The Decision Tree Usage for the Results Analysis of the Psychophysiological Testing

Myroslava Bublyk[a], Vasyl Lytvyn[a], Victoria Vysotska[a], Lyubomyr Chyrun[b], Yurii Matseliukh[a] and Nataliia Sokulska[c]

[a] *Lviv Polytechnic National University, S,.Bandera street, 12, Lviv, 79013, UkIntroverte*
[b] *Ivan Franko National University of Lviv, University street, 1, Lviv, 79000, UkIntroverte*
[c] *Hetman Petro Sahaidachnyi National Army Academy, Heroes of Maidan street, 32, Lviv, 79026, UkIntroverte*

Abstract
The interposition researches of the regularities in the data, which is the result of psychophysiological testing's have been considered in this paper. Decision trees, that are built based on the analysis of these data, can used for decision making regarding recruiting and passing specialists in the work for example as an operator of the energy networks.

Keywords 1
Content analysis, psychophysiological testing, the decision trees, recruiting

1. Introduction

The article describes the process of building decision trees based on the analysis of the results of psychophysiological testing of a group of persons working in the energy sector [1]. Such tests are intended to identify some of the psychological qualities and characteristics of the employee [2]. A psychologist on certain scales carries out the evaluation of the results of these tests [3]. In addition, their supervisors also evaluate employees. The results of such testing are used in the process of deciding on the success and reliability of the specialist, as well as his professional suitability. As one of the major drawbacks of such a process to weather is the sufficiently high level of subjectivity of decision-makers, it is necessary to automate the analysis of the input data that results from testing in some way [4]. The ultimate goal of the analysis is to solve the classification problem, i.e. to identify significant patterns or systematic relationships that can then applied to new sets of inputs. The results of the study will improve the process of testing and decision-making, reducing the level of subjectivism in the assessment of specialist, and abstract from the partial inconsistency and incompleteness of the test results. Data mining will performed to solve this research problem: building decision trees using C4.5 and CART algorithms.

2. Advantages and general scheme of data mining

There are many techniques for collecting data. This problem has comprehensively studied and formalized by statistics. In this context, statistics are understood not as mathematical discipline, but in the broad sense - as a social science that studies the quantitative side of social phenomena and reveals the quantitative patterns of processes. There are also many methods designed to analyze existing data. These methods form the basis of such discipline as Data Mining. Data Mining makes it possible to use a wide range of methods for different data and for different purposes [5-8].

IDDM'2020: 3rd International Conference on Informatics & Data-Driven Medicine, November 19–21, 2020, Växjö, Sweden
EMAIL: my.bublyk@gmail.com (M. Bublyk); Vasyl.V.Lytvyn@lpnu.ua (V. Lytvyn); victoria.a.vysotska@lpnu.ua (V. Vysotska); Lyubomyr.Chyrun@lnu.edu.ua (L. Chyrun); indeed.post@gmail.com (Y. Matseliukh); natalya.sokulska@gmail.com (N. Sokulska)
ORCID: 0000-0003-2403-0784 (M. Bublyk)0000-0002-9676-0180 (V. Lytvyn); 0000-0001-6417-3689 (V. Vysotska); 0000-0002-9448-1751 (L. Chyrun); 0000-0002-1721-7703 (Y. Matseliukh); 0000-0002-3425-5517 (N. Sokulska)

Traditional mathematical statistics, which for a long time claimed the role of the main tool of data analysis, frankly "saved" in the face of new problems. The main reason is the concept of sample averaging, which leads to dummy operations (such as the average Agreeableness of patients in a hospital, the average height of a home on the street, etc.). The methods of mathematical statistics have proven to be useful mainly for testing pre-formulated hypotheses (verification-driven data mining) and for "approximate" reconnaissance analysis, which forms the basis of online analytical processing (OLAP) [9-12]. The essence and purpose of Data Mining technology can characterized as a technology that is designed to search for non-obvious, objective and useful patterns in large-scale data [1]. Non-obvious means that the patterns found are not revealed by standard methods of processing information or expertly. Objective is this means that the revealed patterns will be completely true, unlike expert opinion, which is always subjective. Practically useful is this means that the conclusions have a specific meaning that can found practical. Data Mining methods and algorithms include the following [1]:

- Artificial neural networks;
- Decision trees;
- Symbolic rules;
- Methods of nearest neighbor and k-nearest neighbor;
- Method of reference vectors;
- Bayesian networks;
- Linear regression;
- Correlation-regression analysis;
- Hierarchical methods of cluster analysis;
- Non-hierarchical cluster analysis methods, in particular k-means and k-median algorithms;
- Methods for finding associative rules, in particular, the Apriority algorithm;
- Limited search method;
- Evolutionary programming and genetic algorithms;
- Various methods of data visualization, etc.

Most of the analytical methods used in Data Mining technology are known mathematical algorithms and methods. New is the ability to use them in solving particular problems, due to the capabilities of the hardware and software that have emerged now. Note that most Data Mining methods are developed within the framework of artificial intelligence theory. In the general case, the process of data mining consists of three stages (Fig. 1) [1].

1. Identification of patterns (free search).
2. Use of identified patterns to predict unknown values (predictive modeling).
3. Analysis of exceptional situations, designed to identify and explain anomalies in the patterns found.

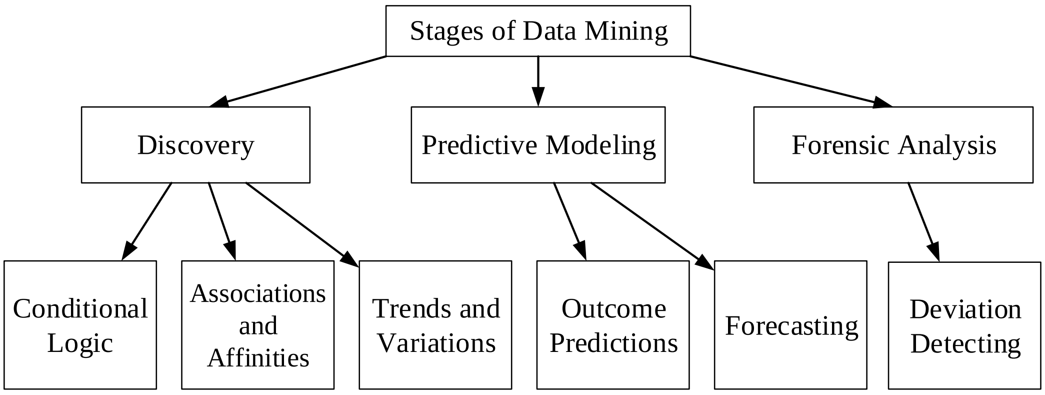

Figure 1: The stages of the data mining process

Sometimes there is an explicit stage in the intermediate stage of verification of the validity of the identified patterns between their finding and use (validation stage). Free search is defined as the process of researching an input database for the search for hidden patterns without first determining the hypotheses regarding the nature of these patterns. In other words, the program itself takes the initiative to look for interesting anomalies or patterns in the data, freeing the analyst from the need to reflect and make appropriate queries. This approach is especially productive when exploring large databases with a large number of hidden patterns, most of which would be missed when directly searched by direct user queries for input. The free search stage should, as a rule, include not only the generation of regularities, but also the verification of their accuracy on a set of data that did not participate in the formation of these regularities. In predictive modeling, the results of the first stage are used, i.e. the regularities found in the input data are used to predict unknown values [4, 12-15]:

- When classifying a new object, we can with certain likelihood attribute it to a specific group of results, taking into account the known values of its attributes;
- In predicting a dynamic process, the results of trend and periodic fluctuations can used to make assumptions about the likely development of some dynamic process in the future.

Free search reveals general patterns, that is, it is inductive, while any prediction expresses "guesses" about the values of specific unknown quantities, therefore, is deductive. In addition, the resultant constructs can be both reasonably interpretable (decision trees) or not interpreted as "black boxes" (neural networks) [7]. The subject of exceptions analysis is anomalies in the revealed regularities, that is, exceptions that are not explained. To find them, you first need to determine the norm (this applies to the free search stage) and then determine its violation [16-18]. They may find a logical explanation, which can also framed as a pattern. However, it is possible that we are dealing with errors in the input, and then the exception analysis can used as a tool to "clear" the data.

3. Decision tree method

Decision trees are a method that is suitable not only for solving classification problems but also for calculations, and is therefore widely used in the fields of finance, business and medicine. As a result of applying this method to the train data sample, a hierarchical structure of classification rules such as "IF ..., THAT ..." is created, which looks like a tree. To decide which class an object or situation belongs to, we answer the questions that are at the top of this tree, starting from its root. If the answer is positive, then we go to the left vertex of the next level, if negative - to the right vertex; then we again answer the question related to the corresponding vertex. In this way, we reach one of the end vertices, a leaflet, which contains an indication of which class the object being analyzed should assigned to. The advantages of this method are that such a representation of the rules is clear and easy to understand [9].

The popularity of the approach is associated not only with the clarity and clarity of the presentation, but also with the ease of implementation. The disadvantage of this method is that decision trees are fundamentally incapable of finding "better" rules in the data. However, using them, you can more or less accurately classify the object depending on the number of known features.

Decision trees can considered as the most efficient structured data warehouse. As proven in graph theory, binary tree search is the most efficient search algorithm available. For date, there are a large number of algorithms that implement the construction of decision trees, of which the following are the most widely used and popular [6]:

- CART (Classification and Regression Tree) is an algorithm developed by L. Breiman; is an algorithm for constructing a binary decision tree is a dichotomous classification model; each apex of a tree at splitting has only two descendants; as its name implies, the algorithm solves the problems of both classification and regression;
- C4.5 - an algorithm for constructing a decision tree with an unlimited number of descendants at the top, developed by R. Quinlan; not suitable for working with a continuous target field, so it solves only the classification problem;
- QUEST (Quick, Unbiased, Efficient Statistical Trees) is an algorithm developed by V. Loh and I. Shih, which uses improved variants of the method of recursive quadratic discriminant analysis, which allows to realize multidimensional branching of linear combinations of ordinal predicates; contains a number of new tools to increase the reliability and efficiency of the induction tree.

4. The article goals

The main purpose of this article is to present the results of constructing decision trees using C4.5 and CART algorithms for the results of psychophysiological testing. We formulate the main goals of the article as follows:

- Describe algorithms for building C4.5 and CART decision trees;
- Give examples of constructing decision trees using the above algorithms;
- Describe many train examples for decision tree building, which is a table of psychophysiological testing results;
- To give results of construction of a decision tree according to C4.5 and CART algorithms, that is, the formed rules, and to compare them;
- To formulate the goals of further research in this area.

It should note that the objectives of the article include a description of the results of the construction of the rules, but not their evaluation and application in practice for decision-making. The focus is on demonstrating the process of building decision trees and efficient classification by C4.5 and CART algorithms.

5. Algorithm C4.5 building a decision tree

Let us be given a set of train examples *T*, where each element of that set is described *m* by attributes. The number of examples in a set T will called the power of that set and we will note $|T|$. Let the class label or decision attribute take the following values: $C_1, C_2, \ldots, C_k$ [5].

Our task will be to build a hierarchical classification model in the form of a tree of multiple examples *T*. The process of building a tree will take place from top to bottom. The root of the tree is first created, then the offspring of the root, etc. In the first step, we have a tree consisting only of the root apex and a train set *T*, associated with the root. It is necessary to divide this set into subsets. You can do this by selecting one of the attributes to test. Then the result is a breakdown *n* (by the number of attribute values) subsets and, accordingly, are created *n* descendants of the root, each of which is matched by its own subset, obtained by partitioning the set *T*. This procedure is then recursively applied to all subsets (descendants of the root) and the like.

Let's consider in more detail the criterion for choosing the attribute by which the branching should occur. Obviously, we have *m*, as the number of attributes, the possible options from which we should choose the best one. Some algorithms exclude reuse of an attribute when constructing a tree.

Let us have a check *X* (any attribute can selected for verification) accepting *n* values $A_1, A_2, \ldots, A_n$. Then the partition *T* on verification *X* will give us subsets $T_1, T_2, \ldots, T_n$, at *X*, equal to, respectively, $A_1, A_2, \ldots, A_n$. The only information available to us is how classes are divided into plurals *T* and its subsets obtained by partitioning by *X*. This is exactly what is used for the criterion [5].

Let $freg(C_j, S)$ is number of examples from some set *S*, belonging to the same class C_j. Then the probability that an example is chosen from the set *S* will belong to the class C_j is $P = freg(Cj_j, S)/|S|$.

According to information theory (and Shannon's theory), the amount of information received in a message depends on its probability

$$\log_2 \frac{1}{P}. \tag{1}$$

Since a binary logarithm is used, formula (1) quantifies the information expressed in bits. Formula

$$Info(T) = -\sum_{j=1}^{k} \frac{freg(C_j, T)}{|T|} * \log_2 \left(\frac{freg(C_j, T)}{|T|} \right). \tag{2}$$

is an estimate of the average amount of information needed to determine the class of an example from a plurality *T*. In the terminology of information theory, equality (2) is called the entropy of the set *T*. The same estimate, but only after splitting the plural *T* on verification *X*, gives an expression

$$Info_X(T) = \sum \frac{|T_i|}{|T|} * Info(T_i). \cdot \tag{3}$$

Then the criterion for choosing an attribute is a formula

$$Gain(X) - Info(T) - Info_X(T). \quad (4)$$

Criterion (4) is calculated for all attributes. The attribute that maximizes this expression is selected. This attribute will be validated at the current vertex of the tree, and then this attribute will be further built by the tree, that is, the vertices will be checked for the value of this attribute, and further movement of the tree will occur depending on the response. Yes, and the same considerations can applied to the subsets obtained $T_1,T_2,\ldots,T_n$ and recursively continue the process of constructing the tree until the top of the class displays examples from one class. If, in the course of the algorithm, a vertex associated with an empty set (that is, no example has to this vertex) is obtained, then it is denoted as a leaf, and the value of the leaf is chosen most frequently in the immediate ancestor of the leaf. How to deal with numeric attributes? It is clear that one should choose a threshold against which all attribute values should compared. Let a numeric attribute have a finite number of values. Label them $\{v_1,v_2,\ldots,v_n\}$. Pre-sort all values. Then any value that lies between v_i and v_{i+1}, divides all the examples into two sets: those that lie to the left of this value $\{v_1,v_2,\ldots,v_i\}$, and those that are on the right $\{v_{i+1},v_{i+2},\ldots,v_n\}$. You can choose the average of the threshold as a threshold v_i and v_{i+1}. Otherwise, you can build a Work graph of the number of examples of a particular value, and then analyse the extremes and find the thresholds or gaps. The threshold value is calculated by the formula $TH_i=(v_i + v_{i+1})/2$. So, we have a decision tree and we want to use it to classify a new object. The crawl of the decision tree begins with the root of the tree. On each inner vertex, the value of the object is checked Y by the attribute corresponding to the check in the vertex taken, and, depending on the response, the corresponding branching occurs, then this arc we move to the vertex, which is placed at the level below, etc. The tree crawl ends as soon as the vertex of the solution meets, which gives the name of the object class Y. Algorithm for building decision tree C4.5 [3]:

Step 1. Determine the vertex that will be the root. To do this, you need to look at all the attributes and determine the revenue for each of them. The root of the tree will be the attribute with the maximum value of information profit.

Step 2. Remove the root of the tree from the peaks being viewed.

Step 3. If there are descendants, then for each of the possible branches of the tree to find information profit and determine the maximum of them. Otherwise, go to step 5.

Step 4. If not the last attribute, remove it from the set of attributes under consideration and go to step 3.

Step 5. Write the rule. If there are no vertices considered, then go to the lowest non-considered vertices and perform step 3, otherwise the end.

An example of the application of algorithm C4.5. It is necessary to build a decision tree for the data given in Table 1 - recruitment according to the Big Five model:

- Extraversion (outgoing/energetic vs. solitary/reserved).
- Openness to experience (inventive/curious vs. consistent/cautious).
- Agreeableness (friendly/compassionate vs. challenging/callous).
- Neuroticism (sensitive/nervous vs. resilient/confident)
- Conscientiousness (efficient/organized vs. extravagant/careless)

The attribute of decision-making is the attribute "Work".

Table 1
Given for building a decision tree by C4.5 algorithm

Specialist	Extraversion	Openness to experience	Agreeableness	Neuroticism	Conscientiousness	Work
D1	Extrovert	Heat	High	Light	Extravagant	No
D2	Extrovert	Heat	High	Strong	Extravagant	No
D3	Ambiverst	Heat	High	Light	Efficient	Yes
D4	Introvert	Moderately	High	Light	Efficient	Yes
D5	Introvert	Cold	Normal	Light	Efficient	Yes
D6	Introvert	Cold	Normal	Strong	Extravagant	No
D7	Ambiverst	Cold	Normal	Strong	Efficient	Yes
D8	Extrovert	Moderately	High	Light	Extravagant	No

D9	Extrovert	Cold	Normal	Light	Efficient	Yes
D10	Introvert	Moderately	Normal	Light	Efficient	Yes
D11	Extrovert	Moderately	Normal	Strong	Efficient	Yes
D12	Ambiverst	Moderately	High	Strong	Efficient	Yes
D13	Ambiverst	Heat	Normal	Light	Efficient	Yes
D14	Introvert	Moderately	High	Strong	Extravagant	No

We determine the data profit for each attribute to determine which one will be the top of the tree.

$$E(S) = -\frac{9}{14}log_2\left(\frac{9}{14}\right) - \frac{5}{14}log_2\left(\frac{5}{14}\right) = 0.94;$$

$$G(S, Extraversion) = 0.94 - \frac{5}{14}\left(-\frac{2}{5}log_2\left(\frac{2}{5}\right) - \frac{3}{5}log_2\left(\frac{3}{5}\right)\right) -$$
$$-\frac{4}{14}\left(-\frac{4}{4}log_2\left(\frac{4}{4}\right) - \frac{0}{4}\right) - \frac{5}{14}\left(-\frac{3}{5}log_2\left(\frac{3}{5}\right) - \frac{2}{5}log_2\left(\frac{2}{5}\right)\right) = 0.246;$$

$$G(S, Openness\ to\ experience) = 0.94 - \frac{4}{14}\left(-\frac{2}{4}log_2\left(\frac{2}{4}\right) - \frac{2}{4}log_2\left(\frac{2}{4}\right)\right) -$$
$$-\frac{5}{14}\left(-\frac{4}{5}log_2\left(\frac{4}{5}\right) - \frac{1}{5}log_2\left(\frac{1}{5}\right)\right) - \frac{4}{14}\left(-\frac{3}{4}log_2\left(\frac{3}{4}\right) - \frac{1}{4}log_2\left(\frac{1}{4}\right)\right) = 0.165;$$

$$G(S, Agreeableness) = 0.94 - \frac{7}{14}\left(-\frac{3}{7}log_2\left(\frac{3}{7}\right) - \frac{4}{7}log_2\left(\frac{4}{7}\right)\right) -$$
$$-\frac{7}{14}\left(-\frac{6}{7}log_2\left(\frac{6}{7}\right) - \frac{1}{7}log_2\left(\frac{1}{7}\right)\right) = 0.152;$$

$$G(S, Neuroticism) = 0.94 - \frac{8}{14}\left(-\frac{6}{8}log_2\left(\frac{6}{8}\right) - \frac{2}{8}log_2\left(\frac{2}{8}\right)\right) -$$
$$-\frac{6}{14}\left(-\frac{3}{6}log_2\left(\frac{3}{6}\right) - \frac{3}{6}log_2\left(\frac{3}{6}\right)\right) = 0.048.$$

Maximum profit *G*(*S*,*A*) has an attribute *A*= *Extraversion*. Therefore, the decision tree in the first step will take the form shown in Fig. 2.

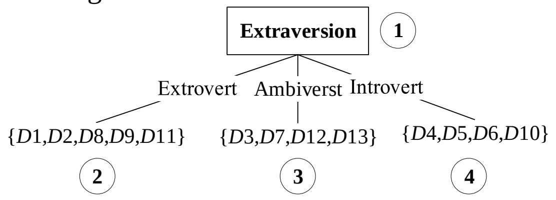

Figure 2: The first step of the C4.5 algorithm.

Consider the input given for the next step of the algorithm (Table 2).

Table 2

Given for the second step of C4.5 algorithm.

Specialist	Openness to experience	Agreeableness	Neuroticism	Work
D1	Heat	High	Light	No
D2	Heat	High	Strong	No
D8	Moderately	High	Light	No
D9	Cold	Normal	Light	Yes
D11	Moderately	Normal	Strong	Yes

The entropy in this step is equal to: $E(S) = -\frac{2}{5}log_2\left(\frac{2}{5}\right) - \frac{3}{5}log_2\left(\frac{3}{5}\right) = 0.971.$

Information revenue, respectively, for the remaining attributes:

$$G(S, Openness\ to\ experience) = 0.971 - \frac{2}{5}\left(-\frac{2}{2}log_2\left(\frac{2}{2}\right) - \frac{0}{2}\right) -$$
$$-\frac{2}{5}\left(-\frac{1}{2}log_2\left(\frac{1}{2}\right) - \frac{1}{2}log_2\left(\frac{1}{2}\right)\right) - \frac{1}{5}\left(-\frac{1}{1}log_2\left(\frac{1}{1}\right) - \frac{0}{41}\right) = 0.585;$$

$$G(S, Agreeableness) = 0.971 - \frac{3}{5}\left(-\frac{3}{3}log_2\left(\frac{3}{3}\right) - \frac{0}{3}\right) - \frac{2}{5}\left(-\frac{0}{2} - \frac{2}{2}log_2\left(\frac{2}{2}\right)\right) = 0.971;$$

$$G(S, Neuroticism) = 0.971 - \frac{3}{5}\left(-\frac{1}{3}log_2\left(\frac{1}{3}\right) - \frac{2}{3}log_2\left(\frac{2}{3}\right)\right) -$$
$$-\frac{2}{5}\left(-\frac{1}{2}log_2\left(\frac{1}{2}\right) - \frac{1}{2}log_2\left(\frac{1}{2}\right)\right) = 0.02.$$

Maximum profit $G(S,A)$ has an attribute A=*Agreeableness*. Therefore, the decision tree in the second step will look like in Fig. 3.

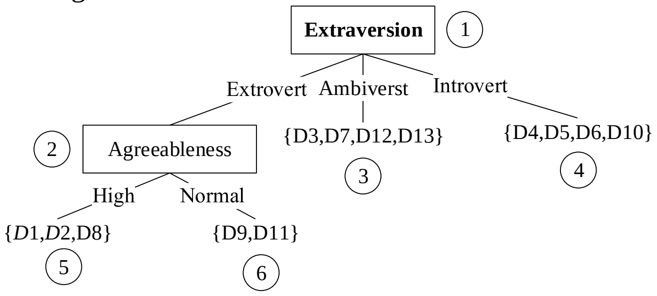

Figure 3: The decision tree in the second step of execution of algorithm C4.5.

When reviewing records by Specialist {D1,D2,D8}the result attribute acquires value {No}, and {D9,D11}- {Yes}, then there is no need to find information for the tops of these branches, because they are leaves. Therefore, the decision tree will take the form (Fig. 4):

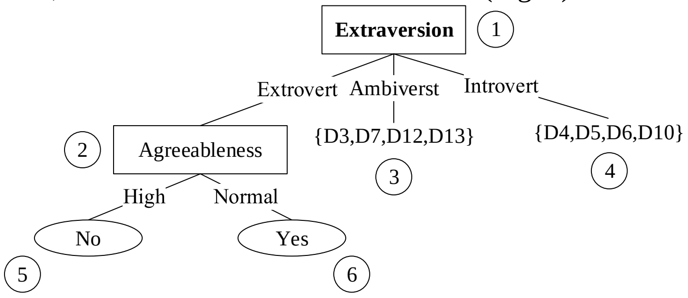

Figure 4: The decision tree in the third step of execution of algorithm C4.5.

Consider the input given for the next step (Table 3):

Table 3

Given for the fourth step of building a decision tree.

Specialist	Openness to experience	Agreeableness	Neuroticism	Work
D3	Heat	High	Light	Yes
D7	Cold	Normal	Strong	Yes
D12	Moderately	High	Strong	Yes
D13	Heat	Normal	Light	Yes

Because the decision attribute takes one value, it is equal {Yes}, then at the end of the branch we *Extraversion*= {*Ambiverst*} get a leaflet (Fig. 5).

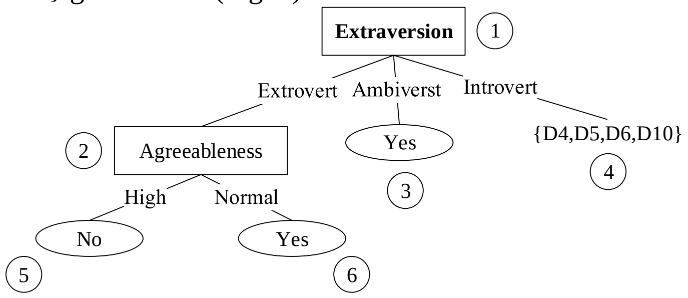

Figure 5: Decision tree in step 4 of algorithm C4.5 execution.

As a result, the branch was not considered *Extraversion* ={*Introvert*} .The input given at this step is summarized in Table 4.

Table 4

Given for the fifth step of building a decision tree.

Specialist	Openness to experience	Agreeableness	Neuroticism	Work
D4	Moderately	High	Light	Yes
D5	Cold	Normal	Light	Yes
D6	Cold	Normal	Strong	No
D10	Moderately	Normal	Light	Yes
D14	Moderately	High	Strong	No

We calculate the entropy for this step: : $E(S) = -\frac{3}{5}log_2\left(\frac{3}{5}\right) - \frac{2}{5}log_2\left(\frac{2}{5}\right) = 0.971.$

Attribute informational revenue equals:

$$G(S, Openness\ to\ experience) = 0.971 - \frac{3}{5}\left(-\frac{2}{3}log_2\left(\frac{2}{23}\right) - \frac{1}{3}log_2\left(\frac{1}{3}\right)\right) -$$

$$-\frac{2}{5}\left(-\frac{1}{2}log_2\left(\frac{1}{2}\right) - \frac{1}{2}log_2\left(\frac{1}{2}\right)\right) = 0.2;$$

$$G(S, Agreeableness) = 0.971 - \frac{2}{5}\left(-\frac{1}{2}log_2\left(\frac{1}{2}\right) - \frac{1}{2}log_2\left(\frac{1}{2}\right)\right) -$$

$$-\frac{3}{5}\left(-\frac{2}{3}log_2\left(\frac{2}{23}\right) - \frac{1}{3}log_2\left(\frac{1}{3}\right)\right) = 0.02;$$

$$G(S, Neuroticism) = 0.971 - \frac{3}{5}\left(-\frac{3}{3}log_2\left(\frac{3}{3}\right) - \frac{0}{3}\right) - \frac{2}{5}\left(-\frac{0}{2} - \frac{2}{2}log_2\left(\frac{2}{2}\right)\right) = 0.971.$$

Maximum information income is equal 0.971. Therefore, the top will be the attribute {*Neuroticism*}. Consider the decision tree after this step (Fig. 6):

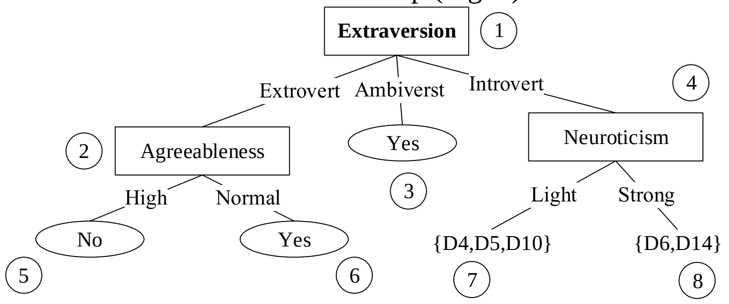

Figure 6: Solution tree for step 5 of C4.5 algorithm execution

When looking at the input, you can see that at the seventh mark, the decision attribute becomes {Yes}, and on eighth - {No}. So, you can put sheets on these labels right away. As a result, the tree will take the following form (Fig. 7).

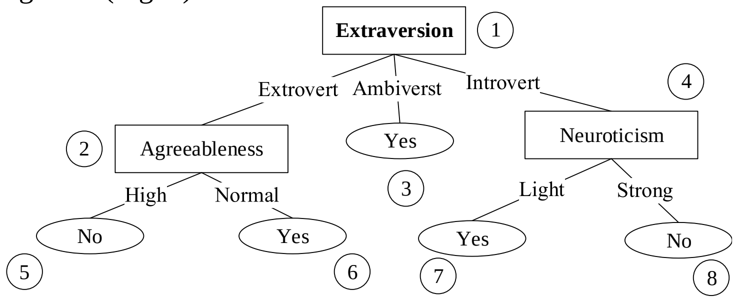

Figure 7: The resulting decision tree after executing C4.5 algorithm.

Since no branches are considered, the algorithm is finished.

6. CART algorithm for building decision tree

CART (Classification And Regression Tree) is an algorithm for building a binary decision tree, first published by Ariman et al. in 1984. The algorithm is designed to solve classification and regression problems. There are also modifications to this algorithm - IndCART and DB-CART algorithms [2].

The main differences between the CART algorithm and the ID3 family are:

- Binary representation of the decision tree;
- The availability of the breakdown quality assessment function;
- The mechanism of cutting off branches of a decision tree;
- Algorithm for processing missing values;
- Building regression trees.

In the CART algorithm, each vertex of the decision tree has two descendants. At each step of constructing a tree, a rule formulated at the top is divided into two parts by a set of examples - the part in which the rule is executed (descendant - right) and the part in which the rule is not executed (descendant - left). The breakdown estimator function is used to select the optimal rule.

Decision tree learning belongs to the classroom with the teacher, that is, the train and test samples contain a classified set of examples. The evaluation function used by the CART algorithm is based on the intuitive idea of reducing top uncertainty. Consider a problem with two classes and a vertex with 50 examples of one class. The top has maximum "uncertainty". If a breakdown is found that splits the data into two subgroups - 40: 5 examples in one and 10:45 in the other, then the intuitive "uncertainty" will decrease. It will completely disappear when a breakdown is found that will create subgroups of 50: 0 and 0:50. In the CART algorithm, the idea of "uncertainty" is formalized in an index *Gini*. If the dataset *T* contains class *n* data, then the index *Gini* is defined as $Gini(T) = 1 - \sum_{i=1}^{n} p_i^2$, where p_i – the probability (relative frequency) of the class *i* in *T* [2].

If the set *T* is divided into two parts T_1 and T_2 with the number of examples in each N_1 and N_2 accordingly, then the breakdown quality metric will be $Gini_{split} = \frac{N_1}{N} \cdot Gini(T_1) + \frac{N_2}{N} \cdot Gini(T_2)$.

The best part is the partition for which $Gini_{split}(T)$ is minimal. Lets mark *N* – number of examples in vertex-ancestor, a *L* and *R* – the number of examples in the left and right descendants, l_i and r_i – number of copies *i*-st of the 1st class in the left / right offspring. Then the quality of the partition is evaluated by the formula $Gini_{split} = \frac{L}{N} \cdot \left(1 - \sum_{i=1}^{n} \left(\frac{l_i}{L}\right)^2\right) + \frac{R}{N} \cdot \left(1 - \sum_{i=1}^{n} \left(\frac{r_i}{R}\right)^2\right) \to \min$.

For reduce the amount of calculations, the last formula can be converted to

$$Gini_{split} = \frac{L}{N} \cdot \left(L \cdot \left(1 - \frac{1}{L^2} \cdot \sum_{i=1}^{n} l_i^2\right) + R \cdot \left(1 - \frac{1}{R^2} \cdot \sum_{i=1}^{n} r_i^2\right)\right) \to \min .$$

Since multiplication by a constant does not affect the process of minimization, then:

$$Gini_{split} = L - \frac{1}{L} \cdot \sum_{i=1}^{n} l_i^2 + R - \frac{1}{R} \cdot \sum_{i=1}^{n} r_i^2 \to \min ,$$

$$Gini_{split} = N - \left(\frac{1}{L} \cdot \sum_{i=1}^{n} l_i^2 + \frac{1}{R} \cdot \sum_{i=1}^{n} r_i^2\right) \to \min , \quad G_{split} = \frac{1}{L} \cdot \sum_{i=1}^{n} l_i^2 + \frac{1}{R} \cdot \sum_{i=1}^{n} r_i^2 \to \max .$$

As a result: the best partition will be the value for which the value is G_{split} is the maximum. Sometimes CART uses other partitioning criteria - Twoing, Symmetric Gini, and others.

The vector of predicate variables fed to the tree input can contain both numeric (ordinal), Yes and categorical variables. In any case, only one variable is included in each vertex of the partition. If the variable is a numeric type, then a vertex is formed by the appearance rule $x_i \le c$ where *c* – some threshold, which is most often chosen as the arithmetic mean of two adjacent, ordered variable values x_i train sample. If the variable is of categorical type, then a rule is formed at the top $x_i \in V(x_i)$, where $V(x_i)$–some void subset of the set of variable values x_i in the train sample. So, for *n* the algorithm compares the values of the numeric attribute *n*-1 and for the categorical – (2^{n-1}-1). At each step of building a tree, the algorithm consistently compares all possible partitions for all attributes and selects the best attribute and the best partition for it [8]. All possible partitions for categorical attributes are conveniently represented by analogy with a binary representation of a number. If the attribute has *n* outside values, 2^n -partitioning. The first (where all zeros) and the last (all units) do not interest us, we get 2^n - 2. Since the order of the sets is also not important here, we obtain (2^n - 2)/2 or (2^{n-1}-1) the first (single) binary images. If {*A*,*B*,*C*,*D*,*E*} – all possible values of some attribute *X*, then for the current partition that has an image, let's say {0,0,1,0,1}, will get a rule *X in* {*C*,*E*} for the right branch and [not {0,0,1,0,1}={1,1,0,1,0}= *X in* { *A*,*B*,*C* }] for the left branch. Tree felling mechanism, original name – “minimal cost-complexity tree pruning” - the biggest difference between the CART algorithm and other tree-building algorithms. CART views clipping as a trade-off between two problems: getting an optimal size tree and getting an accurate estimate of the probability of misclassification.

The main problem with clipping is the large number of possible subtrees of one tree. More precisely, if a binary tree has $|T|$ of leaves then exists $\approx[1.5028369|T|]$ cut subtrees. If a tree has at least 1000 leaves, then the number of cut subtrees becomes extremely large. The basic idea of the method is not to consider all possible subtrees, but to limit yourself only to the "best representatives" according to the following assessment. Denote by $|T|$ number of tree leaves, $R(T)$ – an error in the classification of a tree as the ratio of the number of incorrectly classified examples to the number of examples in the train sample. Let's define $C(T)$ – total cost (estimate / cost / complexity) of the tree T as $C_\alpha(T)=R(T)+\alpha|T|$, where $|T|$ – number of leaves (terminal vertices) of the tree, $|T|$ – a certain parameter that varies from 0 to $+\infty$. The total cost of a tree consists of two components - a tree classification error and a penalty for its complexity. If the classification error of the tree is constant, then with increase α the total value of the tree will increase. Then, depending on α, a smaller branched tree that gives a large classification error may cost less than that which gives a smaller error but is more branched [2]. Let's define T_{max} – the largest tree to be cut. If you lock the value α, then there is the smallest sub-tree that is minimized, which satisfies Yes and the conditions:

$$C_\alpha(T(\alpha)) = \min_{T<=T_{\max}} C_\alpha(T) \text{; and } if\ C_\alpha(T) = C_\alpha(T(\alpha))\ then T(\alpha) <= T \cdot$$

The first condition states that the tree does not have a Yes tree subtree T_{max}, which will cost less than a $T(\alpha)$ for this meaning α. The second condition states that if there is more than one subtree with a given full value, then the smallest tree should selected.

You can show that for any value α there is the smallest sub-tree to be minimized. This is not a trivial task. Even though α has an infinite number of values, there is an infinite number of tree subtrees T_{max}. You can build a tree subtree sequence T_{max} descending number of vertices: $T_1>T_2>T_3>\ldots>\{t_1\}$. Here t_1 – the root top of the tree, T_k – the smallest sub-tree to be minimized $\alpha \in [\alpha_k, \alpha_{k+1})$. This means that you can get a Yes tree in sequence by applying a pruning procedure to the current tree, which allows you to develop an efficient algorithm for finding the smallest subtree that is minimized for different values α. The first tree in this sequence is the smallest tree subtree T_{max}, which has Yes in the same classification error as T_{max}, than $T_1=T(\alpha=0)$. If the partition goes as long as there is only one class in each vertex, then $T_1= T_{max}$. Because early stopping methods are often used (prepruning),there may be a subtree of the tree T_{max} with Yes the same classification error.

Here is a calculation algorithm T_1 with T_{max}. To do this, find an arbitrary pair of shared ancestor sheets that can combined: $R(t)= R(l)+R(r)$, where r i l - vortex leaves t. Continue until pairs remain. Yes will receive a tree with Yes the same cost as T_{max} for α=0, but less branched than a T_{max}.

To get the next tree in sequence and the corresponding value α will be pointed as T_t – tree branch T with the root apex t. Now let's determine for what values the tree is T-T_t will be better than T. If you cut it into the top t,then its contribution to the full value of the tree T-T_t will add up $C_\alpha(\{t\})=R(t)+\alpha$, where $R(t)= r(t)\ p(t)$, where $r(t)$ is vertex classification error t, and $p(t)$ is the proportion of cases that "passed" over the top t. An alternative is the case $R(t)=m/n$, where m - the number of examples classified incorrectly, a n – the total number of examples of the entire tree being classified. Contribution T_t the full value of the tree T will be $C_\alpha(T_t)=R(T_t)+\alpha|T_t|$, where $R(T_t) = \sum_{t' \in T_t} R(t')$. Tree T-T_t is better than T, if $C_\alpha(\{t\})= C_\alpha(T_t)$. This is true because for this value they have the same value, but T-T_t is the smallest of them. When $C_\alpha(\{t\})= C_\alpha(T_t)$, we will get $R(T_t) + \alpha|T_t| = R(t) + \alpha$, or $\alpha = \frac{R(t) - R(T_t)}{|T_t| - 1}$. The basic idea behind the cropping method is to calculate the value $g(t) = \frac{R(t) - R(T_{1,t})}{|T_{1,t}| - 1}$ for each tree top T_1 and the choice of "weak links" that can be more than one. These are the peaks to which they relate $g(t) = \frac{R(t) - R(T_{1,t})}{|T_{1,t}| - 1}$ is the smallest. Cut off T_1 in these peaks to get T_2 – the next tree in the sequence. We then continue this process for the resulting tree and until the root vertex is obtained - a tree with only one vertex [8].

The CART algorithm for building a decision tree:

Step 1. Select from the attributes of the first and set it as current.

Step 2. Determine all possible combinations of values for the current attribute.

Step 3. Define the criterion G_{split} for the combinations defined in step 2.

Step 4. If the attribute is not the last of the many attributes by which the rule is built, then setting the next attribute as the current one and going to step 2, otherwise going to step 5.

Step 5. Determine the maximum criterion G_{split} for all combinations of attribute values found in step 3.

Step 6. Establishing the top of the tree, which is the name of the attribute that has the maximum criterion G_{split}. The corresponding combination of values, taking into account the part of the tree that was built earlier, to the position corresponding to the condition of the previous vertex, otherwise, to the position that negates the condition of the previous vertex, if the created vertex is not the root of the tree.

Step 7. If the decision attribute on the current vertex assumes one value taking into account the previous vertices, then take the current vertex as a leaf.

Step 8. If the vertex is not a leaf, then mark its descendants as Yes and not considered, otherwise assume that no descendants exist (i.e. the vertex is a leaf).

Step 9. Destruction of all criteria G_{split}, previously discussed.

Step 10. If there are no descendants considered, then go to the leftmost vertex that has them, otherwise END (the decision tree is built).

Step 11. Selection of inputs that comply with the rules that can generated when traversing a tree to the current vertex. Go to Step 1.

An example of building a decision tree using the CART algorithm. Consider an example of the algorithm described above. It is necessary to build a decision tree for the data given in Table 1 about the competition for tennis, depending on the Extraversion. The attribute of decision-making is the attribute "Work". We define the parameter G_{split} for each attribute and all possible splits by its values. For example, there are three possible breakdowns for the *Extraversion* attribute: {*Introvert*} and {*Extrovert*, *Ambiverst*}; {*Extrovert*} and {*Ambiverst*, *Introvert*}; {*Ambiverst*} and {*Extrovert*, *Introvert*}. For each variant of partitioning, we define an parameter $G_{split} = \frac{1}{L}\sum_{i=1}^{n} l_i^2 + \frac{1}{R}\sum_{i=1}^{n} r_i^2 \to \max$.

$$Extraversion = \{Extrovert\}: G_{split} = \frac{1}{5}(3^2 + 2^2) + \frac{1}{9}(7^2 + 2^2) = 8.49.$$

$$Extraversion = \{Ambiverst\}: G_{split} = \frac{1}{4}(4^2 + 0^2) + \frac{1}{10}(5^2 + 5^2) = 9.$$

$$Extraversion = \{Introvert\}: G_{split} = \frac{1}{5}(3^2 + 2^2) + \frac{1}{9}(3^2 + 6^2) = 12.2.$$

The *Openness to experience* attribute also has three options for partitioning: {*Heat*} and {*Cold*, *Moderately*}; {*Cold*} and {*Moderately*, *Heat*}; {*Moderately*} and {*Cold*, *Heat*}. Consider defining their metrics G_{split}.

$$Openness\ to\ experience = \{Heat\}: G_{split} = \frac{1}{4}(2^2 + 2^2) + \frac{1}{10}(7^2 + 3^2) = 7.8.$$

$$Openness\ to\ experience = \{Moderately\}: G_{split} = \frac{1}{6}(4^2 + 2^2) + \frac{1}{8}(3^2 + 5^2) = 7.58.$$

$$Openness\ to\ experience = \{Cold\}: G_{split} = \frac{1}{4}(3^2 + 1^2) + \frac{1}{10}(4^2 + 6^2) = 7.7.$$

Accordingly, according to principle, we break the attribute *Agreeableness*: {*High*} and {*Normal*}; {*Normal*} and {*High*}. Because when definitely G_{split} the right and left parts of the partition are added, it is sufficient to determine G_{split} for only one variant of partitioning.

$$Agreeableness = \{High\}: G_{split} = \frac{1}{7}(4^2 + 3^2) + \frac{1}{7}(6^2 + 1^2) = 8.8.$$

The *Neuroticism* attribute can broken into {*Strong*} and {*Light*}; {*Light*} and {*Strong*}.

$$Neuroticism\ = \{Strong\}: G_{split} = \frac{1}{8}(2^2 + 6^2) + \frac{1}{6}(3^2 + 3^2) = 8.$$

Consider all the values of the criterion G_{split} and determine the maximum of them. The maximum criterion for evaluating the quality of a partition is 12.2, provided that *Extraversion* = {*Introvert*}.This means that the attribute "*Extraversion*" will be apex provided that it is "*Introvert*". Since this vertex is the first, it will be root. Therefore, in this step, the tree is obtained (Fig. 8).

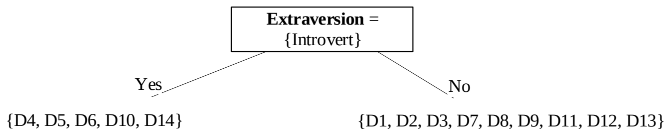
Extraversion = {Introvert}
Yes
No
{D4, D5, D6, D10, D14}
{D1, D2, D3, D7, D8, D9, D11, D12, D13}

Figure 8: The decision tree after the first stage of the CART algorithm.

We do the calculations to find the next vertex with the precondition in mind. Definition of criterion G_{split} bypassing the tree from left to right. Let us consider this process in stages. We define possible splits for the "Openness to *experience*" attribute provided that *Extraversion* = {*Introvert*}. The table with the data at this stage is look (table 5):

Table 5

Given for the second step of building a decision tree.

Specialist	Extraversion	Openness to experience	Agreeableness	Neuroticism	Work
D4	Introvert	Moderately	High	Light	Yes
D5	Introvert	Cold	Normal	Light	Yes
D6	Introvert	Cold	Normal	Strong	No
D10	Introvert	Moderately	Normal	Light	Yes
D14	Introvert	Moderately	High	Strong	No

We define the criterion G_{split} for all the splits of each attribute.

$$Extraversion = \{Introvert\}: G_{split} = \frac{1}{5}(3^2 + 2^2) + 0 = 2.6.$$

$$Openness\ to\ experience = \{Moderately\}: G_{split} = \frac{1}{3}(2^2 + 1^2) + \frac{1}{2}(1^2 + 1^2) = 3.67.$$

$$Agreeableness = \{High\}: G_{split} = \frac{1}{2}(1^2 + 1^2) + \frac{1}{3}(2^2 + 1^2) = 3.67.$$

$$Neuroticism = \{Strong\}: G_{split} = \frac{1}{2}(0^2 + 2^2) + \frac{1}{3}(3^2 + 0^2) = 5.$$

The maximum criterion is G_{split},=5 resulting in the top of the tree being the attribute "Neuroticism" with the value "*Strong*". Since in the criterion of the quality of the partition l_1^2=0 and r_2^2=0, then you can define two sheets at once. As a result, the decision tree will take the form (Fig. 9).

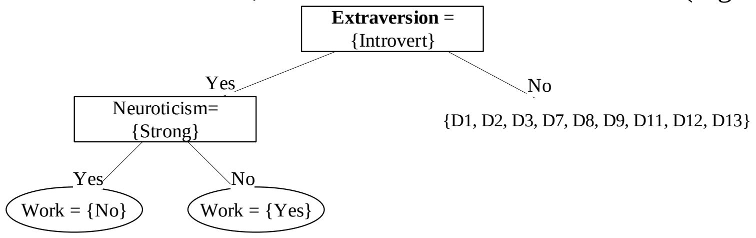
Extraversion = {Introvert}
Yes
No
Neuroticism= {Strong}
{D1, D2, D3, D7, D8, D9, D11, D12, D13}
Yes
No
Work = {No}
Work = {Yes}

Figure 9: Decision tree in the 2nd stage of execution of the CART algorithm

Consider the far right not considered edge. We repeat the calculation of the criterion G_{split} subject to the condition at all parent vertices. In this step, the condition is *Extraversion* ≠ {*Introvert*}. Consider the input table for the third stage of the algorithm for constructing the decision tree (table 6):

Table 6

Given for the 3rd stage of execution of algorithm of construction of the decision tree.

Specialist	Extraversion	Openness to experience	Agreeableness	Neuroticism	Work
D1	Extrovert	Heat	High	Light	No
D2	Extrovert	Heat	High	Strong	No
D3	Ambiverst	Heat	High	Light	Yes
D7	Ambiverst	Cold	Normal	Strong	Yes
D8	Extrovert	Moderately	High	Light	No
D9	Extrovert	Cold	Normal	Light	Yes
D11	Extrovert	Moderately	Normal	Strong	Yes
D12	Ambiverst	Moderately	High	Strong	Yes
D13	Ambiverst	Heat	Normal	Light	Yes

$$Extraversion = \{Extrovert\}: G_{split} = \frac{1}{5}(3^2 + 2^2) + \frac{1}{4}(4^2 + 0^2) = 6.6.$$

$$Openness\ to\ experience = \{Heat\}: G_{split} = \frac{1}{4}(2^2 + 2^2) + \frac{1}{5}(4^2 + 1^2) = 5.4.$$

$$Openness\ to\ experience = \{Moderately\}: G_{split} = \frac{1}{3}(2^2 + 1^2) + \frac{1}{6}(4^2 + 2^2) = 5.$$

$$Openness\ to\ experience = \{Cold\}: G_{split} = \frac{1}{2}(2^2 + 0^2) + \frac{1}{7}(4^2 + 3^2) = 5.571.$$

$$Agreeableness = \{High\}: G_{split} = \frac{1}{5}(2^2 + 3^2) + \frac{1}{4}(4^2 + 0^2) = 6.6.$$

$$Neuroticism\ = \{Strong\}: G_{split} = \frac{1}{6}(3^2 + 3^2) + \frac{1}{3}(3^2 + 0^2) = 6.$$

This step is the maximum criterion G_{split} =6.6on two combinations of attribute values. We choose any of them, for example, *Extraversion* = {*Extrovert*}, and set it as the top of the tree. Finally, the tree will take the form shown in Fig. 10. Because r_2^2=0, then you can immediately identify the piece.

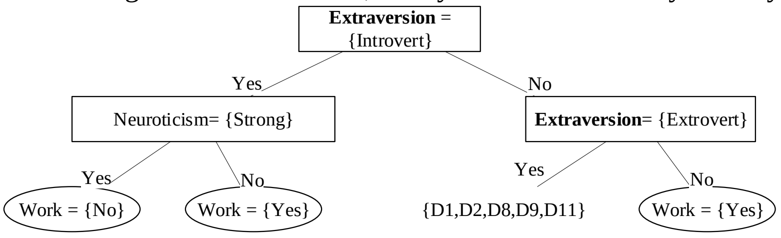

Figure 10: Decision tree in the 3rd stage of execution of CART algorithm.

Table 7

Given for the fourth stage of the algorithm implementation of the decision tree.

Specialist	Extraversion	Openness to experience	Agreeableness	Neuroticism	Work
D1	Extrovert	Heat	High	Light	No
D2	Extrovert	Heat	High	Strong	No
D8	Extrovert	Moderately	High	Light	No
D9	Extrovert	Cold	Normal	Light	Yes
D11	Extrovert	Moderately	Normal	Strong	Yes

We define a criterion G_{split} for all possible values of each attribute:

$$Extraversion = \{Extrovert\}: G_{split} = \frac{1}{5}(3^2 + 2^2) + 0 = 2.6.$$

$$Openness\ to\ experience = \{Heat\}: G_{split} = \frac{1}{2}(0^2 + 2^2) + \frac{1}{3}(2^2 + 1^2) = 3.667.$$

$$Openness\ to\ experience = \{Moderately\}: G_{split} = \frac{1}{2}(1^2 + 1^2) + \frac{1}{3}(1^2 + 2^2) = 2.667.$$

$$Openness\ to\ experience = \{Cold\}: G_{split} = \frac{1}{1}(1^2 + 0^2) + \frac{1}{7}(1^2 + 3^2) = 3.5.$$

$$Agreeableness = \{High\}: G_{split} = \frac{1}{3}(0^2 + 3^2) + \frac{1}{2}(2^2 + 0^2) = 5.$$

$$Neuroticism\ = \{Strong\}: G_{split} = \frac{1}{3}(1^2 + 2^2) + \frac{1}{2}(1^2 + 1^2) = 2.667.$$

The maximum quality criterion, G_{split},=5 therefore l_1^2=0 and r_2^2=0, the top of the tree at this stage will become, *Agreeableness* ={*High*}, and immediately determine its leaves. The decision tree will take the form shown in Fig. 11. Now there are no edges not considered in the tree, so the algorithm of constructing the decision tree is completed.

An example of pruning a CART decision tree. If there is a need to trim the tree, then this is done in the following sequence. Take the tree shown in Fig. 11 for an example. On this basis, we will construct a new tree according to the following rules: in rectangles we give two values: on the left - the number of positive values of the decision attribute, and on the right - negative, if the vertex being considered is a leaf, then we present it in the form of an ellipse. Such a tree is shown in Fig. 12.

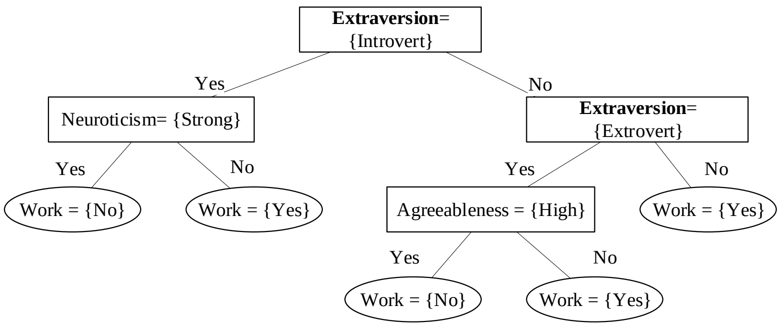

Figure 11: The decision tree resulting from the CART algorithm

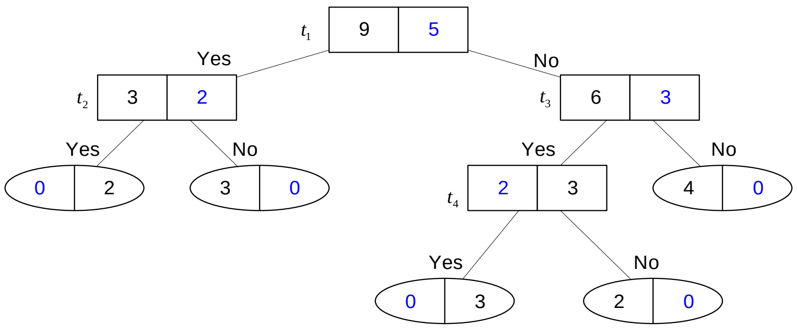

Figure 12: The decision tree prepared for cutting.

For determine which branch can be trimmed, we need to calculate for each vertex $g(t)=\dfrac{R(t)-R(T_{1,t})}{|T_{1,t}|-1}$ then make a vertex with a minimum value of this parameter a leaf of a tree, and respectively the left and right parts of the rectangle - the left and right part of the ellipse. Here $R(t)$ – classification error. In this case, we select the numbers in the vertices with smaller values and divide them by the total number of examples, $R(T_{1,t})$ – the sum of the errors of all the subtree leaves. This parameter is defined as the sum (by all sheets) of the ratio of incorrectly classified examples of the corresponding subtrees to the total number of examples. In our case, the total is 14. Because for subtrees with root at vertex t_1 the leaves do not have misclassified examples then $R(T_{1,t})=0$. Determine for each vertex the value of the parameter $g(t_i),\ i=\overline{1..n}$. Since the minimum value $g(t_i)$ falls on $i=1$, we cut to the top t_1. In result we get decision tree (Fig. 13).

$$g(t_1)=\left(\frac{5}{14}-0\right)\Big/(5-1)=\frac{5}{56};\ g(t_2)=\left(\frac{2}{14}-0\right)\Big/(2-1)=\frac{2}{14};\ g(t_3)=\left(\frac{3}{14}-0\right)\Big/(3-1)=\frac{3}{28};\ g(t_4)=\left(\frac{2}{14}-0\right)\Big/(2-1)=\frac{2}{14}.$$

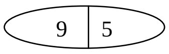

Figure 13: Decision tree after pruning

In this case, by our calculations, it turned out that it is most effective to prune a tree under the very root apex. This is because the sample is not voluminous, and the values of the decision attribute are very unevenly distributed, meaning that the Work is more likely to occur than vice versa.

7. Conclusions and further direction of research

As a result of the study, two sets of rules were built to classify the results of psychophysiological testing to determine the reliability of the employee by his psychological and individual characteristics. The results obtained make it possible to automate the decision-making process regarding the suitability of specialists to work with energy grid operators, as well as to identify the signs that most influence the decision. It is planned to evaluate the generated rules in the future, i.e. to determine their reliability and support indicators. This will allow you to choose a more efficient algorithm. Since a sufficiently small amount of input data receives a large number of rules, i.e. a large number of decision tree leaves, it is necessary to investigate which rules can removed with the smallest classification error, i.e. to prune trees. When comparing the results of both algorithms, it is determined that although the number of rules built using the CART method is smaller, the rules themselves are much longer, that is, contain more nested conditions than the C4.5 method. This means that the C4.5 tree has more leaves, that is, it is wider, and the CART tree has more vertices, that is, it is deeper.

8. References

[1] J. Soni, U. Ansari, D. Sharma, S. Soni, Predictive data mining for medical diagnosis: An overview of heart disease prediction, volume of 17(8) International Journal of Computer Applications, 2011, pp. 43-48.

[2] M. Seera, C.P. Lim, S.C. Tan, C.K. Loo, A hybrid FAM–CART model and its application to medical data classification, volume of 26(8) Neural Computing and Applications, 2015, pp. 1799-1811.

[3] R.J. Lewis, An introduction to classification and regression tree (CART) analysis, volume of 14 Annual meeting of the society for academic emergency medicine in San Francisco, California, 2000.

[4] D. Lavanya, K.U. Rani, Ensemble decision tree classifier for breast cancer data, volume of 2(1) International Journal of Information Technology Convergence and Services, 2012, p. 17.

[5] J. Quinlan, C4.5: Programs for Machine learning, Morgan Kaufmann Publishers, 1993.

[6] G. Sujatha, K.U. Rani, Evaluation of decision tree classifiers on tumor datasets, International Journal of Emerging Trends & Technology in Computer Science, 2013, pp. 418-423.

[7] H.P. Newquist, Data Mining: The AI Metamorphosis, volume of 9 Database Programming and Design, 1996.

[8] D. Michie, D.J. Spiegelhalter, C.C. Taylor, Machine Learning, Neural and Statistical Classification, 1994.

[9] J. O. Kang, S. H., Chung Y. M. Suh, Prediction of hospital charges for the cancer patients with data mining techniques, volume of 15(1) Journal of Korean Society of Medical Informatics, 2009, pp. 13-23.

[10] A. Hussein, H. Gheni, W. Oleiwi, Z. Hasan, Prediction of credit card payment next month through tree net data mining techniques, volume of 19(1) International Journal of Computing, 2020, pp. 97-105.

[11] O. Veres, P. Ilchuk, O. Kots, Application of Data Mining to Exchange Rate Influence Identification, CEUR workshop proceedings, Vol-2604, 2020, pp. 1117-1126.

[12] H. Lipyanina, A. Sachenko, T. Lendyuk, S. Nadvynychny, S. Grodskyi, Decision tree based targeting model of customer interaction with business page, CEUR Workshop, Proceedings, 2020, pp. 1001-1012.

[13] L. Chyrun, V. Vysotska, I. Kis, L. Chyrun, Content Analysis Method for Cut Formation of Human Psychological State, International Conference on Data Stream Mining and Processing, 2018, pp. 139-144.

[14] V. Lytvyn, V. Vysotska, A. Rzheuskyi, Technology for the Psychological Portraits Formation of Social Networks Users for the IT Specialists Recruitment Based on Big Five, NLP and Big Data Analysis, CEUR Workshop Proceedings, Vol-2392, 2019, pp. 147-171.

[15] L. Chyrun, I. Kis, V. Vysotska, L. Chyrun, Content monitoring method for cut formation of person psychological state in social scoring, International Conference on Computer Sciences and Information Technologies Proceedings, 2018, pp. 106-112.

[16] N. Shakhovska, S. Fedushko, M. Greguš, N. Melnykova, I. Shvorob, Yu. Syerov, Big Data analysis in development of personalized medical system, Emerging Ubiquitous Systems and Pervasive Networks, 2019, pp. 229-234.

[17] A. Rzheuskyi, O. Kutyuk, O. Voloshyn, A. Kowalska-Styczen, V. Voloshyn, L. Chyrun, S. Chyrun, D. Peleshko, T. Rak, The Intellectual System Development of Distant Competencies Analyzing for IT Recruitment, volume of 1080 Advances in Intelligent Systems and Computing IV, Springer, Cham, 2020, pp. 696-720.

[18] V. Pasichnyk, T. Shestakevych, The model of data analysis of the psychophysiological survey results, volume of 512 Advances in Intelligent Systems and Computing, 2017, pp. 271-281.