences in the exemplary example, according to one embodiment of the present technique;

FIG. 11 illustrates the prefix tree built using the path of all sequences in the exemplary example, according to one embodiment of the present technique;

FIG. 12 illustrates a portion of a decision tree constructed using a best split attribute in the exemplary example, according to one embodiment of the present technique;

FIG. 13 illustrates a new prefix tree root from which a new prefix tree is built after splitting the prefix tree in the exemplary example, according to one embodiment of the present technique;

FIG. 14 illustrates the splitting process of the prefix tree node where the best split condition is located in the exemplary example, according to one embodiment of the present technique;
FIG. 15 illustrates a portion of the prefix tree node where the prefix tree is split in the exemplary example, according to one embodiment of the present technique;

FIG. 16 illustrates a portion of the new prefix tree built after splitting the prefix tree in the exemplary example, according to one embodiment of the present technique;

FIG. 17 illustrates another portion of the new prefix tree built after splitting the prefix tree in the exemplary example, according to one embodiment of the present technique;

FIG. 18 illustrates another portion of the prefix tree node where the prefix tree is split in the exemplary example, according to one embodiment of the present technique;

FIG. 19 illustrates yet another portion of the new prefix tree built after splitting the prefix tree in the exemplary example, according to one embodiment of the present technique;

FIG. 20 illustrates a retaining portion of the prefix tree after splitting the prefix tree at the prefix tree node where the best split condition is located in the exemplary example, according to one embodiment of the present technique;

FIG. 21 illustrates the new prefix tree after splitting the prefix tree at the prefix tree node where the best split condition is located in the exemplary example, according to one embodiment of the present technique;

FIG. 22 illustrates further construction of the decision tree using the retained prefix tree and the new prefix tree after splitting the prefix tree in the exemplary example, according to one embodiment of the present technique; and

FIG. 23 is a system illustrating a generalized computer network arrangement, in one embodiment of the present technique.
DETAILED DESCRIPTION

The following description is full and informative description of the best method and system presently contemplated for carrying out the present invention, which is known to the inventors at the time of filing the patent application. Of course, many modifications and adaptations will be apparent to those skilled in the relevant arts in view of the following description in view of the accompanying drawings and the appended claims. While the system and method described herein are provided with a certain degree of specificity, the present technique may be implemented with either greater or lesser specificity, depending on the needs of the user. Further, some of the features of the present technique may be used to advantage without the corresponding use of other features described in the following paragraphs. As such, the present description should be considered as merely illustrative of the principles of the present technique and not in limitation thereof, since the present technique is defined solely by the claims.

The present invention relates to a method to construct a decision tree from a dataset in a database in particular the method to quickly construct the decision tree using a prefix tree comprising the dataset.

The following description is presented to enable a person of ordinary skill in the art to make and use the invention and is provided in the context of the requirement for obtaining a patent. The description is the presently best contemplated method for carrying out the present invention. Various modifications to the preferred embodiment will be readily apparent to those skilled in the art and the generic principles of the present invention may be applied to other embodiments, and some features of the present invention may be applied without the corresponding use of other features. Accordingly, the present invention is not intended to be limited to the embodiment shown but is to be accorded the widest scope consistent with the principles and features described herein.

Referring to the figures, FIG. 1 is a flow diagram depicting a method to construct a decision tree from a dataset, according to one possible embodiment of the
present technique. The method comprises a multiple number of steps to construct the decision tree. The input to the method being the dataset in the relation database format and the output to the method being the decision tree.

The method comprising: 1) usage of the dataset in a relation database format (block 105), 2) conversion of the dataset in the relation database format to a sequential database format (block 110), 3) usage of the dataset in the sequential database format (block 115), 4) build a prefix tree by using the dataset in sequential database format (block 120), 5) usage of the built prefix tree (block 125), 6) usage of a Fast Decision Tree (herein referred as "FASTDT") algorithm to construct the decision tree from the prefix tree (block 130), and 7) the decision tree constructed after using the FASTDT algorithm (block 135). Each of the steps will be explained in greater extent in the subsequent sections as follows.

In one embodiment of the present technique, the dataset is a set of records; (herein referred as "records") wherein each records may comprise a predictor attribute(s) or a class attribute(s) or combinations thereof. The class attributes further comprises a class label for each record. The predictor attributes also further comprises a predictor attribute values for each records. The records may further comprise a transaction identity (herein referred as "transaction id") specific to each record. In present technique a novel classification (herein referred as "FASTDT") algorithm is used to construct the decision tree. The FASTDT algorithm may use the records of the dataset to construct the decision tree, wherein the decision tree may result in an ideal distribution of the class label in terms of predictor attribute values. The constructed decision tree should be able to predict the class label of a new record given the predictor attribute values of the new record.

In step 105, the dataset used to construct the decision tree may be in a relation database format. The dataset in the relation database format may comprise all predictor attribute values for each record along with the class label.

In step 110, the dataset in the relation database format is converted into a sequential database format. The dataset in the relation database format may be
converted into the sequential database format on the basis of a primary predictor attribute value. The primary predictor attribute value is chosen from the predictor attribute values of each record. Upon conversion of the dataset into the sequential format the set of records are converted into a set of sequence(s) (herein referred as "sequences"). The dataset in each sequence may comprise either a primary predictor attribute value or a blank value. The blank value in each sequence may be a representation of the non-primary predictor attribute value or the predictor attribute value. The class labels of the dataset in the relation database may be retained as it is against each transaction id in the sequential database.

In one embodiment of the present technique, the dataset in the relation database format may even be converted into the sequential database format on the basis of the predictor attribute value or predictor attribute. The scope of the present technique should not be restrictive in light of the present embodiment.

In one embodiment of the present technique, different predictor attributes may comprise an identical name for the predictor attribute value. In such case, the name of the predictor attribute value may be either prefixed or suffixed with the name of the predictor attribute to differentiate the predictor attribute values from each other. The functionality of the data conversion algorithm remains the same without deviating from the scope of the present technique. Thus, the prefixing or suffixing the name of the predictor attribute to the predictor attribute value should not be limiting the scope of the present invention. Accordingly, the present invention is not intended to be limited to the embodiment shown but is to be accorded the widest scope consistent with the principles and features described herein.

In one embodiment of the present invention, a technique is employed for converting the dataset in the relational database format to the sequential database format. The technique employed is best known to the inventor at the time of filing the present invention. However, it will be apparent to one skilled in the art that any other technique may be employed for converting the dataset in the relation database format to the sequential database format without deviating from the scope of the present technique. In one embodiment of the present invention the technique used to convert
the dataset from relation database format to the sequential database format may be a data conversion algorithm. The data conversion algorithm used to convert the dataset in the relation database format to the sequential database format is as represented below.

Conversion-To-Sequential-Database (DB) {

1 Randomly impose an order among predictor attributes

2 Let the ordered predictor attributes be a-1,a-2,…,a-N

3 for-each binary predictor attribute a-i

4 select one attribute value of a-i as primary attribute value of a-i

5 Sequential Database SDB = {}

6 for-each record with transaction identifier r-i

7 create a sequence with identifier r-i in the SDB

8 for-each attribute a-j

9 if record r-i has primary predictor attribute value for attribute a-j

10 append a-j’s primary attribute value to the sequence r-i

11 return SDB

12 }

However, it will be apparent to the person skilled in the art that any other algorithm or method may be used to convert the dataset in the relation database format to the sequential database format without deviating from the scope of the present technique.
In step 115, the set of records converted into set of sequences is used for constructing the decision tree. However, it will be apparent to one skilled in the art that the technique used for converting the set of records to the set of sequences is optional. The dataset comprising the set of records in the relation database format may also be used for the construction of the decision tree without deviating from the scope of the present technique.

In step 120, the dataset in the form of sequences is used to build a prefix tree. The dataset used may be either a continuous dataset or a categorical dataset. Categorical dataset may have values, which are limited in numbers. The categorical dataset values may not be able to compare among each others and the values are placed in a proper order. Whereas the values of the continuous dataset are huge in numbers. The continuous dataset values may be compared among each others and the values may be placed in a proper order. Examples of a categorical dataset values may be city name, sex etc. whereas the continuous dataset values may be either salary, age etc. This description is based on categorical dataset values. The technique explained in the description may also applicable to the continuous dataset values, for which the continuous dataset values may be converted into categorical dataset values using any of the known techniques. Building the prefix tree starts from a prefix tree root node, where the initial value of the root node would be a null value. The prefix tree is built in sequential mode. The sequential mode is the order of occurrence of the primary predictor attribute value, while converting the set of records in the relation database format to the set of sequences in the sequential database format. The prefix tree is built by creating a path for each set of sequences and incrementing a class label frequencies at each prefix tree node in the path. The prefix tree may comprise the at least one node, wherein each node of the prefix tree comprises the respective predictor attribute value derived out of the set of sequences of the dataset while creating the path of the prefix tree. The prefix tree may also include at least one prefix subtree(s).

In one embodiment of the present invention, building the prefix tree is performed by using a prefix tree building algorithm. The prefix tree building algorithm used in the present invention is as represented below.
Build-Prefix-Tree (SDB) {

    PrefixTreeNode root; root.pav = null, root.classfrequencies = 0:0

    Let the ordered predictor attributes be a-1,a-2,...,a-N

    for-each sequence r-i in SDB

    PrefixTreeNode currentnode = root

    for-each primary predictor attribute value b-j in the order of occurrence
    in r-i

    increment frequency of class to which r-i belongs to in
    child.classfrequencies

    if there exists a child of currentnode with child.pav = b-j

        currentnode = child

    else

        PrefixTreeNode newnode;

        newnode.pav = b-j, newnode.classfrequencies = 0:0

        currentnode.insertchild(newnode)

        increment frequency of class to which r-i belongs to in
        newnode.classfrequencies

        currentnode = newnode

    return root

}
However, it is apparent to the person skilled in the art that any other known method may be used to build the prefix tree and the method employed in the present technique should not be restrictive from the scope of the present invention.

In step 125, completely build prefix tree is used to constructing the decision tree. The prefix tree may comprise the dataset stored in the form of sequences. The construction of the decision tree begins with the usage of the dataset stored in the prefix tree; hence there may not be any further reference or usage of the dataset in the sequential database format or the relation database format for the construction of the decision tree. The usage of the prefix tree for the construction of the decision tree may improve the time required in constructing the decision tree, as well as the memory and system processor requirement for managing the dataset in the construction of the decision tree.

In Step 130, the FASTDT algorithm is used to construct the decision tree using the built prefix tree. The FASTDT algorithm comprises the step of splitting the built prefix tree into at least one new prefix tree based on a best split condition. The best split condition may be the best predictor attribute value of the dataset stored in the prefix tree. The best split condition is determined by the method comprising a first traversing process through which a class label frequency of the predictor attribute values is counted from the prefix tree, storing the frequencies in a frequency table and determining the best predictor attribute based on the frequencies stored in the frequency table. The splitting of the prefix tree starts with a second traversing process of the prefix tree. During the second traversing process each node of the prefix tree is searched for the best split condition. The prefix tree may be split at the node where the best split condition is located while traversing and a new prefix tree is built out of the path of the prefix tree node where the best split condition is located or from the prefix sub trees where the best split condition is located in the prefix tree or combinations thereof. The new prefix tree is built from a new prefix tree root node after splitting the prefix tree. However, it will be apparent to one skilled in the art that the technique used in splitting the prefix tree may be any suitable one without deviating from the scope of the present technique.
In one embodiment of the present technique, construction of the decision tree begins from a decision tree root node. The best split condition is stored in the decision tree root node. The decision tree may comprise at least one decision tree subtree(s). The new prefix tree and the prefix tree after splitting is used to construct a decision tree subtrees. The decision tree constructing may be a recursive process. The recursive process of construction of the decision tree ends upon the corresponding prefix tree or new prefix used in the construction of the decision tree satisfies at least one or more set of predefined condition.

The set of predefined condition may include at least one of a frequencies of the class attribute at the prefix tree node in the path belonging to one class value or all of the predictor attributes has been selected as the best split condition or the frequency of the class attribute is null or combinations thereof. However, it will be apparent to one skilled in the art that the predefined condition used in stopping the construction of the decision tree may be any suitable one without deviating from the scope of the present technique.

In one embodiment of the present technique, the FASTDT algorithm used to construct the decision tree may be as represented below.

FastDT (DB) {

1    Build Prefix Tree out of DB

2    Compute Frequency Table by Traversing Prefix Tree

3    Find Best-Split Attribute

4    Initialize DTRoot with Best-Split Attribute

5    Cut Prefix Tree into multiple trees such that records with same value for Best-Split Attribute are stored in same tree

6    Build DTRoot’s subtrees recursively using the generated
multiple prefix trees

7 Return DTRoot which forms the root of the built Decision Tree

8 }

Convention used

DB = Database

Frequency Table = Stores frequencies of predictor attribute values

DTRoot = Root of the Decision Tree

Best-Split Attr = Class frequencies of records with same value for this attribute are skewed

In step 135, the decision tree is constructed in entirety after undergoing the recursive process. The constructed decision tree might be used as a model for making decision. The model or the constructed decision tree should be able to predict the class label of the new record given the predictor attribute values of the new record.

FIG. 2 is a flow diagram of the method illustrating the functionality of the FASTDT algorithm used to construct the decision tree, according to one embodiment of the present technique. The input to the method is the prefix tree and the output is the decision tree. The method comprising: 1) usage of the built prefix tree as the input in the algorithm (block 205), 2) counting a frequencies of the predictor attribute values (block 210), 4) storing the frequencies in a frequency table (block 215), 4) determining the best split condition (block 220), 5) usage of the built split condition to split the built prefix tree (block 225), 6) splitting the prefix tree (block 230), 7) new prefix tree matching the best split condition (block 230A), 8) prefix tree does not matching the best split condition (block 230B), 9) construct the decision tree (block 235), and 10) the decision tree constructed after using the FASTDT algorithm (block 240). Each of the steps will be explained in greater extent in the subsequent sections as follows.
In one embodiment of the present technique, in step 205, the built prefix tree is used as input in the construction of the decision tree. The built prefix tree may comprise at least one or more nodes, wherein the prefix tree originates from a root node with a null value. Each nodes of the prefix tree stores the predictor attribute value. The prefix tree may comprise at least one prefix sub tree. The prefix tree is built with a set of sequence in the order of occurrence of the predictor attribute value.

In step 210, according to one embodiment of the present technique, the class label frequency of the predictor attribute values is counted by performing a first traversing process of the prefix tree. The process of first traversing the prefix tree may be performed sequentially. While first traversing the prefix tree the class label frequency of occurrence of the predictor attribute value at each node is counted. The class label frequency gets added where the same predictor attribute value is located at two different nodes of the prefix tree during first traversing process. In one embodiment of the present technique a first traversing algorithm may be used to count the class label frequency of the predictor attribute value at each node of the prefix table. The first traversing algorithm used in the step 205 may be as represented below.

```c
Build-Frequency-Table (PrefixTreeNode root, FrequencyTable *frequency_table) {

1 A predictor attribute value a-i’s class frequencies can be accessed from

   frequency_table in the following way: frequency_table.PAV[a-i]

2 For example, if the class1 frequency is 2 and class2 frequency is 5 for a-i,

   then frequency_table.PAV[a-i] will be 2:5

3 When Build-Frequency-Table is called for the first time (ie with ‘root’ as argument),

   the frequencies in frequency_table would be 0:0

4 PrefixTreeNode currentnode = root

5 for-each child of currentnode
```
frequency_table[PAV[child.pav]] += child.classfrequencies

Build-Frequency-Table (child, frequency_table)

}

In one embodiment of the present technique, in step 215, the final class label frequency of the predictor attribute value counted at every node of the prefix tree is stored in the frequency table. The value deciphered out of the frequency table may be used as input in further step for determining the best split condition.

In step 220, the value of the frequency table is used to determine the best split condition, using which the prefix table may be split. The best split condition may be a best split predictor attribute value derived out of the frequencies stored in the frequency table. In one embodiment of the present invention, the best split condition may be determined by using a best split condition algorithm. The best split condition algorithm may be any know algorithm already known in the art. Few of the best split algorithms known the inventor during filing the present invention may include ID3, Gini Index, etc.

In one embodiment of the present technique, in step 225, the best split condition determined by the best split algorithm is used to split the built split tree.

In step 230, the splitting of the prefix tree is initiated as per one embodiment of the present invention. The splitting of the prefix tree starts with the second traversing process, wherein each of the prefix tree nodes is scanned or searched for the best split condition. The path till the node of the prefix tree where the best split condition is matched or located is selected. The selected path is copied or created as a new path from a new prefix root node. The prefix subtree from the node where the best split condition is located is removed in entirety and is appended to the new path in the new prefix tree. In one embodiment of the present technique, splitting the prefix tree is performed by the second traversing process at all nodes of the prefix tree. The node where the best split condition is matched is deleted after splitting the prefix tree into the new prefix tree. A class label frequency of the node in
the prefix tree where the best split condition is located while the second traversing process is incremented to all nodes in the new path of the new prefix tree. As well the class label frequency of the node where the best split condition is located is used to decrement the class label frequencies of all nodes in the path of the prefix tree.

The path of the prefix tree comprising all nodes where the best split condition is not matched is retained in the prefix tree. In one embodiment of the present technique the retained path of the prefix tree is performed after the second traversing process is accomplished in entirety.

In one embodiment of the present technique, in step 230A, the new prefix tree is created after splitting the prefix tree. The new prefix tree is created from the prefix tree after splitting the prefix tree while the second traversing process where the best split condition is matched. The numeral ‘0’ used in the FIG. 2, is for the reference to indicate that the new prefix tree is created out of the prefix tree by the splitting process of the nodes of the prefix tree where the best condition is located.

In one embodiment of the present technique, in step 230B, the path of the prefix tree where the best split condition is not matched while second traversing process during splitting the prefix tree is retained as the retained prefix tree. The numeral ‘1’ used in the FIG. 2 is for the reference to indicate that the path of the prefix tree is retained after splitting the prefix tree where the best condition is not located.

In one embodiment of the present technique the splitting process of the prefix tree is performed using a splitting algorithm. The splitting algorithm may be as represented below.

Split-PrefixTree (PrefixTreeNode root, PrefixTreeNode newroot, PAV bestsplit, PAV[] path) {

1     PAV stands for Primary Predictor Attribute Value

2     PAV[] stands for array of PAV;
array ‘path’ stores the sequence of PAVs of nodes visited to arrive at root

for-each child of root

    if child.pav same as bestsplit

        currentnode = newroot

    for-each ‘pav’ appearing in ‘path’ from the beginning

    if there-exists a currentnode.child whose primary predictor attribute value is ‘pav’

        currentnode.child.classfrequencies

    currentnode = child

else

    PrefixTreeNode newnode;

    newnode.pav = ‘pav’, newnode.classfrequencies = child.classfrequencies

    include newnode in currentnode’s children

    currentnode = newnode

move all subtrees of child to currentnode

subtract child.classfrequencies from classfrequencies of all its ancestor nodes

remove link from root to child and delete child

else

    Split-PrefixTree (child, newroot, bestsplit, path + child.pav)
The splitting algorithm disclosed below is purely an illustrative means, there may be many other features, which may be embedded into the algorithm to achieve the splitting functionality. The disclosed splitting algorithm should not be restrictive from the scope of the present technique.

In step 235, the process of constructing the decision tree is initiated, as per one embodiment of the present technique. The best split condition is stored at a decision tree root node. The decision tree may comprise at least one of a decision subtree(s). The construction of the decision tree starts from the decision tree root node, where the best split condition is stored. The construction of the decision subtree starts with the usage of the prefix tree and new prefix tree based on the best split condition. The input to the decision subtree may be the new prefix tree and the retained decision tree. The process of constructing the decision tree is a recursive process. The recursive process of construction of the decision tree ends upon the corresponding prefix tree or new prefix tree used in the construction of the decision tree satisfies at least one or more set of predefined condition.

In step 240, the decision tree is derived out of the prefix tree. The decision tree may be used as a model for making decision. The model or the constructed decision tree should be able to predict the class label of the new record given the predictor attribute values of the new record.

FIG. 3 is a flow diagram of the method illustrating the steps in constructing the decision tree, according to one embodiment of the present technique. The input to the method being the prefix tree and the output of the method being the decision tree. The constructed decision tree should be able to predict the class label of the new record given the predictor attribute values of the new record.

The method comprising: 1) constructing a decision tree root node (block 305), 2) usage of the best split condition in the decision tree construction (block 310), 3) storing the best split condition at the decision tree root node (block 315), 4) usage of the decision tree root node for further construction of a decision subtree (block
320), 5) usage of the prefix tree for the construction of a first decision subtree (block 325A), 6) usage of the new prefix tree for the construction of the second decision subtree (block 325B), 7) usage of the FASTDT algorithm in constructing the first decision subtree and a second decision subtree (block 330), 8) constructed first decision subtree (block 335A), 9) constructed second decision subtree (block 335B), 10) construction of the decision tree usage the first decision subtree and the second decision subtree from the root node of the decision tree (block 340), and 11) Fully build decision tree (block 345). Each of the steps will be explained in greater extent in the subsequent sections as follows.

In one embodiment of the present technique, in step 305, the construction of the decision tree begins from the decision tree root node. The decision tree root node may be null at the beginning of the construction of the decision tree.

In step 310, the decision tree root node is stored with the best split condition. The best split condition is derived after the first traversing process of the prefix tree and storing the frequencies of the predictor attribute value in the frequency table. The best split condition comprises the best predictor attribute value attained through running a best split algorithm on the frequency table.

In one embodiment of the present technique, in step 315 the best split condition determined by using the best split algorithm is stored in the decision tree root node.

In step 320, the construction of the decision subtree begins with the segregation of the predictor attribute value matching the best split condition, as per one embodiment of the present technique. The decision tree may comprise at least one decision subtree. In one embodiment of the present technique there may be a first decision subtree and a second decision subtree.

In step 325A, the prefix tree is used as an input in the FASTDT algorithm for constructing the first decision subtree. The prefix tree used in the construction of the first decision subtree may be the first portion of the prefix tree where the best split
condition is matched while traversing the prefix tree. The first portion of the prefix tree may be the new prefix tree obtained after splitting the prefix tree.

In step 325B, the prefix tree is used as an input in the FASTDT algorithm for constructing the second decision subtree. The prefix subtree used in the construction of the second decision subtree may be the second portion of the prefix tree where the best split condition is not matched while traversing the prefix tree. The second portion of the prefix tree may be the retained portion of the prefix tree after splitting.

In step 330, the prefix tree is used as the input for constructing the first decision subtree and the second decision subtree. The prefix tree may be split into the first portion and the second portion after running through the steps in the FASTDT algorithm. The first portion may be used for the construction of the first decision subtree. The second portion of the prefix tree may be used for the construction of the second decision subtree. All other functionality of FASTDT algorithm remains the same as briefed in the explanation part of FIG. 2.

In step 335A, the first decision subtree is constructed using the first portion of the prefix tree. The first decision subtree may comprise the entire predictor attribute matching the best split condition. In one embodiment of the present technique, the construction of the first decision subtree into many new subtrees may happens recursively, until at least one of the set of predefined conditions occur in the event of construction of the decision subtree.

In step 335B, the second decision subtree is constructed using the second portion of the prefix tree. The second decision subtree may comprise the entire predictor attribute not matching the best split condition. In one embodiment of the present technique, the construction of the second decision subtree into many new subtrees may happens recursively, until at least one of the set of predefined conditions occur in the event of construction of the decision subtree.

In one embodiment of the present technique, in step 340, the constructed first decision subtree and the second decision subtree may be used together to build
the decision tree in entirety. The step 340 of constructing the decision tree using the decision subtree may be recursive. The recursive process ends upon occurrence of at least one of the predefined conditions.

In step 345, the decision tree is constructed in entirety using the new prefix tree and the retained prefix tree. The decision tree may be readily used as the model to predict the class label of the new record given the predictor attributes values of the new record.

FIG. 4 is a flow diagram illustrating a method to split a prefix tree, in one embodiment of the present technique. The input to the method being the prefix tree and the output of the method being a new prefix tree and the retained prefix tree.

The method comprising: 1) usage of the prefix tree (block 405), 2) finding the node in the prefix tree which matches the best split condition (block 410), 3) decrementing a class label frequencies of all nodes in the path (block 415), 4) inputting the prefix or the new prefix tree if the best split condition is not located (block 420), 5) creating a new prefix tree root node (block 425), 6) replicating the nodes path in the new prefix tree (block 430), 7) Increment class label frequencies in the created path (block 435), and 8) deleting the node (block 440). Each of the steps will be explained in greater extent in the subsequent sections as follows.

In step 405, the prefix tree built using the set of sequences is used as input for splitting the prefix tree, in the process of constructing the decision tree. In one embodiment of the present invention, the built prefix tree may comprise several nodes, wherein the each node further comprises the predictor attribute values of the set of sequences. The prefix tree is built from a prefix tree root node, wherein the value of the root node may be a null value. The prefix tree may further comprise at least one prefix subtree(s). The building of the prefix tree may start with a path from the root node, wherein the path may further comprise at least one node. The prefix tree may have at least one path. The prefix subtree may be originated from any node located in any path of the prefix tree. In one embodiment of the present invention, the
prefix tree is subjected as input for splitting the prefix tree as represented by the reference numeral 402.

In step 410, the splitting of the prefix tree begins with the process of second traversing. In the second traversing process of the prefix tree each node of the prefix tree is examined for the presence of the best split condition. The best split condition may be the best predictor attribute value of the dataset stored in the prefix tree. In one embodiment of the present invention, the best split condition may be a best split predictor attribute value derived out of the frequencies stored in the frequency table. The outcome of the second traversing process may be a match of the node where the best split condition is matched, which is represented by a reference numeral 404. In the second traversing process if the best split condition is not matched, then the resulted prefix tree may be the prefix tree with retained portion (herein also referred as “retained prefix tree” or “the prefix tree”) as represented by reference numeral 406.

In step 415, while in the process of second traversing the prefix tree, if the best split condition is determined the class label frequency of all nodes in the path, till the node in the prefix tree where best split condition is matched is decremented with the node’s class label frequency.

In one embodiment of the present technique, in step 420, while in the process of second traversing the prefix tree, if the best split condition is not determined the second traversing process is stopped and the resultant prefix tree may be either a retained prefix tree.

After decrementing the class label frequency in the path till the node where the best split condition is located a new prefix tree root node is created as reflected in the step 425. In one embodiment of the present technique, the value of the root node is null.

In step 430, after creating the new prefix tree root node and decrementing the class label frequency of all nodes in the path of the prefix tree a new path is created from the new prefix tree root node as represented by the reference numeral 408 and 412. The path till the node of the prefix tree where the best split condition is
matched or located is selected. The selected path is copied or created or replicated as a new path from a new prefix root node. The prefix subtrees from the node where the best split condition is located are also removed in entirety and are appended to the new path in the new prefix tree.

In step 435, after creating the new prefix tree after replicating the new path from the new prefix root node the class label frequency of the node in the prefix tree where the best split condition is located while the second traversing process is incremented to all nodes in the new path of the new prefix tree. The process is incrementing the class label frequency of all node using the class label in the new path with the nodes’ class label frequency of the prefix tree where the best split condition is located is as represented by the reference numeral 414.

In step 440, as per one embodiment of the present technique, the node where the best split condition is matched while traversing the prefix tree is deleted after splitting the prefix tree into the new prefix tree as represented by a reference numeral 416.

The process of splitting the prefix tree is continued till all the nodes in the prefix tree are covered in entirety looking for the best split condition as represented by reference numeral 418. In the process of traversing if the best split condition is not located the resultant prefix tree may be a new prefix tree or a retained prefix tree.

While the present invention has been related in terms of the foregoing embodiments, those skilled in the art will recognize that the invention is not limited to the embodiments depicted. The present invention may be practiced with modification and alteration within the spirit and scope of the appended claims. Thus, the description is to be regarded as illustrative instead of restrictive on the present invention.

In one of the exemplary embodiment, the present technique may be illustrated by representing the steps involved for constructing a decision tree using the sample dataset comprising only few set of records.
The dataset used to illustrate the exemplary example may be as represented below. The dataset may be referred with a reference numeral Table 1.

<table>
<thead>
<tr>
<th>Rid</th>
<th>Age</th>
<th>Car type</th>
<th>Sex</th>
<th>Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Old</td>
<td>Sports</td>
<td>Male</td>
<td>High</td>
</tr>
<tr>
<td>2</td>
<td>Young</td>
<td>Family</td>
<td>Female</td>
<td>Low</td>
</tr>
<tr>
<td>3</td>
<td>Young</td>
<td>Family</td>
<td>Male</td>
<td>High</td>
</tr>
<tr>
<td>4</td>
<td>Old</td>
<td>Sports</td>
<td>Male</td>
<td>Low</td>
</tr>
<tr>
<td>5</td>
<td>Old</td>
<td>Family</td>
<td>Male</td>
<td>Low</td>
</tr>
<tr>
<td>6</td>
<td>Young</td>
<td>Sports</td>
<td>Female</td>
<td>High</td>
</tr>
<tr>
<td>7</td>
<td>Young</td>
<td>Family</td>
<td>Male</td>
<td>Low</td>
</tr>
</tbody>
</table>

**TABLE 1**

In the exemplary example the dataset represented in table 1 may in the relation database format. Table 1 shows an exemplary example dataset of a transaction from an insurance domain. Only three attributes and seven transactions are considered in the example dataset and each attribute may be represented with binary attribute. The three attributes are known a predictor attributes. The three predictor attributes are a Car Type, an Age, and a Sex. Each predictor attributes are considered to have a binary attribute value. The binary attribute value may be referred as a predictor attribute values. The predictor attribute values of the Car type predictor attribute are sports and family. The predictor attribute values of the age predictor attribute are young and old. The predictor attribute values of the sex predictor attribute are male and female. The risk attribute is known as class attribute. The class attribute Risk highlights the class to which a particular transaction belongs to. The value of the class attribute is also considered to be binary. The value of the class attribute may be referred as a class label. The class label may be either high or low. The assumption that the class attributes takes only two class label makes simpler the description of algorithm presented in the subsequent sections to follow.
However, the techniques in the algorithm to be explained in subsequent section to follow may be even applied to a multi class label without any significant modifications. The description should not be restrictive from the scope of the present invention. The dataset contains set of records comprising the predictor attribute values and the respective class label values. Each of the records may be referred with a record identifier or a transaction identity (herein referred as “transaction id”) specific to each record.

In one embodiment of the present invention, the dataset in the relation database format is converted into a sequential database format. The dataset in the relation database format may be converted into the sequential database format on the basis of a primary predictor attribute value. The primary predictor attribute values considered in the exemplary example are the family, the old, and the male. The data conversion algorithm may be used to convert the set of records in the dataset to a set of sequences (also known as “sequences”). The conversion of the set of records into set of sequences happens based on the primary predictor values. The absence of few predictor attribute value in the sequences would mean the presence of the second value of that particular predictor attribute value. It is assumed that the missing predictor attribute value in the sequence makes the representation easy to understand. Each of the sequences may have a specific sequence identifier (herein referred as “Sid”). The dataset in the sequential database format after conversion is as represented in the reference numeral table “Table 2”. The class label is copied as it was in the relation database without any modification. The data conversion algorithm is explained in the above section.

<table>
<thead>
<tr>
<th>Sid</th>
<th>Sequence</th>
<th>Class attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(old, male)</td>
<td>high</td>
</tr>
<tr>
<td>2</td>
<td>(family)</td>
<td>low</td>
</tr>
<tr>
<td>3</td>
<td>(family, male)</td>
<td>high</td>
</tr>
<tr>
<td></td>
<td>(old, male)</td>
<td>low</td>
</tr>
<tr>
<td>---</td>
<td>-------------</td>
<td>--------------</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>(old, family, male)</td>
<td>low</td>
</tr>
<tr>
<td>6</td>
<td>()</td>
<td>high</td>
</tr>
<tr>
<td>7</td>
<td>(family, male)</td>
<td>low</td>
</tr>
</tbody>
</table>

**TABLE 2**

In one exemplary embodiment of the present invention, after conversion of the dataset into the set of sequences the process of building the prefix tree is initiated. The building of the prefix tree starts with a prefix tree node 505.

**FIG. 5** is an exemplary example illustrating the prefix tree node 505, according to one embodiment of the present technique. The value of the prefix tree root node 505 is null. In general the value of the nodes may refer to the frequency of the class label at that particular node. In one exemplary embodiment of the present technique, the frequency of the class label is represented in a High: Low ratio. In the exemplary example, the class label frequency at the root node of the prefix tree may be 0:0.

**FIG. 6** illustrates building of the prefix tree from the prefix tree node using a path of a first sequences in the exemplary example, according to one embodiment of the present technique. Building the path of the prefix tree starts with the first sequence as listed under the sequence identifier 1 (table 2). The class label frequency of the prefix tree root 505 may be 1:0 as per the class label of the sequence id “1” (table 2). From the root node a new node for the old Primary Predictor Attribute Value (herein referred as “PAV”) 605 may be created. The class label frequency of the node old PAV 605 will be added to the node. The class label frequency of the node old PAV 605 may be 1:0. Again one more node may be created from the old PAV 605 named a node male PAV 610. Again the class label frequency of the male PAV 610 will be added to the node. The class label frequency of the node male PAV 610 may be 1:0. Thus all the PAV listed in the sequence id “1” will be added to the path from the prefix tree root node 505.
FIG. 7 illustrates building of the prefix tree from the prefix tree node using a path of a second sequence in the exemplary example, according to one embodiment of the present technique. The PAV listed in the sequence id “2” (table 2) will be appended from the prefix tree root node 505. The class label frequency of the root node may be 0:1 as per the PAV listed in the sequence id “2”. A node family PAV 705 gets added in the path from the prefix tree root node 505. The class label frequency of the node family PAV 705 will be 0:1.

FIG. 8 illustrates a portion of the prefix tree built using the path of the first sequences and the second sequences in the exemplary example, according to one embodiment of the present technique. The portion of the prefix tree is built by adding the prefix trees built using the sequence id “1” and sequence id “2”. The portion of the prefix tree comprises the prefix tree root node, wherein the class label frequency of the prefix tree is the summation of the prefix tree root node build using the sequence id “1” and sequence id “2”. Thus the prefix tree node class label frequency may be 1:1. The path of the prefix tree also gets appended along with the respective class label frequency. Thus the portion of the prefix tree comprises the node old PAV 605 with the class label frequency been 1:0, the node male PAV 610 with the class label frequency been 1:0, and the node family PAV 705 with the class label frequency been 0:1.

FIG. 9 illustrates building of the prefix tree using a path of a third sequences in the exemplary example, according to one embodiment of the present technique. The path of the prefix tree is build using the sequence id “3”. The prefix tree root node 505 at this prefix tree portion may be 1:0. The other two nodes will be created, the node family PAV node 905 will be build from the prefix tree root node 505 with the class label frequency of the node family PAV 905 may be 1:0. The node male PAV node 910 gets added to the path after the node family PAV 905 with the class label frequency of the node family PAV been 1:0.

FIG. 10 illustrates the portion of the prefix tree built using the path of the first sequences, the second sequences, and the third sequences in the exemplary example, according to one embodiment of the present technique. The path of the
prefix tree created using the sequence id “3” gets appended to the portion of the prefix tree build at FIG 8, as mentioned above. The portion of the prefix tree comprising the path with the node family PAV will add up with one more new node named male PAV 1010, wherein the value of the node male PAV 1010 will be 1:0. The class label frequency of the family node 1005 gets appended to 1:1. Similarly, the prefix tree root node class label frequency may also be appended to 2:1.

Similarly, using all the sequences as listed in the table 2, the portion of the prefix tree is build. Once the prefix tree is build using all the sequences (table 2) the prefix tree is build in entirety. FIG. 11 illustrates the prefix tree built using the path of all sequences in the exemplary example, according to one embodiment of the present technique. The prefix tree comprises the root node 505 with the class label frequency been 3:4. The path starts in two direction, the first been the path comprising the node old PAV 1105 with the class label frequency 1:2, while the second been the path comprising the node family PAV 1125 with the class label frequency been 1:2.

The path in the first direction after the node old PAV 1105 further comprises the node family PAV with the class label frequency 0:1 and the node male PAV 1120 with the class label frequency been 1:1. The node family PAV 1110 further comprises the node male PAV with the class label frequency been 0:1.

The path in the second direction after the node family PAV 1125 further comprises the node male PAV 1130 with the class label frequency been 1:1.

Once the prefix tree is built in entirety, the prefix tree is first traversed to count the PAV class label frequency at each node of the prefix tree. The counted PAV at each node are stored in a class label frequency table. The class label frequency table is also listed below table 3.
TABLE 3

<table>
<thead>
<tr>
<th>Predictor attribute</th>
<th>Predictor Attribute Value: frequency</th>
<th>Predictor Attribute Value: frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td>old = 1:2</td>
<td>young = 2:2</td>
</tr>
<tr>
<td>Car type</td>
<td>family = 1:3</td>
<td>sports = 2:1</td>
</tr>
<tr>
<td>Sex</td>
<td>male = 2:3</td>
<td>female = 1:1</td>
</tr>
</tbody>
</table>

The first traversing algorithm may be used to count the class label frequency of the PAV at each node of the prefix table. Once the class label frequency table 3 is created, the best split condition is determined using the best split condition algorithm. The best split condition algorithm used may be any known algorithm known in the art. The best split condition may be as listed below in table 4. The best split condition may also be referred as the best split attribute. The best split attribute is the best PAV as represented in the table 4, which may be the car type predictor attribute. The car type predictor attribute has two PAV. The family PAV has the class label frequency 1:3 and the sports PAV have the class label frequency 2:1. The best split condition is derived based on the class label class label frequency of the PAV.

<table>
<thead>
<tr>
<th>Predictor attribute</th>
<th>Predictor Attribute Value: frequency</th>
<th>Predictor Attribute Value: frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Car type</td>
<td>family = 1:3</td>
<td>sports = 2:1</td>
</tr>
</tbody>
</table>

TABLE 4

The determined best split condition is stored in a decision tree root node. FIG. 12 illustrates a portion of a decision tree constructed using a best split attributes or best split condition in the exemplary example, according to one embodiment of the present technique. The best split condition is stored in the decision tree root node 1205. The decision tree may comprise at least one decision subtree. The first decision subtree 1210 may comprise all the family PAV wherein the second decision
subtree may comprise the sports PAV. The decision subtrees are constructed using the FASTDT algorithm as explained above. The construction process of the decision subtrees starts with the subsequent steps briefed below.

FIG. 13 illustrates a new prefix tree root from which a new prefix tree is built after splitting the prefix tree in the exemplary example, according to one embodiment of the present technique. The new prefix tree root node 1305 may have the class label frequency 0:0. The class label frequency here again refers to the class label value of the node.

The prefix tree is second traversed to locate the node comprising the best split condition. FIG. 14 illustrates the splitting process of the prefix tree node where the best split condition is located in the exemplary example, according to one embodiment of the present technique. Upon second traversing process two nodes are located to comprise the best split condition. The node family PAV 1110 and the node family PAV 1125 are the two nodes where the prefix tree may be split to construct the decision subtree.

FIG. 15 illustrates a portion of the prefix tree node where the prefix tree is split in the exemplary example, according to one embodiment of the present technique. The node family PAV is first split to create the new prefix tree. The class label frequency of the node family PAV 1110 is used to decrement the class label frequency of the node old PAV 1105 and the prefix tree root node 505. The prefix subtree is selected in entirety after the node family PAV 1110 for building the new prefix tree. The selected prefix subtree may be node male PAV 1115.

FIG. 16 illustrates a portion of the new prefix tree built after splitting the prefix tree in the exemplary example, according to one embodiment of the present technique. The new path is added from the new prefix tree root node 1305. The new path is selected from the path of the prefix tree, till the node where the best split condition is located. In the exemplary example the new path may comprise the node old PAV 1605 with the class label frequency been 0:1. The class label frequency of the old PAV 1605 gets incremented to the nodes in the new path. In this exemplary
example only the new prefix tree node 1305 class label frequency may be incremented to 0:1.

FIG. 17 illustrates another portion of the new prefix tree built after splitting the prefix tree in the exemplary example, according to one embodiment of the present technique. The prefix subtree 1115 is the selected prefix subtree of the prefix tree as explained in FIG 15. The node family PAV 1110 is deleted from the path after the new path has been added to the new prefix tree.

FIG. 18 illustrates another portion of the prefix tree node where the prefix tree is split in the exemplary example, according to one embodiment of the present technique. The node family PAV 1125 is selected for splitting the prefix tree. The class label frequency of the node family PAV 1125 is decremented in the path. Thus the prefix tree root node 505 gets decremented by the class label frequency of the node family PAV 1125. The prefix subtree is selected in entirety after the node family PAV 1125 for building the new prefix tree. The selected prefix subtree may be node male PAV 1130.

FIG. 19 illustrates yet another portion of the new prefix tree built after splitting the prefix tree in the exemplary example, according to one embodiment of the present technique. The new path is appended from the new prefix tree root node 1305. The prefix subtree gets appended to the new path. The prefix subtree 1130 is the selected prefix subtree of the prefix tree as explained in FIG 18. The class label frequency of the new prefix tree root node will also be incremented. The node family PAV 1125 is deleted from the after the new path has been added to the new prefix tree.

Thus the prefix tree gets split at the node where the best split condition is located. The new prefix tree is originated from the prefix tree and also there will be the retained portion of the prefix tree, which comprises only the nodes which does not matches the best split condition.

FIG. 20 illustrates a retaining portion of the prefix tree after splitting the prefix tree at the prefix tree node where the best split condition is located in the
exemplary example, according to one embodiment of the present technique. The retained portion of the prefix tree comprises the prefix tree root node 505 with the class label frequency been 2:1, the node old PAV 1105 from the prefix tree root node 505 with the class label frequency 1:1 and the node male PAV 1120 from the node old PAV 1105 with the class label frequency 1:1.

FIG. 21 illustrates the new prefix tree after splitting the prefix tree at the prefix tree node where the best split condition is located in the exemplary example, according to one embodiment of the present technique. The new prefix tree comprises the new prefix tree root node 1305 with the class label frequency been 1:3. The new prefix tree root node 1305 comprises the node old PAV 1605 with the class label frequency been 0:1 and the node male PAV 1905 with the class label frequency 1:1. The node old PAV 1605 further comprise the node male PAV 1705 with the class label frequency 0:1.

The new prefix tree comprises all nodes which has the sequences with the PAV being family. The retained portion of the prefix tree comprises all nodes which has the sequences with the PAV being sports.

FIG. 22 illustrates further construction of the decision tree using the retained prefix tree and the new prefix tree after splitting the prefix tree in the exemplary example, according to one embodiment of the present technique. The first decision subtree 2205 is constructed using the new prefix tree whereas the second decision subtree 2210 is constructed using the retained portion of the prefix tree. The process of constructing the decision tree is recursive. The recursive process of constructing the prefix tree ends when the corresponding prefix subtree satisfies at least one or more set of predefined conditions. The set of predefined conditions includes at least one of the frequencies of the sequences’ class attribute at the prefix tree node in the path belonging to one class value or all of the predictor attributes has been selected as the best split condition or the frequency of the sequences’ class attribute is null or combinations thereof.
Some advantages of this model are: (1) its simple structure makes it an ideal tool for understanding discriminatory correlations and (2) inducing Decision Trees is generally fast. Because of these advantages Decision Tree implementations can be found in many open-source/commercial data mining toolkits like WEKA, SAS, SPSS Data Miner, MLC++, etc.

Exemplary Computing Environment

One or more of the above-described techniques can be implemented in or involve one or more computer systems. Figure 23 illustrates a generalized example of a computing environment 2300. The computing environment 2300 is not intended to suggest any limitation as to scope of use or functionality of described embodiments.

With reference to Figure 23, the computing environment 2300 includes at least one processing unit 2310 and memory 2320. In Figure 23, this most basic configuration 2330 is included within a dashed line. The processing unit 2310 executes computer-executable instructions and may be a real or a virtual processor. In a multi-processing system, multiple processing units execute computer-executable instructions to increase processing power. The memory 2320 may be volatile memory (e.g., registers, cache, RAM), non-volatile memory (e.g., ROM, EEPROM, flash memory, etc.), or some combination of the two. In some embodiments, the memory 2320 stores software 2380 implementing described techniques.

A computing environment may have additional features. For example, the computing environment 2300 includes storage 2340, one or more input devices 2350, one or more output devices 2360, and one or more communication connections 2370. An interconnection mechanism (not shown) such as a bus, controller, or network interconnects the components of the computing environment 2300. Typically, operating system software (not shown) provides an operating environment for other software executing in the computing environment 2300, and coordinates activities of the components of the computing environment 2300.

The storage 2340 may be removable or non-removable, and includes magnetic disks, magnetic tapes or cassettes, CD-ROMs, CD-RWs, DVDs, or any
other medium which can be used to store information and which can be accessed within the computing environment 2300. In some embodiments, the storage 2340 stores instructions for the software 2380.

The input device(s) 2350 may be a touch input device such as a keyboard, mouse, pen, trackball, touch screen, or game controller, a voice input device, a scanning device, a digital camera, or another device that provides input to the computing environment 2300. The output device(s) 2360 may be a display, printer, speaker, or another device that provides output from the computing environment 2300.

The communication connection(s) 2370 enable communication over a communication medium to another computing entity. The communication medium conveys information such as computer-executable instructions, audio or video information, or other data in a modulated data signal. A modulated data signal is a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media include wired or wireless techniques implemented with an electrical, optical, RF, infrared, acoustic, or other carrier.

Implementations can be described in the general context of computer-readable media. Computer-readable media are any available media that can be accessed within a computing environment. By way of example, and not limitation, within the computing environment 2300, computer-readable media include memory 2320, storage 2340, communication media, and combinations of any of the above.

Having described and illustrated the principles of our invention with reference to described embodiments, it will be recognized that the described embodiments can be modified in arrangement and detail without departing from such principles. It should be understood that the programs, processes, or methods described herein are not related or limited to any particular type of computing environment, unless indicated otherwise. Various types of general purpose or specialized computing environments may be used with or perform operations in
accordance with the teachings described herein. Elements of the described embodiments shown in software may be implemented in hardware and vice versa.

In view of the many possible embodiments to which the principles of our invention may be applied, we claim as our invention all such embodiments as may come within the scope and spirit of the following claims and equivalents thereto.