**SPL-1 Project Report, 2019**

**Decision Predictor using Decision Tree and Naive Bayes classifier**

**SE-305: Software Project Lab-1**

Submitted by

***Sabbir Ahmed***

**BSSE Roll No. : 1041**

**BSSE Session: 2018-2019**

Supervised by

***Abdus satter***

**lecturer**

**Institute of Information Technology**

****

**Institute of Information Technology**

**University of Dhaka**

29-05-2019

Table of Contents

[1. Introduction 1](#_Toc516172875)

[1.1. Background Study 1](#_Toc516172876)

[1.2 Challenges 1](#_Toc516172877)

[2.Project Overview 2](#_Toc516172878)

[2.1Package source 2](#_Toc516172879)

[2.1.1. Class: Attribute 2](#_Toc516172880)

[2.1.2. Class: Variable 2](#_Toc516172881)

[2.1.3. Class: Node 2](#_Toc516172882)

[2.1.4. Class: Tree 4](#_Toc516172883)

[2.2. Package: graphics 5](#_Toc516172884)

[2.2.1. Class: SelectionPage 5](#_Toc516172885)

[2.2.2. Class: Main 5](#_Toc516172886)

[2.3. Package: file 5](#_Toc516172887)

[2.3.1. Class: Data 5](#_Toc516172888)

[3. User Manual 6](#_Toc516172889)

[4. Conclusion 9](#_Toc516172890)

[References 9](#_Toc516172891)

# Introduction

Artificial Intelligence is the science of making intelligent computer programs or machines. Artificial intelligence algorithms span several different branches of computer science and mathematics including pattern recognition, [predictive modeling,](http://www.predictivemodelingtechniques.com/) text mining and search, [genetic programming,](http://www.predictivemodelingalgorithms.com/) heuristics, inference, and ontology, and [data analytics.](http://www.dataanalyticssoftware.com/) In this project, predictive modeling and data analytics branches are shortly implemented.

A decision tree represents a function that takes a set of attribute values as input and returns a single output value (decision). In other words, A decision tree is a tree in which each branch node represents a choice between a number of alternatives, and each leaf node represents a decision.

## Background Study

A tree (data structure) can be defined recursively (locally) as a collection of nodes (starting at a root node), where each node is a data structure consisting of a value, together with a list of references to nodes (the "children"), with the constraints that no reference is duplicated, and none points to the root.

A free tree is a connected, acyclic, undirected graph. A rooted tree is a free tree in which one of the vertices is distinguished from the others. The distinguished vertex is known as the root of the tree.2 In this project, the standard data structure of a generic tree construction is followed.

Decision trees are commonly used in [**operations research**,](https://en.wikipedia.org/wiki/Operations_research) specifically in [**decision analysis**,](https://en.wikipedia.org/wiki/Decision_analysis) to help identify a strategy most likely to reach a [**goal**,](https://en.wikipedia.org/wiki/Goal) but are also a popular tool in [**machine**](https://en.wikipedia.org/wiki/Decision_tree_learning) [**learning**.](https://en.wikipedia.org/wiki/Decision_tree_learning) **Decision** trees can also be used to analyze data when the y-outcome is a continuousmeasurement (such as age, blood pressure, ejection fraction for the heart, etc.). Such trees are called regression trees.

For implementing this project, we have to introduce with some data compressing algorithm. For compressing data, no data should be loss. It is the main challenge. The process of reducing the size of a data file is referred to as data compression.3

In this project, The user will give a input data (i.e. weather.arff) file. Then the tree will be built.

**Example:**

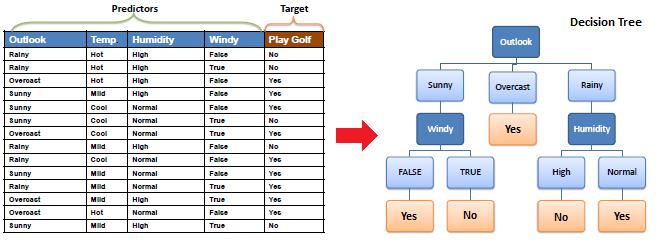


Figure 1: Input Data & Tree construction4

## Challenges

To predict the decision, there are some challenges that it (tree) has to be faced. For better prediction, C4.5 algorithm is used. To save memory (minimize the depth of the tree), ID3 algorithm is used. It also saves time while searching for decision.

# 2.Project Overview

## 2.1Package source

This package contains the principal part of the project. This implementation uses Java’s ArrayList library. The details of the package is given below.

### 2.1.1. Class: Attribute

**Attribute (ArrayList<Integer>, String)**

This is a constructor that takes a list of integer as data set and a string that will define the Attribute instance’s name.

**calculateEntropy () : void**

This method calculates the total entropy (information it contains) of this Attribute instance following C4.5 algorithm.

**getProbability () : String**

This method returns the probability of the Attribute instance in string format.

### 2.1.2. Class: Variable

**Variable (int, String, int, String[], ArrayList<Integer>)**

This is a constructor that receives an integer as instance’s column number, a string as instance’s name, an integer as the number of attributes, an array of string as attribute instances’ names, a list of integer as available data set.

**calculateChildEntropy () : void**

This method calculates the average child entropy of its instance.

**makeAttributeList (String[]) : void**

This method creates all attribute instances of this Variable instance.

**dataForAttributeList () : void**

This method creates data set for all attribute instances of this Variable instance.

**calculateEntropyAndInformationGain () : void**

This method calculates the total entropy and information gain of this Variable instance following C4.5 algorithm and ID3 algorithm.

### 2.1.3. Class: Node

**Node (Attribute)**

This constructor receives an Attribute instance as parameter and creates a Node instance that contains no Variable instance.

**Node (Variable, Attribute)**

This constructor receives an Attribute instance and a Variable as parameter and creates a Node instance using these parameters.

**addSibling (Node) : void**

This method receives an Node instance and makes it as a sibling node of the current instance

**addChild (Node) : void**

This method receives an Node instance and makes this Node as a child node of the current instance.

**nodeInformation () : String**

This method returns the variable’s name of its parent node including the attribute’s name of the instance as string format.

**publishResult () : String**

This method returns the result of all decision attributes with probability (chance it can be happened).

**getResult () : int**

This method returns a decision identity belonging the highest probability (chance).

### 2.1.4. Class: Tree

**Tree (Data)**

The constructor receives a Data (user define class) type as parameter and create a instance of Tree class.

**findRootAndBuildTree () : void**

The method creates the root node based on the data set by calling findMaxInformationGain method.

**findMaxInformationGain (ArrayList<Integer>, ArrayList<Integer>) : Variable**

This method receives a list of integer as the data set and another list of integer to consider the available variables. It creates all possible Variable instances and finds the Variable instance that contains highest information gain among them. The method also returns the Variable instance containing maximum information gain.

**makeTree (Node, ArrayList<Integer>) : void**

The method receives a Node instance and a list of integer as parameter. The method initially takes root node and creates the tree properly by calling itself recursively.

**createDecisionSet (Node) : void**

The method receives a Node instance and creates decision rules by calling itself recursively.

**getDecisionList (): void**

The method prints all the decision rules in the console based on the tree.

**makeDecision (int[]) : String**

The method receives a list of integer as input data set and calls searchForDecisionNode method to reach the decision node and returns the probable result as string format.

**getChild (Node, int) : Node**

The method receives a Node instance and an integer as a index of the attribute number. Then it returns the expected child node of the given Node instance which was passed as a parameter while calling the method.

**searchForDecisionNode (int[], Node) : Node**

The method receives a list of integer as a input data set and traverse the Tree instance to reach the decision node (or leaf). This method returns that decision node.

**trainAndTestDataGenerate (): void**

In this method, 20% of total data set (that generates randomly) is considered as test data set.

The rest of the data set is considered as training data.

**accuracyTesting (): void**

In this method, the test data set is searched decision based on the tree that construct upon the training data set. It calculates the percentage of accurate prediction rate.

### 2.1.5. Class: Naïve Bayes

### Naïve bayes(file)

### In this class I ‘ve trained my data set for calculating the probability of class of my data set . I’ve worked for on my data set to apply the naïve Bayes classifier.

### 2.1.6. Class: label

### Label(class , number of feature)

### In this class I have done the calculation of the accuracy testing for naïve bayes , and calculate the probability considering all feature .



## 2.2. Package: graphics

This project uses Java’s awt and swing libraries to design and display the graphical user interface.

### 2.2.1. Class: SelectionPage

**SelectionPage (Tree, String)**

The constructor receives a Tree instance and a string as parameter. Then it constructs SelectionPage instance.

**initialize () : void**

The method creates the user interface using a frame.

**getTestVariables () : void**

The method is called when the “Submit” button is pressed. In this method, the user given data (query) is parsed as an array of integer data set. The decision is shown based on the given queries.

### 2.2.2. Class: Main

**main (String[]) : void**

In this Class, the main method is called. Program execution starts from this method and it does a 10 fold cross validation.

## 2.3. Package: file

This project uses Java’s ArrayList, Map and File libraries to receive and process data. There is only one class in this package.

### 2.3.1. Class: Data

**Data (String)**

The constructor receives a string that represents the file name. If the given file exists, then the Data instance will be created.

**receiveDataFromFile () : void**

The method opens the desired file (if exists). In this method, the information of the file is parsed which is needed for tree construction.

# User Manual

Sample program output of the program developed has been depicted below.

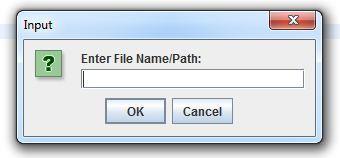


Figure 2: Prompt for asking File Location on Startup

User can type in the file location of the .arff file, which contains the data set upon which the decision tree would be built, in the text field, as displayed in Figure 2.

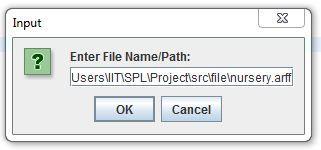


Figure 3: Sample File Directory which contains sample data set

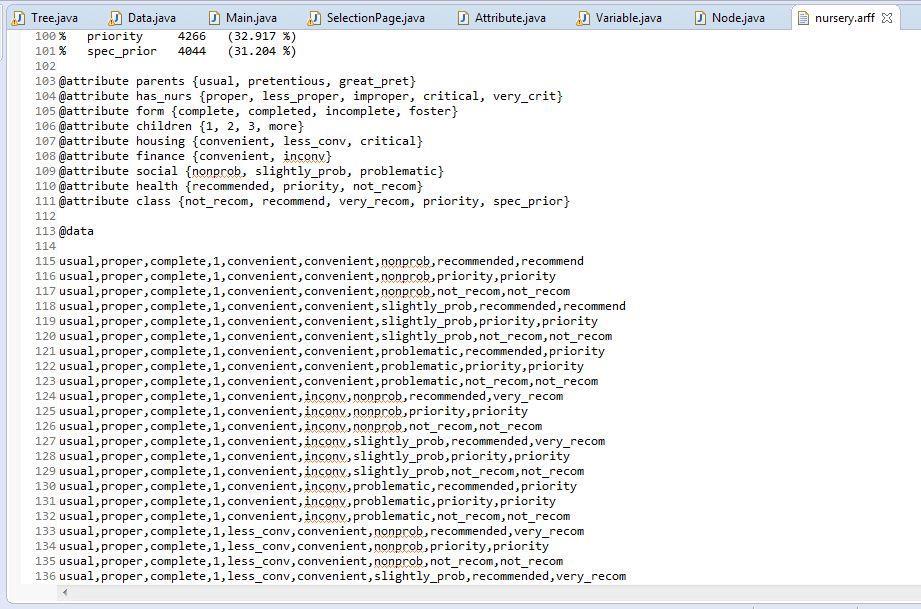


Figure *4*: Data set to be analyzed

Decision Predictor using Decision Tree

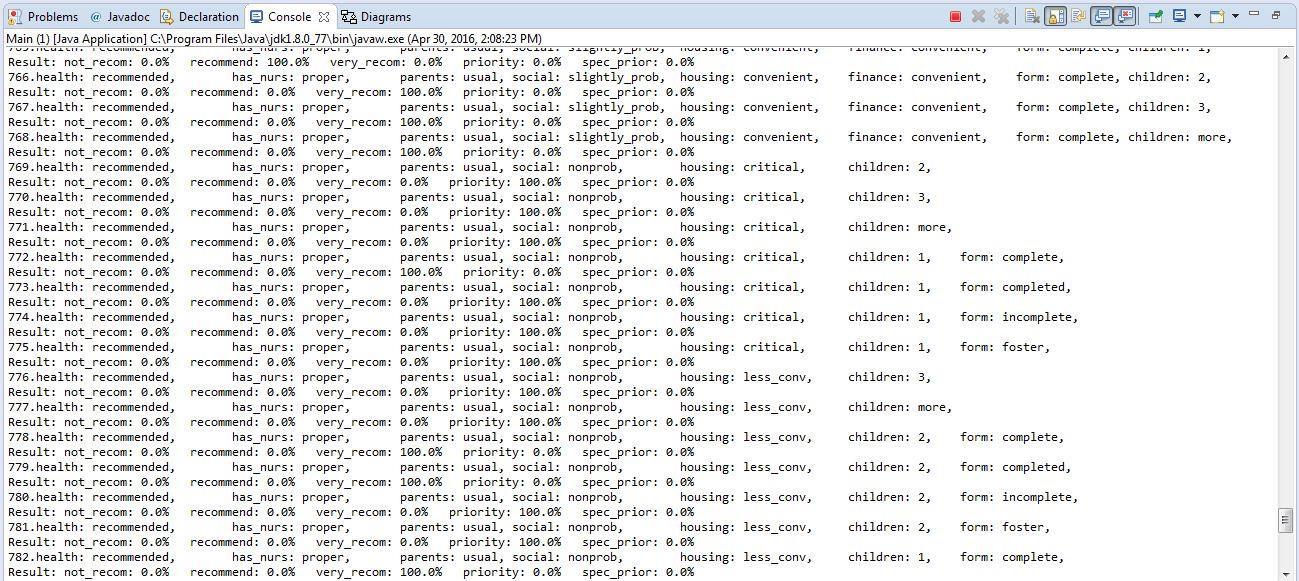
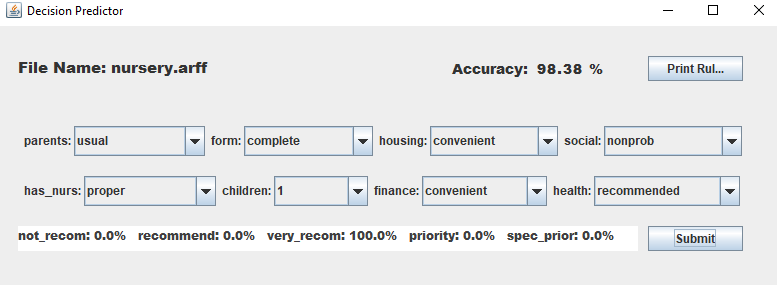


Figure *5*: Decision Rules displayed in Console



# 

Figure 6 :Result Displayed

4.conclusion

I have learned a lot through this project ranging on topics from information theory, bagging theory, classification and regression model, tree construction model and so on. I have learned from some research papers based on this topic. This project was quite challenging. But I have gained experience from this project. I would like to conclude by thanking my project supervisor, for guiding me along the way.

# 

# References

1. https://www.saedsayad.com/decision\_tree.html, Decision Tree – Classification, last accessed on 10-05-2019
2. Introduction to Algorithms, Third Edition, Thomas H. Coreman – Charles E. Leiserson - Ronald L. Rivest – Clifford Stein
3. Machine Learning: An Algorithmic Perspective, Second Edition, Stephen Marsland.