Programming Assignment 1: Decision Trees

Instructor: Dr. Satish Kumar Thittamaranahalli

Group Member: Yu Hou; Haoteng Tang

In this assignment, we used Python and Matlab to implement the decision tree using different code. And then we trained the data of previous experience in New Year’s Eve, and predict whether you will have a good night-out in Jerusalem for the coming New Year’s Eve with a new record. In the first section, we would introduce you our pseudocode. We had different thoughts to implement the decision tree, but with the same algorithm. Also, we would show you the result of the prediction. In the second section, we researched on a good implementation of the decision tree algorithm which was a software named Weka. In the third section, we discussed some interesting applications of decision trees.

**1 pseudocode of decision tree**

(1) In our programming, we used entropy as the basic evidence to divide the tree into a different class. Here is our pseudocode of the decision tree using python with recursion: (YU HOU)

Pseudocode:

**generateTree function**:

If basic case:

Entropy ==0

Put ‘Yes’ or ‘No’ into { key : value }

Return tree ( { key :value ) )

If not basic case:

Find the best entropy using **getEntropy function**

Use best entropy to divide database into different sub-database using **divideTree fuction**

Put sub-database into generateTree function //recursion

Return tree ( {key : value })

**divideTree fuction:**

**getEntropy function:**

(2) We also used Matlab to implement the decision tree, here is another thought without recursion. (HAOTENG TANG)

Pseudocode:

Which attributes should be in used for each branches node.

The original matrix that contained the data points that will be split.

Calculate the **average entropy** in each attribute, and select the smallest to split this node.

Split the data points into son branches

Save the son branches

**If** it is the leaf, label ‘leaf’.

**Else**, repeat all of these steps of split the next node.

Both of these two different methods get the same decision tree. The final result of the decision tree can be shown as follows:



Figure1 The result of decision tree

The decision tree is stored in the **dictionary data structure** which has many pairs of Key and Value. If we print out the decision tree as dictionary just like follows:

*{'Occupied': {'High': {'Location': {'Talpiot': 'No', 'City-Center': 'Yes', 'Mahane-Yehuda': 'Yes', 'German-Colony': 'No', 'Ein-Karem': 'tie'}}, 'Moderate': {'Location': {'Talpiot': {'Size': {'Large': 'No', 'Medium': 'Yes', 'Small': 'tie'}}, 'City-Center': 'Yes', 'German-Colony': {'Size': {'Medium': 'No', 'Large': 'Yes', 'Small': 'tie'}}, 'Ein-Karem': 'Yes', 'Mahane-Yehuda': 'Yes'}}, 'Low': {'Size': {'Large': 'No', 'Medium': {'Price': {'Cheap': 'No', 'Expensive': 'No', 'Normal': 'Yes'}}, 'Small': 'No'}}}}*

For example, the ‘Occupied’ is in the first level, and it has ‘High’, ‘Moderate’, and ‘Low’ these three values, it will be stored as { ‘Occupied’ (key) : {‘High’, ‘Moderate’, ‘Low’ }(values) }. In this data structure, we can store the whole tree.

We also store the decision tree into the two-dimensional matrix. And the result of running the program also was like this according to the instruction of this assignment:

*Print out decision tree:*

*Occupied*

*Location,Location,Size*

*No,Yes,Yes,No,tie,Size,Yes,Size,Yes,Yes,No,Price,No*

*No,Yes,tie,No,Yes,tie,No,No,Yes*

(3) prediction (HAOTENG TANG)

The record need to be predicted is: (size = Large; occupied = Moderate; price = Cheap; music = Loud; location =City-Center; VIP = No; favorite beer = No).

According to the decision tree which we had made, since our input is { Occupied : Moderate}, so we should check the under Moderate. We find that Location which has {Talpiot, City-Center, German-Colony, Ein-Karem, Mahane-Yehuda,} is under Moderate. Since our Location is City-Center. And we find it is a leaf node. And we can determine that Enjoy is Yes.

So due to our input, the output is Yes.

(4) code-level optimization and challenges

We optimized the code in time consuming and memory consuming. For example, we were trying to reduce using nested loop, so that we can reducing the time-consuming. Another example is that we implement the decision tree both using recursion and without recursion.

The big challenge is how to do the recursion. Sometimes we will be lost in the recursion returning the decision tree when it is not the basic case.

**2 Software research**

(1) Weka software library function (YU HOU)

Weka is a free software for the researcher to implement machine learning. This software is implemented by java. As for the decision tree, the author of Weka improved the c4.5 algorithm and name it J48 because of the language is Java. Here is the pseudocode of J48. [[[1]](#endnote-1)][[[2]](#endnote-2)]

**generateDT:**

tree == empty

**if** basic case (meet stopping criteria) then stop

Calculate all information-theoretic criteria for all attributes finding the best attribute

Create a decision node that tests best attribute in the root

Get the divided tree

**For** all divided tree, **do** the **generateDT again** //recursion

Return tree

When we put all data into the Weka and run the J48 decision tree algorithm, we will get our result. In Weka, users are allowed to see the visual tree. Here is the figure of decision tree without any pruning.

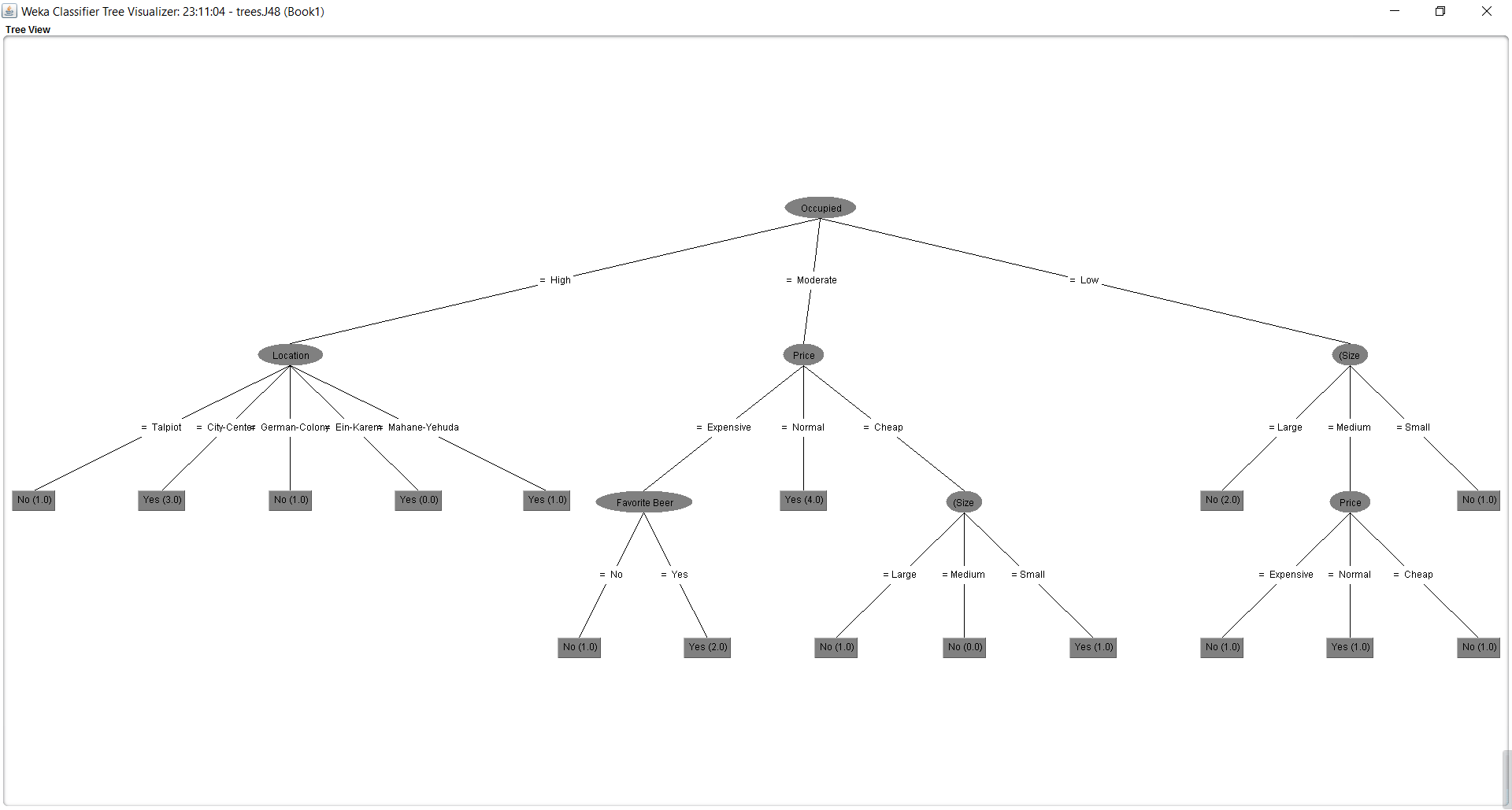


Figure 2 Decision tree from Weka

Also, Weka allows users to prune the tree. As it shown in the figure, you can choose “True” or “False” to prune the tree. And also we can set the minimum objects to improve the decision tree. When we get our decision tree, we also can get a report named the confusion matrix. The confusion matrix can explain the result of the decision as it is shown in the figure.

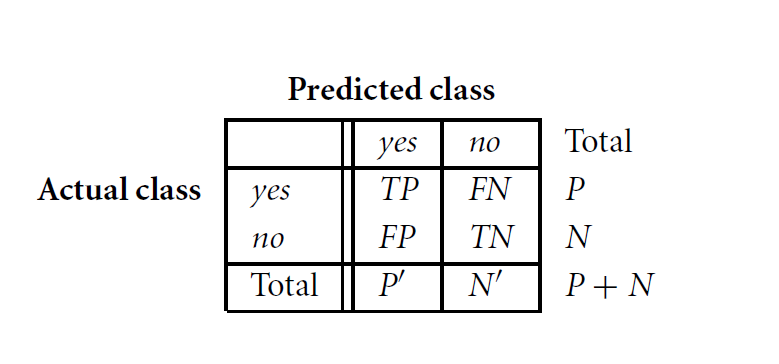


Figure 3 Confusion matrix

The accuracy=(TP+TN)/(P+N), and the error rate=(FP+FN)/(P+N). We can see the result in the figure.

The advantages in Weka software:

1) This decision tree can deal with the continuous values, however, my implementation can only deal with discrete values.

2) It used the Information Gain Ratio to choose the attributes, which avoid the situation that we choose the attribute which can get more potential values of this attribute.

3) This software can prune the decision tree. This can avoid overfitting.

4 )This software can deal with missing values. This can give the missing value a common value, or give it a mean value, or use probabilities to give it a value.

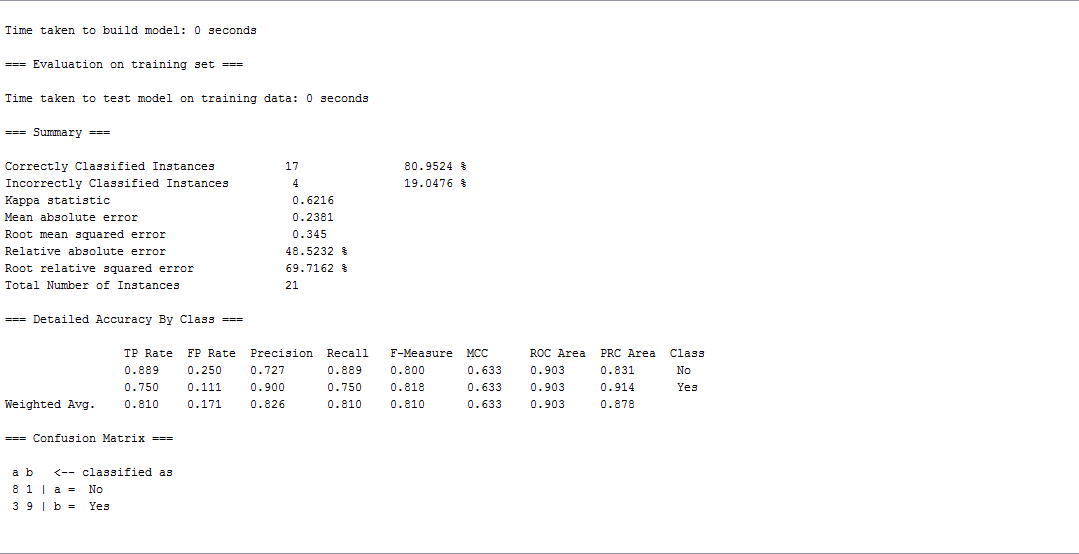
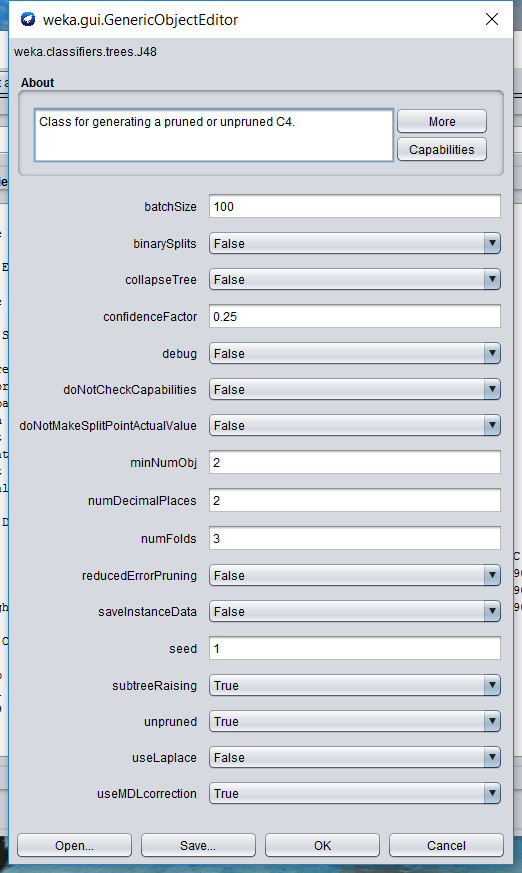


Figure 4 Setting Figure 5 Confusion matrix in one of the result

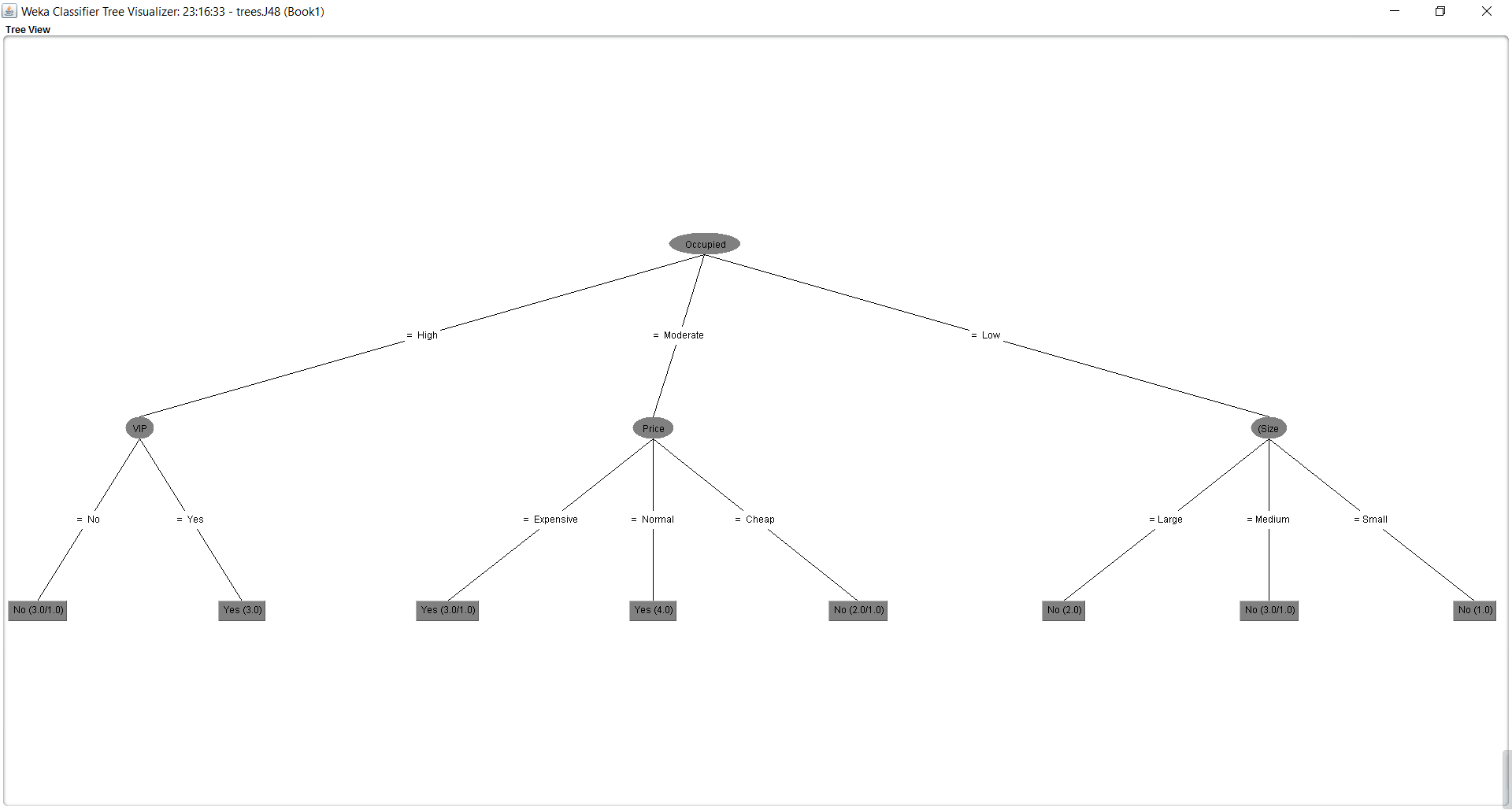


Figure 6 Decision tree after prune

What can I do to improve my code:

1) Learn some methods to prune the decision tree. Like the decision tree using Weka. It can prune the tree avoiding overfitting. So in my code, I can learn to add such function.

2) introduce a new concept- Information gain calculation to improve my code by reducing a bias to multi-valued attributes by taking the number and size of branches into account when choosing an attribute. For example, we have 21 records in a dataset, but one of the attributes has 21 values. This attribute can divide this tree very well but it meaningless.

(2) Matlab library function (HAOTENG TANG)

we used MATLAB library function to build up a decision tree and input the data we want to predict the output.

the method is:

tree=fitrtree(tbl,y);

test\_data=[1,2,3,1,2,1,1];

ouput=predict(tree, test\_data);

* ‘tbl’ is the input training data which is the first 7 column of our training data matrix. And y is the output of our training data which is the last column in our training data matrix. And when we make a prediction, the ‘test\_data ’should be: size = Large; occupied = Moderate; price = Cheap; music = Loud; location = City-Center; VIP = No; favorite beer = No which has already converted to number.
* The predicted output is there is 60% possibility if we choose 1(YES, enjoy). This outcome supports the output from my own algorithm.



What can I do to improve the code:

1) I should set up a recursion method for each level of the decision tree.

2) when I meet a ‘tie’ on the leaf of my decision, I should continue to calculate the probability of the exact value of ‘enjoy’ to help make a decision in the end.

**3 Application**

(1) Application 1-predict the architectural designer (YU HOU)

Decision tree gives us a new way to make a decision based on historical data.

In an architectural company with many architectural designers who are variable in different working abilities, working experience, and technics. When we have a new project, a manager should make a decision which architectural designer should be assigned to this project. The traditional way for a manager is to do this job based on his or her experience, the comprehension of this project, and his or her understand of the architectural engineers. However, with so many designers, a manager sometimes could not always choose an appropriate designer to work efficiently and effectively also to ensure project quality. So it is necessary to establish a classifier helping a manager classify their designers and make a right decision based on the classifier.

First of all, we should get the historical data of our designers in our company. Here is a table of the historical data. In this table, we have “Task Mode”, “Work Item Type”, ”Constraint Type”, “Baseline Work”, “Project Client”, “Task Complexity”, “Task Type”, “Task Urgency”, these kinds of attributes and each record has certain person to do this kind of job. So when we have a new project, we can evaluate this project using these attributes, and input these attribute into the model. We will get the name of the designer. And the designer using this decision tree is appropriate to do this kind of project.

Table 1 historical data of the designers

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| No. | Task Mode | Work Item Type | Constraint Type | Baseline Work | Project | Client | Task Complexity | Task Type | Task Urgency |
| 1 | André | steel | As Soon As Possible | 6 hrs | ADMIN | FABIO | 3-High | 1-New | 2-Medium |
| 2 | Carlos | wood | As Soon As Possible | 4 hrs | ADMIN | ALEXANDRE | 3-High | 1-New | 2-Medium |
| 3 | Eder | concrete | Must Start On | 4 hrs | ADMIN | DANIEL | 2-Medium | 1-New | 2-Medium |
| 4 | Jonas | concrete | As Soon As Possible | 4 hrs | ADMIN | ALEXANDRE | 2-Medium | 1-New | 2-Medium |
| 5 | Josue | concrete | As Soon As Possible | 2 hrs | ADMIN | DANIEL | 1-Low | 1-New | 1-High |
| 6 | Liliane | wood | Start No Earlier Than | 8 hrs | SURVEY | ALEXANDRE | 2-Medium | 2-Revision Improvement | 3-Low |
| 7 | Roberto | wood | Start No Earlier Than | 8 hrs | SURVEY | DANIEL | 2-Medium | 2-Revision Improvement | 2-Medium |
| 8 | Roberto | steel | As Soon As Possible | 2 hrs | FINANCIAL | FABIO | 2-Medium | 1-New | 2-Medium |
| 9 | Roberto | wood | As Soon As Possible | 4 hrs | FINANCIAL | FABIO | 2-Medium | 1-New | 2-Medium |
| 10 | Liliane | steel | As Soon As Possible | 8 hrs | FINANCIAL | FABIO | 2-Medium | 1-New | 2-Medium |
| 11 | André | concrete | As Soon As Possible | 4 hrs | FINANCIAL | ALEXANDRE | 2-Medium | 1-New | 2-Medium |
| 12 | Eder | steel | As Soon As Possible | 2 hrs | FINANCIAL | ALEXANDRE | 2-Medium | 1-New | 2-Medium |
| 13 | Carlos | steel | As Soon As Possible | 3 hrs | FINANCIAL | ALEXANDRE | 3-High | 1-New | 2-Medium |
| 14 | Carlos | wood | As Soon As Possible | 2 hrs | FINANCIAL | DANIEL | 3-High | 1-New | 2-Medium |
| 15 | Carlos | steel | As Soon As Possible | 4 hrs | FINANCIAL | ALEXANDRE | 2-Medium | 1-New | 2-Medium |
| 16 | Eder | concrete | Must Start On | 8 hrs | ADMIN | ALEXANDRE | 2-Medium | 1-New | 2-Medium |
| 17 | André | concrete | As Soon As Possible | 4 hrs | SURVEY | DANIEL | 1-Low | 1-New | 2-Medium |
| 18 | Liliane | wood | As Soon As Possible | 2 hrs | SURVEY | FABIO | 2-Medium | 2-Revision Improvement | 2-Medium |
| 19 | Roberto | wood | Start No Earlier Than | 3 hrs | FINANCIAL | FABIO | 2-Medium | 2-Revision Improvement | 2-Medium |
| 20 | Roberto | steel | Start No Earlier Than | 4 hrs | FINANCIAL | DANIEL | 2-Medium | 1-New | 2-Medium |
| 21 | Jonas | concrete | Must Start On | 3 hrs | FINANCIAL | ALEXANDRE | 2-Medium | 1-New | 2-Medium |

(2) Application 2-predict if the client will subscribe a term deposit (YU HOU)

The data is related with direct marketing campaigns (phone calls) of a Portuguese banking institution. The classification goal is to predict if the client will subscribe a term deposit based on input variables including. Table 2 shows us the brief Introduction on the dataset.[[[3]](#endnote-3)]

Table 2 Brief Introduction on the dataset

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Data Set Characteristics:** | Multivariate | **Number of Instances:** | 45211 | **Area:** | Business |
| **Attribute Characteristics:** | Real | **Number of Attributes:** | 20 | **Date Donated** | 2012/2/14 |
| **Associated Tasks:** | Classification | **Missing Values?** | N/A | **The Number of Web Hits:** | 268601 |

Using Weka we can get the decision tree of this result. Showing in figure 7

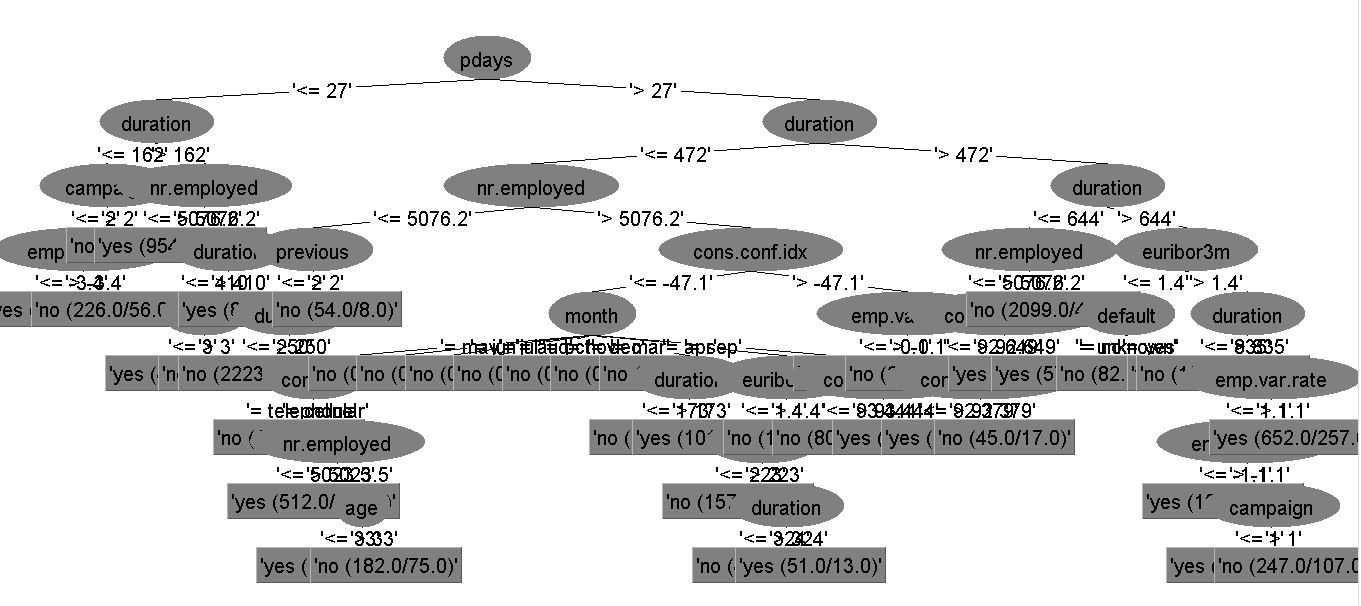


Figure 7 Decision tree of application 2

**Reference：**

1. [] http://www.cs.waikato.ac.nz/ml/weka/index.html [↑](#endnote-ref-1)
2. [] Han, Jiawei, Jian Pei, and Micheline Kamber. Data mining: concepts and techniques. Elsevier, 2011. [↑](#endnote-ref-2)
3. [] http://archive.ics.uci.edu/ml/datasets/Bank+Marketing [↑](#endnote-ref-3)